FUJITSU

FUJITSU Software
NetCOBOL for .NET V8.0

ユーザーズガイド

Windows

B1WD-3497-01Z0(00)
2018年4月

NetCOBOL for .NET ユーザーズガイド

NetCOBOL for .NET は、 .NET プラットフォーム向け COBOL アプリケーションを作成するための 高度な COBOL コンパイラと関連ツールセットを提供します。NetCOBOL for .NET は、MSIL（Microsoft Intermediate Language）コードを生成することができ、 アプリケーションが.NET Framework を十分活用できるようにするための言語拡張を用意しています。

NetCOBOL for .NET は、 Visual Studio 開発環境内での.NET Framework 向け COBOL アプリケーションの開発をサポートしています。

NetCOBOL for .NET オンラインマニュアルの内容

略称について

ユーザーズガイドでは、各製品を次のように略記しています。あらかじめご了承ください。

正式名称	略称
Microsoft(R) Windows Server(R) 2016 Datacenter	Windows Server 2016
Microsoft(R) Windows Server(R) 2016 Standard	
Microsoft(R) Windows Server(R) 2016 Essentials	
Microsoft(R) Windows Server(R) 2012 R2 Datacenter	Windows Server 2012 R2
Microsoft(R) Windows Server(R) 2012 R2 Standard	
Microsoft(R) Windows Server(R) 2012 R2 Essentials	
Microsoft(R) Windows Server(R) 2012 R2 Foundation	
Microsoft(R) Windows Server(R) 2012 Datacenter	Windows Server 2012
Microsoft(R) Windows Server(R) 2012 Standard	
Microsoft(R) Windows Server(R) 2012 Essentials	
Microsoft(R) Windows Server(R) 2012 Foundation	
Microsoft(R) Windows Server(R) 2008 R2 Datacenter	Windows Server 2008 R2
Microsoft(R) Windows Server(R) 2008 R2 Standard	
Microsoft(R) Windows Server(R) 2008 R2 Enterprise	
Microsoft(R) Windows Server(R) 2008 R2 Foundation	
Microsoft(R) Windows Server(R) 2008 Standard	Windows Server 2008
Microsoft(R) Windows Server(R) 2008 Standard without Hyper-V(TM)	
Microsoft(R) Windows Server(R) 2008 Enterprise	
Microsoft(R) Windows Server(R) 2008 Enterprise without Hyper-V(TM)	
Microsoft(R) Windows Server(R) 2008 Foundation	
Microsoft(R) Windows Server(R) 2008 Datacenter	
Microsoft(R) Windows Server(R) 2008 Datacenter without Hyper-V(TM)	
Windows(R) 10 Education	Windows 10

Windows(R) 10 Home	
Windows(R) 10 Pro	
Windows(R) 10 Enterprise	
Windows(R) 8.1	Windows 8.1
Windows(R) 8.1 Pro	
Windows(R) 8.1 Enterprise	
Windows(R) 7 Home Premium	Windows 7
Windows(R) 7 Professional	
Windows(R) 7 Enterprise	
Windows(R) 7 Ultimate	
Windows Vista(R) Home Basic	Windows Vista
Windows Vista(R) Home Premium	
Windows Vista(R) Business	
Windows Vista(R) Enterprise	
Windows Vista(R) Ultimate	
Oracle Solaris	Solaris
Microsoft(R) Visual Studio(R)	Visual Studio
Microsoft(R) .NET Framework 4	.NET Framework 4.x
Microsoft(R) .NET Framework 4.5	
Microsoft(R) .NET Framework 4.5.1	
Microsoft(R) .NET Framework 4.5.2	
Microsoft(R) .NET Framework 4.6	
Microsoft(R) .NET Framework 4.6.1	
Microsoft(R) .NET Framework 4.6.2	
Microsoft(R) .NET Framework 4.7	
Microsoft(R) .NET Framework 4.7.1	
Microsoft(R) MSDN Library for Visual Studio	MSDN Library

Microsoft(R) Visual C++(R) Development system	Visual C++
Microsoft(R) SQL Server(R)	SQL Server
Microsoft(R) Windows Azure(R) SQL Database	Azure SQL データベース
Microsoft(R) Internet Information Server	IIS
Microsoft(R) Internet Information Servers	
Microsoft(R) SQL Server(R) R2 Management Studio Express	SQL Server Management Studio
PowerSORT Server	PowerSORT
Windows 版 NetCOBOL Base Edition 開発パッケージ Windows 版 NetCOBOL Standard Edition 開発パッケージ Windows 版 NetCOBOL Professional Edition 開発パッケージ Windows 版 NetCOBOL Enterprise Edition 開発パッケージ	Windows 版 NetCOBOL
NetCOBOL Base Edition 開発パッケージ for .NET V8.0.0 NetCOBOL Standard Edition 開発パッケージ for .NET V8.0.0 NetCOBOL Enterprise Edition 開発パッケージ for .NET V8.0.0	NetCOBOL for .NET

- 「Windows Server 2016」、「Windows Server 2012 R2」、「Windows 10」、「Windows 8.1」、および「Windows 7」をすべて指す場合は、「Windows」と表記します。
- 「Developer Command Prompt for VS 2017」を指す場合、「Visual Studio コマンドプロンプト」と表記します。

商標について

画面写真および操作手順の表記について

- Microsoft Corporation のガイドラインに従って画面写真を使用しています。
- Visual Studio 2017 Professional Edition バージョン 15.5 を使用して説明しています。 Visual Studio では継続的な更新が行われおり、ご使用の Visual Studio のバージョンによっては、画面や操作手順などが異なる場合があります。 詳細については、以下の NetCOBOL 製品の技術情報ページにて随時お知らせしています。 http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」

輸出管理について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

NetCOBOL シリーズについて

NetCOBOL シリーズの最新情報については、富士通のサイトをご覧ください。

http://www.fujitsu.com/jp/software/cobol/

2018 年 4 月

目次

NetCOBOL for .NET の概要

ここでは、NetCOBOL for .NET を使い始める前に役立つ 基本情報について説明します。

このセクションの内容

NetCOBOL for .NET マニュアルマップ

NetCOBOL for .NET で参照できるドキュメントについて説明します。

NetCOBOL for .NET の特徴

NetCOBOL for .NET の特徴について簡単に説明します。

NetCOBOL for .NET の新機能

Windows 版 NetCOBOL と比較して、 NetCOBOL for .NET で新しく追加された機能について、 簡単に説明します。

NetCOBOL for .NET の追加機能

NetCOBOL for .NET のそれぞれのバージョンで追加された機能について 説明します。

NCOBU00003

NetCOBOL for .NET マニュアルマップ

NetCOBOL for .NET を用いると、 .NET プラットフォームで実行するアプリケーションを作成することができます。 NetCOBOL for .NET は開発環境として、 Visual Studio を使用することもできます。 .NET プラットフォームを十分に活用するには、 NetCOBOL for .NET のマニュアルに加えて、 Visual Studio のドキュメントも参照する必要があります。

主要なドキュメント

主要なドキュメントは以下のとおりです。

Visual Studio のドキュメント	特に、以下の 2 つが重要です。
Visual Studio の統合開発環境	Visual Studio の開発環境を理解するために必要です。
.NET Framework	.NET Framework で利用できる概念と プログラミング機能を理解するために必要です。
NetCOBOL for .NET ユーザーズガイド	現在、表示しているヘルプトピック集です。 NetCOBOL for .NET の利用方法を、 開発ツール、環境、プログラミングの概念、および技術といった観点から 説明しています。 関連した Visual Studio のヘルプトピックへの リンクも提供しています。
COBOL 文法書	従来の COBOL 構文と NetCOBOL for .NET でサポートされた COBOL 構文を 定義しているマニュアルです。 このマニュアルの書式は経験豊富な COBOL プログラマにはおなじみのものでしょう。 [スタート] - アプリ一覧(*) - お使いの NetCOBOL for .NET 製品名 - [オンラインマニュアル]から アクセスすることができる PDF マニュアルです。 * : ご使用の Windows によって、以下の操作をすることで同様の画面になります。 ・ Windows 7 : [スタート]メニュー - [すべてのプログラム] ・ Windows 8.1 および Windows Server 2012 R2 : [スタート]画面 - [↓] - [アプリ]
NetCOBOL for .NET メッセージ集	cobolc コマンドメッセージ、翻訳時メッセージ、実行時メッセージ、実行時例外および システムエラーコードを参照するためのマニュアルです。
NetCOBOL for .NET COBOL ファイルアクセスルーチン ユーザーズガイド	COBOL ファイルアクセスルーチンは、COBOL ファイルを操作するための C#言語用の API(Application Program Interface)関数群です。関数の使い方を説明しています。
NetCOBOL for .NET CBL サブルーチン ユーザーズガイド	NetCOBOL CBL サブルーチンは、Micro Focus COBOL の CBL サブルーチンとの互換機能を提供します。NetCOBOL CBL サブルーチンの機能と仕様を説明しています。
NetCOBOL for .NET AcuCOBOL サブルーチン ユーザーズガイド	NetCOBOL AcuCOBOL サブルーチンは、AcuCOBOL のサブルーチンとの互換機能を提供します。NetCOBOL AcuCOBOL サブルーチンの機能と仕様を説明しています。

NetCOBOL for .NET LE サブルーチン ユーザーズガイド	NetCOBOL LE サブルーチンは、IBM の言語環境(Language Environment : LE)サービスとの互換機能を提供します。 NetCOBOL LE サブルーチンの機能と仕様を説明しています。

NetCOBOL for .NET を使い始める際に役立つリンク

以下のリンクは、 開発作業で必要となる情報を探すときの参考にしてください。

- どのように NetCOBOL for .NET アプリケーションを 開発するかを決定する
  - NetCOBOL for .NET アプリケーションの開発
  - NetCOBOL for .NET チュートリアル
- Visual Studio での開発に関する情報
  - Visual Studio IDE を使った開発
  - プロジェクトの作成と管理
  - Visual Studio エディターを使った編集
  - NetCOBOL for .NET プロジェクトのビルド
  - Visual Studio のドキュメントの「Visual Studio でのアプリケーション開発」
- コマンドラインでの開発に関する情報
  - コマンドラインからの開発
  - コマンドラインからのビルド
  - アプリケーションの実行
- .NET プラットフォーム向け COBOL プログラムの書き方を理解する
  - .NET プラットフォームでの COBOL プログラミング(COBOL での開発に関して)
  - .NET Framework のドキュメントの 「.NET Framework」
  - 他の環境および言語との相互運用
- オブジェクト指向 COBOL プログラミング入門
  - オブジェクト指向プログラミング(オブジェクト指向 COBOL の基本)
- その他の言語と COBOL を混在 (または、クラスから継承) させる方法
  - 他の環境および言語との相互運用
- ASP.NET Web アプリケーションおよび XML Web サービスを記述する方法
  - ASP.NET Web アプリケーションの開発(チュートリアル)
  - XML Web サービスの開発(チュートリアル)
  - Web アプリケーションの開発
  - XML Web サービスの開発
- SQL CLR データベースオブジェクトを記述する方法
  - SQL CLR データベースオブジェクトの開発(チュートリアル)
  - SQL CLR データベースオブジェクトの開発
- 翻訳オプション、実行環境変数、ファイル入出力状態の値、データ型マッピングなどを参照する
  - リファレンス

- さまざまな NetCOBOL for .NET プログラミング技術を説明するサンプル
  - サンプルアプリケーション
- プログラムをビルドする方法
  - ビルド
- プログラムを実行する方法
  - アプリケーションの実行
- プログラムをデバッグする方法
  - デバッグ
  - デバッグ(XML Web サービスのデバッグ)
- ASP.NET を扱う方法
  - Web アプリケーションの開発
  - XML Web サービスの開発
- WCF(Windows Communication Foundation)サービスを記述する方法
  - Windows Communication Foundation サービスの開発(チュートリアル)
  - Windows Communication Foundation を利用したアプリケーションの開発

NetCOBOL for .NET の特徴

NetCOBOL for .NET は COBOL コンパイラと関連ツールから構成されており、 .NET プラットフォーム向けアプリケーションの作成をサポートします。 具体的には、以下のものが提供されています。

- .NET プラットフォームの Microsoft Intermediate Language (MSIL)コードを生成し、Unicode をサポートした COBOL コンパイラ
- .NET プラットフォームの機能にマッピングされたオブジェクト指向プログラミング構文
- 既存のオブジェクト指向 COBOL 構文では記述できない .NET プラットフォームの機能をサポートするための拡張構文
- 生成ファイルの属性を指定するために追加された NetCOBOL for .NET 固有の翻訳オプション
- COBOL での ASP.NET ページ記述のサポート(*1)
- .NET プラットフォーム上でのソリューション および パッケージ構築への COBOL 参加を可能にする機能
- Visual Studio への COBOL の統合
- COBOL ファイルメンテナンスユーティリティ
- COBOL 実行環境変数と ODBC 情報を設定するためのユーティリティ

これらによって、COBOL プログラマは以下のようなことが可能になります。

- 米国 Microsoft Corporation の.NET アーキテクチャと クラスライブラリを十分に活用できます。
- COBOL プログラムと他の言語をシームレスに統合できます。 たとえば、他の言語で作成されたクラスを継承したり、 他の言語で記述されたメソッドを呼び出したり、 他の言語からメソッドを呼び出されたりすることができます。
- 複数言語が混在したアプリケーションをひとつのデバッガー（Visual Studio デバッガー）から デバッグすることができます。
- Web サービスと ASP.NET ページのビジネスロジックに COBOL を使用することができます。
- .NET Framework で最新技術が利用できるようになると、 直ちにその最新技術を利用できます （もはや、その技術の COBOL 対応を待つ必要はありません）。

*1:**Web サイトの説明は互換のために残されています。**Visual Studio では、Web サイトの作成を推奨していません。

NCOBU00005

NetCOBOL for .NET の新機能

ここでは、Windows 版 NetCOBOL と比較して、NetCOBOL for .NET に新しく導入された機能を説明します。機能の詳細な説明へのリンクも含まれています。

概要

.NET Framework はプログラミング言語に依存しない環境となるように開発されています。 つまり、 .NET Framework は、 あらゆるプログラミング言語の実行環境となることができ、 異なる言語で記述されたプログラムがあたかも同じ言語で記述されているかのような相互運用を可能にする共通言語実行基盤（CLI）を提供しています。

NetCOBOL for .NET は、 この環境を十分に活かせるように設計されており、 そのために多くの新機能を導入しました。

.NET Framework との統合

Microsoft Intermediate Language （MSIL）

NetCOBOL for .NET コンパイラは、 Windows 版 NetCOBOL のようにネイティブマシンコードを生成しません。 かわりに、共通言語ランタイム（CLR）配下で実行される、Microsoft Intermediate Language モジュールを作成します。 CLR 配下で実行されるコードを「マネージコード」と呼びます。 通常のネイティブコードモジュールなど、 CLR 配下で実行しないコードを「アンマネージコード」と呼びます。

.NET データ型

異なる言語で記述されたプログラムを実行して相互運用できるように、 .NET Framework は異なる言語間でも扱うことのできるデータ型のセットを定義しています。それらのデータ型はオブジェクトとして定義され、取り扱われます。 一部のデータ型は COBOL の組込みデータ型に直接対応付けられており、 NetCOBOL for .NET コンパイラが COBOL データと対応する.NET データ型との変換を行います。他の.NET データ型は COBOL の組込みデータ型として扱わず、 オブジェクト参照として取り扱います。 詳細は、.NET Framework のデータ型を参照してください。

他の環境および言語との相互運用

.NET Framework は、 異なる言語で記述されたクラスを継承することができるオブジェクト指向環境です。 このような継承を COBOL からも可能にするために、 NetCOBOL for .NET は必要な構文をコンパイラに追加しました。 詳細は、他の.NET 言語との相互運用を参照してください。

XML Web サービスの作成(*1)

COBOL で XML Web サービスが作成できるということは、 .NET Framework クラスを継承することができるという機能から得られる利点のひとつです。 詳細は、XML Web サービスの開発を参照してください。

ASP.NET ページでの COBOL の使用(*1)

NetCOBOL for .NET は、 COBOL を ASP.NET の記述言語として使用するために必要な CodeDOM (Code Document Object Model) を提供しています。 詳細は、Web アプリケーションの開発および XML Web サービスの開発を参照してください。

Windows Communication Foundation(WCF)

WCF は.NET Framework 3.0 で追加された新しい分散テクノロジです。 これまで、XML Web サービスを含め、通信要件に応じてコンポーネントを選ぶ必要がありましたが、WCF ではプログラミングモデルが統一され、ひとつのアプリケーションで様々な通信方法のサービスを提供することができるようになりました。 NetCOBOL for .NET と.NET Framework の Windows Communication Foundation(WCF)を組み合わせることで、WCF サービスアプリケーションを COBOL で作成することができます。 詳細は、Windows Communication Foundation を利用したアプリケーションの開発と Windows Communication Foundation サービスの開発を参照してください。

ASP.NET AJAX 機能(*1)

.NET Framework 3.5 以降では、ASP.NET AJAX 機能が標準化され、NetCOBOL for .NET でも AJAX 対応の Web アプリケーションを簡単に作成することができます。 ASP.NET AJAX 機能を使用することで、応答性に優れた使いやすいユーザインタフェース (UI) をもつ Web ページを作成することができます。 詳細は、ASP.NET AJAX 機能を利用するを参照してください。

COBOL 構文

.NET プラットフォームのプログラミングをサポートするために COBOL 言語がどのように拡張されているかは、COBOL プログラマが詳しく知っておかなければならない重要な項目です。 NetCOBOL for .NET は既存の多くの オブジェクト指向言語要素を.NET の機能に対応づけました。 しかし、.NET プラットフォームのあらゆる機能をサポートするためには、 さらなる拡張構文を追加する必要もありました。

以下に説明する拡張構文は、 COBOL 文法書の「第 11 章 .NET プログラミング機能」に詳しく説明されています。

静的メンバー

.NET クラスには、 クラスごとにただひとつのインスタンスを持つ、 静的データおよび静的メソッドを定義することができます。 これらを一般に「静的メンバー」と呼びます。 NetCOBOL for .NET では、 静的メンバーを定義するために、 クラス定義中に記述するスタティック定義が導入されています。 クラス定義は、 クラス名段落を含む見出し部、環境部、静的データや静的メソッドを定義するスタティック定義と、 インスタンスデータやインスタンスメソッドを定義するオブジェクト定義から構成されます。

プログラム原型定義

プログラム原型定義は他のプログラムの呼出しインタフェースを定義します。 NetCOBOL for .NET ではアンマネージコードを呼び出すためにだけプログラム原型定義を使用します。プログラム原型定義を記述するには、プログラム名段落に PROTOTYPE 句を記述します。 プログラム原型定義のデータ部には連絡節だけが記述でき、 手続き部には手続き部の見出しだけを記述することができます。 詳細は、プログラム原型定義を参照してください。

インタフェース定義

インタフェースはメソッドがどのような形式のものであるか (名前、引数、および戻り値) を表す一種の約束です。 NetCOBOL for .NET では、 インタフェースの定義および実装をサポートしています。 インタフェースの定義は、 見出し部に INTERFACE-ID 段落を記述し、 手続き部にメソッドの形式を宣言します。 インタフェースを実装するには、 オブジェクト段落に IMPLEMENTS 句を記述し、 実装するインタフェース名を宣言します。 そのインタフェース名をリポジトリ段落に記述することによって、 インタフェース名とインタフェースを関連付けることができます。

デリゲート定義

デリゲートは「手続きへのポインタ」を実現するためのタイプセーフな方法です。 NetCOBOL for .NET でデリゲートを定義するには、 見出し部に DELEGATE-ID 段落を記述します。 デリゲート定義には、 デリゲートを通して呼び出される手続きのインタフェースを記述します。 詳細は、デリゲートを使うを参照してください。

カスタム属性

カスタム属性によって出力されるメタデータを拡張することができます。 メタデータとは、 .NET コードが自分自身を記述している情報です。 カスタム属性を指定することによって、 独自の情報を出力コードの中に入れることができます。 カスタム属性は、 System.Attribute クラスを継承したクラスによって定義されます。 特殊名段落の CUSTOM-ATTRIBUTE 句でカスタム属性を導入し、 カスタム属性を設定したい対象を定義する際にそれらを参照します。 詳細は、カスタム属性を使うを参照してください。

プロパティ

NetCOBOL for .NET でのプロパティは、 いくつかの細かい点を除いてオブジェクト指向 COBOL のプロパティと同じです。 また、NetCOBOL for .NET では、 翻訳単位内で使用している内部名とは異なる名前を外部名としてプロパティにつけることができます。 プロパティはオブジェクト中のデータを取得したり設定したりする手段を提供します。 プロパティは、 スタティック定義の作業場所節、または、 オブジェクト定義の作業場所節のデータに PROPERTY 句を付けるか、メソッド名段落に GET PROPERTY 句や SET PROPERTY 句を記述することで定義します。

列挙型

列挙型は名前の付けられた定数のセットです。 NetCOBOL for .NET では、列挙型の定義と参照ができます。 列挙型の定義は、見出し部に ENUM-ID 段落を記述し、 データ部の作業場所節にデータを宣言します。 列挙型の参照は、ENUM 指定子をリポジトリ段落に記述することによって、 手続き中で列挙型を利用することができます。 また、列挙型を操作するために、 ENUM-OR 組込み関数、ENUM-AND 組込み関数および ENUM-NOT 組込み関数が用意されています。 詳細は、列挙体を使うを参照してください。

.NET データ型のサポート

.NET Framework は各言語で共通に利用できる 基本データ型のセットを定義しています。 それらの型は次のものです。

- boolean
- char
- float32 (float)
- float64 (double)
- int16 (short)
- int32 (int)
- int64 (long)
- native int
- object
- string
- unsigned int8 (byte)

NetCOBOL for .NET では、 適切に定義された 01 または 77 レベルのデータ項目を自動的にマッピングしたり、 COBOL の組込みデータ型と.NET データ型の間で 値を変換することによって、これらのデータ型を取り扱うことを可能にしています。

アクセス属性(PRIVATE, PUBLIC, PROTECTED)

.NET プラットフォームでは、 private、public、または protected というアクセス属性を メソッドに指定することができます。 NetCOBOL for .NET ではアクセス属性を メソッド名段落に PRIVATE、PUBLIC、および PROTECTED 句で指定します。

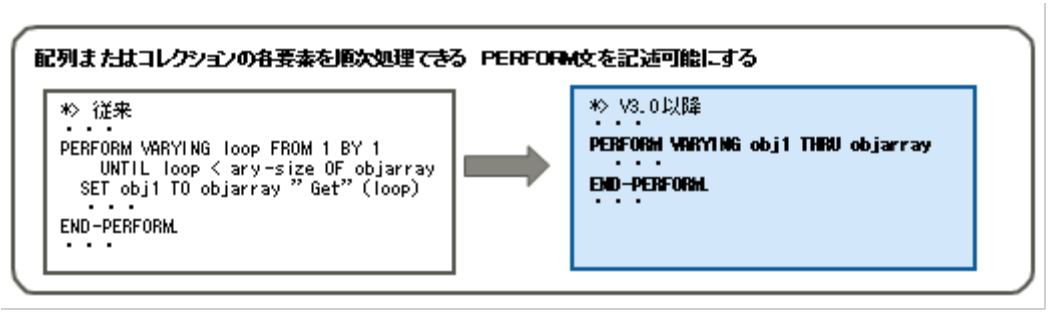
ジェネリック

.NET Framework に導入されたジェネリックの 機能を利用するために、NetCOBOL for .NET では、型パラメタを持つ型を参照できるようにしました。 このため、リポジトリ段落におけるいくつかの指定子の書き方を拡張しています。 また、型パラメタ付きのメソッドを呼び出せるようにするため、メソッド指定子を導入しました。 詳細は、ジェネリックの機能を使うを参照してください。

コレクションに対する反復処理

NetCOBOL for .NET では、C#の foreach 反復処理に相当する構文として、 PERFORM 文を拡張しました。 この構文を使用すると、コレクション（オブジェクトの集まり）の個々の要素を簡単に取り出すことができます。

従来、配列またはコレクションの各要素に対する反復処理を記述する場合、 PERFORM 文中に要素を取り出す処理と要素に対する処理を記述する必要がありました。 拡張した PERFORM 文を使用することで、要素を取り出す処理が不要になります。

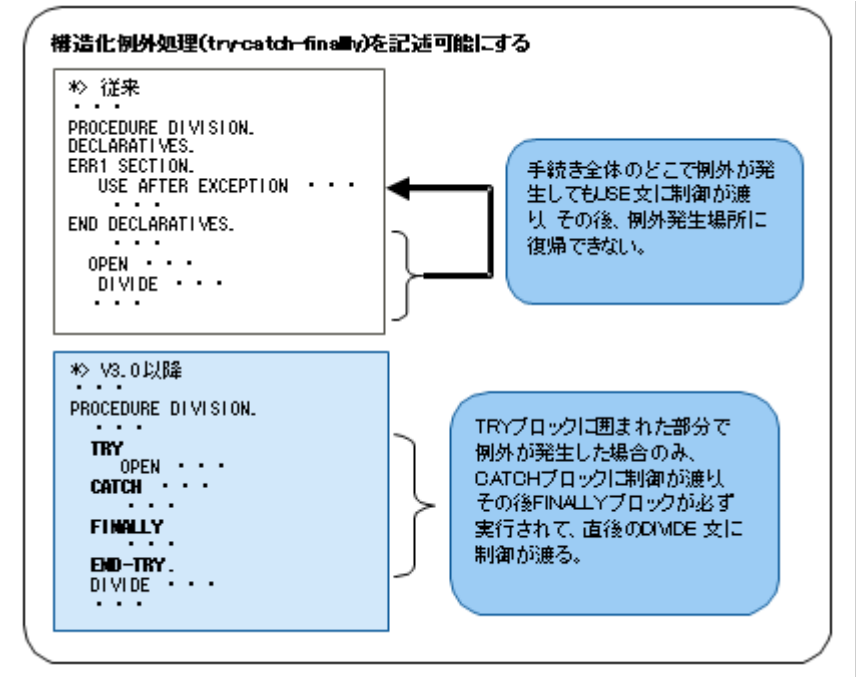


この機能は、NetCOBOL for .NET V3.0 からサポートしています。

構造化例外処理

NetCOBOL for .NET では、COBOL 規格である USE 文による例外処理とは別に、 .NET システム標準である構造化例外処理を使用することができます。 構造化例外処理とは、ひとつの手続き内部をブロック化して例外処理 および終了処理を定義する手法であり、例外処理のネスト(入れ子構造)も 実現します。構造化例外処理を使用する場合、手続きに TRY 文を記述します。 また手続きブロックを任意の位置で終了する EXIT TRY 文、および ブロック内の任意の位置で例外を発生させる RAISE 文も用意しています。 詳細は、COBOL 文法書における各文の記事および構造化例外処理(TRY 文)を参照してください。

従来の USE 文による例外処理では、宣言節に USE 文を記述することから プログラム単位またはメソッド単位でしか例外を拾うことができませんでした。 構造化例外処理を使用すると、局所的に例外を拾うことや、 例外処理後に必ず実行する処理を記述することができ、 例外発生時の処理をより細かに制御できるようになります。



この機能は、NetCOBOL for .NET V3.0 からサポートしています。

開発環境

NetCOBOL for .NET では Visual Studio を使って COBOL プログラムを開発することができます。 また、コマンドラインから開発作業を行うこともできます。 詳細は、NetCOBOL for .NET アプリケーションの開発を参照してください。

IntelliSense 機能

IntelliSense 機能は、ユーザの操作を検出して、 そのときに必要なヒントなどを自動的に表示するコーディング時に役立つ機能です。 NetCOBOL for .NET では、以下をサポートしています。

- メンバーの一覧
- パラメタ ヒント
- 入力候補
- コードのアウトライン表示
- クラス ビュー／オブジェクト ブラウザ（読み取り専用）

詳細は、IntelliSense 機能を参照してください。

[注意]: IntelliSense は、 エディターにソースコードが入力されると、コードを補完するための解析を行います。 IntelliSense パーザは、不完全でエラーを含むコードを補わなければならないため、 信頼性が高くない場合があります。 正しいコードが IntelliSense パーザでは制限と解析されるようなこともあるため、IntelliSense はコード化の補助として使用してください。

この機能は、NetCOBOL for .NET V1.1 からサポートしています。

コード スニペット機能

コード スニペット機能を利用すると、あらかじめ作成されたコードのスニペット（断片）を簡単な操作でエディターに挿入することができます。 詳細は、コード スニペット機能を参照してください。

コンパイラオプション

.NET に固有な設定をサポートするために 多くのコンパイラオプションを追加しています。 コンパイラオプションの一覧は、コンパイラオプション - 詳細を参照してください。

翻訳オプション

従来の翻訳オプションの一部をサポートしています。 翻訳オプションの一覧については、翻訳オプション - 詳細を参照してください。

アセンブリのビルドや利用

.NET アプリケーションは、 アセンブリと呼ばれる単位で配布されます。 NetCOBOL for .NET では、 COBOL を使ってアセンブリを作成することができます。 詳細は、アプリケーションの配置を参照してください。

Visual Studio デバッガーでのデバッグ

NetCOBOL for .NET では Visual Studio デバッガーを使用して COBOL アプリケーションをデバッグできます。 詳細は、デバッグを参照してください。

Team Foundation Server 連携

Microsoft 社が提供するアプリケーションライフサイクル管理ソリューションのプラットフォームである Team Foundation Server と連携できます。

マニフェストファイルの埋め込み

NetCOBOL for .NET で作成するアプリケーションに、マニフェストファイルを埋め込むことができます。 詳細は、以下を参照してください。

- [アプリケーション] ページ (プロジェクト デザイナー)
- /win32manifest (Win32 マニフェストファイルの埋め込み)
- /nowin32manifest (Win32 マニフェストファイルの埋め込み禁止)

*1:**Web サイトの説明は互換のために残されています。**Visual Studio では、Web サイトの作成を推奨していません。

NCOBU00006

NetCOBOL for .NET の追加機能

NetCOBOL for .NET V8.0

NetCOBOL for .NET V8.0 を使用して、 .NET Framework 上で動作するアプリケーションを開発・運用する場合に、 以下の機能を使用できます。

機能	関連トピック
Visual Studio 2017 対応	Visual Studio 2017 を開発環境として使用できます。
.NET Framework 4.6.1/4.6.2/4.7/4.7.1 対応	.NET Framework 4.6.1/4.6.2/4.7/4.7.1 を対象としたアプリケーションを開発、運用することができます。
XML ドキュメントコメントのサポート	XML ドキュメントコメント
	/doc (XML ドキュメントファイルの出力を指定)
NuGet のサポート	NuGet パッケージマネージャーの利用
COMP-6 サポート	COBOL 文法書の「10.7 符号の領域がない内部 10 進項目」
IBM LE サブルーチンサポート	「NetCOBOL for .NET LE サブルーチン ユーザーズガイド」
SQL BigInt 型サポート	算術データの対応表（ADO.NET）
	算術データの対応表（ODBC）
USE 手続き終了後の Windows 版の動作をサポート	COBOL 標準の例外処理の概要
	USEEXTRET (USE 手続き終了後の動作指定)

NetCOBOL for .NET V7.0

NetCOBOL for .NET V7.0 を使用して、 .NET Framework 上で動作するアプリケーションを開発・運用する場合に、 以下の機能を使用できます。

機能	関連トピック
Visual Studio 2015 対応	Visual Studio 2015 を開発環境として使用できます。
.NET Framework 4.5.1/4.5.2/4.6 対応	.NET Framework 4.5.1/4.5.2/4.6 を対象としたアプリケーションを開発、運用することができます。
CSV サポート	COBOL 文法書の「6.4.45 STRING 文」
	COBOL 文法書の「6.4.50 UNSTRING 文」
	CSV 形式データの操作

		@CBR_CSV_OVERFLOW_MESSAGE（CSV 形式データ操作時のメッセージ抑止指定）
		@CBR_CSV_TYPE（生成する CSV 形式のバリエーション）
RECURSIVE 指定および局所記憶節のサポート		COBOL 文法書の「2.3.8 プログラムの再帰属性」
		COBOL 文法書の「5.1 データ部の構成/ 局所記憶節 (LOCAL-STORAGE SECTION)」
ストアドプロシージャ性能改善		@SQL_STOREDPROCEDURE_CACHE（ストアドプロシージャキャッシュ指定）
Micro Focus COBOL 互換強化	COBOL CBL サブルーチンサポート	「NetCOBOL for .NET CBL サブルーチン ユーザーズガイド」
AcuCOBOL 互換強化	ACUCOBOL サブルーチンサポート	「NetCOBOL for .NET AcuCOBOL サブルーチン ユーザーズガイド」
		@CBR_AC_FILENAME_SPACES（ファイル名の空白処理）
COBOL ファイルアクセスルーチンサポート		「NetCOBOL for .NET COBOL ファイルアクセスルーチン ユーザーズガイド」
データプロバイダー拡張機能		データプロバイダー拡張機能を使用したデータベースアクセス
		@CBR_DATAPROVIDER_EXTENSION_FOLDER（データプロバイダー拡張ライブラリのフォルダ指定）
		@SQL_DATAPROVIDER_EXTENSION_DLL（データプロバイダー拡張ライブラリの DLL 名指定）
データベース分散トランザクション		.NET 言語とトランザクション連携
		@SQL_COMMIT_MODE (COMMIT モードの指定)
ACCEPT FROM SSIN 機能追加		@CBR_ACCEPT_COMPATIBILITY （Micro Focus COBOL 仕様による ACCEPT 文動作の指定）
データベース NULL 文字処理		@SQL_X_NULL_TERM（英数字項目に含まれるデータの NULL 値の無効化）
PICTURE 句の文字列を 50 文字まで拡張		COBOL 文法書の「5.4.9 PICTURE 句」

NetCOBOL for .NET V5.0

NetCOBOL for .NET V5.0 を使用して、 .NET Framework 4.5 上で動作するアプリケーションを開発・運用する場合に、 以下の機能を使用できます。

機能		関連トピック
Visual Studio 2012 対応		Visual Studio 2012 を開発環境として使用できます。
.NET Framework 4.5 対応		.NET Framework 4.5 を対象としたアプリケーションを開発、運用することができます。
エディター機能強化	定義へ移動機能のサポート	[定義へ移動]コマンドを使用してメソッドや変数の定義位置へ移動することができます。
	登録集を開く機能のサポート	登録集を開く機能
	参照の強調表示機能のサポート	選択しているシンボルを参照しているすべての個所を強調表示します。
	境界線の表示機能のサポート	[COBOL エディター]ツールバー
	IntelliSense 機能のメンバーの一覧に予約語およびコードスニペットの追加	メンバーの一覧[COBOL 固有] ([オプション] ダイアログ ボックス -[テキスト エディター] - [COBOL])
	型の色付けの追加	COBOL エディターで型を表す語に対して色付けを行います。
[エラー一覧]ウィンドウからヘルプトピックの呼び出しサポート		[エラー一覧]ウィンドウで F1 キーを使用してコンパイルエラーに対応する NetCOBOL for .NET メッセージ集のヘルプトピックを表示できます。

NetCOBOL for .NET V4.2

NetCOBOL for .NET V4.2 を使用して、 .NET Framework 4 上で動作するアプリケーションを開発・運用する場合に、 以下の機能を使用できます。

機能	関連トピック
ファイルサイズの拡張	他のファイルシステムの使用法
	@CBR_FILE_LFS_ACCESS (COBOL ファイルのサイズを拡張する指定)
/debug:pdbonly および/debug:minimal のサポート	/debug (デバッグのための情報を出力)
	Cobolc タスク
	[ビルド] ページ (プロジェクト デザイナー)
入力ホスト変数のパディング空白削除サポート	@SQL_SPACE_TRUNC (ホスト変数の入力値の後置空白の無効化)

FETCH FIRST/LAST 文サポート	COBOL 文法書の「8.6.4 FETCH 文」
	スクロール可能なカーソルを使用したデータの取得

NetCOBOL for .NET V4.1

NetCOBOL for .NET V4.1 を使用して、.NET Framework 4 上で動作するアプリケーションを開発・運用する場合に、以下の機能を使用できます。

機能		関連トピック
Visual Studio 2010 対応		Visual Studio 2010 を開発環境として使用できます。
.NET Framework 4 対応		.NET Framework 4 を対象としたアプリケーションを開発、運用することができます。
実行性能改善	そと PERFORM 文の性能改善	そと PERFORM 文がプログラム定義に記述されている場合、実行の開始に時間がかかる問題を改善しました。
Micro Focus COBOL 互換強化	カーソル自動クローズ機能サポート	@SQL_AUTO_CURSOR_CLOSE (トランザクションによるカーソルの振る舞い指定)
IBM DB2 互換強化	SQLCA 機能サポート	COBOL 文法書の 「8.2.4 SQLSTATE/SQLCODE」、「8.2.6 SQLERRD」
運用時のトラブル監視機能強化	CHECK 機能の実行時抑止	実行時オプションの形式
	DISPLAY 文の出力先にイベントログを追加	イベントログにメッセージを出力する
		@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL (DISPLAY UPON CONSOLE のイベントログ出力時のイベント種類指定)
		@CBR_DISPLAY_CONSOLE_OUTPUT (DISPLAY UPON CONSOLE のイベントログ出力指定)
		@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL(DISPLAY UPON SYSERR のイベントログ出力時のイベント種類指定)
		@CBR_DISPLAY_SYSERR_OUTPUT (DISPLAY UPON SYSERR のイベントログ出力指定)
		@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL(DISPLAY UPON SYSOUT のイベントログ出力時のイベント種類指定)
		@CBR_DISPLAY_SYSOUT_OUTPUT (DISPLAY UPON SYSOUT のイベントログ出力指定)
SQL Server 2005 対応	SNAPSHOT 分離レベルのサポート	@SQL_ISOLATION (トランザクション分離レベルの指定)

FETCH NEXT/PRIOR 文サポート	COBOL 文法書の「8.6.4 FETCH 文」
	スクロール可能なカーソルを使用したデータの取得
ファイルアクセス種別の一括指定サポート	ファイルの高速処理の一括の指定方法
ADO.NET 日付型パラメタのフォーマット指定機能サポート	@SQL_ADONET_DATETIME_FORMAT (日付型の書式指定)

NetCOBOL for .NET V4.0

機能		関連トピック
Visual Studio 2008 対応	WCF サービスの開発操作性の向上	Windows Communication Foundation サービスの開発
		Windows Communication Foundation を利用したアプリケーションの開発
	Web サイトプロジェクトの AJAX 機能サポート	ASP.NET AJAX 機能を利用する
	.NET Framework バージョン切り替え	[アプリケーション] ページ (プロジェクト デザイナー)
Windows Vista/ Windows Server 2008 対応	マニフェストファイルサポート	[アプリケーション] ページ (プロジェクト デザイナー)
		/win32manifest (Win32 マニフェストファイルの埋め込み)
		/nowin32manifest (Win32 マニフェストファイルの埋め込み禁止)
リソース Unicode 改善	DUPCHAR 翻訳オプションサポート	DUPCHAR (重複文字の扱い)
Micro Focus COBOL 互換強化	外部ファイルハンドラサポート	外部ファイルハンドラ
		@CBR_EXFH_API (外部ファイルハンドラで結合するファイルシステムの入口名の指定)
		@CBR_EXFH_LOAD (外部ファイルハンドラで結合するファイルシステムの DLL 名の指定)
ファイル機能強化	Btrieve ファイルに対する START 文にキーの一部を指定可能	他のファイルシステムの使用法

NetCOBOL for .NET V3.1

機能		関連トピック
.NET Framework 3.0 対応	Windows Communication Foundation サポート	Windows Communication Foundation を利用したアプリケーションの開発
		ファイルの種類
		新規ソースファイルの追加
生産性向上	コードスニペット機能サポート	コード スニペット機能
Web サイト強化	ASP.NET 2.0 AJAX Extension サポート	ASP.NET Web サイトを新規作成する
	マスターページサポート	
	テンプレート追加	ファイルの種類
	オプション中での相対パスの使用をサポート	登録集を利用する
記述性向上	RESUME 文サポート	COBOL 文法書の「11.8.3.14 RESUME 文」
	SYNCHRONIZED 句の集団項目サポート	COBOL 文法書の「5.4.14 SYNCHRONIZED 句」
動的プログラム構造の拡張	DLOAD 翻訳オプションサポート	DLOAD (プログラムの動的構造の可否)
ネイティブ連携改善	JMPCINT2/JMPCINT3 のサブルーチン化	JMPCINT2/JMPCINT3 サブルーチンの呼出し形式
組込み関数強化	NATIONAL 関数の MODE3、MODE4 対応(Microsoft マッピング対応)	@CBR_FUNCTION_NATIONAL (NATIONAL 関数の変換モードの指定)

NetCOBOL for .NET V3.0

機能		関連トピック
Visual Studio 2005/ .NET Framework 2.0 対応	MsBuild 対応	MSBuild によるプロジェクトのビルド
		MSBuild
		MSBuild タスク リファレンス (Fujitsu.COBOL.Build)
	Parial type 対応	部分型の使用
		COBOL 文法書の「11.5.1.2 クラス名段落（CLASS-ID）」
		COBOL 文法書の「11.5.1.7 インタフェース名段落（INTERFACE-ID）」
	ジェネリック対応	ジェネリックの機能を使う
		COBOL 文法書の「11.2.2 .NET プログラミング機能の用語」
		COBOL 文法書の「11.3.3.2 メソッドの行内呼出し」
		COBOL 文法書の「11.4.3 総称性」
		COBOL 文法書の「11.6.1.3 リポジトリ段落（REPOSITORY）」
		COBOL 文法書の「11.8.3.9 INVOKE 文」
SQL Server 2005 対応	SQL CLR 対応	-
	コンストラクタサポート	COBOL 文法書の「11.2.2 .NET プログラミング機能の用語」
		COBOL 文法書の「11.5.1.5 メソッド名段落」
		COBOL 文法書の「11.8.1 手続き部の見出し」
	カスタム属性構文拡張	COBOL 文法書の「11.5.1.2 クラス名段落（CLASS-ID）」
		COBOL 文法書の「11.5.1.9 CUSTOM-ATTRIBUTE 句」
	スタティックデータの readonly 化	INITONLY (INITONLY 属性の指定)
	スタティックメソッドの埋込み SQL 文対応	SQLSCOPE (埋込み SQL 文のカーソル・文識別子のスコープと寿命の指定)
		COBOL 文法書の「11.9 データベース（SQL）」
	SQL CLR 開発支援	/optionset (特定のオプションのセットを指定)
		SQL 情報設定のためのアプリケーション構成ファイルの形

		式
	ストアドプロシージャの復帰値取得	ストアドプロシージャの呼出し例
		COBOL 文法書の「8.2.6 SQLERRD」
		COBOL 文法書の「8.10.1 CALL 文」
	埋込み SQL 文の ADO.NET 対応	データベースアクセス
		SQL 情報
		ADO.NET で扱うデータ型との対応
		実行環境設定ユーティリティの使い方
64bit 対応	コンパイラオプション追加	/platform (プラットフォームの指定)
		/stack (スタックサイズの設定)
GS 互換	任意日付取得機能	任意の日付の入力
		@CBR_JOBDATE（任意の日付を取得）
		CURRENT-DATE 関数を利用した西暦の取得
	ファイル連結／追加書き機能	ファイルの連結
		ファイル追加書き
MF 互換	外部 10 進符号部の MF/88 形式対応	他形式の外部 10 進互換モード
記述性向上	int 型 2 進整数データ項目の集団項目サポート	COBOL 文法書の「5.4.17 USAGE 句」
	try-catch-finally 構文追加	COBOL 文法書の「11.8.2.7 例外条件」の“例外オブジェクト”
		COBOL 文法書の「11.8.3.6 EXIT 文」
		COBOL 文法書の「11.8.3.13 RAISE 文」
		COBOL 文法書の「11.8.3.18 TRY 文」
	新 PERFORM 構文(foreach 相当)追加	COBOL 文法書の「11.8.3.12 PERFORM 文」
実行時メッセージの出力方法の指定		@CBR_FILE_USE_MESSAGE (入出力エラーの実行時メッセージの出力)
		@CBR_MESS_LEVEL_CONSOLE(実行時メッセージの重

	大度指定)
	@CBR_MESS_LEVEL_EVENTLOG(実行時メッセージの重大度指定)
WC オプションの連結	/wc (翻訳オプションを指定)

NetCOBOL for .NET V2.1

機能	関連トピック
COBOL 資産移植のためのコード系の扱い	Unicode
	SCS (ソースプログラムのコード系)
	RCS (実行時データのコード系)
	@CBR_DISPLAY_SYSOUT_CODE (DISPLAY 文の出力ファイルのコード系の指定)
	@MessOutFile_CODE (メッセージを出力するファイルのコード系の指定)
	COBOL 文法書の「11.10.2.2 DISPLAY-OF 関数」
	COBOL 文法書の「11.10.2.6 NATIONAL-OF 関数」
アプリケーション構成ファイルでの実行環境設定	アプリケーション構成ファイル
実行環境設定ユーティリティ	実行環境設定ユーティリティ
プログラム原型定義 ウィザード	プログラム原型定義 ウィザード
Pervasive.SQL 対応	Btrieve ファイル
プロクシアセンブリの自動生成ツール	.NET リモート処理プロキシ作成ツール（genrp.exe）
データ例外検査の強化 CHECK(NUMERIC)	CHECK (CHECK 機能の使用の可否)
CHECK オプションのサブオペランド並列指定	
外部属性を持つデータおよびファイルの処理モードの混在	@CBR_EXTERNAL_RCS_OPTION(外部属性の RCS オプションチェックの指定)

NetCOBOL for .NET V2.0

機能	関連トピック
Visual Studio.NET 2003/.NET Framework 1.1 対応	
Visual SourceSafe(VSS)連携	
整列併合	整列併合機能（SORT 文および MERGE 文の使い方）
内部プログラム	
マルチスレッド	
スレッド間共有外部データと外部ファイル	スレッド間共有外部データと外部ファイル
データロックサブルーチン	データロックサブルーチン
他ファイルシステム(RDM ファイル)	PowerRDBconnector
型の機能追加	
BINARY-SHORT SIGNED	.NET 基本データ型
BINARY-LONG SIGNED	
BINARY-DOUBLE SIGNED	
列挙型(ENUM)定義	列挙型定義
インタフェース定義	インタフェース定義
フィールドへのアクセス指定	
カスタム属性サポート範囲の拡大	
ポインタデータ項目	ポインタデータ項目の使用方法
Boolean 型の比較	COBOL 文法書の「11.8.2.5 比較の規則」
	System.Boolean オブジェクトを比較する
16 進数字定数	COBOL 文法書の「10.2 16 進数字定数」
コンパイラオプション・翻訳オプション	
検証可能なコードの生成(/verifiable)	/verifiable (検証可能なタイプセーフコードを生成)
広域最適化(OPTIMIZE)	OPTIMIZE (広域最適化の扱い)
ソースのコード系 UTF-8 対応(SCS)	SCS (ソースプログラムのコード系)

ALPHAL(AUTO)	ALPHAL (プログラム中の英小文字の扱い)
ASCOMP5	ASCOMP5 (2 進項目の解釈)
EQUALS	EQUALS (SORT 文での同一キーデータの処理方法)
FLAGSW	FLAGSW (COBOL 文法の言語要素に対しての指摘メッセージ表示の可否)
INITVALUE	INITVALUE (作業場所節での VALUE 句なし項目の扱い)
SHREXT	SHREXT (外部属性に関する扱い)
SMSIZE	SMSIZE (PowerSORT が使用するメモリ容量を指定)
XREF	XREF (相互参照リストの出力の可否)
スタティック定義、オブジェクト定義、メソッド定義の定数節	COBOL 文法書の「11.4 プログラム構造」
コード変換関数(ACP-OF/UNICODE-OF)	COBOL 文法書の「11.10.2.1 ACP-OF 関数」
	COBOL 文法書の「11.10.2.7 UNICODE-OF 関数」

NetCOBOL for .NET V1.1

機能	関連トピック
IntelliSense 機能	IntelliSense 機能
実行性能改善	
ファイルの高速処理を行なう場合の、順ファイルの最大サイズをシステムの制限まで拡張	ファイルの高速処理
	他のファイルシステムの使用法
印刷処理	印刷処理

NetCOBOL for .NET チュートリアル

NetCOBOL for .NET には、 Visual Studio が同梱されており、 開発環境として、Visual Studio IDE (統合開発環境)を使います。

NetCOBOL for .NET でアプリケーションを開発する場合、 Visual Studio IDE を利用する方法と、 コマンドラインから開発する方法の二つがあります。

ここでは特に Visual Studio IDE を使ってアプリケーションを構築する手順について説明します。 コマンドラインから開発する方法については、コマンドラインからの開発を参照してください。

また、Visual Studio によるアプリケーションの開発については、 以下も参考にしてください。

- Visual Studio IDE を使った開発
- Web アプリケーションの開発
- XML Web サービスの開発
- SQL CLR データベースオブジェクトの開発
- Windows Communication Foundation を利用したアプリケーションの開発
- フォームデザイナの利用
- Visual Studio のドキュメントの 「Visual Studio でのアプリケーション開発」

このセクションの内容

Windows アプリケーションの開発

グラフィカルユーザインタフェース(GUI)を持った Windows アプリケーションを構築する手順について説明します。

ASP.NET Web アプリケーションの開発

ASP.NET Web アプリケーションを構築する手順について説明します。

XML Web サービスの開発

XML Web サービスアプリケーションを構築する手順について説明します。

SQL CLR データベースオブジェクトの開発

SQL CLR データベースオブジェクトを構築する手順について説明します。

Windows Communication Foundation サービスの開発

Windows Communication Foundation(WCF)サービスアプリケーションを構築する手順について説明します。

ASP.NET Web アプリケーションの実行のための事前準備

IIS の設定方法などの ASP.NET Web アプリケーションの実行のための事前準備について説明します。

ASP.NET Web アプリケーションおよび XML Web サービスアプリケーションを動作させるためには、IIS(Microsoft Internet Information Services)の設定が必要です。 IIS の設定方法については、ASP.NET Web アプリケーションの実行のための事前準備を参照してください。

Windows アプリケーションの開発

NetCOBOL for .NET と .NET Framework クラスライブラリを組み合わせることで、 Windows 上で動作する高度なグラフィカルユーザインタフェース(GUI)を持つアプリケーションを 構築することができます。

ここでは、Windows アプリケーションの概要、 Windows フォームを利用した Windows アプリケーションの開発手順について説明します。

このセクションの内容

Windows アプリケーションの概要

Windows アプリケーションの概要について説明します。

Windows アプリケーションのプログラム構造

Windows アプリケーションのプログラム構造について説明します。

Windows アプリケーションの開発手順

Windows アプリケーションの開発手順について説明します。

Windows アプリケーション開発時の注意事項については、Windows アプリケーション開発の注意事項を参照してください。

NCOBU00009

Windows アプリケーションの概要

NetCOBOL for .NET では、 .NET Framework の Windows フォームを使って Windows アプリケーションを開発することができます。

Windows フォームは、 Windows アプリケーションの土台となる Form クラスと、 ユーザインタフェースを作成するための Button や ListBox などのコントロールが収められた control クラスで画面のレイアウトを作成します。 これらは、System.Windows.Forms 名前空間に含まれています。

Windows フォームは、 Visual Studio の Windows フォーム デザイナーを使って画面をデザインします。 デザイナーは、Button の位置やサイズなどのプロパティの設定手続きを自動的に生成するため、 これらのコントロールをドラッグ&ドロップでフォームにレイアウトするだけで、 簡単に画面をデザインできます。

デザイナーは、画面の初期状態を作り出す手続きを、 InitializeComponent メソッドの中に展開します。このため、InitializeComponent メソッドの内容は、 コメントを含めて変更してはいけません。

- Visual Studio による Windows アプリケーションの開発方法については、クライアント アプリケーションの開発を参照してください。
- Windows Graphics Device Interface (GDI) の機能を強化した GDI+によって、さらに高度なグラフィック処理を行なう Windows アプリケーションを開発することができます。 GDI+についての詳細は、.NET Framework のドキュメントの Windows フォームにおけるグラフィックスと描画を参照してください。

Windows アプリケーションのプログラム構造

ここでは、Windows アプリケーションの COBOL プログラムの構造について説明します。

プロジェクトを作成する時に、[Windows アプリケーション]のテンプレートを選択すると、 自動的に次のような構造を持つファイルが作成されます。

Form1.cob

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "WindowsApplication1.Form1" INHERITS CLASS-FORM.        ・・[1]

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.

SPECIAL-NAMES.                                                                  ・・[2]

REPOSITORY.
    CLASS  CLASS-FORM AS "System.Windows.Forms.Form"                            ・・[3]
    .

OBJECT.                                                                         ・・[4]
DATA DIVISION.
PROCEDURE DIVISION.

METHOD-ID. DISPOSE AS "Dispose" OVERRIDE IS PROTECTED.

METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" IS PRIVATE.             ・・[5]

METHOD-ID. NEW.

イベント ハンドラ メソッド定義                                                  ・・[6]

END OBJECT.

END CLASS CLASS-THIS.
```

Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.                                 ・・[7]

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.

SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
    .

REPOSITORY.
    CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
    CLASS CLASS-MAINFORM AS "WindowsApplication1.Form1"
    CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
    .

PROCEDURE DIVISION.
    ...
    INVOKE CLASS-MAINFORM "NEW" RETURNING WK-MAINFORM.
    INVOKE CLASS-APPLICATION "Run" USING BY VALUE WK-MAINFORM.

END PROGRAM MAIN.
```

上記の COBOL プログラム構造のうち、特に重要な部分について説明します。

[1]: クラス定義(CLASS-ID 段落)

クラス定義は、COBOL のオブジェクト指向プログラミング機能で追加された構文です。クラスは、そのクラスから生成されるオブジェクト(インスタンス)の動作と属性を定義したものです。 オブジェクト指向プログラミングの機能の 1 つに、 任意のクラスを継承して新しいクラスを作成する機能があります。 Windows アプリケーションでは、 .NET Framework クラスライブラリに用意されている System.Windows.Forms.Form クラスを継承して、 新しい Windows フォームクラスを作成します。

[2]: 特殊名段落(SPECIAL-NAMES)

.NET Framework の重要な機能の 1 つであるカスタム属性を定義する場合、 ここに定義します。

[3]: リポジトリ段落(REPOSITORY)

この環境部(ENVIRONMENT DIVISION)を含むソース単位内で使われるクラス名やプロパティ名を指定します。 また、.NET Framework のクラス名(外部名)を、 ソース単位内で参照するための名前(内部名)と対応付けて指定します。

たとえば、上記の場合、.NET Framework のクラス名が"System.Windows.Forms.Form"、プログラム内で参照する場合の名前が"CLASS-FORM"になります。

デザイナーを利用してフォーム上に配置したコントロールのクラスやプロパティは、 デザイナーが自動的にリポジトリ段落に必要な情報を追加します。 また、Visual Studio エディターの IntelliSense 機能でも、 コントロールのプロパティの外部名に対応する内部名をリポジトリ段落に追加できます。

[4]: オブジェクト定義(OBJECT)

オブジェクトのデータやメソッドを定義します。 重要なメソッドやデータは、テンプレートやデザイナーで自動的に生成されますが、 必要に応じて独自のメソッドを定義することができます。

[5]: InitializeComponent メソッド

画面のレイアウトを作成する手続きが記述されます。 この手続きはデザイナーが自動生成します。

デザイナーで設計した画面のレイアウトは、 このコメントを含む COBOL の手続きによって表現されているため、 デザイナー以外でこの手続きを変更すると、 デザイナーが画面レイアウトを復元できなくなる場合があります。 このため、このメソッドを直接変更してはいけません。

[6]: イベント ハンドラ メソッド定義

Windows フォームで作成する Windows アプリケーションは、 「ボタンがクリックされた」などの動作を契機として手続きを実行します。 この契機をイベントといいます。 従来の上から下に処理が流れていく手続き型プログラミングモデルとは異なり、 イベント駆動型プログラミングモデルによってアプリケーションを作成します。

イベント ハンドラは、メソッドとして実装されます。 そして、デリゲートという機能によって、 イベントと結び付けられます。 デザイナーはイベント ハンドラ メソッド定義の最小限の手続きを自動的に生成するので、 プログラマはメソッド定義のデータ部や手続き部を記述するだけで済みます。

[7]: アプリケーション エントリ ポイント

アプリケーションを起動するためのエントリ ポイントです。プロジェクト プロパティ ページに表示される[アプリケーション] ページのスタートアップ オブジェクト に、 このプログラム名を指定します。プロジェクト プロパティ ページは、 Visual Studio の[プロジェクト]メニューから[(プロジェクト名)のプロパティ]を 選択することで表示されます。

NetCOBOL for .NET におけるオブジェクト指向機能については、オブジェクト指向プログラミングを参照してください。 また、COBOL の構文規則については、 COBOL 文法書の「第 11 章 .NET プログラミング機能」を参照してください。

NCOBU00011

Windows アプリケーションの開発手順

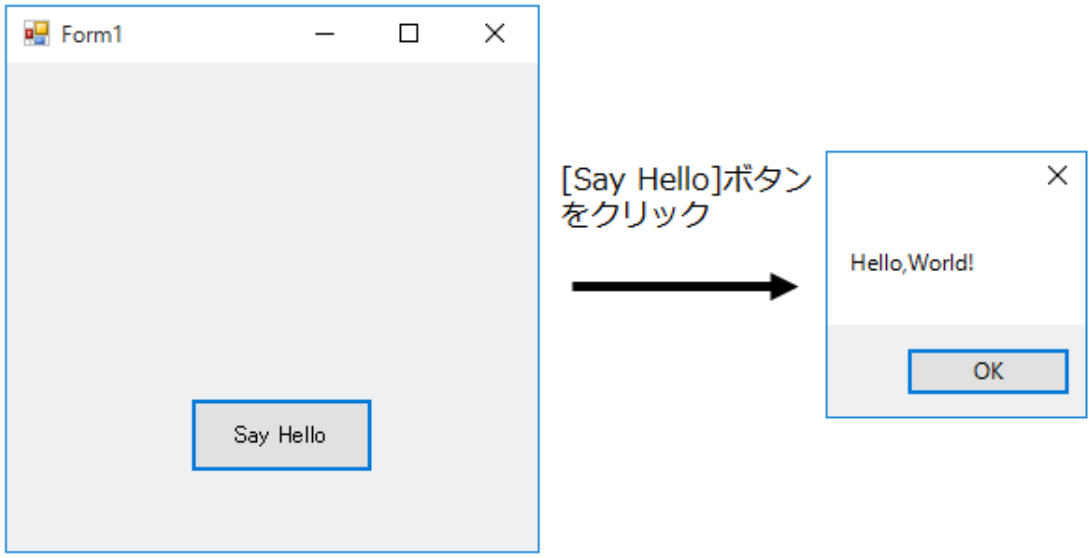
ここでは、簡単なサンプルを例にして、 Visual Studio で Windows アプリケーションを開発する場合の手順について説明します。

説明にしたがってアプリケーションを構築することによって、 Windows アプリケーション開発方法の基本を学習することができます。

このセクションの内容

メッセージボックスを表示する

ここでは、ボタンをクリックすると「Hello,World!」というメッセージボックスを表示する 簡単な Windows アプリケーションを作成してみましょう。

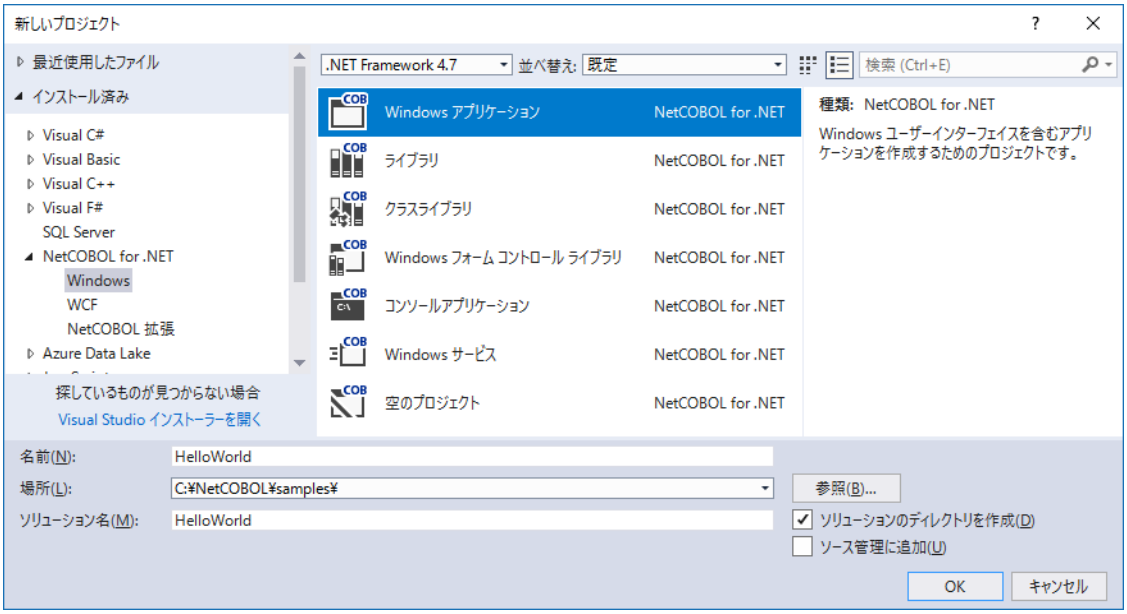


作成手順は以下の とおりです。

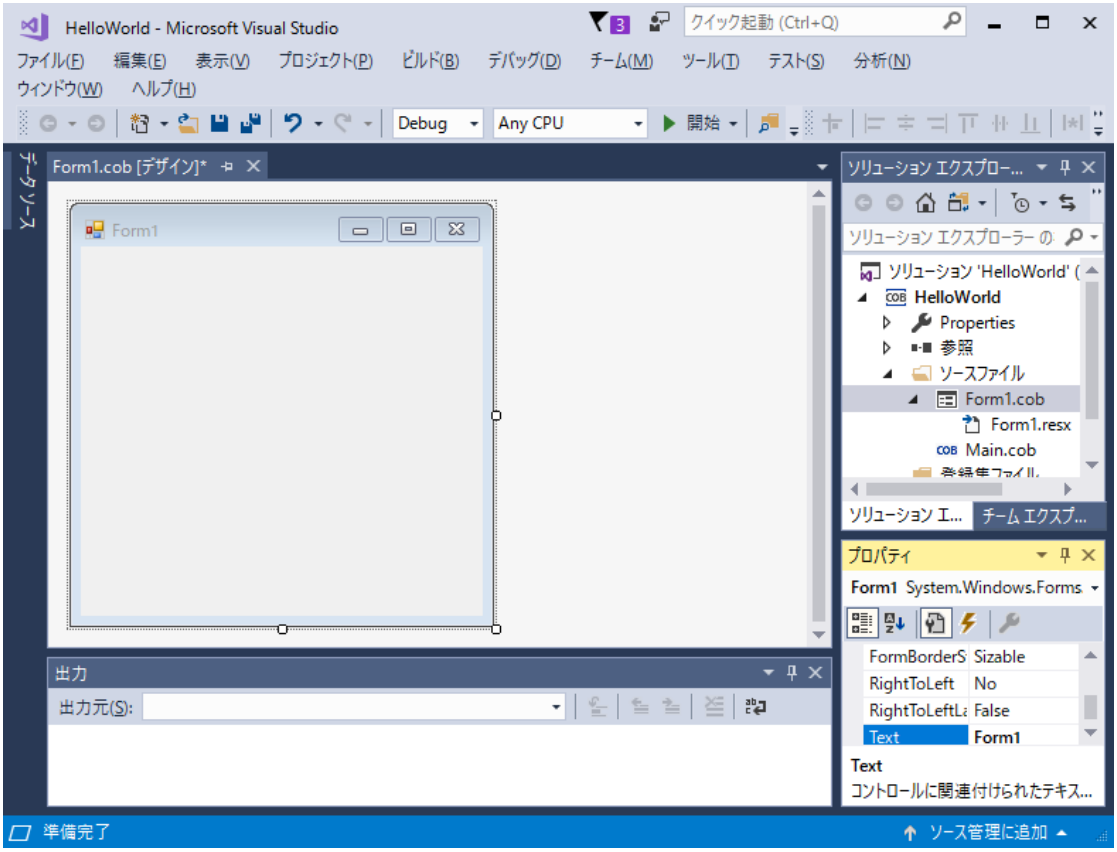
1. プロジェクトの作成
2. Windows フォームにボタンを貼り付ける
3. アプリケーションの手続きの記述

プロジェクトの作成

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[プロジェクト]を選択します。
3. [新しいプロジェクト]ダイアログボックスが表示されます。



4. 左ペインで、[NetCOBOL for .NET]を選択します。
5. 右ペインで、[Windows アプリケーション]を選択します。
6. [名前]に、新規プロジェクトの名前を入力します。 ここでは、「HelloWorld」とします。
7. [場所]で、新規プロジェクトの格納場所を選択します。 ここでは、「`c:¥NetCOBOL¥samples`」とします。
8. [OK]ボタンをクリックします。 これで、アプリケーションの構築に必要なファイル一式が作成されました。 プロジェクトが作成されると、Windows フォームデザイナーが自動的に起動されます。



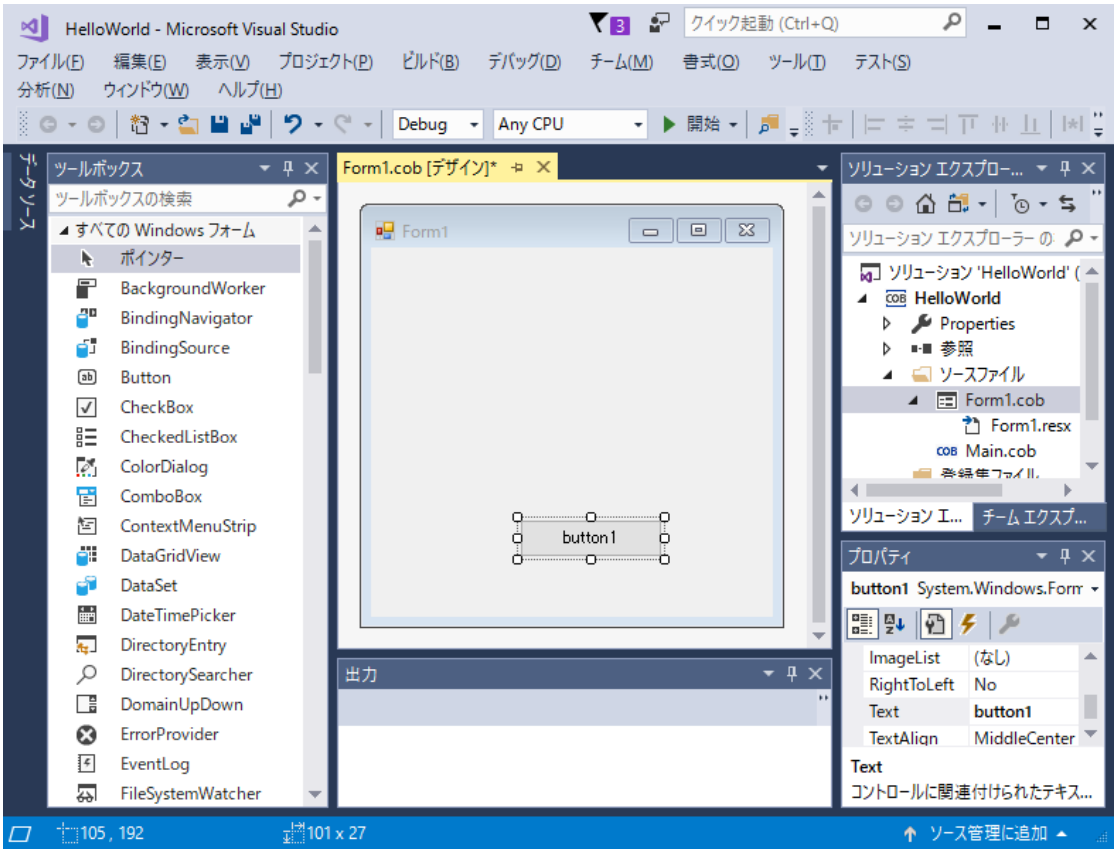
ソリューションエクスプローラーには、 プロジェクトの構成をあらわすツリーが表示されます。 ソリ

ューションエクスプローラーは、Visual Studio の[表示]メニューから、 [ソリューション エクスプローラー]を選択することで表示されます。

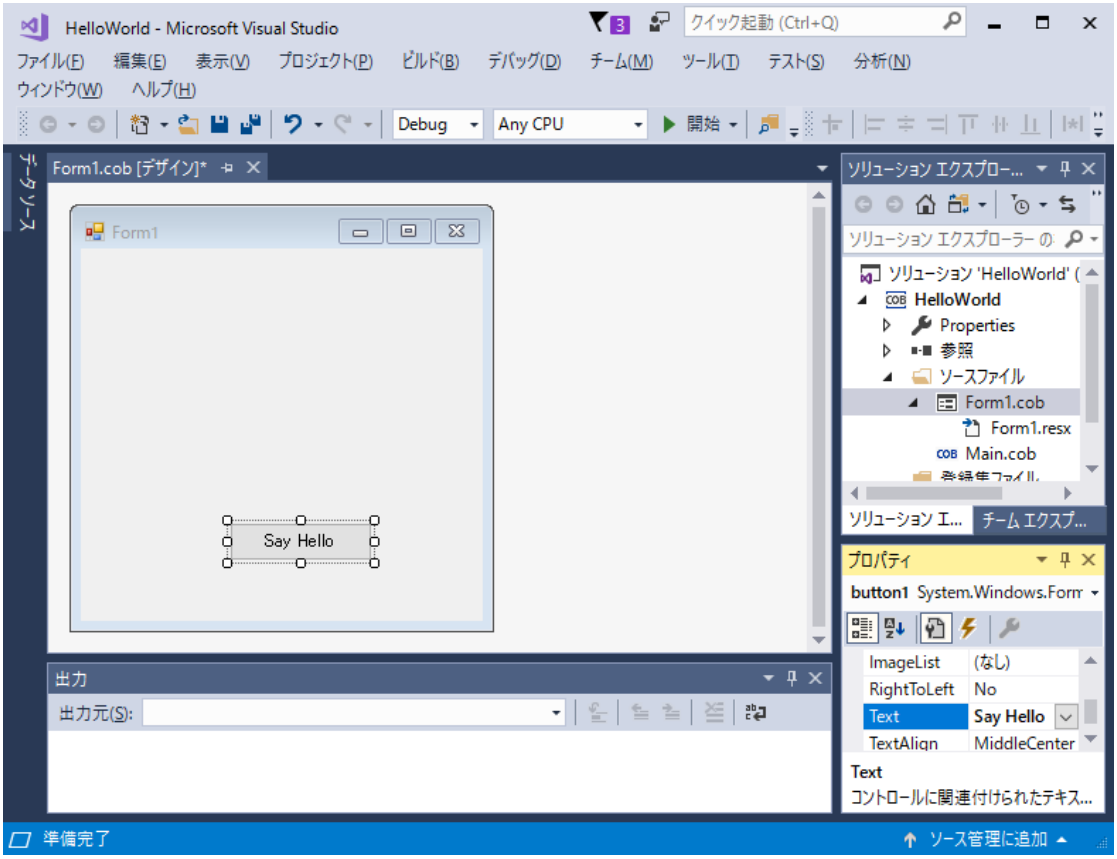
Windows フォームにボタンを貼り付ける

Windows フォームを使って、Button コントロールをフォームに貼り付けてみましょう。

1. ツールボックスを表示させ、 コントロールの一覧から「Button」アイコンをクリックします。 その後、マウスカーソルをフォームの適当な位置に移動させ、 ドラッグして配置します。 ツールボックスは、[表示]メニューを選択し、 [ツールボックス]をクリックすれば表示されます。



2. ボタンの表面には「button1」という文字列が表示されます。 これを「Say Hello」という文字列に変更します。 ボタン表面の文字列を変更する場合は、Visual Studio の[プロパティ] ウィンドウ を使って、Button コントロールの Text プロパティに「Say Hello」を設定します。 [プロパティ] ウィンドウは、Visual Studio の[表示]メニューから[プロパティ ウィンドウ] を選択することで開くことができます。 [プロパティ] ウィンドウ が開いているときに、フォームに貼り付けた Button コントロールをクリックすると、 Button コントロールのプロパティが表示されます。



参考

Windows フォームデザイナーの使い方については、 Visual Studio のドキュメントの Windows フォーム デザイナーを参照してください。 また、[プロパティ]ウィンドウの詳細については、[プロパティ] ウィンドウを参照してください。

アプリケーションの手続きの記述

実行中のアプリケーションのフォーム上のボタンをダブルクリックすると、 Click イベントが生成されます。 Click イベントは、 Click イベントに割り当てられた手続き(イベントハンドラといいます)を実行します。

ここでは、[Say Hello]ボタンがクリックされた時にメッセージボックスを表示させるため、 Click イベントに手続きを追加します。

1. Button コントロールをダブルクリックして、 COBOL ソースプログラムを編集するためのコードエディターを開きます。

   この操作で、イベントハンドラの挿入位置にカーソルが移動します。 この場合は、button1_Click メソッドの手続き部の入力位置に移動します。

2. 以下のクラス指定子をリポジトリ段落に挿入します。

```
       REPOSITORY.
           CLASS CLASS-MESSAGEBOX AS "System.Windows.Forms.MessageBox"
```

   このクラス指定子は、 System.Windows.Forms.MessageBox というクラス名(外部名)を、 CLASS-MESSAGEBOX という名前(内部名)に関連付けます。

   リポジトリ段落には、デザイナーが自動的に追加した、 System.Windows.Forms.Form クラスや

System.Windows.Forms.Button クラスなどのクラス指定子が既に指定されています。 このため、COBOL ソースプログラム内でこれらのクラスを参照する場合は、 ここで定義されている内部名で参照する必要があります。

なお、クラス指定子の挿入位置は、リポジトリ段落のどこでもかまいません。 ただし、リポジトリ段落の最終行に挿入する場合は、ピリオドの位置に注意してください。

AS 指定の書き方については、 COBOL 文法書の「11.3.4 外部名と内部名」および 「11.6.1.3 リポジトリ段落」を参照してください。

3. 以下の INVOKE 文を挿入します。

```
METHOD-ID. button1_Click PRIVATE.
DATA DIVISION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    INVOKE CLASS-MESSAGEBOX "Show" USING N"Hello,World!"
END METHOD button1_Click.
```

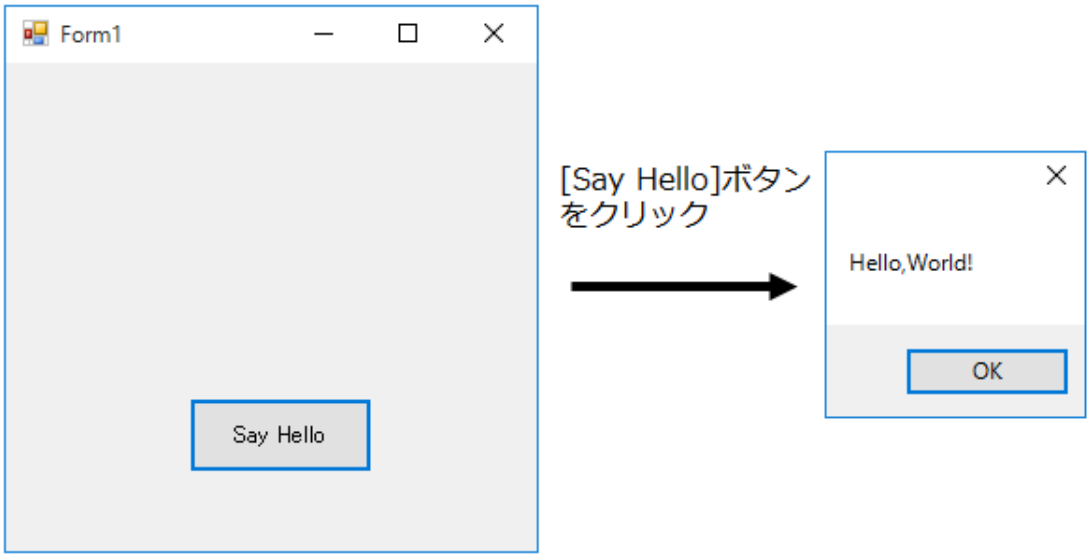
この INVOKE 文は、System.Windows.Forms.MessageBox クラスの Show メソッドを呼び出します。System.Windows.Forms.MessageBox は、System.Windows.Forms 名前空間に含まれ、 メッセージボックスを表示するための機能を提供します。 Show メソッドは static メソッドのため、 クラスインスタンスを生成することなく呼び出すことができます。System.Windows.Forms.MessageBox クラスの詳細については、 .NET Framework のドキュメントの MessageBox クラスを参照してください。

文は、B 領域(12 カラム以降)から記述します。 テンプレートの[Windows アプリケーション]を選択して生成された COBOL ソースプログラムには翻訳オプション NOALPHAL が指定されるため、 利用者語の英大小文字を区別します。 ただし、通常は区別されないため、 同じ名前で英大小文字だけが異なるクラス名やメソッド名を作成しないように注意してください。

4. COBOL ソースプログラム(Form1.cob)の変更を保存します。 保存する場合、[ファイル]メニューから[Form1.cob の保存]を選択します。

アプリケーションのビルドと実行

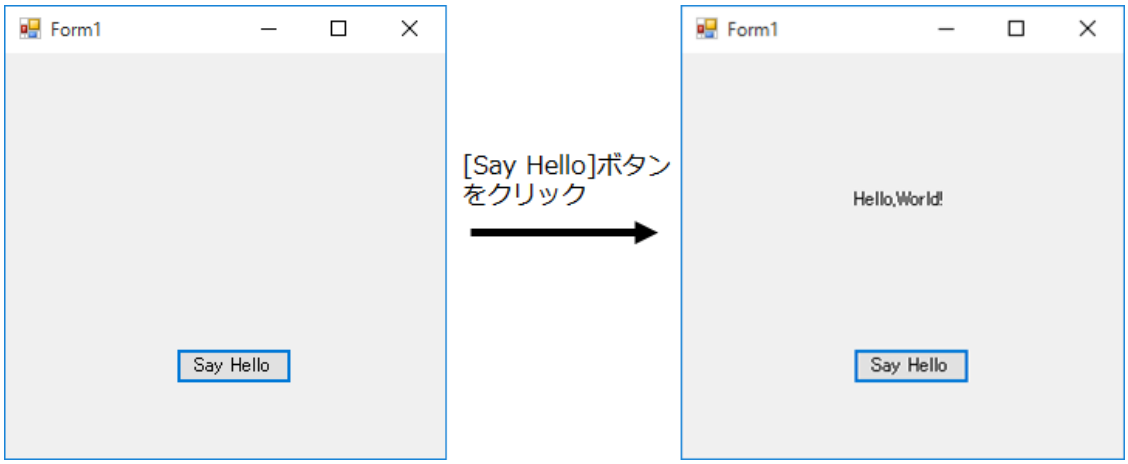
1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー配下で実行されます。
2. アプリケーションが実行されると、 [Say Hello]ボタンが貼り付けられたウィンドウが表示されます。[Say Hello]ボタンをクリックすると、 「Hello,World!」という文字列がメッセージボックスに表示されます。



3. メッセージボックスの[OK]ボタンをクリックし、 さらにアプリケーションの右上の「×」ボタンを押して、Windows フォームを閉じます。

ラベルに文字列を設定する

ここでは、ボタンをクリックすると「Hello,World!」という文字列を Windows フォームのラベルに設定する簡単な Windows アプリケーションを作成してみましょう。



作成手順は以下の とおりです。

1. プロジェクトの作成
2. Windows フォームにボタンとラベルを貼り付ける
3. アプリケーションの手続きの記述

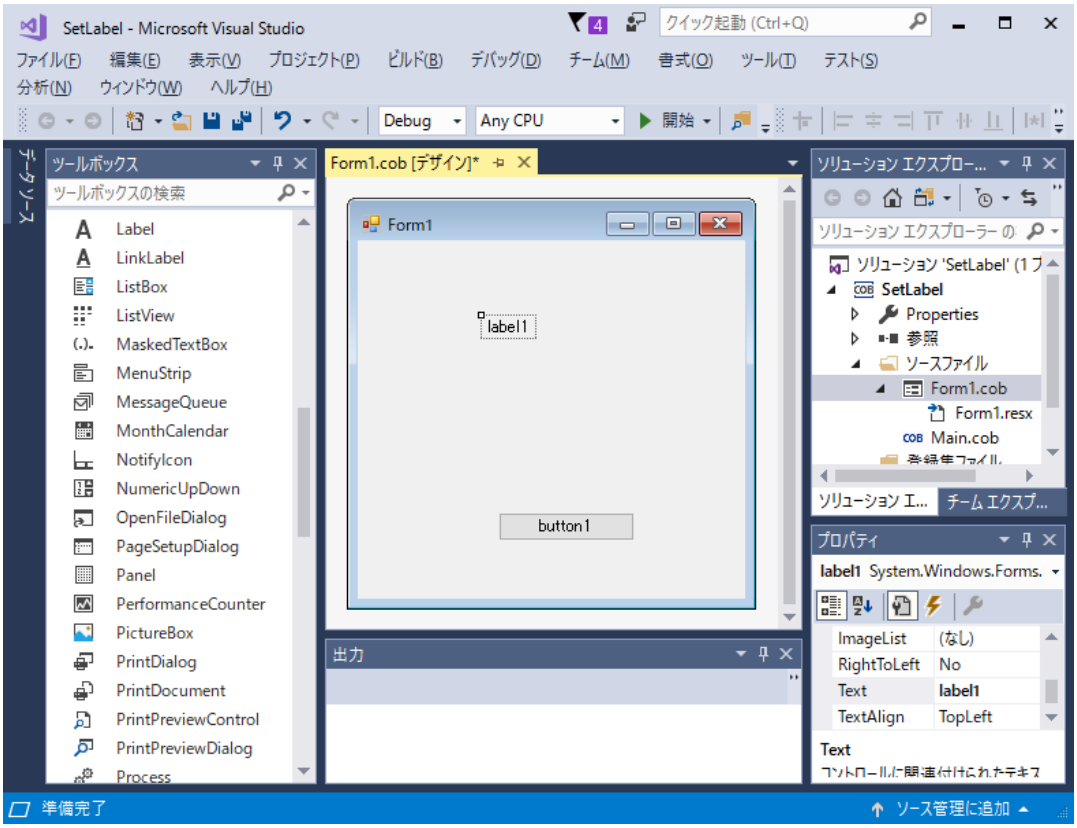
プロジェクトの作成

プロジェクトの作成は、メッセージボックスを表示するの「プロジェクトの作成」と同じ方法で作成します。 ここでは、プロジェクト名に、「SetLabel」を指定します。

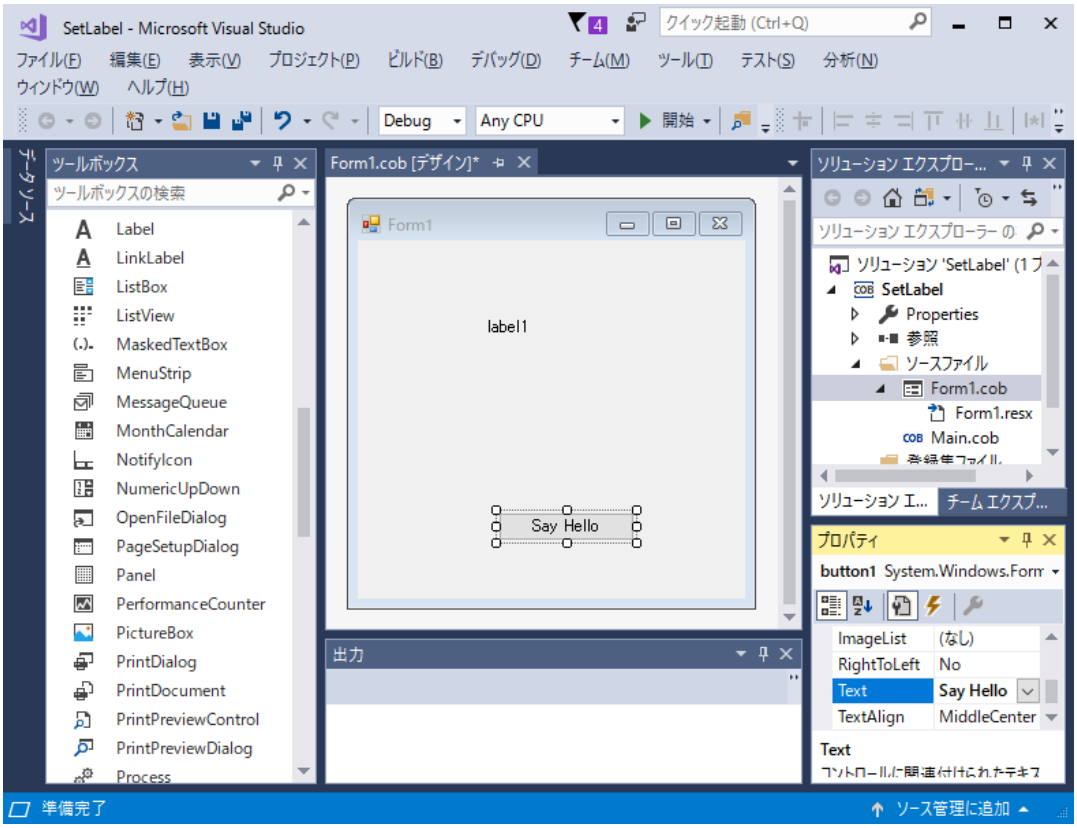
Windows フォームにボタンとラベルを貼り付ける

Windows フォームを使って、 Button コントロールと Label コントロールをフォームに貼り付けてみましょう。

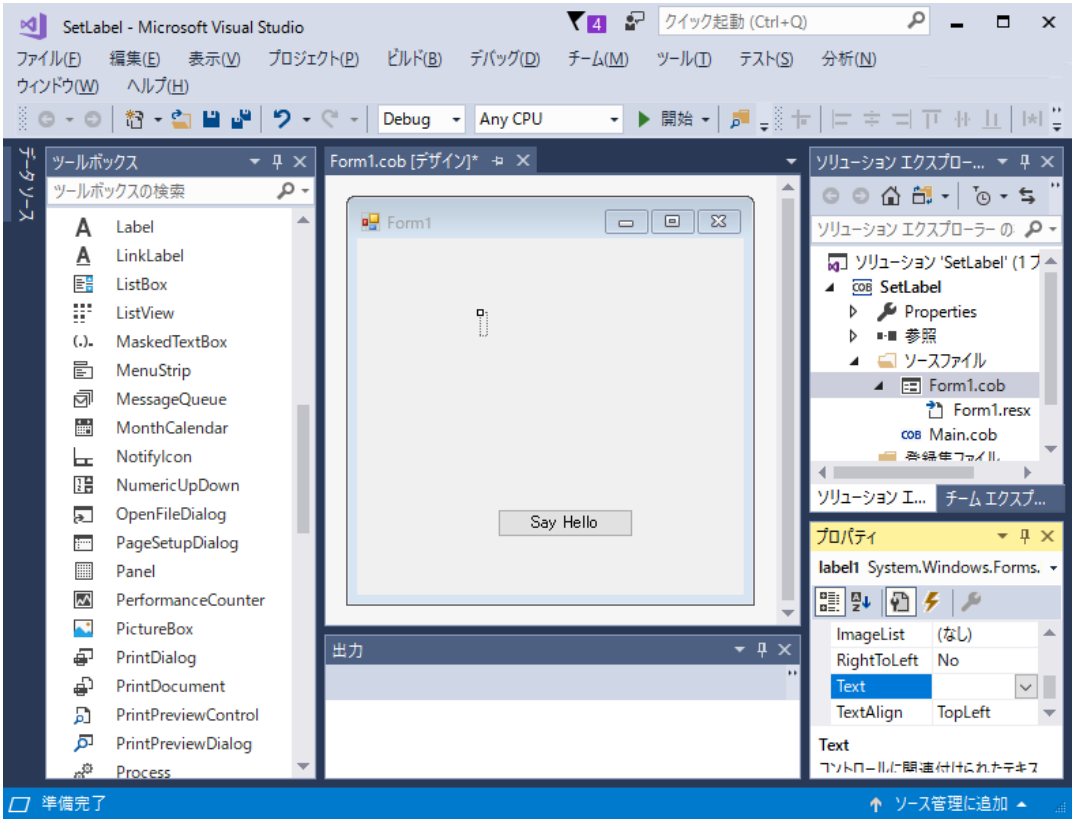
1. ツールボックスを表示させ、 コントロールの一覧から「Button」アイコンをクリックします。 その後、マウスカーソルをフォームの適当な位置に移動させ、ドラッグして配置します。 ツールボックスは、[表示]メニューを選択し、 [ツールボックス]をクリックすれば表示されます。 また、同じように Label コントロールも配置します。



2. ボタンの表面には「button1」という文字列が表示されます。 これを「Say Hello」という文字列に変更します。 ボタン表面の文字列を変更する場合は、Visual Studio の[プロパティ] ウィンドウ を使って、Button コントロールの Text プロパティに「Say Hello」を設定します。 [プロパティ] ウィンドウは、Visual Studio の[表示]メニューから[プロパティ ウィンドウ] を選択することで開くことができます。[プロパティ] ウィンドウ が開いているときに、フォームに貼り付けた Button コントロールをクリックすると、 Button コントロールのプロパティが表示されます。



3. 次に、Label コントロールをクリックし、 Label コントロールのプロパティウィンドウを表示させて、Text プロパティの内容をクリアします。



また、Name プロパティが「label1」であることを確認します。 もし、label1 でないなら、Name プロパティの値を変更するか、 これ以降の説明に出てくる「label1」の箇所を設定されている Name プロパティの値に置き換えてください。

アプリケーションの手続きの記述

実行中のアプリケーションのフォーム上のボタンをダブルクリックすると、 Click イベントが生成されます。Click イベントは、Click イベントに割り当てられた手続き(イベントハンドラといいます)を実行します。

ここでは、[Say Hello]ボタンがクリックされた時に、 Label コントロールに文字列を渡すための手続きを追加します。

1. COBOL ソースプログラムを編集するためのコードエディターを開きます。 コードエディターを開く方法はいくつかありますが、 ここでは Button コントロールをダブルクリックします。 この操作で、イベントハンドラの挿入位置にカーソルが移動します。 この場合は、button1_Click メソッドの手続き部の入力位置に移動します。

2. 以下の SET 文を挿入します。

```
METHOD-ID. button1_Click PRIVATE.
DATA DIVISION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    SET PROP-TEXT OF label1 TO N"Hello,World!"
END METHOD button1_Click.
```

この SET 文は、Label コントロールのクラスインスタンスが格納されている label1 オブジェクトの Text プロパティに、 文字列"Hello,World!"を設定します。

この時、「SET label1」と入力してから、IntelliSense 機能のメンバーの一覧を利用して"Text"プロパテ

ィを選択すると、 ソースプログラム上に「SET PROP-TEXT OF label1」が展開されます。 この PROP-TEXT は、以下のようにリポジトリ段落のプロパティ指定子によって、Text という外部名に関連付けられた内部名です。

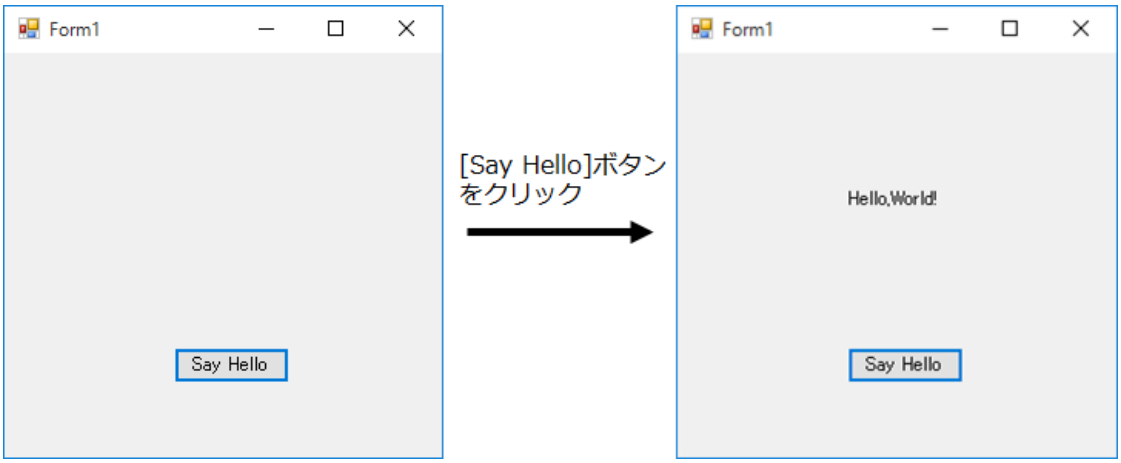
```
REPOSITORY.
    PROPERTY PROP-TEXT AS "Text"
```

IntelliSense 機能によってプロパティを展開した場合は、このプロパティ指定子が自動生成されますが、IntelliSense 機能を使わずに内部名のプロパティ名 を記述した場合は、上記のようにプロパティ指定子をリポジトリ段落に追加する必要があります。なお、PROP-TEXT などの一部の内部名は、テンプレートから作成した場合は、既に追加されています。

Windows フォームデザイナーが定義したプロパティの内部名は、 COBOL プログラムの手続きでも利用することができます。 どのプロパティが定義済みなのかは、 プロパティ指定子の定義が外部名でソートされているため、簡単に探すことができます。 なお、利用者が追加する場合は、挿入位置はリポジトリ段落のどこでも構いませんが、 デザイナーを開くことによって、プロパティの外部名で自動的にソートされます。

アプリケーションのビルドと実行

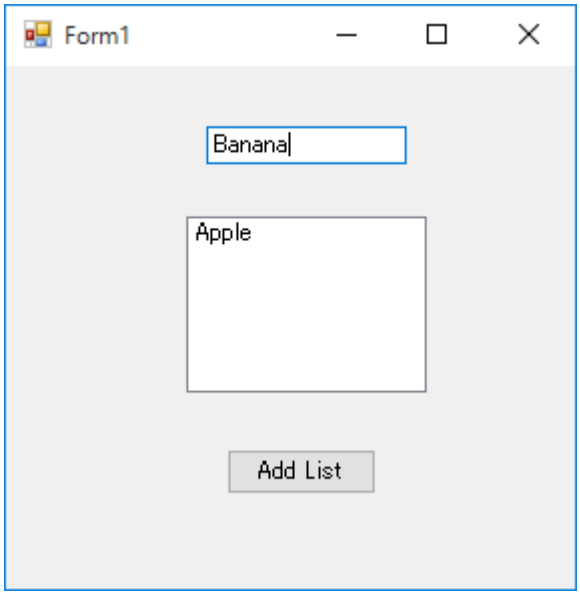
1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー配下で実行されます。
2. アプリケーションが実行されると、 [Say Hello]ボタンが貼り付けられたウィンドウが表示されます。[Say Hello]ボタンをクリックすると、 「Hello,World!」という文字列が Label コントロールを配置した位置に表示されます。



3. アプリケーションの右上の「×」ボタンを押して、Windows フォームを閉じます。

TextBox に入力された文字列を ListBox の項目に追加

ここでは TextBox に入力された文字列を、 ListBox に追加する Windows アプリケーションを作成してみましょう。



作成手順は以下のとおりです。

1. Windows フォームの作成
2. アプリケーションの手続きの記述
3. アプリケーションのビルドと実行

Windows フォームの作成

1. Visual Studio を起動し、AddList という Windows アプリケーション用プロジェクトを作成します。プロジェクト名は自由につけて構いません。
2. Windows フォームデザイナーを使って、ツールボックスからフォームに、 TextBox コントロール、Button コントロール、および ListBox コントロールを、 ドラッグして貼り付けます。
3. 貼り付けた Button コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティに、「Add List」と設定します。

アプリケーションの手続きの記述

1. 貼り付けた Button コントロールの Click イベントにイベント ハンドラを追加します。
2. 次の文をイベントハンドラの WORKING-STORAGE SECTION の直下に、A 領域から記述します。

```
01 WORK-STR OBJECT REFERENCE CLASS-STRING.
```

上記は、TextBox の Text プロパティの内容を一時的に保存する作業領域を宣言する文です。ここでは、System.String クラスのオブジェクト参照として宣言しています。 System.String クラスの詳細については、 .NET Framework のドキュメントの String クラスを参照してください。

CLASS-STRING は、TextBox コントロールを貼り付けた時に、自動で追加されます。

3. 次の文をイベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 listBox1 のプロパティ PROP-ITEMS は、 テキストエディターの IntelliSense 機能で、 一覧から Items プロパティを選択することによって、 自動的に内部名 PROP-ITEMS がリポジトリ段落と文に追加されます。

```
        SET WORK-STR TO PROP-TEXT OF textBox1.
        INVOKE PROP-ITEMS OF listBox1 "Add" USING WORK-STR.
```

上記は、textBox1 の Text プロパティの内容を、WORK-STR に設定し、 そのデータを引数にして、listBox1 の Items プロパティから ListBox.ObjectCollection の Add メソッドを呼び出す文です。ListBox 内の項目は、 ListBox.ObjectCollection というコレクションによって表されています。

コレクションについては、.NET Framework のドキュメントのコレクションの定義を参照してください。TextBox コントロール クラスの詳細については、TextBox クラスを参照してください。 ListBox コントロール クラスの詳細については、ListBox クラスを参照してください。 ListBox.ObjectCollection クラスの詳細については、ListBox.ObjectCollection クラスを参照してください。

4. 3.の手順で listBox1 のプロパティをテキストエディターの IntelliSense 機能を使わずに記述した場合は、次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的に追加されます。

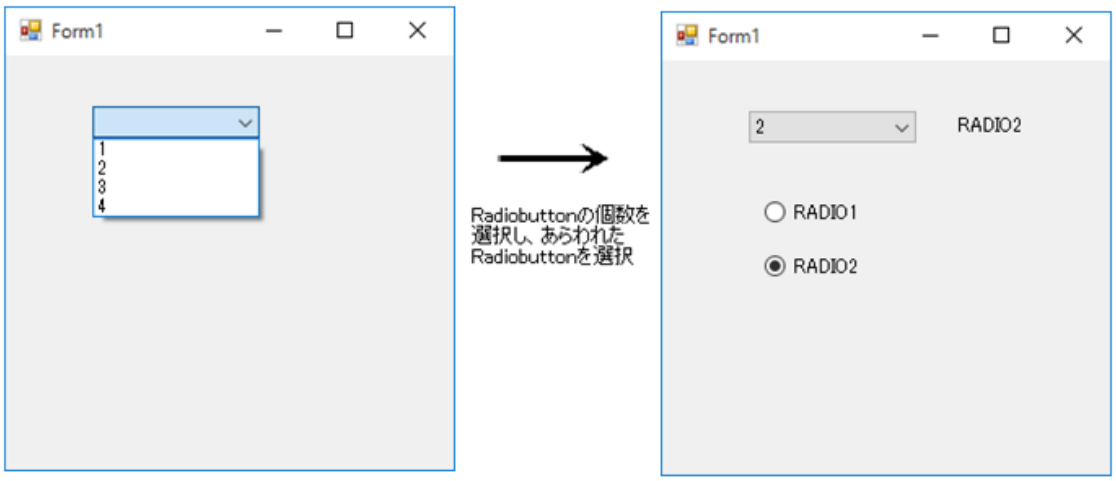
```
        PROPERTY PROP-ITEMS    AS "Items"
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。
2. TextBox に任意の文字列を入力して「Add List」ボタンを押します。 ListBox に入力した文字列が追加されるかを確認します。
3. アプリケーションの右上の「×」ボタンを押して、Windows フォームを閉じます。

コントロールの動的生成

ここではコントロールを動的に生成して利用する Windows アプリケーションを作成してみましょう。



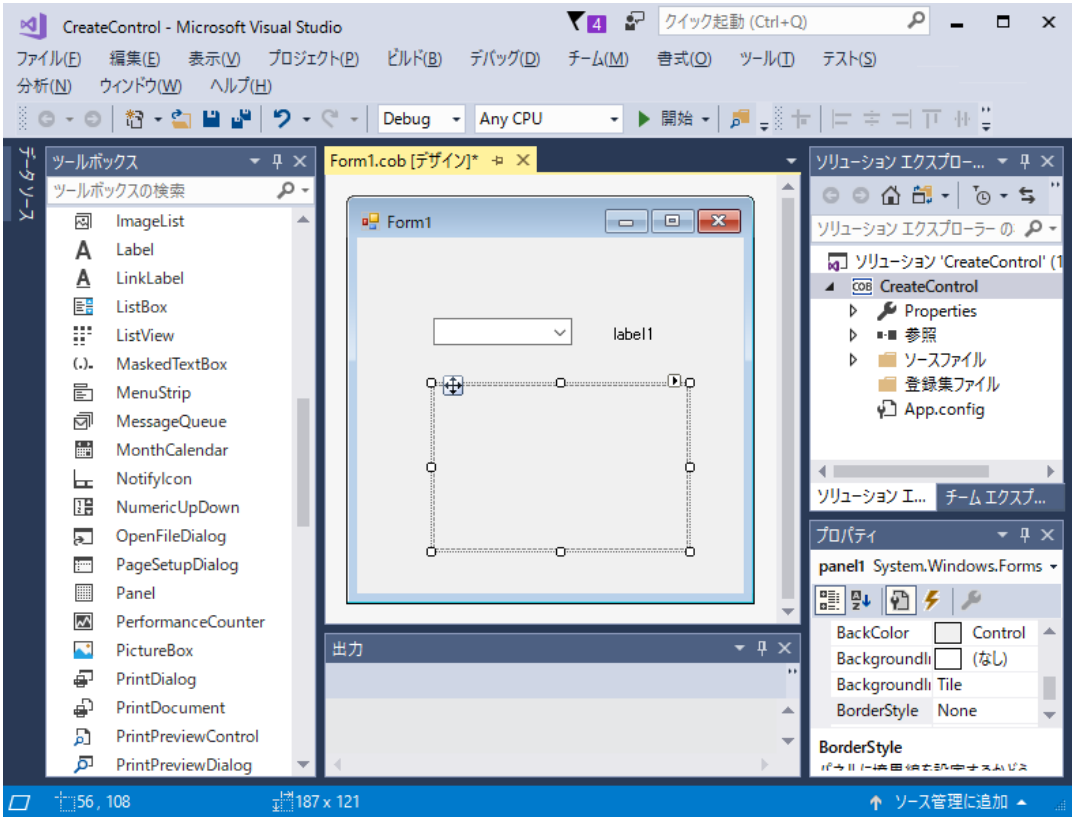
ここでは、 ComboBox コントロールで項目の個数を選択し、 その個数分の RadioButton コントロールを動的に配置する Windows アプリケーションを作成します。 配置された RadioButton は項目を選択させるために使用するため、 複数の RadioButton で 1 つのグループになるようにします。 RadioButton がクリックされてオン状態になった場合には、 RadioButton の内容を Label に表示します。 また、このチュートリアルでは、 COBOL でコレクションプロパティを利用する方法についても説明します 。

作成手順は以下のとおりです。

1. Windows フォームの作成
2. アプリケーションの手続きの記述
3. アプリケーションのビルドと実行

Windows フォームの作成

1. Visual Studio を起動し、 CreateControl という Windows アプリケーション用プロジェクトを作成します。 プロジェクト名は自由につけて構いません。
2. Windows フォームデザイナーを使って、 ツールボックスからフォームに、 ComboBox コントロール、 Label コントロール、および Panel コントロールを、 ドラッグして貼り付けます。 ComboBox と Label は、フォームの上の位置に、横に並べて配置します。 その下に Panel コントロールを大きめに配置します。 動的に生成する RadioButton は、この Panel の中に配置され、 1 つのグループとして動作するようにします。



3. 貼り付けた ComboBox コントロールをクリックして選択し、プロパティ ウィンドウで Items プロパティ(コレクション)をクリックし、 右側に現れた「...」ボタンを押します。 すると、「文字列コレクション エディター」が起動されるので、 数字の 1～4 を各行に入力します (1 行目に数字の 1、2 行目に数字の 2、という具合に入力)。 ここで入力した内容が、初期状態で ComboBox のリスト部に登録されます。

4. ComboBox の DropDownStyle プロパティをクリックし、 選択肢から DropDownList を選びます。 これによって入力フィールドを持たない ComboBox となります。

5. 貼り付けた Label コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティの内容をクリアします。

アプリケーションの手続きの記述

1. 貼り付けた ComboBox コントロールの SelectedIndexChanged イベントに イベント ハンドラを追加します。 SelectedIndexChanged イベントのイベント ハンドラは、ComboBox コントロールをダブルクリックして追加します。

2. 次の文をイベントハンドラの WORKING-STORAGE SECTION の直下に、A 領域から記述します。

```
01 WORK-NUM USAGE BINARY-LONG SIGNED.
01 WORK-CONTROL OBJECT REFERENCE CLASS-CONTROL.
01 WORK-STR OBJECT REFERENCE CLASS-STRING.
01 WORK-OBJECT OBJECT REFERENCE CLASS-OBJECT.
01 RADIOBUTTON OBJECT REFERENCE CLASS-RADIOBUTTON.
01 EVENT-HANDLER OBJECT REFERENCE DELEGATE-EVENTHANDLER.
01 RADIOBUTTON-LOCATION OBJECT REFERENCE CLASS-POINT.
01 LOCATION-X USAGE BINARY-LONG SIGNED.
01 LOCATION-Y USAGE BINARY-LONG SIGNED.
01 SIZE-HEIGHT USAGE BINARY-LONG SIGNED.
01 SIZE-MARGIN USAGE BINARY-LONG SIGNED VALUE 10.
01 IDX USAGE BINARY-LONG SIGNED.
```

3. 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。

```
        CLASS CLASS-CONTROL AS "System.Windows.Forms.Control"
        CLASS CLASS-RADIOBUTTON AS "System.Windows.Forms.RadioButton"
```

4. 次の文を SelectedIndexChanged イベントのイベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を使用して一覧からプロパティを選択することで、 外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加され、 文には内部名が挿入されます。

```
    * コントロールのレイアウト ロジックを一時的に中断
        INVOKE panel1 "SuspendLayout"

    * Panel に格納済の RadioButton がある場合、それを解放
        SET WORK-NUM TO PROP-COUNT OF PROP-CONTROLS OF panel1.
        PERFORM WORK-NUM TIMES
            SET WORK-OBJECT TO PROP-CONTROLS OF panel1 ::"get_Item" (0)
            SET RADIOBUTTON TO WORK-OBJECT AS CLASS-RADIOBUTTON
            INVOKE RADIOBUTTON "remove_CheckedChanged"
                   USING BY VALUE EVENT-HANDLER
            INVOKE PROP-CONTROLS OF panel1 "RemoveAt" USING 0
            INVOKE RADIOBUTTON "Dispose"
        END-PERFORM.

    * Panel に RadioButton を追加
        SET WORK-NUM TO PROP-SELECTEDINDEX OF comboBox1.
        ADD 1 TO WORK-NUM.
        MOVE 10 TO LOCATION-X.
        MOVE 10 TO LOCATION-Y.
        MOVE 1 TO IDX.
        PERFORM WORK-NUM TIMES
            *> RadioButton のインスタンスを生成
            SET RADIOBUTTON TO CLASS-RADIOBUTTON ::"NEW"
            *> RadioButton の座標を設定
            SET RADIOBUTTON-LOCATION TO CLASS-POINT ::"NEW"
                   (LOCATION-X LOCATION-Y)
            SET PROP-LOCATION OF RADIOBUTTON TO RADIOBUTTON-LOCATION
            *> RadioButton の Text を設定
            *> (ここでは System.String の Concat メソッドを使って文字列を作成)
            SET WORK-STR TO CLASS-STRING ::"Concat" (N"RADIO" IDX)
            SET PROP-TEXT OF RADIOBUTTON TO WORK-STR
            *> 次の RadioButton の Y 座標を計算
            SET SIZE-HEIGHT TO PROP-HEIGHT OF PROP-SIZE OF RADIOBUTTON
            COMPUTE LOCATION-Y = LOCATION-Y + SIZE-HEIGHT + SIZE-MARGIN
            *> Panel に RadioButton を格納
            INVOKE PROP-CONTROLS OF panel1 "Add" USING RADIOBUTTON
            *> RadioButton の CheckedChanged イベントに、イベント ハンドラを追加
            INVOKE DELEGATE-EVENTHANDLER "NEW"
                   USING BY VALUE SELF
                         BY VALUE N"radioButton_CheckedChanged"
                   RETURNING EVENT-HANDLER
            INVOKE RADIOBUTTON "add_CheckedChanged"
                   USING BY VALUE EVENT-HANDLER
            *> RadioButton の Text に利用する IDX をインクリメント
            ADD 1 TO IDX
        END-PERFORM

    * RadioButton グループの先頭の TabStop を True に設定
        IF WORK-NUM > 0 THEN
            SET WORK-OBJECT TO PROP-CONTROLS OF panel1 ::"get_Item" (0)
            SET RADIOBUTTON TO WORK-OBJECT AS CLASS-RADIOBUTTON
            SET PROP-TABSTOP OF RADIOBUTTON TO B"1"
        END-IF

    * コントロールのレイアウト ロジックを再開
        INVOKE panel1 "ResumeLayout".
```

上記の手続きは、panel に追加済の RadioButton を解放し、 新たに ComboBox で選択された個数の RadioButton を Panel に追加する手続きです。 また、RadioButton のチェック状態が変更された場合に、 RadioButton_CheckedChanged イベント ハンドラ を呼び出すように デリゲートを使用しています。 さらに、RadioButton の生成直後でも、 ComboBox にフォーカスがある状態から、 Tab キー

によってフォーカスが移動できるように、RadioButton グループの先頭の TabStop プロパティを True に設定しています。

コントロールは、 Control.ControlCollection オブジェクトのメンバとして管理されるため、Control.ControlCollection の Add メソッドによって追加します。Control.ControlCollection は、Form、Panel、GroupBox クラスなどの、Control プロパティとして参照できます。 Control.ControlCollection 内のコントロールは、 Control.ControlCollection の Item プロパティによって参照できます。 Item プロパティはインデックス付きプロパティ(インデクサ)として提供されており、 NetCOBOL for .NET では、プロパティ名の前に "get_" または "set_" を付加したメソッドとして INVOKE 文を使ってアクセスする必要があります。 次のような INVOKE 文によってアクセスします。

```
INVOKE control1 "get_Item" USING BY VALUE WORK-INDEX RETURNING object1
SET object2 TO object1 AS class-object2
```

なお、上記の INVOKE 文は、 メソッドの行内呼出しを使って次のように記述することもできます。

```
SET object1 TO control1 ::"get_Item" (WORK-INDEX)
SET object2 TO object1 AS class-object2
```

RadioButton クラスについては、 .NET Framework のドキュメントの RadioButton クラスを参照してください。 Panel クラスについては、Panel クラスを参照してください。 Control.ControlCollection クラスについては、Control.ControlCollection クラスを参照してください。 インデックス付きプロパティ(インデクサ)の使い方については、インデクサを利用するを参照してください。

5. 4.の手順でコントロールのプロパティをテキストエディターの IntelliSense 機能を使わずに記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的にリポジトリ段落に追加されます。

```
PROPERTY PROP-COUNT AS "Count"
PROPERTY PROP-HEIGHT AS "Height"
PROPERTY PROP-SELECTEDINDEX AS "SelectedIndex"
PROPERTY PROP-TABSTOP AS "TabStop"
```

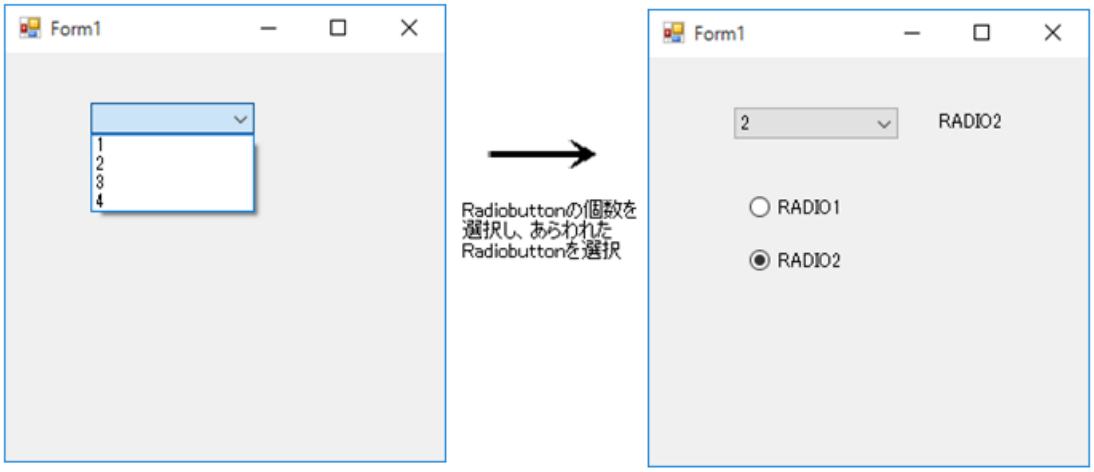
6. 次のメソッド定義の手続きを、END OBJECT の直前に挿入します。

```
METHOD-ID. radioButton_CheckedChanged PRIVATE.
DATA DIVISION.
LINKAGE SECTION.
01 SENDER OBJECT REFERENCE CLASS-OBJECT.
01 E OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE SENDER E.
   SET PROP-TEXT OF label1 TO PROP-TEXT OF SENDER AS CLASS-RADIOBUTTON.
END METHOD radioButton_CheckedChanged.
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。

2. ComboBox の ドロップダウン矢印をクリックし、RadioButton の個数を選択します。 表示された RadioButton を選択して、 Label に RadioButton の Text が設定されるかを確認します。



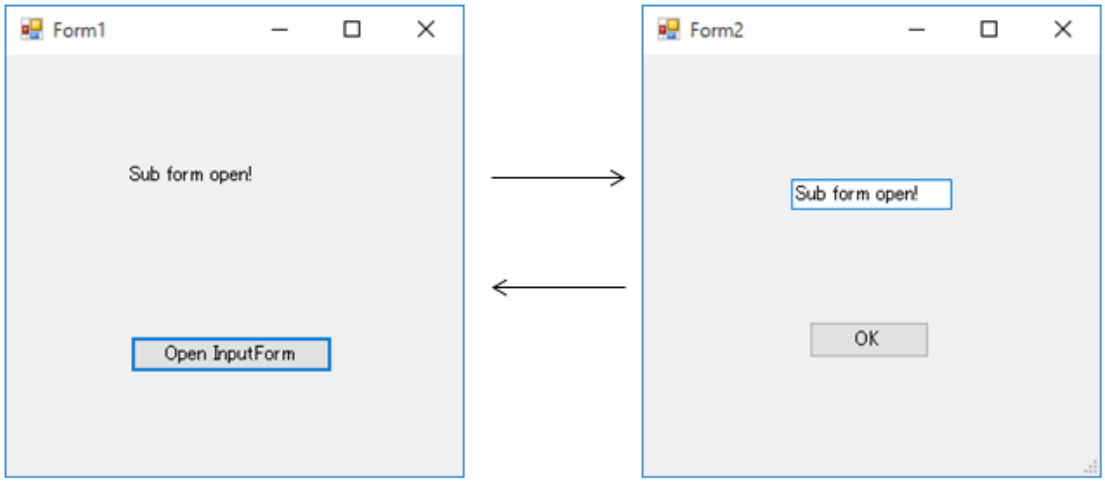
3. アプリケーションの右上の「×」ボタンを押して、Windows フォームを閉じます。

NCOBU00016

別フォームとの連携

ここでは Windows フォームを複数使った Windows アプリケーションを作成してみましょう。

このチュートリアルでは、 主フォーム上の Button をクリックして入力用の副フォームを開き、 入力用フォームに入力された文字列を呼び出し元のフォームに通知します。



作成手順は以下のとおりです。

1. 主 Windows フォームの作成
2. 副 Windows フォームの作成
3. アプリケーションの手続きの記述
4. アプリケーションのビルドと実行

主 Windows フォームの作成

1. Visual Studioを起動し、CreateFormというWindows アプリケーション用プロジェクトを作成します。 ここで付けた名前は、 以降の手順で記述する手続きの中で使用するため注意してください。
2. Windows フォームデザイナーを使って、 ツールボックスからフォームに、 Button コントロールおよび Label コントロールをドラッグして貼り付けます。
3. 貼り付けた Button コントロールをクリックして選択し、 プロパティ ウィンドウで Text プロパティの内容を「Open InputForm」と設定します。
4. 貼り付けた Label コントロールをクリックして選択し、 プロパティ ウィンドウで Text プロパティの内容をクリアします。

副 Windows フォームの作成

1. ソリューション エクスプローラー を開き、 ソースファイル ノードを右クリックして、コンテキストメニューから [追加]-[新しい項目]を選択します。
2. [新しい項目の追加]ダイアログが表示されたら、 テンプレートから、Windows フォームを選択し、 ファイル名に「Form2.cob」を入力して、[追加]ボタンをクリックします。 これで二つ目の Windows フォームが追加されました。
3. Windows フォームデザイナーを使って、 ツールボックスから副フォームに、 TextBox コントロールおよび Button コントロールをドラッグして貼り付けます。

4. 貼り付けた Button コントロールをクリックして選択し、 プロパティ ウィンドウで Text プロパティの内容を「OK」と設定します。

アプリケーションの手続きの記述

1. 副フォームに貼り付けた Button コントロールの Click イベントに イベント ハンドラを追加します。
2. 次の文をイベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。

```
        SET InputText TO PROP-TEXT OF textBox1.
        INVOKE SELF "Close"
```

   上記は、TextBox の内容をプロパティとして公開する InputText に設定し、 副フォームを閉じるメソッドを呼び出す手続きです。

3. 次の文を OBJECT 段落の WORKING-STORAGE の直下に、A 領域から記述します。

```
    01 InputText OBJECT REFERENCE CLASS-STRING PROPERTY.
```

   この文によって、主フォームから参照可能なプロパティが、 副フォームに定義されます。

4. 主フォームに貼り付けた Button コントロールの Click イベントに イベント ハンドラを追加します。
5. 次の文をイベントハンドラの WORKING-STORAGE SECTION の直下に、A 領域から記述します。

```
    01 INPUTFORM OBJECT REFERENCE CLASS-INPUTFORM.
```

6. 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。

```
        CLASS CLASS-INPUTFORM AS "CreateForm.Form2"
```

   CLASS 指定子の AS の右側に記述する定数は、 副フォームの CLASS-ID に指定されている外部名を指定します。 もしプロジェクト名や副フォーム名に、手順と異なる名前を指定した場合には、 上記の CLASS 指定子の定数を変更してください。

7. 次の文をイベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を使用して一覧からプロパティを選択することによって、 外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加され、 文には内部名が挿入されます。

```
        SET INPUTFORM TO CLASS-INPUTFORM::"NEW".
        INVOKE INPUTFORM "ShowDialog".
        SET PROP-TEXT OF label1 TO PROP-INPUTTEXT OF INPUTFORM.
        INVOKE INPUTFORM "Dispose".
```

   上記は、副フォームのインスタンスを生成し、 副フォームのオブジェクトの ShowDialog メソッドを呼び出す文です。 ShowDialog メソッドは、副フォームが継承している、 System.Windows.Forms.Form クラス が実装しています。 ShowDialog メソッドは、副フォームが閉じるまで返ってきません。 副フォームが閉じると、副フォームで入力した文字列を表すプロパティを参照して Label に設定します。その後、副フォームを破棄します。

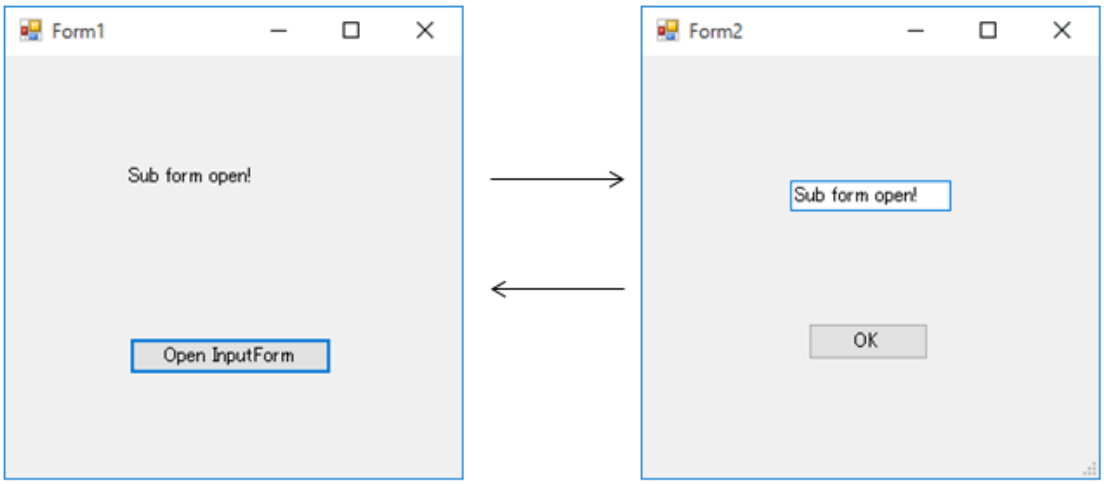
8. 7.の手順でプロパティをテキストエディターの IntelliSense 機能を使わずに記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 IntelliSense 機能を使ってプロパティを挿入した場合、 次の文はリポジトリ段落に自動的に追加されます。

```
        PROPERTY PROP-INPUTTEXT AS "InputText"
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。

2. 「Open InputForm」ボタンをクリックし、副フォームが開くか確認します。 その後、副フォームの TextBox に任意の文字列を入力し、 副フォームの「OK」ボタンをクリックし、副フォームを閉じて、 主フォームの Label に入力した文字列が表示されるかを確認します。



3. アプリケーションの右上の「×」ボタンを押して、Windows フォームを閉じます。

Windows アプリケーションの開発のためのヒント

NetCOBOL for .NET チュートリアルの Windows アプリケーションの開発の学習によって、簡単な Windows アプリケーションは構築できるようになります。そして、Windows アプリケーション構築には、.NET Framework クラスライブラリを活用する必要があることが、 おわかりいただけるでしょう。

より高度な Windows アプリケーションを構築するには、 さらに.NET Framework クラスライブラリ、特に、System.Windows.Forms 名前空間で提供されている機能を理解する必要があります。

Visual Studio には、 多数のクラスライブラリについてのドキュメントがありますが、 ヘルプの検索機能を使って、効率よく必要な情報を見つけ出すことが重要です。 そのためには、.NET Framework クラス ライブラリおよび Windows アプリケーションの概念を理解する必要があります。 そして、.NET Framework クラスライブラリを使用したプログラミングを数多くこなすことで、 より効率よく必要な情報を見つけ出すことができるようになるでしょう。

NCOBU00018

ASP.NET Web アプリケーションの開発

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。

NetCOBOL for .NET と ASP.NET を組み合わせることで、 Web ブラウザからアクセスする、Web ベースのアプリケーションを構築することができます。

ここでは ASP.NET Web アプリケーション(ASP.NET Web サイトとも呼ばれる)の概要、 ASP.NET Web ページ(Web フォームとも呼ばれる)を利用した ASP.NET Web アプリケーションの開発手順について説明します。

このセクションの内容

ASP.NET Web アプリケーションの概要

ASP.NET Web アプリケーションの概要について説明します。

ASP.NET Web アプリケーションのプログラム構造

ASP.NET Web アプリケーションのプログラム構造について説明します。

ASP.NET Web ページの処理

ASP.NET Web アプリケーションのイベント処理の流れと ASP.NET Web ページの処理の注意事項 について説明します。

ASP.NET Web ページの状態管理

ASP.NET Web ページの状態管理について説明します。

ASP.NET Web アプリケーションの開発手順

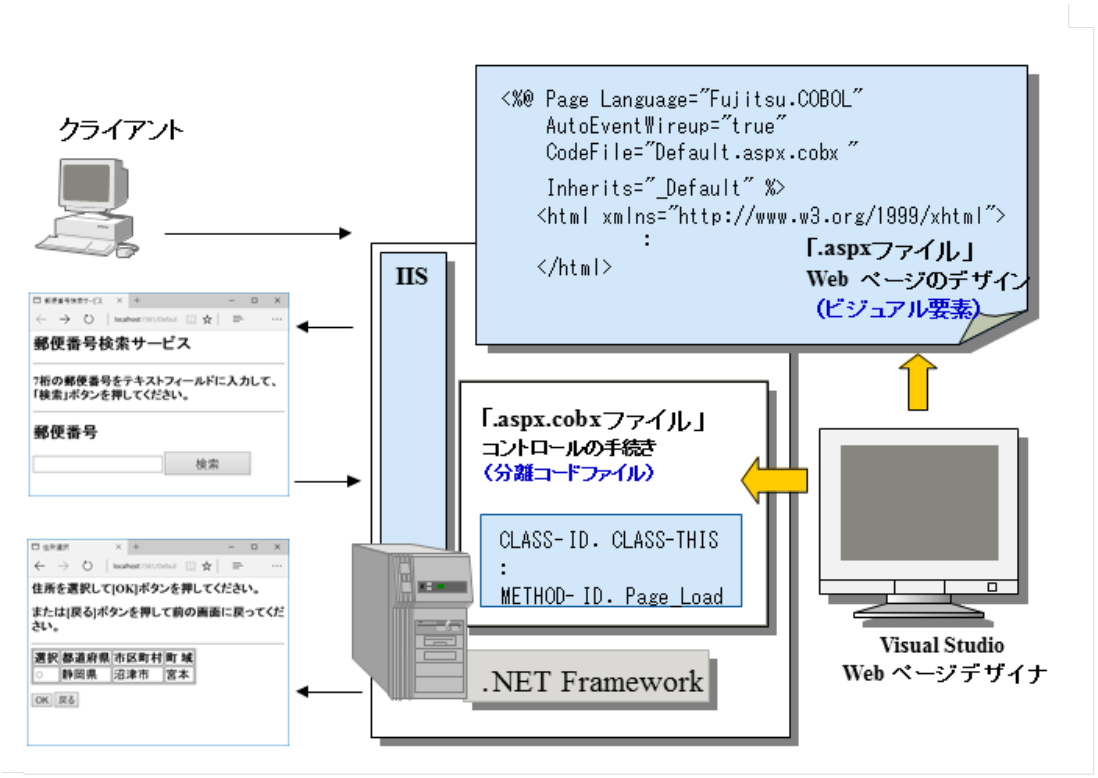
ASP.NET Web アプリケーションの開発手順について説明します。

ASP.NET Web アプリケーション開発時の注意事項については、ASP.NET アプリケーション開発の注意事項を参照してください。

ASP.NET Web アプリケーションの概要

ASP.NET は、Web アプリケーションを開発および実行するためのプラットフォームであり、.NET Framework の一部です。ASP.NET Web アプリケーション(ASP.NET Web サイトとも呼ばれる)は、ASP.NET のページフレームワークなどの機能を使って作成します。

ASP.NET Web アプリケーションの動的コンテンツを作成するための技術として ASP.NET Web ページ(Web フォーム ページ とも呼ばれる)があります。 ブラウザから ASP.NET Web ページを要求すると、そのページは Web サーバ上でプログラムとして実行されます。 プログラムの実行によって動的に生成された Web ページの出力が、ブラウザに送信されます。



NetCOBOL for .NET では、Visual Studio を使用して ASP.NET Web アプリケーションを開発することができます。

ASP.NET Web ページは、 Web ページ デザイナーを使って、画面をデザインします。 Web ページ デザイナーは、Button の位置やサイズなどのプロパティの設定手続きを自動的に生成するため、 これらのコントロールをドラッグ&ドロップでページにレイアウトするだけで、 簡単に画面をデザインできます。

新規に ASP.NET Web ページを作成すると、ユーザインタフェース要素を保持する.aspx ファイル(コントロール要素や HTML 要素を含んだテキストファイル)と、 ボタンが押されたときの手続きを保持する.aspx.cobx ファイルが作成されます。

ASP.NET Web ページは Page クラスから派生し、 ASP.NET のページフレームワーク上に構築されます。 実行時には、このページオブジェクト内に、 Button や TextBox などの ASP.NET Web サーバー コントロールが格納されます。

参考

Visual Studio による ASP.NET Web アプリケーションのプログラミング方法についての詳細は、ASP.NET Web ページのプログラミングなどを参照してください。

NCOBU00020

ASP.NET Web アプリケーションのプログラム構造

ここでは、ASP.NET Web アプリケーション (ASP.NET Web サイト とも呼ばれる)の COBOL プログラムの構造について説明します。 プロジェクトを作成する時に、 [ASP.NET Web サイト]のテンプレートを選択すると、自動的に ASP.NET Web ページ (Web フォーム ページ とも呼ばれる)を構成する以下のファイルが作成されます。

.aspx.cobx ファイル	コード (ビジネスロジック)
.aspx ファイル	レイアウト (コントロールの配置)

ASP.NET Web ページには、 コードとレイアウトを一つの.aspx ファイルに記述する「単一ファイル ページ」と .aspx ファイルと.aspx.cobx ファイルの二つのファイルに分離して記述する「分離コード ページ」 の二種類があります。 Visual Web Developer を使って ASP.NET Web ページ を作成する場合は、「分離コード ページ」となります。このチュートリアルでは、「分離コード ページ」 を対象にして説明します。

「単一ファイル ページ」と「分離コード ページ」の違いについては、ASP.NET Web ページのコード モデルを参照してください。

.aspx.cobx ファイル

.aspx.cobx ファイルには、発生したイベントに対するビジネスロジックを記述します。

ASP.NET Web アプリケーションの COBOL ソースプログラムの構造は、 以下のようになります。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "_Default" INHERITS CLASS-PAGE IS PARTIAL.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.

SPECIAL-NAMES.

REPOSITORY.
    CLASS  CLASS-PAGE AS "System.Web.UI.Page"
    .

OBJECT.
DATA DIVISION.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".

イベント ハンドラ メソッド定義

その他のメソッド定義

END OBJECT.

END CLASS CLASS-THIS.
```

Windows アプリケーションの構造とほとんど変わりませんが、 以下の点が異なります。

- ASP.NET Web アプリケーションは、 ASP.NET から利用されるクラスライブラリとして構築されるため、 アプリケーションのエントリはありません。
- ASP.NET Web ページが読み込まれた時に Page オブジェクトが生成され、 それと同時に実行される Page_Load イベントハンドラのメソッド定義が、 既定で生成されます。

上記以外のクラス定義の内容については、Windows アプリケーションのプログラム構造を参照してください。

.aspx ファイル

.aspx ファイルは、ページレイアウトを保持します。

.aspx ファイルは、.aspx.cobx ファイルに記述した COBOL のクラスから派生します。 ユーザが、 ASP.NET Web ページを要求した時に、 ASP.NET Web ページのコントロールを HTML に変換して表示します。

COBOL のクラスと.aspx との関連付けは、以下のように.aspx ファイルの@ Page ディレクティブの CodeFile 属性によって行なわれます。また、.aspx.cobx ファイルによって定義されているクラスの外部名を Inherits 属性によって指定することで、そのクラスから派生しているページであることを示します。

```
<%@ Page Language="Fujitsu.COBOL" AutoEventWireup="true"
    CodeFile="Default.aspx.cobx" Inherits="_Default" %>
```

.aspx.cobx ファイルで定義したクラスのクラス名を変更した場合、 .aspx ファイルの @ Page ディレクティブの Inherits に指定されたクラス名は自動的に変更されないため、手動で変更する必要があります。

NCOBU00021

ASP.NET Web ページの処理

ASP.NET Web アプリケーション (ASP.NET Web サイトとも呼ばれる) のイベント処理の流れと ASP.NET Web ページ (Web フォーム ページとも呼ばれる)の処理の注意事項について説明します。

前提知識

- イベントを発生させる元となる GUI 部品(テキストボックスやコンボボックス、コマンドボタンなど)を「ASP.NET Web サーバー コントロール」と呼びます。 また、これらの「ASP.NET Web サーバー コントロール」を配置するページのことを 「ASP.NET Web ページ(Web フォーム ページとも呼ばれる)」と呼びます。 Visual Studio では「Web ページ デザイナー」を用いて、この ASP.NET Web ページ上に ASP.NET Web サーバー コントロールを配置することができます。
- Windows アプリケーションやクライアントサイドの Web アプリケーションでは、イベントはクライアントで発生し、クライアント上で処理されます。 ASP.NET Web アプリケーションでは、 イベントはクライアントで発生し、サーバ側で処理されます。

ASP.NET Web アプリケーションのイベント処理の流れ

イベントの処理の流れは以下のようになります。

1. クライアント(ユーザ)がサーバにページを要求する。
2. サーバは、 ASP.NET Web ページのコントロールを HTML に変換して表示する。(ページの表示)
3. ユーザは、表示されたページに対して、文字の入力や ボタンを押すなどの操作を行なう。(イベントの発生)
4. 「ボタンが押された」という情報とユーザの操作によって入力されたフォームの内容が リクエストとしてサーバに送信(ポスト)される。
5. サーバで、そのイベントに対する処理が実行され、 処理の結果をレスポンスとしてクライアントに送信する。

イベントが発生した時に、 実行がサーバとクライアントの間を行き来する「リクエスト」から「レスポンス」までのシーケンスを、 「ラウンドトリップ」と 呼びます。

サーバコントロールをクリックするとクリックイベントが発生し、 ラウンドトリップを行なうために、フォームの情報がサーバに送信(ポスト)されます。 これを「ポストバック」と 呼びます。

ASP.NET Web ページの処理の注意事項

- ASP.NET Web アプリケーションでは、 サーバとのラウンドトリップが行なわれます。 Windows アプリケーションと違い、頻繁なラウンドトリップは、 パフォーマンスに著しく影響します。 そのため、マウスがコントロールの上を通過したなどのイベント(DHTML での onmouseover に相当)のような頻繁に発生するイベントは、 サーバコントロールではサポートされていません。
- サーバコントロールには、クリックイベントの他にも、 テキストの内容が変更されたことをあらわすイベントなど、 いくつかのイベントが用意されていますが、 基本的にこれらのイベントが発生した時には「ポストバック」はされません。 ポストバックされないイベント(非ポストバックイベント)は、 ポストバックされるイベントが発生するまで貯えられます。 貯えられたイベントは、ポストバックイベントが発生した時に、 ポストバックイベントより前に処理されますが、その順番は不定です。 そのため、非ポストバックイベントは発生順序に依存するロジックを記述できません。
- ページはラウンドトリップごとに再作成されます。 サーバはクライアントへの送信を完了すると、ページ情報を破棄し、 サーバリソースを解放します。 これによってサーバは多数のユーザをサポートすることが可能になります。 多数のユーザとラウンドトリップ間で情報をやり取りする場合、 特別な処理が必要になります。 ASP.NET Page Framework には、 情報を保存(状態管理を実現)するためのオプションが用意されています。このオプションについては ASP.NET Web ページの状態管理を参照してください。

- ページが再作成される場合、ページがロードされた時のイベントが、 最初のページの要求とポストバックによる要求の両方で発生することになりますが、 ASP.NET Page Framework では、これらの要求を区別することができるため、 状況に応じたページの初期化コードを記述することができます。

NCOBU00022

ASP.NET Web ページの状態管理

ASP.NET Web ページ の状態管理オプションには、 大きく分けてクライアントベースの状態管理オプションとサーバベースの状態管理オプションがあります。

クライアントベースの状態管理では、 ページ内またはクライアントコンピュータ上に情報を格納します。 サーバには情報が格納されません。 サーバベースの状態管理では、サーバに情報が格納されます。

Web サーバコントロールのプロパティは、クライアントベースの状態管理オプションであるビューステートによって、 状態を自動的に保存することができます。 ただし、Web サーバコントロールやオブジェクトを動的に生成するようなケースでは、 それらのオブジェクトをラウンドトリップ間で再作成しなければならないため、 別途情報を保存する必要があります。

状態管理の詳細については、Visual Studio のドキュメントの ASP.NET の状態管理の概要を参照してください。

ASP.NET Web アプリケーションの開発手順

ここでは、簡単なサンプルを例にして、 Visual Studio で、 ASP.NET Web アプリケーションを開発する場合の手順について説明します。

説明にしたがってアプリケーションを構築することによって、 ASP.NET Web アプリケーションの開発方法の基本を学習することができます。

なお、本チュートリアルでは、ファイル システム Web サイトを使用します。 この場合、IIS(Microsoft Internet Information Services)の設定は必要ありません。 Web サイトの種類については、Visual Studio のドキュメントの Visual Web Developer における Web サイトの種類を参照してください。

このセクションの内容

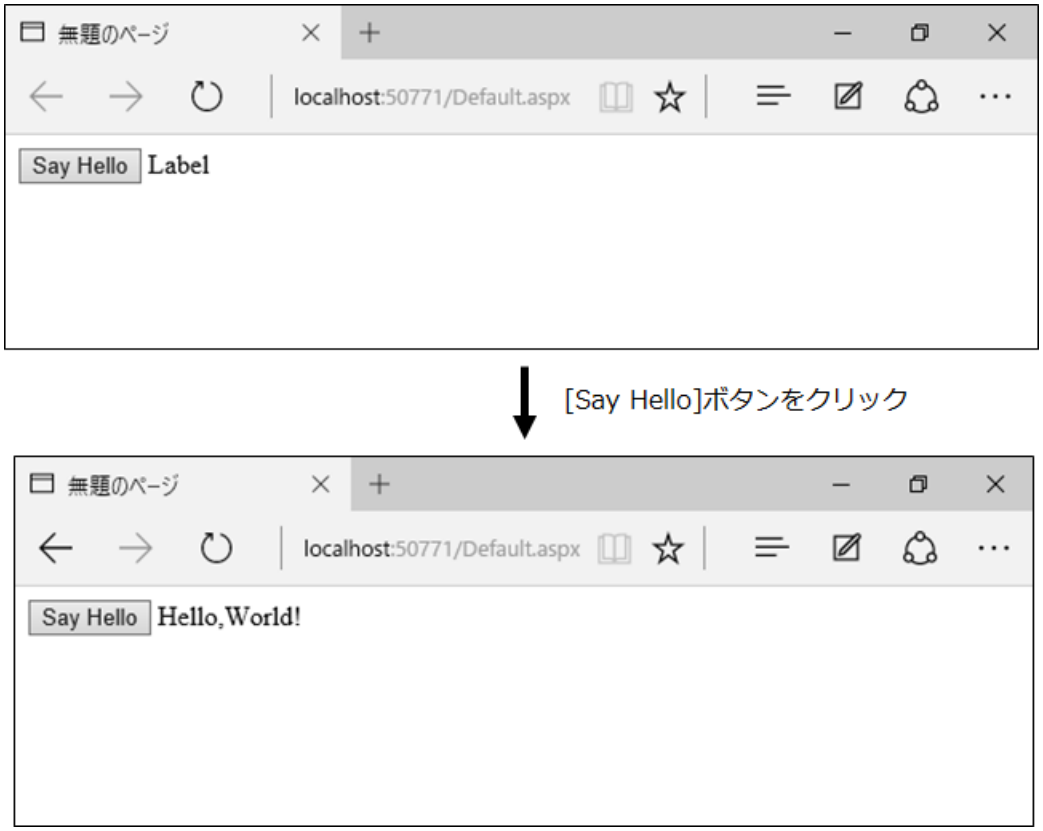
NCOBU00024

ラベルに文字列を設定する

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。

NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ (http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

ここでは、ボタンをクリックすると「Hello,World!」という文字列を表示する 簡単な Web アプリケーションを作成してみましょう。



作成手順は以下の とおりです。

1. Web サイトの作成
2. Web フォーム(ASP.NET Web ページ)にボタンとラベルを貼り付ける
3. アプリケーションの手続きの記述

Web サイトの作成

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[Web サイト]を選択します。
3. [新しい Web サイト]ダイアログボックスが表示されます。
4. 左ペインで、[NetCOBOL for .NET]を選択します。
5. 右ペインで、[ASP.NET Web サイト]を選択します。
6. [Web の場所]ボックスは、[ファイル システム]を選択し、 Web サイトを置くフォルダのパスを指定します。 ここでは、「c:¥WebSite¥HelloWorld」とします。
7. [OK]ボタンをクリックします。これで、Web サイトの構築に必要なファイル一式が作成されます。Web サイトが作成されると、Web ページ デザイナーが自動的に起動されます。

   ソリューションエクスプローラーには、 Web サイトの構成をあらわすツリーが表示されます。 ソリューションエクスプローラーが表示されていない場合は、Visual Studio の [表示]メニューから[ソリューション エクスプローラー]を選択します。

Web フォーム(ASP.NET Web ページ) にボタンとラベルを貼り付ける

Web ページ デザイナー を使って、 Button コントロールと Label コントロールをフォームに貼り付けます。

1. Web ページ デザイナー ウィンドウの下にある [デザイン]タブをクリックして、デザイン ビューに切り替えます。
2. ツールボックスを表示させ、 コントロールの一覧から「Button」アイコンをクリックします。 その後、マウスカーソルをフォームの上に移動させ、 ドラッグして配置します。 ツールボックスは、Visual Studio の[表示]メニューから[ツールボックス]を選択することで表示されます。
3. ボタンの表面には「Button」という文字列が表示されます。 これを「Say Hello」という文字列に変更します。 ボタン表面の文字列を変更する場合は、Visual Studio の[プロパティ] ウィンドウ を使って、Button コントロールの Text プロパティに「Say Hello」を設定します。 [プロパティ] ウィンドウは、Visual Studio の[表示]メニューから[プロパティ ウィンドウ] を選択することで開くことができます。 [プロパティ] ウィンドウ が開いているときに、フォームに貼り付けた Button コントロールをクリックすると、 Button コントロールのプロパティが表示されます。
4. 次に、ツールボックスから「Label」アイコンをクリックし、 テキストを表示する場所に配置します。 Label コントロールは、ユーザが編集できないテキストを表示する目的で使用します。 このプログラムでは「Say Hello」ボタンをクリックした時に、 テキストを表示する位置に Label コントロールを配置します。

アプリケーションの手続きの記述

フォーム上のボタンをダブルクリックすると、Click イベントが生成されます。 Click イベントは、 Click イベントに割り当てられた手続き(イベントハンドラといいます)を実行します。

ここでは、[Say Hello]ボタンがクリックされた時に Label コントロールの位置にテキストを表示させるための手続きを、 Click イベントに追加します。

1. Button コントロールをダブルクリックして、 COBOL ソースプログラムを編集するためのコードエディターを開きます。

   この操作で、イベントハンドラの挿入位置にカーソルが移動します。 この場合は、Button1_Click メソッドの手続き部の入力位置に移動します。
2. 以下の SET 文を挿入します。 この SET 文は、Label コントロールのクラスインスタンスが格納されている Label1 オブジェクトの Text プロパティに、 文字列"Hello,World!"を設定します。

```
METHOD-ID. Button1_Click.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 PARAM-SENDER OBJECT REFERENCE CLASS-OBJECT.
01 PARAM-E OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE PARAM-SENDER BY VALUE PARAM-E.
     SET PROP-TEXT OF Label1 TO N"Hello,World!".
END METHOD Button1_Click.
```

この時、「SET Label1」と入力してから、IntelliSense 機能のメンバーの一覧を利用して"Text"プロパティを選択すると、 ソースプログラム上に「SET PROP-TEXT OF Label1」が展開されます。IntelliSense 機能を利用してプロパティを参照すると、 自動的にリポジトリ段落にも参照したプロパティのプロパティ指定子が追加されます。

文は、B 領域(12 カラム以降)から記述します。 テンプレートの [ASP.NET Web サイト]を利用して生成された COBOL ソースプログラムには翻訳オプション NOALPHAL が指定されるため、 利用者語の英大小文字を区別します。 ただし、通常は区別されないため、 同じ名前で英大小文字だけが異なるクラス名やメソッド名を作成しないように注意してください。

3. 上の処理で、IntelliSense 機能を使わずにプロパティを参照した場合、 次の文をリポジトリ段落に手動で追加する必要があります。

```
REPOSITORY.
   PROPERTY PROP-TEXT AS "Text"
```

このプロパティ指定子は、Text というプロパティ名(外部名)を、 PROP-TEXT という名前(内部名)に関連付けます。

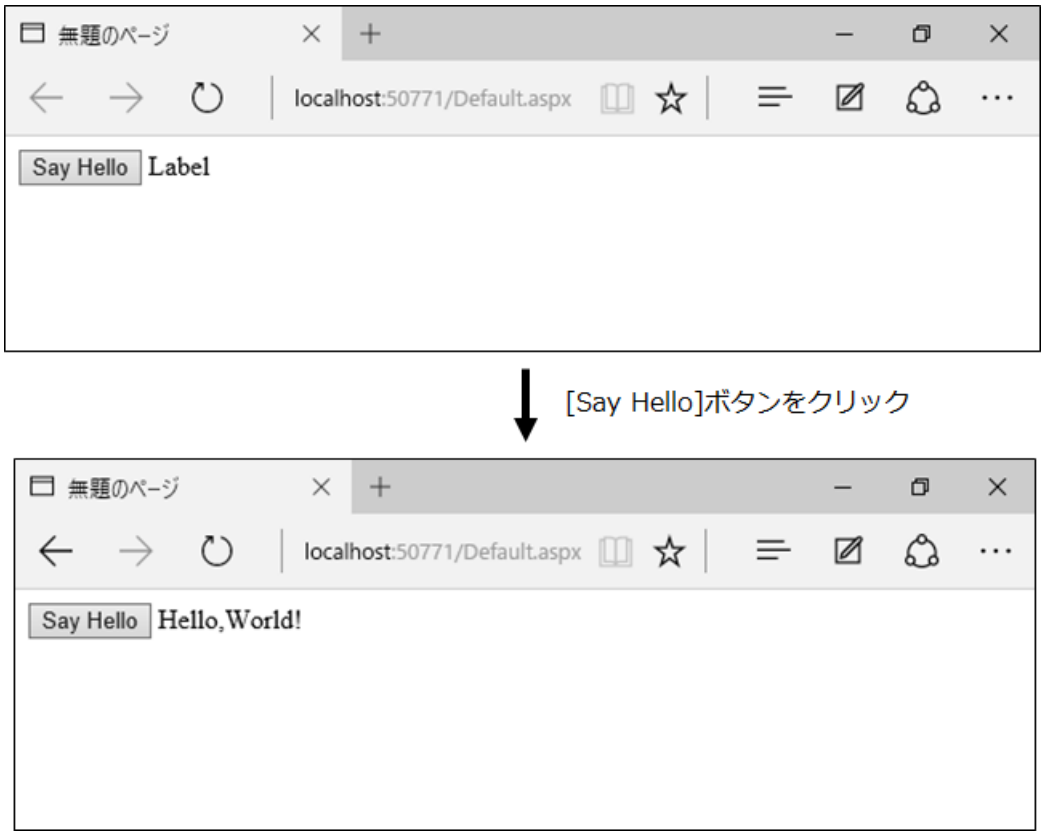
プロパティ指定子の挿入位置は、リポジトリ段落のどこでもかまいません。 ただし、リポジトリ段落の最終行に挿入した場合、ピリオドの位置に注意が必要です。

AS 指定の書き方については、 COBOL 文法書の「11.3.4 外部名と内部名」および「11.6.1.3 リポジトリ段落」 を参照してください。

4. COBOL ソースプログラム(Default.aspx.cobx)の変更を保存します。 保存する場合、ツールバーの [Default.aspx.cobx の保存]アイコン( )をクリックします。

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー配下で実行されます。 Web ブラウザは自動的に起動されます。
2. アプリケーションが実行されると、[Say Hello]ボタンが貼り付けられたブラウザが表示されます。[Say Hello]ボタンをクリックすると、 「Hello,World!」という文字列が Label コントロールを配置した位置に表示されます。



3. Web ブラウザを閉じて、アプリケーションを終了します。

NCOBU00025

ページの状態の保存

ASP.NET Web ページの処理で説明したように、ページはラウンド トリップごとに再作成されます。そのため、実行時に設定された情報は基本的にすべて消えてしまいます。 そこでページの情報を何らかの方法で保存する必要があります。たとえばページの中に複数の入力用パネルがあり、カレントの入力パネルを特定するために、パネルのインデックスを保存するといったケースも考えられます。 ここでは、ビューステートを使用して情報を保存する例として、 ボタンが押された回数をカウントするサンプルアプリケーションを作成してみましょう。

作成手順は以下のとおりです。

1. Web フォーム (ASP.NET Web ページ) の作成
2. アプリケーションの手続きの記述
3. プリケーションのビルドと実行

Web フォーム (ASP.NET Web ページ) の作成

1. ラベルに文字列を設定するの[Web サイトの作成]と同様の操作をして、 Count という ASP.NET Web アプリケーション用 Web サイトを作成します。 Web サイト名は自由につけて構いません。
2. Web ページ デザイナー を使って、 ツールボックスから Web サーバ コントロールの Label コントロールと Button コントロールを、 フォームに貼り付けます。
3. 貼り付けた Button コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティに、「COUNT」と設定します。
4. 貼り付けた Label コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティの内容をクリアします。

アプリケーションの手続きの記述

1. ページが最初に要求されたときにビューステートの初期値を設定するため、 Page の Load イベントにイベント ハンドラを追加します。

2. 次の文をイベントハンドラの DATA DIVISION の直下に、A 領域から記述します。

```
WORKING-STORAGE SECTION.
01 WORK-VALUE USAGE BINARY-LONG SIGNED VALUE 0.
```

3. 次の文を Page の Load イベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を利用することで、 プロパティの外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加されます。

```
IF PROP-ISPOSTBACK OF SELF = B"0" THEN
   INVOKE PROP-VIEWSTATE OF SELF "Add"
         USING BY VALUE N"COUNT" BY VALUE WORK-VALUE
END-IF
```

   上記は、ページが最初に要求されたとき(Page の IsPostBack が False)に、 ビューステートに WORK-VALUE の内容を保存する文です。 ビューステートを識別する名前には「COUNT」という文字列を使っています。ビューステートは Page クラスの ViewState プロパティを使ってアクセスできます。

   ViewState プロパティの詳細は、.NET Framework のドキュメントの Control.ViewState プロパティを参照してください。

4. 3. の手順でテキストエディターの IntelliSense 機能を使わずにプロパティを記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 テキストエディターの IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的にリポジトリ段落に追加されます。

```
PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
PROPERTY PROP-VIEWSTATE AS "ViewState"
```

5. 貼り付けた Button コントロールの Click イベントにイベント ハンドラを追加します。

6. 次の文を Button の Click イベントハンドラの WORKING-STORAGE SECTION の直下に、A 領域から記述します。

```
01 WORK-OBJECT OBJECT REFERENCE CLASS-OBJECT.
01 WORK-VALUE USAGE BINARY-LONG SIGNED.
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
```

7. 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。

```
CLASS CLASS-INT32 AS "System.Int32"
CLASS CLASS-STRING AS "System.String"
```

8. 次の文をイベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を利用することで、 プロパティの外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加されます。

```
SET WORK-OBJECT TO PROP-VIEWSTATE OF SELF::"get_Item" (N"COUNT").
SET WORK-VALUE TO WORK-OBJECT AS CLASS-INT32.
ADD 1 TO WORK-VALUE.
SET WORK-STRING TO CLASS-STRING::"Concat"
                  (N"ボタンが押された回数は、"
                   WORK-VALUE
                   N"回です。").
SET PROP-TEXT OF Label1 TO WORK-STRING
INVOKE PROP-VIEWSTATE OF SELF "Add"
      USING BY VALUE N"COUNT" BY VALUE WORK-VALUE.
```

上記は、ビューステートから今までにボタンが押された回数の値を取得し、 その値に 1 を加算して、文字列を組み立てて、 Label の Text プロパティに設定する手続きです。 ビューステートから取り出したオブジェクトは、System.Object 型であるため、 数値として取り出すためには、 System.Int32 のオブジェクト指定子によって修飾する必要があります。

9. 8.の手順でテキストエディターの IntelliSense 機能を使わずにプロパティを記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 テキストエディターの IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的にリポジトリ段落に追加されます。

```
        PROPERTY PROP-TEXT AS "Text"
```

アプリケーションのビルドと実行

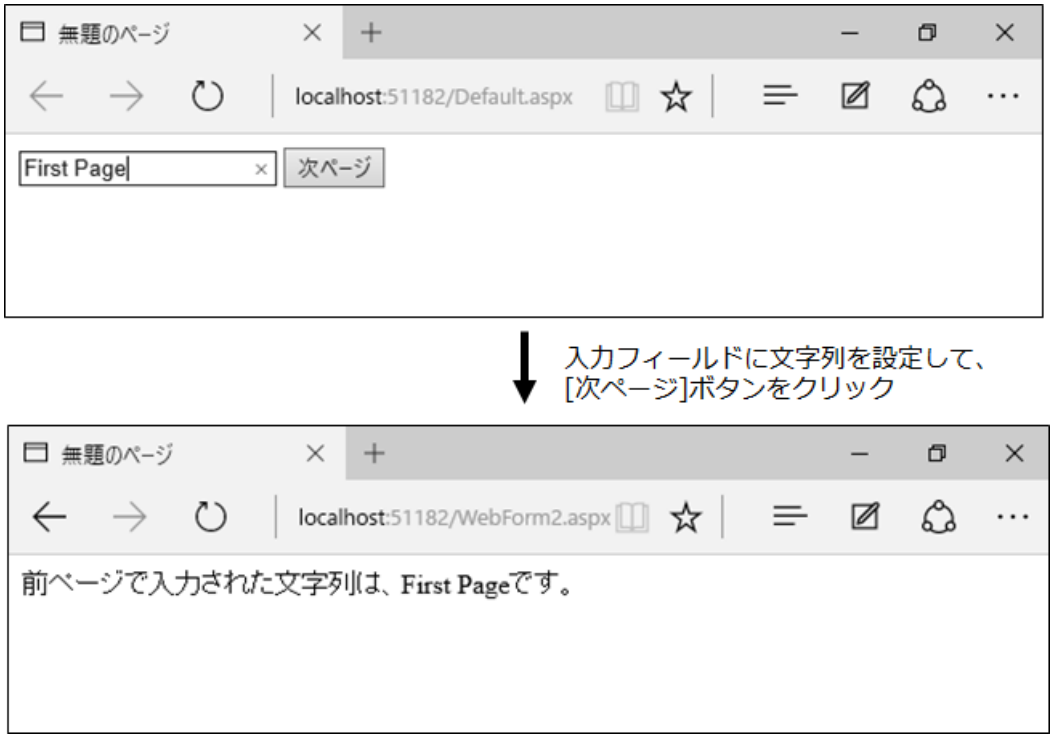
1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー配下でアプリケーションが実行されます。 Web ブラウザは自動的に起動されます。
2. ブラウザ上で、「COUNT」ボタンを数回クリックし、 ボタンが押された回数がカウントされているかを確認します。
3. Web ブラウザを閉じてアプリケーションを終了します。

ページ間の連携

最後に、Web フォームを複数使った ASP.NET Web アプリケーションを作成してみましょう。 ページ間で情報をやりとりするにはいくつかの方法があります。 ここでは、クエリ文字列を使った情報のやりとりについて説明します。 クエリ文字列は、URL に情報を付加して次のページに渡します。 たとえば、商品コードを入力するページから、 商品の在庫状況を表示するページに対して商品コードを渡したいような場合、 URL の後ろに「?ProductCode=100」といったものを追加します。

クエリ文字列を使う場合は、ブラウザによる URL の長さの制限や、 URL を通してインターネットに情報が公開されるためにセキュリティ上問題がある点に注意する必要があります。

ASP.NET の状態管理オプションの選択基準については、 Visual Studio のドキュメントの ASP.NET の状態管理に関する推奨事項を参照してください。



作成手順は以下のとおりです。

1. 1 ページ目の Web フォームの作成
2. 2 ページ目の Web フォームの作成
3. 1 ページ目のアプリケーションの手続きの記述
4. 2 ページ目のアプリケーションの手続きの記述
5. アプリケーションのビルドと実行

1 ページ目の Web フォームの作成

1. ラベルに文字列を設定するの[Web サイトの作成]と同様の操作をして、 MultiPage という ASP.NET Web アプリケーション用 Web サイトを作成します。 Web サイト名は自由につけて構いません。
2. Web ページ デザイナー を使って、ツールボックスから Web サーバ コントロールの TextBox コントロールと Button コントロールを、フォームに貼り付けます。
3. 貼り付けた Button コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティに、「次ページ」と設定します。

2 ページ目の Web フォームの作成

1. ソリューション エクスプローラー を開き、 Web サイト ノードを右クリックして、 [新しい項目の追加]メニューを選択します。
2. [新しい項目の追加] ダイアログが表示されたら、 テンプレートから [Web フォーム] を選択し、 ファイル名に「WebForm2.aspx」を入力して、[追加]ボタンをクリックします。 これで二つ目の Web フォームが追加されました。
3. Web ページ デザイナー を使って、ツールボックスから、 WebForm2.aspx に Web サーバ コントロールの Label コントロールを貼り付けます。
4. 貼り付けた Label コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティの内容をクリアします。

1 ページ目のアプリケーションの手続きの記述

1. WebForm1 に貼り付けた Button コントロールの Click イベントに イベント ハンドラを追加します。
2. 次の文を Button の Click イベントハンドラの WORKING-STORAGE SECTION の直下に、A 領域から記述します。

```
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
```

3. 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。

```
CLASS CLASS-STRING AS "System.String"
```

4. 次の文を Button の Click イベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を利用することで、 プロパティの外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加されます。

```
SET WORK-STRING TO PROP-TEXT OF TextBox1
SET WORK-STRING TO CLASS-STRING::"Concat"
       (N"WebForm2.aspx?Param=" WORK-STRING)
INVOKE PROP-RESPONSE OF SELF "Redirect" USING WORK-STRING
```

上記は、入力された文字列をクエリ文字列として、 2 ページ目のフォームである、WebForm2.aspx にリダイレクトしている文です。 クエリ文字列は"Param"を要素の名前、入力文字列を要素の値としてしています。 リダイレクトには、 Page クラスの HttpResponse メンバの Redirect メソッドを使用しています。

5. 4.の手順でテキストエディターの IntelliSense 機能を使わずにプロパティを記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 テキストエディターの IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的にリポジトリ段落に追加されます。

```
        PROPERTY PROP-TEXT AS "Text"
        PROPERTY PROP-RESPONSE AS "Response"
```

2 ページ目のアプリケーションの手続きの記述

1. WebForm2.aspx の Page の Load イベントにイベント ハンドラを追加します。
2. 次の文を OBJECT 段落の直下に、A 領域から記述します。

```
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
```

3. 次の文を環境部のリポジトリ段落に記述します。文は B 領域から記述します。

```
        CLASS CLASS-STRING AS "System.String"
```

4. 次の文を Page の Load イベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を利用することで、 プロパティの外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加されます。

```
        SET WORK-STRING TO
            PROP-QUERYSTRING OF PROP-REQUEST OF SELF
              ::"get_Item" (N"Param").
        IF WORK-STRING NOT = NULL THEN
            SET PROP-TEXT OF Label1 TO
                CLASS-STRING::"Concat"
                  (N"前ページで入力された文字列は、"
                   WORK-STRING
                   N"です。")
        ELSE
            SET PROP-TEXT OF Label1 TO
                N"文字列が指定されませんでした。"
        END-IF
```

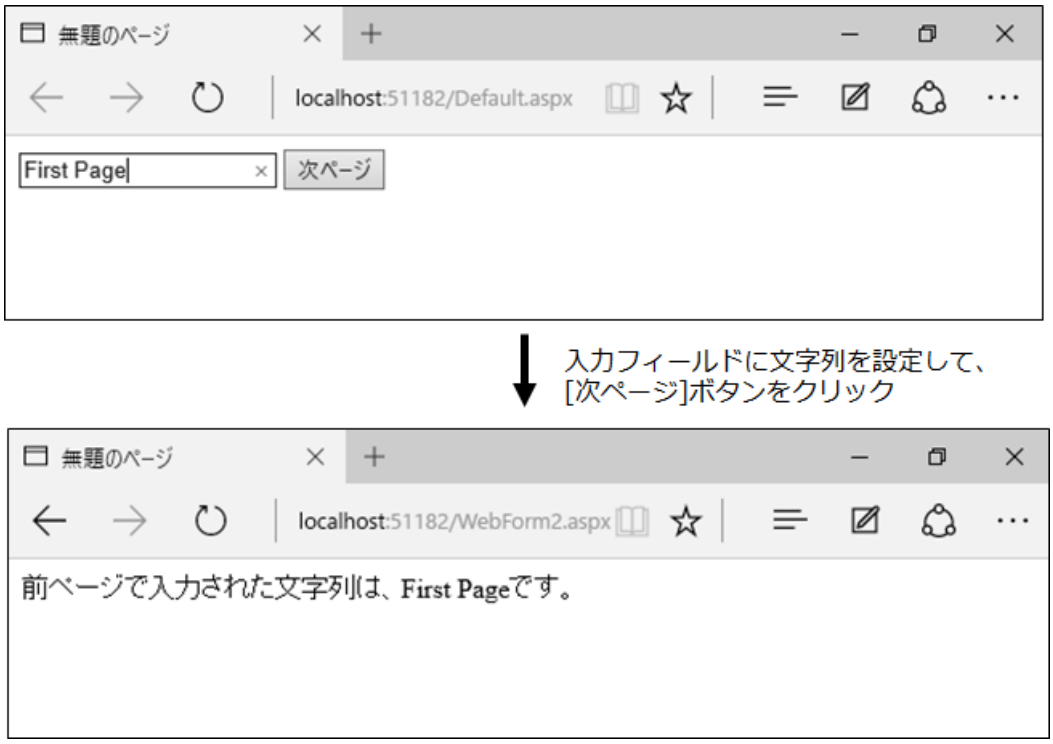
上記は、クエリ文字列から"Param"を要素の名前として入力された文字列を取得し、 Label に設定する文です。 クエリ文字列が指定されていない場合のために、 クエリ文字列の参照が NULL かどうか判定しています。

5. 4.の手順でテキストエディターの IntelliSense 機能を使わずにプロパティを記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 テキストエディターの IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的にリポジトリ段落に追加されます。

```
        PROPERTY PROP-QUERYSTRING AS "QueryString"
        PROPERTY PROP-REQUEST AS "Request"
        PROPERTY PROP-TEXT AS "Text"
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。 Web ブラウザは自動的に起動されます。
2. ブラウザ上で、入力フィールドに文字列を入力して、 「次ページ」ボタンをクリックし、次のページに文字列が渡っているかを確認します。



3. Web ブラウザを閉じてアプリケーションを終了します。

XML Web サービスの開発

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

NetCOBOL for .NET と ASP.NET を組み合わせることで、 XML Web サービスを COBOL で作成することができます。

XML Web サービスは、アプリケーションロジックなどの特定の機能要素を提供するコンポーネントです。 XML Web サービスは、 HTTP、XML、XSD、SOAP、WSDL などの標準プロトコルを使用して、 異なるプラットフォーム上のアプリケーションを統合することができます。

ここでは XML Web サービスを構築するための開発手順について説明します。

このセクションの内容

XML Web サービスの概要

XML Web サービスの概要について説明します。

XML Web サービスのプログラム構造

XML Web サービスのプログラム構造について説明します。

XML Web サービスの開発手順

XML Web サービスの開発手順について説明します。

NCOBU00028

XML Web サービスの概要

NetCOBOL for .NET では、 ASP.NET 技術をベースにして XML Web サービスを開発することができます。

ASP.NET は、Web サーバにおけるアプリケーションの開発および実行のプラットフォームです。

ASP.NET を使用することで、XML Web サービスを作成できます。 ASP.NET を使用して作られた XML Web サービスは、 その実装が XML や SOAP といった標準に基づいているため、 任意のプラットフォーム上のクライアントから利用することができます。ASP.NET を使用して作成された XML Web サービスは、ASP.NET Web サービスとも呼ばれます。 ASP.NET Web サービスは、 IIS(Microsoft Internet Information Services)で構成された Web サーバで動作します。

Visual Studio で XML Web サービスを作成すると、 XML Web サービスの URL アドレスをサービスの手続きと関連付ける.asmx ファイルと、 XML Web サービス の実装クラスを定義する .cob ファイルが作成されます。 XML Web サービスの実装クラスのメソッドには、 XML Web サービスの手続きを記述します。 XML Web サービスの実装クラス、および XML Web サービスメソッドには、 それぞれ次のような特徴があります。

XML Web サービスの実装クラス

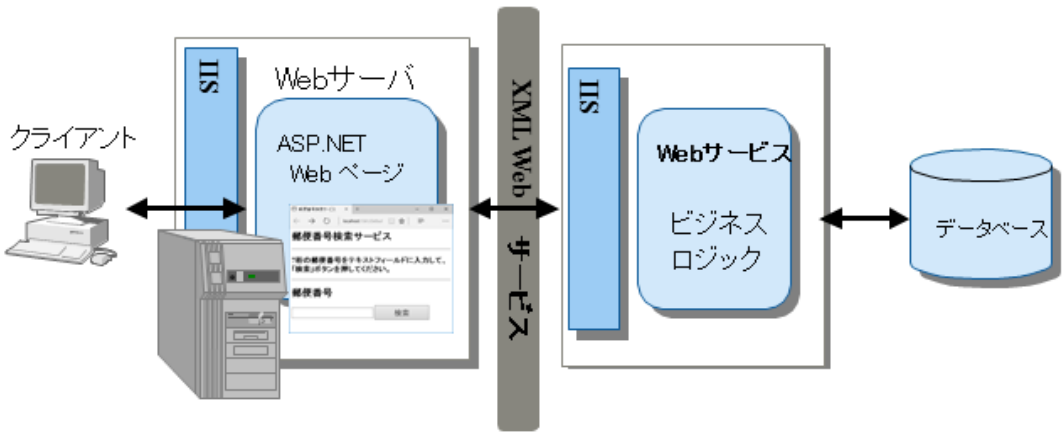
XML Web サービス の実装クラスは、 System.Web.Services.WebService クラスを継承します。 このクラスを継承することで、 アプリケーション状態やセッション状態などの ASP.NET の状態管理オプションを利用することができます。

XML Web サービスメソッド

XML Web サービスの手続きを記述するメソッドには、 System.Web.Services.WebMethodAttribute カスタム属性を付加します。 このカスタム属性を付加することで、 そのメソッドが、XML Web サービスメソッドとして認識されます。 また、この場合、メソッドは PUBLIC 属性でなければなりません。

クライアントと XML Web サービスメソッドは、 既定では SOAP メッセージを使用して通信します。 しかし、実際に XML Web サービスを作成したり、利用したりする場合には、 XML や SOAP を意識する必要はありません。

以下は、 Web サーバからビジネスロジックオブジェクトを利用する場合の通信インタフェースとして、 XML Web サービスを利用している例です。



- System.Web.Services.WebService クラスの詳細については、 .NET Framework のドキュメントの WebService クラスを参照してください。
- ASP.NET を使用した XML Web サービスの作成の詳細については、XML Web サービスの開発および .NET Framework のドキュメントの ASP.NET と XML Web サービス クライアントを使用して作成した XML Web サービスを参照してください。

NCOBU00029

XML Web サービスのプログラム構造

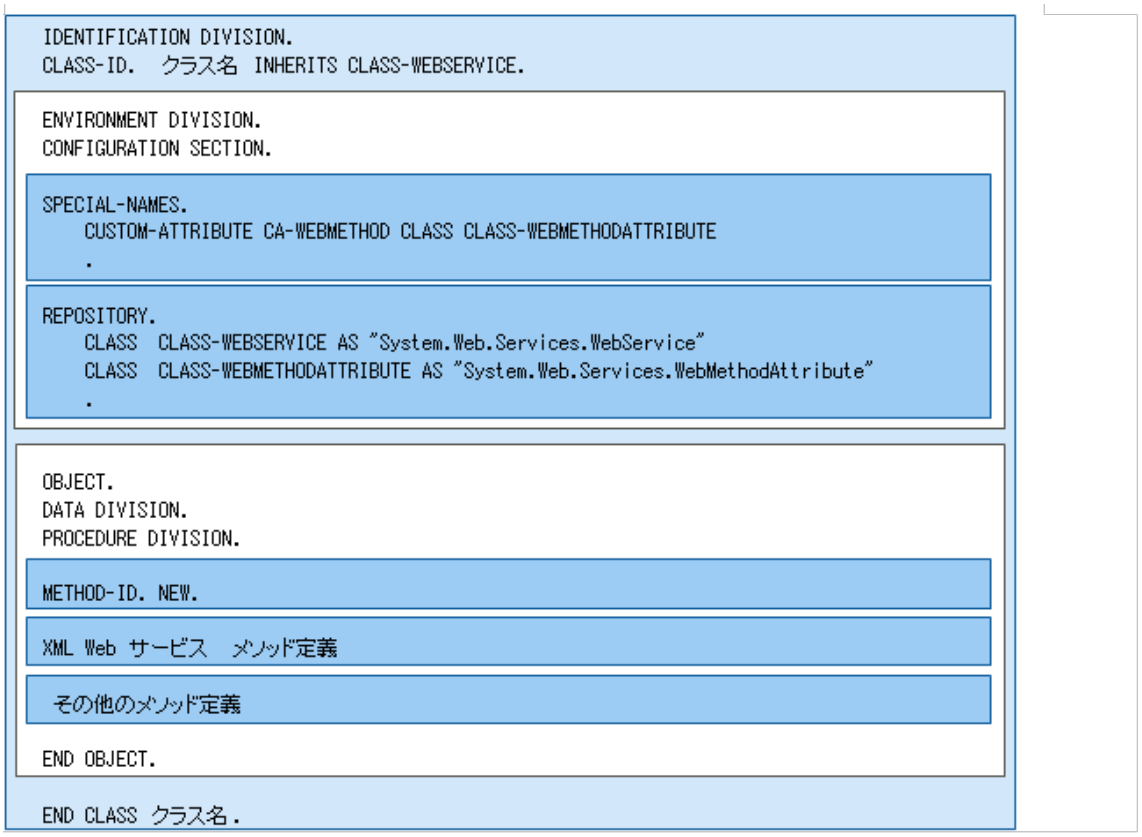
ここでは、XML Web サービスの COBOL プログラムの構造について説明します。 Visual Studio では、プロジェクトを作成する時に、 ASP.NET Web サービスのテンプレートを選択すると、 自動的に XML Web サービスを構成する以下のファイルが作成されます。

.cob ファイル	XML Web サービスの手続き
.asmx ファイル	@ WebService ディレクティブの指定。 XML Web サービスの URL アドレスをサービスの手続きと関連付ける。

.cob ファイル

.cob ファイルには、XML Web サービスの手続きを記述します。

XML Web サービスの COBOL ソースプログラムの構造は、以下のようになります。



XML Web サービスのプログラムの構造は Windows アプリケーションや ASP.NET Web アプリケーションの構造と大きな違いはありません。 メソッド名段落に、 System.Web.Services.WebMethodAttribute カスタム属性を付加した XML Web サービスメソッドを追加することで、 Web 上で動作可能な XML Web サービスを作成できます。 XML Web サービス以外のクラス定義の内容については、Windows アプリケーションのプログラム構造を参照してください。

.asmx ファイル

.asmx ファイルは、 XML Web サービスのエントリポイントとしての機能があります。

.asmx ファイルには、XML Web サービスであることを示す、 @ WebService ディレクティブが記述されています。 .cob ファイルと.asmx との関連付けは、 @ WebService ディレクティブによって行なわれます。

```
<%@ WebService Language="Fujitsu.COBOL" Codebehind="Service1.cob"
   Class="Service1" %>
```

Language	" Fujitsu.COBOL"固定
Codebehind	関連付ける COBOL ソースファイル名(.cob)
Class	COBOL ソースファイルの中で定義されているクラス名の外部名

.cob ファイルで定義したクラスのクラス名を変更した場合、.asmx ファイルの@ WebService ディレクティブの Class に指定されたクラス名は自動的に変更されないため、 手動で変更する必要があります。

NCOBU00030

XML Web サービスの開発手順

ここでは、簡単なサンプルを例にして、 Visual Studio で、 XML Web サービスを開発する場合の手順について説明します。

説明にしたがってアプリケーションを構築することによって、 XML Web サービスの開発方法の基本を学習することができます。

なお、本チュートリアルでは、ローカル IIS Web サイトを使用します。 この場合、あらかじめ IIS(Microsoft Internet Information Services)の設定をしておく必要があります。IIS の設定方法については、ASP.NET Web アプリケーションの実行のための事前準備を参照してください。 Web サイトの種類については、Visual Studio のドキュメントの Visual Web Developer における Web サイトの種類を参照してください。

このセクションの内容

文字列を返却する

文字列を返却する、XML Web サービスの作成について説明します。

二つの数値を加算し結果を返却

二つの引数を加算して結果を返却する、 XML Web サービスの作成について説明します。

XML Web サービスのクライアントからの利用

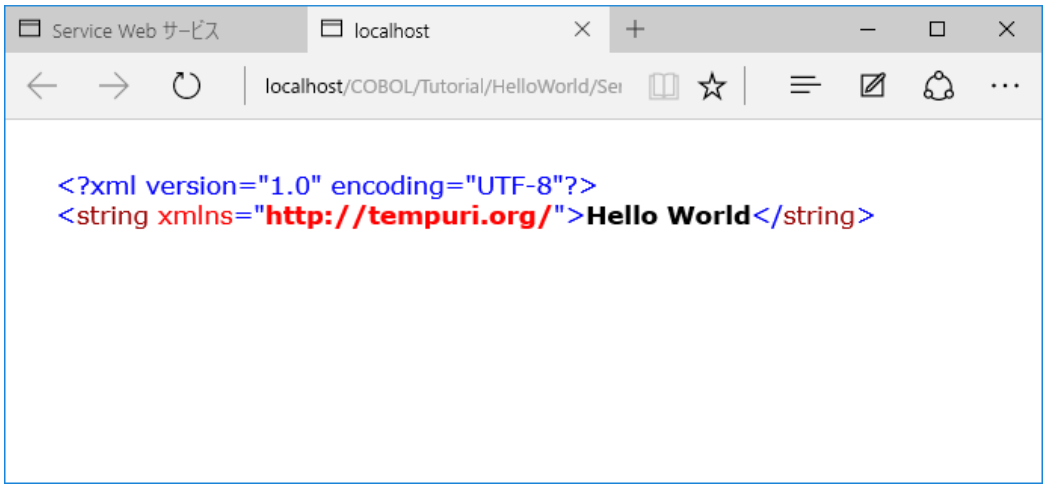
XML Web サービスを利用するクライアントアプリケーションの作成について説明します。

文字列を返却する

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ(http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

ここでは、「Hello,World!」という文字列を返却する、 簡単な XML Web サービスを作成してみましょう。

以下は、XML Web サービスメソッドである、 HelloWorld メソッドが処理された結果です。 結果は XML として出力されます。



作成手順は以下の とおりです。

1. XML Web サービスの Web サイトの作成
2. XML Web サービスメソッドの手続きの記述
3. アプリケーションのビルドと実行

XML Web サービスの Web サイトの作成

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[Web サイト]を選択します。
3. [新しい Web サイト]ダイアログボックスが表示されます。

4. ダイアログボックスの上に表示されているボックスでは、対象の.NET Framework を指定します。このサンプルでは、[.NET Framework 4.7]を選択します。
5. 左ペインで、[NetCOBOL for .NET]を選択します。
6. 右ペインで、[ASP.NET Web サービス]を選択します。
7. [Web の場所]ボックスは、Web サイトの構築場所を指定します。 ここでは、ローカル IIS Web サイトを使用することにし、 [Web の場所]ボックスに[HTTP]を選択して、Web サイトの URL に「`http://localhost/COBOL/Tutorial/HelloWorld`」と設定します。

   なお、ローカル IIS Web サイトを使用する場合は、あらかじめ IIS を設定しておく必要があります。 IIS

の設定方法については、ASP.NET Web アプリケーションの実行のための事前準備を参照してください。

8. [OK]ボタンをクリックします。 これで、アプリケーションの構築に必要なファイル一式が作成されました。

   ソリューションエクスプローラーには、 プロジェクトの構成を表すツリーが表示されます。 ソリューションエクスプローラーは、Visual Studio の[表示]メニューから [ソリューション エクスプローラー]を選択すると表示されます。

XML Web サービス メソッドの手続きの記述

XML Web サービスクラスのメソッドの手続きを記述します。

1. XML Web サービスクラスのコードを表示させます。コードは、ソリューションエクスプローラーから、[App_Code]フォルダの.cob ファイルをダブルクリックするか、または.cob ファイルのノードを右クリックして表示されるメニューから [開く]を 選択することによって、手続きコードが表示されます。

2. 次に、 XML Web サービスメソッドのメソッド定義を OBJECT 定義に追加します。 なお、以下のメソッド定義は、テンプレートによって自動生成されているので、これをそのまま使用します。

   上記の HelloWorld メソッドは、引数がなく、 復帰値として System.String 型の文字列を返します。 復帰値の文字列には、 "Hello World" を設定しています。 また、メソッド名段落に System.Web.Service.WebMethodAttribute カスタム属性が付加されています。 このカスタム属性によって、 このメソッドが XML Web サービスメソッドであることを宣言しています。

   カスタム属性については、カスタム属性を使うを参照してください。

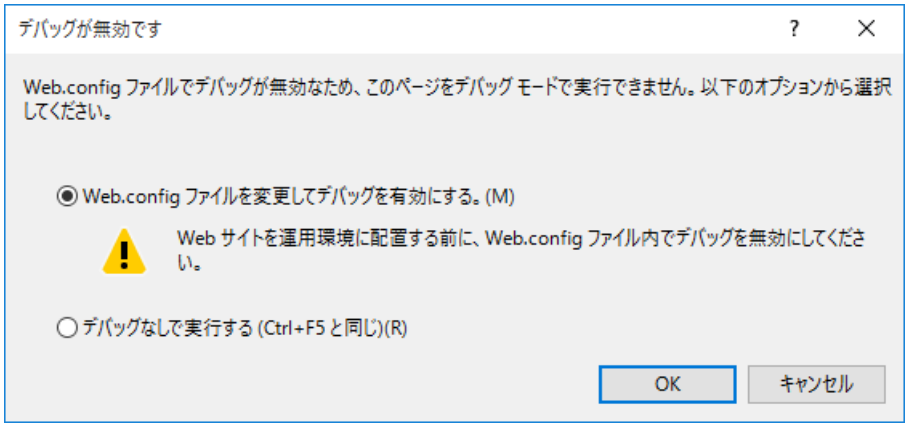
アプリケーションのビルドと実行

1. XML Web サービスをデバッグするため、サービスのエントリポイントとなる Service.asmx ファイルをスタートページに設定します。 スタートページに設定するには、ソリューションエクスプローラーで Service.asmx ファイルを選択し、ノードを右クリックして表示されるメニューから[スタート ページに設定]を選択します。

2. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー配下でアプリケーションが実行されます。

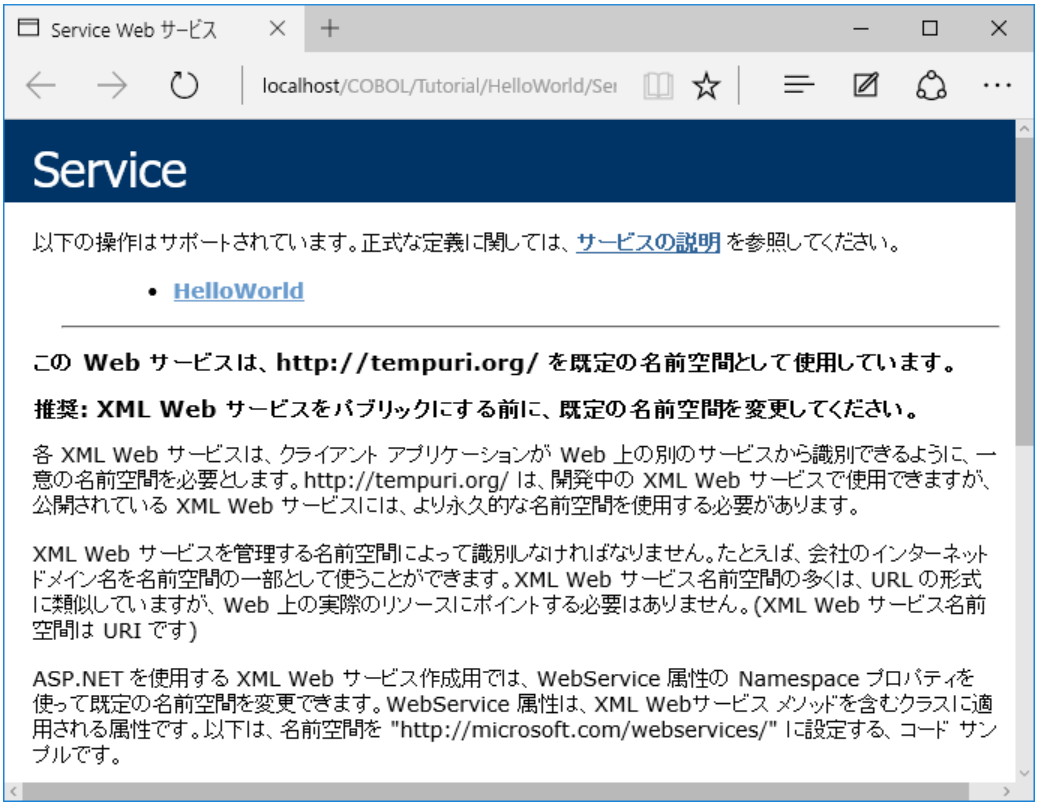
   テンプレートから自動生成した場合はデバッグが有効になっていないため、デバッガー配下で Web アプリケーションを実行すると以下のメッセージボックスが表示されます。

   デバッグを行う場合は、[デバッグを有効にするために Web.config ファイルを変更する]を選択し、[OK]ボタンをクリックします。

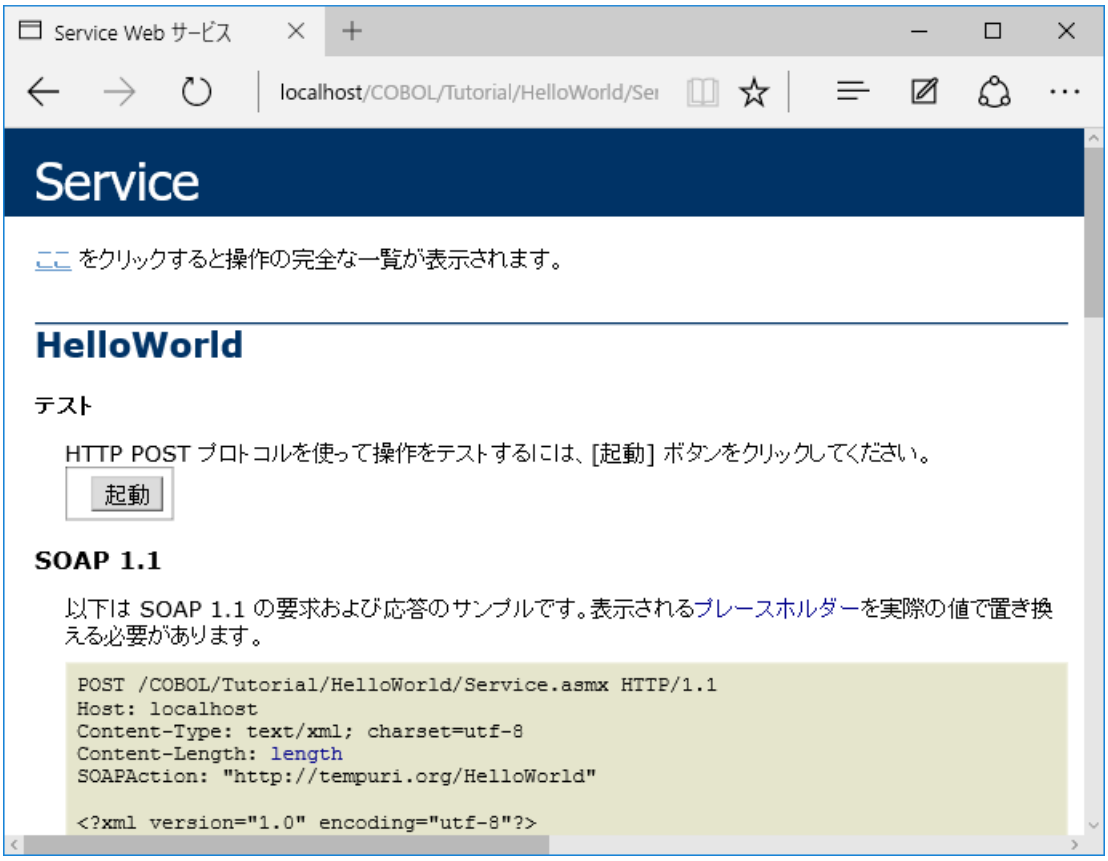
3. Visual Studio は、Web ブラウザを起動し、 XML Web サービスエントリポイントとなる URL のページを要求します。 ASP.NET は、XML Web サービスをテストするためのテストページを動的に生成します。

   テンプレートから自動生成された状態では、XML Web サービスの名前空間は既定の `http://tempuri.org/`が設定されているため、以下の画面が表示されます。 テストページには、 利用可能な XML Web サービスメソッドの一覧が表示されるため、 この中から「HelloWorld」をクリックして次に進みます。

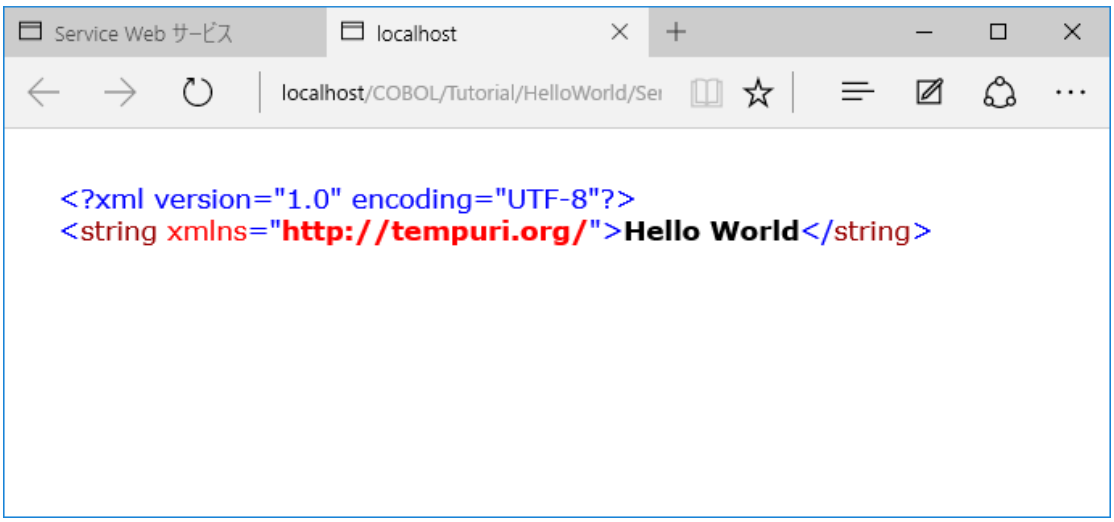


COBOL で XML Web サービスに名前空間の属性を指定する場合は、 XML Web サービスクラスに付加されている System.Web.Services.WebServiceAttribute カスタム属性の、 Namespace プロパティに指定されている名前空間の文字列を変更します。

4. HelloWorld メソッドを起動するページが開くため、 ここで[起動]ボタンをクリックします。



5. HelloWorld メソッドが実行され、XML に出力された結果が表示されます。



6. Web ブラウザを閉じます。

二つの数値を加算し結果を返却

次に、二つの数値を引数として入力し、加算した結果を返却する、 XML Web サービスメソッド を作成してみましょう。

作成手順は以下のとおりです。

1. XML Web サービスの Web サイトの作成
2. アプリケーションの手続きの記述
3. アプリケーションのビルドと実行

XML Web サービスの Web サイトの作成

1. 文字列を返却するの[XML Web サービスの Web サイトの作成]と同様の操作をして、 AddMe という XML Web サービス用 Web サイトを作成します。 Web サイト名は自由につけて構いません。

アプリケーションの手続きの記述

1. XML Web サービスクラスの手続きを開きます。
2. 次の XML Web サービスメソッド定義を OBJECT 段落に追加します。 ここでは、END OBJECT の直前に挿入することにします。

```
METHOD-ID. ADDME AS "AddMe" CUSTOM-ATTRIBUTE IS CA-WEBMETHOD.
DATA DIVISION.
LINKAGE SECTION.
01 OPERAND-1 USAGE BINARY-LONG SIGNED.
01 OPERAND-2 USAGE BINARY-LONG SIGNED.
01 RET-VAL   USAGE BINARY-LONG SIGNED.
PROCEDURE DIVISION USING BY VALUE OPERAND-1
                         BY VALUE OPERAND-2
                         RETURNING RET-VAL.
   ADD OPERAND-1 OPERAND-2 TO RET-VAL.
END METHOD ADDME.
```

上記は、二つの値を引数にして、加算した結果を返す手続きです。 引数と戻り値の型は、XML スキーマ定義(XSD: XML Schema Definition)言語の Part 2 で説明されているデータ型が使用できます。 これらのデータ型は、.NET Framework のデータ型に対応します。 詳細については、.NET Framework のドキュメントの ASP.NET を使用して作成した XML Web サービスによってサポートされるデータ型を参照してください。COBOL のデータ型との対応については、.NET Framework のデータ型と COBOL のデータ型の対応を参照してください。

ほぼすべてのデータ型は、 XML にシリアル化(オブジェクトを簡単に転送できる形式に変換する処理)できます。 しかし、COBOL 独自データ型を引数に持つような XML Web サービスメソッドは、 データのマーシャリング(プロセスの境界を越えて、メソッドパラメタのパッケージ化と送信を行うこと)がおこなわれず、 単にバイナリのデータとしてしか扱われません。 そのため COBOL 独自データ型は、実際には利用することができません。

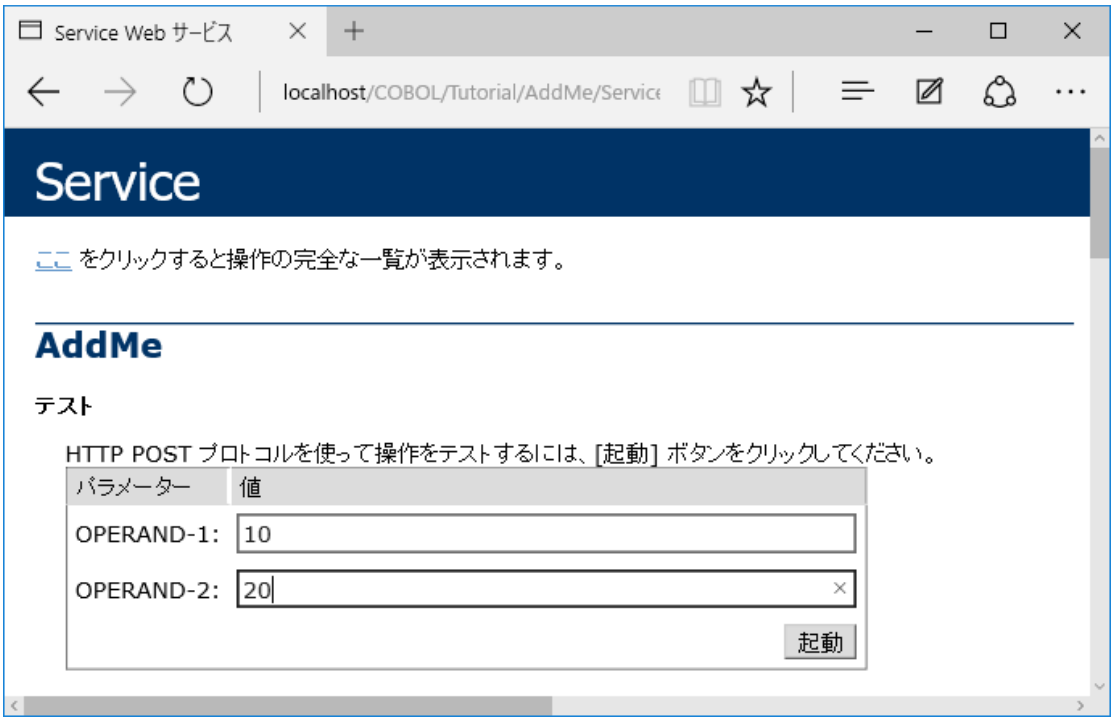
アプリケーションのビルドと実行

1. XML Web サービスをデバッグするため、サービスのエントリポイントとなる Service.asmx ファイルをスタートページに設定します。 スタートページに設定するには、ソリューションエクスプローラーで Service.asmx ファイルを選択し、ノードを右クリックして表示されるメニューから[スタート ページに設定]を選択します。
2. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビル

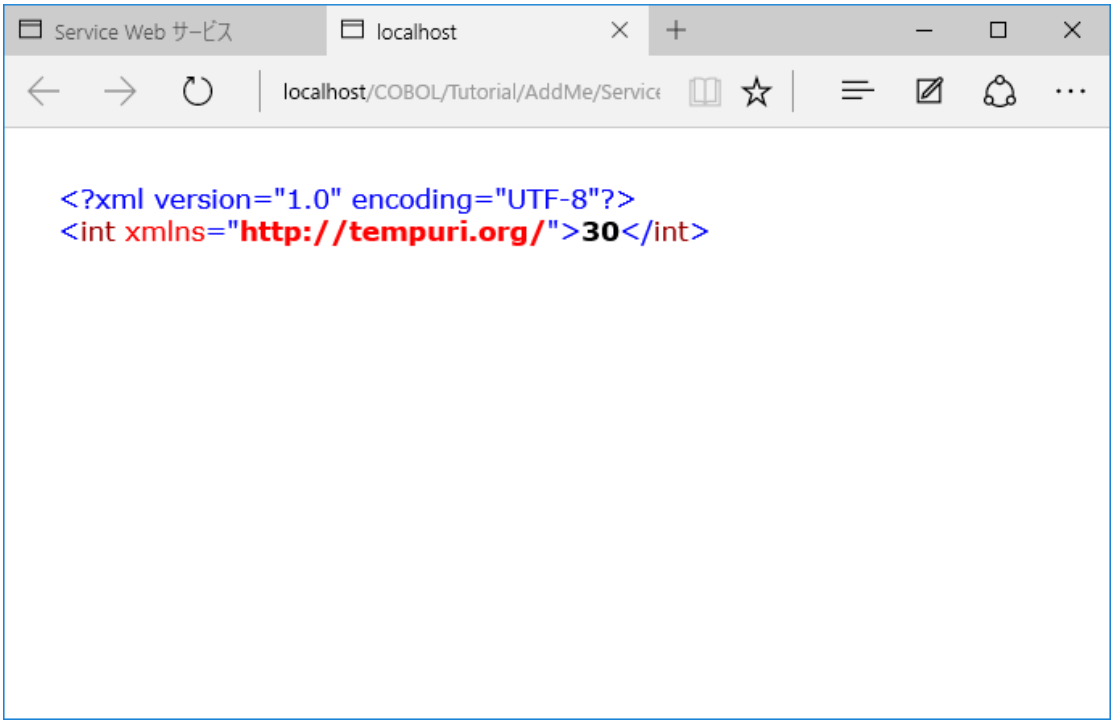
ドされ、デバッガー 配下でアプリケーションが実行されます。 Visual Studio は、Web ブラウザを起動し、 スタートページに設定したページを要求します。 ASP.NET は、 XML Web サービスをテストするためのテストページを動的に生成します。

3. テストページには、 利用可能な XML Web サービスメソッドの一覧が表示されています。 一覧に表示されている「AddMe」をクリックします。

4. 「AddMe」XML Web サービスメソッドを起動するページが開くので、OPERAND-1 と、OPENRAND-2 にそれぞれ任意の数値を指定し、 [起動]ボタンをクリックします。



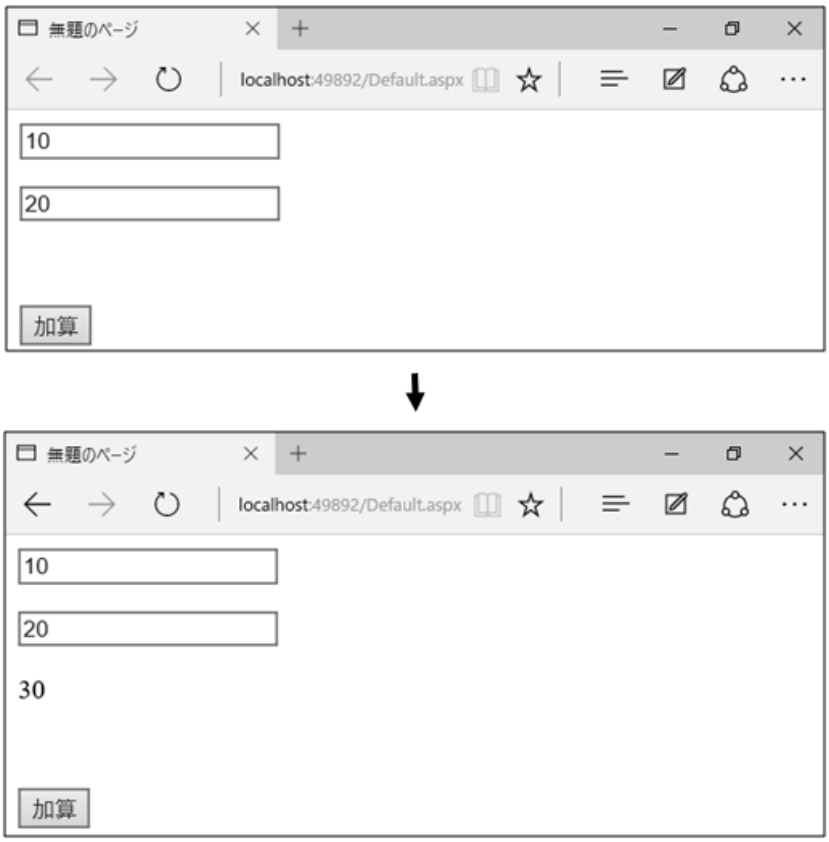
結果が XML として渡され、内容が別のブラウザ ウィンドウ上に表示されます。 指定した二つの数値が加算された値が結果として渡ってきたかを確認します。



5. Web ブラウザを閉じます。

XML Web サービスのクライアントからの利用

XML Web サービスを利用するクライアントを作成してみましょう。 ここでは、二つの数値を加算し結果を返却で作成した XML Web サービスを利用する、 ASP.NET Web アプリケーションを作成します。



作成手順は以下のとおりです。

1. Web フォームの作成
2. Web 参照設定
3. アプリケーションの手続きの記述
4. アプリケーションのビルドと実行

Web フォームの作成

1. ASP.NET Web アプリケーションの開発で説明しているラベルに文字列を設定するの [Web サイトの作成]と同様の操作をして、 AddMePage という ASP.NET Web アプリケーション用 Web サイトを作成します。 ここでつけた名前は以降の作業でも参照します。
2. Web ページ デザイナー を使って、ツールボックスから Web サーバ コントロールの TextBox コントロールを二つと、 Label コントロール、および Button コントロールをフォームに貼り付けます。
3. 貼り付けた Button コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティに、「加算」と設定します。
4. 貼り付けた Label コントロールをクリックして選択し、プロパティ ウィンドウで Text プロパティの内容をクリアします。

Web 参照設定

1. ソリューション エクスプローラーで、 Web サイトのノードを右クリックし、 ショートカットメニューから[サービス参照の追加]メニューを選択します。
2. [サービス参照の追加]ウィンドウの[詳細設定]ボタンを選択します。
3. [サービス参照設定]ウィンドウの[Web 参照の追加]ボタンを選択します。
4. 「Web 参照の追加」ウィンドウのアドレス入力フィールドに、 「二つの数値を加算し結果を返却」で作成した XML Web サービスのアドレスを指定します。
5. 「AddMe」XML Web サービスが表示されたら[参照の追加]ボタンをクリックします。 これによって、 XML Web サービスとアプリケーションとの間で やりとりされる引数のマーシャリング (プロセスの境界を越えてメソッド パラメタのパッケージ化と送信を行う) を処理する XML Web サービス プロキシ クラスが自動的に作成されます。 このプロキシ クラスは、XML Web サービスメソッドを公開します。 XML Web サービスメソッドを呼び出す場合は、 プロキシ クラスのオブジェクトを仲介します。 XML Web サービス プロキシ クラスの詳細については、 .NET Framework のドキュメントの XML Web サービス プロキシの作成を参照してください。

アプリケーションの手続きの記述

1. 貼り付けた Button コントロールの Click イベントにイベント ハンドラを追加します。
2. 次の文をイベントハンドラの WORKING-STORAGE SECTION の直下に、A 領域から記述します。

```
01 WORK-INT32-1 USAGE BINARY-LONG SIGNED.
01 WORK-INT32-2 USAGE BINARY-LONG SIGNED.
01 WORK-INT32-3 OBJECT REFERENCE CLASS-INT32.
01 WORK-STRING  OBJECT REFERENCE CLASS-STRING.
01 WEBSERVICE   OBJECT REFERENCE CLASS-WEBSERVICE.
```

3. 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。

```
CLASS CLASS-WEBSERVICE AS "localhost.Service1"
CLASS CLASS-INT32 AS "System.Int32"
CLASS CLASS-STRING AS "System.String"
```

XML Web サービス プロキシ クラスの内部名 CLASS-WEBSERVICE に対応する外部名は、 「アプリケーションの名前空間.ホスト名.サービス名」という形式です。 この例では、「localhost.Service1」という外部名としていますが、 アプリケーションで実際に使用している名前を使ってコーディングしてください。

4. 次の文を Button の Click イベントハンドラに挿入します。 文はイベント ハンドラの PROCEDURE DIVISION の直下に、B 領域から記述します。 テキストエディターの IntelliSense 機能を利用することで、 プロパティの外部名と内部名とを対応付ける文がリポジトリ段落に自動的に追加されます。

```
*> TextBox から数値を取り出す
SET WORK-STRING TO PROP-TEXT OF TextBox1
MOVE CLASS-INT32::"Parse" (WORK-STRING) TO WORK-INT32-1
SET WORK-STRING TO PROP-TEXT OF TextBox2
MOVE CLASS-INT32::"Parse" (WORK-STRING) TO WORK-INT32-2

*> AddMe XML Web サービス メソッドを呼び出す
SET WEBSERVICE TO CLASS-WEBSERVICE::"NEW"
SET WORK-INT32-3 TO
   WEBSERVICE::"AddMe" (WORK-INT32-1 WORK-INT32-2)

*> AddMe の結果を Label に設定する
SET WORK-STRING TO WORK-INT32-3::"ToString"
SET PROP-TEXT OF Label1 TO WORK-STRING.
```

上記は、二つの TextBox から数値を取り出して、 XML Web サービス メソッドを呼び出し、 結果をラベルに設定する手続きです。 入力された文字列を数値に変換するために、 ここでは System.Int32 の

Parse メソッドを利用しています。 System.Int32 の Parse メソッドの詳細については、 .NET Framework のドキュメントの Int32.Parse メソッドを参照してください。

[Web 参照の追加]によって、XML Web サービス プロキシ クラスが生成されているので、 そのクラスインスタンスを生成します。 XML Web サービス メソッドは、 プロキシ クラス のメソッドとして公開されているので、それを呼び出します。

なお、TextBox に入力された文字列は数値に変換できない場合もあるため、 数値以外の文字列を System.Int32 の Parse メソッドに渡すと例外が発生します。 ここではサンプルを単純にするため文字列の入力検査を省略していますが、 ASP.NET Web アプリケーションの場合であれば、 RegularExpressionValidator コントロールおよび RequiredFieldValidator コントロールを使うことで入力検査ができます。 これらのコントロールは、DHTML をサポートしているブラウザであれば、 クライアント側で入力検査をおこなうため、ラウンドトリップがおこなわれません。 DHTML がサポートされていないブラウザであれば、ラウンドトリップがおこなわれて、 サーバ側で入力検査がおこなわれますが、検査用の手続きを記述する必要がありません。 単純な入力検査であれば、検査用コントロールを使用する方法がお勧めです。 その他にも、入力された文字列を手続きの中でチェック する方法や、USE 文を使用した例外処理をおこなう方法があります。

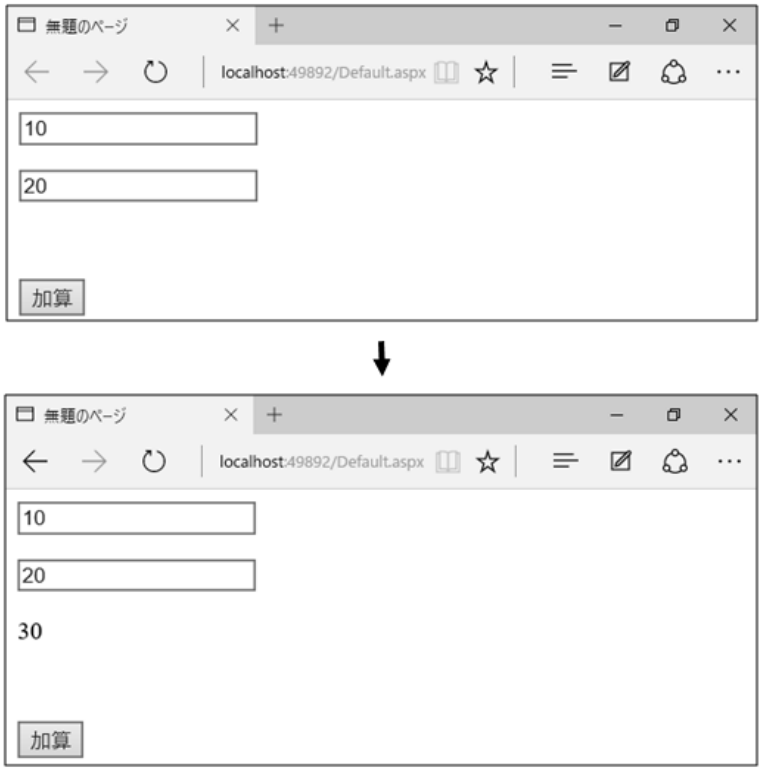
5. 4.の手順でテキストエディターの IntelliSense 機能を使わずにプロパティを記述した場合は、 次の文を環境部のリポジトリ段落に挿入します。 文は B 領域から記述します。 テキストエディターの IntelliSense 機能を使ってプロパティを挿入した場合、 次の文は自動的にリポジトリ段落に追加されます。

```
        PROPERTY PROP-TEXT AS "Text"
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。 Web ブラウザは自動的に起動されます。

2. ブラウザ上で、二つの TextBox に任意の数値を入力し「加算」ボタンをクリックします。 加算された結果が Label に表示されるかを確認します。



3. Web ブラウザを閉じてアプリケーションを終了します。

ここでは、ASP.NET Web アプリケーションを例にしましたが、 XML Web サービスを利用するクライアントが Windows アプリケーションであっても基本的に変わりません。

XML Web サービス クライアントの作成についての、 より詳細な情報は Visual Studio のドキュメントの XML Web サービスのクライアントの作成を参照してください。

SQL CLR データベースオブジェクトの開発

NetCOBOL for .NET と.NET Framework を組み合わせることで、 SQL Server の内部で実行される SQL CLR データベースオブジェクトを構築することができます。

ここでは SQL CLR データベースオブジェクトを構築するための開発手順について説明します。

このセクションの内容

SQL CLR データベースオブジェクトの概要

SQL CLR データベースオブジェクトの概要について説明します。

SQL CLR データベースオブジェクトのプログラム構造

SQL CLR データベースオブジェクトのプログラム構造について説明します。

SQL CLR データベースオブジェクトの開発手順

SQL CLR データベースオブジェクトの開発手順について説明します。

このチュートリアルでは、SQL CLR に関する知識が必要となります。 SQL CLR については、 .NET Framework のドキュメントの SQL Server の共通言語ランタイム統合および、SQL Server オンラインブックの CLR (共通言語ランタイム) 統合のプログラミング概念を参照してください。

NCOBU00801

SQL CLR データベースオブジェクトの概要

NetCOBOL for .NET では、 SQL CLR 技術をベースにして、 SQL CLR データベースオブジェクトを開発することができます。

SQL Server ではストアドプロシージャを開発する場合、 Transact-SQL 言語を使用していましたが、 SQL Server 2005 以降では、.NET Framework CLR(共通言語ランタイム)と統合することで、 .NET Framework 上で利用可能な言語を用いたマネージコードによるデータベースプログラミングが可能となりました。 この技術を SQL CLR と呼びます。

この SQL CLR 技術により、 例外処理やクラスライブラリの利用など Transact-SQL では実現できなかった複雑な処理を行うことが可能となり、 ストアドプロシージャを開発することにおいてより幅広いプログラミングができるようになりました。

SQL CLR 技術によるデータベースオブジェクトには以下のものがあります。

スタティックメソッドにマップされるデータベースオブジェクト

- ユーザ定義スカラ値関数
- ユーザ定義テーブル値関数
- ユーザー定義ストアドプロシージャ
- ユーザー定義トリガ

クラス全体にマップされるデータベースオブジェクト

- ユーザ定義型
- ユーザ定義集約関数

これらのデータベースオブジェクトは、SQL CLR データベースオブジェクトまたは、 CLR ルーチンと呼ばれます。

NetCOBOL for .NET では、スタティックメソッド定義を使用し、 以下の SQL CLR データベースオブジェクトを作成することができます。

SQL CLR ユーザ定義スカラ値関数

文字列値、整数値などの単一値を返すユーザ定義関数を作成できます。

SQL CLR ユーザ定義テーブル値関数

テーブルを返すユーザ定義関数を作成できます。

SQL CLR ストアドプロシージャ

ユーザ定義ストアドプロシージャを作成できます。

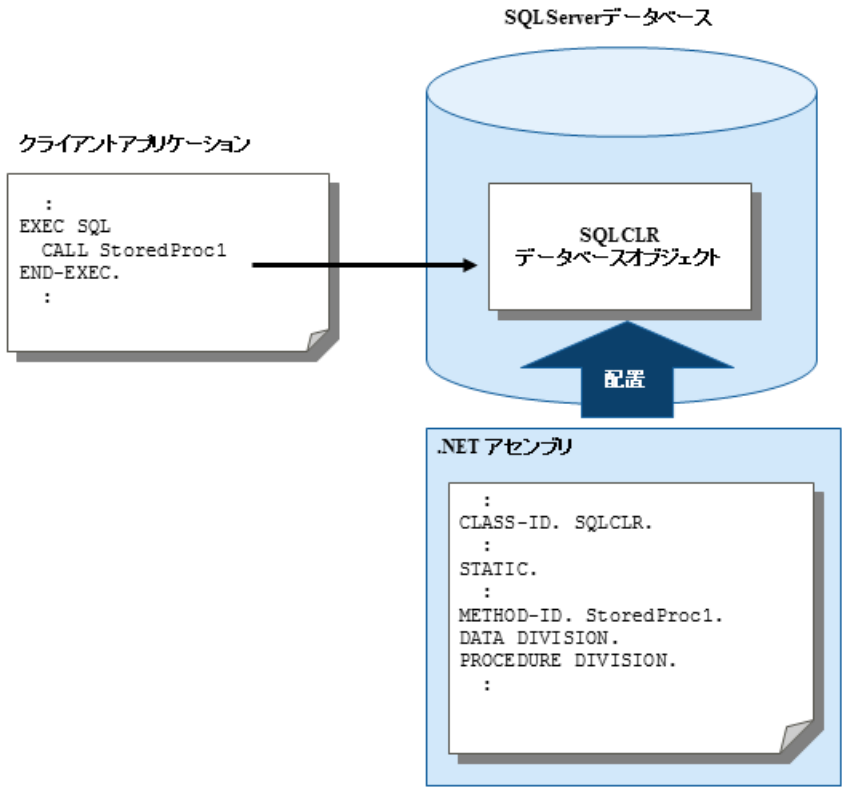
SQL CLR トリガ

テーブルの更新や削除といった、言語イベントの実行時に自動的に実行される特殊なストアドプロシージャ(トリガ)を作成できます。

注意

NetCOBOL for .NET では、ユーザ定義型およびユーザ定義集約関数を作成することはできません。

SQL CLR データベースオブジェクトは、 単純にビルドしただけでは、通常のクラスライブラリ(DLL)と相違はありませんが、 SQL Server のデータベースに登録(配置)することにより、 Transact-SQL で作成されたストアドプロシージャと同様に、クライアントアプリケーションのデータベースアクセスからの呼び出しにより、SQL Server 内部で動作します。



参考

関連トピック

- .NET Framework のドキュメントの SQL Server の共通言語ランタイム統合
- SQL Server オンラインブックの CLR (共通言語ランタイム) 統合のプログラミング概念

NCOBU00802

SQL CLR データベースオブジェクトのプログラム構造

ここでは､SQL CLR データベースオブジェクトの COBOL プログラムの構造について説明します。Visual Studio では、プロジェクトを作成する時に、 [SQL Server プロジェクト]のテンプレートを選択すると、 自動的に SQL CLR データベースオブジェクトを構成するプロジェクトが作成されます。

プロジェクトを作成した時点では、 SQL CLR データベースオブジェクトを作成するためのソースコードを含むファイルは追加されていないため、 [プロジェクト]メニューから[新しい項目の追加]を選択し、 作成したい SQL CLR データベースオブジェクトのテンプレートから COBOL ソースコードファイルを追加します。 SQL CLR データベースオブジェクトのテンプレートには以下の種類があります。

[ストアド プロシージャ]

SQL CLR ストアドプロシージャを作成するためのテンプレートです。

[ユーザ定義の関数]

SQL CLR ユーザ定義スカラ値関数または SQL CLR ユーザ定義テーブル値関数を作成するためのテンプレートです。

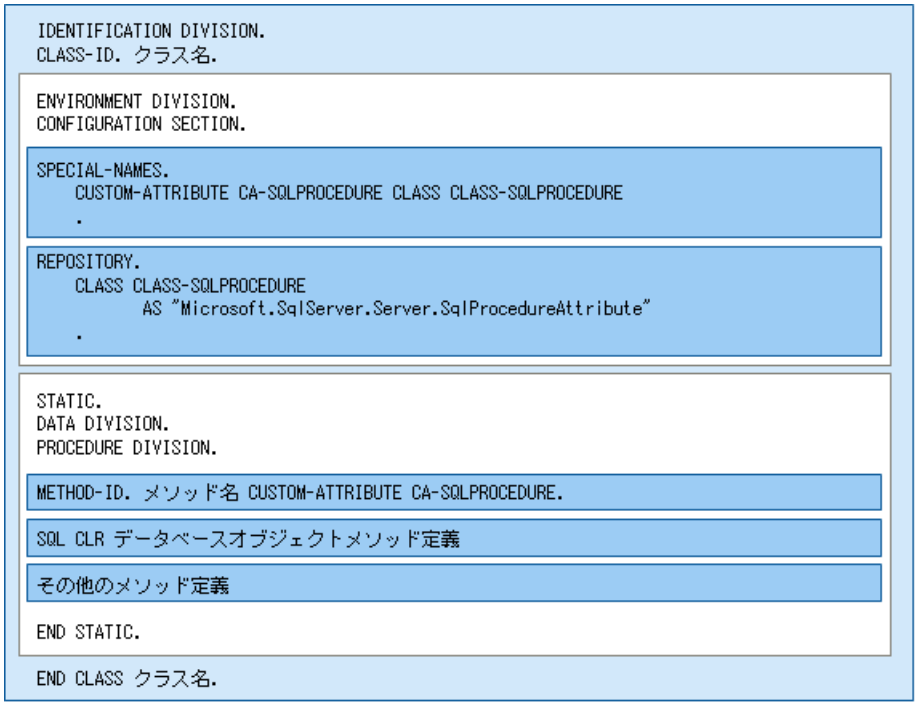
[トリガ]

SQL CLR トリガを作成するためのテンプレートです。

追加した COBOL ソースコードファイルには、SQL CLR データベースオブジェクトの手続きを記述します。

各 SQL CLR データベースオブジェクトのプログラム構造は基本的に同じです。 ここでは、ストアドプロシージャを例に、SQL CLR データベースオブジェクトのプログラム構造について説明します。

ストアドプロシージャの COBOL ソースプログラム構造は、以下のようになります。



ストアドプロシージャの COBOL ソースプログラムは、 スタティック定義にメソッドを定義することで、 SQL Server 上で動作可能なストアドプロシージャを作成できます。 これは、ユーザ定義関数やトリガの場合も同じです。 ただし、以下の点に注意してください。

- SQL CLR ストアドプロシージャのメソッド名段落には、Microsoft.SqlServer.Server.SqlProcedureAttribute カスタム属性が必要です。
- SQL CLR ユーザ定義関数のメソッド名段落には、Microsoft.SqlServer.Server.SqlFunctionAttribute カスタム属性が必要です。
- SQL CLR トリガのメソッド名段落には、Microsoft.SqlServer.Server.SqlTriggerAttribute カスタム属性が必要です。

ストアドプロシージャや SQL CLR データベースオブジェクトのメソッド以外のクラス定義の構造については Windows アプリケーションのプログラム構造を参照して下さい。

NCOBU00803

SQL CLR データベースオブジェクトの開発手順

ここでは、簡単なサンプルを例にして、 Visual Studio で、 SQL CLR データベースオブジェクトを開発する場合の手順について説明します。

説明にしたがってアプリケーションを構築することによって、 SQL CLR データベースオブジェクトの開発方法を学習することができます。

このチュートリアルで使用する SQL Server の環境は以下のとおりです。

サーバ名¥インスタンス名	(localdb)¥MSSQLLocalDB
データベース名	COBOLSample

- Microsoft SQL Server 2016 Express LocalDB を使用しています。
- "COBOLSample"データベースのセットアップ方法については、NetCOBOL for .NET サンプルデータベースのセットアップを参照してください。

このセクションの内容

SQL CLR データベースオブジェクトを使用するための事前準備

SQL CLR データベースオブジェクトを使用するために必要な作業について説明します。

SQL CLR ストアドプロシージャの作成

SQL CLR ストアドプロシージャを作成する方法について説明します。

SQL CLR ユーザ定義関数の作成

SQL CLR ユーザ定義関数を作成する方法について説明します。

SQL CLR トリガの作成

SQL CLR トリガを作成する方法について説明します。

SQL CLRデータベースオブジェクトを使用するための事前準備

SQL CLR データベースオブジェクトを使用する場合、SQL Server 上で SQL CLR が利用できることを前提にしています。 また、SQL Server が動作しているコンピュータには、NetCOBOL for .NET ランタイムシステムがインストールされ、 かつ、SQL CLR データベースオブジェクトを配置するデータベースには、NetCOBOL for .NET ランタイムアセンブリがあらかじめ配置されている必要があります。

SQL CLR を有効にする

SQL Server のデフォルトの設定では、 SQL CLR 機能は有効になっていません。 SQL CLR 機能を有効にするためには、sqlcmd ユーティリティを使用して SQL Server 上の'clr enabled'サーバ構成オプションを変更します。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB
1> sp_configure 'clr enabled',1
2> GO
1> RECONFIGURE
2> GO
```

詳細については SQL Server オンラインブックのサーバー構成オプションおよび、clr enabled サーバー構成オプションを参照してください。

また、sqlcmd ユーティリティの使い方の詳細については、sqlcmd ユーティリティの使用を参照してください。

NetCOBOL for .NET ランタイムアセンブリの登録

NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトを配置するデータベースには、 あらかじめ、NetCOBOL for .NET ランタイムアセンブリを登録しておく必要があります。

登録方法については、ランタイムアセンブリを登録するを参照してください。

NCOBU00805

SQL CLR ストアドプロシージャの作成

ここでは、簡単なサンプルを例にして、 Visual Studio で、 SQL CLR ストアドプロシージャを作成する手順について説明します。

SQL CLR ストアドプロシージャ についての詳細は、SQL Server オンラインブックの CLR ストアドプロシージャを参照してください。

このセクションの内容

メッセージを返すストアドプロシージャ(基本編)

メッセージを返すストアドプロシージャを作成する方法について説明します。

パラメタとデータベース機能を使用するストアドプロシージャ(応用編)

パラメタとデータベース機能を使用するストアドプロシージャを作成する方法について説明します。

SQL CLR ストアドプロシージャのクライアントからの利用

SQL CLR ストアドプロシージャを利用するクライアントアプリケーションを作成する方法について説明します。

メッセージを返すストアドプロシージャ(基本編)

ここでは、「Hello World!」というメッセージを返す簡単なストアドプロシージャを作成してみましょう。

以下は、"HelloWorld"というストアドプロシージャを実行した結果です。 結果の確認は sqlcmd ユーティリティを使用しています。

```
1>sqlcmd -S (localdb)¥MSSQLLocalDB -E -d COBOLSample
2>exec HelloWorld
3>go
Hello World!
1>
```

作業手順は以下のとおりです。

1. SQL CLR ストアドプロシージャの プロジェクトの作成
2. SQL CLR ストアドプロシージャの手続きの記述 とビルド
3. SQL Server データベースプロジェクトの作成
4. SQL Server データベースプロジェクト のビルドと配置
5. SQL CLR ストアドプロシージャの実行

SQL CLR ストアドプロシージャの プロジェクトの作成

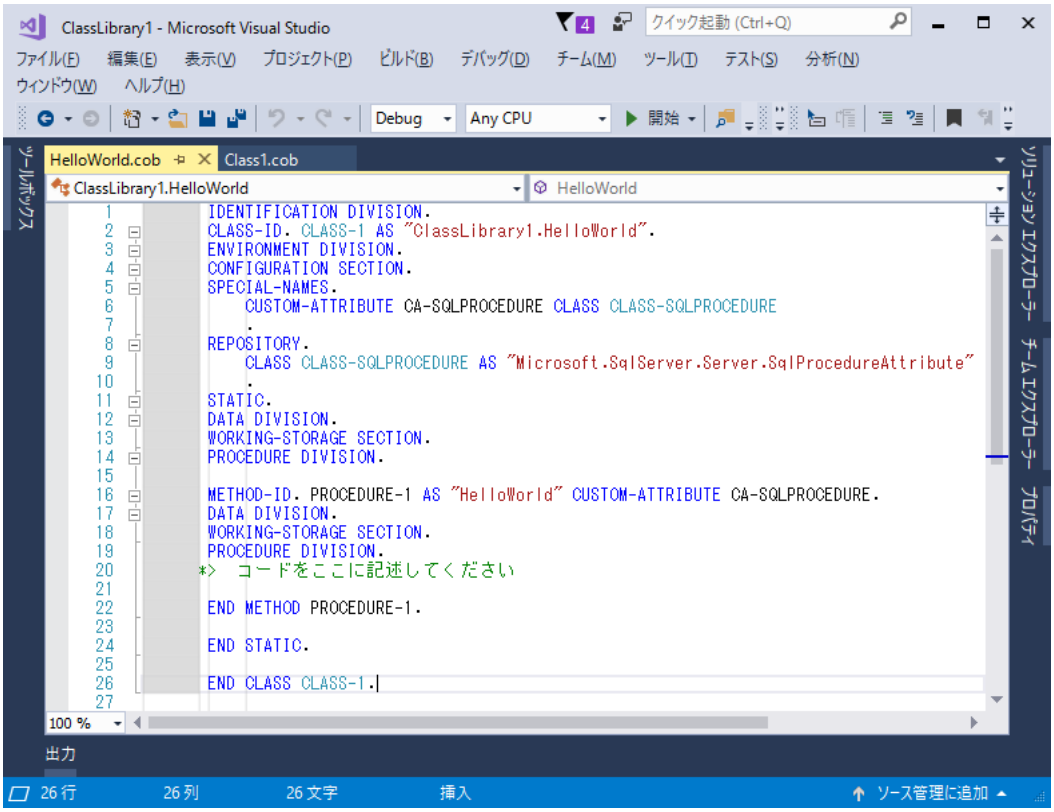
1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[プロジェクト]を選択します。
3. [新しいプロジェクト]ダイアログボックスが表示されます。
4. 左ペインで、[NetCOBOL for .NET]-[クラスライブラリ]を選択します。
5. [プロジェクト名]に、新規プロジェクトの名前を入力します。 プロジェクト名は「 ClassLibrary1 」とします。 (プロジェクト名は自由につけて構いません。)
6. [OK]ボタンをクリックします。

ソリューションエクスプローラーには、 プロジェクトの構成を表すツリーが表示されます。 ソリューションエクスプローラーが表示されない場合は、[表示]メニューから、 [ソリューション エクスプローラー]を選択すれば表示されます。

SQL CLR ストアドプロシージャの手続きの記述とビルド

SQL CLR ストアドプロシージャの手続きを記述します。

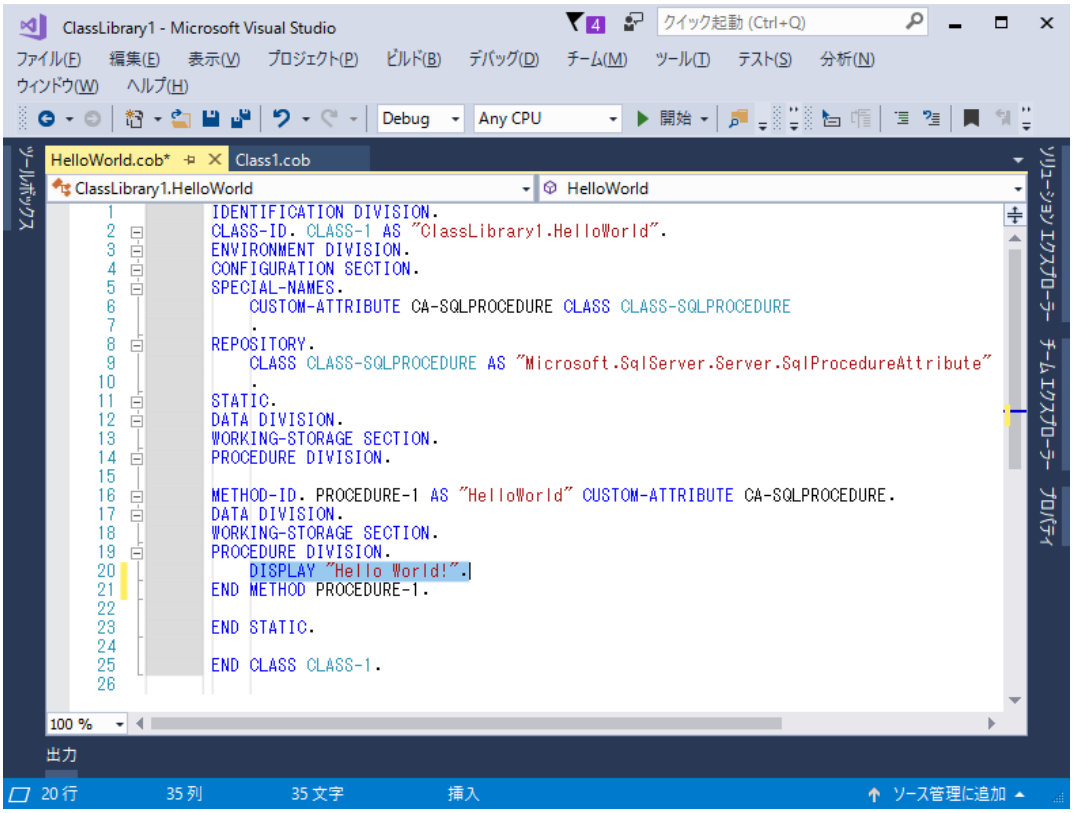
1. ソリューションエクスプローラーにおいて、プロジェクトを選択します。
2. Visual Studio の[プロジェクト]メニューから、 [新しい項目の追加]を選択します。
3. [新しい項目の追加]ダイアログボックスが表示されます。
4. 左ペイン から、[NetCOBOL for .NET]-[SQL CLR]- [ストアド プロシージャ]を選択します。
5. [ファイル名]に、新規 COBOL ソースファイルの名前を入力します。 ここでは、「HelloWorld.cob」とします。
6. [追加]ボタンをクリックします。 これで、ストアドプロシージャのための COBOL ソースファイルがプロジェクトに追加され、 ソースコードが表示されます。
7. 次に、 ストアドプロシージャのメソッド定義を STATIC 定義に追加します。 ここでは、END STATIC の直前に追加することにします。 なお、以下のメソッド定義は、テンプレートによって、PROCEDURE-1 というメソッド定義がすでに挿入されています。



PROCEDURE-1 メソッド定義の手続き部に、手続きを記述します。ここでは以下の 1 行を追加します。

```
        DISPLAY "Hello World!".
```

SQL CLR データベースオブジェクトで DISPLAY 文を使用した場合は、 クライアントにユーザ定義メッセージが送信されます。 詳細は、ユーザ定義メッセージを送信するを参照してください。



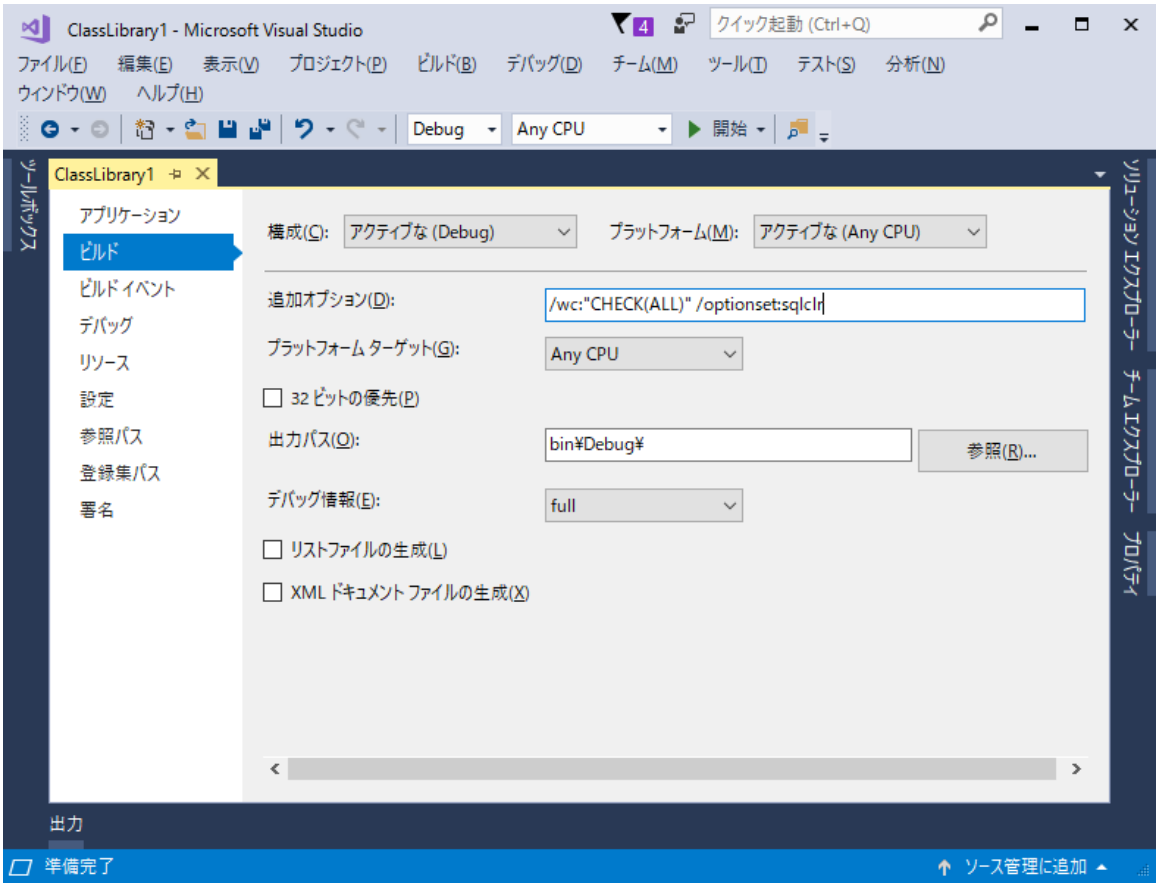
8. 上記の PROCEDURE-1 メソッドは、"HelloWorld"という外部名が指定されているため、 "HelloWorld" というストアドプロシージャ名としてデータベースに登録されます。 また、メソッド名段落に Microsoft.SqlServer.Server.SqlProcedureAttribute カスタム属性が付加されています。このカスタム属性によって、 このメソッドがストアドプロシージャであることを宣言しています。

   カスタム属性については、カスタム属性を使うを参照してください。

9. SQL CLR で指定する翻訳オプションセットを指定します。 Visual Studio の[プロジェクト]メニューから、 [ClassLibrary1 のプロパティ]を選択します。

10. [ビルド]ペインで、「追加オプション」に以下を追加します。

```
/optionset:sqlclr
```



11. Visual Studio の[ビルド]メニューから「ClassLibrary1 のビルド」を選択し、 ビルドを行います。

注意

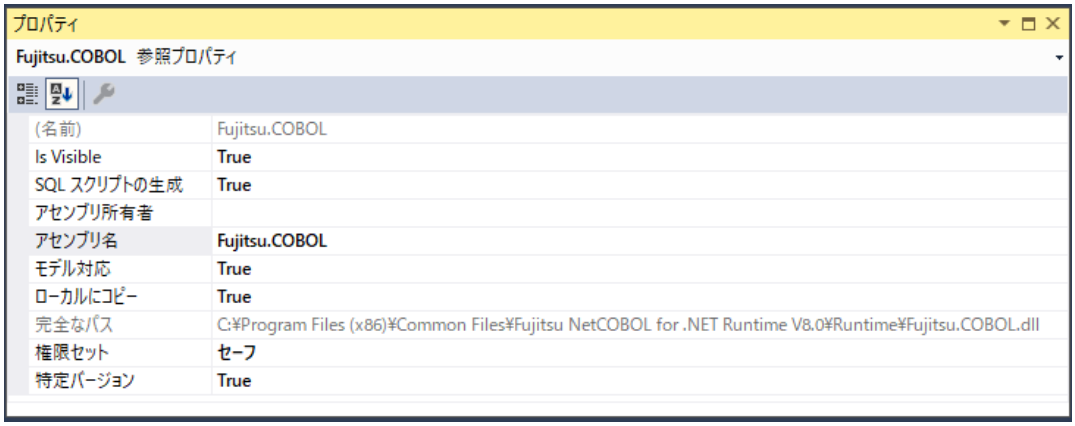
/optionset:sqlclr を指定しない場合以下のエラーが発生します。

```
SQL72014: .Net SqlClient Data Provider: メッセージ 6212、レベル 16、状態 1、行 1
CREATE ASSEMBLY failed because method 'HelloWorld' on type
'ClassLibrary1.HelloWorld' in safe assembly 'ClassLibrary1' is storing to a static field.
Storing to a static field is not allowed in safe assemblies.
```

SQL Server データベースプロジェクトの作成

1. Visual Studio の[ファイル]メニューから、 [追加]-[新しいプロジェクト]を選択します。
2. [新しいプロジェクトの追加]ダイアログボックスが表示されます。
3. 左ペインで、[SQL Server]-[SQL Server データベースプロジェクト]を選択します。
4. [名前]を入力します。ここでは、「COBOLSample」とし、[OK]ボタンをクリックします。
5. 「COBOLSample」を選択した状態で、Visual Studio の[プロジェクト]メニューから、[参照の追加]を選択します。
6. [参照マネージャー]ダイアログで、「SQL CLR ストアドプロシージャの プロジェクトの作成」で作成したライブラリ（ClassLibrary1 プロジェクト）と、 「Fujitsu.COBOL.dll」を選択します。 「SQL CLR ストアドプロシージャの プロジェクトの作成」で作成した ClassLibrary1 は、 「プロジェクト」-「ソリューション」から選択します。 「Fujitsu.COBOL.dll」は、[参照マネージャー]ダイアログの[参照]ボタンを選択し、 インストールされているパスの「Fujitsu.COBOL.dll」を追加します。 インストール先は、ランタイムアセンブリを登録するを参照してください。
7. ソリューションエックスプローラー上で追加した「Fujitsu.COBOL.dll」を選択し、 右クリックで表示されるコンテキストメニューからプロパティ画面を表示します。以下のように属性を変更します。「ClassLibrary1」の属性も同様の設定になっていることを確認してください。

Is Visible	True
SQL スクリプトの生成	True
モデル対応	True
ローカルにコピー	True



SQL Server データベースプロジェクトのビルドと配置

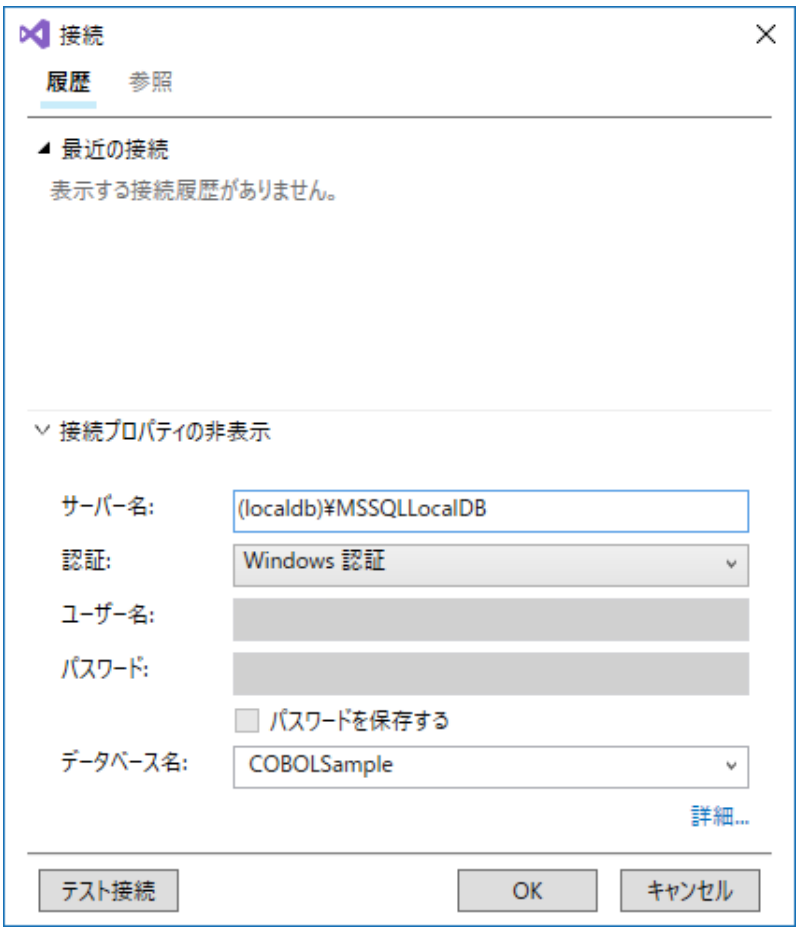
データベースプロジェクトを配置する際は、Fujitsu.COBOL.dll のポリシーファイルの関連付けをはずす必要があります。 関連つけられたままの場合、以下のエラーが発生します。ランタイムアセンブリを登録するを参照してください。

```
SQL72014: .Net SqlClient Data Provider: Msg 6586, Level 16, State 1, Line 1 Assembly 'Fujitsu.COBOL' could not be installed because existing policy would keep it from being used.
```

1. 「COBOLSample」を選択した状態で、Visual Studio の[ビルド]メニューから「COBOLSample のビルド」を選択し、 ビルドを行います。

2. 「COBOLSample」を選択した状態で、Visual Studio の[ビルド]メニューから「COBOLSample の公開」を選択します。

3. [データベースの公開]ダイアログボックスで、ターゲットデータベースの接続情報を指定します。 [編集]ボタンをクリックすると、[接続のプロパティ]ダイアログが表示されます。 サーバー名に“(localdb)¥MSSQLLocalDB”を指定し、 データベース名の選択または入力で“COBOLSample”を選択して、[OK]ボタンをクリックします。



注意

公開する際に、ポリシーを退避しておく必要があります。 詳細は、ランタイムアセンブリを登録するを参照してください。

4. [データベースの公開]ダイアログボックスの[公開]ボタンをクリックします。 これにより、手順 1 で作成した SQL CLR ストアドプロシージャがデータベースに配置されます。

SQL CLR ストアドプロシージャの実行

それでは、sqlcmd ユーティリティを使用して実際に、登録したストアドプロシージャを実行してみましょう。

NetCOBOL for .NET コマンドプロンプトウィンドウを開き、以下のコマンドを実行します。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB -d COBOLSample
```

配置先を変更した場合は、 sqlcmd コマンドの -S オプションにサーバ名を、-d オプションにはデータベース名を配置先にあわせて指定してください。

sqlcmd ユーティリティが起動したら、以下の SQL コマンドを実行し"HelloWorld"ストアドプロシージャを呼び出します。

```
1> exec HelloWorld
2> GO
```

正常に実行されると、結果は以下のように表示されます。

```
C:¥sqlcmd -S (localdb)¥MSSQLLocalDB -E -d COBOLSample
1> exec HelloWorld
2> go
Hello World!
1>
```

実行時に以下のエラーが発生する場合は、SQL CLR データベースオブジェクトを使用するための事前準備を参考に、SQL CLR を有効にしてください。

```
Execution of user code in the .NET Framework is disabled. Enable "clr enabled" configuration option.
```

パラメタとデータベース機能を使用するストアドプロシージャ(応用編)

次に、パラメタとデータベース機能を使用するストアドプロシージャを作成してみましょう。

ここでは、以下の形式の" ManageStockCLR "というストアドプロシージャを作成します。 このストアドプロシージャは、サンプルデータベースの STOCK 表を利用して、在庫管理対象のデータを取り出し、入庫または出庫を切り分けて在庫数量を計算し、再び表に格納します。

Transact-SQL での呼出し形式

```
[@return_state =]
ManageStockCLR
  [@INPUT_PRODUCT_NUMBER = ] input_product_number,
  [@INOUT_KIND = ] inout_kind,
  [@INOUT_QUANTITY = ] inout_quantity,
  [@UPDATED_QUANTITY = ] updated_quantity OUTPUT,
  [@ERROR_MESSAGE = ] error_message OUTPUT
```

パラメタ

- `[@INPUT_PRODUCT_NUMBER = ]` *input_product_number*

  製品番号を指定する入力パラメタです。*input_product_number* のデータ型は `smallint` です。

- `[@INOUT_KIND = ]` *inout_kind*

  入出区別を指定する入力パラメタです。 1(入庫)または 2(出庫)を指定します。*inout_kind* のデータ型は `smallint` です。

- `[@INOUT_QUANTITY = ]` *inout_quantity*

  入出数量を指定する入力パラメタです。*inout_quantity* のデータ型は `int` です。

- `[@UPDATED_QUANTITY = ]` *updated_quantity* `OUTPUT`

  更新された在庫量が設定される出力パラメタです。*updated_quantity* のデータ型は `int` です。

- `[@ERROR_MESSAGE = ]` *error_message* `OUTPUT`

  エラーが発生したときにメッセージが設定される出力パラメタです。*error_message* のデータ型は `nvarchar(max)`です。

戻り値

0(成功)または 0 以外(失敗)を返します。データ型は `int` です。

作業手順は以下のとおりです。

1. SQL CLR ストアドプロシージャのプロジェクトの作成
2. SQL CLR ストアドプロシージャの手続きの記述とビルド
3. SQL Server データベースプロジェクトの作成
4. SQL Server データベースプロジェクトのビルドと配置
5. SQL CLR ストアドプロシージャの実行

SQL CLR ストアドプロシージャのプロジェクトの作成

メッセージを返すストアドプロシージャ(基本編)の 「SQL CLR ストアドプロシージャのプロジェクトの作成」と同様の操作をして、 「 ManageStockCLR」というストアドプロシージャ用プロジェクトを作成します。 プロジェクト名は自由につけて構いません。

SQL CLR ストアドプロシージャの手続きの記述とビルド

1. メッセージを返すストアドプロシージャ(基本編)の 「SQL CLR ストアドプロシージャの手続きの記述とビルド」を参考に、 「ストアド プロシージャ」テンプレートから、 新規 COBOL ソースファイルを追加します。ここでは、「ManageStockCLR.cob」というファイル名を指定します。

2. テンプレートによってすでに STATIC 段落に定義されている PROCEDURE-1(ManageStockCLR)メソッド定義を以下のように変更します。

```
       IDENTIFICATION DIVISION.
       CLASS-ID. CLASS-1 AS "ManageStockCLR.ManageStockCLR".
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.                                                    *> [1]
           CUSTOM-ATTRIBUTE CA-SQLPROCEDURE CLASS CLASS-SQLPROCEDURE
           CUSTOM-ATTRIBUTE CA-OUT CLASS CLASS-OUT
           .
       REPOSITORY.                                                       *> [2]
           CLASS CLASS-SQLPROCEDURE AS "Microsoft.SqlServer.Server.SqlProcedureAttribute"
           CLASS CLASS-OUT AS "System.Runtime.InteropServices.OutAttribute"
           CLASS CLASS-STRING AS "System.String"
           .
       STATIC.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
       METHOD-ID. PROCEDURE-1 AS "ManageStockCLR" CUSTOM-ATTRIBUTE CA-SQLPROCEDURE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.                                          *> [3]
           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01 SQLSTATE            PIC X(5).
       01 SQLCODE             PIC S9(9) COMP-5.
       01 SQLMSG              PIC X(256).
       01 INVENTORY_QUANTITY    PIC S9(9) COMP-5.
       01 PRODUCT_NUMBER       PIC S9(4) COMP-5.
           EXEC SQL END DECLARE SECTION END-EXEC.
       LINKAGE SECTION.                                                  *> [4]
       01 INPUT_PRODUCT_NUMBER  BINARY-SHORT SIGNED.
       01 INOUT_KIND           BINARY-SHORT SIGNED.
       01 INOUT_QUANTITY       BINARY-LONG SIGNED.
       01 UPDATED_QUANTITY     BINARY-LONG SIGNED CUSTOM-ATTRIBUTE CA-OUT.
       01 ERROR_MESSAGE        OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE CA-OUT.
       01 RETURN_STATE         BINARY-LONG.
       PROCEDURE DIVISION USING BY VALUE INPUT_PRODUCT_NUMBER
                                         INOUT_KIND
                                         INOUT_QUANTITY
                                BY REFERENCE UPDATED_QUANTITY
                                             ERROR_MESSAGE
                                RETURNING RETURN_STATE.                  *> [5]

           EXEC SQL WHENEVER SQLERROR GO TO :EXIT-PROC END-EXEC.
           EXEC SQL WHENEVER NOT FOUND GO TO :EXIT-PROC END-EXEC.

           MOVE 0 TO RETURN_STATE.
           MOVE INPUT_PRODUCT_NUMBER TO PRODUCT_NUMBER.

           EXEC SQL
             SELECT QOH INTO  :INVENTORY_QUANTITY FROM STOCK
             WHERE GNO = :PRODUCT_NUMBER
           END-EXEC

           EVALUATE INOUT_KIND
           WHEN 1
             ADD INOUT_QUANTITY TO INVENTORY_QUANTITY
           WHEN 2
             SUBTRACT INOUT_QUANTITY FROM INVENTORY_QUANTITY
           WHEN OTHER
             GO TO EXIT-PROC
           END-EVALUATE
```

```
        EXEC SQL
          UPDATE STOCK SET QOH = :INVENTORY_QUANTITY
                    WHERE GNO = :PRODUCT_NUMBER
        END-EXEC.

    EXIT-PROC.
        MOVE INVENTORY_QUANTITY TO UPDATED_QUANTITY.

        EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.
        EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.

        SET ERROR_MESSAGE TO SQLMSG.
        MOVE SQLCODE TO RETURN_STATE.

    END METHOD PROCEDURE-1.

    END STATIC.

    END CLASS CLASS-1.
```

プログラムの説明

[1] 特殊名段落(SPECIAL-NAMES)

CA-SQLPROCEDURE カスタム属性は、ストアドプロシージャであることを示すための属性で、メソッドに付加します。

CA-OUT カスタム属性は、ストアドプロシージャの出力専用パラメタであることを示す属性で、連絡節のデータ項目に付加します。

[2] リポジトリ段落(REPOSITORY)

CLASS-SQLPROCEDURE は、CA-SQLPROCEDURE カスタム属性の定義で使用する Microsoft.SqlServer.Server.SqlProcedureAttribute クラスを定義しています。

CLASS-OUT は、CA-OUT カスタム属性の定義で使用する System.Runtime.InteropServices.OutAttribute クラスを定義しています。

CLASS-STRING は、メソッドで使用する System.String クラスを定義しています。

[3] 作業場所節(WORKING-STORAGE SECTION)

メソッドのローカル変数の宣言をします。ここでは、埋込み SQL 宣言節を記述し STOCK 表にアクセスするためのホスト変数を宣言します。 ホスト変数宣言の規則については、COBOL 文法書の「8.2.2 ホスト変数定義」を参照してください。

[4] 連絡節(LINKAGE SECTION)

メソッドのパラメタを宣言します。

- データ項目名は、Transact-SQL 上の ストアドプロシージャやユーザ定義関数のパラメタ名としてそのまま使用されるため、 COBOL の利用者語の規則に加えて、Transact-SQL の標準識別子の規則に従わなければなりません。たとえば、COBOL の利用者語では指定可能な"-"(ハイフン)をデータ名に使用することはできません。 (Transact-SQL 上では、"@"(アットマーク)のプレフィクスが付加されます。)

Transact-SQL の標準識別子の規則については、 SQL Server オンラインブックのデータベース識別子を参照してください。

- SQL CLR ストアドプロシージャやユーザ定義関数のパラメタや戻り値に使用できる型は、SQL Server 型と同等の SQL Server CLR データ型または、 .NET 基本データ型である必要があります。 " ManageStockCLR"では int 型、smallint 型、nvarchar(max)型を使用するため、 NetCOBOL for .NET では以下のように対応付けられます。

SQL Server 型	SQL Server CLR データ型	.NET Framework .NET 基本データ型
int	System.Data.SqlTypes.SqlInt32 型	BINARY-LONG SIGNED
smallint	System.Data.SqlTypes.SqlInt16 型	BINARY-SHORT SIGNED
nvarchar(max)	System.Data.SqlTypes.SqlString 型 または System.Data.SqlTypes.SqlChars 型	System.String 型

このサンプルでは、.NET 基本データ型を使用します。

SQL Server 型と CLR データ型の対応については、 SQL Server オンラインブックの CLR パラメーター データのマッピングを参照してください。

また、COBOL データ型と.NET 基本データ型の対応については.NET データ型の COBOL 表現を、パラメタ型とホスト変数との対応については SQL CLR データベースオブジェクトのパラメタ型とホスト変数の対応をそれぞれ参照してください。

- 出力パラメタとなるデータ項目には、CA-OUT カスタム属性を付加しています。入力パラメタ、出力パラメタの指定方法については、次の「[5] 手続き部(PROCEDURE DIVISION)」で説明します。

[5] 手続き部(PROCEDURE DIVISION)

ストアドプロシージャの手続きを記述します。

- 手続き部の見出しでは、USING 句以降の記述がそのままストアドプロシージャのインタフェースになります。 値渡し(BY VALUE 指定)のパラメタはストアドプロシージャでは入力パラメタになり、参照渡し(BY REFERENCE 指定)のパラメタは出力パラメタになります。なお、参照渡しパラメタのデータ項目に CA-OUT(System.Runtime.InteropServices.OutAttribute)カスタム属性が指定されている場合は出力専用パラメタとして、指定されていない場合は入出力パラメタとして使用することができます。 戻り値(RETURNING)が指定されている場合は、ストアドプロシージャの戻り値になります。
- 手続き部分には、埋込み SQL 文を利用したデータの操作を行う手続きを記述します。

クライアントアプリケーションで COBOL データベース機能を使用する場合、埋込み SQL 文の CONNECT 文を使用してサーバに接続する必要がありますが、 SQL CLR データベースオブジェクトはクライアントから接続されたコンテキストを利用し SQL Server 内部で直接データベースに接続します。これを SQL CLR ではコンテキスト接続と呼んでいます。 COBOL のデータベース機

能で SQL 文を実行する場合は自動的にコンテキスト接続を行うため、 接続(CONNECT)/変更(SET CONNECTION)/切断(DISCONNECT)といった処理は不要です。

SQL CLR データベースオブジェクトでデータアクセスを行う場合は、CLR データベース オブジェクトからのデータ アクセスも参考にしてください。 コンテキスト接続については、SQL Server オンラインブックのコンテキスト接続を参照してください。

3. 手続きを記述後に、ビルドしてください。

SQL Server データベースプロジェクトの作成

メッセージを返すストアドプロシージャ(基本編)の 「SQL Server データベースプロジェクトの作成」と同様に、SQL Server データベースプロジェクトを作成してください。

SQL CLR ストアドプロシージャのビルドと配置

メッセージを返すストアドプロシージャ(基本編)の 「SQL Server データベースプロジェクトのビルドと配置」と同様の操作をして、 データベースに公開します。

SQL CLR ストアドプロシージャの実行

NetCOBOL for .NET コマンドプロンプトウィンドウを開き、 以下のコマンドを実行します。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB -d COBOLSample
```

配置先を変更した場合は、 sqlcmd コマンドの -S オプションにサーバ名を、 -d オプションにはデータベース名を配置先にあわせて指定してください。 sqlcmd ユーティリティが起動したら、以下の SQL コマンドを実行します。

```

-----------------------------------------------------------------------------------------
-- Stored procedure
-----------------------------------------------------------------------------------------
DECLARE @UPDATED_QUANTITY int
DECLARE @MESSAGE nvarchar(max)
DECLARE @RETURN_STATE int
SET @UPDATED_QUANTITY =0
SET @MESSAGE=''
EXEC @RETURN_STATE =
          ManageStockCLR 110, 1, 20, @UPDATED_QUANTITY OUTPUT, @MESSAGE OUTPUT
SELECT @RETURN_STATE, @UPDATED_QUANTITY, @MESSAGE

```

上記のスクリプトの例では、ManageStockCLR の第 1 パラメタに正常な値(110)を設定していますが、 誤った値(112 など)を指定した場合は、 @RETURN_STATE に 0 以外の値が設定され、@MESSAGE にエラーメッセージが返却されます。

NCOBU00808

SQL CLR ストアドプロシージャのクライアントからの利用

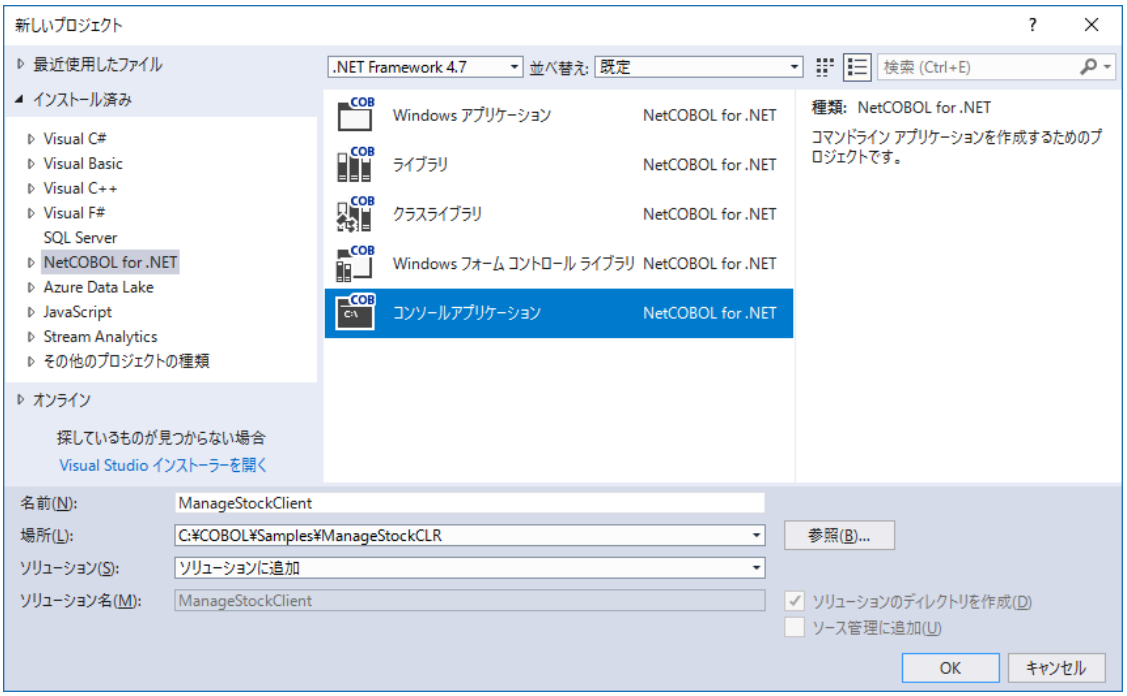
SQL CLR ストアドプロシージャを利用するクライアントアプリケーションを作成してみましょう。 ここでは、パラメタとデータベース機能を使用するストアドプロシージャ(応用編)で作成した SQL CLR ストアドプロシージャを利用するコンソールアプリケーションを作成します。

作業手順は以下のとおりです。

1. コンソールアプリケーションプロジェクトの作成
2. クライアントアプリケーションの手続きの記述
3. クライアントアプリケーションの実行環境の設定
4. クライアントアプリケーションのデバッグ
5. クライアントアプリケーションの実行

コンソールアプリケーションプロジェクトの作成

1. このサンプルでは、パラメタとデータベース機能を使用するストアドプロシージャ(応用編)で使用したソリューションをそのまま利用します。 プロジェクトが開かれていない場合は、Visual Studio の[ファイル]メニューから、 [開く]-[プロジェクト/ソリューション]を選択して、パラメタとデータベース機能を使用するストアドプロシージャ(応用編)で作成したソリューションを開いてください。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[プロジェクト]を選択します。
3. [新しいプロジェクト]ダイアログボックスが表示されるので、「コンソールアプリケーション」を選択し、プロジェクト名は「ManageStockClient」と入力します。(プロジェクト名は自由につけて構いません。) 現在開いているソリューションに新しいプロジェクトを追加するため、 [ソリューション]では、[ソリューションに追加]を選択します。



4. [OK]ボタンをクリックします。 これで、クライアントアプリケーションの構築に必要なプロジェクトが作成されました。

クライアントアプリケーションの手続きの記述

クライアントアプリケーションの手続きを記述します。

1. ソリューションエクスプローラーには、プロジェクトの構成をあらわすツリーが表示されます。 ツリー上にある Main.cob というファイルを開きソースコードを表示します。

2. 以下のようにソースコードを変更します。

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. MAIN AS "ManageStockClient.Main".
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
       REPOSITORY.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
          EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01 SQLSTATE            PIC   X(5).
       01 SQLINFOA.
         02 SQLERRD  PIC S9(9) COMP-5 OCCURS 6 TIMES.
       01 製品番号   PIC S9(4) COMP-5.
       01 入出区別   PIC S9(4) COMP-5.
       01 入出数量   PIC S9(9) COMP-5.
       01 在庫量     PIC S9(9) COMP-5.
       01 メッセージ PIC X(256).
          EXEC SQL END   DECLARE SECTION END-EXEC.
       01 終了要求   PIC X(1).
       PROCEDURE DIVISION.
          EXEC SQL CONNECT TO DEFAULT END-EXEC.

          PERFORM TEST AFTER UNTIL 終了要求  = "Y"
            DISPLAY "製品番号を入力してください"
            ACCEPT  製品番号
            DISPLAY "入出区別(1or2)を入力してください"
            ACCEPT  入出区別
            DISPLAY "入出数量を入力してください"
            ACCEPT  入出数量

            EXEC SQL
              CALL  ManageStockCLR (:製品番号, :入出区別, :入出数量, :在庫量, :メッセージ)
            END-EXEC

            IF SQLERRD(1) NOT = 0
              DISPLAY メッセージ
            ELSE
              DISPLAY "在庫量: " 在庫量
            END-IF
            DISPLAY "在庫管理を終了しますか？(Y/N)"
            ACCEPT  終了要求
          END-PERFORM
          EXEC SQL COMMIT WORK END-EXEC
          EXEC SQL DISCONNECT DEFAULT END-EXEC
       END PROGRAM MAIN.
```

プログラムの説明

クライアントアプリケーションから COBOL のデータベース機能を使用して SQL CLR ストアドプロシージャを呼び出す場合、 特別な構文は必要ありません。 Transact-SQL で記述されたストアドプロシージャと同様な方法で呼び出すことができます。

- COBOL のデータベース機能については、データベースアクセスを参照してください。
- ストアドプロシージャの呼び出し方法については、ストアドプロシージャの呼出しを参照してください。

3. COBOL ソースプログラム(Main.cob)の変更を保存します。 保存する場合、ツールバーの[Main.cob の保存]アイコン( )をクリックします。

クライアントアプリケーションの実行環境の設定

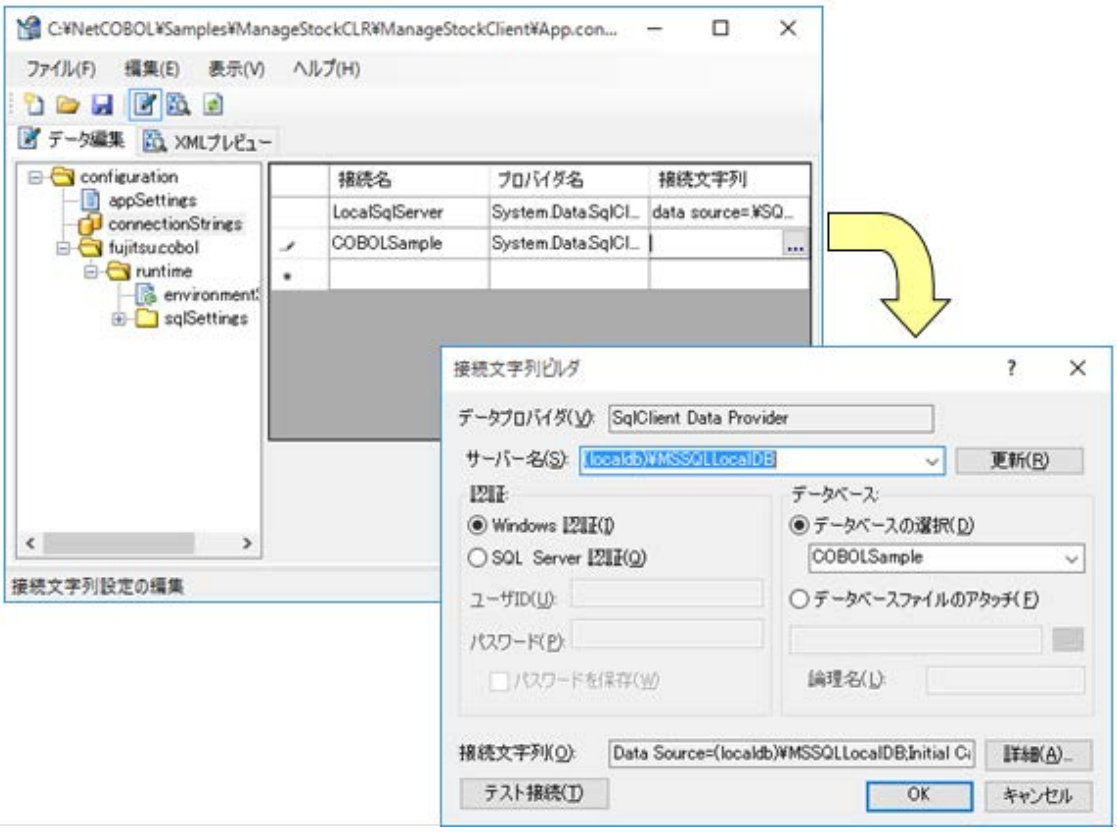
次に実行環境設定ユーティリティを使用して SQL 情報の設定を行います。

1. ソリューションエクスプローラーで、"ManageStockClient"プロジェクトを選択し、Visual Studio の[プロジェクト]メニューから、[COBOL 実行環境設定の編集]を選択し、実行環境設定ユーティリティを開きます。(このとき、App.config というファイルがプロジェクトに追加されます。)

2. まず、"connectionStrings"セクションを選択し、接続文字列の設定を行います。

   ここでは、接続名を"COBOLSample"とします。接続文字列には以下の情報を設定します。

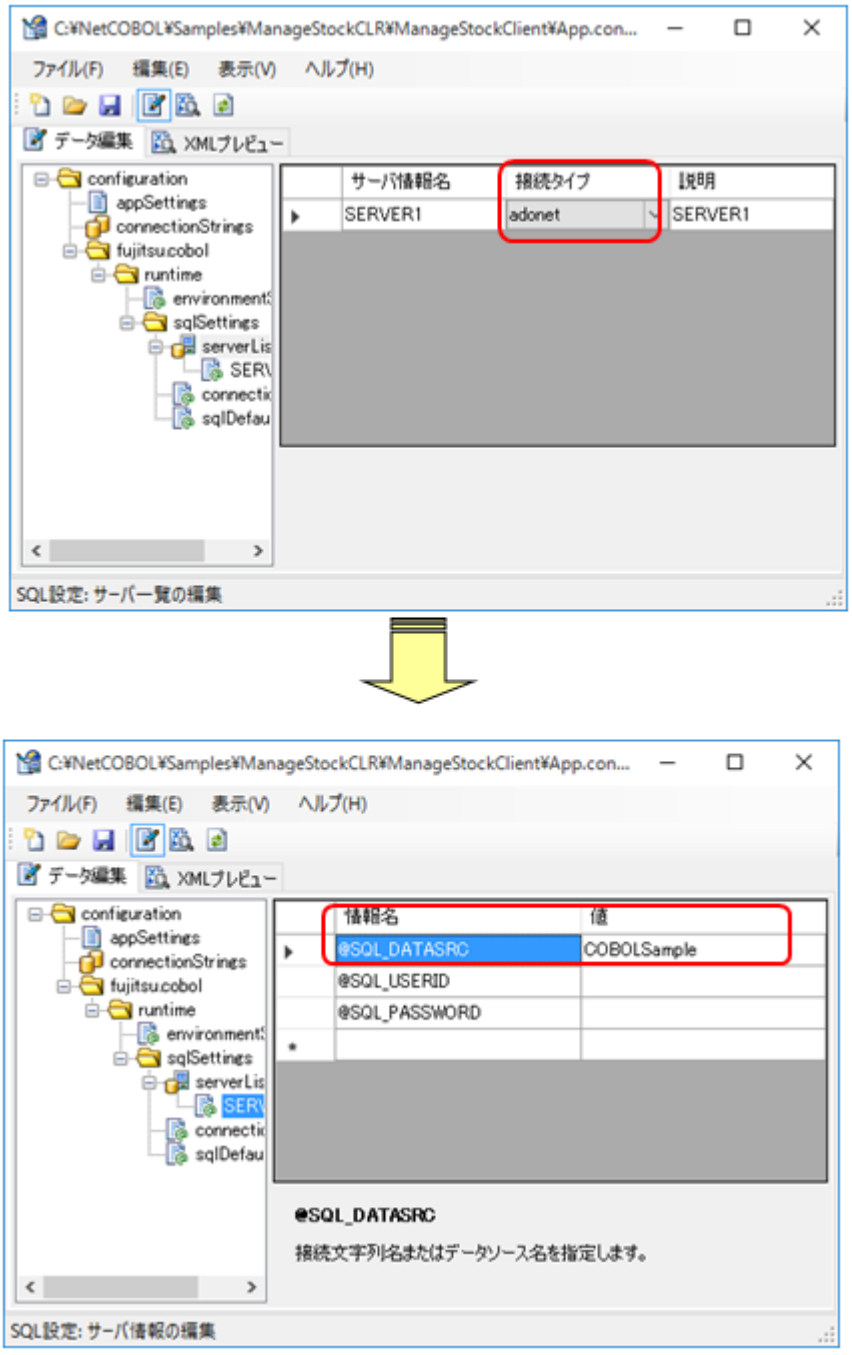
サーバ名	(localdb)¥MSSQLLocalDB
認証方法	Windows 認証
データベース名	COBOLSample



   詳細な手順については、接続文字列の設定方法(ADO.NET)を参照してください。

3. 次に SQL 設定を行います。実行環境設定ユーティリティの[編集]メニューから[SQL 設定の追加]を選択し、"SERVER1"という名前のサーバ情報を追加します。ADO.NET 接続を行うため、"serverList"セクションを選択して、サーバ情報"SERVER1"の[接続タイプ]が"adonet"に設定されていることを確認してください。

   再び、"SERVER1"セクションを選択し、@SQL_DATASRC に、接続名の"COBOLSample"を指定します。また、このサンプルでは Windows 認証を行うため、@SQL_USERID および@SQL_PASSWORD は何も設定しません。



詳細な手順については、実行環境設定ユーティリティの使い方を参照してください。

4. 実行環境設定ユーティリティの[ファイル]メニューから[保存]を選択して、SQL 情報の設定を終了します。

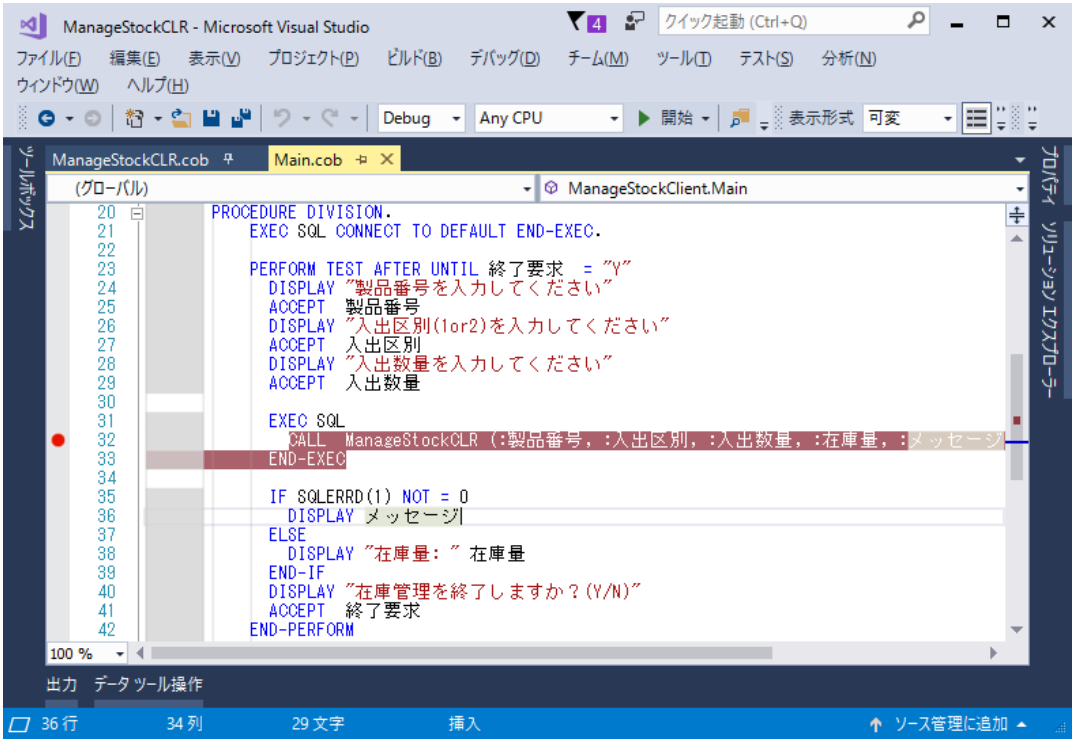
クライアントアプリケーションのデバッグ

クライアントアプリケーションから SQL CLR ストアドプロシージャ をシームレスにデバッグする方法を説明します。

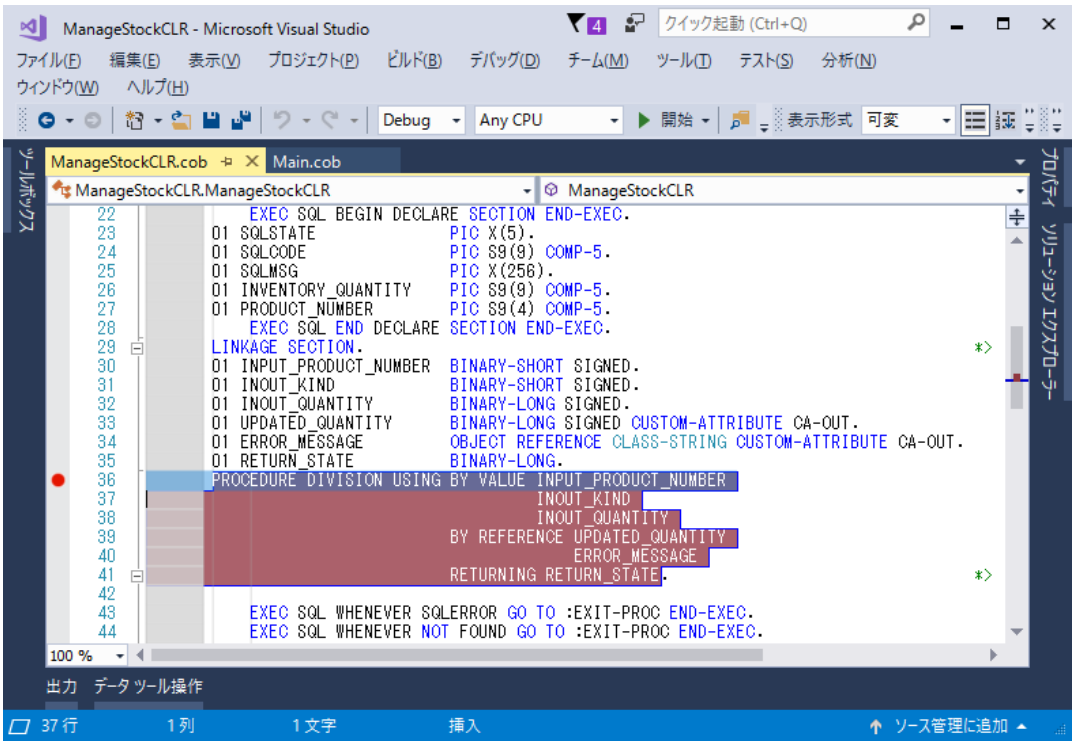
1. ソリューションエクスプローラーで、"ManageStockClient"プロジェクトを選択し、Visual Studio の[プロジェクト]メニューから[ManageStockClient のプロパティ]を選択し、プロジェクトのプロパティページを開きます。

[デバッグ]タブを選択し、[SQL Server デバッグを有効にする]をチェックし、 SQL Server のデバッグを有効にします。

2. 以下の、ストアドプロシージャ" ManageStockCLR"を呼び出している部分にブレークポイントを設定しておきます。



パラメタとデータベース機能を使用するストアドプロシージャ(応用編)で作成した"ManageStockCLR" プロジェクトからソースコード(ManageStockCLR.cob)を開き、"PROCEDURE DIVISION"にもブレークポイントを設定しておきます。

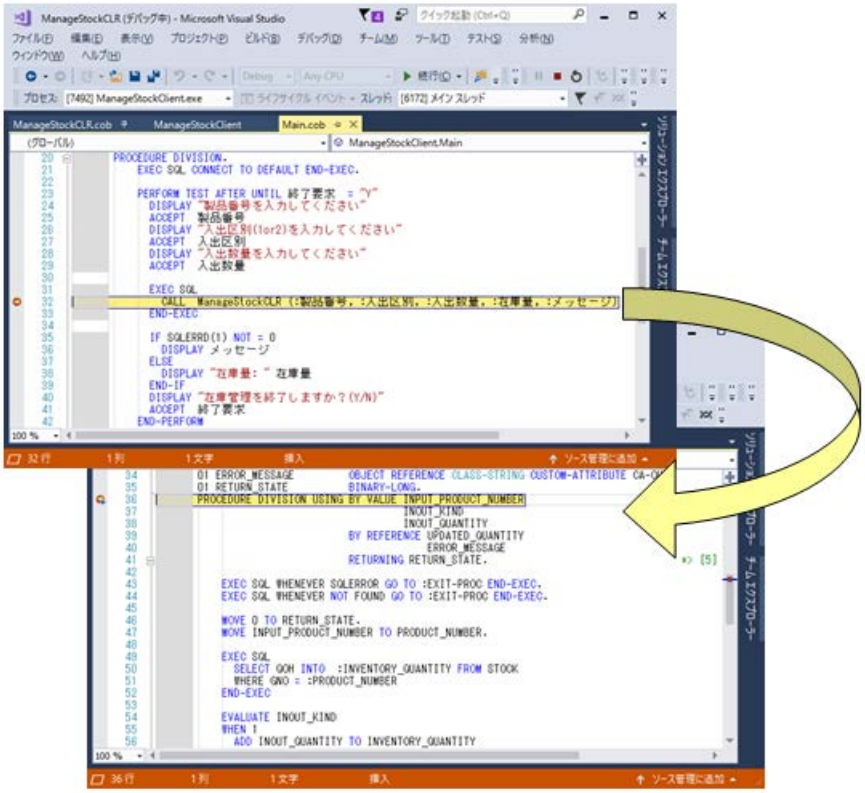


3. ManageStockClient を右クリックし、「スタートアッププロジェクトに設定」を選択します。 [デバッグ]メニューから[デバッグの開始]を選択すると、 デバッガ配下でアプリケーションが実行されます。

注意

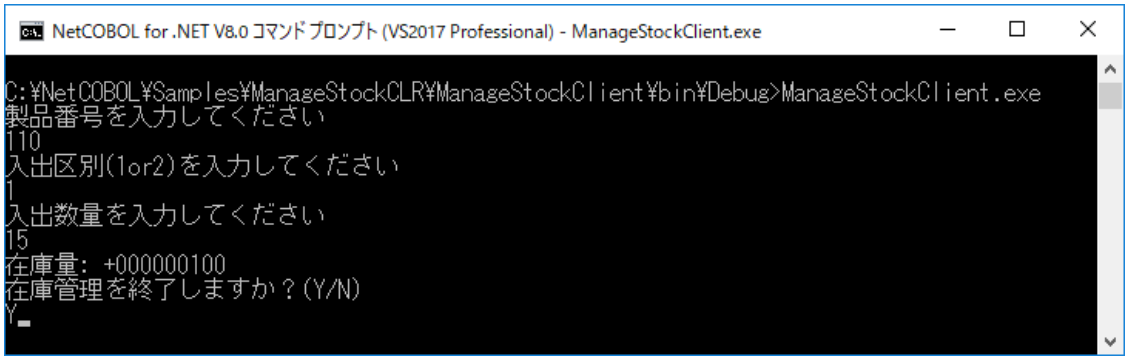
Visual Studio 2017 の[デバッグ]メニューから[プロセスにアタッチ]を選択し、 [使用可能なプロセス]から、sqlserver.exe プロセスを選択し、アタッチ後にデバッグ開始してください。

4. 先程設定した、ブレークポイントに到達したところで、[デバッグ]メニューから[続行]を選択すると、SQL CLR ストアドプロシージャ" ManageStockCLR"メソッドにステップインすることができます。



クライアントアプリケーションの実行

コマンドプロンプトを表示してクライアントアプリケーションを実行すると、 実行結果は以下のように表示されます。



NCOBU00809

SQL CLR ユーザ定義関数の作成

ここでは、簡単なサンプルを例にして、 Visual Studio で、 SQL CLR ユーザ定義関数を作成する手順について説明します。

SQL CLR ユーザ定義関数 についての詳細は、SQL Server オンラインブックのユーザー定義関数を参照してください。

このセクションの内容

スカラ値を返すユーザ定義関数の作成

ここでは、スカラ値を返すユーザ定義関数を作成してみましょう。

以下の形式の"GoodsName"というユーザ定義スカラ値関数を作成します。サンプルデータベースの STOCK 表を利用して、製品番号をパラメタに指定して製品名(文字列)を返すユーザ定義関数を作成します。

Transact-SQL での呼出し形式

```
GoodsName(product_number)
```

パラメタ

- *product_number*

  製品番号を指定する入力パラメタです。*product_number* のデータ型は `smallint` です。

戻り値

製品名を返します。データ型は `nchar(20)` です。

SQL CLR ユーザ定義スカラ値関数 についての詳細は、SQL Server オンラインブックの方法 : スカラー値のユーザー定義関数の使用を参照してください。

作業手順は以下のとおりです。

1. SQL CLR ユーザ定義スカラ値関数の プロジェクトの作成
2. SQL CLR ユーザ定義スカラ値関数の手続きの記述とビルド
3. SQL Server データベースプロジェクトの作成
4. SQL Server データベースプロジェクト のビルドと配置
5. SQL CLR ユーザ定義スカラ値関数の実行

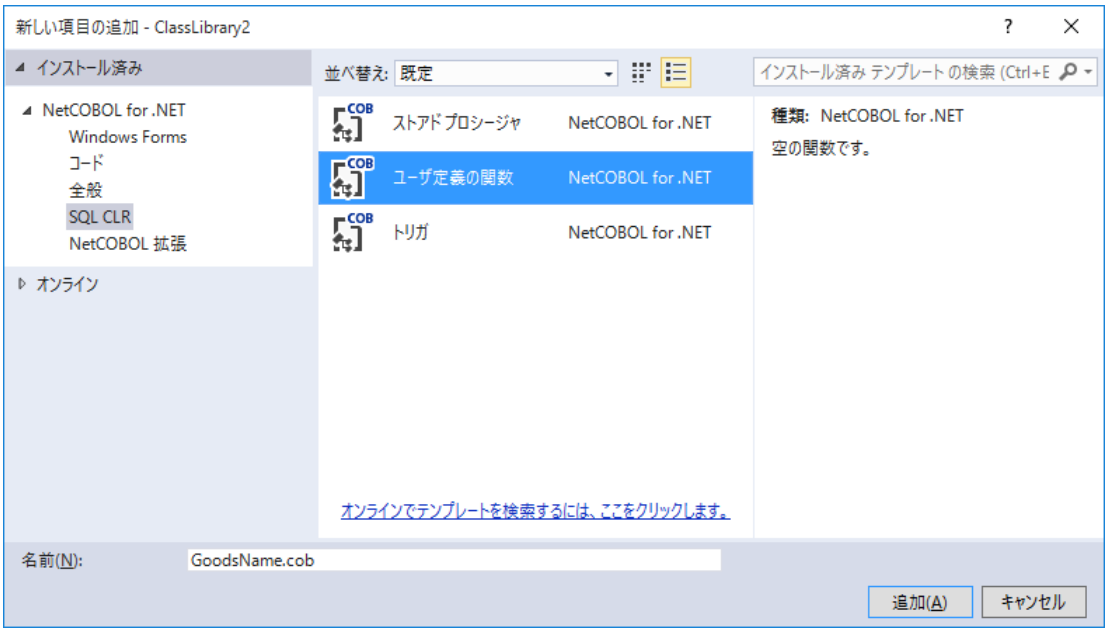
SQL CLR ユーザ定義スカラ値関数の プロジェクトの作成

メッセージを返すストアドプロシージャ(基本編)の 「SQL CLR ストアドプロシージャのプロジェクトの作成」と同様の操作をして、 「ClassLibrary2」を作成します。

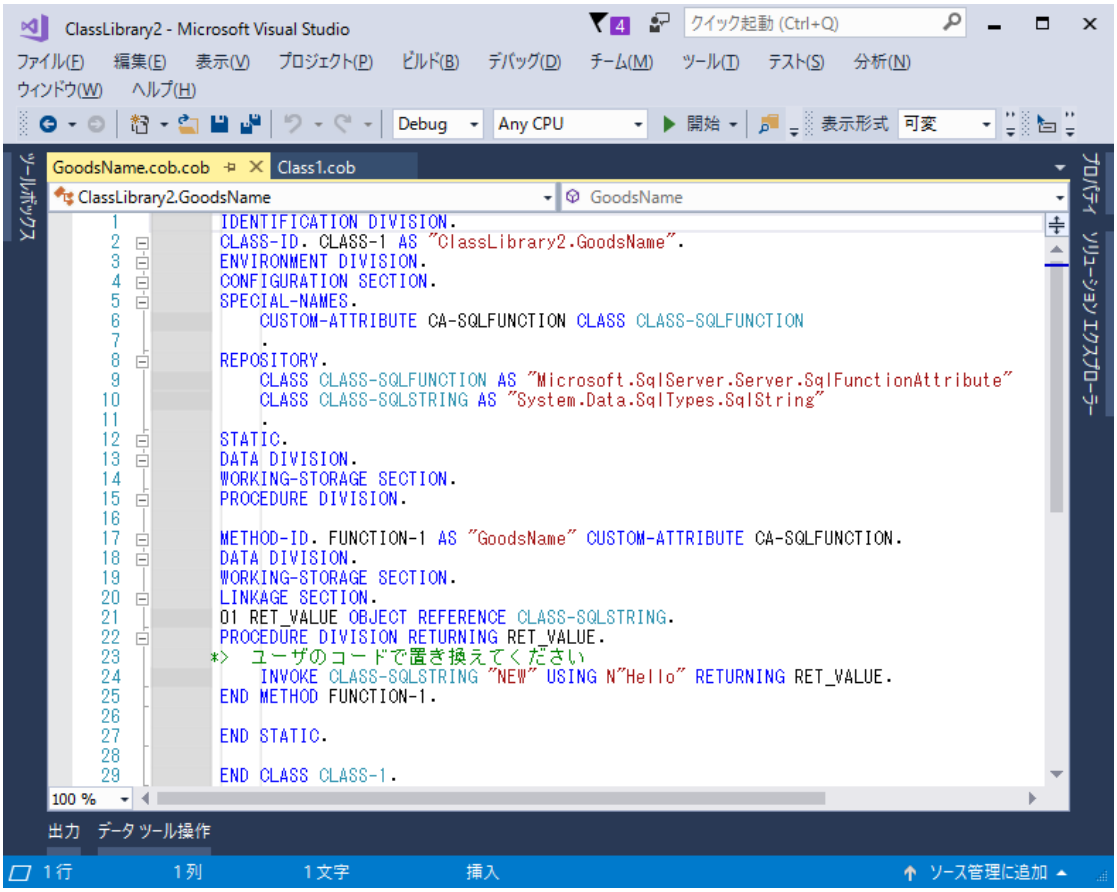
SQL CLR ユーザ定義スカラ値関数の手続きの記述とビルド

1. メッセージを返すストアドプロシージャ(基本編)の 「SQL CLR ストアドプロシージャの手続きの記述とビルド」を参考に、 [新しい項目の追加]ダイアログボックスを表示します。

SQL CLR ユーザ定義関数を作成するため、 左ペインから[NetCOBOL for .NET]-[SQL CLR]- 「ユーザ定義の関数」テンプレートを選択し、 新規 COBOL ソースファイルを追加します。 ここでは、「GoodsName.cob」というファイル名を指定します。



2. テンプレートによってすでに STATIC 段落に定義されている FUNCTION-1(GoodsName)メソッド定義が挿入されているため、 このメソッドを目的のユーザ定義スカラ値関数にあわせて変更します。 デフォルトでは、"Hello"という文字列を返すユーザ定義スカラ値関数のソースコードが記述されています。



3. 作成するユーザ定義関数はデータアクセスを行うため、 環境部(ENVIRONMENT DIVISION)の特殊名段落(SPECIAL-NAMES)およびリポジトリ段落(REPOSITORY)を変更し、 テンプレートによってあらかじめ記述されている CA-SQLFUNCTION(Microsoft.SqlServer.Server.SqlFunctionAttribute) カス

タム属性 の DataAccess 名前付きパラメタに Read を設定します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-SQLFUNCTION CLASS CLASS-SQLFUNCTION
                    PROPERTY PROP-DATAACCESS IS PROP-READ OF ENUM-DATAACCESSKIND
       ...
REPOSITORY.

   CLASS CLASS-SQLFUNCTION AS "Microsoft.SqlServer.Server.SqlFunctionAttribute"
   ENUM ENUM-DATAACCESSKIND AS "Microsoft.SqlServer.Server.DataAccessKind"
    PROPERTY PROP-DATAACCESS AS "DataAccess"
    PROPERTY PROP-READ AS "Read"
       ...
```

また、関数の戻り値の型を SQL Server 型の nchar(20)として定義するため、Microsoft.SqlServer.Server.SqlFacetAttribute カスタム属性を追加し、IsFixedLength 名前付きパラメタに B'1' (True)、MaxSize 名前付きパラメタに 20 を設定し、 CA-FIXEDLEN20 という カスタム属性を定義しています。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.

SPECIAL-NAMES.
       ...
   CUSTOM-ATTRIBUTE CA-FIXEDLEN20  CLASS CLASS-SQLFACET
                    PROPERTY PROP-ISFIXEDLENGTH IS B'1'
                    PROPERTY PROP-MAXSIZE IS 20
       ...
REPOSITORY.
       ...
   CLASS CLASS-SQLFACET AS "Microsoft.SqlServer.Server.SqlFacetAttribute"
    PROPERTY PROP-ISFIXEDLENGTH AS "IsFixedLength"
    PROPERTY PROP-MAXSIZE AS "MaxSize"
       ...
```

4. メソッド本体は以下のように変更します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
       ...
REPOSITORY.
       ...
   CLASS CLASS-SQLSTRING AS "System.Data.SqlTypes.SqlString"
       ...
STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.


METHOD-ID. FUNCTION-1 AS "GoodsName" CUSTOM-ATTRIBUTE CA-SQLFUNCTION.
DATA DIVISION.
WORKING-STORAGE SECTION.                                          *> [1]
   EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE PIC X(5).
01 SQLMSG PIC X(256).
01 GNO   PIC S9(9) COMP-5.
01 GOODS PIC X(20).
    EXEC SQL END DECLARE SECTION END-EXEC.
LINKAGE SECTION.                                                  *> [2]
01 PRODUCT_NUMBER BINARY-SHORT SIGNED.
01 RET_VALUE OBJECT REFERENCE CLASS-SQLSTRING CUSTOM-ATTRIBUTE CA-FIXEDLEN20.
PROCEDURE DIVISION USING BY VALUE PRODUCT_NUMBER RETURNING RET_VALUE.       *> [3]
   MOVE PRODUCT_NUMBER TO GNO.
    EXEC SQL
       SELECT GOODS INTO :GOODS FROM STOCK WHERE GNO = :GNO
    END-EXEC
    SET RET_VALUE TO CLASS-SQLSTRING::"NEW"(GOODS).

END METHOD FUNCTION-1.
```

プログラムの説明

[1] 作業場所節(WORKING-STORAGE SECTION)

メソッドのローカル変数を宣言します。 ここでは、埋込み SQL 宣言節を記述し STOCK 表にアクセスするためのホスト変数を宣言します。

[2] 連絡場所節(LINKAGE SECTION)

メソッドのパラメタを宣言します。

PRODUCT_NUMBER は、smallint 型のパラメタであるため、BINARY-SHORT SIGNED を使用しています。

RET_VALUE は、nchar(20)型の戻り値とするため、 SQL Server CLR データ型の System.Data.SqlTypes.SqlString 型をそのまま使用し、 環境部(ENVIRONMENT DIVISION)で定義した、カスタム属性 CA-FIXEDLEN20 を付加しています。

パラメタの名前の規則や、 パラメタや戻り値に指定できる型については、パラメタとデータベース機能を使用するストアドプロシージャ(応用編)の 「SQL CLR ストアドプロシージャの手続きの記述」 を参照してください。

[3] 手続き部(PROCEDURE DIVISION)

埋込み SQL 文を使用して実行したクエリの結果を、 戻り値(RETURNING)データ RET_VALUE に転記します。

5. SQL CLR で指定する翻訳オプションセットを指定します。 Visual Studio の[プロジェクト]メニューを選択し、 [ClassLibrary2 のプロパティ]を選択します。
6. [ビルド]ペインで、「追加オプション」に以下を追加します。

```
/optionset:sqlclr
```

7. Visual Studio の[ビルド]メニューから「ClassLibrary2 のビルド」を選択し、ビルドを行います。

SQL Server データベースプロジェクトの作成

メッセージを返すストアドプロシージャ(基本編)の 「SQL Server データベースプロジェクトの作成」と同様の操作をして、 SQL Server データベースプロジェクトを作成します。 ここでは、プロジェクト名として、「FunctionSample」とします。

SQL Server データベースプロジェクトのビルドと配置

メッセージを返すストアドプロシージャ(基本編)の 「SQL Server データベースプロジェクトのビルドと配置」と同様の操作をして、 「FunctionSample」プロジェクトをビルドし、データベースに公開します。

SQL CLR ユーザ定義スカラ値関数の実行

sqlcmd ユーティリティを使用して実行した場合は、以下のように表示されます。

```
C:¥>sqlcmd -S (localdb)¥MSSQLLocalDB -E -d COBOLSample
1> select dbo.GoodsName(110)
2> go

------------------------------------------------------
TELEVISION


(1 行処理されました)
1>
```

テーブル値を返すユーザ定義関数の作成

ここでは、テーブル値を返すユーザ定義関数を作成してみましょう。

以下の形式の"GetOrders"というユーザ定義テーブル値関数を作成します。サンプルデータベースの STOCK 表、ORDERS 表および COMPANY 表を関連づけて、指定した製品名を取り扱っている会社名と発注数量をテーブルに抽出するユーザ定義関数を作成します。

Transact-SQL での呼出し形式

```
GetOrders(product_name)
```

パラメタ

- *product_name*

  製品名を指定する入力パラメタです。*product_name* のデータ型は nvarchar(max)です。

戻り値

以下の形式の表を返します。

カラム名	データ型
COMPANY_NAME	nchar(20)
ORDER_QUANTITY	int

SQL CLR テーブル値ユーザ定義関数 についての詳細は、SQL Server オンラインブックの CLR テーブル値関数を参照してください。

作業手順は以下のとおりです。

1. SQL CLR ユーザ定義テーブル値関数の手続きの記述
2. SQL CLR ユーザ定義テーブル値関数のビルドと配置
3. SQL CLR ユーザ定義テーブル値関数の実行

SQL CLR ユーザ定義テーブル値関数の手続きの記述

1. このサンプルでは、スカラ値を返すユーザ定義関数の作成で使用したプロジェクトをそのまま利用します。 プロジェクトが開かれていない場合は、Visual Studio の[ファイル]メニューから、 [開く]-[プロジェクト/ソリューション]を選択して、スカラ値を返すユーザ定義関数の作成で作成したプロジェクトを開いてください。
2. スカラ値を返すユーザ定義関数の作成の「SQL CLR ユーザ定義スカラ値関数の手続きの記述」と同様に「ユーザ定義の関数」テンプレートを選択し、 新規 COBOL ソースファイルを追加します。 ここでは、「GetOrders.cob」というファイル名を指定します。

3. テンプレートによってすでに STATIC 段落に定義されている FUNCTION-1(GetOrders)メソッド定義が挿入されているため、 このメソッドを目的のユーザ定義テーブル値関数にあわせて変更します。 デフォルトでは、"Hello"という文字列を返すユーザ定義スカラ値関数のソースコードが記述されています。

4. 作成するユーザ定義関数はデータアクセスを行うため、 環境部(ENVIRONMENT DIVISION)の特殊名段落(SPECIAL-NAMES)およびリポジトリ段落(REPOSITORY)を変更し、 テンプレートによってあらかじめ記述されている CA-SQLFUNCTION(Microsoft.SqlServer.Server.SqlFunctionAttribute) カスタム属性 の DataAccess 名前付きパラメタに Read を設定します。

   更に、SQL CLR ユーザ定義テーブル値関数を作成する場合は、CA-SQLFUNCTION カスタム属性に以下の名前付きパラメタを指定する必要があります。

パラメタ名	値	説明
FillRowMethodName	"FillRow"	関数の戻り値となるテーブルの 1 レコードを作成するスタティックメソッド名を指定します。 この関数では、"FillRow"という名前のメソッドを定義します。 ここで指定されたメソッドは SQL CLR ユーザ定義テーブル値関数を定義するメソッドと同一のクラス定義内に定義しなければなりません。
TableDefinition	"COMPANY_NAME nchar(20), ORDER_QUANTITY int"	関数の戻り値となるテーブルの定義を文字列で指定します。

   特殊名段落(SPECIAL-NAMES)の CA-SQLFUNCTION カスタム属性は以下のように変更します。

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
           CUSTOM-ATTRIBUTE CA-SQLFUNCTION CLASS CLASS-SQLFUNCTION
                            PROPERTY PROP-DATAACCESS IS PROP-READ OF ENUM-DATAACCESSKIND
                            PROPERTY PROP-FILLROWMETHODNAME IS "FillRow"
                            PROPERTY PROP-TABLEDEFINITION IS
                                       "COMPANY_NAME nvarchar(20), ORDER_QUANTITY int"
               ...
       REPOSITORY.
           CLASS CLASS-SQLFUNCTION AS "Microsoft.SqlServer.Server.SqlFunctionAttribute"
            ENUM ENUM-DATAACCESSKIND AS "Microsoft.SqlServer.Server.DataAccessKind"
            PROPERTY PROP-DATAACCESS AS "DataAccess"
            PROPERTY PROP-READ AS "Read"
            PROPERTY PROP-FILLROWMETHODNAME AS "FillRowMethodName"
            PROPERTY PROP-TABLEDEFINITION AS "TableDefinition"
               ...
```

   また、戻り値となるテーブルの、"COMPANY_NAME"カラムは、 nchar(20)型であるため、Microsoft.SqlServer.Server.SqlFacetAttribute カスタム属性を追加し、IsFixedLength 名前付きパラメタに B'1' (True)、MaxSize 名前付きパラメタに 20 を設定し、 CA-FIXEDLEN20 という カスタム属性を定義しています。

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
               ...
           CUSTOM-ATTRIBUTE CA-FIXEDLEN20  CLASS CLASS-SQLFACET
                            PROPERTY PROP-ISFIXEDLENGTH IS B'1'
                            PROPERTY PROP-MAXSIZE IS 20
               ...
       REPOSITORY.
               ...
           CLASS CLASS-SQLFACET AS "Microsoft.SqlServer.Server.SqlFacetAttribute"
            PROPERTY PROP-ISFIXEDLENGTH AS "IsFixedLength"
            PROPERTY PROP-MAXSIZE AS "MaxSize"
               ...
```

5. 次に、SQL CLR ユーザ定義テーブル値関数の本体となる FUNCTION-1(GetOrders)を変更します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
        ...
    CLASS CLASS-STRING AS "System.String"
    INTERFACE INTERFACE-IENUMERABLE AS "System.Collections.IEnumerable"
    CLASS CLASS-INT32 AS "System.Int32"
    CLASS CLASS-DATATABLE AS "System.Data.DataTable"
    CLASS CLASS-DATAROW AS "System.Data.DataRow"
    CLASS CLASS-TYPE AS "System.Type"
    PROPERTY PROP-COLUMNS AS "Columns"
    PROPERTY PROP-ROWS AS "Rows"
        ...
STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. FUNCTION-1 AS "GetOrders" CUSTOM-ATTRIBUTE CA-SQLFUNCTION.
DATA DIVISION.
WORKING-STORAGE SECTION.                                          *> [1]
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE PIC X(5).
01 GOODS PIC X(20).
01 RESULT_SET.
  02 COMPANY_NAME PIC X(20).
  02 ORDER_QUANTITY PIC S9(9) COMP-5.
    EXEC SQL END DECLARE SECTION END-EXEC.
01 DATA_TABLE OBJECT REFERENCE CLASS-DATATABLE.
01 DATA_ROW OBJECT REFERENCE CLASS-DATAROW.
01 WORK_TYPE OBJECT REFERENCE CLASS-TYPE.
01 CURSOR_OPEN_STATUS PIC 1 VALUE B'0'.
   88 IS_CURSOR_OPENED VALUE B'1'.
LINKAGE SECTION.                                                 *> [2]
01 PRODUCT_NAME OBJECT REFERENCE CLASS-STRING.
01 RET_VALUE OBJECT REFERENCE INTERFACE-IENUMERABLE.
PROCEDURE DIVISION USING BY VALUE PRODUCT_NAME RETURNING RET_VALUE.
    SET GOODS TO PRODUCT_NAME.                                    *> [3]
    EXEC SQL WHENEVER NOT FOUND GO TO :P-END END-EXEC.
    EXEC SQL WHENEVER SQLERROR GO TO :P-END END-EXEC.
    EXEC SQL
      DECLARE CUR1 CURSOR FOR
          SELECT NAME, OOH  FROM STOCK, ORDERS, COMPANY
                          WHERE GOODS = :GOODS AND
                               GNO = GOODSNO  AND
                               COMPANYNO = CNO
    END-EXEC.
P-START.
    SET DATA_TABLE TO CLASS-DATATABLE::"NEW".
    SET WORK_TYPE TO TYPE OF CLASS-STRING.
    INVOKE PROP-COLUMNS OF DATA_TABLE "Add" USING "COMPANY_NAME" WORK_TYPE.
    SET WORK_TYPE TO  TYPE OF CLASS-INT32.
    INVOKE PROP-COLUMNS OF DATA_TABLE "Add" USING "ORDER_QUANTITY" WORK_TYPE.
    EXEC SQL OPEN CUR1 END-EXEC.
    SET IS_CURSOR_OPENED TO TRUE.
P-LOOP.
    EXEC SQL
      FETCH CUR1 INTO :RESULT_SET
    END-EXEC.

    INVOKE DATA_TABLE "NewRow" RETURNING DATA_ROW.
    INVOKE DATA_ROW "set_Item" USING "COMPANY_NAME" COMPANY_NAME.
    INVOKE DATA_ROW "set_Item" USING "ORDER_QUANTITY" ORDER_QUANTITY.
    INVOKE PROP-ROWS OF DATA_TABLE "Add" USING DATA_ROW.
    GO TO P-LOOP.
P-END.
    IF IS_CURSOR_OPENED THEN
      EXEC SQL CLOSE CUR1 END-EXEC
    END-IF.
    SET RET_VALUE TO PROP-ROWS OF DATA_TABLE.
END METHOD FUNCTION-1.
```

プログラムの説明

[1] 作業場所節(WORKING-STORAGE SECTION)

メソッドのローカル変数の宣言をします。

[2] 連絡場所節(LINKAGE SECTION)

メソッドのパラメタを宣言します。

入力パラメタ(PRODUCT_NAME)は、nvarchar(max)型であるため、System.String オブジェクトを使用します。 戻り値(RET_VALUE)は、SQL CLR ユーザ定義テーブル値関数の場合は、System.Collections.IEnumerable インタフェースを指定します。

パラメタの名前の規則や指定できる型については、パラメタとデータベース機能を使用するストアドプロシージャ(応用編)の 「SQL CLR ストアドプロシージャの手続きの記述」を参照してください。

[3] 手続き部(PROCEDURE DIVISION)

COBOL のデータベース機能のカーソル機能を利用して、System.Data.DataTable クラスのオブジェクトにクエリ式の結果を格納します。 戻り値(RET_VALUE)には、DataTable オブジェクトの Rows プロパティを設定します。Rows プロパティは、DataTable に格納されているレコードオブジェクト(System.Data.DataRow クラス) のコレクションオブジェクト(System.Data.DataRowCollection クラス)であり、IEnumerable インタフェースが実装されているオブジェクトです。

6. CA-SQLFUNCTION カスタム属性に指定した"FillRow"メソッド定義を、 FUNCTION-1 メソッドの下に挿入します。 このメソッド定義は FILLROW という名前で定義し、外部名として"FillRow"を指定します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
       ...
   CLASS CLASS-OBJECT AS "System.Object"
    CLASS CLASS-SQLSTRING AS "System.Data.SqlTypes.SqlString"
    CLASS CLASS-SQLINT32 AS "System.Data.SqlTypes.SqlInt32"
       ...
STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
       ...
METHOD-ID. FILLROW AS "FillRow".
DATA DIVISION.
WORKING-STORAGE SECTION.                                        *> [1]
01 DATA_ROW OBJECT REFERENCE CLASS-DATAROW.
01 WORK_VALUE OBJECT REFERENCE CLASS-OBJECT.
01 WORK_COMPANY_NAME OBJECT REFERENCE CLASS-STRING.
01 WORK_ORDER_QUANTITY BINARY-LONG SIGNED.
LINKAGE SECTION.                                               *> [2]
01 OBJ OBJECT REFERENCE CLASS-OBJECT.
01 COMPANY_NAME OBJECT REFERENCE CLASS-SQLSTRING CUSTOM-ATTRIBUTE CA-FIXEDLEN20.
01 ORDER_QUANTITY OBJECT REFERENCE CLASS-SQLINT32.
PROCEDURE DIVISION USING BY VALUE OBJ
                      BY REFERENCE COMPANY_NAME
                      BY REFERENCE ORDER_QUANTITY.
    SET DATA_ROW TO OBJ AS CLASS-DATAROW.                          *> [3]
    SET WORK_VALUE TO DATA_ROW::"get_Item"("COMPANY_NAME").
    SET WORK_COMPANY_NAME TO WORK_VALUE AS CLASS-STRING.
    SET COMPANY_NAME TO CLASS-SQLSTRING::"NEW" (WORK_COMPANY_NAME).

    SET WORK_VALUE TO DATA_ROW::"get_Item"("ORDER_QUANTITY").
    SET WORK_ORDER_QUANTITY TO WORK_VALUE AS CLASS-INT32.
    SET ORDER_QUANTITY TO CLASS-SQLINT32::"NEW"(WORK_ORDER_QUANTITY).
END METHOD FILLROW.
```

プログラムの説明

[1] 作業場所節(WORKING-STORAGE SECTION)

メソッドのローカル変数の宣言をします。

[2] 連絡場所節(LINKAGE SECTION)

メソッドのパラメタを宣言します。

第 1 パラメタは値渡しパラメタの 1 レコード分のオブジェクト(System.Object クラス)です。 このオブジェクトは、 SQL CLR ユーザ定義テーブル値関数の本体で返すコレクション(IEnumerable)のひとつの要素です。 このサンプルでは、System.Data.DataRow クラスのオブジェクトが 1 レコード分のオブジェクトとして渡されます。

第 2 パラメタ以降は、ユーザ定義テーブル値関数の戻り値となるテーブルの各カラムデータとなる出力専用の参照渡しパラメタです。 このサンプルでは、 第 2 パラメタが"COMPANY_NAME"カラム、第 3 パラメタが"ORDER_QUANTITY"カラムにそれぞれ対応します。 第 2 パラメタに対応する"COMPANY_NAME"カラムの、SQL Server 型は `nchar(20)`であるため、 このパラメタには、CA-FIXEDLEN20 カスタム属性を付加します。 このメソッド自体には、戻り値はありません。

[3] 手続き部(PROCEDURE DIVISION)

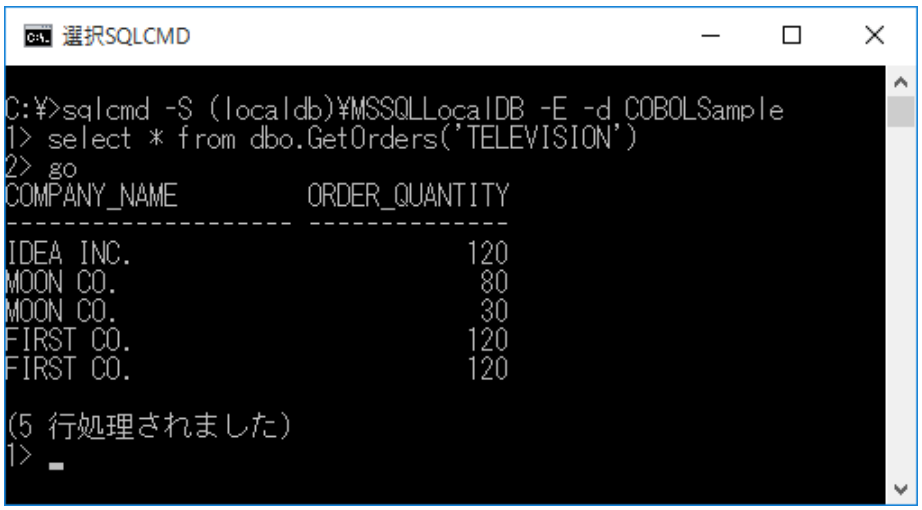
第 1 パラメタで渡されたオブジェクト(DataRow オブジェクト)を元に、 各カラムデータを取り出し、対応する出力パラメタに設定します。

SQL CLR ユーザ定義テーブル値関数のビルドと配置

メッセージを返すストアドプロシージャ(基本編)の 「SQL Server データベースプロジェクトのビルドと配置」と同様の操作をして、 「FunctionSample」プロジェクトをビルドし、データベースに公開します。

SQL CLR ユーザ定義テーブル値関数の実行

sqlcmd ユーティリティを使用して実行した場合は、以下のように表示されます。



NCOBU00812

SQL CLR トリガの作成

ここでは、簡単なサンプルを例にして、 Visual Studio で、 SQL CLR トリガを作成する手順について説明します。

SQL CLR トリガ についての詳細は、SQL Server オンラインブックの CLR トリガを参照してください。

SQL CLR トリガを作成してみましょう。

ここでは、サンプルデータベースの STOCK 表を利用して、 STOCK 表のデータ操作によって実行される、"StockTrigger"というトリガを作成します。 "StockTrigger"は、STOCK 表のデータの挿入、更新、削除操作が行われたときに実行されます。

作業手順は以下のとおりです。

1. SQL CLR トリガの プロジェクトの作成
2. SQL CLR トリガの手続きの記述とビルド
3. SQL CLR トリガの配置
4. SQL CLR トリガの実行
5. SQL CLR トリガのアンインストール方法

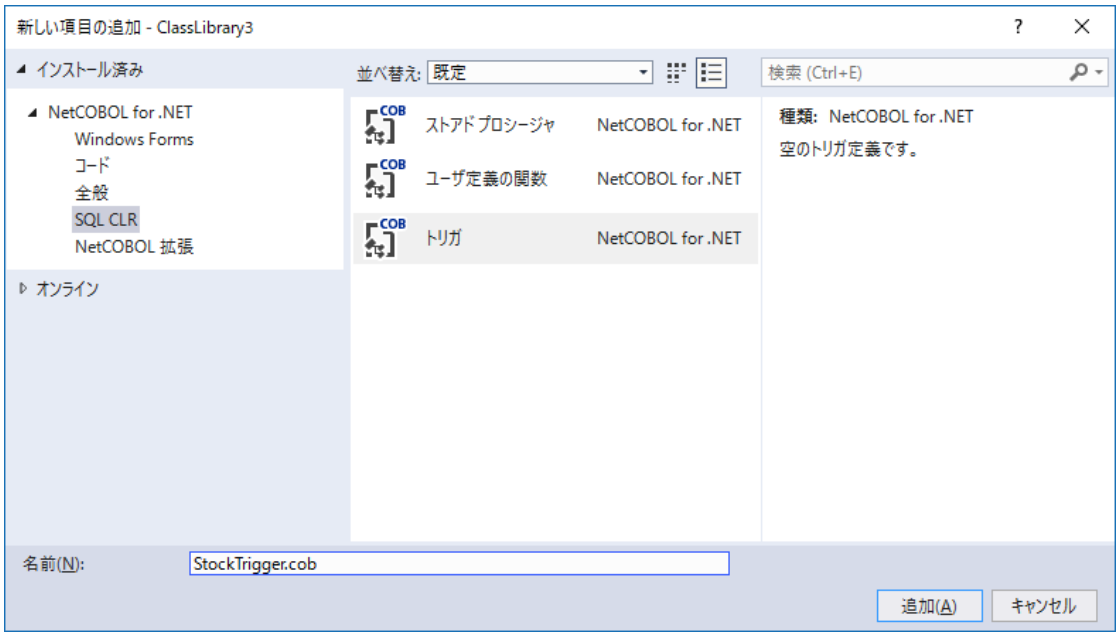
SQL CLR トリガの プロジェクトの作成

メッセージを返すストアドプロシージャ(基本編)の 「SQL Server プロジェクトの作成」と同様の操作をして、「ClassLibrary3」を作成します。

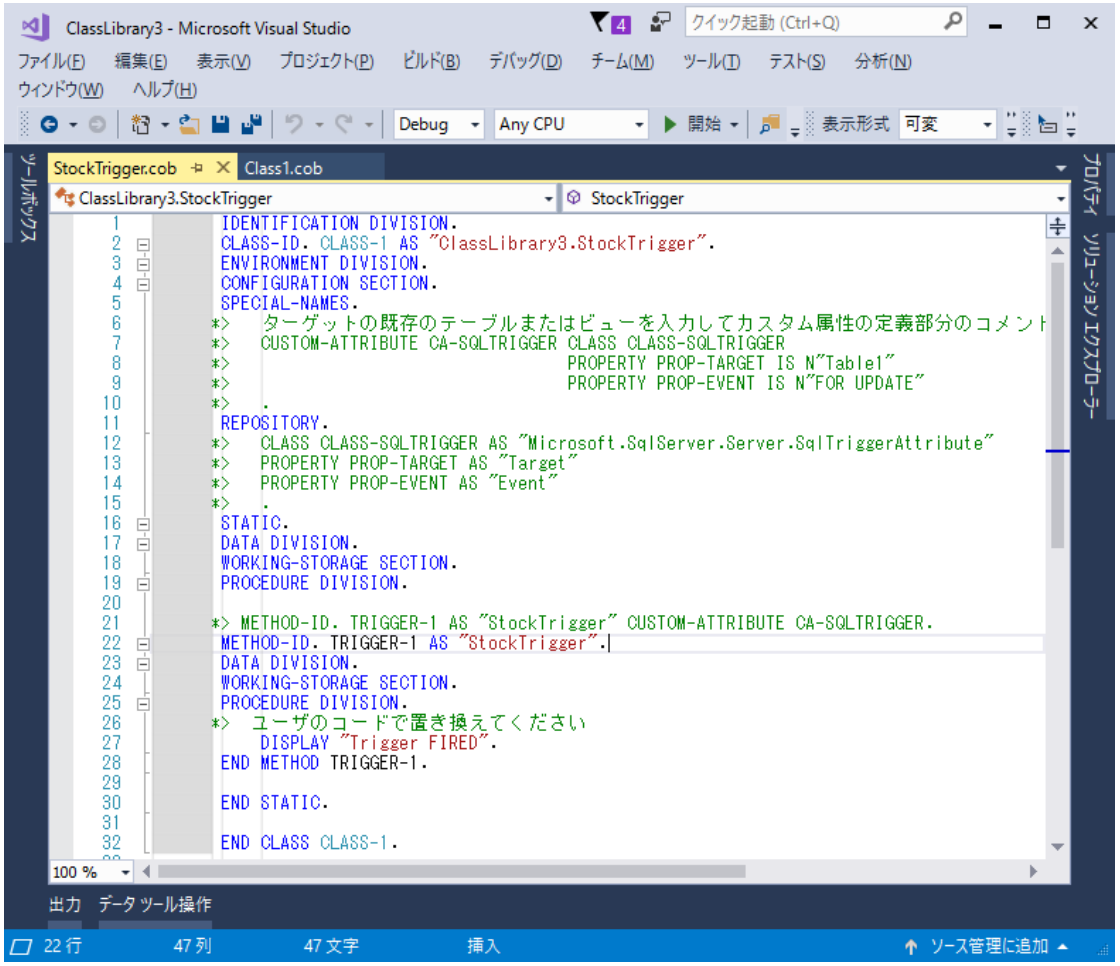
SQL CLR トリガの手続きの記述とビルド

1. メッセージを返すストアドプロシージャ(基本編)の 「SQL CLR ストアドプロシージャの手続きの記述とビルド」を参考に、 [新しい項目の追加]ダイアログボックスを表示します。

   SQL CLR トリガを作成するため、 左ペインから[NetCOBOL for .NET]-[SQL CLR]- [トリガ]テンプレートを選択し、 新規 COBOL ソースファイルを追加します。 ここでは、「StockTrigger.cob」というファイル名を指定し、［追加］ボタンをクリックします。

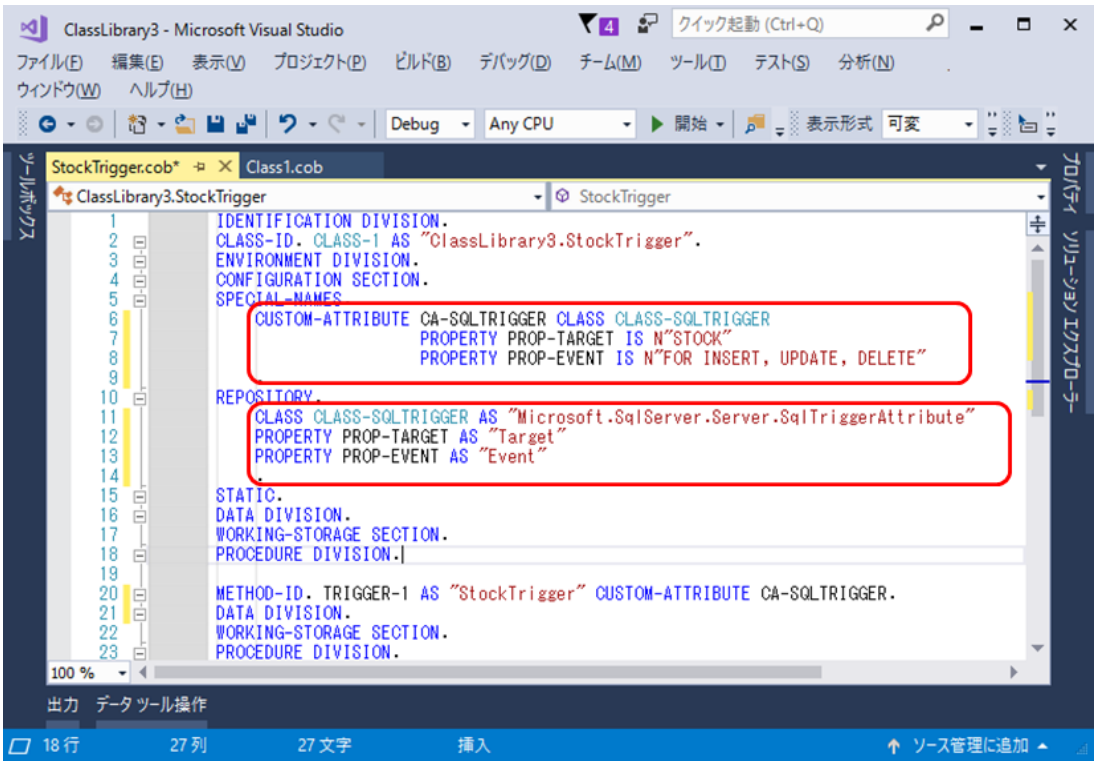


2. テンプレートによってすでにSTATIC段落に定義されているTRIGGER-1(StockTrigger)メソッド定義が挿入されているため、 このメソッドを目的のトリガにあわせて変更します。 デフォルトでは、以下のようなソースコードが表示されています。



サンプルの実行では、コメント化された CUSTOM-ATTRIBUTE を使用します。CUSTOM-ATTRIBUTE と REPOSITORY のコメント部分を解除してください。

3. "StockTrigger"は、STOCK 表へのデータの挿入時、更新時および削除時に呼び出されるようにするため、CA-SQLTRIGGER(Microsoft.SqlServer.Server.SqlTriggerAttrubute) カスタム属性では、Target 名前付きパラメタには"STOCK"を、Event 名前付きパラメタには"FOR INSERT, UPDATE, DELETE"を指定します。



4. 次にメソッド本体を以下のように変更します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
        ...
REPOSITORY.
        ...
    CLASS CLASS-SQLCONTEXT AS "Microsoft.SqlServer.Server.SqlContext"
     PROPERTY PROP-TRIGGERCONTEXT AS "TriggerContext"
     PROPERTY PROP-TRIGGERACTION AS "TriggerAction"
     ENUM ENUM-TRIGGERACTION AS "Microsoft.SqlServer.Server.TriggerAction"
     PROPERTY PROP-INSERT AS "Insert"
     PROPERTY PROP-UPDATE AS "Update"
     PROPERTY PROP-DELETE AS "Delete"
        ...
STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. TRIGGER-1 AS "StockTrigger" CUSTOM-ATTRIBUTE CA-SQLTRIGGER.
DATA DIVISION.
WORKING-STORAGE SECTION.                                              *> [1]
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE PIC X(5).
01 SQLMSG PIC X(1024).
01 SQLCODE PIC S9(9) COMP-5.
01 在庫表.
  02 製品番号   PIC S9(4) COMP-5.
  02 製品名     PIC X(20).
  02 在庫数量   PIC S9(9) COMP-5.
  02 倉庫番号   PIC S9(4) COMP-5.
    EXEC SQL END DECLARE SECTION END-EXEC.
01 データ件数   PIC 9(2).
01 挿入カーソル状態 PIC 1 VALUE B'0'.
```

```
        88 挿入カーソルオープン VALUE B'1'.
      01 削除カーソル状態 PIC 1 VALUE B'0'.
        88 削除カーソルオープン VALUE B'1'.

      PROCEDURE DIVISION.
                                                                    *> [2]
         EXEC SQL WHENEVER SQLERROR GO TO :EXIT-PROC END-EXEC.


          EXEC SQL
            DECLARE INS-CUR CURSOR FOR SELECT * FROM INSERTED
          END-EXEC.


          EXEC SQL
            DECLARE DEL-CUR CURSOR FOR SELECT * FROM DELETED
          END-EXEC.


          EVALUATE PROP-TRIGGERACTION OF PROP-TRIGGERCONTEXT OF CLASS-SQLCONTEXT

          WHEN PROP-INSERT OF ENUM-TRIGGERACTION
             PERFORM INSERT-PROC

          WHEN PROP-UPDATE OF ENUM-TRIGGERACTION
             PERFORM UPDATE-PROC
          WHEN PROP-DELETE OF ENUM-TRIGGERACTION
             PERFORM DELETE-PROC
          END-EVALUATE.



      EXIT-PROC.
          IF 挿入カーソルオープン THEN
            EXEC SQL CLOSE INS-CUR END-EXEC
          END-IF.


          IF 削除カーソルオープン THEN
            EXEC SQL CLOSE DEL-CUR END-EXEC
          END-IF.

          EXIT METHOD.


      INSERT-PROC.
          EXEC SQL OPEN INS-CUR END-EXEC.
          SET 挿入カーソルオープン TO TRUE.

          MOVE 0 TO データ件数.

          EXEC SQL FETCH INS-CUR INTO :在庫表  END-EXEC.
          PERFORM TEST BEFORE UNTIL SQLSTATE = "02000"
            DISPLAY "以下のデータが挿入されました。"
            PERFORM DISPLAY-PROC
            EXEC SQL FETCH INS-CUR INTO :在庫表  END-EXEC
          END-PERFORM.


          DISPLAY " ".
          DISPLAY  データ件数 "件のデータが挿入されました。".

      UPDATE-PROC.
          EXEC SQL OPEN INS-CUR END-EXEC.
          SET 挿入カーソルオープン TO TRUE.

          MOVE 0 TO データ件数.

          EXEC SQL FETCH INS-CUR INTO :在庫表  END-EXEC.
          PERFORM TEST BEFORE UNTIL SQLSTATE = "02000"
            DISPLAY "以下のデータが更新されました。"
            PERFORM DISPLAY-PROC
            EXEC SQL FETCH INS-CUR INTO :在庫表  END-EXEC
          END-PERFORM.
```

```
        DISPLAY " ".
        DISPLAY  データ件数 "件のデータが更新されました。".

    DELETE-PROC.
        EXEC SQL OPEN DEL-CUR END-EXEC.
        SET 削除カーソルオープン TO TRUE.

        MOVE 0 TO データ件数.

        EXEC SQL FETCH DEL-CUR INTO :在庫表  END-EXEC.
        PERFORM TEST BEFORE UNTIL SQLSTATE = "02000"
          DISPLAY "以下のデータが削除されました。"
          PERFORM DISPLAY-PROC
          EXEC SQL FETCH DEL-CUR INTO :在庫表  END-EXEC
        END-PERFORM.

        DISPLAY " ".
        DISPLAY  データ件数 "件のデータが削除されました。".

    DISPLAY-PROC.
        ADD 1 TO データ件数.
        DISPLAY データ件数 "件目のデータ："
                "  製品番号 ＝ " 製品番号
                ", 製品名 ＝ " QUOTE 製品名 QUOTE.
        DISPLAY "                在庫数量 ＝ " 在庫数量
                ", 倉庫番号 ＝ " 倉庫番号.

    END METHOD TRIGGER-1.
```

プログラムの説明

[1] 作業場所節(WORKING-STORAGE SECTION)

ホスト変数やメソッドのローカル変数の宣言をします。

[2] 手続き部(PROCEDURE DIVISION)

SQL CLR トリガでは、Microsoft.SqlServer.Server.SqlContext クラスの TriggerContext プロパティ(Microsoft.SqlServer.Server.SqlTriggerContext クラスのオブジェクト)から、TriggerAction プロパティ(Microsoft.SqlServer.Server.TriggerAction 列挙体)の値を参照することで、実行された操作を判定することができます。 Transact-SQL の INSERT 文が実行された場合は TriggerAction プロパティの値には Insert が設定されます。 同様に、UPDATE 文が実行された場合は Update が、DELETE 文が実行された場合は Delete がそれぞれ設定されます。 以下は、実行された操作を判定し処理を切り分けています。

```
        EVALUATE PROP-TRIGGERACTION OF PROP-TRIGGERCONTEXT OF CLASS-SQLCONTEXT

        WHEN PROP-INSERT OF ENUM-TRIGGERACTION
            PERFORM INSERT-PROC

        WHEN PROP-UPDATE OF ENUM-TRIGGERACTION
            PERFORM UPDATE-PROC
        WHEN PROP-DELETE OF ENUM-TRIGGERACTION
            PERFORM DELETE-PROC
        END-EVALUATE.
```

SQL CLR トリガでは、"INSERTED"および"DELETED"といった特殊なテーブルを操作することができます。 SQL CLR トリガによって、"INSERTED"テーブルには、 挿入および更新されたデータがターゲットのテーブルと同じ形式で格納されます。 同様に"DELETED"テーブルには、削除されたデータが格納されます。 以下は、埋込み SQL 文でカーソル機能を使用して、"INSERTED"テーブルにアクセスしています。

```
        EXEC SQL
          DECLARE INS-CUR CURSOR FOR SELECT * FROM INSERTED
        END-EXEC.
            ...
        WHEN PROP-INSERT OF ENUM-TRIGGERACTION
           PERFORM INSERT-PROC
            ...
    EXIT-PROC.
        IF 挿入カーソルオープン THEN
          EXEC SQL CLOSE INS-CUR END-EXEC
        END-IF.
            ...
    INSERT-PROC.
        EXEC SQL OPEN INS-CUR END-EXEC.
        SET 挿入カーソルオープン TO TRUE.

        MOVE 0 TO データ件数.

        EXEC SQL FETCH INS-CUR INTO :在庫表  END-EXEC.
        PERFORM TEST BEFORE UNTIL SQLSTATE = "02000"
          DISPLAY "以下のデータが挿入されました。"
          PERFORM DISPLAY-PROC
          EXEC SQL FETCH INS-CUR INTO :在庫表  END-EXEC
        END-PERFORM.
            ...
```

5. SQL CLR で指定する翻訳オプションセットを指定します。 Visual Studio の[プロジェクト]メニューを選択し、 [ClassLibrary3 のプロパティ]を選択します。

6. [ビルド]ペインで、「追加オプション」に以下を追加します。

```
/optionset:sqlclr
```

7. Visual Studio の[ビルド]メニューから「ClassLibrary3 のビルド」を選択し、ビルドを行います。

SQL CLR トリガの配置

SQLCMD コマンドを使用して配置する方法を示します。 データベースに接続し、以下のクエリを実行してください。

```
  ---  NetCOBOL ランタイムを配置します
CREATE ASSEMBLY [Fujitsu.COBOL]
    FROM 'C:¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime
V7.0¥Runtime¥Fujitsu.COBOL.dll'
    WITH PERMISSION_SET = SAFE
GO

--  SQL CLR トリガを配置します。FROM  以降は、クラスライブラリの DLL を指定します。
CREATE ASSEMBLY [ClassLibrary3]
    FROM 'C:¥…¥ClassLibrary3.dll'
WITH PERMISSION_SET = SAFE
GO

CREATE TRIGGER [StockTrigger] ON STOCK
FOR INSERT, UPDATE, DELETE
AS EXTERNAL NAME [ClassLibrary3].[ClassLibrary3.StockTrigger].[StockTrigger];
GO
```

SQL CLR トリガの実行

1. 以下のテスト用の SQL スクリプトの例では、STOCK 表に対してデータの挿入、更新、削除を行うスクリプトを記述しています。 それぞれの文が実行される度に、デバッグ対象の"StockTrigger"が呼び出されます。

```
insert into stock (gno,goods,qoh,whno) values (400,'DVD PLAYER',200,1)
insert into stock (gno,goods,qoh,whno) values (410,'HDD RECORDER',200,1)
update stock set qoh=qoh+100 where gno=400
update stock set qoh=qoh-100 where gno=410
delete stock where gno=400 or gno=410
```

2. 上記の SQL スクリプトが実行され、 スクリプトに記述されている Transact-SQL 文のイベントとしてデバッグ対象のトリガが呼び出されることによって、 通常のアプリケーションと同様に、アプリケーションが実行されます。

3. 以下のように実行されます。

```
   :
以下のデータが挿入されました。
01 件目のデータ：  製品番号 ＝ +0400, 製品名 ＝ "DVD PLAYER        "
                在庫数量 ＝ +000000200, 倉庫番号 ＝ +0001

01 件のデータが挿入されました。
以下のデータが挿入されました。
01 件目のデータ：  製品番号 ＝ +0410, 製品名 ＝ "HDD RECORDER       "
                在庫数量 ＝ +000000200, 倉庫番号 ＝ +0001

01 件のデータが挿入されました。
以下のデータが更新されました。
01 件目のデータ：  製品番号 ＝ +0400, 製品名 ＝ "DVD PLAYER        "
                在庫数量 ＝ +000000300, 倉庫番号 ＝ +0001

01 件のデータが更新されました。
以下のデータが更新されました。
01 件目のデータ：  製品番号 ＝ +0410, 製品名 ＝ "HDD RECORDER       "
                在庫数量 ＝ +000000100, 倉庫番号 ＝ +0001

01 件のデータが更新されました。
以下のデータが削除されました。
01 件目のデータ：  製品番号 ＝ +0410, 製品名 ＝ "HDD RECORDER       "
                在庫数量 ＝ +000000100, 倉庫番号 ＝ +0001
以下のデータが削除されました。
02 件目のデータ：  製品番号 ＝ +0400, 製品名 ＝ "DVD PLAYER        "
                在庫数量 ＝ +000000300, 倉庫番号 ＝ +0001

02 件のデータが削除されました。
   :
```

SQL CLR トリガのアンインストール方法

SQLCMD コマンドを使用してアンインストールする方法を示します。

```
DROP TRIGGER [StockTrigger]
GO
DROP ASSEMBLY [ClassLibrary3]
GO
DROP ASSEMBLY [Fujitsu.COBOL]
GO
```

Windows Communication Foundation サービスの開発

NetCOBOL for .NET と.NET Framework の Windows Communication Foundation(WCF)を組み合わせることで、 WCF サービスアプリケーションを COBOL で作成することができます。

WCF は.NET Framework 3.0 で追加された新しい分散テクノロジです。 これまで、XML Web サービスを含め、通信要件に応じてコンポーネントを選ぶ必要がありましたが、WCF ではプログラミングモデルが統一されひとつのアプリケーションで様々な通信方法のサービスを提供することが可能になりました。

ここでは WCF サービスを構築するための開発手順について説明します。

このセクションの内容

WCF サービスの概要

WCF サービスの概要について説明します。

WCF サービスのプログラム構造

WCF サービスのプログラム構造について説明します。

WCF サービスの開発手順

WCF サービスの開発手順について説明します。

NCOBU01062

WCF サービスの概要

NetCOBOL for .NET では、 .NET Framework の Windows Communication Foundation(WCF)を利用することによって WCF サービスアプリケーションを開発することができます。

WCF は、XML Web サービス(ASMX)を始め、Web Services Enhancements(WSE)、.NET リモート処理、Enterprise Services(COM+)、Microsoft Message Queuing(MSMQ)といった、既存の.NET 分散テクノロジのプログラミング・モデルを統一した、分散アプリケーションを構築するためのコンポーネントです。

WCF を使用することで、様々な通信方式の WCF サービスアプリケーションを作成できます。 そのため、WCF サービスでは Microsoft Internet Information Services(IIS)に限らず、 コンソールアプリケーション、Windows フォームアプリケーションおよび Windows サービスアプリケーションといったマネージコードによるアプリケーションでもホストすることができ、HTTP 以外にも、TCP、名前付パイプといった通信プロトコルでのサービスを実行することができます。

- WCF を使用した WCF サービスの作成の詳細については、Windows Communication Foundation を利用したアプリケーションの開発および .NET Framework のドキュメントの Windows Communication Foundation を参照してください。

WCF サービスのプログラム構造

ここでは、WCF サービスの COBOL プログラムの構造について説明します。 Visual Studio では、プロジェクトを作成する時に、 WCF サービスや WCF サービス ライブラリのテンプレートを選択すると、 自動的に WCF サービスを構成するための以下のファイルが作成されます。

.cob ファイル	WCF サービスの手続き
.svc ファイル	@ ServiceHost ディレクティブの指定。 WCF サービスの URL アドレスをサービスの手続きと関連付ける。 .svc ファイルは、Web サイトで WCF サービスのテンプレートを選択した場合のみ作成されます。
.config ファイル	WCF の構成を設定する。 Web サイトで WCF サービスのテンプレートを選択した場合は、Web.config が、 WCF サービス ライブラリのテンプレートを選択した場合は、App.config がそれぞれ作成されます。

.cob ファイル

.cob ファイルには、WCF サービスの手続きを記述します。

WCF サービスの COBOL ソースプログラムは以下のソースプログラムから構成されます。

サービスコントラクト

WCF サービスを公開するインタフェース定義ファイルです。 詳細については、サービスコントラクトを作成するを参照してください。

サービスコントラクトのソースプログラムの構造は以下のとおりです。

```
IDENTIFICATION DIVISION.                                          ・・[1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-SERVICECONTRACT CLASS CLASS-SERVICECONTRACT
    CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT CLASS CLASS-OPERATIONCONTRACT
    .
REPOSITORY.
    CLASS CLASS-SERVICECONTRACT AS "System.ServiceModel.ServiceContractAttribute"
    CLASS CLASS-OPERATIONCONTRACT AS "System.ServiceModel.OperationContractAttribute"
    .
PROCEDURE DIVISION.
METHOD-ID メソッド名 CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT.       ・・[2]
END INTERFACE インタフェース名.
```

[1]: インタフェース定義(INTERFACE-ID 段落)

WCF サービスのサービスとして公開するインタフェース定義です。 WCF サービスのサービスコントラクトとして公開するためには System.ServiceModel.ServiceContractAttribute カスタム属性を付加します。

[2]: メソッド定義(METHOD-ID 段落)

WCF サービスのオペレーション(オペレーションコントラクト)を示すメソッド定義です。オペレーションコントラクトとして公開するためには、System.ServiceModel.OperationContractAttribute カスタム属性を付加します。

データコントラクト

サービスで交換するデータの型を示すクラスのクラス定義ファイルです。 詳細については、データコントラクトを作成するを参照してください。

データコントラクトのソースプログラムの構造は以下のとおりです。

```
IDENTIFICATION DIVISION.
CLASS-ID クラス名 CUSTOM-ATTRIBUTE CA-DATACONTRACT.                    ・・[1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-DATACONTRACT CLASS CLASS-DATACONTRACT
    CUSTOM-ATTRIBUTE CA-DATAMEMBER CLASS CLASS-DATAMEMBER
    .
REPOSITORY.
    CLASS CLASS-DATACONTRACT AS "System.Runtime.Serialization.DataContractAttribute"
    CLASS CLASS-DATAMEMBER AS "System.Runtime.Serialization.DataMemberAttribute"
    .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 データ名 データ型 CUSTOM-ATTRIBUTE CA-DATAMEMBER PUBLIC.            ・・[2]
END OBJECT.
END CLASS クラス名.
```

[1]: クラス定義(CLASS-ID 段落)

WCF サービスで交換するデータが構造体の場合は、クラスを定義します。 データコントラクトとして公開するためには、System.Runtime.Serialization.DataContractAttribute カスタム属性を付加します。

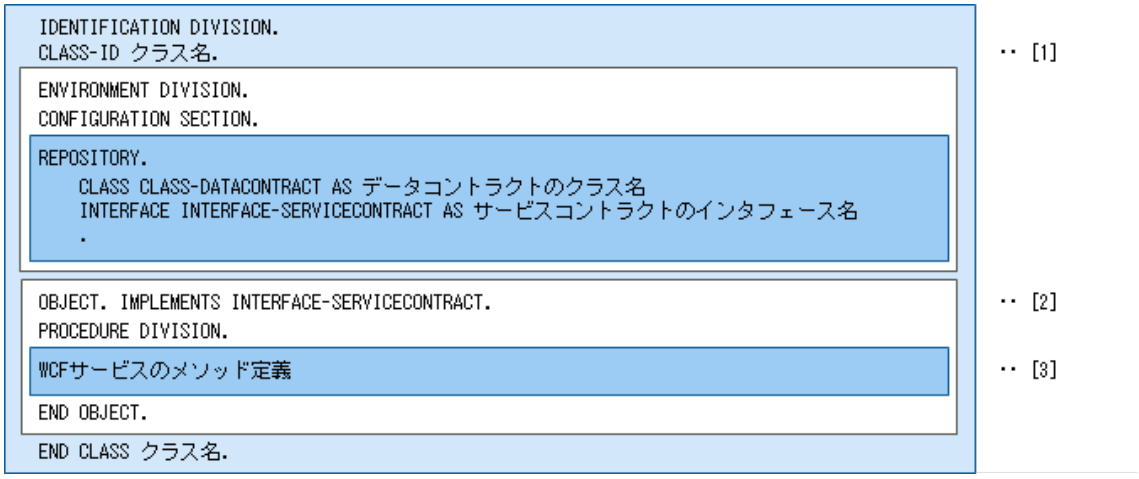
[2]: 作業場所節

データコントラクトのデータメンバーは、オブジェクト定義(OBJECT)の作業場所節で宣言します。 データコントラクトのデータメンバーとして公開したいデータには、System.Runtime.Serialization.DataMemberAttribute カスタム属性を付加します。

サービスクラス

WCF サービスを実装するクラス定義ファイルです。 詳細については、サービスクラスを作成するを参照してください。

サービスクラスのソースプログラムの構造は以下のとおりです。



[1]: クラス定義(CLASS-ID 段落)

サービスコントラクトを実装するクラス定義です。

[2]: オブジェクト定義(OBJECT)

実装するサービスコントラクトのインタフェース名を IMPLEMENTS 句に指定します。

[3]: メソッド定義(METHOD-ID 段落)

サービスコントラクトのオペレーションコードを実装するためのメソッド定義です。

また、WCF サービス ライブラリでは、以下のソースプログラムが追加されます。

サービスホスト

WCF サービスをホストするためのクラス定義ファイルです。 詳細については、Windows アプリケーションのマネージコードでホストするを参照してください。

サービスホストのソースプログラムの構造は以下のとおりです。

[1]: クラス定義(CLASS-ID 段落)

WCF サービスをホストするためのクラス定義です。System.ServiceModel.ServiceHost クラスを継承することで、 WCF サービスのホストクラスとして動作することが可能になります。

[2]: NEW メソッド

サービスホストクラスを初期化するための NEW メソッド(コンストラクター)です。

WCF サービスのサービスクラスの Type オブジェクトやデフォルトの URL(アドレス)をベースクラスである System.ServiceModel.ServiceHost のコンストラクターに渡すことで、サービスホストクラスを初期化できます。

.svc ファイル

.svc ファイルは、 WCF サービスのエントリポイントとしての機能があります。

.svc ファイルには、WCF サービスをホストすることを示す、 @ ServiceHost ディレクティブが記述されています。 .cob ファイルと.svc との関連付けは、 @ ServiceHost ディレクティブによって行なわれます。

```
<%@ ServiceHost Language="Fujitsu.COBOL" Debug="true"
            CodeBehind="~/App_Code/Service.cob" Service="Service" %>
```

Language	"Fujitsu.COBOL"固定
Debug	デバッグシンボルを使ってデバッグするかどうかの指定
Codebehind	関連付ける COBOL ソースファイル名(.cob)
Service	COBOL ソースファイルの中で定義されているサービスコントラクトを実装したサービスクラスの外部名

.svc ファイルについては、Web サイトで WCF サービスをホストするを参照してください。

.cob ファイルで定義したクラスのクラス名を変更した場合、 .svc ファイルの@ ServiceHost ディレクティブの Service に指定されたクラス名は自動的に変更されないため、 手動で変更する必要があります。

.config ファイル

.config ファイルには、WCF サービスのエントリポイント情報や動作情報などの構成を記述します。

エンドポイント情報は、WCF サービスを公開するアドレス(URL)、通信に使用するトランスポート・プロトコル、そして公開するサービスのサービスコントラクトを定義します。

.config ファイルについては、サービスのエンドポイントを設定するを参照してください。

WCF サービスの開発手順

ここでは、簡単なサンプルを例にして、 Visual Studio で、 WCF サービスを開発する場合の手順について説明します。

説明にしたがってアプリケーションを構築することによって、 WCF サービスの開発方法の基本を学習することができます。

なお、本チュートリアルでは、ローカル IIS Web サイトを使用します。 この場合、あらかじめ IIS(Microsoft Internet Information Services)の設定をしておく必要があります。IIS の設定方法については、ASP.NET Web アプリケーションの実行のための事前準備を参照してください。 Web サイトの種類については、Visual Studio のドキュメントの Visual Web Developer における Web サイトの種類を参照してください。

このセクションの内容

文字列を返却する

文字列を返却する、WCF サービスの作成について説明します。

二つの数値を加算し結果を返却

二つの引数を加算して結果を返却する、 WCF サービスの作成について説明します。

WCF サービスをホストする

WCF サービスをホストするアプリケーションの作成について説明します。

WCF サービスのクライアントからの利用

WCF サービスを利用するクライアントアプリケーションの作成について説明します。

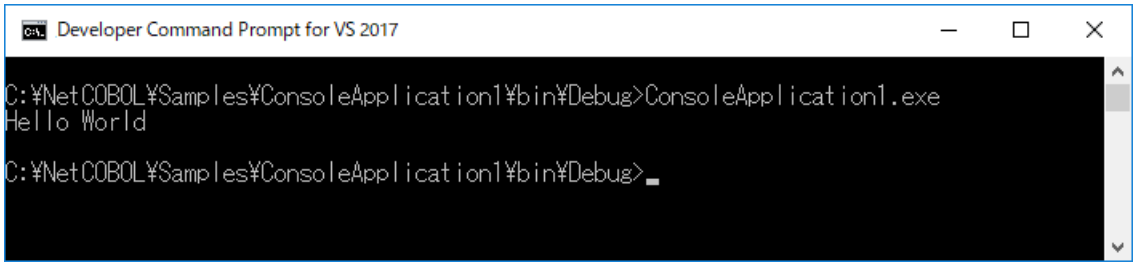
NCOBU01065

文字列を返却する

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ(http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

ここでは、「Hello World」という文字列を返却する、 簡単な WCF サービスを作成してみましょう。

以下は、WCF サービスのオペレーション(メソッド)である、 “HelloWorld”が処理された結果です。



このサンプルでは、Web サイトを利用した WCF サービスを作成します。 作成手順は以下のとおりです。

1. WCF サービスの Web サイトの作成
2. WCF サービスメソッドの手続きの記述
3. アプリケーションのビルドと実行

WCF サービスの Web サイトの作成

1. Visual Studio を管理者権限で起動します。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[Web サイト]を選択します。
3. [新しい Web サイト]ダイアログボックスが表示されます。
4. ダイアログボックスの上に表示されているボックスでは、[.NET Framework 4.7]を選択します。
5. 左ペインで、[NetCOBOL for .NET]を選択します。
6. 右ペインで、[WCF サービス]を選択します。
7. [Web の場所]ボックスには、Web サイトの構築場所を指定します。 ここでは、ローカル IIS Web サイトを使用することにし、 [Web の場所]ボックスに[HTTP]を選択して、Web サイトの URL に「`http://localhost/COBOL/Tutorial/WCFHelloWorld`」と設定します。

   なお、ローカル IIS Web サイトを使用する場合は、あらかじめ IIS を設定しておく必要があります。IIS の設定方法については、ASP.NET Web アプリケーションの実行のための事前準備を参照してください。
8. [OK]ボタンをクリックします。 これで、アプリケーションの構築に必要なファイル一式が作成されました。

   ソリューションエクスプローラーには、 プロジェクトの構成を表すツリーが表示されます。 ソリューションエクスプローラーは、Visual Studio の[表示]メニューから [ソリューション エクスプローラー]を選択すると表示されます。

WCF サービス の手続きの記述

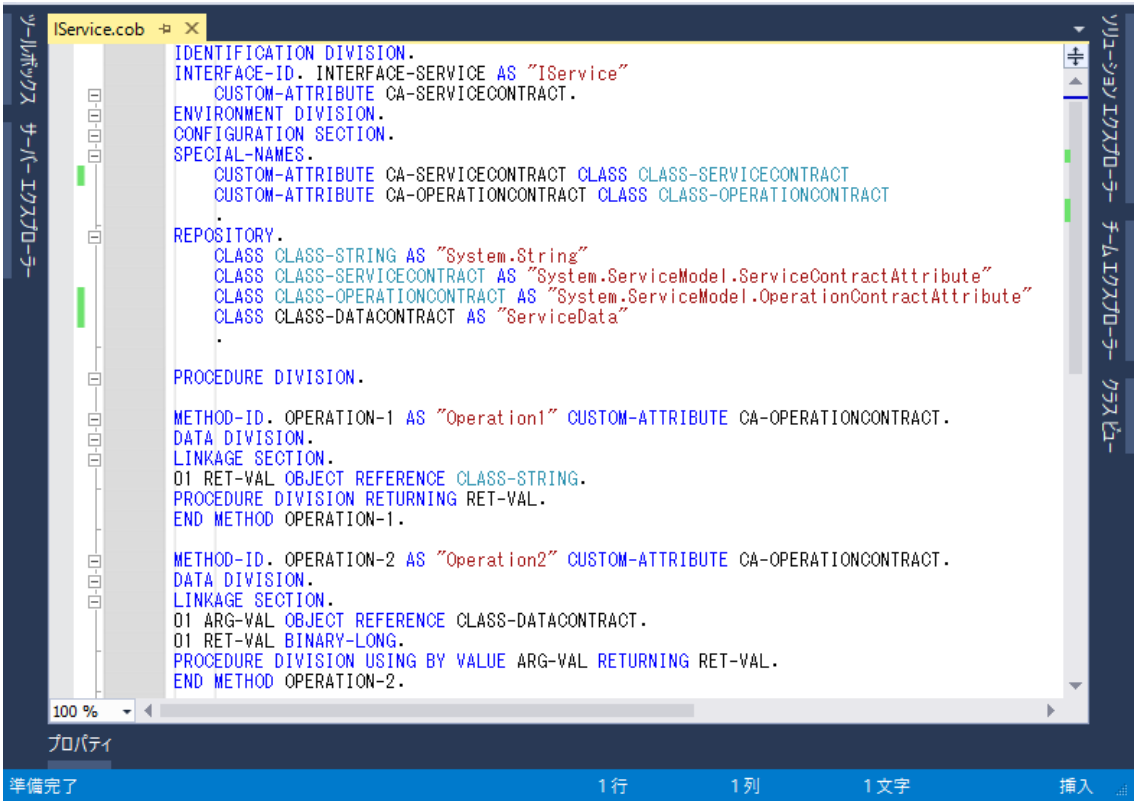
WCF サービスの手続きを記述します。

1. まず、サービスコントラクトとなる、インタフェースを定義します。 サービスコントラクトのサンプルコードは、 テンプレートによって自動生成されているので、ここではこのコードを元に作成します。

   サービスコントラクトのコードは、IService.cob というファイルに生成されています。 ソリューションエクスプローラーから、 [App_Code]フォルダの IService.cob ファイルをダブルクリックするか、 または IService.cob ファイルのノードを右クリックして表示されるメニューから [開く]を選択することによって、 手続きコードが表示されます。

2. サービスコントラクトのインタフェース定義では、 WCF サービスのオペレーションとなるメソッド定義を追加します。

   テンプレートによって自動生成されているメソッドには、 Operation1 と Operation2 という 2 つのメソッドが定義されていますが、このサンプルでは、Operation1 というメソッドをそのまま使用します。

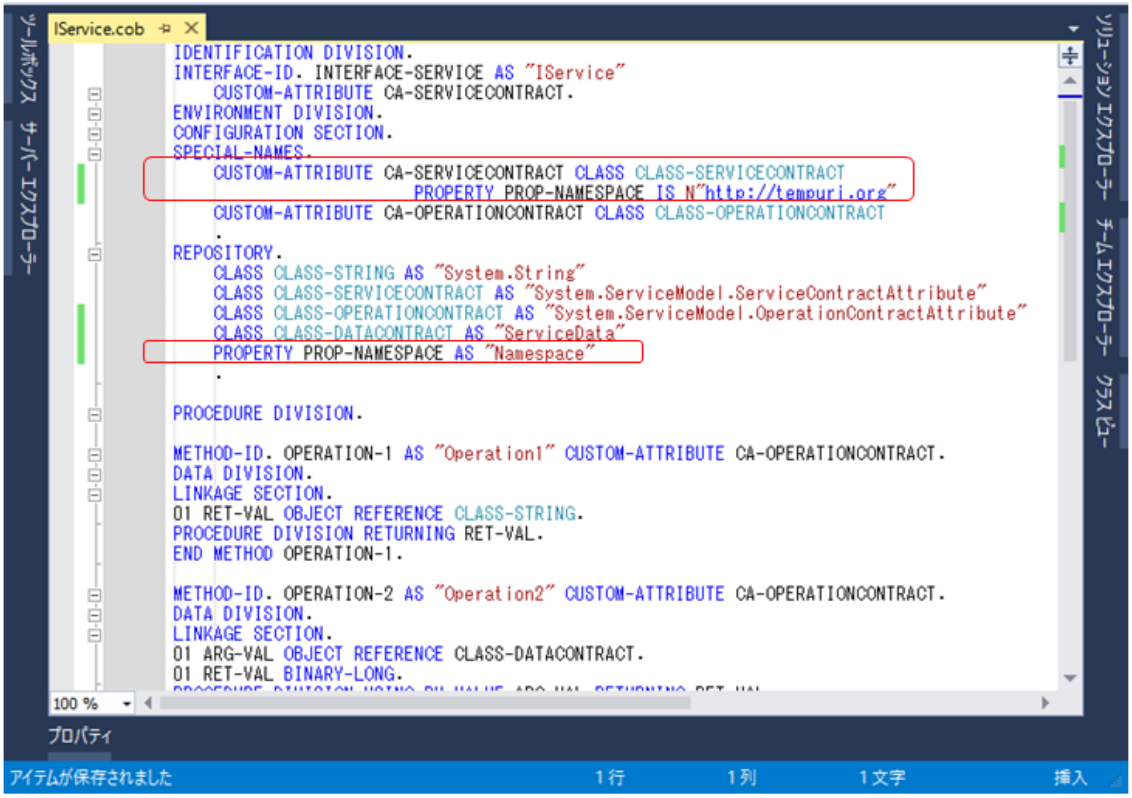


上記の IService インタフェース定義には、System.ServiceModel.ServiceContractAttribute カスタム属性が付加されています。このカスタム属性によって、IService インタフェース定義がサービスコントラクトであることを宣言しています。

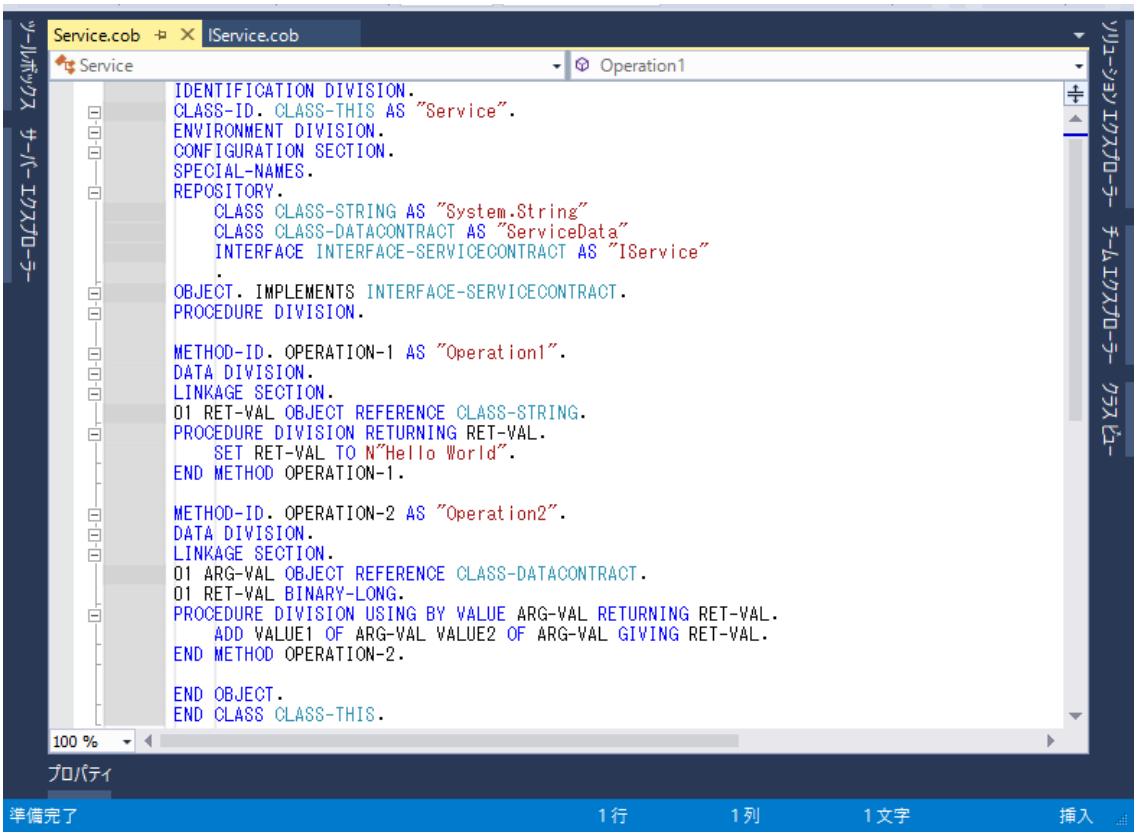
また、 Operation1 メソッド定義が、オペレーションコントラクトであることを宣言するために、System.ServiceModel.OperationContractAttribute カスタム属性が付加されています。

カスタム属性については、カスタム属性を使うを参照してください。

WCF サービスをクライアントコードから呼び出すために、WCF サービスを識別するための名前空間を設定します。 名前空間は、サービスコントラクト(IService インタフェース)に付加されている、System.ServiceModel.ServiceContractAttribute カスタム属性の Namespace プロパティに設定します。以下の例では、"http://tempuri.org"という名前空間を設定しています。



3. 続いて、サービスコントラクトを実装するサービスクラスを定義します。サービスクラスは、[App_Code] フォルダの Service.cob というファイルに生成されているため、ソリューションエクスプローラーからこのファイルを表示します。



上記の Service クラスは IService インタフェースを実装した単純なクラス定義です。 OBJECT 段落の IMPLEMETS 句に IService インタフェースのインタフェース名を指定し、IService インタフェースが

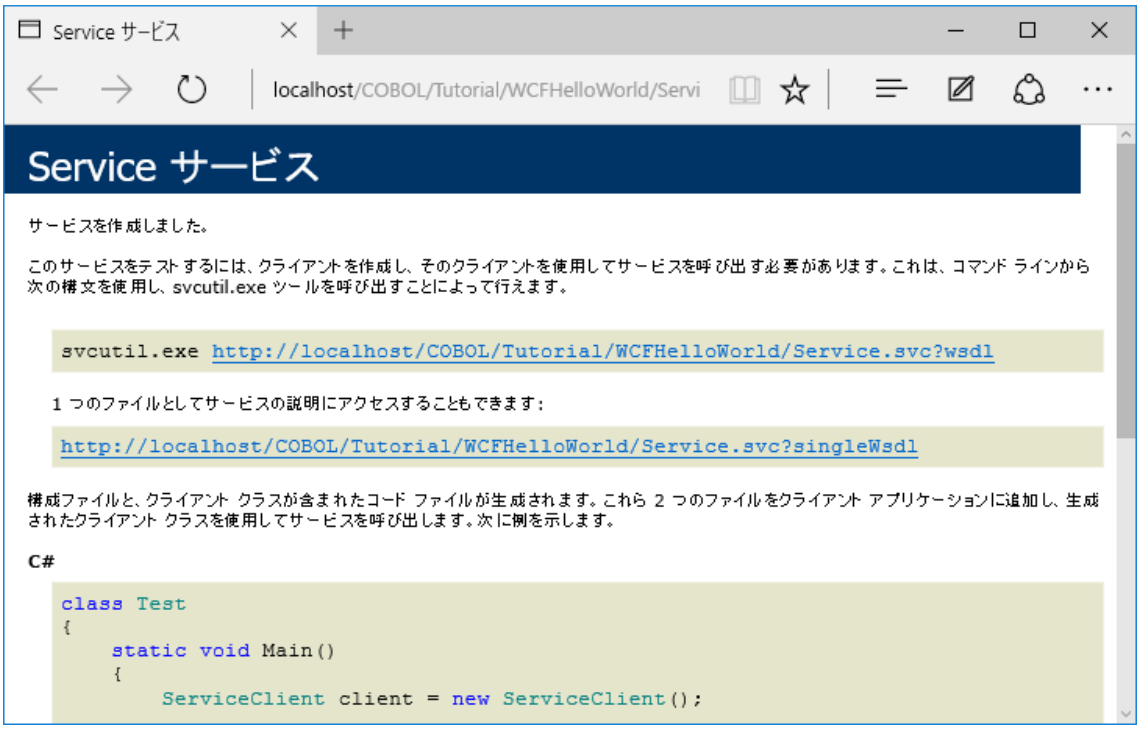
実装されていることを宣言しています。

実装する IService インタフェースに宣言されているすべてのメソッド定義を、 Service クラス定義に追加し、メソッド定義の手続き部にサービスのオペレーションとなる手続きコードを記述します。 Operation1 メソッドでは、復帰値として System.String 型の文字列を返却するメソッドであるため、文字列を返却する手続きを記述します。 テンプレートで自動生成されたコードでは復帰値の文字列に"Hello World"を設定しているため、これをそのまま使用します。

アプリケーションのビルドと実行

1. WCF サービスを 作成 するため、サービスのエントリポイントとなる Service.svc ファイルをスタートページに設定します。 スタートページに設定するには、ソリューションエクスプローラーで Service.svc ファイルを選択し、 ノードを右クリックして表示されるメニューから[スタート ページに設定]を選択します。
2. ソリューションエクスプローラーで Service.svc ファイルを選択し、 ノードを右クリックして表示されるメニューから[ブラウザーで表示]を選択します。
3. Visual Studio は、Web ブラウザを起動し、 WCF サービスのエントリポイントとなる URL のページを動的に生成します。

   WCF サービスが正しくビルドされている場合は以下の画面が表示されます。



4. Web ブラウザを閉じます。

WCF サービスの動作を確認するためのクライアントアプリケーションを作成します。

1. 生成したプロキシコードを呼び出すために、コンソールアプリケーションを作成します。

   コンソールアプリケーションは以下の手順で作成します。

   1. Visual Studio の[ファイル]メニューから[新規作成]-[プロジェクト]を選択します。
   2. [新しいプロジェクト]ダイアログボックスで、[コンソールアプリケーション]を選択します。
   3. ダイアログボックスの右上に表示されているボックスでは、[.NET Framework 4.7]を選択します。

4. 任意のプロジェクト名を指定し、[OK]ボタンを押します。 このサンプルでは、ConsoleApplication1 とします。

2. メタデータ ユーティリティ ツール (Svcutil.exe)を使用してプロキシコードを作成します。以下は、NetCOBOL for .NET によるプロキシコードを生成するコマンドの例です。

```
Svcutil.exe /syncOnly /language:"Fujitsu.COBOL.COBOLCodeProvider, Fujitsu.COBOL.CodeDom,
  Version=8.0.124.0, Culture=neutral, PublicKeyToken=fac0fe3cab973246"
  http://localhost/COBOL/Tutorial/WCFHelloWorld/Service.svc
```

(上記の例では複数行に分けていますが実際には 1 行で記述してください。 また、ServiceModel メタデータ ユーティリティ ツールは Visual Studio または Microsoft Windows SDK のコマンドプロンプトで実行してください。)

ServiceModel メタデータ ユーティリティ ツールが正しく実行できた場合は、以下の 2 のファイルが生成されます。

Service.cobx	参照する WCF サービスのプロキシコードが生成されます。
output.config	参照する WCF サービスに接続するための構成設定が生成されます。

生成された Service.cobx ファイルの文字コードは UTF-8 です。

3. 生成された Service.cobx と output.config を ConsoleApplication1 プロジェクトに追加します。

   プロジェクトには以下の手順で追加します。

   1. ソリューションエクスプローラーで ConsoleApplication1 プロジェクトを選択します。
   2. [プロジェクト]メニューから[既存項目の追加]を選択し、[既存項目の追加]ダイアログボックスを開きます。
   3. Service.cobx と output.config が生成されたフォルダに移動します。
   4. [オブジェクトの種類]ボックスでは、[すべてのファイル (*.*)]を選択します。
   5. Service.cobx と output.config を選択し、[OK]ボタンをクリックします。

   Service.cobx をソリューションエクスプローラーから開き、 [ファイル]メニューから[保存オプションの詳細設定]で[保存オプションの詳細設定]ダイアログボックスを開き、[エンコード]を"日本語 (シフト JIS) - コードページ 932"に変更します。

   また、output.config ファイルをそのままアプリケーション構成ファイルとして使用するため、App.config という名前に変更します。

4. プロキシコードは以下のシステムアセンブリを参照するため、プロジェクトの[参照]で追加します。

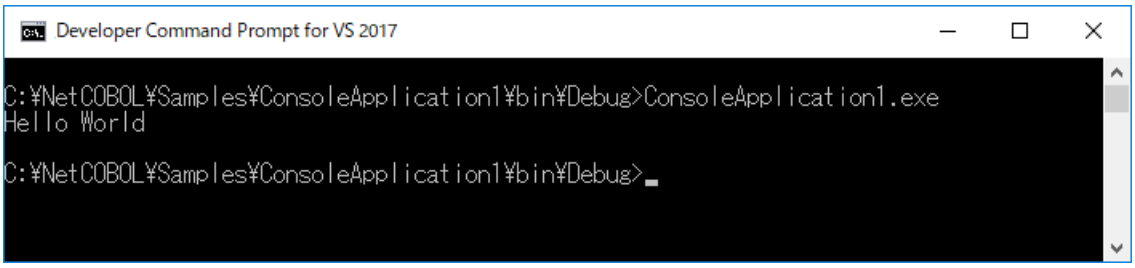
   - System.ServiceModel
   - System.Runtime.Serialization

5. Main.cob を開き、以下のようにプロキシコードを呼び出すコードを追加します。

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. MAIN AS "ConsoleApplication1.Main".
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
       REPOSITORY.
           CLASS CLASS-CLIENT AS "ServiceClient"
           CLASS CLASS-STRING AS "System.String"
           .
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 CLIENT OBJECT REFERENCE CLASS-CLIENT.
       01 RETVAL OBJECT REFERENCE CLASS-STRING.
       01 WK-TEXT PIC X(20).
       PROCEDURE DIVISION.
      *  WCF サービスクライアント(プロキシ)のインスタンスを作成し、
      *  サービスに接続します。
           SET CLIENT TO CLASS-CLIENT::"NEW".

      *  サービスのオペレーションを呼び出します。
           INVOKE CLIENT "Operation1" RETURNING RETVAL.

      *  オペレーションから返却された文字列を表示します。
           SET WK-TEXT TO RETVAL.
           DISPLAY WK-TEXT.

      *  サービスクライアントを終了します。
           INVOKE CLIENT "Close".

       END PROGRAM MAIN.
```

6. ConsoleApplication1 のクライアントアプリケーションをビルドし、実行します。

   正常に実行されると、結果は以下のように表示されます。



プロキシコードについては、プロキシコードを利用してクライアントを作成するを参照してください。

NCOBU01066

二つの数値を加算し結果を返却

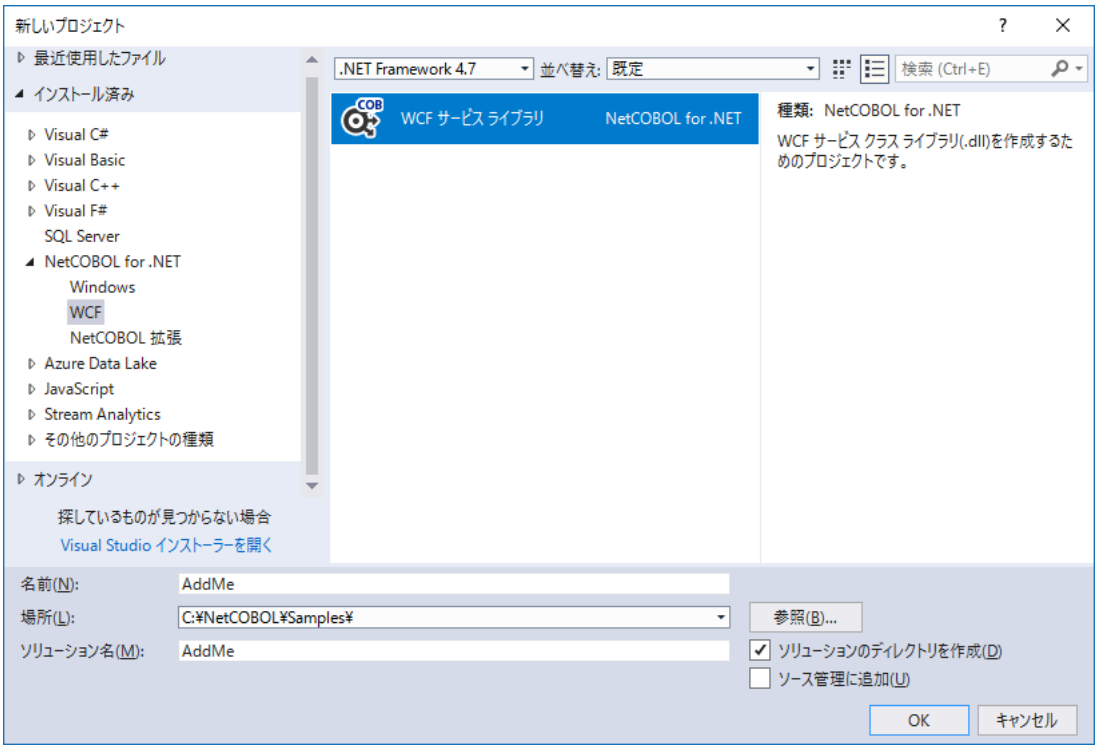
次に、二つの数値を引数として入力し、加算した結果を返却する、 WCF サービスを作成してみましょう。

このサンプルでは、WCF サービス ライブラリを使用した WCF サービスを作成します。 作成手順は以下のとおりです。

1. WCF サービス ライブラリプロジェクトの作成
2. アプリケーションの手続きの記述
3. アプリケーションのビルドと実行

WCF サービス ライブラリプロジェクトの作成

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから、[新規作成]-[プロジェクト]を選択します。
3. [新しいプロジェクト]ダイアログボックスが表示されます。
4. 左ペインで、[NetCOBOL for .NET]-[WCF]を選択します。
5. 右ペインで、[WCF サービス ライブラリ]を選択します。
6. [名前]に、新規プロジェクトの名前を入力します。 このサンプルではプロジェクト名は「AddMe」とします。 (プロジェクト名は自由につけて構いません。)



アプリケーションの手続きの記述

1. WCF サービスの手続きを記述します。

   WCF サービスのサービスコントラクト(IService1.cob)、サービスクラス(Service1.cob)を記述します。このサンプルではテンプレートから生成されたソースコードをそのまま使用します。

   WCF サービス ライブラリプロジェクトのテンプレートから生成される WCF サービスの内容は、Web サイトの[WCF サービス]テンプレートで自動生成される内容と同じです。 このサンプルでは、Operation2 というメソッドを使用します。

```
IDENTIFICATION DIVISION.
INTERFACE-ID. INTERFACE-SERVICE AS "AddMe.IService1"
   CUSTOM-ATTRIBUTE CA-SERVICECONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-SERVICECONTRACT CLASS CLASS-SERVICECONTRACT
   CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT CLASS CLASS-OPERATIONCONTRACT
   .
REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-SERVICECONTRACT AS "System.ServiceModel.ServiceContractAttribute"
   CLASS CLASS-OPERATIONCONTRACT AS "System.ServiceModel.OperationContractAttribute"
   CLASS CLASS-DATACONTRACT AS "AddMe.Service1Data"
   .

PROCEDURE DIVISION.

METHOD-ID. OPERATION-1 AS "Operation1" CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT.
DATA DIVISION.
LINKAGE SECTION.
01 RET-VAL OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION RETURNING RET-VAL.
END METHOD OPERATION-1.

METHOD-ID. OPERATION-2 AS "Operation2" CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT.
DATA DIVISION.
LINKAGE SECTION.
01 ARG-VAL OBJECT REFERENCE CLASS-DATACONTRACT.
01 RET-VAL BINARY-LONG.
PROCEDURE DIVISION USING BY VALUE ARG-VAL RETURNING RET-VAL.
END METHOD OPERATION-2.

END INTERFACE INTERFACE-SERVICE.
```

2. Operation2 では、引数に加算する値を渡します。 この引数は、構造体形式のデータ型であるためデータコントラクトとしてクラスを定義します。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-DATA AS "AddMe.Service1Data"
   CUSTOM-ATTRIBUTE CA-DATACONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-DATACONTRACT CLASS CLASS-DATACONTRACT
   CUSTOM-ATTRIBUTE CA-DATAMEMBER CLASS CLASS-DATAMEMBER
   .
REPOSITORY.
   CLASS CLASS-DATACONTRACT AS "System.Runtime.Serialization.DataContractAttribute"
   CLASS CLASS-DATAMEMBER AS "System.Runtime.Serialization.DataMemberAttribute"
   .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 VALUE1 BINARY-LONG CUSTOM-ATTRIBUTE CA-DATAMEMBER PUBLIC.
01 VALUE2 BINARY-LONG CUSTOM-ATTRIBUTE CA-DATAMEMBER PUBLIC.
END OBJECT.

END CLASS CLASS-DATA.
```

上記の テンプレートから自動生成された AddMe.Service1Data には、System.Runtime.Serialization.DataContractAttribute カスタム属性が付加されています。このカスタム属性によって、AddMe.Service1Data クラス定義がデータコントラクトであることを宣言しています。

さらに、作業場所節に宣言されている VALUE1 と VALUE2 には、System.Runtime.Serialization.DataMemberAttribute カスタム属性が付加されており、データコントラクトのデータメンバーであることを表しています。

3. ソリューションエクスプローラーから Service1.cob ファイルを開きます。オペレーションとなる手続きコードを記述します。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "AddMe.Service1".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-DATACONTRACT AS "AddMe.Service1Data"
   INTERFACE INTERFACE-SERVICECONTRACT AS "AddMe.IService1"
   .
OBJECT. IMPLEMENTS INTERFACE-SERVICECONTRACT.
PROCEDURE DIVISION.

METHOD-ID. OPERATION-1 AS "Operation1".
DATA DIVISION.
LINKAGE SECTION.
01 RET-VAL OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION RETURNING RET-VAL.
   SET RET-VAL TO N"Hello World".
END METHOD OPERATION-1.

METHOD-ID. OPERATION-2 AS "Operation2".
DATA DIVISION.
LINKAGE SECTION.
01 ARG-VAL OBJECT REFERENCE CLASS-DATACONTRACT.
01 RET-VAL BINARY-LONG.
PROCEDURE DIVISION USING BY VALUE ARG-VAL RETURNING RET-VAL.
   ADD VALUE1 OF ARG-VAL VALUE2 OF ARG-VAL GIVING RET-VAL.
END METHOD OPERATION-2.

END OBJECT.
END CLASS CLASS-THIS.
```

上記の AddMe.Service1 クラスでは、AddMe.IService1 インタフェースを実装します。 Operation2 メソッドでは、引数で渡された二つの数値を加算し結果を返却するオペレーションを記述します。 テンプレートから自動生成されたコードでは、引数の AddMe.Service1Data のメンバーの VALUE1 と VALUE2 を加算し、返却値に設定しているため、これをそのまま使用します。

4. ソリューションエクスプローラーから Service1Host.cob ファイルを開きます。

WCF サービス ライブラリでは、WCF サービスをホストするためのクラス定義のサンプルコードもテンプレートから自動生成されます。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-HOST AS "AddMe.Service1Host" INHERITS CLASS-SERVICEHOST.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-SERVICEHOST AS "System.ServiceModel.ServiceHost"
   CLASS CLASS-SERVICE AS "AddMe.Service1"
   CLASS CLASS-URI AS "System.Uri"
   CLASS CLASS-TYPE AS "System.Type"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. NEW.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 URI OBJECT REFERENCE CLASS-URI.
01 SERVICE-TYPE OBJECT REFERENCE CLASS-TYPE.
01 SVC OBJECT REFERENCE CLASS-SERVICE.
```

```
       PROCEDURE DIVISION.
           SET URI TO
               CLASS-URI::"NEW"("http://localhost:8081/COBOL/Tutorial/AddMe/Service1").
           SET SERVICE-TYPE TO TYPE OF CLASS-SERVICE.
           INVOKE SUPER "NEW" USING SERVICE-TYPE URI
       END METHOD NEW.

       END OBJECT.
       END CLASS CLASS-HOST.
```

上記の AddMe.Service1Host クラスでは、System.ServiceModel.ServiceHost クラスを継承し、コンストラクター (NEW メソッド) で WCF サービスのデフォルト URL とサービスの型を System.ServiceModel.ServiceHost のコンストラクターに設定しています。

このサービスホストクラスをインスタンス化されることで、WCF サービスが利用可能になります。

なお、このサンプルで公開する WCF サービスの URL アドレスは、ここでは、`http://localhost:8081/COBOL/Tutorial/AddMe/Service1` としています。

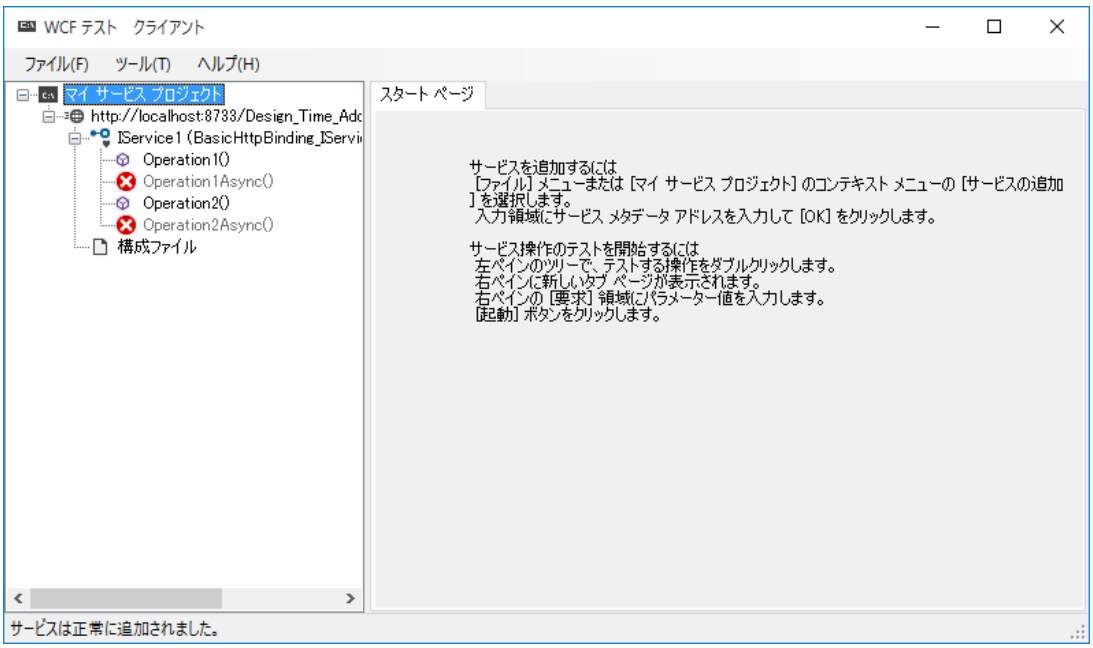
ここで追加したサービスホストクラスは、WCF サービスをホストするで使用します。後述する「アプリケーションのビルドと実行」のデバッグでは使用されません。

アプリケーションのビルドと実行

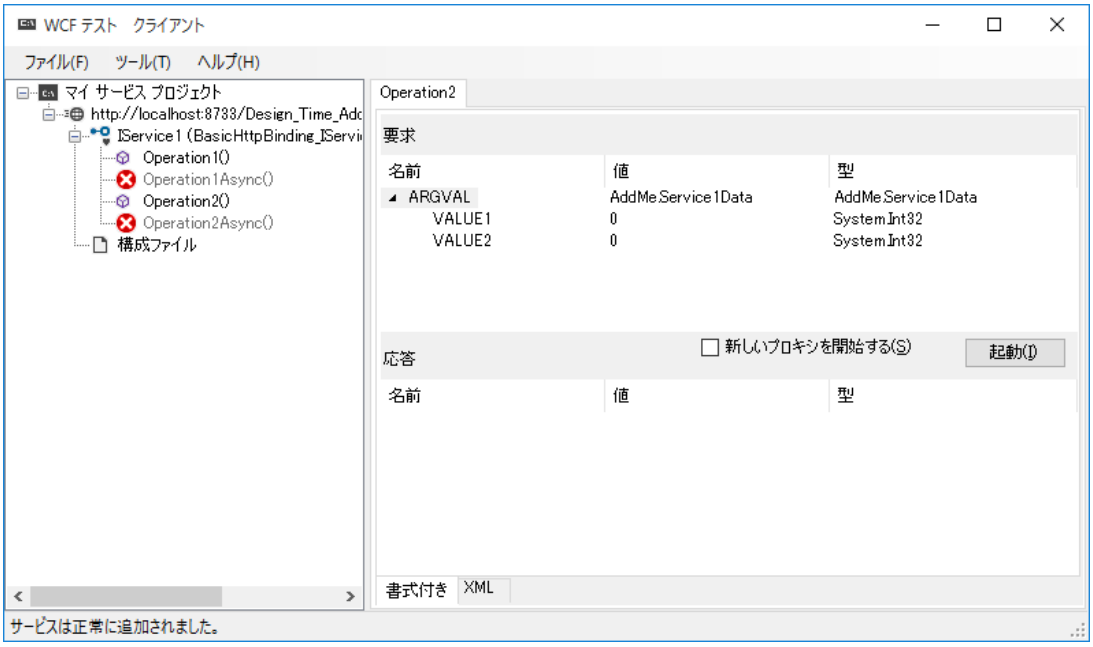
1. Visual Studio 上で、[デバッグ]メニューから [デバッグ 開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。

   WCF サービス ライブラリでは、デバッグが開始されると、WCF サービス ホスト (WcfSvcHost.exe) が起動されます。このツールはプロジェクト内の WCF サービスを検出して自動的にホストします。

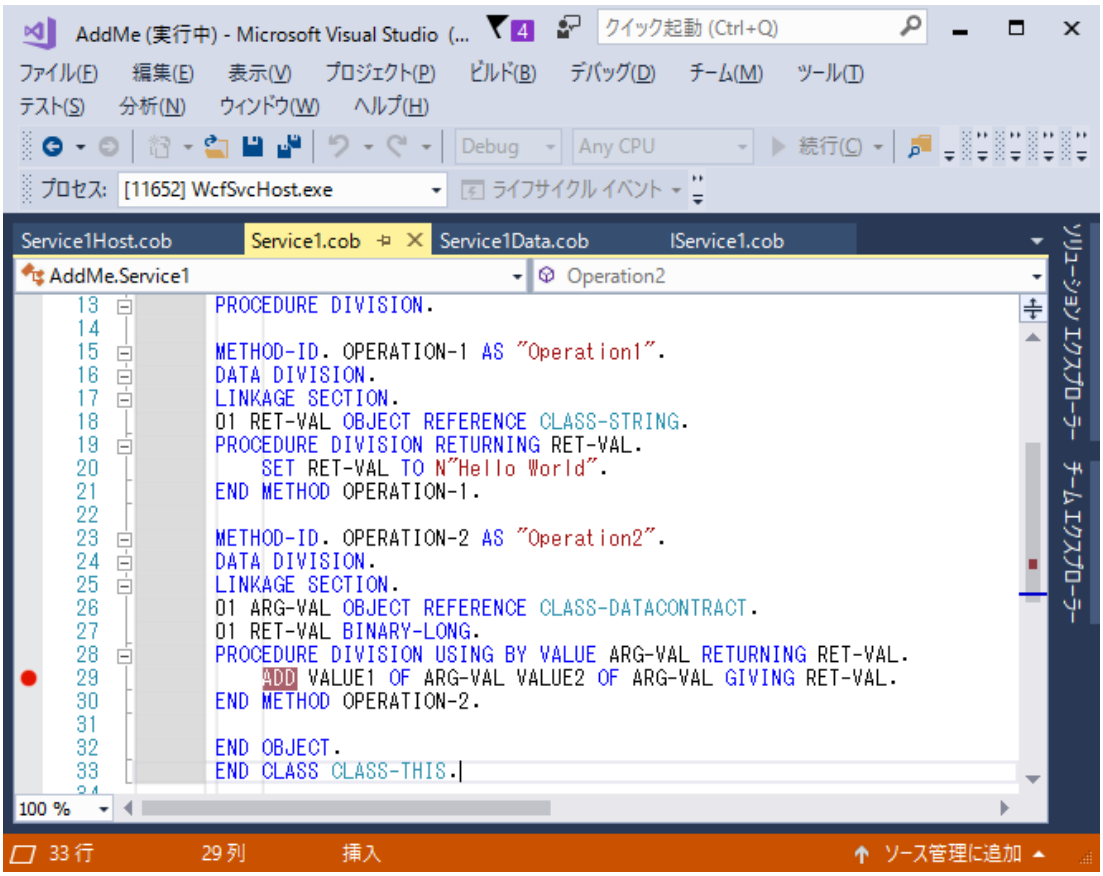
2. 続いて、WCF のテスト用クライアント (WcfTestClient.exe)が起動されます。このツールは、WCF サービス ホストでホストされている WCF サービスと通信し、GUI からテストしたいオペレーションを選択して、サービスへの要求と応答をテストすることができます。



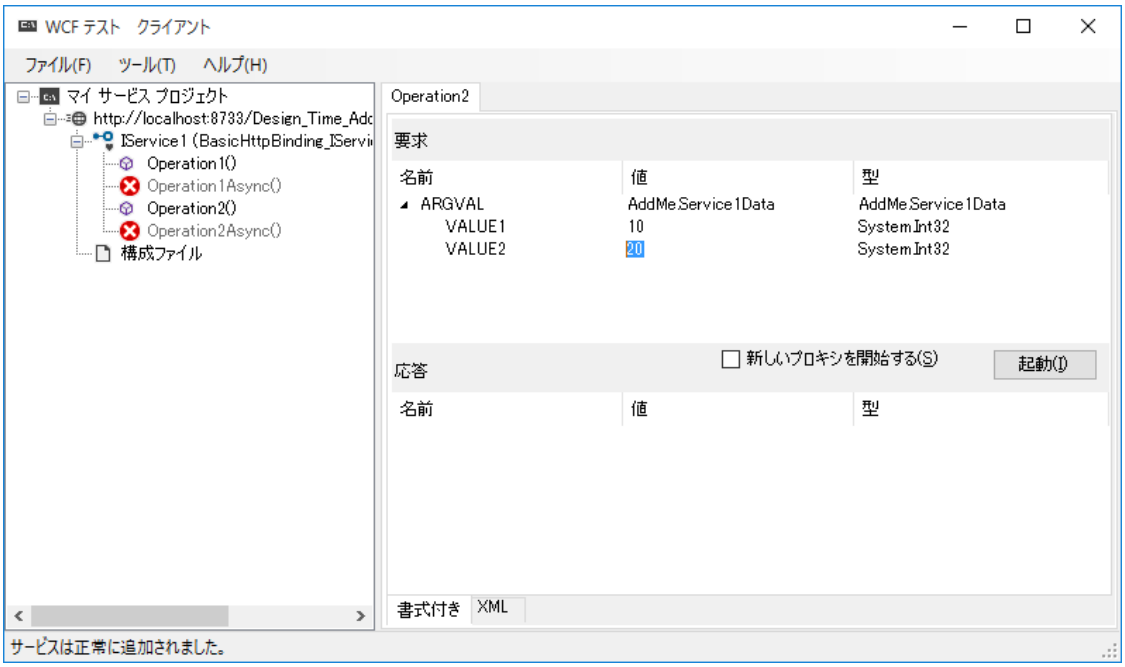
3. ここでは、Operation2 をテストするため、WCF のテスト用クライアントで [Operation2]をダブルクリックします。



4. Visual Studio で Service1.cob を開き、Operation2 メソッドのブレークポイントを設定します。

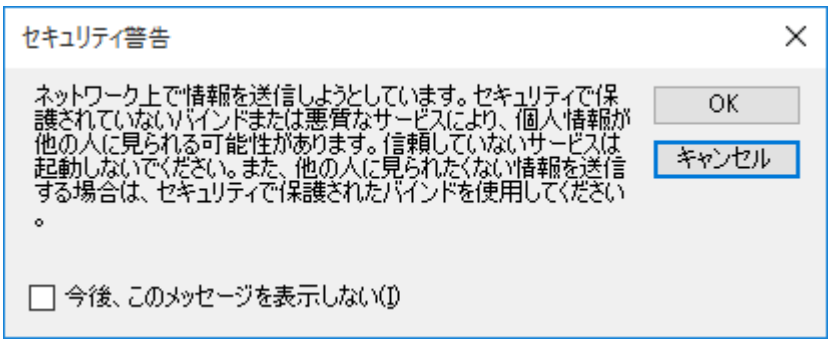


5. 再び、WCF のテスト用クライアントのウィンドウに戻り、[要求]側で送信する値を入力します。ここでは、VALUE1 に “10”、VALUE2 に “20” を指定しています。

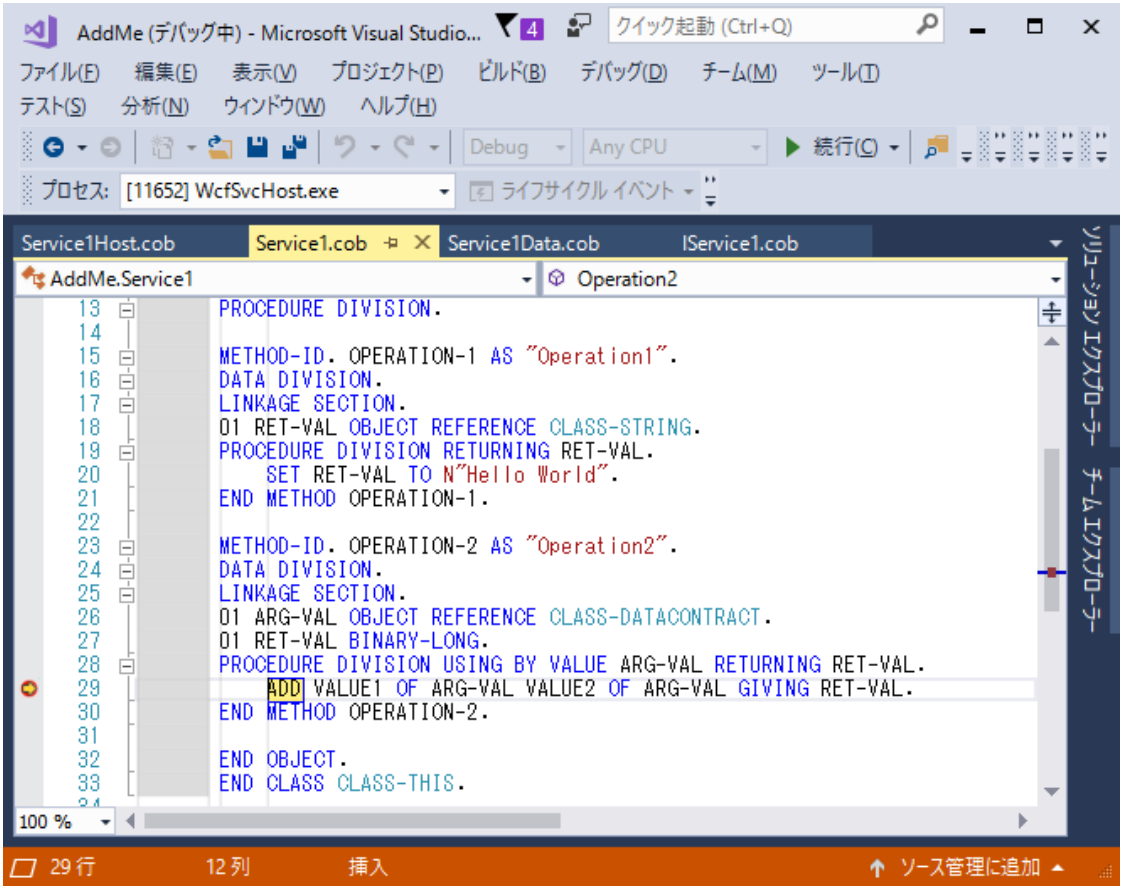


6. [起動]ボタンをクリックすると、選択した Operation2 メソッドが呼び出されます。

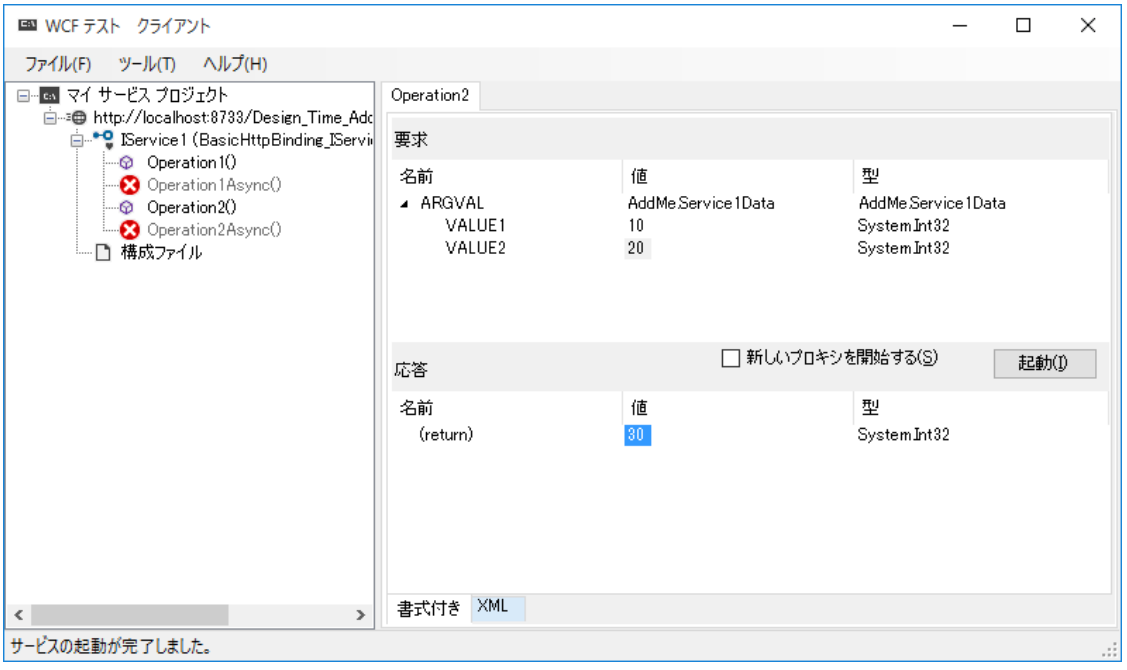
   このとき、以下のメッセージボックスが表示されることがあります。ここでは、[OK]ボタンをクリックしてデバッグを続行します。



   Operation2 メソッドが正しく呼び出されると、設定したブレークポイントで停止します。



7. デバッグを続行すると、WCF のテスト用クライアントに戻り、[応答]側に結果が表示されます。



8. WCF のテスト用クライアントを終了し、デバッグを終了します。

WCF サービスをホストする

WCF サービスをホストするアプリケーションを作成してみましょう。 ここでは、二つの数値を加算し結果を返却で作成した WCF サービスをホストする簡単な Windows アプリケーションを作成します。

作成手順は以下のとおりです。

1. Windows アプリケーションの作成
2. アプリケーションの手続きの記述
3. アプリケーションのビルドと実行

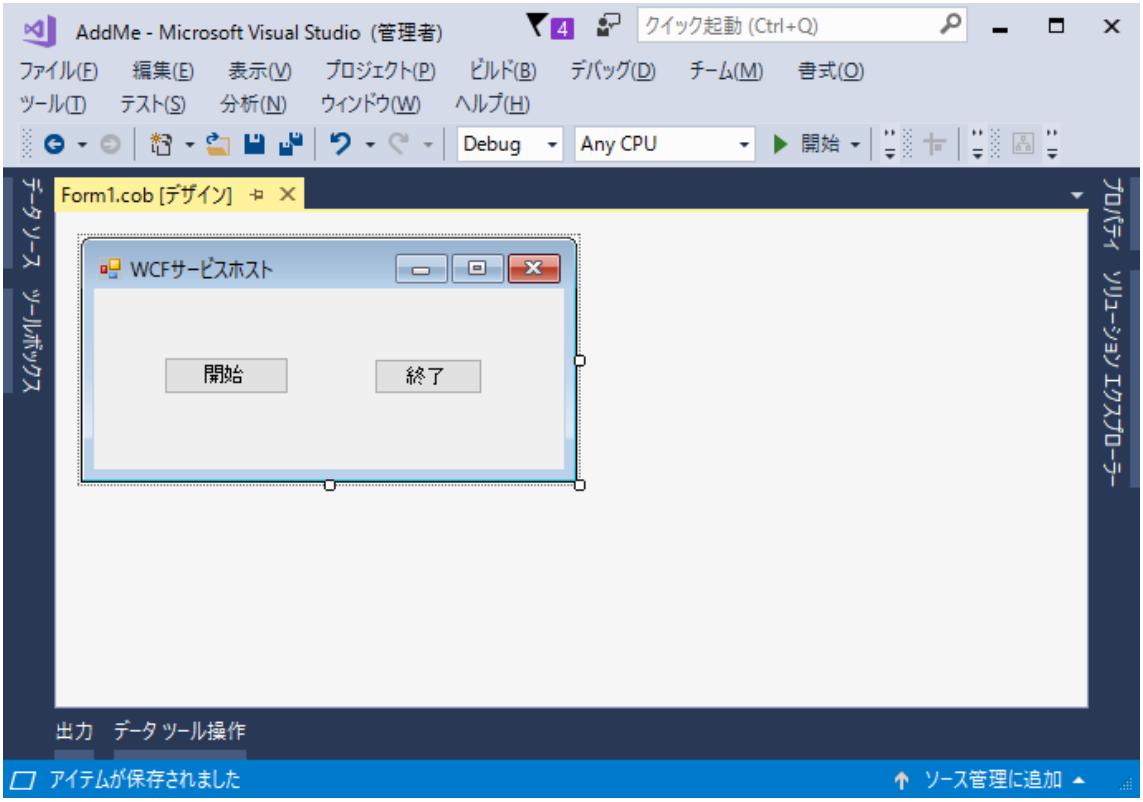
Windows アプリケーションの作成

1. 二つの数値を加算し結果を返却で作成した WCF サービス ライブラリと同じソリューションに、Windows フォーム プロジェクトを追加します。

   Windows フォーム プロジェクトの作成方法については、Windows アプリケーションの開発手順を参考にしてください。 ここでは、AddMeHost という名前の Windows アプリケーションプロジェクトを作成します。(プロジェクト名は自由につけて構いません。)

   対象のフレームワークは、“.NET Framework 4.7”に設定します。

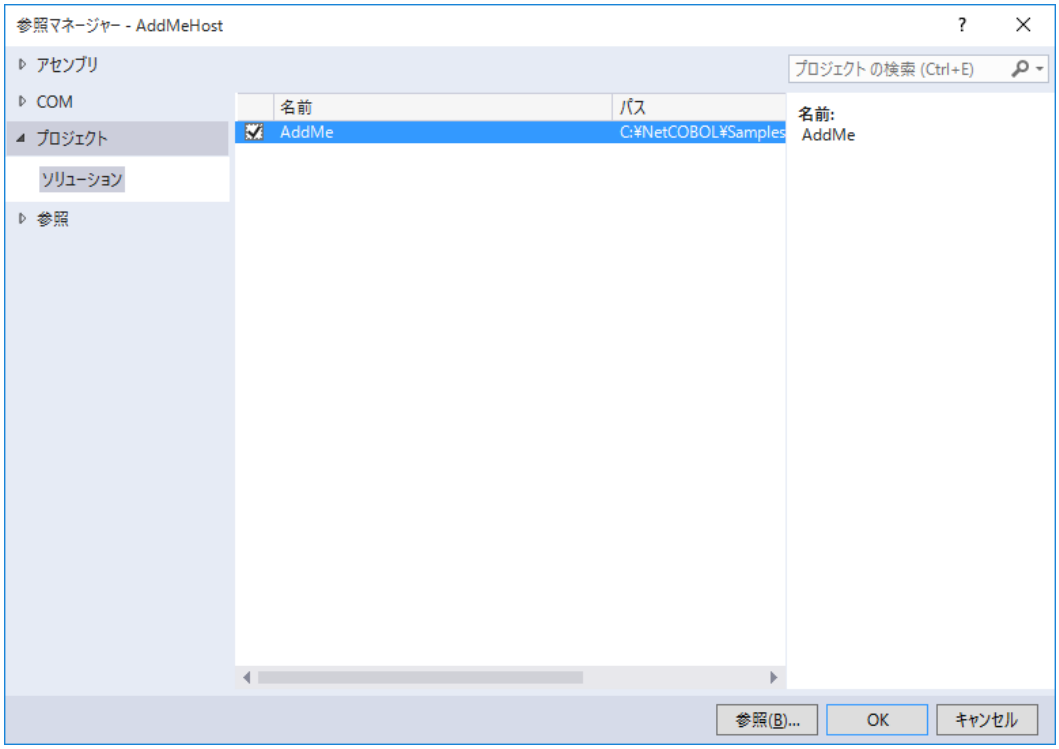
2. ここでは、以下のように Windows フォーム上に二つのボタンを配置します。



3. 二つの数値を加算し結果を返却で作成した WCF サービス ライブラリへの参照を追加します。参照の追加は以下の手順で行います。

   1. [プロジェクト]メニューから[参照の追加]を選択します。
   2. [参照マネージャー]ダイアログボックスで、 [プロジェクト]-[ソリューション] タブを表示します。

3. 参照したい WCF サービス ライブラリプロジェクト名を選択し、[OK]ボタンをクリックします。



4. また、WCF サービスを使用するため[参照マネージャー]ダイアログボックスの[アセンブリ]-[フレームワーク]タブで、以下のコンポーネントの参照を追加します。

   - System.ServiceModel
   - System.Runtime.Serialization

アプリケーションの手続きの記述

1. ソリューションエクスプローラーで、Windows フォーム(Form1.cob)のノードを選択し、右クリックして表示されるメニューから [コードの表示]で手続きコードを開きます。

2. オブジェクト定義(OBJECT 段落)の作業場所節に、二つの数値を加算し結果を返却で作成したサービスホストクラス(AddMe.Service1Host)のオブジェクト参照を追加します。 リポジトリ段落では、AddMe.Service1Host クラスの外部名を CLASS-SERVICE1HOST として定義しています。

   環境部のリポジトリ段落:

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-SERVICE1HOST AS "AddMe.Service1Host"
```

   オブジェクト定義の作業場所節:

```
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 button1 OBJECT REFERENCE CLASS-BUTTON.
01 button2 OBJECT REFERENCE CLASS-BUTTON.
01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
01 HOST1 OBJECT REFERENCE CLASS-SERVICE1HOST.
```

データ名は自由につけて構いません。ここでは、HOST1 としています。

3. Windows フォームのデザイン画面を開き、編集中の Windows フォーム上の[開始]ボタンをダブルクリックして、[開始]ボタンのクリックイベントを手続きを追加します。

4. ここで、WCF サービスホストを開始する手続きを記述します。

```
METHOD-ID. button1_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.

* HOST1 がインスタンス化されていない場合はインスタンス化します。
    IF HOST1 = NULL
        INVOKE CLASS-SERVICE1HOST "NEW" RETURNING HOST1
    END-IF.

* WCF サービスを開始します。
    INVOKE HOST1 "Open".

* 開始ボタンを無効にします。
    SET PROP-ENABLED OF button1 TO B"0".
* 終了ボタンを有効にします。
    SET PROP-ENABLED OF button2 TO B"1".

END METHOD button1_Click.
```

IntelliSense 機能を利用しない場合は、リポジトリ段落で Enabled プロパティの外部名を PROP-ENABLED として定義します。

```
    PROPERTY PROP-ENABLED AS "Enabled"
```

5. [開始]ボタンと同様に[終了]ボタンのクリックイベントの手続きを追加します。 ここでは、WCF サービスホストを終了する手続きを記述します。

```
METHOD-ID. button2_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.

* WCF サービスを終了します。
    IF HOST1 NOT = NULL
        INVOKE HOST1 "Close"
        SET HOST1 TO NULL
    END-IF.

* 開始ボタンを有効にします。
    SET PROP-ENABLED OF button1 TO B"1".
* 終了ボタンを無効にします。
    SET PROP-ENABLED OF button2 TO B"0".

END METHOD button2_Click.
```

6. アプリケーション構成ファイル(App.config)を記述します。 ここでは、WCF サービス ライブラリで追加された[アプリケーション構成ファイル]の<system.serviceModel>～</system.serviceModel>の要素を<configuration>要素内にコピーします。

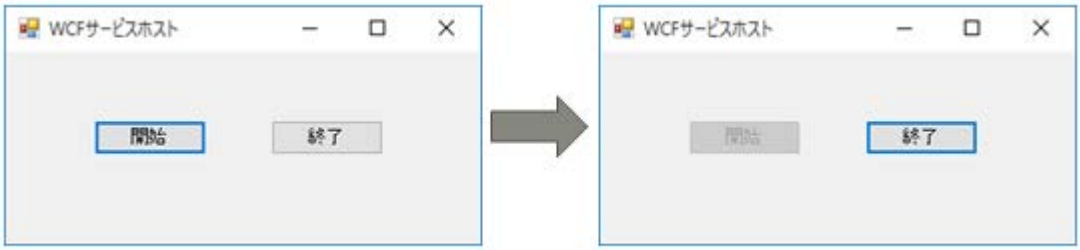
なお、<baseAddress>はサービスホストクラスで指定されているため削除します。 また、Metadata Exchange エンドポイントの定義もここでは削除します。

WCF サービスのエンドポイントが構成されたアプリケーション構成ファイルは以下のとおりです。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6" />
    </startup>
  <system.serviceModel>
    <services>
      <service name="AddMe.Service1">
        <!-- Service Endpoints -->
        <endpoint address="" binding="basicHttpBinding" contract="AddMe.IService1">
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True" />
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから [デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。
2. アプリケーションが実行されると、[開始]ボタンと[終了]ボタンが貼り付けられたウィンドウが表示されます。[開始]ボタンをクリックすると WCF サービスが開始され、[開始]ボタンが無効になります。



3. [終了]をクリックすると WCF サービスは終了します。 アプリケーション右上の[×]ボタンをクリックして、アプリケーションを終了します。

作成したホストアプリケーションから、WCF サービスが正しく起動されているかどうか確認する方法として、メタデータ ユーティリティ ツール (Svcutil.exe)を利用することができます。WCF サービスが正しく起動されている場合は、 プロキシコードが生成されます。以下に、コマンドの例を示します。

```
Svcutil.exe /syncOnly http://localhost:8081/COBOL/Tutorial/AddMe/Service1
```

WCF サービスのクライアントからの利用

WCF サービスを利用するクライアントを作成してみましょう。 ここでは、二つの数値を加算し結果を返却で作成した WCF サービスを利用する、Windows アプリケーションを作成します。

作成手順は以下のとおりです。

1. Windows アプリケーションの作成
2. サービス参照の追加
3. アプリケーションの手続きの記述
4. アプリケーションのビルドと実行

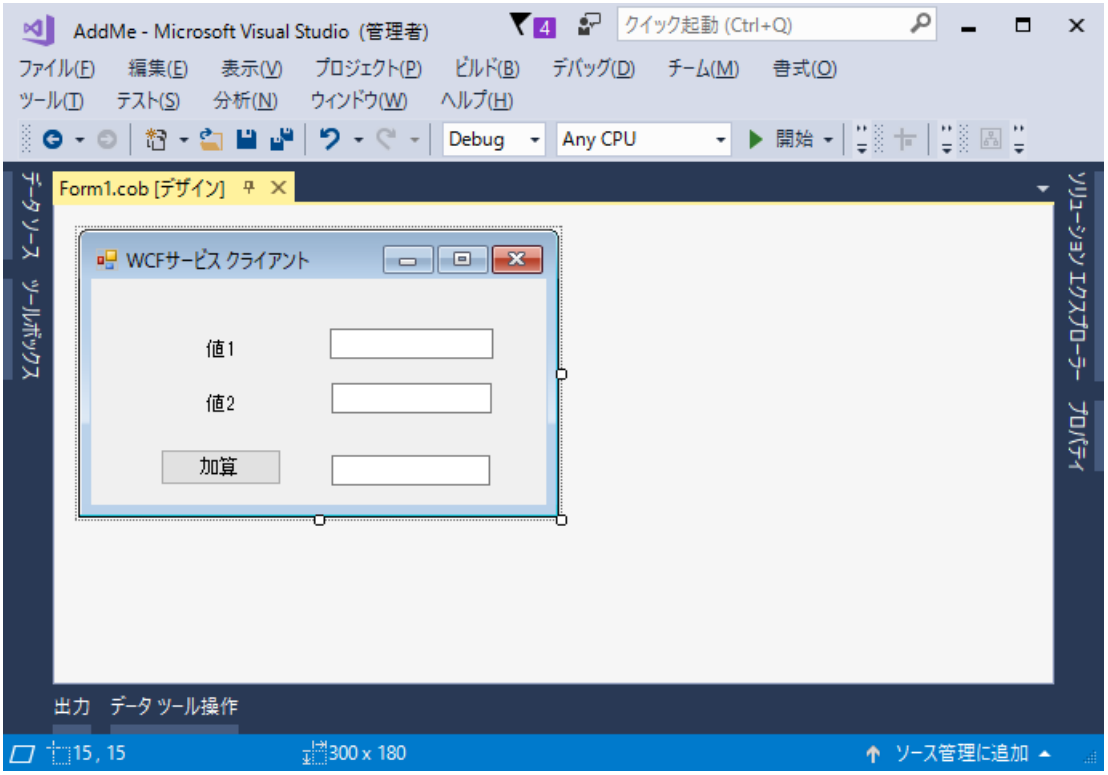
Windows アプリケーションの作成

1. 二つの数値を加算し結果を返却で作成した WCF サービス ライブラリと同じソリューションに、Windows フォーム プロジェクトを追加します。

   Windows フォーム プロジェクトの作成方法については、Windows アプリケーションの開発手順を参考にしてください。ここでは、プロジェクト名を AddMeClient とします。(プロジェクト名は自由につけて構いません。)

   対象のフレームワークは、“.NET Framework 4.7”に設定します。

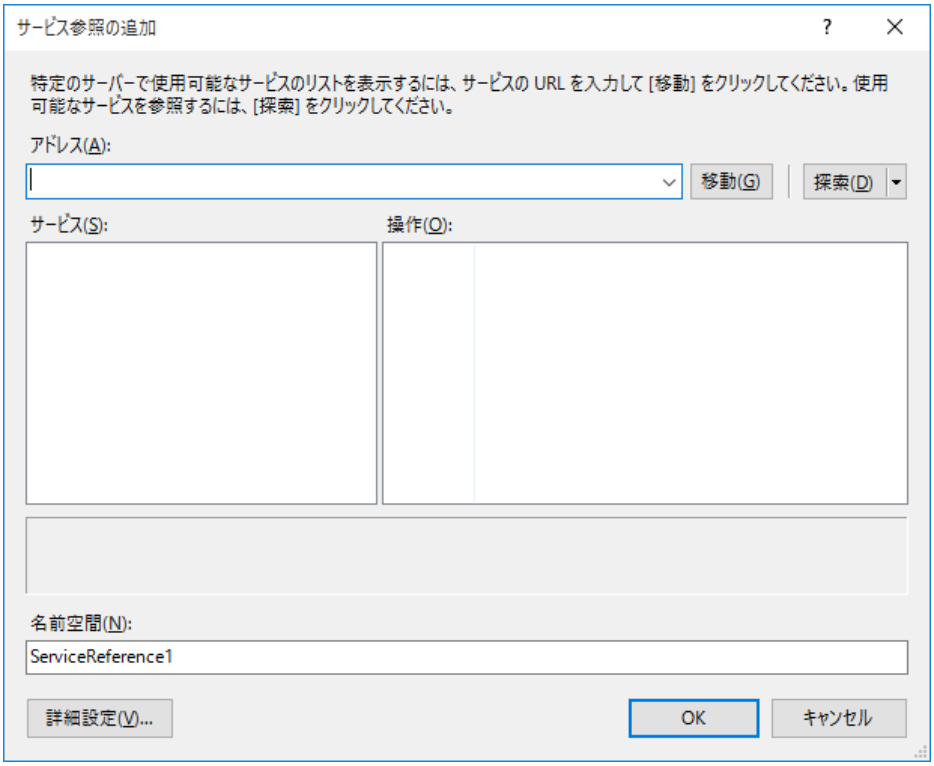
2. ここでは、以下のように Windows フォーム上に二つのラベル、値を入力するための二つのテキストボックスと結果を表示するためのテキストボックス、 および WCF サービスのオペレーションを呼び出して加算処理を実行するためのボタンを配置します。 また、結果を表示するためのテキストボックスは ReadOnly プロパティを True に設定し表示専用とします。



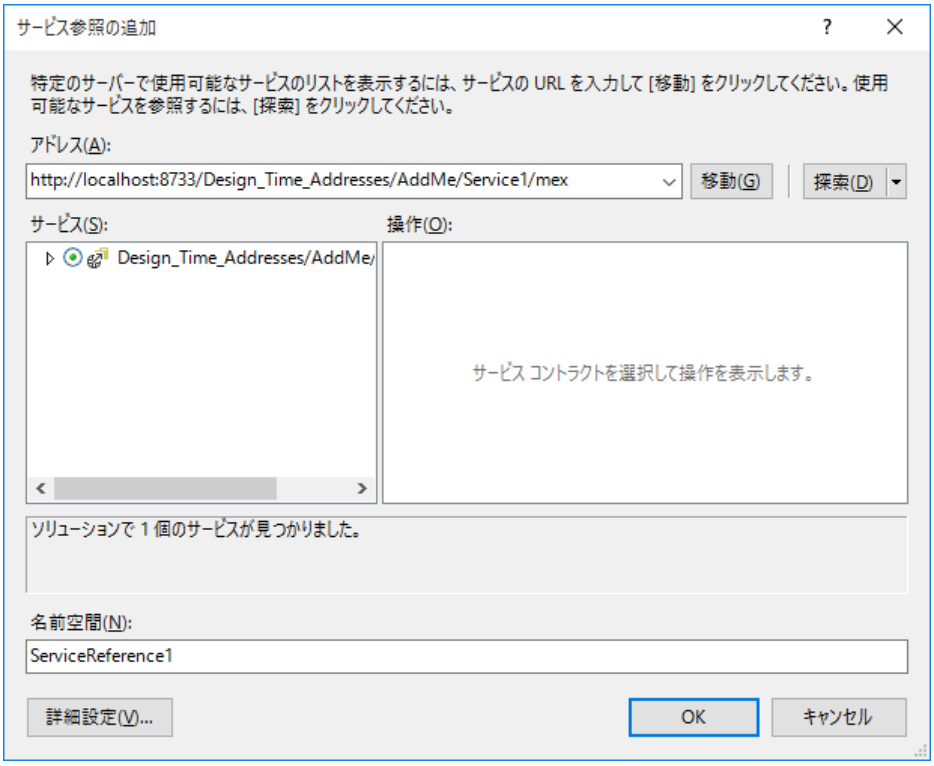
サービス参照の追加

1. 二つの数値を加算し結果を返却で作成した WCF サービスの参照を追加します。

   [プロジェクト]メニューまたはソリューションエクスプローラーの[参照]ノードを右クリックして表示されるメニューから[サービス参照の追加]を選択し、[サービス参照の追加]ダイアログボックスを表示します。



2. [探索]ボタンをクリックし、ソリューション内の WCF サービスを表示します。



   [OK]ボタンをクリックしダイアログボックスを閉じます。 (ここでは[名前空間]ボックスの値は、

“ServiceReference1”のままとします。)

3. 二つの数値を加算し結果を返却で作成した WCF サービスのプロキシコードが自動生成されプロジェクトに追加されます。 このとき、ソリューションエクスプローラーには、"Connected Services"ノードと"ServiceReference1"ノードが作成され、"ServiceReference1"ノードにはプロキシコードを含むファイルがいくつか追加されます。

   App.config ファイルには、WCF サービスに接続するためのクライアントの構成設定が追加されます。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
    </startup>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_IService1" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:8733/Design_Time_Addresses/AddMe/Service1/"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IService1"
                contract="ServiceReference1.IService1" name="BasicHttpBinding_IService1" />
        </client>
    </system.serviceModel>
</configuration>
```

サービス参照に追加については、Visual Studio のドキュメントの方法 : サービス参照を追加、更新、または削除するを参照してください。

アプリケーションの手続きの記述

1. ソリューションエクスプローラーで、Windows フォーム(Form1.cob)のノードを選択し、右クリックして表示されるメニューから [コードの表示]で手続きコードを開きます。

2. 環境部のリポジトリ段落に、以下を追加します。 CLASS-SERVICE1DATA は、プロキシコードとして生成されたデータコントラクト(AddMeClient.ServiceReference1.Service1Data)のクラス名です。 CLASS-SERVICE1CLIENT は、プロキシコードとして生成されたサービスクライアント(AddMeClient.ServiceReference1.Service1Client)のクラス名です。 (プロジェクト名を AddMeClient 以外で作成した場合は、それにあわせて記述します。)

```
REPOSITORY.
    CLASS CLASS-SERVICE1CLIENT AS "AddMeClient.ServiceReference1.Service1Client"
    CLASS CLASS-SERVICE1DATA AS "AddMeClient.ServiceReference1.Service1Data"
    CLASS CLASS-INT32 AS "System.Int32"
```

3. Windows フォームのデザイン画面を開き、編集中の Windows フォーム上の[加算]ボタンをダブルクリックして、[加算]ボタンのクリックイベントを手続きを追加します。

```
METHOD-ID. button1_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-CLIENT OBJECT REFERENCE CLASS-SERVICE1CLIENT.
```

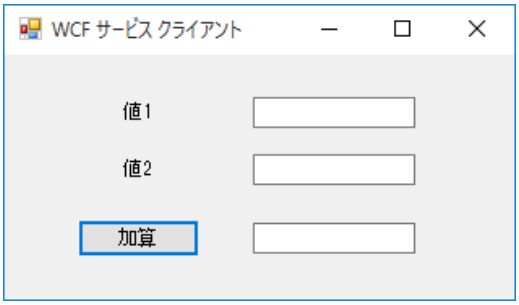
```
       01 WK-DATA OBJECT REFERENCE CLASS-SERVICE1DATA.
       01 WK-TEXT1 OBJECT REFERENCE CLASS-STRING.
       01 WK-TEXT2 OBJECT REFERENCE CLASS-STRING.
       01 WK-RETVAL OBJECT REFERENCE CLASS-INT32.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.

      * データコントラクトクラスを初期化します。
          INVOKE CLASS-SERVICE1DATA "NEW" RETURNING WK-DATA.

      * textBox1 の Text プロパティから文字列を取得します。
          SET WK-TEXT1 TO PROP-TEXT OF textBox1.

      * 取得した文字列を Int32 に変換し、データコントラクトのメンバー(VALUE1)に設定します。
          SET PROP-VALUE1 OF WK-DATA TO CLASS-INT32::"Parse"(WK-TEXT1).

      * textBox2 の Text プロパティから文字列を取得します。
          SET WK-TEXT2 TO PROP-TEXT OF textBox2.

      * 取得した文字列を Int32 に変換し、データコントラクトのメンバー(VALUE2)に設定します。
          SET PROP-VALUE2 OF WK-DATA TO CLASS-INT32::"Parse"(WK-TEXT2).

      * WCF サービスのクライアントクラスを初期化します。
          INVOKE CLASS-SERVICE1CLIENT "NEW" RETURNING WK-CLIENT.

      * WCF サービスの Operation2 メソッドを呼び出します。
          INVOKE WK-CLIENT "Operation2" USING WK-DATA RETURNING WK-RETVAL.

      * WCF サービスクライアントをクローズします。
          INVOKE WK-CLIENT "Close".

      * 結果を表示します。
          SET PROP-TEXT OF textBox3 TO WK-RETVAL::"ToString".

       END METHOD button1_Click.
```

IntelliSense 機能を利用しない場合は、リポジトリ段落で VALUE1 および VALUE2 プロパティの外部名をそれぞれ、PROP-VALUE1、PROP-VALUE2 として定義します。

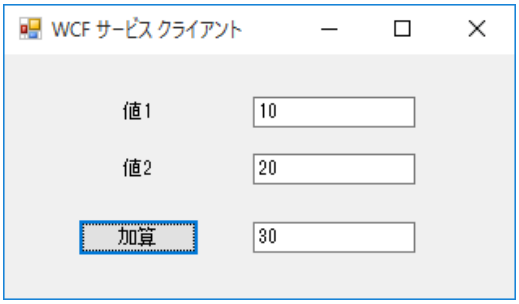
```
           PROPERTY PROP-VALUE1 AS "VALUE1"
           PROPERTY PROP-VALUE2 AS "VALUE2"
```

アプリケーションのビルドと実行

1. Visual Studio 上で、[デバッグ]メニューから[デバッグの開始]を選択すると、 アプリケーションがビルドされ、デバッガー 配下でアプリケーションが実行されます。二つの数値を加算し結果を返却で作成した WCF サービス ライブラリと同じソリューションであるため、同時に WCF のテスト用クライアント(WcfTestClient.exe)も起動します。



2. アプリケーション上で、二つの TextBox に任意の数値を入力し[加算]ボタンをクリックします。 加算された結果が表示されるかを確認します。



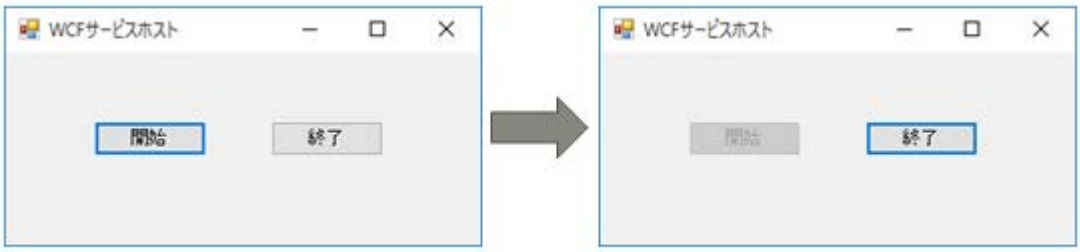
3. アプリケーションを終了します。
4. 続いて、WCF サービスをホストするで作成したホストアプリケーションから起動した WCF サービスに接続してみましょう。
5. まず、クライアントアプリケーションの App.config ファイルを開き、 ホストアプリケーションのエンドポイントの構成設定にあわせて、クライアントアプリケーションのエンドポイントの構成設定を変更します。

   エンドポイントの構成は以下のとおりです。

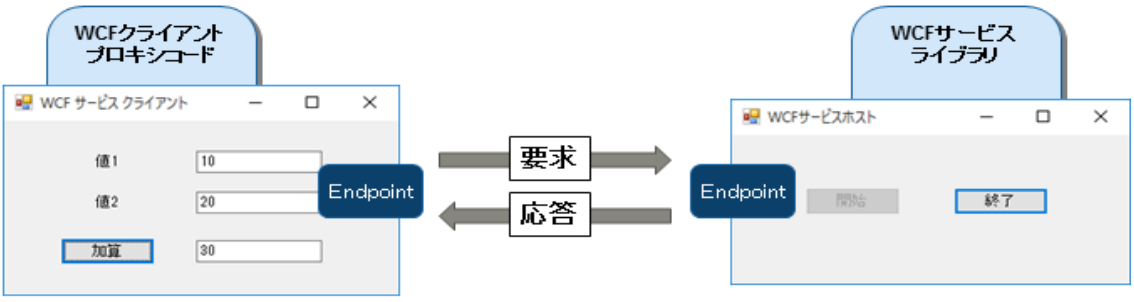
Address	http://localhost:8081/COBOL/Tutorial/AddMe/Service1
Binding	basicHttpBinding
Contract	ServiceReference1.IService1

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6" />
    </startup>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_IService1" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:8081/COBOL/Tutorial/AddMe/Service1"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IService1"
                contract="ServiceReference1.IService1" name="BasicHttpBinding_IService1"> />
        </client>
    </system.serviceModel>
</configuration>
```

6. WCF サービスをホストするで作成したホストアプリケーションを起動します。 Visual Studio から起動する場合は、以下の手順で起動します。
   1. ソリューションエクスプローラーでホストアプリケーションのプロジェクトを選択します。
   2. [プロジェクト]メニューから[スタートアッププロジェクトの設定]を選択します。
   3. [デバッグ]メニューから[デバッグなしで開始]を選択します。
7. ホストアプリケーションの[開始]ボタンをクリックし、クライアントからの要求を待ち受けます。



8. 続いて、クライアントアプリケーションを起動します。
9. ソリューションエクスプローラーで、クライアントアプリケーションプロジェクトを選択し、 [プロジェクト]メニューから[スタートアッププロジェクトの設定]を選択します。
10. クライアントアプリケーション上で、二つの TextBox に任意の数値を入力し[加算]ボタンをクリックし、WCF サービスからの応答を待ちます。



通信が正しく行われると、結果が表示されます。

11. ホストアプリケーションの[終了]ボタンをクリックし、WCF サービスを終了します。
12. ホストアプリケーションとクライアントアプリケーションをそれぞれ終了します。

ASP.NET Web アプリケーションの実行のための事前準備

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

ASP.NET Web サイトのテストまたは実行には、Web サーバーが必要となります。 ASP.NET Web サイトの運用時には、Web サーバーとして IIS(Microsoft Internet Information Services)を 使用します。

Visual Studio 2017 では、 ASP.NET Web サイトを次の場所に構築してテストすることができます。

- ローカル IIS Web サイト
- ファイル システム Web サイト
- FTP 配置 Web サイト
- リモート IIS Web サイト

これらの Web サイトの詳細は、Visual Studio のドキュメントの Visual Web Developer における Web サイトの種類を参照してください。

ファイル システム Web サイトを使用する場合は、IIS の代わりに IIS Express を使って ASP.NET Web サイトをテストすることができます。

ASP.NET Web サイトをローカル IIS Web サイトで実行させるためには、 IIS(Microsoft Internet Information Services)および ASP.NET を、あらかじめ適切に構成しておく必要があります。

IIS がインストールされているときに.NET Framework をインストールした場合は、 ASP.NET が適切に構成されますが、IIS がインストールされていないときに .NET Framework をインストールした場合、または IIS を再インストール した場合には、ASP.NET が適切に構成されていません。この場合は、ASP.NET IIS 登録ツール を使って ASP.NET を再構成する必要があります。ASP.NET を再構成する場合は、次のコマンドを NetCOBOL for .NET コマンド プロンプトから実行します。 NetCOBOL for .NET コマンド プロンプトについては、コマンドプロンプトの起動を参照してください。

```
"%windir%¥Microsoft.NET¥Framework¥v4.0.30319¥aspnet_regiis.exe" -i
```

ASP.NET IIS 登録ツールの詳細は、 .NET Framework のドキュメントの ASP.NET IIS 登録ツール (Aspnet_regiis.exe)を参照してください。

注意事項

- プロジェクトが自動生成する web.config は、そのプロジェクトを含むディレクトリが、 仮想ディレクトリのルートであることを想定して作成されます。 そのため、 仮想ディレクトリとしたディレクトリのサブディレクトリにプロジェクトを作成する場合は、そのままでは動作しません。 以下の操作が必要です。

  サブディレクトリに作成したプロジェクトを部品として使用する場合

  通常、アプリケーションは仮想ディレクトリのルートに作成します。 そして、アプリケーションは 1 つの web.config(構成ファイル)を持ちます。 部品としてのプロジェクトをサブディレクトリに作成する場合は、 ルートディレクトリの構成ファイルが有効になるようにします。 そのためには、以下の操作を行なってください。

  - サブディレクトリ配下の web.config を削除します。
  - 部品として作成した.dll を、仮想ディレクトリ配下の¥bin ディレクトリに配置

します。

サブディレクリに作成したプロジェクトをアプリケーションとして使用する場合

仮想ディレクトリのサブディレクトリに作成したプロジェクトを、 アプリケーションとして動作させたい場合は、IIS 上でサブディレクトリを選択し、プロパティを表示させて、[ディレクトリ]タブで「アプリケーションの設定」の[作成]ボタンをクリックします。

- 仮想ディレクトリ配下のファイルは、 そのまま IIS 上ですべてアクセス可能となっています。 .aspx、.asmx、.htm、.svc およびイメージファイル以外は、 IIS によって個別に適切なセキュリティ設定をおこなうことをお勧めします。

  - NetCOBOL for .NET で自動生成する web.config には、 以下の記述が含まれます。

```
<compilers>
  <compiler extension=".cobx;.cob" language="Fujitsu.COBOL;NetCOBOL;cobol"
   type="Fujitsu.COBOL.COBOLCodeProvider,Fujitsu.COBOL.CodeDom,
   Version=8.0.124.0,Culture=neutral,PublicKeyToken=fac0fe3cab973246,
   processorArchitecture=MSIL" compilerOptions="/wc:SCS(ACP)" />
</compilers>
```

    この情報はこの構成ファイルが有効な環境下で、 COBOL で作成したアプリケーションを利用できることを指示しますが、 マシン情報ファイルに上記の情報を組み込むことによって、 意識することなく COBOL で作成したアプリケーションを利用できるようになります。

    マシン構成ファイル(machine.config)は、
    `%windir%¥Microsoft.NET¥Framework¥(version)¥CONFIG` にあります。

  - IIS の詳細については IIS のドキュメントを参照してください。 (http://localhost/IIShelp)

    なお、http://localhost が参照できない場合は、ブラウザの プロキシ サーバーの設定が不十分 である可能性があります。 Microsoft Internet Explorer をお使いの場合は、 以下の操作を行なって設定を確認してください。

    1. [ツール]メニューから[インターネットオプション]をクリックし、 [接続]タブを選択します。
    2. インターネットオプションページが表示されます。 [ローカルエリア ネットワーク(LAN)の設定]で[LAN の設定]ボタンをクリックします。
    3. ローカルエリア ネットワーク(LAN)の設定ページが表示されます。 [プロキシ サーバー]の[LAN にプロキシ サーバーを使用する]がチェックされている場合、[ローカルアドレスにはプロキシサーバを使用しない]をチェックします。

NetCOBOL for .NET アプリケーションの開発

NetCOBOL for .NET で開発を行う場合、 Visual Studio IDE（統合開発環境）を利用する方法とコマンドラインから開発する方法が選択できます。 Visual Studio IDE を使って開発する方法では、 高度なビジュアルインタフェースを利用して、アプリケーションデザインから、 コンパイル、デバッグまでの多くの作業を行うことができます。 ユーザは、マウスを数回クリックするだけでこれらの機能を利用できるため、 簡単にメンテナンスのしやすい高度なアプリケーションを作成することができます。

NetCOBOL for .NET を使ったアプリケーションの開発では、 この IDE を利用した生産性の高い環境下での開発をお薦めします。

ここでは、 NetCOBOL for .NET でアプリケーションを開発する場合の作業の流れに合わせて、それぞれの手順について説明していきます。

まず、Visual Studio IDE(統合開発環境)で作業する場合の流れと、 コマンドラインで作業する場合の流れを簡単に説明し、 次に、それぞれの作業の詳細な説明に移ります。

このセクションの内容

開発のための準備

開発作業を行う前に必要な準備について説明します。

開発の流れ

Visual Studio IDE を利用して開発する方法と コマンドラインから開発する方法について説明します。

COBOL が使用するファイル

NetCOBOL for .NET で使用する、 COBOL 特有のファイルについて説明します。

プロジェクトの作成と管理

Visual Studio でプロジェクトを作成する方法について説明します。

COBOL ソースプログラムの作成・編集

Visual Studio エディターを使用した COBOL ソースプログラムの編集方法と COBOL の正書法について説明します。

ビルド

翻訳オプションを指定する方法や Visual Studio で作成したプロジェクトのビルド方法、 cobolc コマンドを使ったビルド方法について説明します。

アプリケーションの実行

実行環境を設定する方法や作成したアプリケーションを実行する方法について説明します。

デバッグ

Visual Studio デバッガーの使い方について説明します。 プロジェクトから作成したアプリケーションは、 このデバッガーを使用してデバッグすることができます。

アプリケーション開発時の注意事項

ASP.NET アプリケーションの開発および Windows アプリケーションの開発における注意事項について説明します。

ツール

NetCOBOL for .NET が提供するツールについて説明します。

開発のための準備

Visual Studio 2017 を利用して NetCOBOL for .NET アプリケーションを開発する場合、 開発するアプリケーションの種類に応じて、必要なワークロードを準備します。 ワークロードは、Visual Studio インストーラーを起動して選択することにより、インストールします。 以下に各アプリケーションの開発に必要なワークロードを示します。

アプリケーションの種類	必要な Visual Studio ワークロード
すべてのアプリケーション	.NET デスクトップ開発
ASP.NET アプリケーション	ASP.NET と Web 開発
SQL Server アプリケーション	データの保存と処理
WCF サービスアプリケーション	ASP.NET と Web 開発(必要となる“Windows Communication Foundation”コンポーネントが含まれています)
ClickOnce 配置を行なうアプリケーション	.NET デスクトップ開発(必要となる“ClickOnce Publishing”コンポーネントが含まれています)

SQL Server 向けアプリケーション開発のための準備

リモートコンピュータの SQL Server に配置された SQL Server 向け NetCOBOL for .NET アプリケーションをデバッグする場合は、リモートデバッグを行う必要があります。 リモートデバッグのセットアップ方法に関しては、 「リモートデバッグのセットアップ手順」を参照してください。

また、SQL Server 向けアプリケーションを運用するには、 NetCOBOL サーバ運用パッケージ for .NET が必要です。

リモートデバッグのための準備

リモートデバッグのセットアップ手順について説明します。

NCOBU00913

リモートデバッグのセットアップ手順

リモートデバッグを行うためには、 デバッグ対象アプリケーションを動かすリモートマシンに対して以下の手順で準備作業を行う必要があります。

1. .NET Framework をインストールする
2. NetCOBOL for .NET ランタイムシステムをインストールする
3. Visual Studio 2017 のリモートデバッグコンポーネントをインストールする
4. NetCOBOL for .NET のリモートデバッグコンポーネントをインストールする

リモートデバッグを開始する方法に関しては、リモートデバッグを参照してください。

.NET Framework をインストールする

.NET Framework アプリケーションを実行するには、 .NET Framework が必要です。 リモートマシンに.NET Framework 4 以降がインストールされていない場合は、 インストールします。

NetCOBOL for .NET ランタイムシステムをインストールする

NetCOBOL for .NET アプリケーションを実行するには、 リモートマシンに NetCOBOL for .NET ランタイムシステムをインストールする必要があります。 NetCOBOL for .NET ランタイムシステムは、NetCOBOL for .NET 製品によりインストールされます。

Visual Studio 2017 のリモートデバッグコンポーネントをインストールする

.NET Framework アプリケーションのリモートデバッグを可能にするために、 リモートマシンに Visual Studio 2017 のリモートデバッグコンポーネントをインストールします。 詳細は、Visual Studio のドキュメントの「リモート デバッグのセットアップ」を参照してください。

ファイアウォールを構成する

ファイアウォールを有効にしている環境では、 Visual Studio 2017 のリモートデバッグコンポーネントをインストールしたあとに、 追加のファイアウォールの構成が必要となる場合があります。

ファイアウォールの構成はホストマシンとリモートマシンの双方に対し、以下に示す受信(着信)の規則を追加します。 ファイアウォールの構成方法については、ご利用の Windows またはセキュリティソフトによって異なるため、 それぞれのドキュメントを参考にしてください。

ホストマシン:

- 次のローカルポートへの接続を許可します。

```
TCP 135 (必須)
UDP 500, 4500 (ドメイン ポリシーによって IPSec を使用したネットワーク通信を行う必要がある場合)
```

- 次のプログラムへの接続を許可します。

```
<Visual Studio のインストールフォルダー(*)>¥Common7¥IDE¥devenv.exe
```

*: <Visual Studio のインストールフォルダー>は通常は、
" %ProgramFiles%¥Microsoft Visual Studio¥2017¥(エディション名)"です。

リモートマシン:

- 次のローカルポートへの接続を許可します。

```
TCP 135, 139, 445 (必須)
UDP 137, 138 (必須)
UDP 500, 4500 (ドメイン ポリシーによって IPSec を使用したネットワーク通信を行う必要がある場合)
```

- 次のプログラムへの接続を許可します。

```
<Visual Studio のインストールフォルダー(*)>¥Common7¥IDE¥Remote Debugger¥x86¥msvsmon.exe
<Visual Studio のインストールフォルダー(*)>¥Common7¥IDE¥Remote Debugger¥x64¥msvsmon.exe
(64 ビットマシンの場合)
```

*: <Visual Studio のインストールフォルダー>は通常は、
"%ProgramFiles%¥Microsoft Visual Studio 15.0"、または、
"%ProgramFiles%¥Microsoft Visual Studio¥2017¥(エディション名)"です。

なお、ファイアウォールの構成変更によって、セキュリティが低下する場合があるため、 リモートアクセスのコンピュータの範囲を制限することをお勧めします。

NetCOBOL for .NET のリモートデバッグコンポーネントをインストールする

NetCOBOL for .NET アプリケーションのリモートデバッグを可能にするために、リモートマシンに NetCOBOL for .NET のリモートデバッグコンポーネントをインストールします。

x86 プラットフォームのアプリケーションをリモートデバッグする場合

NetCOBOL 開発パッケージ for .NET をインストールしたフォルダの `Remote Debugging Components¥x86¥setup.exe` をリモートマシン上にコピーして実行します。

x64 プラットフォームのアプリケーションをリモートデバッグする場合

NetCOBOL 開発パッケージ for .NET をインストールしたフォルダの `Remote Debugging Components¥x64¥setup.exe` をリモートマシン上にコピーして実行します。

NCOBU00036

開発の流れ

NetCOBOL for .NET で開発を行う場合、 Visual Studio IDE を利用する方法と コマンドラインから開発する方法が選択できます。 ここでは、これらの開発の流れをコンソールアプリケーションの開発を例にして、 簡単に説明します。

Visual Studio IDE を使った開発

Visual Studio IDE を利用して開発する方法について説明します。 プロジェクトの作成からソースプログラムの編集、ビルド、実行、 デバッグまで簡単に説明します。

コマンドラインからの開発

コマンドラインから開発する方法について説明します。 ソースプログラムの編集、ビルド、実行まで簡単に説明します。

コマンドラインからの開発では､デバッグは IDE から行うか、または Microsoft Windows SDK 付属のデバッガーで行います。 これらについて、ここでは詳しく説明しません。 Microsoft Windows SDK 付属のデバッガーの詳細については、 .NET Framework のドキュメントの.NET Framework ツールの「デバッグ ツール」を参照してください。

Visual Studio IDE を使った開発

プロジェクトの作成からソースプログラムの編集、ビルド、実行、デバッグまでの流れを以下の順で簡単に説明します。 詳細な説明については、関連項目を参照してください。

	手順	関連項目
1	Visual Studio を起動する	
2	プロジェクトを作成する	プロジェクトの作成と管理
3	COBOL ソースプログラムを作成・編集する	COBOL ソースプログラムの作成・編集
4	ビルドする	NetCOBOL for .NET プロジェクトのビルド
5	実行する	アプリケーションの実行
6	デバッグする	デバッグ

ここでは、 簡単なサンプルプログラム(Hello World)をコンソールアプリケーションとして作成します。

Windows アプリケーション、ASP.NET Web アプリケーション、 XML Web サービス、および Windows Communication Foundation サービスの開発の開発の流れについては、 以下のチュートリアルの最初のステップを参照してください。

アプリケーションの種類	チュートリアルの参照先
Windows アプリケーションの開発	メッセージボックスを表示する
ASP.NET Web アプリケーションの開発	ラベルに文字列を設定する
XML Web サービスの開発	文字列を返却する
Windows Communication Foundation サービスの開発	文字列を返却する

以下に、コンソールアプリケーションの開発手順を示します。

1. Visual Studio を起動する

   [スタート] - アプリ一覧(*) - [Visual Studio 2017]を選択します。

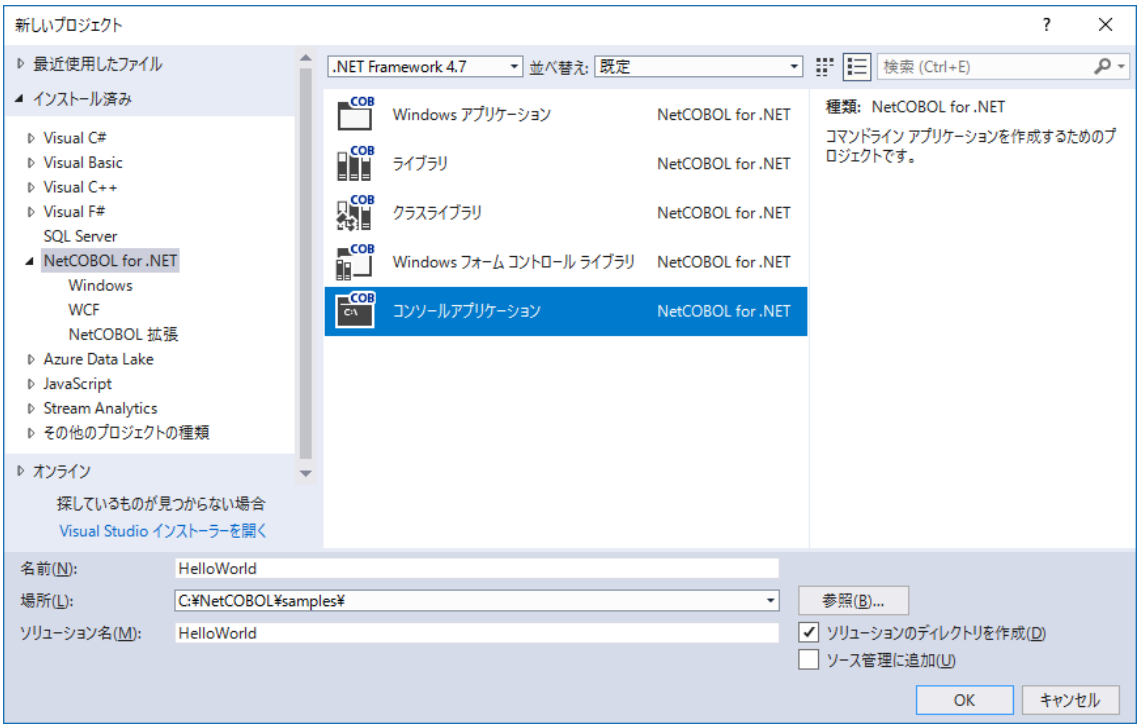
   * : ご使用の Windows によって、以下の操作をすることで同様の画面になります。

   - Windows 7 : [スタート]メニュー - [すべてのプログラム]
   - Windows 8.1 および Windows Server 2012 R2 : [スタート]画面 - [↓] - [アプリ]

2. プロジェクトを作成する

   新しくアプリケーションを作成するには、 まずプロジェクト(プログラムを管理する単位)を作成し、 アプリケーションの種類と格納場所を指定します。

   [新しいプロジェクト]ダイアログボックスは、スタートページウィンドウの[新しいプロジェクトの作成]をクリックするか、 [ファイル]メニューから、[新規作成]-[プロジェクト]を選択します。

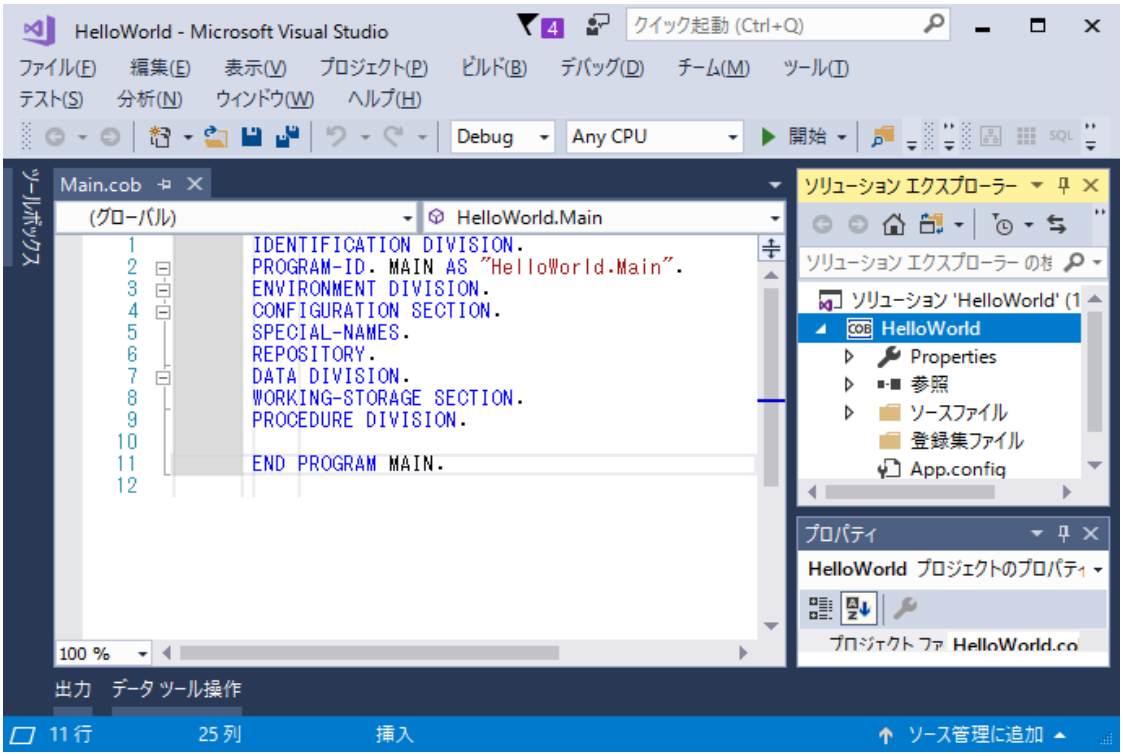


ここでは、以下の設定でプロジェクトを作成します。

対象のフレームワーク	.NET Framework 4.7
左ペイン	NetCOBOL for .NET
右ペイン	コンソールアプリケーション
名前	HelloWorld
場所(*1)	C:¥NetCOBOL¥Samples
ソリューションのディレクトリを作成	チェック

*1: 説明の都合上、プロジェクトの格納フォルダを、任意のフォルダにしています。

[OK]ボタンをクリックすると、 アプリケーションの構築に必要なファイル一式が作成されます。



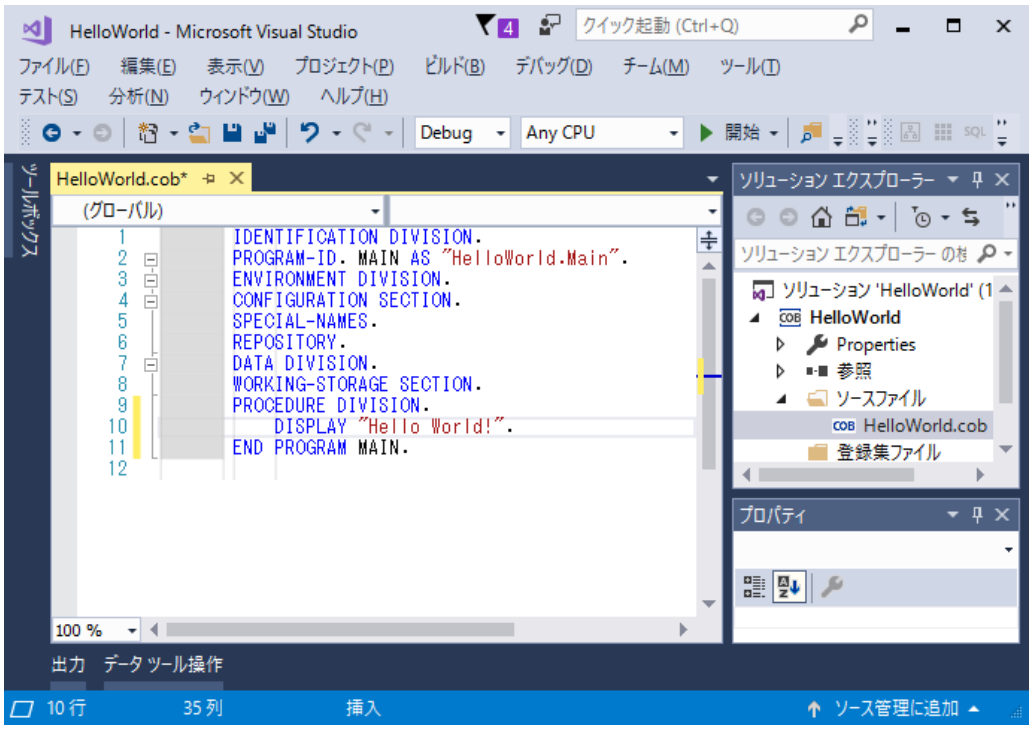
3. COBOL ソースプログラムを作成・編集する

    1. ソースプログラムの名前を変更します。 ソースファイルのツリーの下にある「 Main.cob」を「HelloWorld.cob」に変更してください。

    2. ソースプログラムを作成します。 「HelloWorld.cob」をダブルクリックすると、ソースプログラムの編集画面が表示されます。

    3. この時点では、テンプレートから生成されたソースプログラムが表示されます。

    4. ソースプログラムに、DISPLAY 文を追加してみましょう。 DISPLAY 文は B 領域(12 カラム目以降)から書き始めます。



5. ソースプログラムを保存します。 [ファイル]メニューから[HelloWorld.cob を保存]を選択すると、保存されます。

4. ビルドする

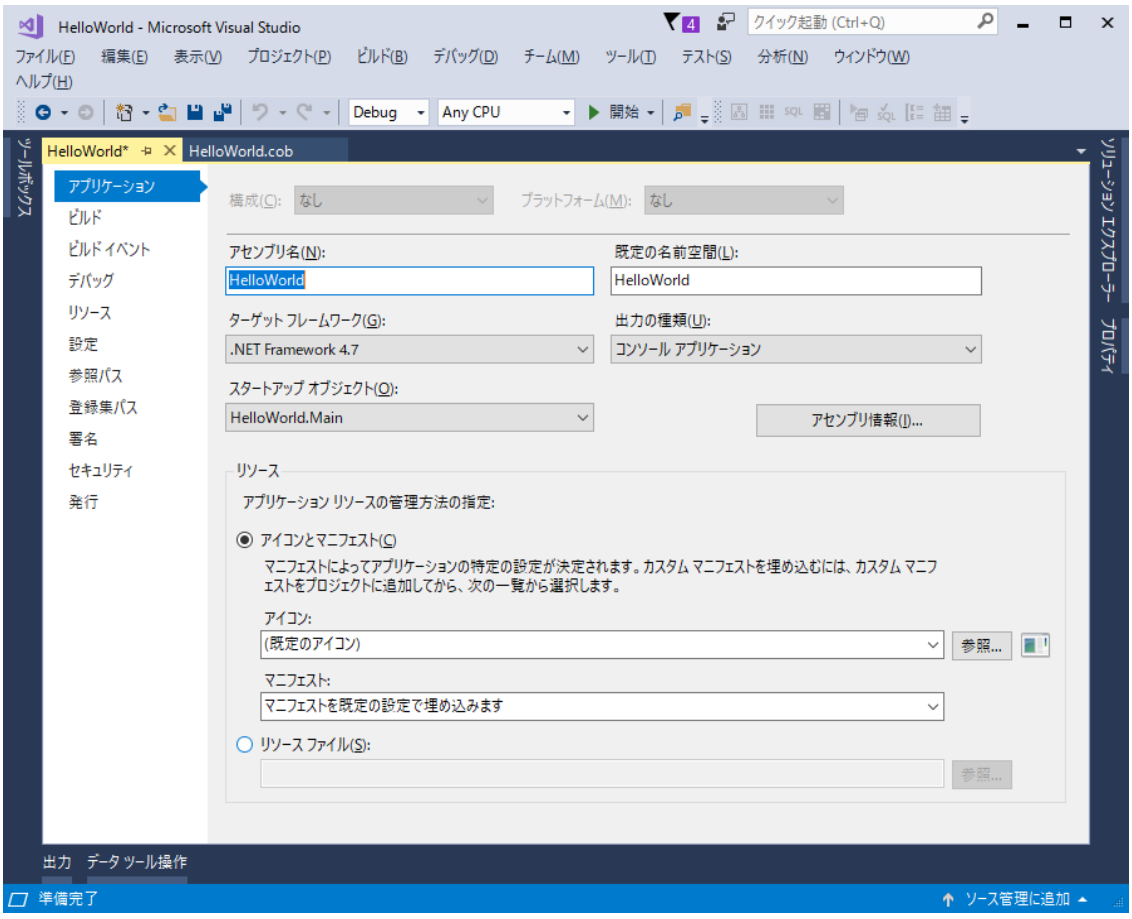
ビルドする前に、翻訳オプションやデバッグ情報の生成を指定します。 これらの指定はプロジェクトのプロパティページから行います。 また、出力ファイル先も確認しておきましょう。

まず、プロジェクトのプロパティ ウィンドウで、プロジェクトフォルダを確認します。 プロジェクトのプロパティ ウィンドウは、次の方法で表示させることができます。

1. ソリューションエクスプローラーからプロジェクトを選択します。
2. [表示]メニューから[プロパティ ウィンドウ]を選択します。

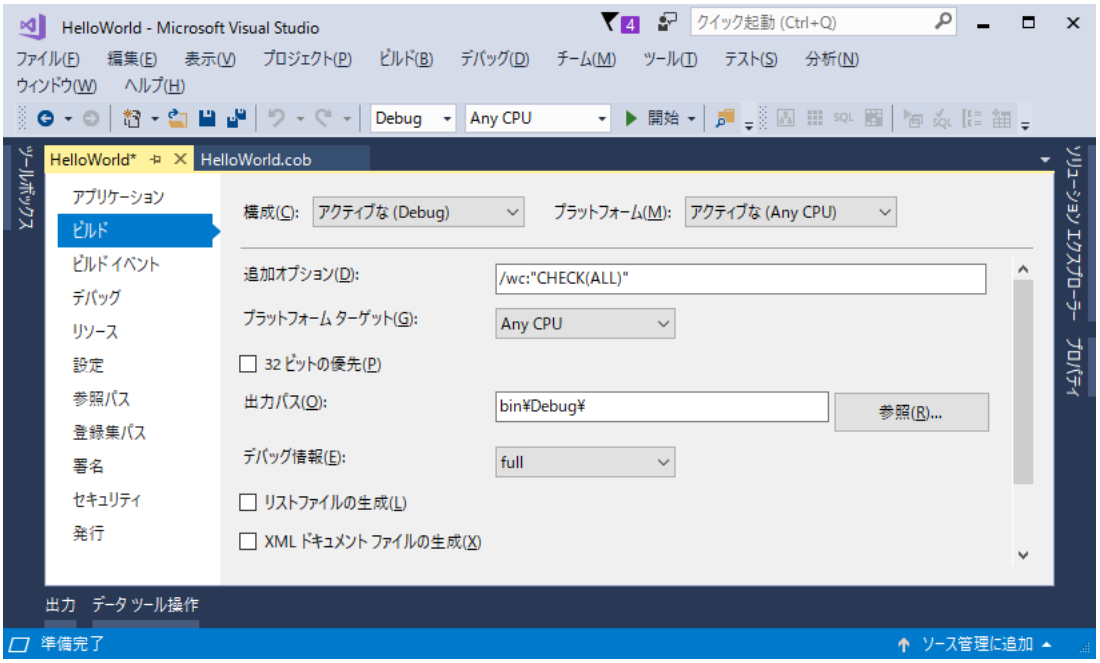
"プロジェクト フォルダ"プロパティには、ここでは `C:¥NetCOBOL¥samples¥HelloWorld¥HelloWorld¥`が表示されています。

プロパティページを表示するには、 [プロジェクト]メニューから[HelloWorld のプロパティ]を選択します。[HelloWorld]のプロパティページが表示されたら、左ペインの [アプリケーション] を選択します。



右ペインの[アセンブリ名]を確認します。ここでは、HelloWorld となっています。

アセンブリ名に、アプリケーションタイプに従った拡張子を付けた名前が、出力ファイル名となります。コンソールアプリケーションの場合、拡張子は .exe になります。 ここでは出力ファイルは HelloWorld.exe となります。 次に、左ペインの[ビルド]を選択します。



- デバッグ情報の生成では、あとでデバッグできるように、 デバッグ情報の生成を指定します。右ペインの[デバッグ情報]が full になっていることを確認してください。 full が選択されていない場合は、full を選択します。
- 翻訳オプションを指定する場合は、[追加オプション]の欄にコンパイラオプションを指定します。ここでは、翻訳オプション CHECK(ALL)をコンパイラオプション/wc を使用して指定しています。
- 出力パスを確認します。 ここでは、「`bin¥Debug¥`」となっています。 出力パスは変更することができます。

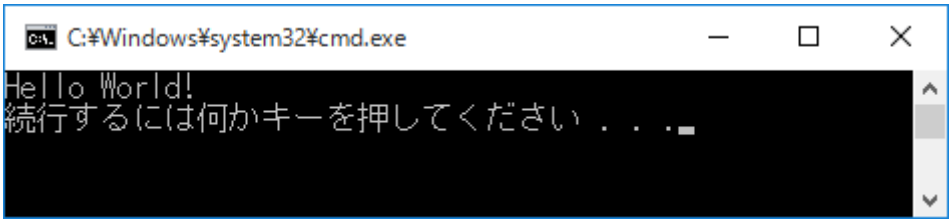
出力ファイル(HelloWorld.exe)は、 プロジェクトフォルダと出力パスで指定された場所に作成されます。ここでは、`C:¥NetCOBOL¥Samples¥HelloWorld¥HelloWorld¥bin¥Debug¥`に作成されます。

準備ができたら、[HelloWorld]のプロパティページを閉じ、 ビルドしてみましょう。

[ビルド]メニューから[ソリューションのビルド]または[ソリューションのリビルド]を選択します。

5. 実行する

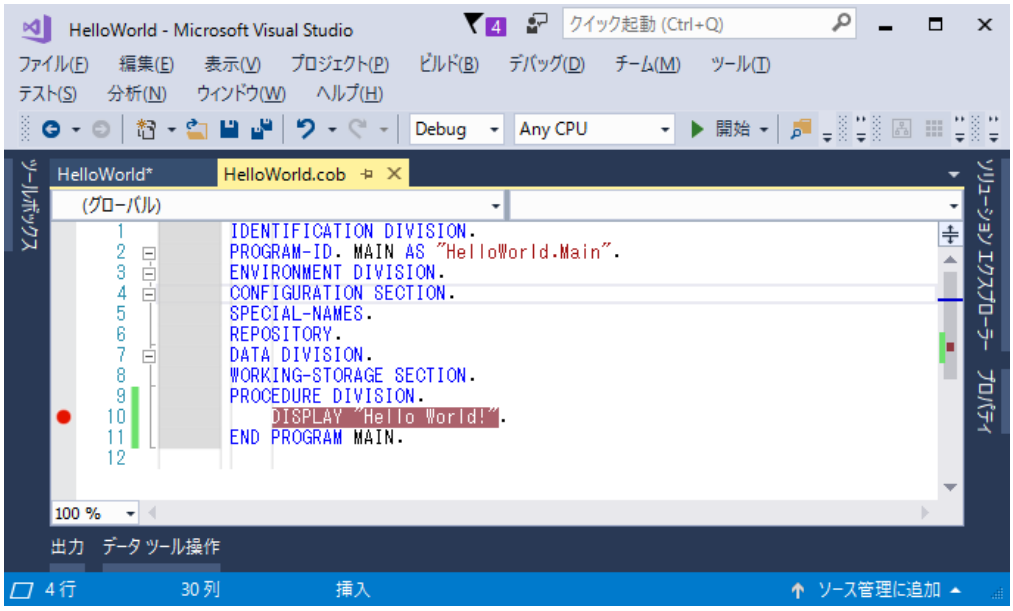
デバッグをしない実行では、[デバッグ]メニューから[デバッグなしで開始]を選択します。



「Hello World!」が表示されます。

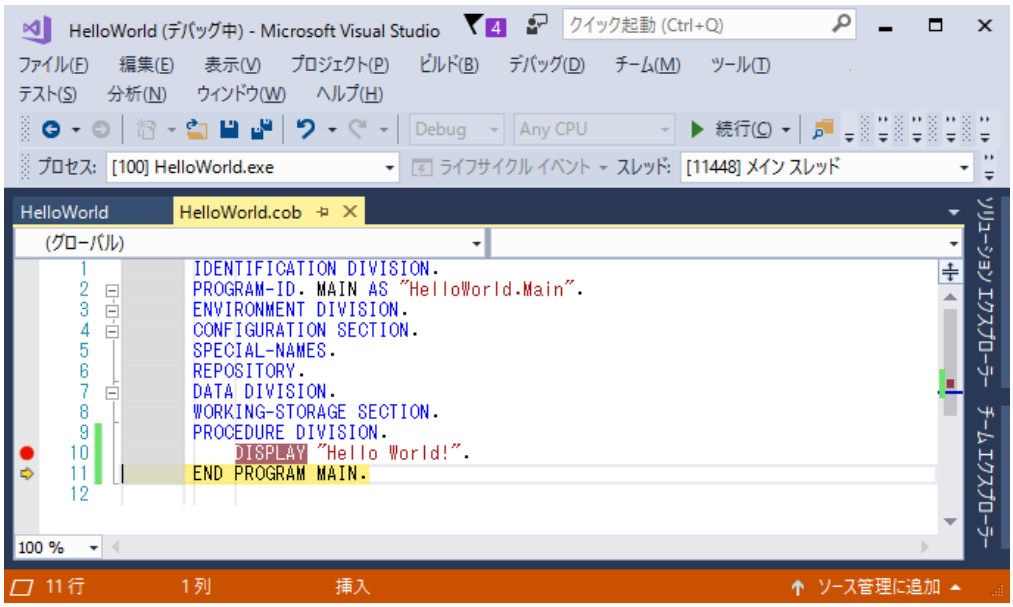
6. デバッグする

1. まず、ブレークポイントを DISPLAY 文の位置に設定してみましょう。 HelloWorld.cob のソースプログラムを表示させます。 10 行目の DISPLAY 文の位置にカーソルを合わせ、右クリックします。 [ブレークポイントの挿入]を選択すると、ブレークポイントが設定されます。

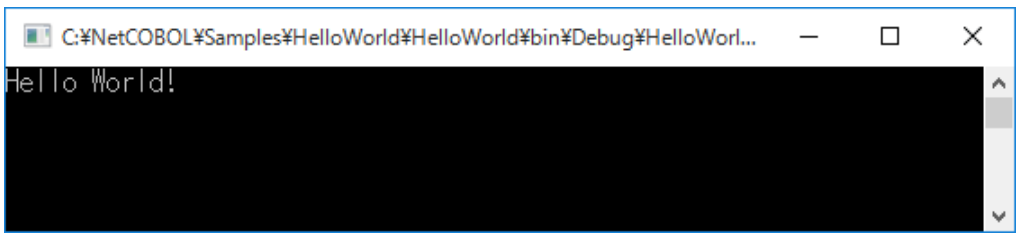


2. デバッグを開始します。 [デバッグ]メニューから[デバッグの開始]を選択すると、コンソールウィンドウを表示した後、 ブレークポイントで止まります。 このとき、DISPLAY 文は実行されていないため、 コンソールウィンドウには何も出力されていません。

3. DISPLAY 文だけを実行します。 [デバッグ]メニューから[ステップイン]を選択します。



コンソールウィンドウに「Hello World!」が出力されます。



4. [デバッグ]メニューから[続行]を選択すると、 アプリケーションの最後まで実行されます。

コマンドラインからの開発

ビルドから実行までの流れを簡単に説明します。 詳細な説明については、関連項目を参照してください。

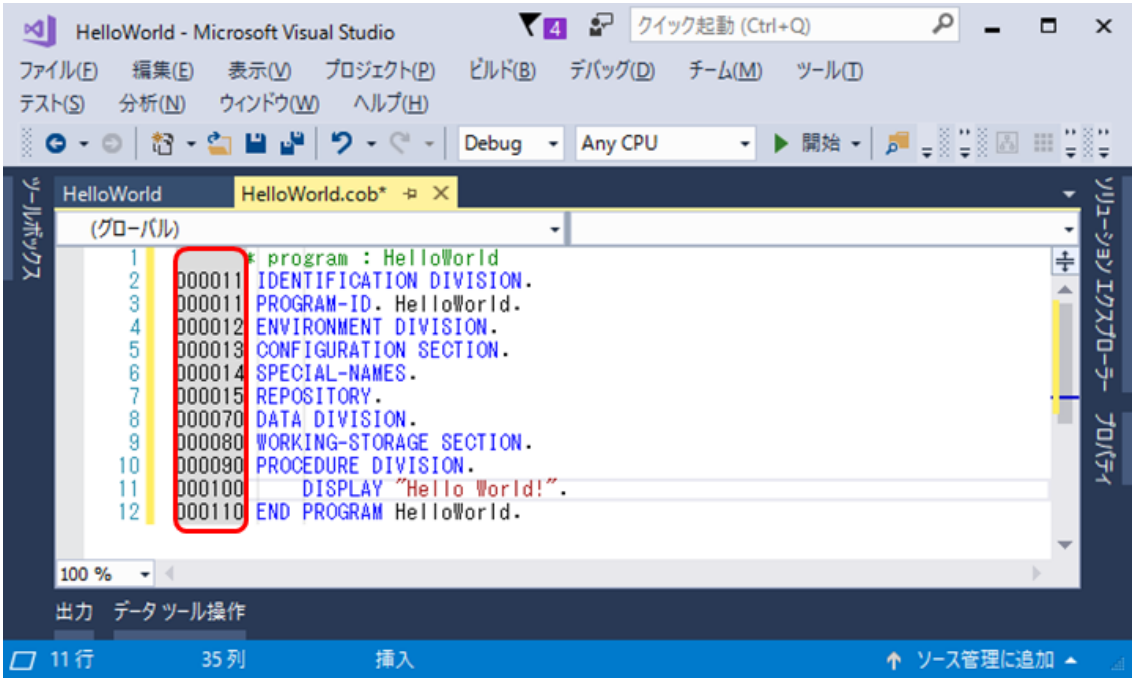
	手順	関連項目
1	COBOL ソースプログラムを作成・編集する	COBOL ソースプログラムの作成・編集
2	ビルドする	コマンドラインからのビルド
3	実行する	アプリケーションの実行

コマンドラインで開発する場合、既存の COBOL ソースプログラムを使う場合がほとんどでしょう。ここでは、以下の内容を持つ「HelloWorld.cob」を使って、 コンソールアプリケーションを作成します。

```
     * program : HelloWorld
000011 IDENTIFICATION DIVISION.
000011 PROGRAM-ID. HelloWorld.
000012 ENVIRONMENT DIVISION.
000013 CONFIGURATION SECTION.
000014 SPECIAL-NAMES.
000015 REPOSITORY.
000070 DATA DIVISION.
000080 WORKING-STORAGE SECTION.
000090 PROCEDURE DIVISION.
000100     DISPLAY "Hello World!".
000110 END PROGRAM HelloWorld.
```

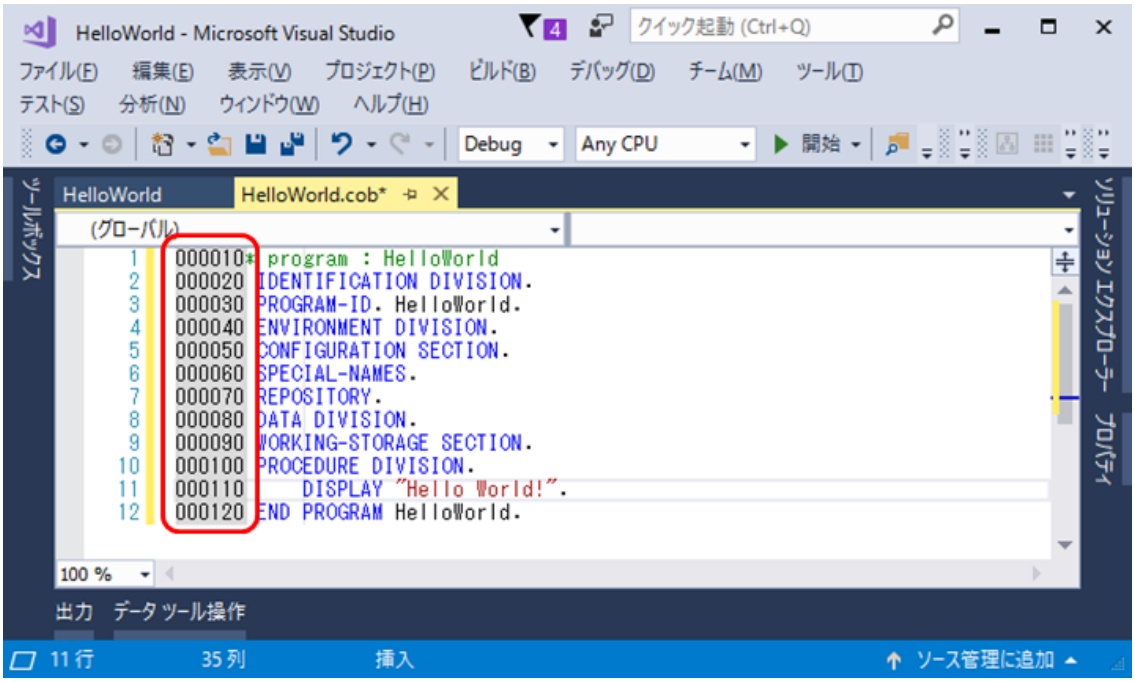
1. COBOL ソースプログラムを作成・編集する

   COBOL ソースプログラムを編集するには、使い慣れたエディターも良いですが、 COBOL の予約語のチェックや一連番号の更新などができる Visual Studio エディターを使うことをお薦めします。 Visual Studio エディターで COBOL ソースプログラムを開くには、 [ファイル]メニューから[開く]-[ファイル]を選択し、[ファイルを開く]ダイアログボックスから編集する COBOL ソースプログラムを選択します。



   ここでは、一連番号をリナンバしてみましょう。 COBOL ソースプログラムを開いたまま、 [編集]メニ

ューから[詳細]-[リナンバ]を選択すると、一連番号がリナンバされます。



2. ビルドする

NMAKE ユーティリティを使う方法もありますが、 ここでは、cobolc コマンドを使ってビルドします。cobolc コマンドは、コマンドプロンプトから実行します。

Windows 7

コマンドプロンプトは、[スタート] - [すべてのプログラム] - お使いの NetCOBOL for .NET 製品名 - [NetCOBOL]-[NetCOBOL for .NET コマンドプロンプト]を選択すると表示されます。

Windows 7 以外

コマンドプロンプトは、[スタート] - アプリ一覧(*) - お使いの NetCOBOL for .NET 製品名 - [NetCOBOL for .NET コマンドプロンプト]を選択すると表示されます。

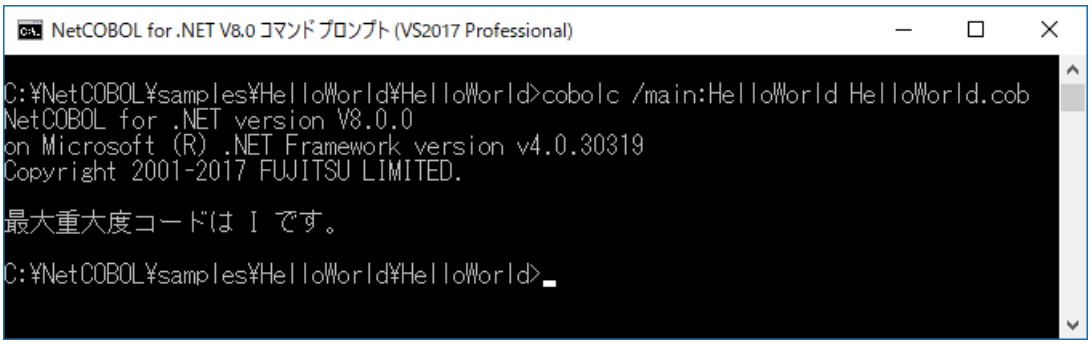
* : Windows 8.1 および Windows Server 2012 R2 の場合、[スタート]画面 - [↓] - [アプリ]を操作をすることで同様の画面になります。

cobolc コマンドは以下の形式で指定します。

```
cobolc [オプションの並び] ファイル名 …
```

ここでは、オプションの並びに、/main:HelloWorld を指定します。 これは、アプリケーションのエントリポイントの指定です。 ファイル名には、HelloWorld.cob を指定します。

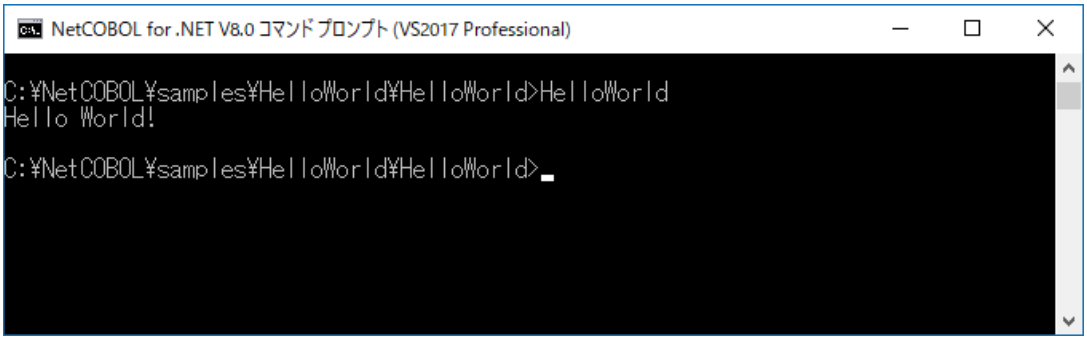
ここでは、カレントフォルダを COBOL ソースプログラムがあるフォルダに移してから、 cobolc コマンドを実行してみましょう。



ビルドが終了すると、HelloWorld.exe ができます。

3. 実行する

コマンドプロンプトから HelloWorld.exe を実行します。



「Hello World!」が表示されます。

NCOBU00039

COBOL が使用するファイル

ここでは、NetCOBOL for .NET が使用する、 COBOL 特有のファイルについて説明します。

それぞれのファイルの使用方法については関連トピックスを参照してください。

翻訳で使用するファイル

ファイルの種類	拡張子	ファイルの内容	関連するトピック
COBOL ソースファイル	.cob、.cbl、.cobol、.cobx	COBOL ソースプログラム	COBOL ソースプログラムの作成・編集
			COBOL ソースプログラムの形式
			プロジェクトへのソースファイルの追加
登録集ファイル(*1)	.cbl(*2)	登録集原文	プロジェクトへの登録集ファイルの追加
			登録集パスの設定
			/copypath (登録集ファイルへのパスの指定)
			/copyname (登録集名および位置の指定)
			/copyext (登録集ファイルの拡張子を指定)
帳票定義体ファイル	.pmd、.pxd、.smd(*3)	帳票定義体	/formext (帳票定義体ファイルの拡張子を指定)
			/formpath (帳票定義体ファイルへのパスを指定)
翻訳リストファイル	.lst	翻訳リスト	/print (翻訳リストファイルの出力を指定)
			SOURCE (ソースプログラムリストの出力の可否)
			COPY (登録集原文の表示)
			XREF (相互参照リストの出力の可否)
			MESSAGE (オプション情報リスト、翻訳単位統計情報リストの出力の可否)
			NUMBER (ソースプログラムの一連番号領域の指定)
			LINECOUNT (翻訳リストの 1 ページあたりの行数)
			LINESIZE (翻訳リストの 1 行あたりの文字数)

*1: 登録集ファイルの検索方法は、COPY 文の書き方によって異なります。 以下に登録集ファイルの検索順序を示します。

●IN/OF 登録集名の指定がない COPY 文を記述した場合

・ Visual Studio IDE を使用したビルドの場合

1. プロジェクトに追加した登録集ファイルを検索します。 [参照] “プロジェクトへの登録集ファイルの追加”
2. プロジェクトの［登録集パス］に設定したフォルダを検索します。 複数のフォルダを指定した場合、指定した順序で検索します。 [参照] “登録集パスの設定”
3. カレントフォルダを検索します。 Visual Studio IDE からビルドする場合、プロジェクトフォルダがカレントフォルダとなります。

[参考] ASP.NET Web サイトをビルドする場合、登録集の利用方法については、登録集を利用するを参照してください。

・ コマンドラインからビルドする場合

1. /copypath コンパイラオプションに指定したフォルダを検索します。 複数のフォルダを指定した場合、指定した順序で検索します。
2. カレントフォルダを検索します。

●IN/OF 登録集名の指定がある COPY 文を記述した場合

/copyname コンパイラオプションによって登録集名に関連付けたフォルダを検索します。 Visual Studio IDE を使用したビルドの場合、/copyname コンパイラオプションは、 プロジェクトの［ビルド］ページの［追加オプション］に指定します。

IN/OF 登録集名の指定がある COPY 文では、/copypath コンパイラオプションに指定されたフォルダ、 プロジェクトの登録集パスに指定されたフォルダ、カレントフォルダは検索対象にはなりません。

*2: 拡張子は、/copyext コンパイラオプションを使って任意の文字列に変更できます。/copyext コンパイラオプションを指定しない場合、以下の順で検索します。

1. 拡張子(cbl)
2. 拡張子(cob)
3. 拡張子(cobol)

*3: それぞれの拡張子は、/formext コンパイラオプションを使って任意の文字列に変更できます。

実行で使用するファイル

ファイルの種類	拡張子	ファイルの内容	関連するトピック
アプリケーション構成ファイル	.config	アプリケーションの実行に必要な実行環境変数、エントリ情報およびSQL 情報(ODBC 情報)を格納	アプリケーション構成ファイル
実行用の初期化ファイル	.cbr(*1)	アプリケーションの実行に必要な実行環境変数を格納	実行用の初期化ファイル
エントリ情報ファイル	任意	副プログラム名とその副プログラムを含む DLL 名またはアセンブリ名の関連付け	エントリ情報ファイル
印刷情報ファイル	任意	出力する帳票の状態制御情報(FORMAT 句なし印刷ファイルを利用して帳票出力を行う場合に使用)	印刷情報ファイル
ODBC 情報ファイル	任意	クライアントとサーバ間のコネクション(CONNECT 文などで指定する)を確立するために必要な情報	ODBC 情報ファイル

*1: 変更可能

プロジェクトの作成と管理

ここでは、Visual Studio を使ったプロジェクトの作成方法と管理方法について説明します。

このセクションの内容

NetCOBOL for .NET プロジェクトと ASP.NET Web サイト

NetCOBOL for .NET プロジェクトと ASP.NET Web サイトの違いについて説明します。

NetCOBOL for .NET プロジェクトの作成と管理

NetCOBOL for .NET プロジェクトの作成方法と管理方法について説明します。

ASP.NET Web サイト作成と管理

ASP.NET Web サイトの作成方法と管理方法について説明します。

プロジェクトのアップグレード

NetCOBOL for .NET V2.1 以前のバージョンのプロジェクトをアップグレードする際に注意すべき事項について説明します。

NCOBU01019

NetCOBOL for .NET プロジェクトと ASP.NET Web サイト

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。

Visual Studio 2017 のプロジェクトのうち、 NetCOBOL for .NET が関係するプロジェクトには以下のものがあります。

- NetCOBOL for .NET プロジェクト
- ASP.NET Web サイト

NetCOBOL for .NET プロジェクトは.NET Framework のアセンブリを作成するためのプロジェクトです。NetCOBOL for .NET プロジェクトのプロジェクト機能は NetCOBOL for .NET が提供しています。

NetCOBOL for .NET プロジェクトは、 Windows クラシックデスクトップ(従来のデスクトップ)向けのアプリケーションを開発するプロジェクトです。

一方、ASP.NET Web サイトは ASP.NET Web アプリケーションを作成するためのプロジェクトです。ASP.NET Web サイトのプロジェクト機能は Visual Web Developer が提供しており、 NetCOBOL for .NET が COBOL 言語のサポートを Visual Web Developer に追加しています。

Visual Studio 2003 以前では、 各開発言語のプロジェクトが ASP.NET Web アプリケーション開発機能を提供していました。 しかし、Visual Studio 2005 以降、 ASP.NET Web アプリケーション開発機能は Visual Web Developer がまとめて提供するように仕様変更されました。

このため、NetCOBOL for .NET V2.1 以前の Web プロジェクトは、 ASP.NET Web サイトに置き換えられました。

NetCOBOL for .NET プロジェクトの作成と管理

ここでは、Visual Studio を使った NetCOBOL for .NET プロジェクトの作成方法と管理方法について説明します。

このセクションの内容

NCOBU00041

COBOL プロジェクト

COBOL プロジェクトの種類

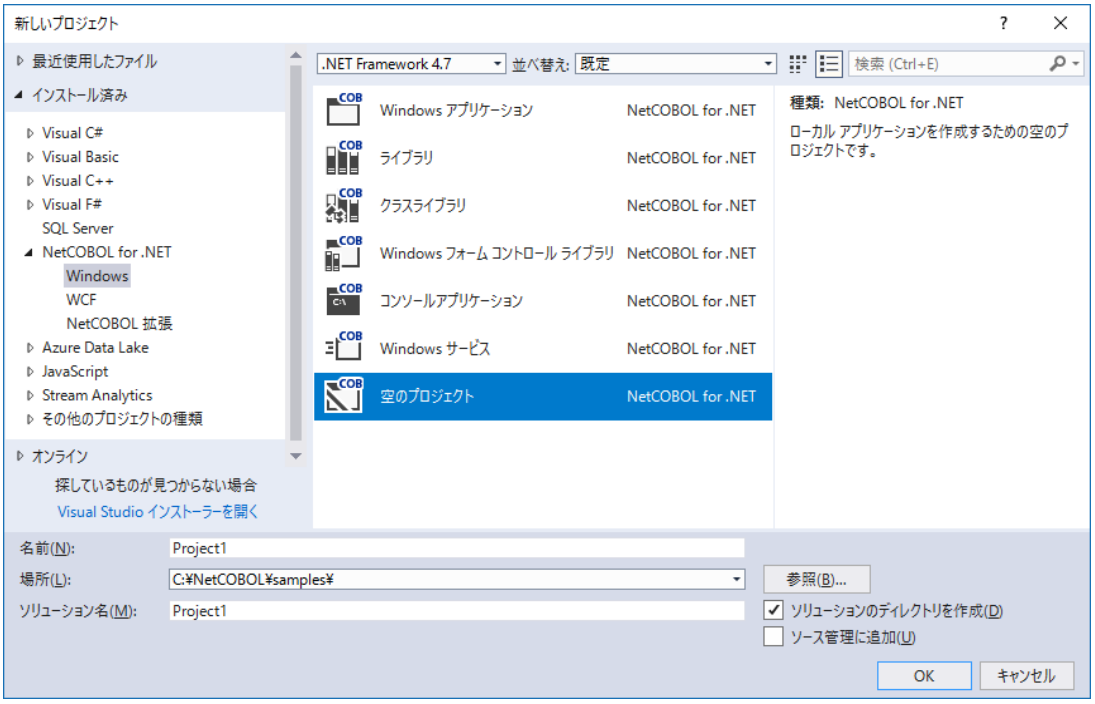
Visual Studio で作成できる COBOL プロジェクトには、 以下の種類があります。 これらのプロジェクトは、プロジェクト作成時に、 目的に応じたテンプレートを選択することで作成できます。

プロジェクトの種類	用途
空のプロジェクト	ローカル アプリケーションを作成するための空のプロジェクトです。
ライブラリ	他のアプリケーションで使用するプログラムを作成するためのプロジェクトです。
クラスライブラリ	他のアプリケーションで使用するクラスを作成するためのプロジェクトです。
コンソールアプリケーション	コマンドライン アプリケーションを作成するためのプロジェクトです。
Windows アプリケーション	Windows ユーザーインタフェースを含むアプリケーションを作成するためのプロジェクトです。
Windows フォーム コントロール ライブラリ	Windows アプリケーションで使用するコントロールを作成するためのプロジェクトです。
Windows サービス	Windows サービスを作成するためのプロジェクトです。
WCF サービス ライブラリ	WCF サービスを作成するためのプロジェクトです。

プロジェクトの作成方法

Visual Studio で COBOL プロジェクトを作成するには、 以下の手順で行います。

1. [ファイル]メニューから[新規作成]-[プロジェクト]を選択します。
2. [新しいプロジェクト]ダイアログボックスが表示されます。



3. 左ペインで、[NetCOBOL for .NET]を選択します。
4. 対象のフレームワークを選択します。ここでは、 [.NET Framework 4.7] を選択しています。
5. 右ペインで、用途に合わせたプロジェクトを選択します。 ここでは、[空のプロジェクト]を選択しています。
6. [名前]に、新規プロジェクトの名前を入力します。
7. [場所]で、新規プロジェクトの格納場所を選択します。
8. [OK]ボタンをクリックします。

新規プロジェクトが作成され、ソリューションエクスプローラーの画面に反映されます。

ソリューションエクスプローラーが表示されない場合、 [表示]メニューから[ソリューションエクスプローラー]を選択すれば表示されます。

NCOBU00042

プロジェクトへのソースファイルの追加

ここでは、プロジェクトにソースファイルとして追加するファイルの種類と、 追加する手順について説明します。

このセクションの内容

ソースファイルの種類

ソースファイルとしてプロジェクトに追加するファイルには、以下のファイルがあります。

COBOL ソースファイル(*.cob、*.cbl、*.cobol、*.cobx)	プログラム
	クラス
	インタフェース
	デリゲート
	ENUM
	プログラム原型
	登録集
リソースファイル(*.resx)	アセンブリリソースファイル
COBOL 固有ファイル(注 1)	帳票定義体ファイル(*.pmd、*.pxd、*.smd)
	実行用の初期化ファイル
	エントリ情報ファイル
	印刷情報ファイル
	ODBC 情報ファイル
マニフェスト ファイル(.manifest)	アプリケーション マニフェスト ファイル

注 1

COBOL 固有ファイルは、COBOL が使用するファイルに示したファイルのうち、COBOL ソースファイル、登録集ファイルおよびアプリケーション構成ファイルを除いたファイルすべてを指しています。

上記に示す以外にも COBOL アプリケーションを構成する COBOL 固有ファイルがある場合は、 ソースファイルとして追加してください。

NCOBU00044

新規ソースファイルの追加

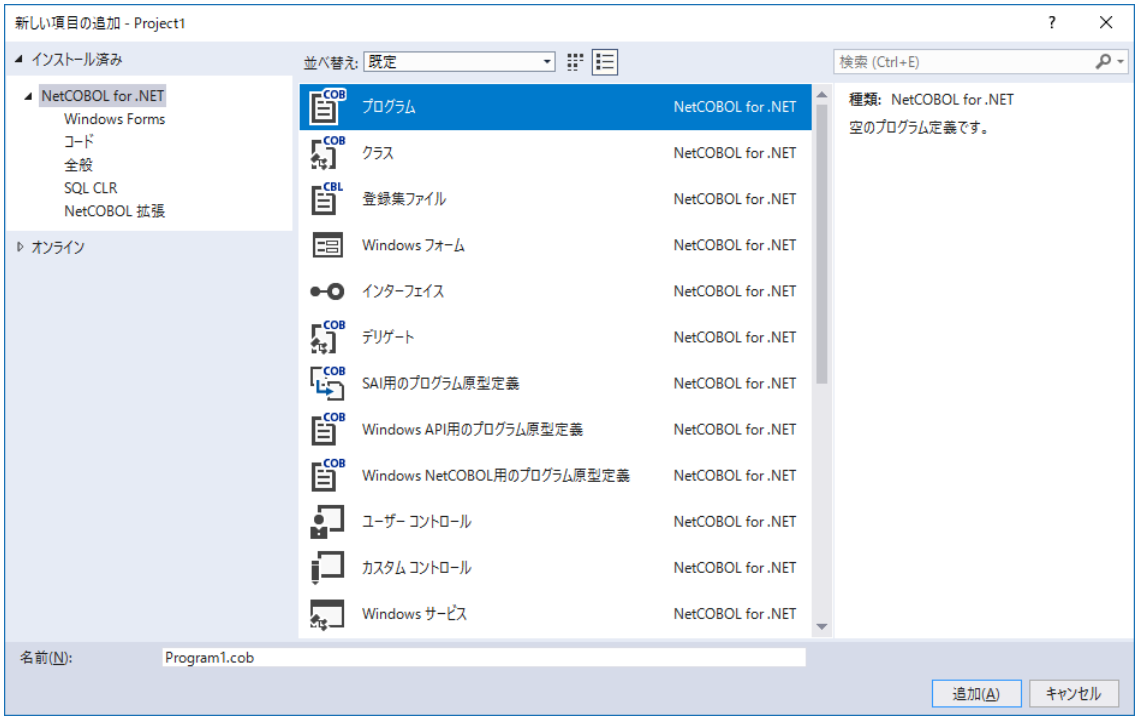
新規にソースファイルを作成するには、3 つの方法があります。

- ソリューションエクスプローラーからプロジェクト名を選択して作成する方法
- ソリューションエクスプローラーから[ソースファイル]フォルダを選択して作成する方法
- [プロジェクト]メニューから作成する方法

いずれの方法も大きな違いはありません。それぞれの方法について説明します。

ソリューションエクスプローラーからプロジェクト名を選択して追加する方法

1. ソリューションエクスプローラーからプロジェクト名を選択し、右クリックします。
2. コンテキストメニューから[追加]-[新しい項目]を選択します。
3. [新しい項目の追加]ダイアログボックスが表示されます。



4. 作成するソースファイルのテンプレートを選択し、ファイル名を指定します。
5. [追加]ボタンをクリックすると、ソースファイルの編集画面が表示されます。

ソリューションエクスプローラーから[ソースファイル]フォルダを選択して追加する方法

1. ソリューションエクスプローラーから「ソースファイル」フォルダを選択し、右クリックします。

上記の 2.～5.と同じ手順を行います。

[プロジェクト]メニューから追加する方法

1. [プロジェクト]メニューから[新しい項目の追加]を選択します。

上記の 3.～5.と同じ手順を行います。

既存ソースファイルの追加

既存のソースファイルをプロジェクトに追加するには、3 つの方法があります。

- ソリューションエクスプローラーからプロジェクト名を選択して追加する方法
- ソリューションエクスプローラーから[ソースファイル]フォルダを選択して追加する方法
- [プロジェクト]メニューから追加する方法

いずれの方法も大きな違いはありません。それぞれの方法について説明します。

ソリューションエクスプローラーからプロジェクト名を選択して追加する方法

1. ソリューションエクスプローラーからプロジェクト名を選択し、右クリックします。
2. コンテキストメニューから[追加]-[既存の項目]を選択します。
3. [既存項目の追加]ダイアログボックスが表示されます。
4. 追加したいファイルを指定し、[追加]ボタンをクリックします。

ソリューションエクスプローラーから[ソースファイル]フォルダを選択して追加する方法

1. ソリューションエクスプローラーから[ソースファイル]フォルダを選択し、右クリックします。

上記の 2.～4.と同じ手順を行います。

[プロジェクト]メニューから追加する方法

1. [プロジェクト]メニューから[既存項目の追加]を選択します。

上記の 3.～4.と同じ手順を行います。

NCOBU00046

プロジェクトへの登録集ファイルの追加

このセクションの内容

ここでは、プロジェクトに登録集ファイルを追加する手順について説明します。

新規登録集ファイルの追加

新規に登録集ファイルを作成し、プロジェクトに追加する方法について説明します。

既存登録集ファイルの追加

既存の登録集ファイルを追加する手順について説明します。

登録集パスの設定

登録集パスの設定方法について説明します。

新規登録集ファイルの追加および既存登録集ファイルの追加でプロジェクトに追加した登録集ファイルは、 ビルド時に自動的に取り込まれます。 プロジェクトに追加しない登録集ファイルは、自動で取り込まれないため、登録集パスの設定が必要です。

登録集ファイルを[ソースファイル]フォルダに追加すると、 その登録集ファイルが COBOL ソースプログラムとして翻訳されるため、 構文エラーが発生する原因となります。 この場合、登録集ファイルとして翻訳されるように、上記の手順で追加しなおしてください。

新規登録集ファイルの追加

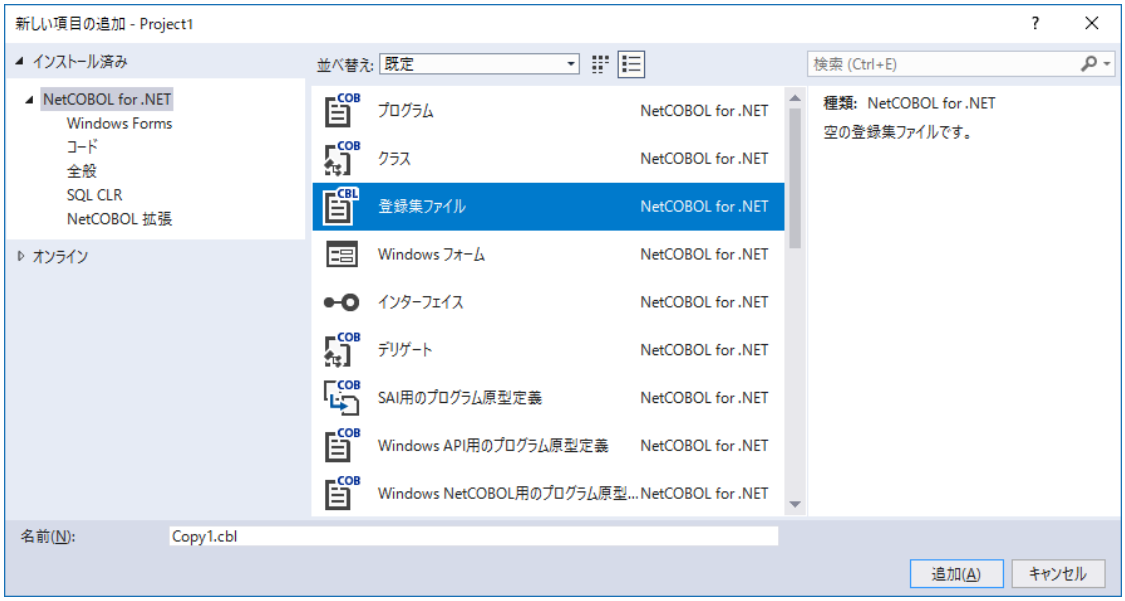
新規に登録集ファイルを作成するには、3 つの方法があります。

- ソリューションエクスプローラーから[登録集ファイル]フォルダを選択して作成する方法
- ソリューションエクスプローラーから[ソースファイル]フォルダを選択して作成する方法
- ソリューションエクスプローラーからプロジェクト名を選択して作成する方法

それぞれの方法について説明します。

ソリューションエクスプローラーから[登録集ファイル]フォルダを選択して追加する方法

1. ソリューションエクスプローラーから[登録集ファイル]フォルダを選択し、右クリックします。
2. コンテキストメニューから[追加]-[新しい項目]を選択します。
3. [新しい項目の追加]ダイアログボックスが表示されます。



4. テンプレートから[登録集ファイル]を選択し、登録集ファイル名を指定します。
5. [追加]ボタンをクリックすると、登録集ファイルの編集画面が表示されます。

ソリューションエクスプローラーから[ソースファイル]フォルダを選択して追加する方法

1. ソリューションエクスプローラーから[ソースファイル]フォルダを選択し、右クリックします。

上記の 2.～5.と同じ手順を行います。

ソリューションエクスプローラーからプロジェクト名を選択して追加する方法

1. ソリューションエクスプローラーからプロジェクト名を選択し、右クリックします。

上記の 2.～5.と同じ手順を行います。

NCOBU00048

既存登録集ファイルの追加

既存の登録集ファイルをプロジェクトに追加するには、3 つの方法があります。

- ソリューションエクスプローラーから[登録集ファイル]フォルダを選択して追加する方法
- ソリューションエクスプローラーから[ソースファイル]フォルダを選択して追加する方法
- ソリューションエクスプローラーからプロジェクト名を選択して追加する方法

それぞれの方法について説明します。

ソリューションエクスプローラーから[登録集ファイル]フォルダを選択して追加する方法

1. ソリューションエクスプローラーから[登録集ファイル]フォルダを選択し、右クリックします。
2. コンテキストメニューから[追加]-[既存の項目]を選択します。
3. [既存項目の追加]ダイアログボックスが表示されます。
4. 追加したいファイルを指定し、[追加]ボタンをクリックします。

ソリューションエクスプローラーから[ソースファイル]フォルダを選択して追加する方法

1. ソリューションエクスプローラーから[ソースファイル]フォルダを選択し、右クリックします。
2. コンテキストメニューから[追加]-[既存登録集の追加]を選択します。

上記の 3.～4.と同じ手順を行います。

ソリューションエクスプローラーからプロジェクト名を選択して追加する方法

1. ソリューションエクスプローラーからプロジェクト名を選択し、右クリックします。
2. コンテキストメニューから[追加]-[既存登録集の追加]を選択します。

上記の 3.～4.と同じ手順を行います。

登録集パスの設定

登録集ファイルの検索パスをプロジェクトに設定する方法について説明します。

登録集パスを設定すると、プロジェクトに追加していない登録集ファイルも検索できるようになります。

1. ソリューションエクスプローラーからプロジェクト名を選択し、右クリックします。
2. コンテキストメニューから[プロパティ]を選択します。
3. プロパティページが表示されます。
4. 左ペインの [登録集パス]を選択します。
5. [フォルダ]テキストボックスに検索対象にするフォルダを入力し、登録集パス(ユーザー)または登録集パス(プロジェクト)の [フォルダの追加]ボタンをクリックします。
6. [登録集パス]に入力したフォルダが追加されます。

「登録集パス(ユーザー)」に追加された登録集パスは、プロジェクトユーザオプションファイル(*.cobproj.user)に保存されます。 「登録集パス(プロジェクト)」に追加された登録集パスは、プロジェクトファイル(*.cobproj)に保存されます。

NCOBU00050

プログラム原型定義の追加

ここでは、Windows 版 NetCOBOL で作成されたプログラムまたは Windows API や C 言語の関数を呼び出すためのプログラム原型定義を自動生成し、プロジェクトに追加する方法について説明します。

プログラム原型定義には、以下の３つのテンプレートを用意しています。

- Windows NetCOBOL 用のプログラム原型定義
- SAI 用のプログラム原型定義
- Windows API 用のプログラム原型定義

ここでは、Windows 版 NetCOBOL で出力した SAI ファイルの情報を利用して、 Windows 版 NetCOBOL のためのプログラム原型定義を生成する場合について説明します。 SAI ファイルについては、Windows 版 NetCOBOL ユーザーズガイドを参照してください。プログラム原型定義をプロジェクトに追加するには、以下の手順で行います。

1. ソリューション エクスプローラーで、プロジェクト名を右クリックし、[追加]-[新しい項目] をクリックします。

   [新しい項目の追加] ダイアログ ボックスが表示されます。

2. [新しい項目の追加] ダイアログ ボックスで、[SAI 用のプログラム原型定義]を選択します。
3. [名前]フィールドにファイル名を入力し、[追加] ボタンをクリックすると、プログラム原型定義 ウィザードが表示されます。

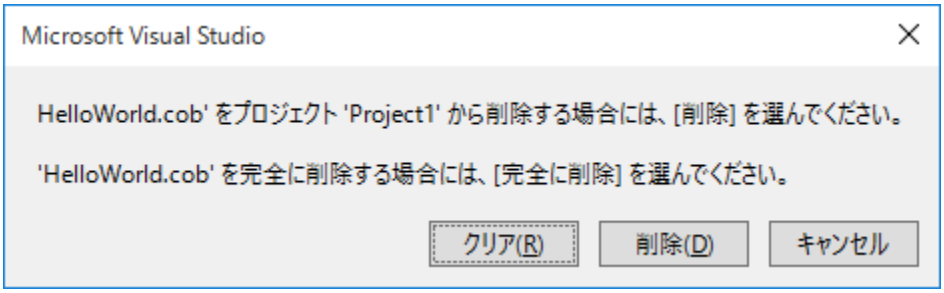
プログラム原型定義 ウィザードの指示にしたがって処理を進めることで、 プログラム原型定義が生成され、プロジェクトに追加されます。

プログラム原型定義については、COBOL からアンマネージコードの呼出しを参照してください。

プロジェクトからファイルを削除

プロジェクトからソースファイルおよび登録集ファイルを削除するには、以下の手順で行います。

1. ソリューションエクスプローラーから削除するファイル名を選択し、右クリックします。
2. コンテキストメニューから[削除]を選択すると、メッセージボックスが表示されます。



3. 目的に合わせて、以下のボタンをクリックします。

[クリア]ボタン	プロジェクトからファイルを登録解除します。
[削除]ボタン	プロジェクトからファイルを登録解除し、ファイルシステムからそのファイルを削除します。

プロジェクトがソース管理されている場合、[削除]コマンドを使用するとソース管理上のファイルは"削除"状態となり、チェックイン時にソース管理から削除されます。 プロジェクトからファイルを取り除くだけの場合には、以下の手順で行います。

1. ソリューションエクスプローラーから除外するファイル名を選択し、右クリックします。
2. コンテキストメニューから[プロジェクトから除外]を選択します。

NCOBU00813

参照パスの設定

.NET Framework コンポーネントの参照の解決に使用する参照パスを設定する方法について説明します。

1. ソリューションエクスプローラーからプロジェクト名を選択し、右クリックします。
2. コンテキストメニューから[プロパティ]を選択します。
3. プロパティページが表示されます。
4. 左ペインの[参照パス]を選択します。
5. [フォルダ]テキストボックスに検索対象にするフォルダを入力し、参照パス(ユーザー)または参照パス(プロジェクト)の [フォルダの追加]ボタンをクリックします。
6. [参照パス]に入力したフォルダが追加されます。

「参照パス(ユーザー)」に追加された参照パスは、プロジェクトユーザオプションファイル(*.cobproj.user)に保存されます。 「参照パス(プロジェクト)」に追加された参照パスは、プロジェクトファイル(*.cobproj)に保存されます。

参照の解決

プロジェクトでコンポーネントを使用する場合、プロジェクトにコンポーネントへの参照を追加します。コンポーネントには以下の種類があります。

- .NET Framework コンポーネント
- COM コンポーネント
- ローカルプロジェクトで作成された再利用可能なコンポーネント

参照の追加と削除の方法は、Visual Studio のドキュメントの方法 ：Visual Studio で参照を追加または削除するを参照してください。

NetCOBOL for .NET では、プロジェクトに.NET Framework コンポーネント を追加するとき、ファイルが次の場所に存在する場合は、アセンブリの参照情報としてアセンブリ名だけを プロジェクトファイルに保存します。

- プロジェクトフォルダ
- プロジェクトの「参照パス(ユーザー)」プロパティに指定されたパス(*1)
- プロジェクトの「参照パス(プロジェクト)」プロパティに指定されたパス(*1)
- .NET Framework パス
- Visual Studio にアセンブリフォルダとして登録されたパス(*2)
- グローバルアセンブリキャッシュ
- アセンブリの出力先

ファイルが上記の場所以外に存在した場合は、アセンブリ名、および プロジェクトフォルダからの相対パス（異なるドライブ上にある場合は絶対パス） を HintPath としてプロジェクトファイルに保存します。

プロジェクトに保存された.NET Framework コンポーネントの参照を解決 する場合は、次の順番でファイルの存在を確認し、最初に見つかったファイルを使用します。

1. プロジェクトフォルダ
2. プロジェクトの「参照パス(ユーザー)」プロパティに指定されたパス(*1)
3. プロジェクトの「参照パス(プロジェクト)」プロパティに指定されたパス(*1)
4. HintPath に設定されているファイルのパス
5. .NET Framework パス
6. Visual Studio にアセンブリフォルダとして登録されたパス(*2)
7. グローバルアセンブリキャッシュ
8. アセンブリの出力先

なお、参照のプロパティで「特定バージョン」が True の場合は、アセンブリのバージョンなども 含めて一致しているか確認します。

***1:** 参照パスの設定方法については、参照パスの設定を参照してください。

***2:** 次のレジストリのキーに登録されているフォルダです。

```
HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥.NETFramework¥<version>¥AssemblyFoldersEx
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥.NETFramework¥<version>¥AssemblyFoldersEx
```

<version>は、.NET Framework のバージョンを表します。

NCOBU00053

プロジェクトオプションの設定

プロジェクトオプションとして、以下を設定することができます。

- ビルド時に使用されるコンパイラオプション
- デバッガー起動時に使用されるデバッグオプション

プロジェクトオプションは、プロジェクトのプロパティページで設定します。

プロパティページは、ソリューションエクスプローラーのプロジェクト名を右クリックし、コンテキストメニューから[プロパティ]を選択することで表示されます。ソリューションエクスプローラーが表示されない場合は、[表示]メニューを選択し、[ソリューションエクスプローラー]を選択することで表示されます。

[プロパティページ] の左ペインには、[アプリケーション] や[ビルド]などのタブがあります。 表示されるタブはアプリケーションの種類によって異なります。

以下のタブはすべてのアプリケーションの種類で表示されます。

- [アプリケーション]
- [ビルド]
- [ビルドイベント]
- [デバッグ]
- [参照パス]
- [登録集パス]
- [署名]

コンソール アプリケーション、および Widnows アプリケーションの場合には以下のタブが追加表示されます。

- [リソース]
- [設定]
- [セキュリティ]
- [発行]

ライブラリの場合には以下のタブが追加表示されます。

- [リソース]
- [設定]

以下のプロパティページでは Debug/Release といった構成に関係なくすべての構成で共通となるオプションを設定することができます。

- [アプリケーション]
- [ビルドイベント]
- [参照パス]
- [登録集パス]
- [署名]
- [リソース]
- [設定]

- [セキュリティ]
- [発行]

以下のプロパティページでは、同一のオプションに対して構成ごとに異なる値を設定することができます。 これらのプロパティページのオプションを設定する場合には、設定する構成を[構成] リストおよび[プラットフォーム] リストの中から選択します。

- [ビルド]
- [デバッグ]

- コンパイラオプションの設定方法については、翻訳オプションの設定を参照してください。
- デバッグオプションの設定方法については、デバッグオプションの設定を参照してください。

NCOBU00054

プロジェクトの実行環境設定

プロジェクトの実行環境を設定するには、以下の 2 つの方法があります。

- アプリケーション構成ファイルによる実行環境設定
- 実行用の初期化ファイルによる実行環境設定

NetCOBOL for .NET では、 アプリケーション構成ファイルによる実行環境設定を推奨しています。

アプリケーション構成ファイルによる実行環境設定

アプリケーション構成ファイルに実行環境を設定するには、実行環境設定ユーティリティを使用します。 プロジェクトから実行環境設定ユーティリティを起動するには、以下の手順で行います。

1. ソリューション エクスプローラーのプロジェクト名を右クリックし、 コンテキストメニューから[COBOL 実行環境設定の編集]を選択します。 アプリケーション構成ファイル(App.config ファイル)がソリューション エクスプローラーのプロジェクトに存在しない場合は、 自動的にアプリケーション構成ファイルがプロジェクトに追加されます。 アプリケーション構成ファイルがソリューション エクスプローラーのプロジェクトに存在する場合は、 アプリケーション構成ファイルを右クリックし、 コンテキストメニューから[COBOL 実行環境設定の編集]を選択することもできます。
2. 実行環境設定ユーティリティが起動されます。
3. 実行環境の設定を行い、保存操作を行います。 これにより、設定内容がアプリケーション構成ファイルに反映されます。
4. 実行環境設定ユーティリティを終了し、Visual Studio に戻ります。

アプリケーション構成ファイルの内容については、アプリケーション構成ファイルを参照してください。

実行用の初期化ファイルによる実行環境設定

実行用の初期化ファイルに実行環境を設定するには、実行環境設定ツールを使用します。 プロジェクトから実行環境設定ツールを起動するには、以下の手順で行います。

1. 実行用の初期化ファイル(COBOL85.CBR)がプロジェクトに存在しない場合は、 ソリューション エクスプローラーのプロジェクト名を右クリックし、 コンテキストメニューから[追加]-[新しい項目]を選択します。 これにより、 [新しい項目の追加]ダイアログが表示されます。 [新しい項目の追加]ダイアログで[COBOL 実行用の初期化ファイル]を選択し、 [OK]ボタンをクリックします。 これにより、COBOL85.CBR が追加されます。
2. ソリューション エクスプローラーの COBOL85.CBR を右クリックし、 コンテキストメニューから[COBOL 実行用の初期化ファイル(CBR)の編集]を選択します。
3. 実行環境設定ツールが起動されます。
4. 実行環境設定を行い、保存操作を行います。 これにより、設定内容が実行用の初期化ファイルに反映されます。
5. 実行環境設定ツールを終了し、Visual Studio に戻ります。

実行用の初期化ファイルの内容については、実行用の初期化ファイルを参照してください。

- NetCOBOL for .NET では、 実行用の初期化ファイルよりアプリケーション構成ファイルを優先します。したがって、アプリケーション構成ファイルによって実行環境を設定する場合で、かつ、プロジェクトに実行用の初期化ファイルが存在するとき、 実行環境設定ユーティリティは、実行用の初期化ファイルの内容をアプリケーション構成ファイルに取り込みます。 アプリケーション構成ファイルに取り込んだ実行用の初期化ファイルはプロジェクトから削除するか、実行用の初期化ファイルの[出力ディレクトリへのコピー]プロパティを[コピーしない]に設定することをお勧めします。

- ソリューション エクスプローラーのプロジェクトに追加されたアプリケーション構成ファイル(App.config)は、 プロジェクトのビルドを行うと、**アセンブリ名.拡張子**.config という形式のファイル名で出力パスにコピーされます。 出力パスのアプリケーション構成ファイルを直接編集していた場合は、ビルド操作によってファイルが上書きされますので注意してください。

- ソリューション エクスプローラーのプロジェクトに追加された、ファイルの[出力ディレクトリへのコピー]プロパティが [常にコピーする]または[最新の場合にコピーする]の場合、 プロジェクトのビルドを行うと、 ファイルが出力パスにコピーされます。 [出力ディレクトリへのコピー]プロパティが[コピーしない]の場合、 プロジェクトのビルドを行うと、 出力パスにファイルが存在する場合は、ファイルが削除されます。 出力パスに、ソリューションエクスプローラーに追加されているファイルと同名のファイルが存在する場合は、 ビルド操作によってファイルが上書きまたは削除されますので注意してください。

NCOBU01198

NuGet パッケージマネージャーの利用

NetCOBOL for .NET では、 Visual Studio の拡張機能であるライブラリのパッケージ管理システム「NuGet パッケージマネージャー」を利用することができます。

ソリューション エクスプローラーのプロジェクト名を右クリックし、コンテキストメニューから[NuGet パッケージの管理]を選択することで、 NuGet パッケージマネージャーを利用することができます。

NuGet パッケージマネージャーを利用することによって、次の操作ができるようになります。

- パッケージのインストール NuGet パッケージソース(NuGet Gallery など)から、 パッケージをダウンロードしプロジェクトにライブラリやファイルを追加することができます。
- パッケージのアップデート プロジェクトに追加されたパッケージより新しいバージョンのパッケージが NuGet パッケージソースに存在する場合は、 パッケージを更新することができます。
- パッケージのアンインストール パッケージのプロジェクトに追加されたパッケージを削除することができます。

NuGet パッケージマネージャーの詳細については NuGet のドキュメントを参照してください。

ASP.NET Web サイト作成と管理

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。

ここでは、NetCOBOL for .NET による Visual Studio への COBOL 言語サポートを利用する上で固有な事項を説明します。

このセクションの内容

ASP.NET Web サイトを新規作成する

ASP.NET Web サイトを新規作成する方法について説明します。

ファイルの種類

ASP.NET Web サイトで使用するファイルの種類について説明します。

登録集を利用する

ASP.NET Web サイトで登録集を利用する方法について説明します。

コンパイラオプションを指定する

ASP.NET Web サイトでコンパイラオプションを指定する方法について説明します。

実行環境を設定する

ASP.NET Web サイトで実行環境を設定する方法について説明します。

NCOBU01021

ASP.NET Web サイトを新規作成する

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ(http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

ASP.NET Web サイトのテンプレート

NetCOBOL for .NET は ASP.NET Web サイトを新規作成するためのテンプレートとして 以下を用意しています。

テンプレート	用途
空の Web サイト	空の ASP.NET Web サイトを作成するためのテンプレートです。
ASP.NET Web フォームサイト	Web Form を含む ASP.NET Web サイトを作成するためのテンプレートです。
ASP.NET Web サービス	XML Web サービスを含む ASP.NET Web サイトを作成するためのテンプレートです。
WCF サービス	IIS にホストする WCF サービスを含む ASP.NET Web サイトを作成するためのテンプレートです。

ASP.NET Web サイトの作成方法

Visual Studio で ASP.NET Web サイトを作成するには、 以下の手順で行います。

1. [ファイル]メニューから[新規作成]-[Web サイト]を選択します。
2. [新しい Web サイト]ダイアログボックスが表示されます。
3. 左ペインで、[NetCOBOL for .NET]を選択します。
4. 対象のフレームワークを選択します。ここでは、[.NET Framework 4.7]を選択します。
5. 右ペインで、[ASP.NET Web フォーム サイト]を選択します。
6. [Web の場所]で、ASP.NET Web サイトの格納場所を選択します。
7. [OK]ボタンをクリックします。

新規 ASP.NET Web サイトが作成され、ソリューションエクスプローラーの画面に反映されます。

ソリューションエクスプローラーが表示されない場合、 [表示]メニューから[ソリューションエクスプローラー]を選択すれば表示されます。

ファイルの種類

ASP.NET Web サイトに追加するファイルには、以下のファイルがあります。

COBOL ソースファイル (*.cob、*.cbl、*.cobol、*.cobx)	プログラム
	クラス
	インタフェース
	デリゲート
	ENUM
	プログラム原型
	登録集
ASP.NET 関連ファイルファイル (*.aspx、*.asmx、*.asax、*.ascx、*.master)	Web フォーム
	Web サービス
	Web ユーザー コントロール
	マスターページ
サービスファイル(*.svc)	Windows Communication Foundation サービス
リソースファイル(*.resx)	アセンブリリソースファイル
Web 構成ファイル（Web.config）	ASP.NET Web サイトのアプリケーション構成ファイル
Web 関連ファイルファイル (*.html、*.htm、*.css、*.txt、*.xml、*.xsl、*.js、*.vbs)	HTML ファイル
	スタイルシートファイル
	テキストファイル
	XML ファイル
	スクリプトファイル

NCOBU01023

登録集を利用する

ASP.NET Web サイトでは、 以下の理由により、 登録集を App_Code サブフォルダに格納することを推奨します。

- NetCOBOL for .NET は、ASP.NET Web サイトの翻訳時には、App_Code サブフォルダを登録集パスとして自動的に設定します。
- App_Code サブフォルダの内容が変更された場合、ASP.NET Web サイトは変更されたとみなされ、ビルド時に再翻訳されます。App_Code サブフォルダ以外に登録集を格納し、その登録集を変更した場合、依存関係が認識されず、ビルド時に再翻訳が必要なソースが再翻訳されない場合があります。
- App_Code サブフォルダの内容は、発行時に対象 Web サイトへコピーされません。 翻訳結果のアセンブリのみが発行先へコピーされます。

ASP.NET Web サイトに登録集をテンプレートから新規追加すると、 App_Code サブフォルダに追加されます。 また、作成された登録集ファイルのコード系は ACP となります。SCS 翻訳オプションを変更した場合には、登録集ファイルのコード系を適切に変更してください。

ASP.NET Web サイトでは、ファイルが翻訳対象かどうかは拡張子によって判断されます。 NetCOBOL for .NET では、 以下の拡張子を翻訳対象として登録しています。

- .cob
- .cobx

このため、 登録集のファイル名にはこれら以外の拡張子（.cbl など）をつけてください。 登録集のファイル名の拡張子が上記の拡張子と一致した場合、 その登録集は一個の翻訳対象として翻訳されます。

/copypath または/copyname コンパイラオプションを指定することによって、 登録集のパスを指定することもできます。 ASP.NET Web サイトでは、 これらのオプション中で、 「~¥」によって Web サイトのパスを表すことができます。 以下は、Web サイトの App_Code¥Copybooks サブフォルダを登録集パスとして指定する例です。

```
/copypath:~¥App_Code¥Copybooks
```

ASP.NET Web サイトでコンパイラオプションを設定する方法については、 「コンパイラオプションの設定」を参照してください。

コンパイラオプションを指定する

ASP.NET Web サイトをビルドするためのコンパイラオプションを設定する方法は、コンパイラオプションの設定を参照してください。

NCOBU01025

実行環境を設定する

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ(http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

ASP.NET Web サイトを実行する際の実行環境は、 以下の方法によって設定します。

- アプリケーション構成ファイルによる実行環境設定

詳細は、プロジェクトの実行環境設定を参照してください。

ただし、以下の注意点があります。

- ASP.NET Web サイトでは、 アプリケーション構成ファイルは"Web.config"という固定の名前のファイルになります。 "Web.config"は Web 構成ファイルとも呼ばれます。
- ASP.NET Web サイトでは、[プロジェクト]メニューやプロジェクトや Web 構成ファイル(Web.config)のコンテキストメニューには[COBOL 実行環境設定の編集]は表示されません。 Web 構成ファイルに実行環境を設定する場合は、以下の手順で行います。

  1. [Web サイト]メニューから[新しい項目の追加]を選択して、[新しい項目の追加]ダイアログボックスを開きます。
  2. [Web 構成 ファイル]を選択し、[OK]ボタンをクリックします。
  3. Visual Studio では、[ツール]メニューから[NetCOBOL for .NET ユーティリティ]-[COBOL 実行環境設定ユーティリティ]を選択します。
  4. 実行環境設定ユーティリティの[ファイル]メニューから[開く]を選択し、追加した Web 構成ファイル(Web.config)を開きます。
  5. 実行環境の設定を行い、保存操作を行います。 これにより、設定内容が Web 構成ファイルに反映されます。
  6. 実行環境設定ユーティリティを終了し、Visual Studio に戻ります。

ASP.NET Web サイトでは、 ASP.NET が DLL 形式のプログラムを実行するアプリケーション形態となるため、 実行用の初期化ファイル（COBOL85.CBR など）をコマンドラインから直接指定する、 などの方法による実行環境設定を行うことはできません。

プロジェクトのアップグレード

以前のバージョンのプロジェクトを開こうとすると、Visual Studio が自動的にプロジェクトファイルをアップグレードします。 その際に注意すべき事項を説明します。

このセクションの内容

NCOBU00922

プロジェクトをアップグレードする際の全般的な注意事項

ここでは、以前のバージョンの NetCOBOL for .NET プロジェクトをアップグレードする際の全般的な注意事項について説明します。

V1.x の NetCOBOL for .NET プロジェクトのアップグレードはサポートされていない

アップグレード対象となる NetCOBOL for .NET プロジェクトは、 V2.0 以降のプロジェクトです。

プロジェクト資産は書き込み可能な状態でなければならない

アップグレード対象のプロジェクト資産は書き込み可能な状態でなければなりません。 特に、プロジェクトがソース管理下にある場合、 自動的にチェックアウトできないことがあります。 また、V2.1 以前のプロジェクトで ASP.NET 関連項目を含んでいる場合には、その項目は自動的にチェックアウトされません。 あらかじめプロジェクト資産をチェックアウトするか、 またはプロジェクトをコピーして書き込み可能な状態にし、 そのコピーをアップグレードするようにしてください。

バックアップ対象となる項目

プロジェクトのアップグレード時に、「変換前にバックアップを作成する」オプションを選択した場合、 バックアップ対象になるのはプロジェクトフォルダ下にある項目のみです。 プロジェクトフォルダの外部にある項目はバックアップ対象になりません。

非 Web プロジェクトは ASP.NET 関連項目を含むことができない

Visual Studio 2005 以降、 Web プロジェクトの機能は Visual Web Developer が提供するようになりました。 これに伴い、NetCOBOL for .NET プロジェクトが ASP.NET 関連項目を含むことができなくなります。 このため、Web プロジェクトではない NetCOBOL for .NET プロジェクトが ASP.NET 関連項目を含んでいる場合、 その項目はアップグレード後のプロジェクトから削除されます。 その場合は、ASP.NET 関連項目を削除したことがアップグレードレポートに出力されます。

Web プロジェクトをアップグレードする際の注意事項

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

V2.1 以前の NetCOBOL for .NET Web プロジェクトをアップグレードする場合の注意事項について説明します。NetCOBOL for .NET プロジェクトをアップグレードする際の全般的な注意事項に関しては、 「プロジェクトをアップグレードする際の全般的な注意事項」を参照してください。

プロジェクト構造の変更にかかわる注意点

Visual Studio 2005 以降、 Web プロジェクトの機能は NetCOBOL for .NET ではなく Visual Web Developer が提供します。 Visual Web Developer はプロジェクトファイルを使わず、 プロジェクトフォルダに含まれるすべてのファイルをプロジェクト資産とみなします。 このため、以下の注意が必要です。

他のプロジェクトとフォルダを共有しない

以前のバージョンの Web プロジェクトで、 他のプロジェクトとフォルダを共有している場合、 意図しないファイルが Web プロジェクト項目とみなされることがあります。 アップグレード前の Web プロジェクトを編集して、 アップグレード対象の Web プロジェクトがプロジェクトフォルダを占有するようにしてください。

出力フォルダのファイルを削除する

以前のバージョンの Web プロジェクトでは、 既定の設定だと、出力フォルダはプロジェクトフォルダの下（¥bin¥）にあります。 Web プロジェクトをアップグレードする前に、出力ファイルや、（もし存在すれば）中間生成ファイルをあらかじめプロジェクトフォルダから取り除いてください。

Web プロジェクトに Web.config ファイルを含める

アップグレード対象の Web プロジェクトには、 プロジェクトフォルダのルートに Web.config ファイルが含まれていなければなりません。

登録集パスの設定を見直す

Visual Web Developer の仕様により、ASP.NET 関連項目以外の COBOL ソースは App_Code サブフォルダに移動されます。 この結果、ソースと登録集の相対位置がずれる可能性があります。 アップグレード後に登録集パスの設定を見直してください。

「Debug|.NET」構成がアップグレード対象となる

Visual Web Developer は一つの構成（Debug|.NET）しかサポートしません。 そのため、以前のバージョンの Web プロジェクトの「Debug|.NET」構成のみが新しい Web プロジェクトへアップグレードされます。

ASP.NET 項目はファイル形式が変換される

ASP.NET 関連項目は、ASP.NET2.0 の形式に変換されます。 特に、.aspx ファイルと.ascx ファイルはコード分離モデルの形式に変換され、 その結果クラスを表す内部名は CLASS-THIS に変更されます。

ソースプログラム中の翻訳指示文(@OPTIONS)を取り除く

以前の版の ASP.NET 関連項目には、ソースプログラム中に翻訳指示文(@OPTIONS)が記述されているものがあります。 しかし、分離コードの COBOL ソースには翻訳指示文を記述できません。 このため、ASP.NET 関連項目のすべての翻訳指示文を調べて、適切に翻訳オプションを記述し、 分離コード形式の ASP.NET 関連項目の翻訳指示文を削除しなければなりません。

翻訳オプションは、 ASP.NET 構成ファイル(Web.config)中の、 language 属性に"Fujitsu.COBOL"という名前を含む compiler 要素の compilerOptions 属性に記述します。 詳細は、コンパイラオプションの設定を参照してください。

相対パスの設定を見直す必要がある場合がある

Visual Web Developer がビルドを行う際のカレントフォルダはプロジェクトフォルダではありません。そのため、登録集パスなどパスを指定するオプションはフルパスで記述する必要があります。

アップグレードで引き継がれる設定

アップグレードによって引き継がれるプロジェクト設定は以下のみです。 その他は手動で設定しなおす必要があります。

- プロジェクト項目（参照と Web 参照を除く）
- Web.config の設定
- プロジェクトのプロパティの「追加オプション」

新しい Web プロジェクトで利用できない設定

以下の設定は、Visual Web Developer に該当する機能がないため、 新しい Web プロジェクトで利用することはできません。

- プロジェクトのプロパティの「ビルドイベント」
- プロジェクトのプロパティの「参照パス」
- プロジェクト項目のプロパティの「カスタムビルドステップ」

COBOL ソースプログラムの作成・編集

COBOL ソースプログラムの編集には、COBOL の予約語のチェック、一連番号の更新、および IntelliSense 機能などが利用できる Visual Studio エディターが便利です。

ここでは、Visual Studio エディターを使った COBOL ソースプログラムの作成と編集方法について説明します。

このセクションの内容

Visual Studio エディターを使った編集

NetCOBOL for .NET では、 Visual Studio エディターに COBOL 言語のサポートを追加しています。 これにより、IntelliSense 機能やアウトライン表示などの Visual Studio がサポートする便利な機能に加え、 COBOL の予約語表示や一連番号編集など COBOL 独特の便利な機能も利用できます。 これらの機能の利用方法について説明します。

COBOL ソースプログラムの形式

COBOL の正書法(固定形式、可変形式、自由形式)および翻訳指示文について説明します。

フォームデザイナの利用

Windows アプリケーションや ASP.NET Web アプリケーション、および XML Web サービスを開発する場合、 フォームデザイナーを利用すると、 これらの開発を簡単に行なうことができます。 フォームデザイナーの利用方法と注意事項について説明します。

NCOBU00056

Visual Studio エディターを使った編集

NetCOBOL for .NET では、 Visual Studio エディターに COBOL 言語のサポートを追加しています。 これにより、Intellisense 機能やアウトライン表示などの Visual Studio がサポートする便利な機能に加え、 COBOL の予約語表示や一連番号編集など COBOL 独特の便利な機能が利用できるようになりました。

多くの便利な機能を備えた Visual Studio エディターを使用することで、 COBOL プログラムの編集がより簡単になり、エラーも少なくすることができます。

ここでは、Visual Studio エディターで利用できる便利な機能とその利用方法について説明します。

このセクションの内容

IntelliSense 機能

IntelliSense 機能は、 Visual Studio エディターがサポートしている編集補助機能です。 利用者の操作を検出して、 その時に必要なヒントなどを自動的に表示することができます。 この利用方法について説明します。

コード スニペット機能

コード スニペット機能は、 Visual Studio エディターがサポートしている編集補助機能です。 あらかじめ作成されたコードのスニペット（断片）を簡単な操作でエディターに挿入することができます。 この利用方法について説明します。

アウトライン表示

アウトライン機能は、クラス単位、メソッド単位、データ部単位など、 意味的にまとまったテキストをグループ化して、 それぞれを表示、非表示することができます。 ソースプログラムの全体的な構造を把握するのに便利です。 この利用方法について説明します。

予約語の表示

従来の NetCOBOL でもサポートしていた COBOL の予約語を表示する機能です。 この機能は予約語のスペルミスを防ぐために役立ちます。 この利用方法について説明します。

登録集を開く機能

登録集を開く機能は、 COPY 文で指定されている登録集ファイルを Visual Studio エディターで開きます。 この利用方法について説明します。

[COBOL エディター]ツールバー

Visual Studio エディターでの COBOL ソースプログラム編集中に利用する、[COBOL エディター]ツールバーについて説明します。

COBOL 固有 テキストエディター オプション ダイアログ

テキストエディターに関するさまざまな設定を行う、 COBOL 固有テキストエディターダイアログボックスについて説明します。

Visual Studio エディター使用時の注意事項

Visual Studio エディター使用時の注意事項について説明します。

IntelliSense 機能

Visual Studio エディターでは IntelliSense 機能を利用することができます。

Intellisense 機能を利用すると、 選択したクラスや集団項目に対する有効なメンバーやデータを表示させ、 表示された一覧からメンバーやデータを選択してプログラムに挿入したり、 データの型情報を表示させたりすることができます。

Intellisense 機能には、以下があります。

- メンバーの一覧
- パラメータヒント
- クイックヒント
- 入力候補

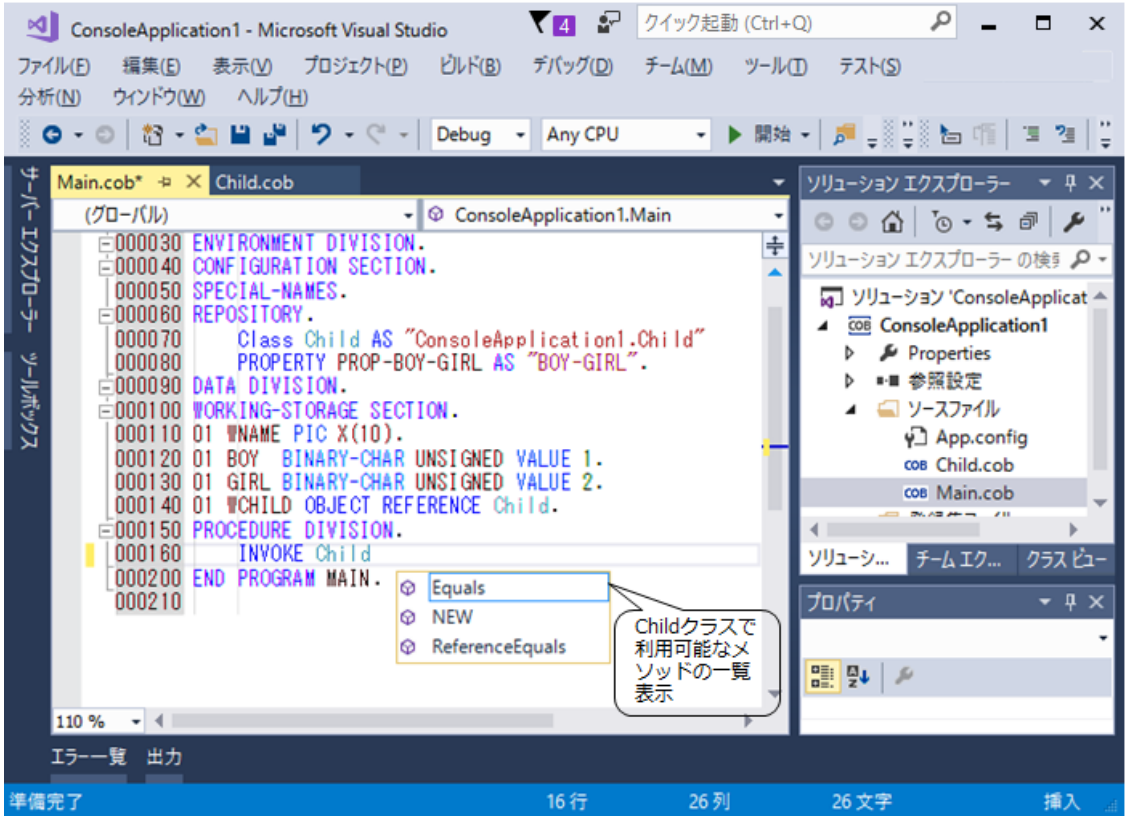
Intellisense に関する詳細な説明は、 Visual Studio のドキュメントの IntelliSense の使用方法を参照してください。

NCOBU00058

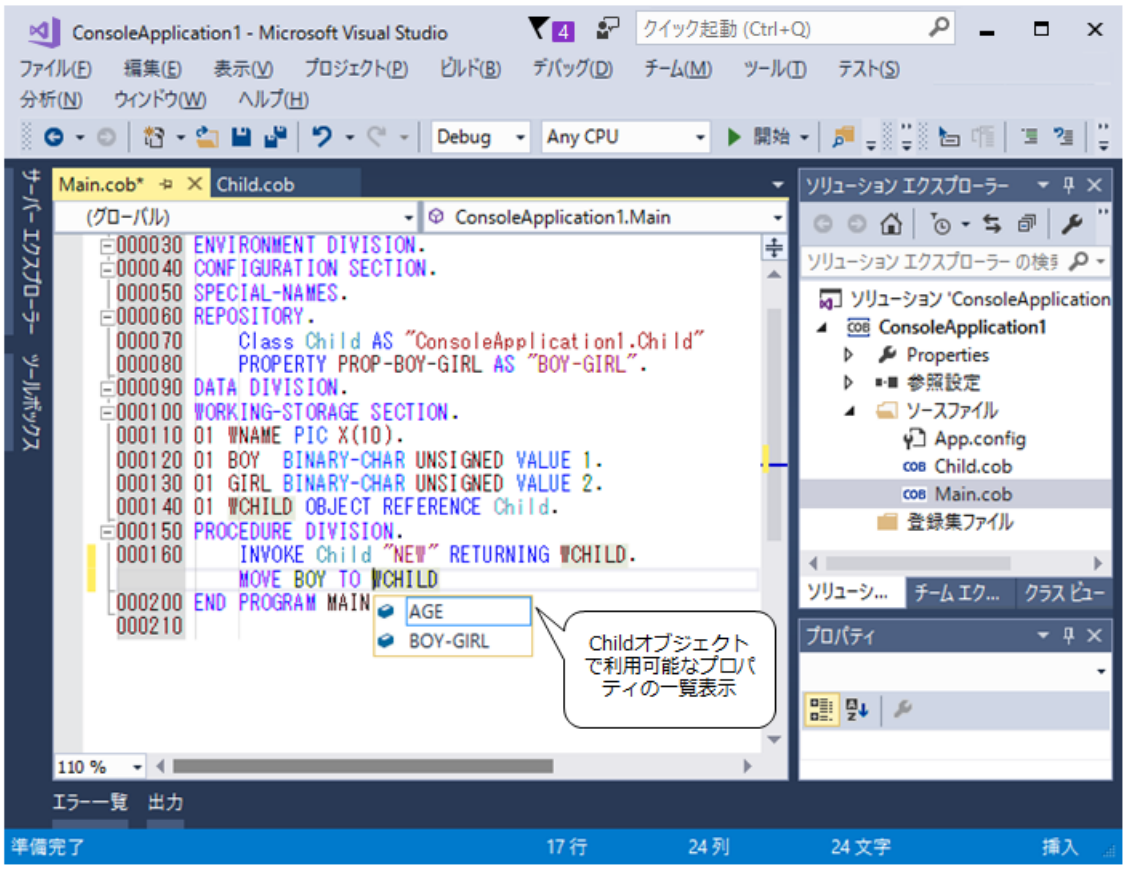
メンバーの一覧

IntelliSense オプションの「メンバーの一覧」が有効になっている場合、 COBOL ソースプログラムの編集中に COBOL の予約語、コード スニペットの他に、 データ名、クラス名またはオブジェクトやクラスで利用できるメンバー名の一覧が表示されます。

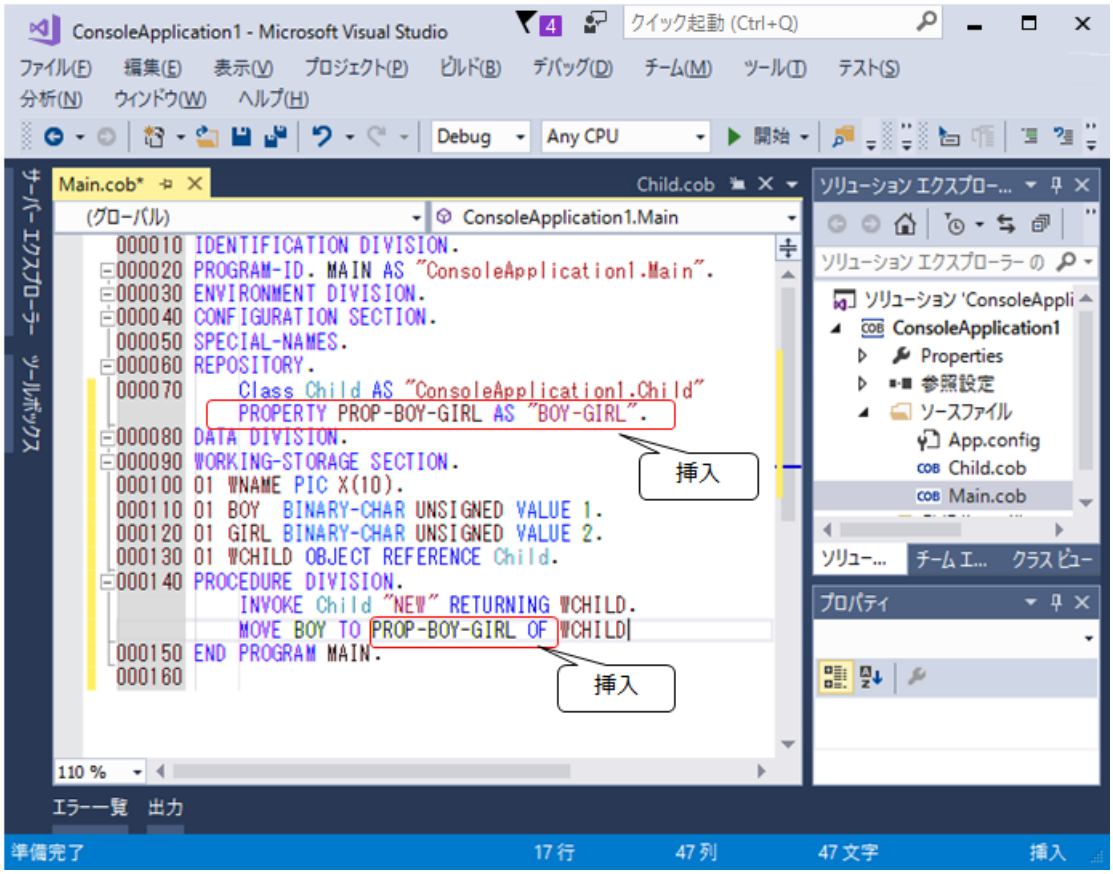
詳細については、Visual Studio のドキュメントの IntelliSense の使用方法の[メンバーの一覧] を参照してください。



また、 クラスやオブジェクトで利用できるプロパティやフィールドのメンバーの一覧を表示させて、 ソースプログラムに挿入することもできます。



一覧の中から、「BOY-GIRL」を選択すると、赤枠で囲った部分が挿入されます。



メンバーの一覧表示を有効にするには、以下の設定を行ってください。 なお、既定ではメンバーの一覧表示は有

効になっています。

- データ名、クラス名、クラスやオブジェクトで利用できるメンバー名の一覧表示

  1. [ツール]メニューから[オプション]を選択します。
  2. [オプション]ダイアログボックスが表示されたら、 [テキストエディター]フォルダ-[COBOL]フォルダの[全般]を選択します。
  3. 全般プロパティページの設定方法については、[全般] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])を参照してください。

- COBOL 予約語、コード スニペット、クラスやオブジェクトで利用できるプロパティ名やフィールド名の一覧表示

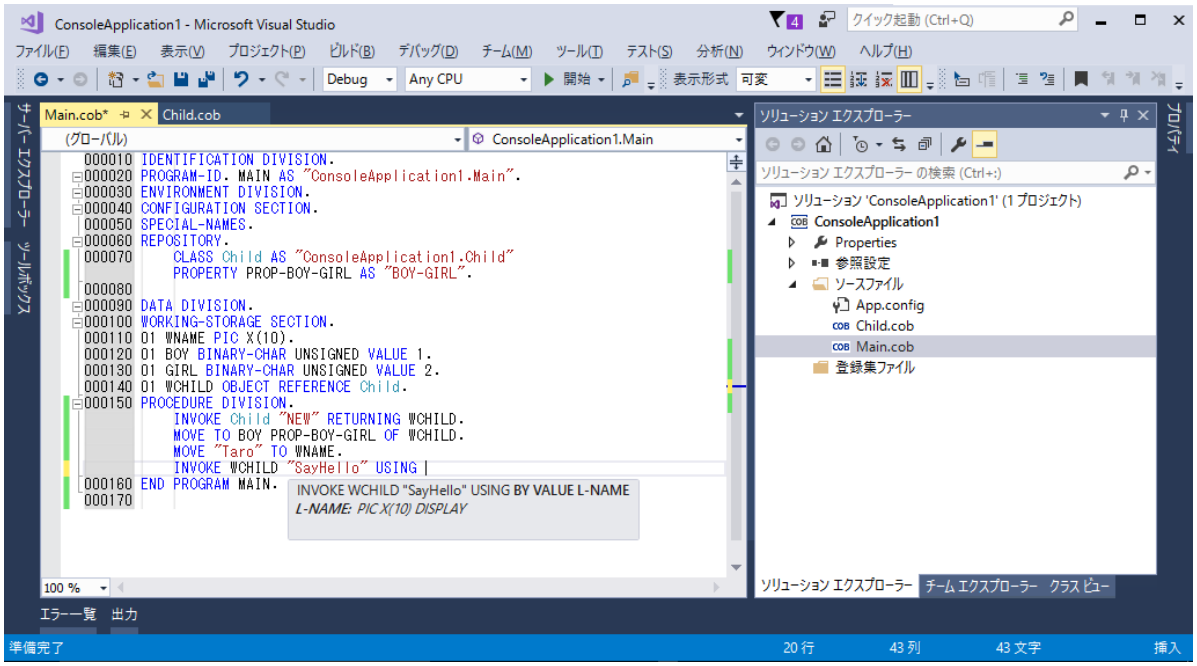
  1. [ツール]メニューから[オプション]を選択します。
  2. [オプション]ダイアログボックスが表示されたら、 [テキストエディター]フォルダ-[COBOL]フォルダの[COBOL 固有]を選択します。
  3. COBOL 固有プロパティの設定方法については、[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])を参照してください。

IntelliSense 機能が起動しない場合、Visual Studio エディター使用時の注意事項の「コード要素の解析を伴う機能を利用する場合」および「IntelliSense 機能を利用する場合」 を参照してください。

パラメータヒント

IntelliSense オプションの「パラメータヒント」が有効になっている場合、 COBOL ソースプログラムの編集で区切り文字(空白など)を入力すると、 INVOKE 文や CALL 文の引数情報が表示されます。

詳細については、Visual Studio のドキュメントの IntelliSense の使用方法の[パラメーター ヒント] を参照してください。



パラメータヒントを有効にするには、以下の設定を行ってください。 なお、既定ではパラメータヒントは有効になっています。

1. [ツール]メニューから[オプション]を選択します。
2. [オプション]ダイアログボックスが表示されたら、 [テキストエディター]フォルダ-[COBOL]フォルダの[全般]を選択します。
3. 全般プロパティページの設定方法については、[全般] ([オプション] ダイアログ ボックス -[テキスト エディター] - [COBOL])を参照してください。

注意

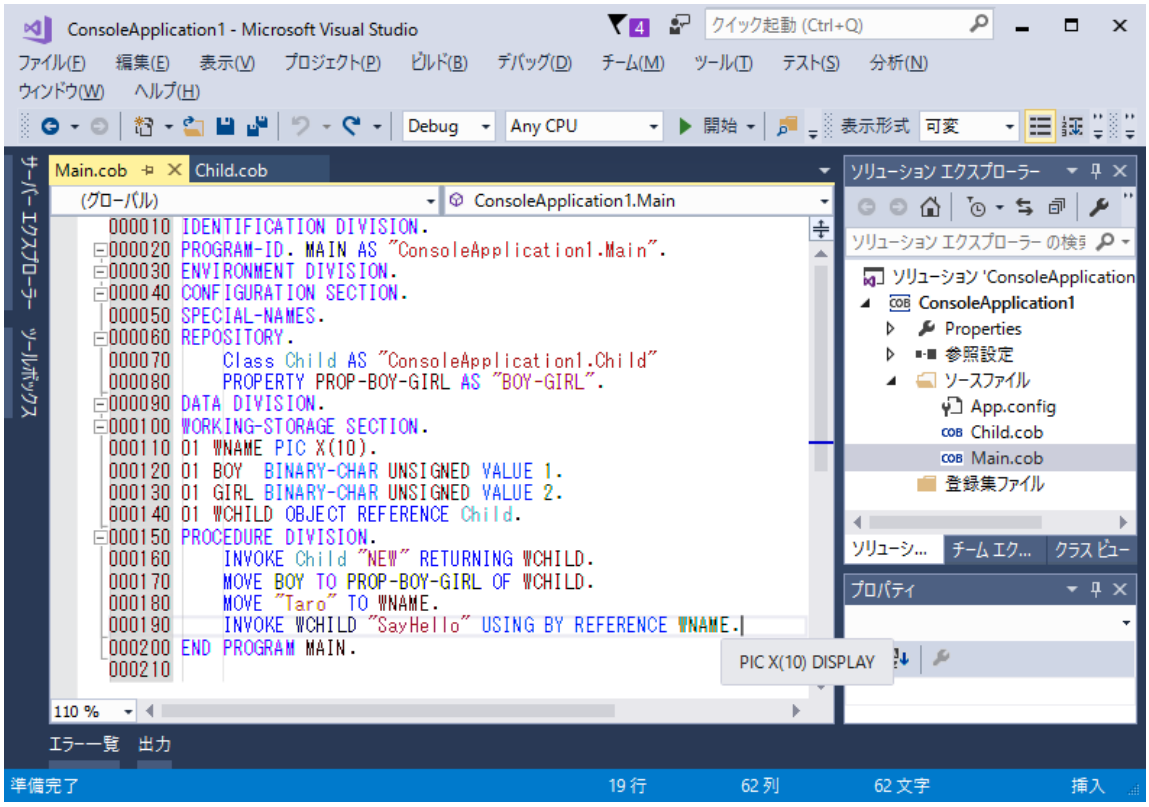
IntelliSense 機能が起動しない場合、Visual Studio エディター使用時の注意事項の「コード要素の解析を伴う機能を利用する場合」および「IntelliSense 機能を利用する場合」 を参照してください。

NCOBU00060

クイックヒント

[クイック ヒント] は、マウスポインタをデータ名やクラス名に合わせることで、 データの宣言やクラス名の外部名がポップアップボックスに表示されます。

詳細については、Visual Studio のドキュメントの IntelliSense の使用方法の[クイック ヒント] を参照してください。



注意

IntelliSense 機能が起動しない場合、Visual Studio エディター使用時の注意事項の「コード要素の解析を伴う機能を利用する場合」および「IntelliSense 機能を利用する場合」 を参照してください。

入力候補

[入力候補] は、データ名やクラス名を特定できる部分まで入力し、 残りの部分を自動的に補完する機能です。

[入力候補]を起動するには、名前を途中まで入力してから、 Alt キーを押しながら→キーを押すか、Ctrl キーを押しながら Space キーを押します。

詳細については、Visual Studio のドキュメントの IntelliSense の使用方法の[入力候補] を参照してください。

IntelliSense 機能が起動しない場合、Visual Studio エディター使用時の注意事項の「コード要素の解析を伴う機能を利用する場合」および「IntelliSense 機能を利用する場合」 を参照してください。

NCOBU00960

コード スニペット機能

Visual Studio エディターではコード スニペット機能を利用することができます。

コード スニペット機能を利用すると、あらかじめ作成されたコードのスニペット（断片）を簡単な操作でエディターに挿入することができます。

ここでは、エディターでの挿入方法とコード スニペットの作成および追加方法について説明します。

- コード スニペットの挿入
- コード スニペットの作成
- コード スニペットの追加

コード スニペットに関する詳細な説明は、 Visual Studio のドキュメントのコード スニペットを参照してください。

コード スニペットの挿入

コード スニペットを挿入するには、スニペットピッカーを使用する方法とショートカット名による挿入方法があります。

スニペットピッカーを使用したコード スニペットの挿入

スニペットピッカーを使用してコード スニペットを挿入します。

1. エディターでコード スニペットの挿入位置にカーソルを移動します。
2. [編集]メニューから[IntelliSense]-[スニペットの挿入]、または、コンテキストメニューから[スニペットの挿入]を選択し、スニペットピッカーを表示します。
3. コード スニペットの名前を入力または選択し、Tab キーまたは Enter キーを押すことでコード スニペットが挿入されます。
4. 編集を必要とするフィールドがある場合は、そのフィールドが四角い枠で囲まれて編集状態になります。
5. 各フィールドを適切に編集します。フィールドが複数ある場合は、Tab キーまたは Shift+Tab キーで各フィールドを移動することができます。
6. Esc キーまたは Enter キーでフィールドの編集状態を解除します。

ショートカット名によるコード スニペットの挿入

コード スニペットにショートカット名が定義されている場合は、ショートカット名を使用した挿入を行うことができます。

1. エディターでコード スニペットの挿入位置にカーソルを移動します。挿入位置は行の先頭または区切り文字（空白など）の後に位置付けてください。
2. ショートカット名を入力します。
3. Tab キーを押すことで、コード スニペットが挿入されます。
4. 編集を必要とするフィールドがある場合は、そのフィールドが四角い枠で囲まれて編集状態になります。
5. 各フィールドを適切に編集します。フィールドが複数ある場合は、Tab キーまたは Shift+Tab キーで各フィールドを移動することができます。
6. Esc キーまたは Enter キーでフィールドの編集状態を解除します。

Ctrl+K キーを押したあとに Ctrl+J キーを押すと、ショートカット名の一覧を表示することができます。一覧からショートカット名を選択し、Tab キーまたは Enter キーを押すことでショートカット名を入力することができます。

NCOBU00962

コード スニペットの作成

コード スニペットは、ファイル名の拡張子を .snippet とした XML ファイルとして作成します。 ここでは、スニペットファイルの簡単な記述例を示し説明します。

詳細については、 Visual Studio のドキュメントのチュートリアル: コード スニペットを作成するおよびコード スニペット スキーマ リファレンスを参照してください。

コード スニペットの記述例

COBOL ソースコードに DISPLAY 文を挿入する簡単な記述例を示します。

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>MySnippet</Title>                    <!-- [1] -->
      <Description>Snippet Sample</Description>
      <SnippetTypes>
         <SnippetType>Expansion</SnippetType>       <!-- [2] -->
      </SnippetTypes>
    </Header>
    <Snippet>
      <Code Language="Fujitsu.COBOL">              <!-- [3] -->
        <![CDATA[DISPLAY "Hello World"]]>          <!-- [4] -->
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

このファイルの内容を説明します。

[1]: コード スニペットの挿入で表示される名前を指定します。

[2]: スニペットの挿入で使用することを示すために、SnippetType 要素に Expansion を指定します。

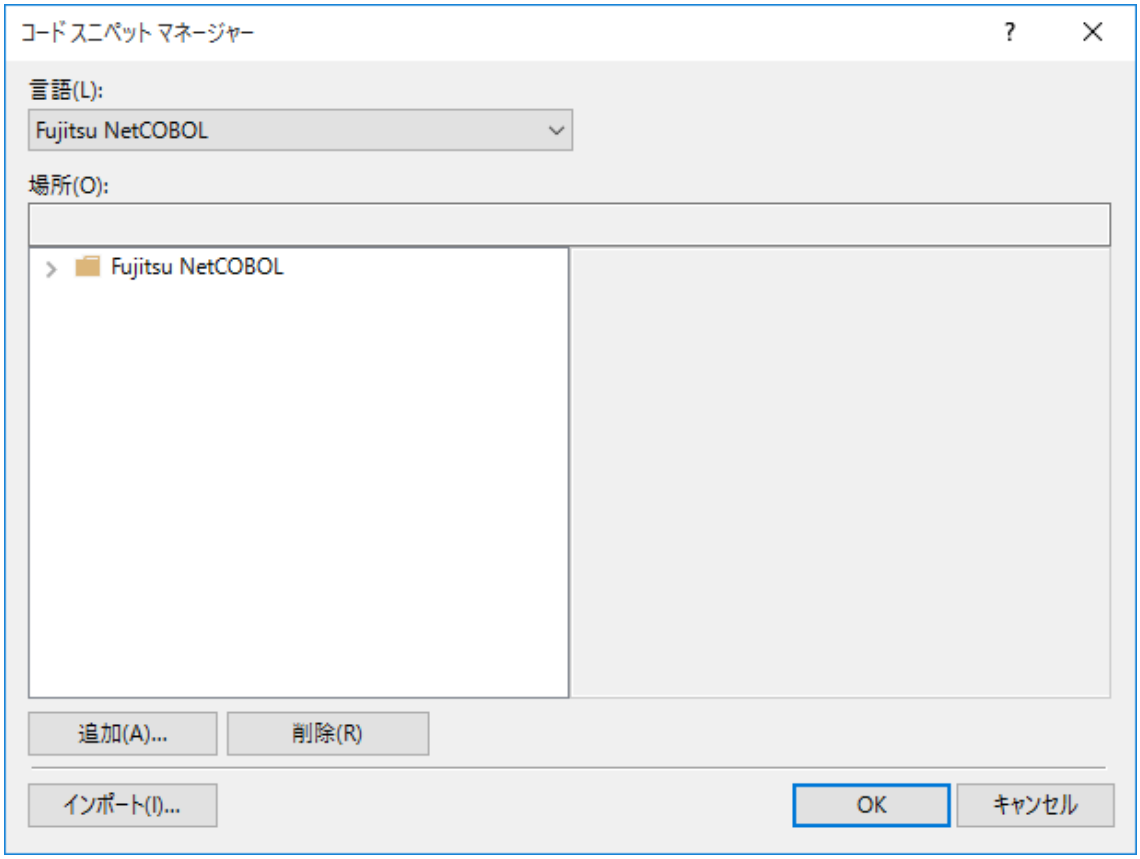
[3]: Code 要素の Language 属性は、Fujitsu.COBOL を指定します。

[4]: 挿入される COBOL のコードを指定します。 この記述例では、以下のコードが挿入されます。

```
DISPLAY "Hello World"
```

コード スニペットの追加

作成したコードスニペットを利用するには、作成したコードスニペットがあるフォルダをコードスニペットマネージャーに追加します。コードスニペットマネージャーは[ツール]メニューから[コードスニペットマネージャー]を選択することで表示します。



1. [言語]で[Fujitsu NetCOBOL]を選択します。
2. [追加]ボタンで、作成したコードスニペットのあるフォルダを追加します。

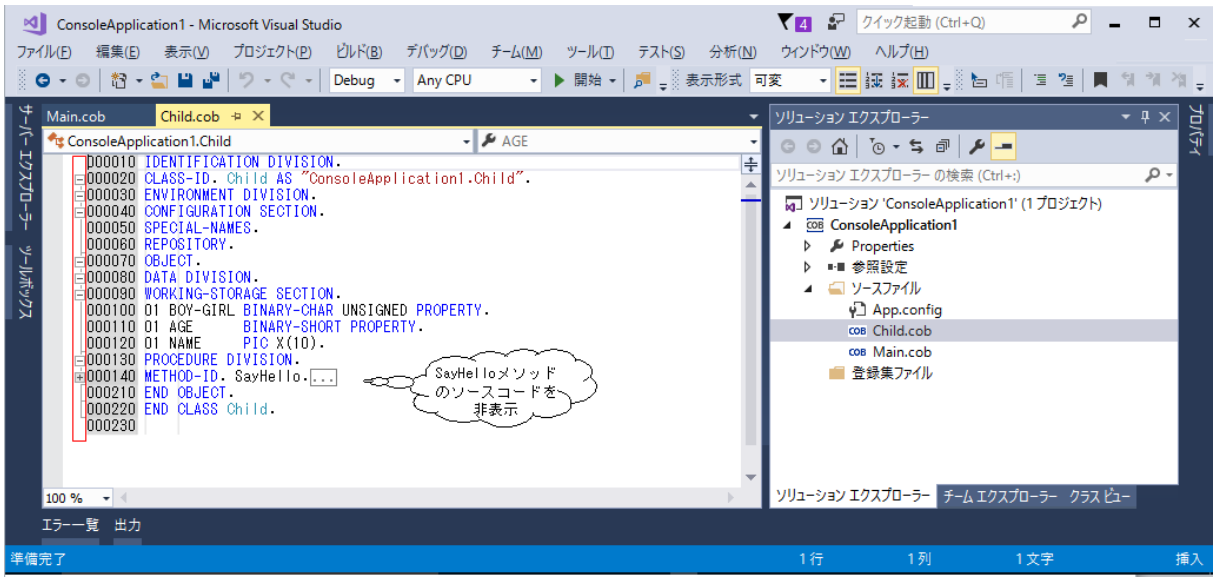
NCOBU00062

アウトライン表示

アウトライン機能は、クラス単位、メソッド単位、データ部単位など、 意味的にまとまったテキストをグループ化して、 それぞれを表示、非表示することができます。 この機能は、ソースプログラムの全体的な構造を把握するのに便利です。

アウトライン機能の詳細については、Visual Studio のドキュメントのアウトラインを参照してください。

下の例では、メソッド部分を非表示にしています。



アウトライン機能を有効にするには、以下の設定を行なってください。

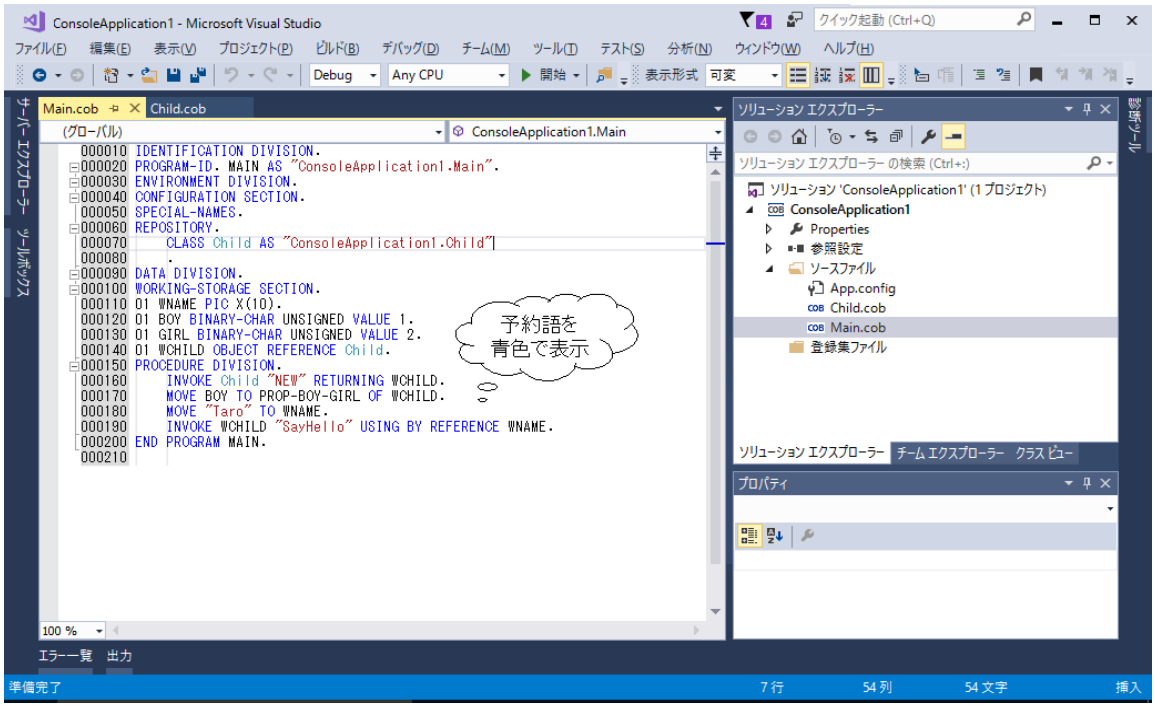
1. カレントビューを COBOL ソースプログラムを編集しているビューに合わせる。
2. [編集]メニューから[アウトライン]をクリックし、表示する展開の種類を選択する。

また、左側の行番号が表示されていない場合は、以下の手順で表示できます。

1. [ツール]メニューから[オプション]を選択します。
2. [オプション]ダイアログボックスが表示されたら、 [テキスト エディター]フォルダ-[COBOL]フォルダの[全般]を選択します。
3. 全般プロパティページの[行番号]をチェックします。 全般プロパティページの詳細については、[全般]([オプション] ダイアログ ボックス -[テキスト エディター] - [COBOL])を参照してください。

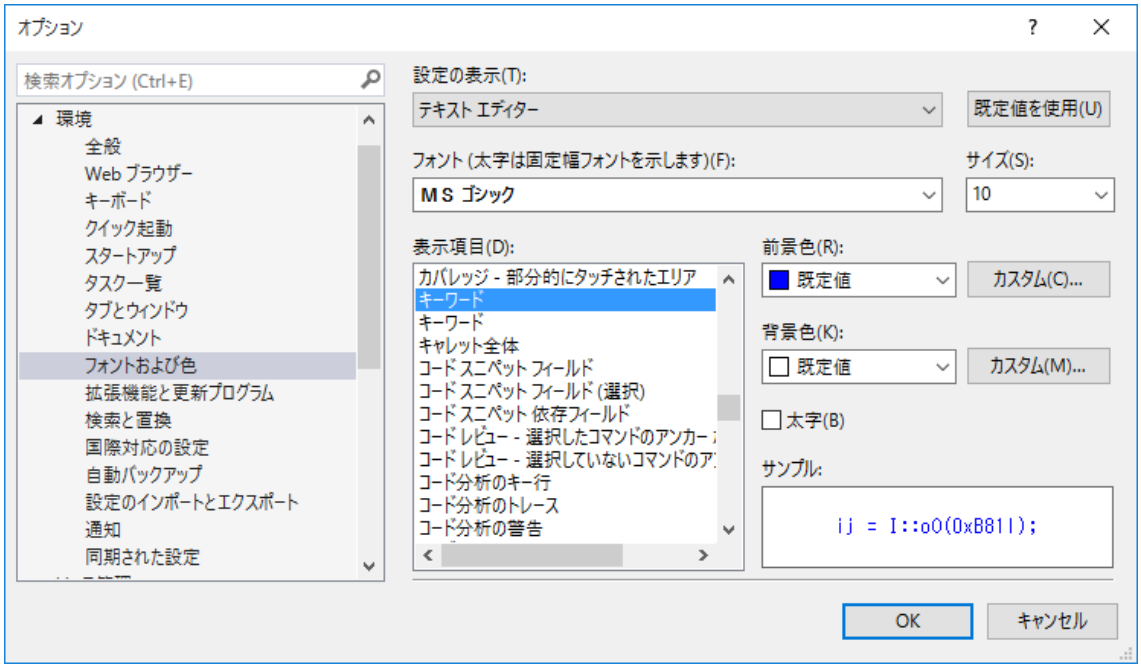
予約語の表示

Visual Studio エディターでは、 入力した COBOL の予約語が指定した色で表示されます。 この機能は、予約語のスペルチェックに役立ち、コーディングミスを防ぐことができます。



表示する色を変更するには、以下の設定を行ってください。

1. [ツール]メニューから[オプション]を選択します。
2. [オプション]ダイアログボックスが表示されたら、[環境]フォルダの[フォントおよび色]を選択します。
3. [表示項目]リストボックスから、[キーワード]を選択し、[前景色]で変更後の色を選択します。



予約語を英大文字に変換して表示させる

COBOL の予約語を英大文字に変換して表示させることもできます。この機能を有効にする方法は、[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])を参照してください。

予約語のほかに、文字定数および数字定数を指定した色で表示することができます。 上の[オプション]ダイアログボックスで、 [表示項目]リストボックスから「文字列」または「数字」を選択して色を変更します。

登録集を開く機能

登録集を開く機能を利用すると、COPY 文で指定されている登録集ファイルを簡単な操作で開くことができます。登録集の内容の確認や編集を行うときに便利です。

この機能を利用するためには、次の操作を行います。

1. エディターで COPY 文が記述されている位置にカーソルを移動します。
2. コンテキストメニューから[登録集を開く]、または、ショートカットキー[Ctrl + Shift + G] を選択します。

COPY 文で指定している登録集が見つからない場合はエラーが表示されます。 Visual Studio を利用した登録集パスの設定方法については、登録集パスの設定を参照してください。

NCOBU01159

[COBOL エディター]ツールバー

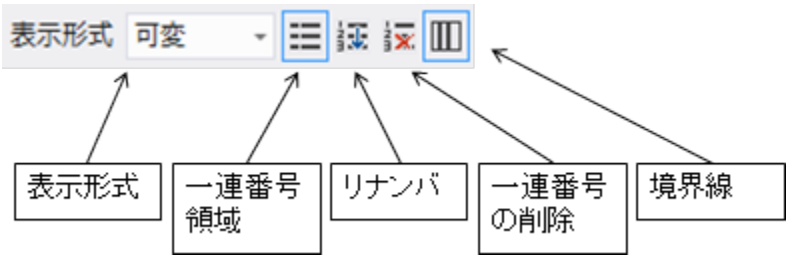
[COBOL エディター]ツールバーでは、COBOL 特有の正書法に従って一連番号領域の編集や境界線を表示する機能を提供しています。 COBOL の正書法については、COBOL ソースプログラムの形式を参照してください。

[COBOL エディター]ツールバーには以下の機能があります。

- 表示形式の選択
- 一連番号領域の表示
- 一連番号のリナンバ
- 一連番号の削除
- 境界線の表示

[COBOL エディター]ツールバーは、以下の操作で表示されます。

1. [表示]メニューから[ツールバー]を選択し、[COBOL エディター]をチェックします。
2. [COBOL エディター]ツールバーが表示されます。



表示形式の選択

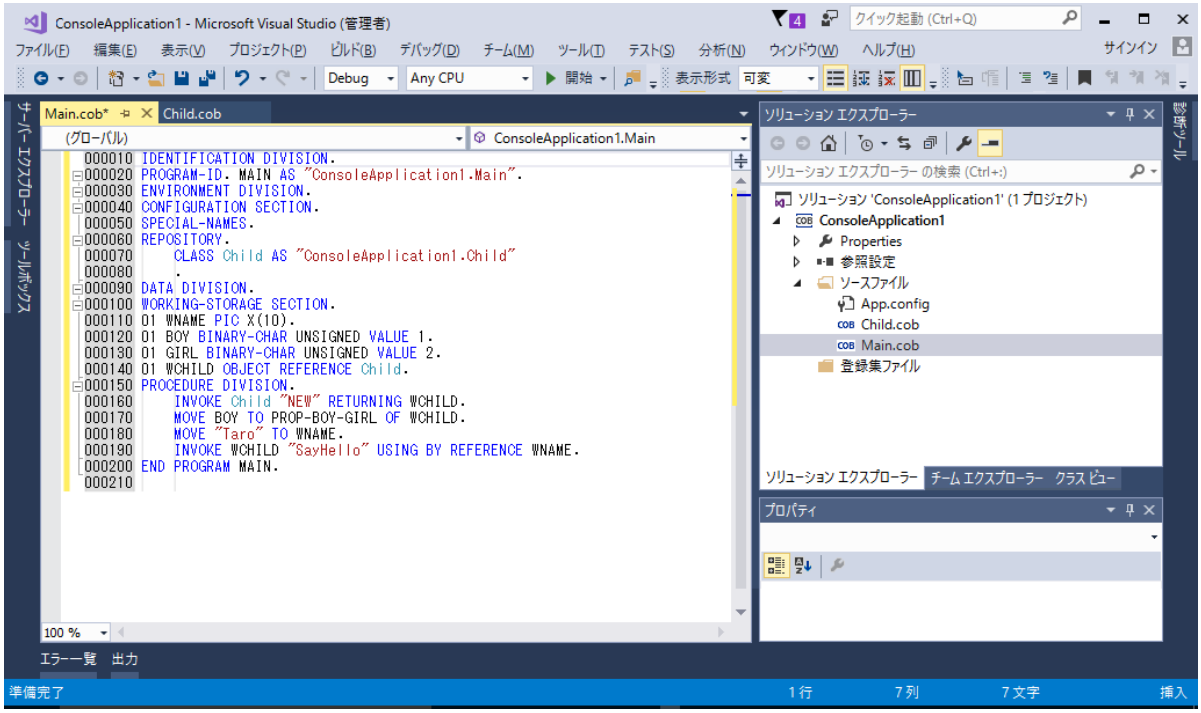
COBOL ソースプログラムの形式に対応した表示や操作に切り替えることができます。 エディター上での表示や操作などが切り替わるだけで、プロジェクトや COBOL ソースなどへの変更は行われません。

一連番号領域の表示

一連番号領域の背景色を表示します。

一連番号のリナンバ

一連番号領域の番号が昇順にリナンバされます。



一連番号の自動更新は、挿入した行に自動的に番号を付加することができます。付加される番号は前後の行の一連番号を意識した番号になります。

一連番号の自動更新を有効にする方法は、[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])を参照してください。

一連番号の削除

一連番号領域を空白文字に置き換えます。

境界線の表示

COBOL ソースプログラムの形式に対する各領域の境界線を表示します。 表示される境界線の位置は、エディターで使用されているフォントや文字により実際のカラム位置と異なることがあります。 各領域の目安として使用してください。

NCOBU00065

COBOL 固有 テキストエディター オプション ダイアログ

COBOL ソースプログラムをより編集しやすくするために、 テキストエディターにさまざまな設定を行なうことができます。 先に説明した IntelliSense 機能やアウトライン表示などもそのひとつです。 これらのテキストエディターに関するさまざまな設定を行なうのが、 COBOL 固有 テキストエディター オプション ダイアログボックスです。

ここでは、COBOL 固有 テキストエディター オプション ダイアログについて説明します。

COBOL 固有 テキストエディター オプション ダイアログの表示

COBOL 固有 テキストエディター オプション ダイアログは、以下の手順で表示されます。

1. [ツール]メニューから、[オプション]を選択します。 これによって、[オプション]ダイアログボックスが表示されます。
2. [オプション]ダイアログボックスで、 [テキスト エディター]フォルダ-[COBOL]フォルダを選択すると、以下の各プロパティページを表示できます。

- [全般]プロパティページ
- [スクロール バー]プロパティページ
- [タブ]プロパティページ
- [COBOL 固有]プロパティページ

このセクションの内容

[全般] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[全般]プロパティページについて説明します。

[スクロール バー] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[スクロール バー]プロパティページについて説明します。

[タブ] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

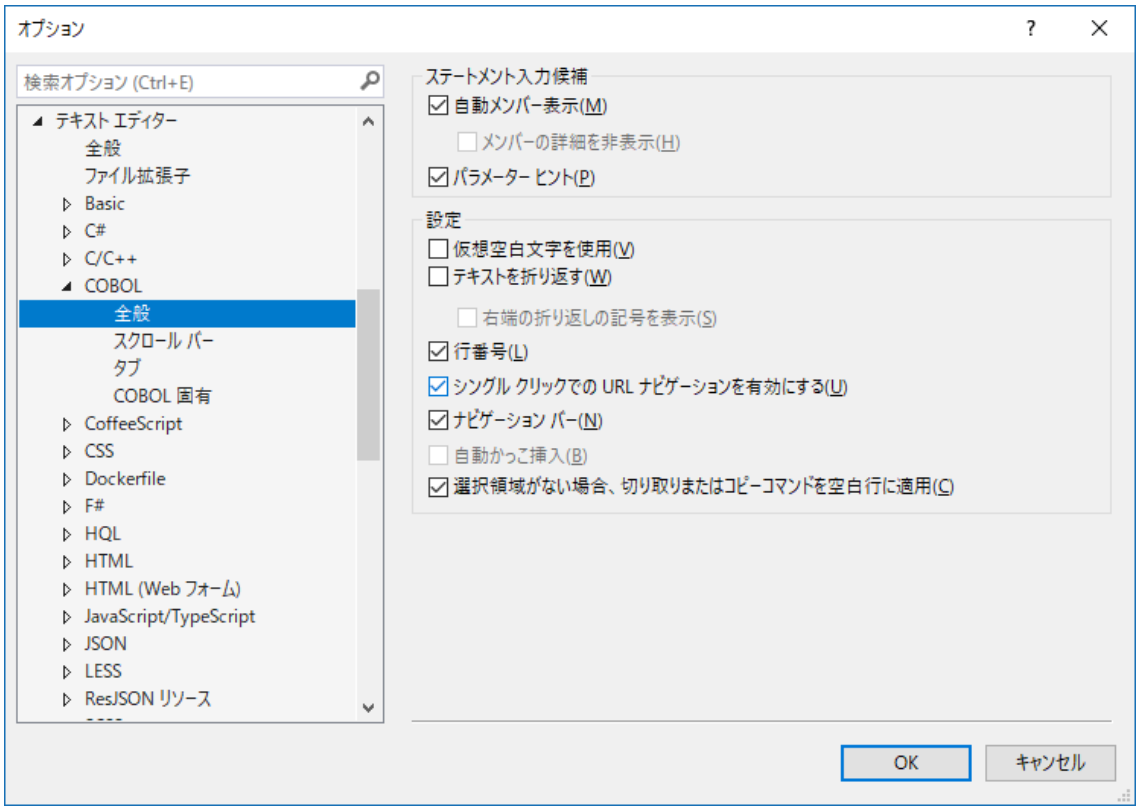
[タブ]プロパティページについて説明します。

[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[COBOL 固有]プロパティページについて説明します。

[全般] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[オプション] ([ツール] メニュー) ダイアログボックスの [テキスト エディター] フォルダにある[COBOL]フォルダの [全般] プロパティページでは、次のプロパティを指定します。



[ステートメント入力候補]

[自動メンバー表示]

オンにすると、コード行を入力するときに、データやクラスおよびオブジェクトで利用できるメンバーの一覧が適切な位置に表示されます。詳細については、メンバーの一覧を参照してください。なお、対象となるクラスやオブジェクトで利用可能なプロパティやフィールドメンバーを自動的に表示する場合は[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])の[プロパティ名の自動メンバー表示]オプションもオンにします。

[メンバーの詳細を非表示]

このオプションは、NetCOBOL for .NET では利用できません。

[パラメーター ヒント]

オンにすると、可能な場合に、INVOKE 文や CALL 文の引数情報が表示されます。詳細については、パラメータヒントを参照してください。

[設定]

[仮想空白文字を使用]

オンにすると、テキストの行末の右側で文字が存在しない場所にカーソルを移動できるようになります。この場所にカーソルを移動し、入力を開始すると、現在位置と最後の単語の間の空間に空白文字が自動的に挿入されます。この設定がオンになっていない場合は、行の最後の文字の右側にカーソルを移動することはできません。

[テキストを折り返す]

オンにすると、エディターの編集画面の表示範囲よりも長い行は、自動的に折り返されます。

[右端の折り返しの記号を表示する]

オンにすると、右端で折り返された行に改行インジケータが表示されます。

[行番号]

オンにすると、各行のテキストの左側に行番号が表示されます。

[シングル クリックでの URL ナビゲーションを有効にする]

オンにすると、テキストに埋め込まれた URL をシングルクリックすると、その Web ページを表示できます。

[ナビゲーション バー]

オンにすると、エディターの一番上にナビゲーションバーが表示されます。ナビゲーションバーを使用して、手続き内のメソッドなどに移動することができます。

[選択領域がない場合、切り取りまたはコピー コマンドを空白行に適用]

オンにすると、空白行にカーソルがあり、何も選択せずに切り取りまたはコピーを行った場合に、その行の切り取りとコピーが可能になります。

関連トピックス

[タブ] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

テキストエディターの既定のタブ設定を変更できます。

[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

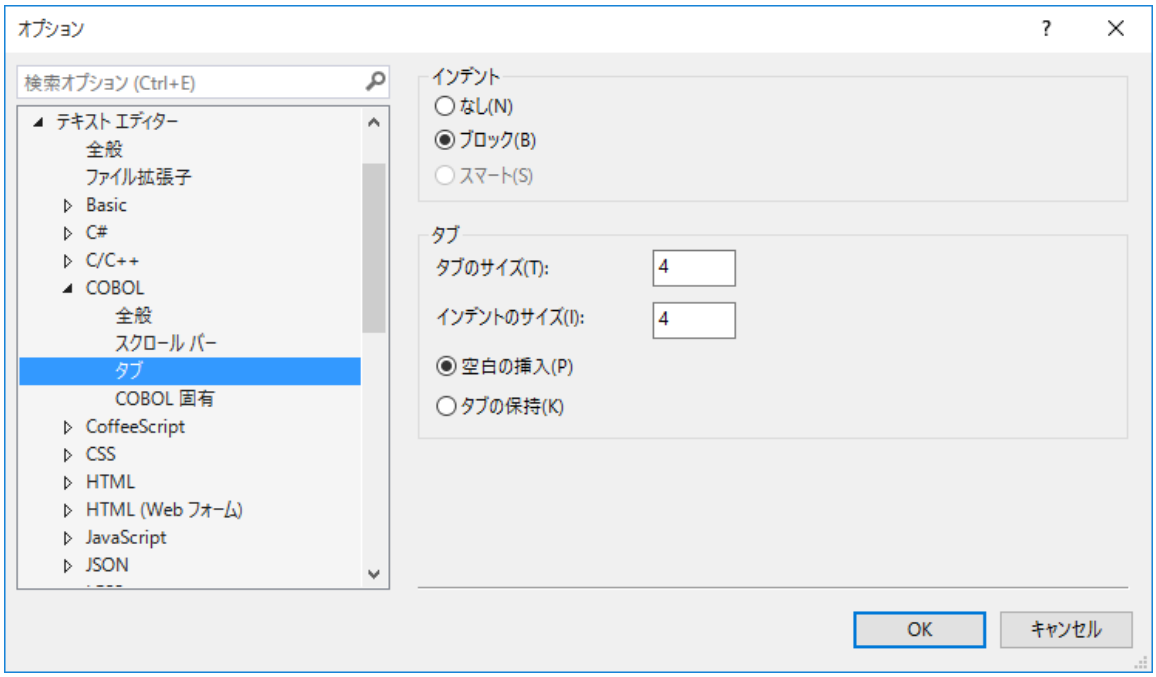
[プロパティ名の自動表示]オプションはここで設定します。

Visual Studio のドキュメントの[全般] ([オプション] ダイアログ ボックス - [環境])

統合開発環境 (IDE: integrated development environment) の既定の設定を変更します。

[タブ] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[オプション] ([ツール] メニュー) ダイアログボックスの[テキスト エディター]フォルダにある[COBOL]フォルダの [タブ]プロパティページでは、テキストエディターの既定のタブ設定を変更できます。



[インデント]

[なし]

オンにすると、Enter キーを押して新しい行に移動しても、自動インデントは行われません。 カーソルは、次の行の最初の列に移動します。

[ブロック]

オンにした場合、Enter キーを押すと、新しい行のテキストは、前の行と同じタブ位置に自動的にインデントされます。

[スマート]

このオプションは、NetCOBOL for .NET では利用できません。

[タブ]

[タブ サイズ]

タブ文字で表されるスペースの数を指定します。

[インデント サイズ]

Tab キーを押したときにインデント操作によって挿入されるスペースの数を指定します。挿入されたスペースは、タブ文字またはスペース文字の組み合わせになる場合がありま

す。

[空白の挿入]

オンにすると、Tab キーを押したときにタブ文字ではなく空白文字が挿入されます。たとえば、インデント サイズを 5 に設定すると、Tab キーを押したときに、テキスト内の現在の行に 5 つのスペースが挿入されます。

[タブの保持]

オンにすると、可能な限り、Tab キーを押したときに空白文字ではなくタブ文字が挿入されます。 タブ文字の長さは、[タブ サイズ] で指定したスペースの数に相当します。インデント サイズがタブ サイズの倍数ではない場合は、残りのスペースに空白文字が挿入されます。

関連トピックス

[全般] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

COBOL テキストエディターの全般的な設定について説明します。

[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

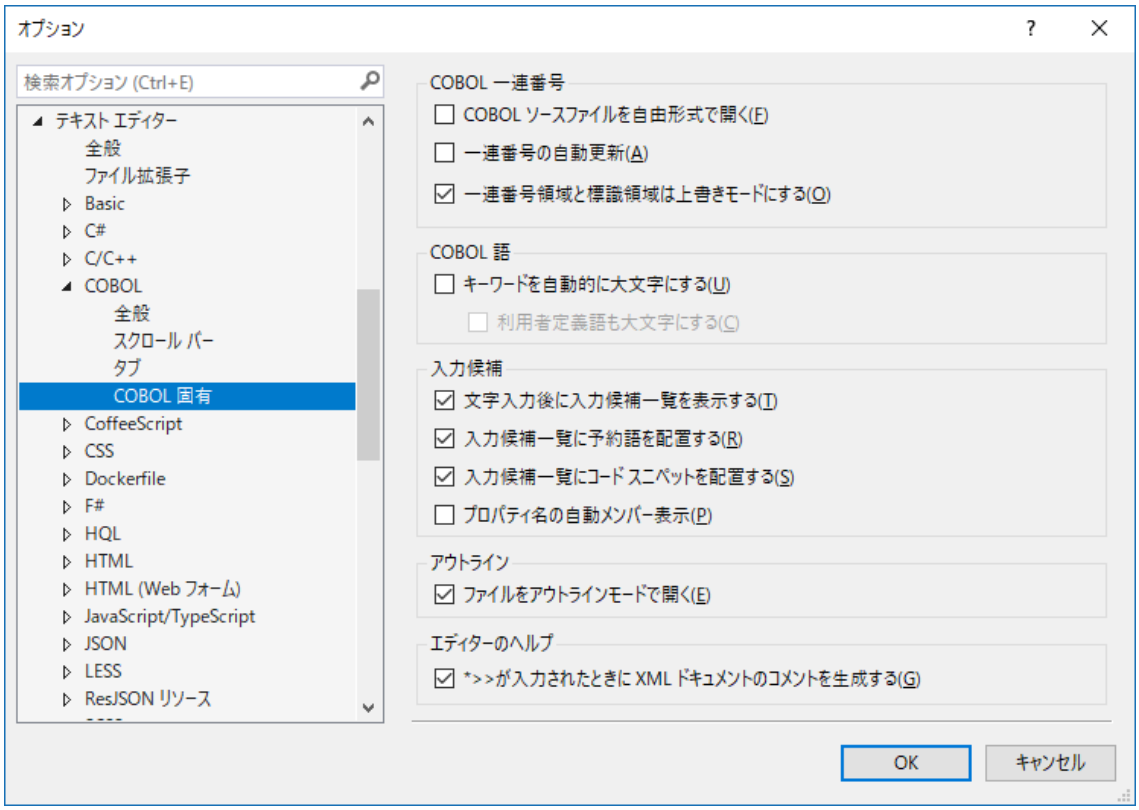
[プロパティ名の自動表示]オプションはここで設定します。

Visual Studio のドキュメントの[全般] ([オプション] ダイアログ ボックス - [環境])

統合開発環境 (IDE: integrated development environment) の既定の設定を変更します。

[COBOL 固有] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[オプション] ([ツール] メニュー) ダイアログ ボックスの [テキスト エディター] フォルダにある [COBOL] フォルダの [COBOL 固有] プロパティ ページでは、次のプロパティを指定できます。



[COBOL 一連番号]

[COBOL ソースファイルを自由形式で開く]

オンにすると、COBOL ソースファイルを常に自由形式で開きます。 なお、オフの場合は翻訳オプション SRF (正書法の種類)により決定します。

COBOL ソースプログラムの形式の詳細については、正書法(固定形式、可変形式、自由形式)を参照してください。 また、翻訳オプション SRF の設定方法については、翻訳オプションの設定を参照してください。

[一連番号の自動更新]

オンにすると、COBOL ソースプログラムを編集すると、左側の一連番号が自動的に更新されます。 COBOL ソースが自由形式で開かれている場合、このオプションは無効になります。

[一連番号領域と標識領域は上書きモードにする]

オンにすると、左側の一連番号領域と標識領域(1 カラムから 7 カラム)が上書きモードになります。 COBOL ソースが自由形式で開かれている場合、このオプションは無効になります。

[COBOL 語]

[キーワードを自動的に大文字にする]

オンにした場合、編集中の COBOL 文が確定された場合に予約語などのキーワードが自動的に大文字に変換します。

[利用者定義語も大文字にする]

オンにすると、編集中の COBOL 文が確定された場合にキーワードと利用者語が大文字に変換します。[キーワードを自動的に大文字にする]オプションがオフの場合、このオプションは無効になります。

[入力候補]

[文字入力後に入力候補一覧を表示する]

オンにすると、エディターでキー入力をするときに、IntelliSense により自動的に入力候補一覧が表示されます。オフの場合も、ショートカットキー(Ctrl+Space など)やメニューを使って、入力候補一覧を使用できます。

[入力候補一覧に予約語を配置する]

オンにすると、IntelliSense により、COBOL の予約語が入力候補一覧に追加されます。

[入力候補一覧にコード スニペットを配置する]

オンにすると、IntelliSense により、COBOL のコード スニペットの識別名が入力候補一覧に追加されます。コード スニペットの識別名が予約語と同じ場合、予約語はコード スニペットで置き換えられます。

[プロパティ名の自動メンバー表示]

オンにすると、コード行を入力するときに、対象のオブジェクトまたはクラスで利用できるプロパティやフィールドの一覧を対象のオブジェクトの直前に表示します。このオプションを利用する場合は、[全般] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])の[自動メンバー表示]オプションもオンにします。

[アウトライン]

[ファイルをアウトラインモードで開く]

オンにすると、ファイルを開くときにアウトライン表示機能を有効にします。

[エディターのヘルプ]

[*>>が入力されたときに XML ドキュメントのコメントを生成する]

オンにすると、*>>が入力されたときに、 型およびメンバーの XML ドキュメントコメントの<summary>開始タグと終了タグを自動的に追加します。

関連トピックス

[タブ] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

テキストエディターの既定のタブ設定を変更できます。

[全般] ([オプション] ダイアログ ボックス - [テキスト エディター] - [COBOL])

[自動メンバー表示]オプションはここで設定します。

Visual Studio のドキュメントの[全般] ([オプション] ダイアログ ボックス - [環境])

統合開発環境 (IDE: integrated development environment) の既定の設定を変更します。

NCOBU01161

Visual Studio エディター使用時の注意事項

ここでは、Visual Studio エディターを使用する場合の注意事項について説明します。

コード要素の解析を伴う機能を利用する場合

ここでの注意事項に該当する機能には、以下があります。

- IntelliSense
- COBOL 語の色付け表示
- 定義へ移動
- 定義をここに表示
- クラス ビュー
- オブジェクト ブラウザー
- アウトライン表示

各機能の注意事項について説明します。

- COBOL ソースプログラムに誤りがある場合､各機能が意図どおりに動作しない場合があります。また、ソースに誤りがなくても、 以下の場合には動作しないことがあります。
  - COBOL ソースプログラム中に REPLACE 文が記述されている場合。
  - COBOL ソースプログラム中に COPY 文が記述されている場合。 COBOL のコード要素の解析では COPY 文は単純に読み飛ばされます。 COPY 文を読み飛ばした結果、プログラムの構造が解析できなくなった場合は各機能が意図とおりに動作しません。

IntelliSense 機能を利用する場合

- 通常、IntelliSense はその言語固有の区切り文字を入力することで起動されます。 しかし、COBOL では区切り文字によって文脈が決定できないことが多く、 場合によっては IntelliSense が自動的に起動しないことがあります。 そのような場合はショートカットキー(Ctrl+Space など)やメニューを使って手動で IntelliSense を起動してください。
- メンバーの一覧からプロパティを選択した場合、リポジトリ段落にプロパティ指定子が自動的に追加される場合があります。 このとき、プロパティ指定子の内部名は他の内部名となるべく衝突しないように生成されますが、まれに重複した内部名が生成されることがあります。 もし内部名が重複することで翻訳エラーが発生した場合は、プロパティ指定子の内部名を修正してください。
- IntelliSense の自動一致機能はサポートされていません。
- IntelliSense のコードコメント表示はサポートされていません。
- NetCOBOL for .NET では、[テキストエディター] ツールバーの [クイック ヒント] ツールバーボタンをクリックしても、クイックヒントは起動しません。

COBOL の正書法に従った境界線の表示機能を利用する場合

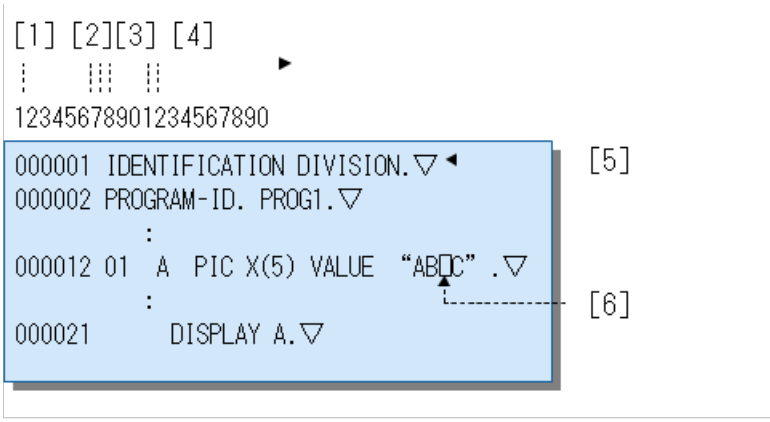
境界線は半角の文字数で区切ります。 全角文字を使用する場合、表示されている境界線の領域と実際の領域が異なる場合があります。 本機能は COBOL ソースプログラムを編集するときの各領域の目安として、ご使用ください。

COBOL ソースプログラムの形式

COBOL ソースプログラムの 1 行の形式は、正書法として COBOL の文法で規定しています。 エディターを使って COBOL ソースプログラムを作成する場合、 一連番号や COBOL の文および手続き名などを、正書法に従った位置から記述する必要があります。

正書法の概要については、正書法(固定形式、可変形式、自由形式)を参照してください。 また、翻訳指示文の書き方については、翻訳指示文を参照してください。

正書法が固定形式または可変形式の場合、一連番号や COBOL の文を、以下の形式で入力します。



※ 上の図では説明のため、改行文字を▽、タブ文字を□で表示しています。

	名称	説明
[1]	一連番号(カラム 1～6)	一連番号には、行番号を記述します。行番号は省略することもできます。
[2]	標識領域(カラム 7)	標識領域は、行を継続する場合や行を注記行にするに使用します。行を継続しない場合や注記行にしない場合は、必ず空白を入力してください。
[3]	A 領域(カラム 8～11)	通常、COBOL の部、節、段落およびプログラムの終わり見出しなどは、この領域から記述します。レベル番号が 01 および 77 のデータ項目もこの領域から記述します。
[4]	B 領域(カラム 12 以降)	通常、COBOL の文およびレベル番号が 01 や 77 でないデータ項目などは、この領域から記述します。
[5]	改行文字(行の終わり)	各行の終わりには、改行文字を入力します。
[6]	TAB 文字	COBOL ソースプログラムの文字定数には、TAB 文字が使用できます。TAB 文字は、文字定数中で 1 バイトを占めます。

自由形式の場合、COBOL ソースプログラムは、行のどの位置からでも記述することができます。ただし、注記、デバック行および継続のための特別な規則があるので注意が必要です。

NCOBU00070

正書法(固定形式、可変形式、自由形式)

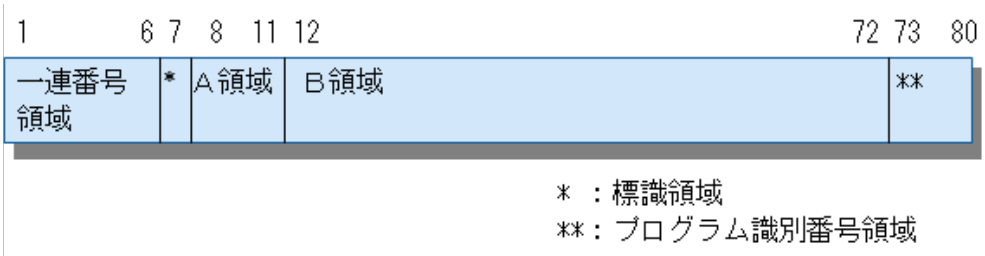
COBOL ソースプログラムの各行は、改行文字で区切ります。そして、その 1 行の形式は、正書法で定める規則に従って記述しなければなりません。

正書法には、固定形式、可変形式および自由形式の 3 つの形式があります。COBOL ソースプログラムは、可変形式で作成するのが一般的です。固定形式または自由形式で COBOL ソースプログラムを作成した場合、翻訳時に翻訳オプション SRF (正書法の種類)を指定する必要があります。

以下にそれぞれの形式について説明します。なお、各行の区切りの改行文字は、行の一部とはみなされません。

- 固定形式

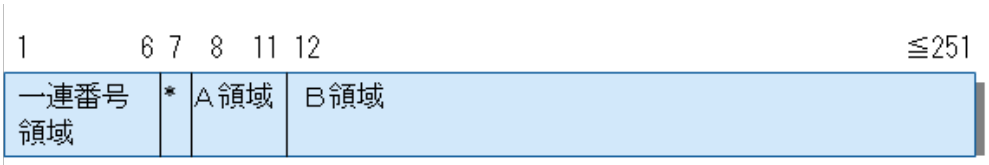
  固定形式では、COBOL ソースプログラムの 1 行の長さを 80 バイトの固定として記述します。



＊ ：標識領域
＊＊：プログラム識別番号領域

- 可変形式

  可変形式では、COBOL ソースプログラムの 1 行の長さを 251 バイト以下の任意のバイト数で記述します。

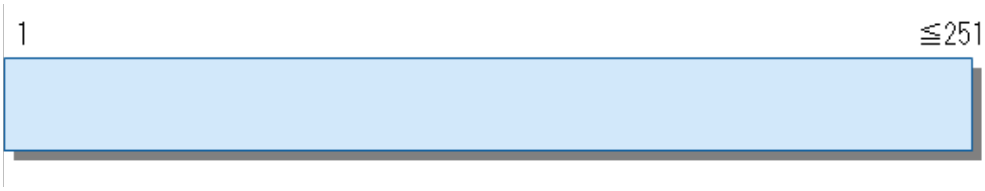


＊ ：標識領域

- 自由形式

  自由形式では、注記、デバッグ行および継続のための特別な規則があることを除いて、COBOL ソースプログラムは、行のどの文字位置からでも記述することができます。

  1 行の文字位置の数は、行ごとに最小 0 から最大 251 の範囲で変更することができます。



登録集原文の正書法の形式は、その登録集原文を取り込む COBOL ソースプログラムの形式と異なってもかまいません。ただし、1 つのプログラムで複数の登録集原文を取り組む場合には、すべての登録集原文の正書法の形式を同一にする必要があります。登録集原文の正書法の形式は、COBOL ソースプログラムと同様に翻訳オプション SRF (正書法の種類)で指定します。

A 領域、B 領域、標識領域については、COBOL ソースプログラムの形式を参照してください。

NCOBU00071

翻訳指示文

翻訳指示文は、翻訳オプションを COBOL コンパイラに指示するために使います。 通常、翻訳オプションは翻訳オプションの設定(プロジェクトのプロパティページ)や cobolc コマンドで説明している方法で指定しますが、 ソースプログラム中に指定することもできます。

翻訳指示文の記述形式を、以下に示します。

```
@OPTIONS [翻訳オプション][,翻訳オプション]...]
```

- @OPTIONS は、8 カラム目から記述します。
- @OPTIONS と翻訳オプションの間には、1 つ以上の空白が必要です。
- 各翻訳オプションの間は、1 つのコンマ(,)で区切る必要があります。 各翻訳オプションの間に空白を記述することはできません。
- 翻訳指示文は、翻訳単位の先頭に記述します。 翻訳指示文は複数行記述できます。 指定する翻訳オプションは、その翻訳指示文の翻訳単位にだけ適用されます。

以下に、翻訳指示文の記述列を示します。

例:

```
     @OPTIONS NOALPHAL,APOST
     IDENTIFICATION DIVISION.
```

- @OPTIONS 翻訳指示文の分離符として TAB 文字を用いることはできません。
- 翻訳オプションによっては、翻訳指示文に指定できないものもあります。
- 翻訳指示文に指定した翻訳オプションは、翻訳指示文が翻訳された後に有効になります。

  例 : /wc オプションに SRF(FIX)を指定した場合、翻訳指示文に SRF(VAR)を記述していたとしても、 その行は 72 カラム目までしか有効になりません。 翻訳オプションの文字列が 72 カラムを超える場合、翻訳オプションの区切りで分割し、 複数行の翻訳指示文で記述してください。

  cobolc の指定

```
cobolc /main:HELLO /wc:SRF(FIX) HELLO.cob
```

  ソースプログラムの記述

```
1     8                                                      72
+------+-------------------------------------------------------------------+--
      @OPTIONS SRF(VAR),BINARY(WORD),RCS(UTF8-UCS2),ALPHAL(WORD)       -+ ここまで
SRF(FIX)が有効：72 カラム目まで
      @OPTIONS CHECK(ALL)                                            -+
      IDENTIFICATION DIVISION.                                        -+
      PROGRAM-ID. HELLO.                                              |SRF(VAR)が有効
      ...                                                           |
```

フォームデザイナの利用

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

Visual Studio では、 Windows アプリケーション、 ASP.NET Web アプリケーション、 XML Web サービス、および Windows サービス を開発する場合、 フォームデザイナーを利用して、 ドラッグ&ドロップでコントロールをフォームにレイアウトし、 簡単にユーザインタフェースを設計することができます。

フォームデザイナーを利用して生成された COBOL ソースには、配置したコントロールを操作するためのクラス定義が含まれているため、利用者はコードエディターを使って、コントロールの手続きを記述するだけでアプリケーションが作成できます。

それぞれの開発の概要とフォームデザイナーの利用方法については、以下を参照してください。

- Windows アプリケーションの開発
- ASP.NET Web アプリケーションの開発
- XML Web サービスの開発

ここでは、フォームデザイナーの利用する場合の注意事項について説明します。

このセクションの内容

デザイナーを利用する場合の共通注意事項

デザイナーを使用する場合の共通注意事項について説明します。

Windows フォームデザイナーを利用する場合の注意事項

Windows フォームデザイナーを使用する場合の注意事項について説明します。

ASP.NET 関連のデザイナーを利用する場合の注意事項

ASP.NET 関連のデザイナーを使用する場合の注意事項について説明します。

NCOBU00073

デザイナーを利用する場合の共通注意事項

ここではデザイナーを利用する場合に共通な注意事項について説明します。

- デザイナーを利用して生成された COBOL ソースファイルに含まれる InitializeComponent メソッドはデザイナーが自動生成するものです。 InitializeComponent メソッド中のコメント行にはデザイナーの動作に必要な情報が記述されています。 コメントを含めて、このメソッドの内容を変更しないでください。
- デザイナーで編集される項目名は英字の大文字と小文字を区別する環境下で実行されることを想定しています。 そのため、これらの項目に関連付けられた COBOL ソースファイルを翻訳する場合は、 翻訳オプション NOALPHAL を指定する必要があります。
- デザイナーで編集される項目にはクラス定義以外を記述することはできません。 クラス定義が複数記述されている場合、デザイナーは最初のクラス定義を読み込みます。
- デザイナーで編集される項目に関連付けられた COBOL ソースファイルには、 以下の COPY 文を記述することができます。
  - メソッド定義の手続き部の中に COPY 文を記述することができます。
  - 取り込む登録集がデータ記述項（複数可）の完全な記述を含んでいる場合、作業場所節と連絡節の中に COPY 文を記述することができます。
- デザイナーで編集される項目に関連付けられた COBOL ソースファイルには、以下を記述することができません。
  - REPLACE 文
  - 以下の COPY 文
    - データ記述項の中に記述する COPY 文
    - データ記述項をまたがるような場所に記述する COPY 文
  - メソッド名段落の OF 指定
  - 手続き部見出しの RAISING 指定
  - 以下のすべての条件に該当するデータ記述項に対する VALUE 句
    - 作業場所節に記述されている、かつ
    - 01 レベルまたは 77 レベルである、かつ
    - .NET データ型に対応するデータ型である

    上の条件に該当するデータ記述項に初期値を設定したい場合は、手続きで初期値を設定するようにしてください。
  - 手続き部見出しから参照されない連絡節のレコード記述項
  - 埋込み SQL 文
  - 定数節
- コントロール名には COBOL の語として有効な名前を設定してください。

  デザイナーでコントロールやイベントハンドラの名前が COBOL の語として有効ではない場合、その名前に対して翻訳エラーが発生する場合があります。その場合はコントロールやイベントハンドラの名前を COBOL の語として有効になるように変更してください。特に、COBOL の語は 30 文字以下である必要があります。

また、デザイナーで「'<名前>'は有効な識別子ではありません」というエラーメッセージが表示された場合、以下の可能性が考えられます。 各原因に応じて対処してください。

- コントロール名が COBOL の語として不正です。コントロール名には COBOL の語として有効な名前を設定してください。
- デザイナーによって自動生成されたイベントハンドラ名が COBOL の語として不正です。以下のいずれかの対処を行ってください。
  - 自動生成されたイベントハンドラ名が COBOL の語として有効になるようにコントロール名を調整します。 特に、イベントハンドラ名が 30 文字以上である場合は、コントロール名を短くしてイベントハンドラ名が 30 文字以内になるようにします。
  - 有効な名前をもつイベントハンドラをあらかじめソース上に記述し、デザイナーを使ってそれをコントロールのイベントと関連付けます。

・ デザイナーが COBOL ソースファイルを生成する際に自動生成する名前は、 既に記述されているメソッド名、作業場所節や連絡節のデータ名、 リポジトリ段落で定義された名前、カスタム属性名とは重ならないように生成されます。 しかし、 それ以外の名前（ファイル名や特殊名段落で定義されたカスタム属性名以外の名前など）との重複はチェックされません。 また、COPY 文で取り込んだ登録集に記述されている名前との重複もチェックされません。 名前が重複して翻訳エラーが発生した場合は、重複した名前を修正してください。

NCOBU00074

Windows フォームデザイナーを利用する場合の注意事項

デザイナーを利用する場合の共通の注意事項については、デザイナーを利用する場合の共通注意事項を参照してください。

ここでは、Windows フォームデザイナーを利用する場合の注意事項について説明します。

Windows フォームに関連して、デザイナーで編集できるものには以下があります。ここでは、これらを「Windows フォーム関連項目」と呼びます。

- Windows フォーム(拡張子.cob)
- ユーザコントロール(拡張子.cob)
- カスタムコントロール(拡張子.cob)
- Windows サービス(拡張子.cob)

前提知識

Windows フォームとは

Windows フォームは、.NET Framework を利用した Windows クライアントアプリケーションを開発するためのフレームワークです。Windows アプリケーションの土台となる Form クラスと、ユーザインタフェースを作成するための複数のコントロールから画面のレイアウトを作成します。

作成された Windows フォームは、COBOL ソースプログラム(拡張子(.cob))として保存されます。

Windows フォームコントロール(ユーザコントロール、カスタムコントロール)

Windows フォームには、すぐに使用できる多くのコントロールが用意されています。 また、開発者が独自にコントロールを開発するための仕組みも用意されています。 これらの仕組みを利用して、既存の複数のコントロールを組み合わせたり、拡張したり、 またカスタムコントロールを作成したりできます。 これら独自に開発したコントロールを、カスタムコントロールまたはユーザコントロールといいます。 Windows フォームコントロールの詳細については、 .NET Framework のドキュメントの.NET Framework を使用したカスタム Windows フォーム コントロールの開発を参照してください。

作成されたユーザコントロールおよびカスタムコントロールは、 COBOL ソースプログラム(拡張子(.cob))として保存されます。

Windows サービス

Windows サービスは、ユーザインタフェースがなく Windows の起動時に連動して長い時間実行させることができるアプリケーションであり、サーバ上で動作するアプリケーションに適しています。

Windows サービスアプリケーションについての詳細は Visual Studio のドキュメントの Windows サービス アプリケーションの開発を参照してください。

作成された Windows サービスは、 COBOL ソースプログラム(拡張子(.cob))として保存されます。

注意事項

- 「既存の項目の追加」で Windows 関連項目を追加した場合、 その項目に関連付けられた.resx ファイルはその項目と一緒に追加されません。 デザイナーが.resx ファイルを必要とした時点で関連付けが行なわれます。
- NetCOBOL for .NET では、コントロールの「Modifiers」プロパティの値で有効なのは以下のものです。

Modifiers プロパティの値	対応する COBOL のアクセス指定
Public	PUBLIC
Family	PROTECTED
Private	PRIVATE

それ以外の値を設定した場合、 その値は COBOL ソースプログラムを生成する際に「Public」または「Family」に変更されます。

- ユーザコントロールまたはカスタムコントロールを作成した場合、 それらを利用するには、 まず以下の手順でツールボックスに作成したコントロールを登録します。
  - [ツール]メニューを選択し、[ツールボックスアイテムの選択]をクリックします。
  - [ツールボックスアイテムの選択]ダイアログボックスが表示されます。[参照]ボタンから作成したコントロールを含むアセンブリを選択し、[OK]をクリックします。

  登録が完了すると、ツールボックスから登録したコントロールが利用できます。
- ユーザコントロールおよびカスタムコントロールを Windows フォームに貼り付けた場合、 そのコントロールが含まれるアセンブリファイルは使用中になります。 アセンブリファイルが使用中のままコントロールを再度ビルドしようとすると、 出力アセンブリファイルを上書きできないためエラーが発生します。 コントロールを再度ビルドする場合は、 Visual Studio を再起動してください。

NCOBU00075

ASP.NET 関連のデザイナーを利用する場合の注意事項

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。

デザイナーを利用する場合の共通の注意事項については、デザイナーを利用する場合の共通注意事項を参照してください。

ここでは、Web フォームデザイナーを利用する場合の注意事項について説明します。

Web フォームデザイナーで編集できるものには以下があります。ここでは、これらを「ASP.NET 関連項目」と呼びます。

- Web フォームページ(拡張子.aspx)
- XML Web サービス(拡張子.asmx)
- Global.asax ファイル(拡張子.asax)
- Web ユーザーコントロール(拡張子.ascx)

前提知識

Web フォームページ

Web フォームページは、.NET Framework クラス ライブラリの Page クラスから派生したオブジェクトで、 ASP.NET Page Framework 上に構築されます。 このページオブジェクト内に、 Button や TextBox などの Web サーバコントロールオブジェクトが格納されます。

Visual Studio で Web フォーム ページを作成すると、ユーザインタフェース要素を保持する.aspx ファイル(コントロール要素を含んだ HTML テキストファイル)が作成されます。

XML Web サービス

ASP.NET を使用することで、XML Web サービスを作成できます。 ASP.NET を使用して作られた XML Web サービスは、 その実装が XML や SOAP といった標準に基づいているため、 任意のプラットフォーム上のクライアントから利用することができます。 ASP.NET を使用して作成された XML Web サービスは、ASP.NET Web サービスとも呼ばれます。

Visual Studio で XML Web サービスを作成すると、 XML Web サービスの URL アドレスをサービスの手続きと関連付ける.asmx ファイルが作成されます。

Global.asax ファイル

Web アプリケーションで使用され、 アプリケーションレベルのイベント(Session_OnStart や Application_OnStart など)に応答するコードを記述します。

Web ユーザーコントロール

利用者が Web フォームのページ用に作成したコントロールで、 他の Web フォームページに埋め込んで使用することができます。

注意事項

- Visual Studio 2017 のデザイナーは、 部分型形式の COBOL ソースを生成します。 部分型の機能や注意事項に関しては、「部分型の使用」を参照してください。
- 部分型形式の COBOL ソース(分離コード)には翻訳指示文(@OPTIONS)は記述できません。
- ASP.NET 関連項目はプロジェクトフォルダになければなりません。
- COBOL では、ASP.NET 関連項目中に直接ソースを埋め込む形式の ASP.NET 関連項目を直接編集することはできません。
- ASP.NET 関連項目に関連付けられている COBOL ソースファイルに含まれるクラスの外部名を変更した場合、 その項目を XML エディターまたは HTML エディターで開いて ASP.NET ディレクティブの「Inherits」属性の値を新しいクラス名に置き換えなければなりません。

  ASP.NET ディレクティブは一般にページの先頭にあり、 「<%@」で始まり「%>」で終わるブロックです。 ASP.NET ディレクティブの詳細については、.NET Framework のドキュメントの ASP.NET ページの構文および XML Web サービスのディレクティブを参照してください。
- Global.asax ファイルに関連付けられる.resx ファイルは、 デザイナーが.resx ファイルを必要とした時点で自動的に生成されます。
- Web ユーザーコントロール（.ascx ファイル）を作成した後、 それを Web フォームページに貼り付けるには、 Web フォームページの HTML を編集する必要があります。 詳細については、Visual Studio のドキュメントの方法 : Web ページに ASP.NET ユーザー コントロールを組み込むを参照してください。

NCOBU00814

部分型の使用

部分型とは

部分型は同一翻訳単位の内容を二箇所以上に分散して記述することを可能にする機能です。 Visual Studio 2017 のデザイナーは、部分型形式の COBOL ソースを生成します。

コンパイラは部分型形式で記述された翻訳単位を一つの翻訳単位に編集し、 それを翻訳します。 例えば、以下のソースは、

```
CLASS-ID. PARTIAL-TYPE-SAMPLE AS "PartialTypeSample" IS PARTIAL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-1 OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION.

METHOD-ID. NEW.
PROCEDURE DIVISION.
   SET WK-1 TO N"初期値".
END METHOD NEW.

END OBJECT.
END CLASS PARTIAL-TYPE-SAMPLE.


CLASS-ID. PARTIAL-TYPE-SAMPLE AS "PartialTypeSample" IS PARTIAL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-ENVIRONMENT AS "System.Environment"
   PROPERTY PROP-MACHINENAME AS "MachineName"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-2  PIC N(64) VALUE N"初期値".
PROCEDURE DIVISION.

METHOD-ID. WRITE-WK-2.
PROCEDURE DIVISION.
   DISPLAY WK-2.
END METHOD WRITE-WK-2.

END OBJECT.
END CLASS PARTIAL-TYPE-SAMPLE.
```

次のように編集された後に翻訳されます。

```
CLASS-ID. PARTIAL-TYPE-SAMPLE AS "PartialTypeSample".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-ENVIRONMENT AS "System.Environment"
   PROPERTY PROP-MACHINENAME AS "MachineName"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-1 OBJECT REFERENCE CLASS-STRING.
01 WK-2  PIC N(64) N"初期値".
```

```
       PROCEDURE DIVISION.

       METHOD-ID. NEW.
       PROCEDURE DIVISION.
          SET WK-1 TO N"初期値".
       END METHOD NEW.

       METHOD-ID. WRITE-WK-2.
       PROCEDURE DIVISION.
          DISPLAY WK-2.
       END METHOD WRITE-WK-2.

       END OBJECT.
       END CLASS PARTIAL-TYPE-SAMPLE.
```

部分型形式で記述された翻訳単位の編集

部分型形式で記述された翻訳単位の編集は以下のように行われます。

- コンパイラはソースの翻訳に先立って、 拡張子が.cobx であるソースの中身を走査し、 部分型形式で記述された翻訳単位の有無をチェックします。 部分型形式で記述された翻訳単位があれば、 それを元のソースから抜き出し、 同一の外部名をもつ部分型形式の翻訳単位をそれぞれ一つの翻訳単位へと編集し、それらを翻訳対象とします。
- 部分型形式の翻訳単位の編集は、 後述する「累積的な内容」を累積させていくことによって行います。これに対し、「非累積的な内容」は各部分型形式の翻訳単位で一致していなければなりません。
- 累積的な内容には、以下のものがあります。
    - クラス定義に対するカスタムアトリビュート指定
    - クラス定義の特殊名段落の内容
    - クラス定義のリポジトリ段落の内容
    - オブジェクト定義の実装インタフェース指定
    - STATIC 定義またはオブジェクト定義のデータ部の内容
    - STATIC 定義またはオブジェクト定義に含まれるメソッド定義
- 非累積的な内容には、以下のものがあります。
    - 親クラス指定
- 非累積的な内容は、 それが記述されている場合は同一の翻訳単位へ編集されるすべての部分型形式の翻訳単位で一致していなければなりません。 ただし、少なくとも一つの部分型形式の翻訳単位である非累積的な内容が記述されている場合、 その非累積的な内容の記述を他の部分型形式の翻訳単位で省略することができます。 これにより、非累積的な内容の変更を一箇所の変更で済ませることができます。

部分型利用上の注意点

部分型を利用する上での注意事項は以下の通りです。

- 部分型機能はデザイナーからの利用を想定して提供されています。 通常のソースを記述するために部分型機能を利用しないでください。
- 部分型形式で記述できる翻訳単位はクラス定義とインタフェース定義です。
- 部分型形式の翻訳単位を含むソースファイルの拡張子は、 cobx でなければなりません。 ファイルの拡張子が cobx の場合にのみ、コンパイラは部分型形式の翻訳単位の有無をチェックします。
- 各部分型形式の翻訳単位は、PARTIAL 句を除いた場合に、 一つの翻訳単位として翻訳可能なものでなければなりません。
- 部分型形式の翻訳単位の中に REPLACE 文を記述することはできません。

- 同一の翻訳単位へと編集されるすべての部分型形式の翻訳単位中の以下の名前は、 すべて一致していなければなりません。
  - 翻訳定義の内部名（クラス名）
  - クラス名段落に INHERITS 句が記述されている場合、INHERITS 句に指定された親クラス名
- 同一の翻訳単位へと編集されるすべての部分型形式の翻訳単位には、 すべて同一の翻訳オプションが指定されていなければなりません。
- 以下の場所以外に COPY 文を記述することはできません。
  - メソッド定義の手続き部
  - 取り込む登録集がデータ記述項（複数可）の完全な記述を含んでいる場合、作業場所節または連絡節

ビルド

ここでいうビルドとは、プログラムを構成する各モジュールをコンパイルして、 中間コードファイルを生成し、さらにリンクして、 実行可能ファイルを生成することをいいます。

ここでは、Visual Studio IDE を使用したビルド方法およびコマンドラインからのビルド方法について説明します。また、翻訳オプションの設定方法についても説明します。

このセクションの内容

NCOBU00077

オプションの種類

ここでは、コンパイラオプションと翻訳オプションの違いと、それぞれのオプションの指定方法について説明します。また、応答ファイルの使用方法についても説明します。

このセクションの内容

コンパイラオプションと翻訳オプション

NetCOBOL for .NET では、コンパイラの動作を調整するために、多くのコンパイラオプションと翻訳オプションが用意されています。

コンパイラオプションは、NetCOBOL for .NET で利用できる新しいカテゴリのオプションで、特に .NET プラットフォームのために導入されています。

翻訳オプションは、従来の COBOL の翻訳オプションに相当します。

翻訳オプションを指定する場合は、コンパイラオプションの一部として、/wc に指定します。たとえば、翻訳オプション SRF(FREE,FREE)を指定する場合、以下のように指定します。

```
/wc:"SRF(FREE,FREE)"
```

コンパイラオプションの種類については、コンパイラオプション - 詳細を参照してください。 また、翻訳オプションの種類については、翻訳オプション - 詳細を参照してください。

NCOBU00079

応答ファイルの使用

NetCOBOL for .NET では、コンパイラオプションを応答ファイル(.rsp)と呼ばれるテキストファイルに保存することができます。

応答ファイルを使用する場合は、@オプションを使って指定します。

応答ファイルの利点

応答ファイルを使用すると、 コマンドラインや Visual Studio プロジェクトで指定するオプション文字列を減らすことができます。 また、 異なるプログラム間で共通なコンパイラオプションをまとめてグループ化するのに便利です。

応答ファイルの内容

応答ファイルには、オプションの並びを記述します。 「#」で始まる行はコメント行として扱われます。以下は、応答ファイルの例です。

```
# build the first output file
/wc:ALPHAL(AUTO) /reference:System.Windows.Forms.DLL,System.Drawing.DLL
```

応答ファイルの使用方法

応答ファイルを指定するには、cobolc コマンドに@オプションを指定します。 以下は、cobolc コマンドに指定する例です。 ここで応答ファイルは、option1.rsp です。

```
cobolc @option1.rsp /target:winexe /main:HELLO,MAIN Hellowin.cob
```

応答ファイルの指定の詳細については、@ (コンパイラのコマンド引数を応答ファイル(.rsp)で指定)を参照してください。

コンパイラオプションの選択

NetCOBOL for .NET では、コンパイラの動作を調整するために、多くのコンパイラオプションと翻訳オプションを用意しています。

アプリケーションの動作条件はさまざまなので、目的に合った最適なオプションを選択することが大切です。

NetCOBOL for .NET を使用し始めたばかりで、簡単なアプリケーションを作成中の場合は、以下のコンパイラオプションを覚えれば十分でしょう。

- /main (アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定)
- /reference (参照するメタデータを含むアセンブリファイルを指定)
- /target (出力ファイルの種類を指定)
- /out (出力ファイルのファイル名を指定)

NCOBU00081

NetCOBOL for .NET プロジェクトのビルド

ここでは、NetCOBOL for .NET プロジェクトのビルド方法について説明します。

このセクションの内容

翻訳オプションの設定

翻訳オプションの設定方法について説明します。

プロジェクトのビルド

NetCOBOL for .NET プロジェクトのビルド方法について説明します。

MSBuild によるプロジェクトのビルド

NetCOBOL for .NET プロジェクトを MSBuild を使用してビルドする方法について説明します。

翻訳オプションの設定

Visual Studio IDE を使用してビルドする場合、コンパイラオプションは、以下の手順で指定します。ここでは、翻訳オプションとコンパイラオプションの両方を指して、 コンパイラオプションといいます。

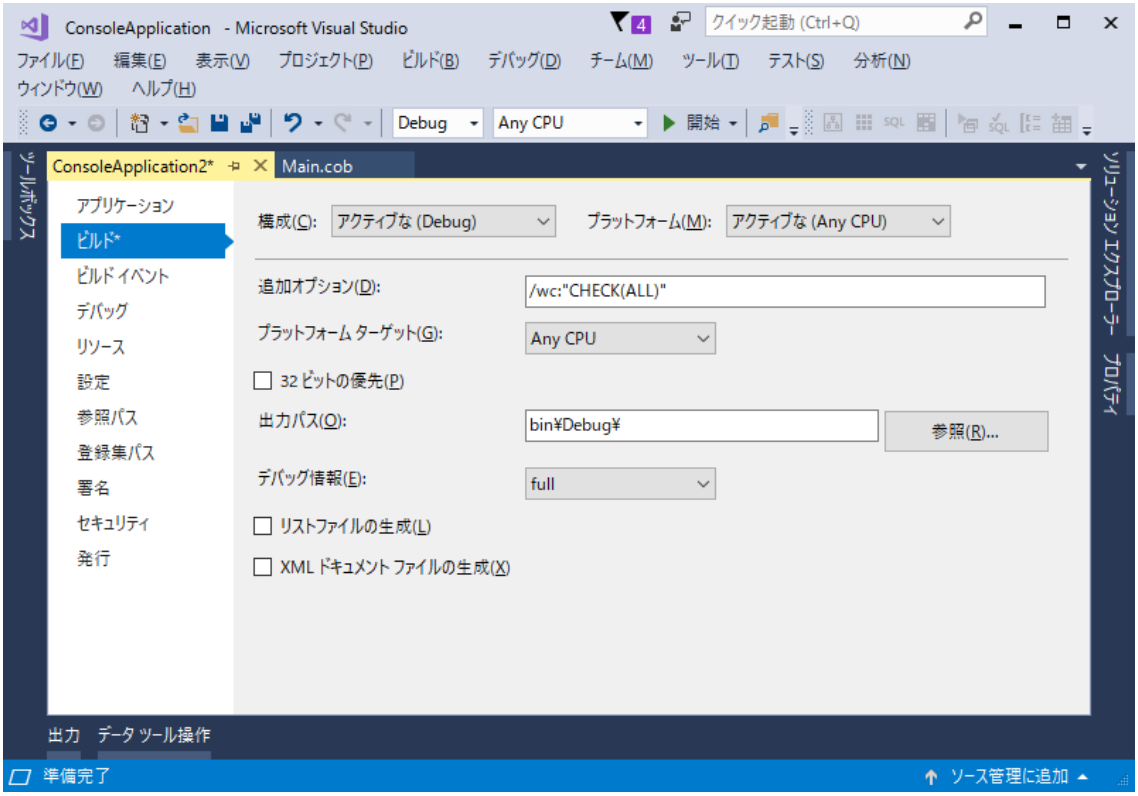
1. ソリューションエクスプローラーのプロジェクト名を右クリックし、コンテキストメニューから[プロパティ]を選択します。

   ソリューションエクスプローラーが表示されない場合は、[表示]メニューを選択し、 [ソリューションエクスプローラー]をクリックすることで表示されます。

   これにより、プロジェクトの[プロパティページ]ダイアログボックスが表示されます。

2. [プロパティページ]ダイアログボックスの左ペインにある[ビルド]をクリックします。

3. [追加オプション]プロパティに、コンパイラオプションを指定します。



翻訳リストファイルを作成する場合

[リストファイルの生成]チェックボックスをチェックします。

翻訳リストファイルに出力される各種リストには、以下があります。

- オプション情報リスト
- 診断メッセージリスト
- 翻訳単位統計情報リスト
- 相互参照リスト

それぞれのリストを出力するための翻訳オプションについては、COBOL が使用するファイルの「翻訳リストファイル」を参照してください。

XML ドキュメントファイルを作成する場合

[XML ドキュメント ファイルの生成]チェックボックスをチェックします。

ソースコードから XML ドキュメントコメントを検索して、XML ドキュメントファイルを生成します。 XML ドキュメントコメントについては、XML ドキュメントコメントを参照してください。

登録集のパスを設定する場合

登録集パスの設定を参照してください。

作成するアプリケーションが以下の場合、プログラムに記述したクラス名などの大文字小文字を意識して、オプションを指定する必要があります。

- COBOL 以外の言語で記述されたアプリケーションを呼び出す場合
- COBOL 以外の言語で記述されたアプリケーションから呼び出される場合

オプションの注意事項については、翻訳オプション ALPHAL に関する注意事項を参照してください。

関連トピックス

コンパイラオプション - 詳細

コンパイラオプションの一覧を参照できます。

翻訳オプション - 詳細

翻訳オプションの機能ごとの一覧を参照できます。

プロジェクトのビルド

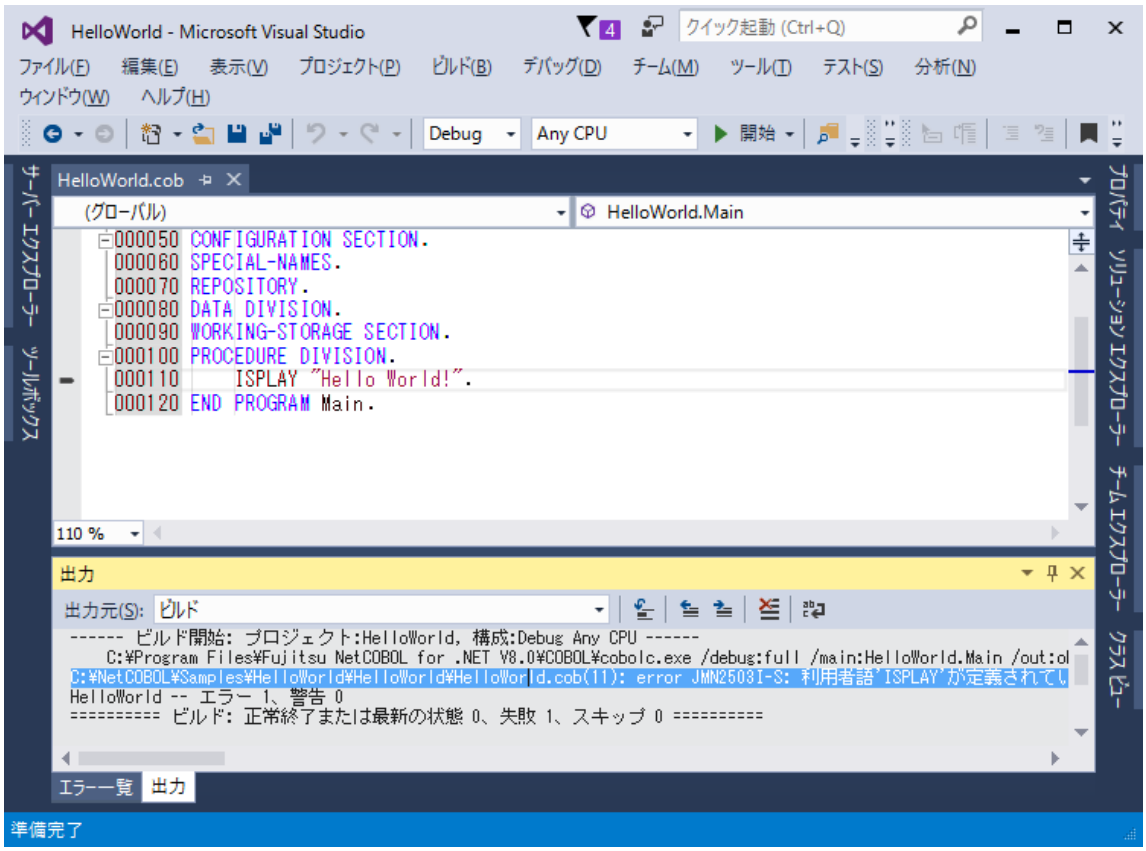
プロジェクトのビルドは、以下の手順で行います。

1. ソリューションエクスプローラーのプロジェクト名を右クリックし、コンテキストメニューから[ビルド]、または[リビルド]を選択します。

   または、 [ビルド]メニューから[ソリューションのビルド]、または[ソリューションのリビルド]を選択します。

   - 「ビルド」は、前回のビルドに失敗したプログラム、または前回のビルドの後で変更したプログラムがある場合だけビルドを行います。
   - 「リビルド」は、すべてのプログラムのビルドを強制的に行います。

[出力]ウィンドウでは、発生したビルドエラーはすべて表示します。エラーが発生した場合、エラーメッセージをダブルクリックすることによって、それに対応するソースコードがエディターで表示されます。



また、「エラー一覧」ウィンドウにも、エラーを修正する時にそのエラーを照合できるように、すべてのビルドエラーの一覧を含んでいます。

注意

登録集パスに相対パスが設定されていると、登録集ファイルの中でエラーが発生した場合に、タグジャンプが正しく行われません。 この場合は、登録集パスにフルパスを設定してください。

NCOBU00815

MSBuild によるプロジェクトのビルド

NetCOBOL for .NET のプロジェクトファイル(.cobproj)は、MSBuild が扱える形式で作成されています。NetCOBOL for .NET のコマンドプロンプトを起動し、コマンドラインから以下のコマンドを実行してビルドします。 また、ソリューションに含まれるすべてのプロジェクトをビルドしたい場合には、ソリューションファイル(.sln)を指定することもできます。

```
MSBuild Application1.cobproj
```

NetCOBOL for .NET のプロジェクトでは、MSBuild の主なターゲットとして、以下のターゲットを用意しています。

- Build （デフォルト)
- Rebuild
- Clean

たとえば、プロジェクトをリビルドするには、以下のように指定します。 デフォルトでは、Build が指定されたとみなします。

```
MSBuild /target:Rebuild Application1.cobproj
```

また、プロジェクトが複数の構成（Debug/Release）やプラットフォーム(AnyCPU/x86/x64)を持っているなら、以下のように指定することで、各構成や各プラットフォームごとにビルドすることができます。 デフォルトでは、Debug および AnyCPU が指定されたとみなします。

```
MSBuild /property:Configuration="Release" /property:Platform="x86" Application1.cobproj
```

MSBuild の詳細については、 Visual Studio のドキュメントの MSBuild を参照してください。 NetCOBOL for .NET のコマンドプロンプトを起動する方法については、コマンドプロンプトの起動を参照してください。

- プロジェクトのビルドイベントなどで、マクロ(COBOLInstallDir など)を使用するには、環境変数または MSBuild のプロパティとして設定されている必要があります。 MSBuild の実行には、NetCOBOL for .NET のコマンドプロンプトを使用してください。

ASP.NET Web サイトのビルド

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

ここでは、ASP.NET Web サイトのビルド方法について説明します。

このセクションの内容

コンパイラオプションの設定

コンパイラオプションの設定方法について説明します。

ASP.NET Web サイトのビルド

ASP.NET Web サイトのビルド方法について説明します。

NCOBU01027

コンパイラオプションの設定

ASP.NET Web サイトでは、 コンパイラに対するコンパイラオプションは ASP.NET の構成ファイル（Web.config）中に記述します。

ASP.NET Web サイトの Web 構成ファイル(Web.config)中の NetCOBOL for .NET コンパイラを表す'compiler'要素の'compilerOptions'属性にコンパイラオプションを記述します。 'compiler'要素の詳細は、.NET Framework のドキュメントの compilation の compilers 要素 (ASP.NET 設定スキーマ)を参照してください。

以下は、コンパイラオプションとして翻訳オプション SCS を指定している例です。

```
<compilers>
  <compiler
    extension=".cobx;.cob"
    language="Fujitsu.COBOL;NetCOBOL;cobol"
    type=" Fujitsu.COBOL.CodeDom, Version=8.0.124.0, Culture=neutral,
           PublicKeyToken=fac0fe3cab973246, processorArchitecture=MSIL"
    compilerOptions="/wc:SCS(ACP)"
  />
</compilers>
```

ASP.NET Web サイトのビルド

プロジェクトのビルドは、以下のいずれかの手順で行います。

- ソリューションエクスプローラーの ASP.NET Web サイトを右クリックし、コンテキストメニューから[Web サイトのビルド]を選択します。
- [ビルド]メニューから[Web サイトのビルド]、または[Web サイトのリビルド]を選択します。

- 「ビルド」は、前回のビルドに失敗したプログラム、または前回のビルドの後で変更したプログラムがある場合だけビルドを行います。
- 「リビルド」は、すべてのプログラムのビルドを強制的に行います。

NCOBU00084

コマンドラインからのビルド

ここでは、コマンドラインからのビルド方法について説明します。

このセクションの内容

コマンドプロンプトの起動

コマンドラインインタフェースを使うには、 NetCOBOL for .NET コマンドプロンプトを起動する必要があります。

このコマンドプロンプトは NetCOBOL for .NET の動作環境が適切に設定されているため、 .NET コンポーネントを確実に利用することができます。以下の手順で、起動します。

1. コマンドプロンプトを起動します。

   Windows 7

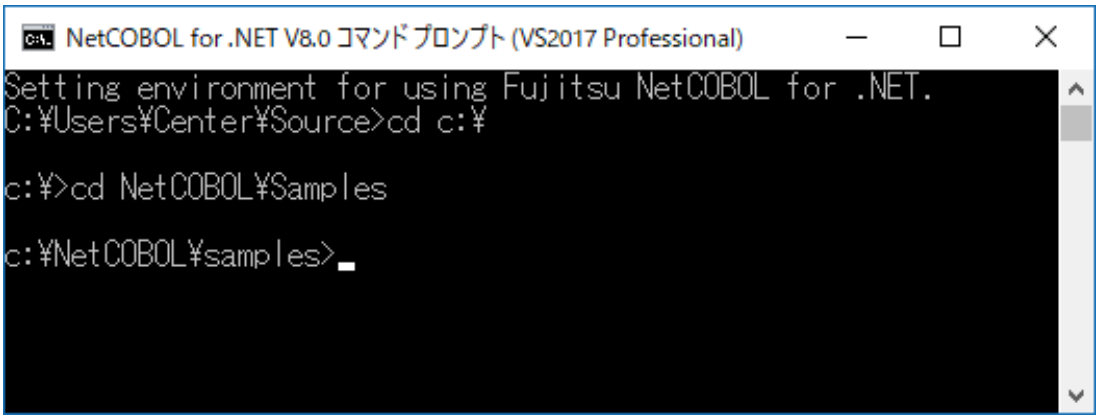
   コマンドプロンプトは、[スタート] - [すべてのプログラム] - お使いの NetCOBOL for .NET 製品名 - [NetCOBOL]-[NetCOBOL for .NET コマンドプロンプト]を選択すると表示されます。

   Windows 7 以外

   コマンドプロンプトは、[スタート] - アプリ一覧(*) - お使いの NetCOBOL for .NET 製品名 - [NetCOBOL for .NET コマンドプロンプト]を選択すると表示されます。

   * : Windows 8.1 および Windows Server 2012 R2 の場合、[スタート]画面 - [↓] - [アプリ]を操作をすることで同様の画面になります。

2. 開発作業をするフォルダに移動します。



参考

NetCOBOL for .NET に限らず、 Visual Studio 製品がインストールされている場合は、 Visual Studio のコマンドプロンプトも利用できます。

Visual Studio コマンドプロンプトは、 以下の手順で起動することができます。

1. コマンドプロンプトを起動します。

   [スタート] - アプリ一覧(*) - [Visual Studio 2017] > [開発者コマンド プロンプト for VS2017] をクリックする。

2. 開発作業をするフォルダに移動します。

NCOBU00086

cobolc コマンド

NetCOBOL for .NET では、cobolc コマンドを利用してビルドすることができます。

cobolc コマンドは、COBOL ソースファイルを翻訳し、実行可能ファイルを生成します。また、必要に応じてオプションを指定することができます。

cobolc コマンドは、PATH 環境変数などが適切に設定されている環境で使用してください。コマンドプロンプトの起動によって、適切に環境設定されているコマンドプロンプトを簡単に起動することができます。

コマンド構文

コマンドラインから、cobolc コマンドを使用する場合のコマンド構文は以下のとおりです。

```
cobolc [オプション] ソースファイル
```

オプション	スラッシュ(/)またはハイフン(-)から始まるコンパイラオプション(複数可)の並び。
ソースファイル	翻訳するソースファイル名(複数可)を指定します。ソースファイル名はオプションの前に指定しても後に指定してもかまいません。

作成するアプリケーションが以下の場合、プログラムに記述したクラス名などの大文字小文字を意識して、オプションを指定する必要があります。オプションの注意事項については、翻訳オプション ALPHAL に関する注意事項を参照してください。

- COBOL 以外の言語で記述されたアプリケーションを呼び出す場合
- COBOL 以外の言語で記述されたアプリケーションから呼び出される場合

また、コマンドラインの文字コードは ACP(Shift JIS)でなければなりません。

cobolc コマンドの使用例

cobolc コマンドの使用例について、説明します。

例 1: コンソールアプリケーション(.EXE)の作成

単純なコンソールアプリケーションプログラムを記述した COBOL ソースファイル(Hello.cob)を翻訳し、実行可能ファイル(.exe)を作成する場合、以下のように指定します。

```
cobolc /main:HELLO Hello.cob
```

ビルド終了後、実行可能ファイル(Hello.exe)が出力されます。

/main には、アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定します。Hello.cob では、プログラム名として"HELLO"を記述しているため、/main:HELLO となります。

実行可能ファイル(.exe)を作成する場合、エントリポイントとなるプログラムがひとつしか含まれない場合でも、必ず/main を指定する必要があります。

[補足]: /target を省略している場合、/target:exe が指定されているとみなされます。

例 2: コンソールアプリケーション(.DLL)の作成

単純なコンソールアプリケーションプログラムを記述した COBOL ソースファイル(Hello.cob)を翻訳し、ダイナミックリンクライブラリ(.dll)を作成する場合、以下のよう指定します。

```
cobolc /target:library Hello.cob
```

/target:library は、ライブラリを作成する指定です。

例 3: 複数のソースファイルからなるコンソールアプリケーションの作成

複数の COBOL ソースファイルを翻訳し、ひとつの実行可能ファイル（.exe)を作成する場合、以下のように指定します。ここでは、3 つの COBOL ソースファイル(mymain.cob、subprog1.cob、subprog2.cob)を指定し、エントリポイントを mymain.cob に含まれるプログラム名"MYMAIN"としています。

```
cobolc /main:MYMAIN mymain.cob subprog1.cob subprog2.cob
```

例 4: Windows アプリケーションの作成

単純な Windows アプリケーションプログラムを記述した COBOL ソースファイル(Hellowin.cob)を翻訳し、Windows アプリケーション(.exe)を作成する場合、以下のよう指定します。

```
cobolc /main:HELLO,MAIN /target:winexe Hellowin.cob
/reference:System.Windows.Forms.DLL,System.Drawing.DLL
```

/main:HELLO,MAIN は、HELLO クラスの MAIN メソッドをエントリポイントとすることを指示しています。クラス名とメソッド名をカンマ(,)で区切っていることに注意してください。

/target:winexe は、Windows アプリケーション(.exe)を作成する指定です。

/reference は、参照するアセンブリファイル(.dll)の指定です。

例 5: 複数のソースファイルからなる Windows アプリケーションの作成

複数の COBOL ソースファイルを翻訳し、ひとつの Windows アプリケーション（.exe)を作成する場合、以下のように指定します。ここでは、3 つの COBOL ソースファイル(walkthedog.cob、reminder.cob、about.cob)を指定し、エントリポイントを walkthedog.cob に含まれる WALK クラスの MAIN メソッドとしています。

```
cobolc /main:WALK,MAIN /target:winexe walkthedog.cob reminder.cob about.cob
/reference:System.Windows.Forms.DLL,System.Drawing.DLL,System.DLL
```

関連トピックス

コンパイラオプション - 詳細

コンパイラオプションの一覧を参照できます。

翻訳オプション - 詳細

翻訳オプションの機能ごとの一覧を参照できます。

NMAKE ユーティリティ

Visual Studio には、 "Makefile"(拡張子無し)というファイルを入力して、 ビルドを行う"NMAKE"というユーティリティが付属しています。 NMAKE を使用することで、ファイルの更新状況に応じてコンパイルを行うなど、より高度なビルド操作を行うことができます。

NMAKE は、Visual Studio 2017 のワークロード「C++ によるデスクトップ開発」がインストールされている場合に利用することができます。

"Makefile"および"NMAKE"の詳細については、 Visual Studio のドキュメントの NMAKE リファレンスを参照してください。

Makefile の例

NetCOBOL for .NET の「HelloWin」サンプルアプリケーションの Makefile は以下のようになっています。

```
all: HelloWin.exe         # [1]
HelloWin.exe: HelloWin.cob  # [2]
      cobolc -reference:System.Windows.Forms.DLL,System.Drawing.DLL -Main:HELLO,MAIN
HelloWin.cob
clean:                    # [3]
      -del /q *.exe *.pdb
```

このメイクファイルの内容を簡単に説明します。

[1]: この Makefile ファイルの目的（all）は、HelloWin.exe を作成することです。

[2]: HelloWin.exe はコマンド cobolc を使用して HelloWin.cob から生成されます。

[3]: clean ターゲットを指定して NMAKE を実行した場合、該当するファイルを削除します。

このように、NMAKE を用いてそれぞれのコンポーネントのビルド方法を指示することで、巨大なプロジェクトもビルドすることができます。

NCOBU00816

MSBuild

.NET Framework には、XML 形式で記述されたファイルを入力して、ビルドを行う"MSBuild"というユーティリティが付属しています。 また、NetCOBOL for .NET では、MSBuild を使用して COBOL コンパイラを実行するためのタスクを提供しています。 MSBuild を使用することで、ファイルの更新状況に応じてコンパイルを行うなど、より高度なビルド操作を行うことができます。

MSBuild の詳細については、Visual Studio のドキュメントの MSBuild を参照してください。また、NetCOBOL for .NET が提供するタスクについては、MSBuild タスク リファレンス (Fujitsu.COBOL.Build)を参照してください。

MSBuild の例

HellowWorld.cob をコンパイルして HelloWorld.exe を出力する場合は、以下のように記述することができます。

```
<Project ToolsVersion="4.0" DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

    <Import Project="$(MSBuildExtensionsPath)¥Fujitsu¥NetCOBOL
for .NET¥V8.0¥Fujitsu.COBOL.Build.Tasks" /> <!-- [1] -->

    <PropertyGroup>
        <AppName>HelloWorld</AppName>
    </PropertyGroup>

    <ItemGroup>
        <Compile Include = "HelloWorld.cob" />
    </ItemGroup>

    <Target Name="Build" Inputs="@(Compile)" Outputs="$(AppName).exe">    <!-- [2] -->
        <Cobolc
            Sources="@(Compile)"
            MainEntryPoint="MAIN"
            OutputAssembly="$(AppName).exe"
            />
    </Target>

    <Target Name="Clean">                                                 <!-- [3] -->
        <Delete Files="$(AppName).exe" />
    </Target>

</Project>
```

このファイルの内容を簡単に説明します。

[1]: NetCOBOL for .NET が提供するタスクをインポートします。Cobolc タスクを使用するにはこの記述が必要となります。

[2]: Cobolc タスクを使用して HelloWorld.cob から HelloWorld.exe を生成します。

[3]: Clean ターゲットを指定して MSBuild を実行した場合、Delete タスクを使用して該当するファイルを削除します。

アプリケーションの実行

ここでは、NetCOBOL for .NET アプリケーションの実行方法と、NetCOBOL for .NET アプリケーションを実行させるために必要となる情報の内容と設定方法について説明します。

このセクションの内容

NCOBU00089

実行環境の設定

NetCOBOL for .NET アプリケーションの特定の動作や、プログラム内で使用するデータファイルがディスク上のどのファイルにマップされるかという情報を実行時に指定できます。

これらの NetCOBOL for .NET アプリケーションを実行するために必要となる情報を実行環境情報といいます。

実行環境情報の種類

実行環境情報は、以下の 2 種類に分けられます。

情報の種類	内容
環境変数情報	ファイル名のマッピング、デバッグ方法、メッセージ出力などを指定するための情報です。
エントリ情報	副プログラムの配置や DLL ファイル内でのエントリポイントを指示します。

実行環境情報の設定

実行環境情報を設定するには、以下の方法があります。NetCOBOL for .NET では、アプリケーション構成ファイルに設定する方法を推奨しています。

情報の種類	設定方法		
環境変数情報	アプリケーション構成ファイル	実行用の初期化ファイル	SET コマンドでの設定
			OS のユーザ環境変数に設定
エントリ情報		エントリ情報ファイル	なし

各情報の詳細については、それぞれの説明を参照してください。

環境変数情報

環境変数情報は、ファイル名のマッピング、デバッグ方法、メッセージ出力などを指定するための情報です。

環境変数情報の内容

NetCOBOL for .NET には目的に応じてさまざまな種類の環境変数情報が用意されています。 内容の詳細については、実行環境変数 - 詳細を参照してください。

環境変数情報の設定方法

環境変数情報を設定するには、以下の方法があります。最適な設定方法は、COBOL の実行環境状況によって異なります。

- アプリケーション構成ファイルを使用する
- 実行用の初期化ファイルを使用する
- SET コマンドでの設定
- OS のユーザ環境変数に設定

NetCOBOL for .NET では、アプリケーション構成ファイルを使用する設定を推奨しています。

各設定方法の詳細については、それぞれの説明を参照してください。

アプリケーション構成ファイルに指定された実行環境情報は、COBOL プログラムの実行環境開設時に取り込まれるため、実行性能に影響します。 したがって、実行するプログラムで共通な情報は、プログラムの起動前にバッチファイルなどでユーザの環境変数に設定することをお勧めします。アプリケーション構成ファイルには、実行するプログラムに固有な情報だけを設定してください。

環境変数情報の優先順位

環境変数情報の優先順位を以下に示します。

1. アプリケーション構成ファイル、もしくは、実行用の初期化ファイルの値(注 1)
2. SET コマンドで設定した情報の値
3. OS のユーザ環境変数に設定した情報の値

注 1:

以下の順序で検索し、最初に見つかったファイルの情報だけが有効になります。

1. コマンドラインの/CBR に指定した実行用の初期化ファイル
2. アプリケーション構成ファイルの<environmentSettings>要素
3. アプリケーション構成ファイルの<appSettings>要素の@CBR_CBRFILE に指定した実行用の初期化ファイル
4. AppDomain の BaseDirectory 内の"COBOL85.CBR" ファイル
5. システム環境変数@CBR_CBRFILE に指定した実行用の初期化ファイル

NetCOBOL for .NET では、COBOL プログラムの実行環境開設時に取り込まれた環境変数情報をシステムの環境変数に設定しません。 環境変数情報は現在のアプリケーション ドメインのアプリケーション ドメイン プロパティに、環境変数名をプロパティの名前、環境変数の文字列値をプロパティの値として保持します。 アプリケーションドメイン プロパティについては、.NET Framework のドキュメントの AppDomain クラスを参照してください。

エントリ情報

エントリ情報は、副プログラムの配置や DLL ファイル内でのエントリポイントを指示します。

エントリ情報は、実行可能プログラムの構造が動的プログラム構造の場合、呼び出すプログラムが格納されている DLL を特定するために必要になります。

エントリ情報の内容

エントリ情報には、DLL またはアセンブリを指定することができます。すなわち、副プログラム名とその副プログラムを含む DLL 名またはアセンブリ名を関連付けます。内容の詳細については、エントリ情報の形式を参照してください。

エントリ情報の設定方法

エントリ情報を設定するには、以下の方法があります。

- アプリケーション構成ファイルを使用する
- エントリ情報ファイルを使用する

NetCOBOL for .NET では、アプリケーション構成ファイルを使用する設定を推奨しています。

各設定方法の詳細については、それぞれの説明を参照してください。

エントリ情報の検索順序

以下の順序で検索し、最初に見つかったファイルの情報だけが有効になります。

1. 実行環境変数@CBR_ENTRYFILE に指定したエントリ情報ファイル
2. アプリケーション構成ファイルの<entrySettings>要素

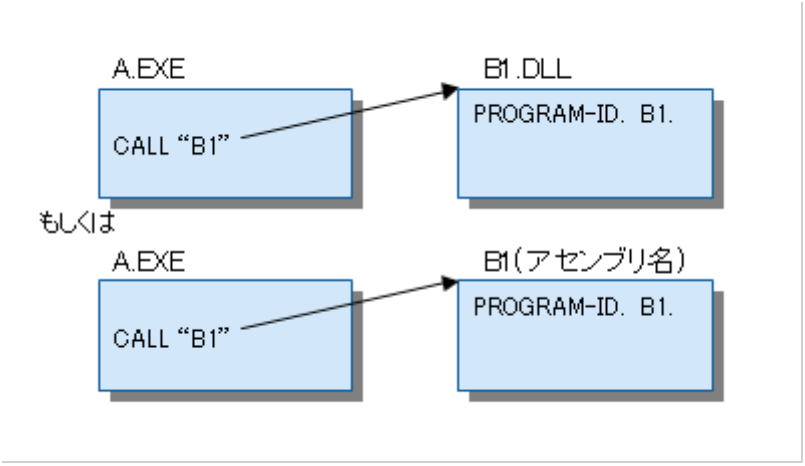
アセンブリ(DLL)の配置について

.NET におけるアセンブリ(DLL)の配置については、以下で詳しく説明しています。

- .NET Framework のドキュメントの共通言語ランタイムのアセンブリ- プログラムの構造に関する情報とアセンブリについて説明しています。
- .NET Framework のドキュメントのアセンブリの配置- アセンブリの配置場所について説明します。
- .NET Framework のドキュメントのランタイムがアセンブリを検索する方法- NetCOBOL for .NET ランタイムが動的にアセンブリをロードする方法を示しています。

エントリ情報の指定について

動的プログラム構造の場合でも、呼び出すプログラムの DLL 名が "プログラム名.DLL" の場合は、エントリ情報を指定する必要がありません。 たとえば、以下のように、呼び出すエントリ名が "B1" で、DLL 名が "B1.DLL" の場合、エントリ情報の指定は必要ありません。



動的プログラム構造については、動的構造を参照してください。

エントリ情報の形式

エントリ情報には、DLL またはアセンブリを指定することができます。すなわち、副プログラム名とその副プログラムを含む DLL 名またはアセンブリ名を関連付けます。

副プログラム名={DLL ファイル名 | アセンブリ名}

DLL ファイル名={相対パス名 | 絶対パス名}

アセンブリ名= 単純テキスト名,Version= バージョン番号

,Culture= カルチャ情報,PublicKeyToken= 公開キートークン

副プログラム名

呼び出されるプログラムのプログラム名(PROGRAM-ID 段落に指定した名前)です。副プログラムを含む DLL ファイル名またはアセンブリ名と関連付けます。

DLL ファイル名

副プログラムを含む DLL ファイルの絶対パスか相対パスの名前です。

アセンブリ名

厳密アセンブリ名を指定します。厳密アセンブリ名の要素(Version,Culture,PublicKeyToken)については、 .NET Framework のドキュメントの厳密な名前のアセンブリを参照するに示されています。

指定するフォルダ名またはファイル名にコンマ（,）が含まれている場合、そのフォルダ名またはファイル名は、二重引用符（"）内に指定されている必要があります。

エントリ情報がパス名やアセンブリ名に強く参照付けられているかどうか、区別する必要がない場合は、以下のように名前を扱います。

- DLL 名としてロードする
- アセンブリ名としてロードする

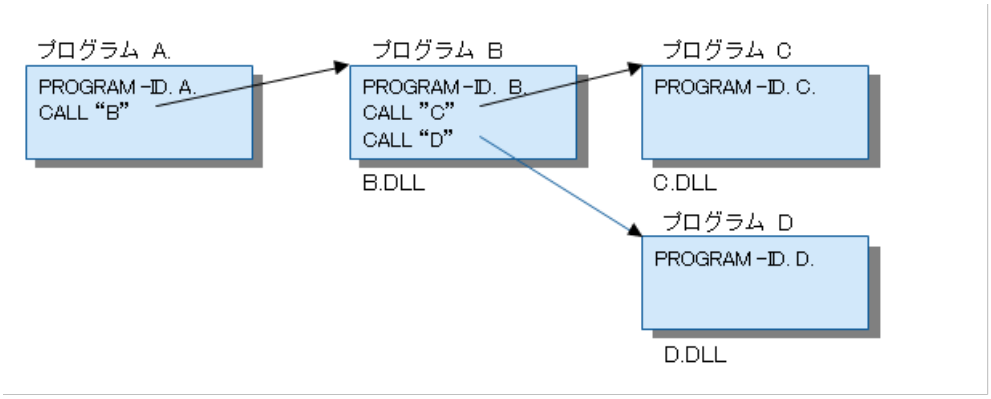
DLL 名としてロードする場合には、 指定された DLL ファイル名が絶対パス名ならそのファイルを、 相対パス名なら COBOL の実行環境を開設した DLL ファイルあるいは EXE ファイル(その環境内で最初に実行されたファイルが実行環境を開設します)からの相対パスにあるファイルをロードします。

アセンブリ名としてロードする場合、アセンブリの検索順序については、 .NET Framework のドキュメントのランタイムがアセンブリを検索する方法を参照してください。 ここでは、COBOL ランタイムが動的にアセンブリをロードする方法を示しています。

もし、DLL 名とアセンブリ名が同じ名前(例えば FOO.DLL)なら、DLL 名を変更して違う名前にする必要があります(FOO.DLL を BAR.DLL に変更する)。

DLL が 1 つの副プログラムで構成されている場合の例

プログラムの呼び出し関係



エントリ情報ファイルを A.ENT とした場合、以下のようになります。

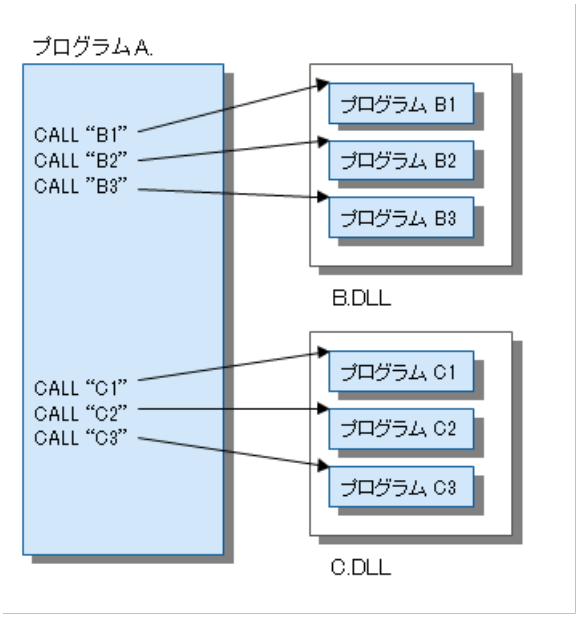
```
@CBR_ENTRYFILE=A.ENT
```

エントリ情報ファイル(A.ENT)の内容

```
[ENTRY]
B=B.DLL
C=C.DLL
D=D.DLL
```

注意: この例では、DLL 名が "プログラム名.DLL" となっているため、エントリ情報を省略することができます。

DLL が複数の副プログラムで構成されている場合の例



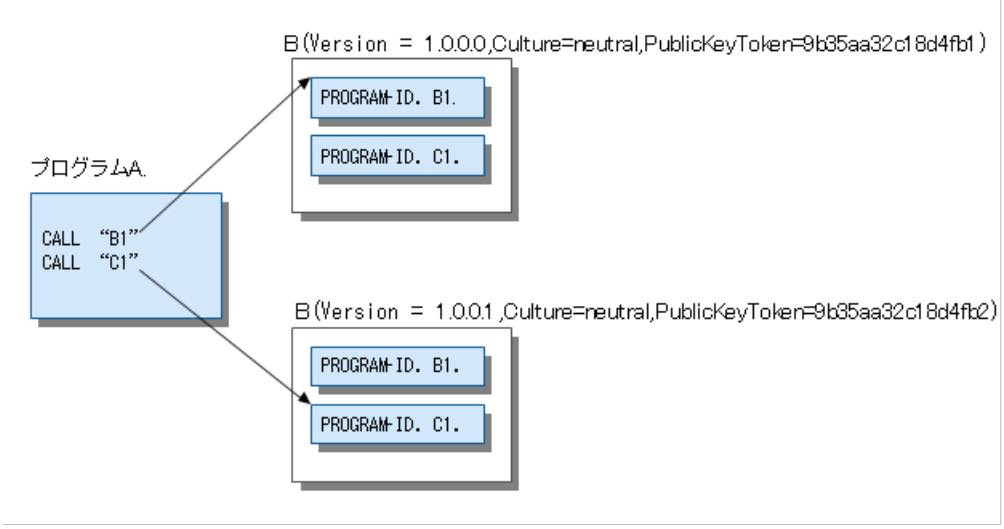
エントリ情報ファイルを A.ENT とした場合、以下のようになります。

```
@CBR_ENTRYFILE=A.ENT
```

エントリ情報ファイル(A.ENT)の内容

```
[ENTRY]
B1=B.DLL
B2=B.DLL
B3=B.DLL
C1=C.DLL
C2=C.DLL
C3=C.DLL
```

副プログラムを含むアセンブリの例



エントリ情報ファイルを FILE とした場合、以下のようになります。

```
@CBR_ENTRYFILE=FILE
```

エントリ情報ファイル(FILE)の内容

```
B1=B,Version=1.0.0.0,Culture=neutral,PublicKeyToken=9b35aa32c18d4fb1
C1=B,Version=1.0.0.1,Culture=neutral,PublicKeyToken=9b35aa32c18d4fb2
```

NCOBU00093

アプリケーション構成ファイル

アプリケーション構成ファイルとは

アプリケーション構成ファイルは XML 形式のファイルで、 実行時に必要な情報を設定することができます。 このファイルに実行環境情報を設定することで、 従来の実行用の初期化ファイルやエントリ情報ファイル、 ODBC 情報ファイルを使用せずに実行環境情報を設定することができます。

アプリケーション構成ファイルの名前は、「exe 名.config」または「web.config」です。

構成ファイルの詳細については、.NET Framework のドキュメントの構成ファイルを参照してください。

このセクションの内容

アプリケーション構成ファイルの形式

アプリケーション構成ファイルに NetCOBOL for .NET のための実行環境設定を行う場合、 <fujitsu.cobol>要素-<runtime>要素に設定したい構成セクション要素を作成し、 その構成セクション要素に情報を指定します。

構成セクション要素の詳細については、.NET Framework のドキュメントの ASP.NET 構成ファイルの構造 (セクションおよびセクション ハンドラ)を参照してください。

指定可能な構成セクション要素は以下のとおりです。

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings/>
    <entrySettings/>
    <sqlSettings/>
  </runtime>
</fujitsu.cobol>
```

<environmentSettings>要素

COBOL 実行環境情報を設定します。 詳細については、COBOL 実行環境情報のためのアプリケーション構成ファイルの形式を参照してください。

<entrySettings>要素

エントリ情報を設定します。 詳細については、エントリ情報設定のためのアプリケーション構成ファイルの形式を参照してください。

<sqlSettings>要素

SQL 情報を設定します。 詳細については、SQL 情報設定のためのアプリケーション構成ファイルの形式を参照してください。

- セクション要素名は大文字・小文字を区別します。 アプリケーション構成ファイルの編集は慎重に行わなければなりません。 アプリケーション構成ファイルを編集する場合は、実行環境設定ユーティリティの利用をお薦めします。
- アプリケーション構成ファイル内に同じセクション要素が複数存在する場合はひとつのセクション要素にマージされます。 また、 同一のセクション要素内で同一のキーを持つ子要素(たとえば<add>要素の key 属性など)が存在する場合は、 最後に指定された要素が有効になります。

NCOBU00095

COBOL 実行環境情報のためのアプリケーション構成ファイルの形式

アプリケーション構成ファイルで実行環境設定を行うためには、COBOL 実行環境情報のための構成セクションを作成する必要があります。

COBOL 実行環境情報の設定

COBOL 実行環境情報の設定は<environmentSettings>要素内の<add>要素の key 属性に変数名を、value 属性に値をそれぞれ指定します。実行環境変数の詳細は実行環境変数 - 詳細を参照してください。

以下に COBOL 実行環境情報の設定の形式を示します。

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="環境変数名" value="値" />
          :
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

例:

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="@MessOutFile" value="A.out" />
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

同一の環境変数名を複数個指定しないでください。同一の環境変数名を複数個指定した場合の動作については保証されません。

エントリ情報設定のためのアプリケーション構成ファイルの形式

アプリケーション構成ファイルでエントリ情報の設定を行うためには、エントリ情報設定のための構成セクションを作成する必要があります。

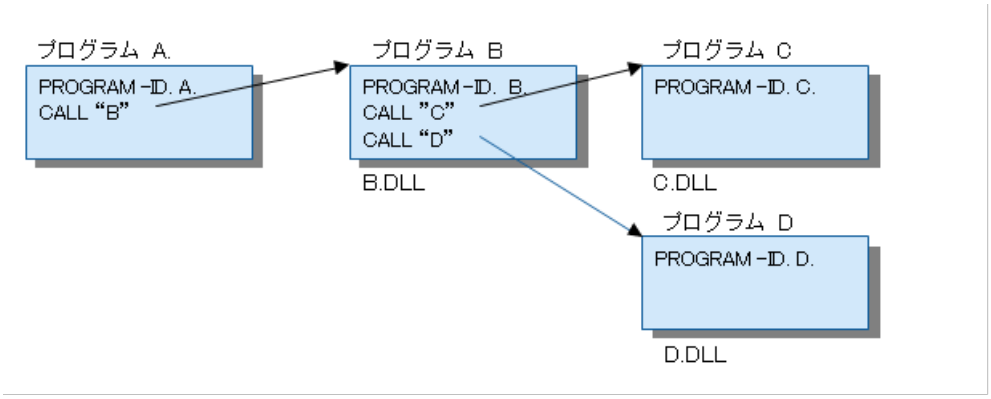
エントリ情報の設定

エントリ情報の設定は<entrySettings>要素内の<add>要素の key 属性に副プログラム名を、value 属性に DLL 名またはアセンブリ名をそれぞれ指定します。詳細については、エントリ情報の形式を参照してください。

以下にエントリ情報の設定の形式を示します。

```
<fujitsu.cobol>
  <runtime>
    <entrySettings>
      <add key="副プログラム名" value="DLL 名またはアセンブリ名" />
          :
    </entrySettings>
  </runtime>
</fujitsu.cobol>
```

例:




```
<fujitsu.cobol>
  <runtime>
    <entrySettings>
      <add key="B" value="B.DLL" />
      <add key="C" value="C.DLL" />
      <add key="D" value="D.DLL" />
    </entrySettings>
  </runtime>
</fujitsu.cobol>
```

注意

同一の副プログラム名を複数個指定しないでください。同一の副プログラム名を複数個指定した場合の動作については保証されません。

NCOBU00097

SQL 情報設定のためのアプリケーション構成ファイルの形式

アプリケーション構成ファイルで SQL 情報の設定を行うためには、SQL 情報設定のための構成セクションを作成する必要があります。

SQL 情報の設定

SQL 情報の設定は<sqlSettings>要素内の各子要素に記述します。

以下に各要素の概要を示します。

<serverList>要素

サーバ情報の集合を示します。各サーバ情報は<server>要素に設定されます。

<server>要素

サーバ情報の詳細情報を示します。 各属性に指定する内容は以下のとおりです。

- name 属性 CONNECT 文またはデフォルトコネクション情報に指定したサーバ名
- type 属性データベースの接続方法を示す"adonet"または"odbc"
- description 属性サーバ情報に対する説明

各設定情報は、<add>要素の key 属性には情報名を、value 属性には設定内容を指定します。 各情報については、サーバ情報を参照してください。

```
<fujitsu.cobol>
  <runtime>
          :
    <sqlSettings>
          :
      <serverList>
        <server name="サーバ名 1" type="odbc" description="説明 1">
          <add key="情報名" value="設定内容" />
            :
        </server>
        <server name="サーバ名 2" type="adonet" description="説明 2">
          <add key="情報名" value="設定内容" />
            :
        </server>
      </serverList>
          :
    </sqlSettings>
          :
  </runtime>
</fujitsu.cobol>
```

例:

```
<fujitsu.cobol>
  <runtime>
    <sqlSettings>
      <serverList>
        <server name="SV1" type="odbc" description="ODBC 情報のサーバ SV1">
          <add key="@SQL_DATASRC" value="SRC1" />
        </server>
      </serverList>
    </sqlSettings>
  </runtime>
</fujitsu.cobol>
```

<sqlDefaultInf>要素

デフォルトコネクション情報を示します。各設定情報は、<add>要素の key 属性には情報名を、value 属性には設定内容を指定します。この要素は ODBC 情報ファイルの[SQL_DEFAULT_INF]セクションに対応します。各情報についてはデフォルトコネクション情報を参照してください。

```
<fujitsu.cobol>
  <runtime>
          :
    <sqlSettings>
          :
      <sqlDefaultInf>
        <add key="情報名" value="設定内容" />
          :
      </sqlDefaultInf>
          :
    </sqlSettings>
          :
  </runtime>
</fujitsu.cobol>
```

例:

```
<fujitsu.cobol>
  <runtime>
    <sqlSettings>
      <sqlDefaultInf>
        <add key="@SQL_SERVER" value="SV1" />
      </sqlDefaultInf>
    </sqlSettings>
  </runtime>
</fujitsu.cobol>
```

<connectionScope>要素

コネクションの有効範囲に関する情報を示します。この要素は ODBC 情報ファイルの[CONNECTION_SCOPE]セクションに対応します。各設定情報は、<add>要素の key 属性には情報名を、value 属性には設定内容を指定します。 各情報については、コネクション有効範囲を参照してください。

```
<fujitsu.cobol>
  <runtime>
          :
    <sqlSettings>
          :
      <connectionScope>
        <add key="情報名" value="設定内容" />
      </connectionScope>
          :
    </sqlSettings>
          :
  </runtime>
</fujitsu.cobol>
```

例:

```
<fujitsu.cobol>
  <runtime>
    <sqlSettings>
      <connectionScope>
        <add key="@SQL_CONNECTION_SCOPE" value="THREAD" />
      </connectionScope>
    </sqlSettings>
  </runtime>
</fujitsu.cobol>
```

<sqlOptionInf>要素

オプション情報の集合を示します。各オプション情報は<option>要素に設定されます。

<option>要素

オプション情報の詳細情報を示します。name 属性にはサーバ情報の @SQL_ODBC_CURSOR_DEFAULT や@SQL_ODBC_CURSOR_UPDATE に指定したオプション名を、description 属性にはオプション情報に対する説明を指定します。

各設定情報は、<add>要素の key 属性にはオプション情報名を、value 属性には設定内容を指定します。 各情報については、SQL オプション情報を参照してください。

```
<fujitsu.cobol>
  <runtime>
          :
    <sqlSettings>
          :
      <sqlOptionInf>
        <option name="オプション名 1" description="説明 1">
          <add key="オプション情報名" value="設定内容" />
          :
        </option>
        <option name="オプション名 2" description="説明 2">
          <add key="オプション情報名" value="設定内容" />
          :
        </option>
      </sqlOptionInf>
          :
    </sqlSettings>
          :
  </runtime>
</fujitsu.cobol>
```

例:

```
<fujitsu.cobol>
  <runtime>
    <sqlSettings>
      <sqlOptionInf>
        <option name="option1" description="読み取り専用カーソルオプション">
          <add key="@SQL_CONCURRENCY" value="READ_ONLY" />
        </option>
      </sqlOptionInf>
    </sqlSettings>
  </runtime>
</fujitsu.cobol>
```

各要素内に同一の情報名およびオプション情報名を複数個指定しないでください。同一の情報名およびオプション情報名を複数個指定した場合の動作については保証されません。

アプリケーション構成ファイルの編集

アプリケーション構成ファイルに COBOL の実行環境情報を設定するには、以下のツールを使うことで簡単に行なうことができます。

- 実行環境設定ユーティリティ

  実行環境設定ユーティリティは、アプリケーション構成ファイルに COBOL の実行環境情報のための構成セクションの宣言を行い、COBOL の実行環境情報を設定することができます。また、実行用の初期化ファイルの内容をアプリケーション構成ファイルに展開させることもできます。

- アプリケーション構成ファイル作成コマンド (CBRtoConfig.exe)

  コマンドベースで実行用の初期化ファイルをアプリケーション構成ファイルへ変換することができます。

NCOBU00099

実行用の初期化ファイル

実行用の初期化ファイルとは

実行用の初期化ファイルは、アプリケーションに必要なすべての実行環境変数を格納できるファイルです。実行用の初期化ファイルは、通常 ".CBR" という拡張子が付いたテキストファイルです。実行用の初期化ファイルは、実行環境設定ツールを使用して編集することができます。

Windows 版 NetCOBOL では、デフォルトで、実行可能ファイル(EXE)を含むフォルダ内で"COBOL85.CBR"という実行用の初期化ファイルを検索していました。この動作は、NetCOBOL for .NET でもサポートされています。

実行用の初期化ファイルの検索については、実行用の初期化ファイルの検索順序を参照してください。

NetCOBOL for .NET では、COBOL アプリケーションを DLL として起動した場合に、DLL のフォルダ配下に COBOL85.CBR を配置しても有効になりません。

このセクションの内容

実行用の初期化ファイルの形式

実行用の初期化ファイルの形式について説明します。

実行用の初期化ファイルの指定方法

実行用の初期化ファイルは、コマンドライン、アプリケーション構成ファイル、および システム環境変数に指定できます。これらの指定方法について説明します。

実行用の初期化ファイルの検索順序

実行用の初期化ファイルを検索する場合の検索順序について説明します。

実行用の初期化ファイルの形式

実行用の初期化ファイルは、１つの共通セクションから構成されます。これは、従来の Windows 版 NetCOBOL の実行用の初期化ファイルの構成とは異なるため注意してください。また、プログラムごとの環境変数情報は記述できません。環境変数情報は、ひとつのプロジェクト内におけるすべてのプログラムで共通です。

以下に実行用の初期化ファイルの形式を示します。

```
環境変数名=値
```

- １つの行には、１つの環境変数情報しか指定できません。
- セミコロン(;)ではじまる行は、注釈行です。セミコロンから改行文字までのテキスト文字は、コメントとして認識されます。

例:

```
@MessOutFile=A.out
```

同一の環境変数名を複数個指定しないでください。同一の環境変数名を複数個指定した場合の動作については保証されません。

関連トピックス

実行環境変数 - 詳細

NetCOBOL for .NET が提供する実行環境変数の一覧を参照できます。

NCOBU00101

実行用の初期化ファイルの指定方法

実行用の初期化ファイルの指定方法には、以下の方法があります。

- コマンドラインの/CBR に指定する方法
- アプリケーション構成ファイルに指定する方法
- 環境変数の@CBR_CBRFILE に指定する方法

これらの指定方法について説明します。

コマンドラインの/CBR オプションに指定する方法

実行用の初期化ファイル名は、識別子 "/CBR" または "-CBR" の後ろに指定します。

```
PROG1.EXE /CBR ABC.INI
```

上の例では、実行用の初期化ファイル名として、ABC.INI を指定しています。(PROG1.EXE が主プログラムの場合)

アプリケーション構成ファイルに指定する方法

アプリケーション構成ファイルを使用して、実行用の初期化ファイル名を指定できます。

この操作を行うには、config ファイル（以下にコンテキストを表示）に、以下の<appSettings>タグを組み込みます。

```
<configuration>
  <!-- This is already declared in Machine.config for you. -->
  <section name="appSettings" type="System.Web.Configuration.NameValueSectionHandler"/>
  <!-- ADDITIONAL CONFIG SECTIONS GO HERE -->
  <appSettings>
    <add key="@CBR_CBRFILE" value="TEST.CBR"/>
  </appSettings>
</configuration>
```

上の例では、実行用の初期化ファイル名として、TEST.CBR を指定しています。

config ファイルの詳細については、 .NET Framework のドキュメントの構成ファイルを参照してください。

環境変数の@CBR_CBRFILE に指定する方法

環境変数@CBR_CBRFILE を使用して、実行用の初期化ファイル名を指定できます。

```
@CBR_CBRFILE=TEST.CBR
```

上の例では、実行用の初期化ファイル名として、TEST.CBR を指定しています。

詳細については、@CBR_CBRFILE (実行用の初期化ファイルの指定)を参照してください。

実行用の初期化ファイルの検索順序

実行用の初期化ファイルの検索順序は以下のようになります。

1. コマンドラインの /CBR オプションに指定されたファイル。
2. アプリケーション構成ファイルの<appSettings>セクションの@CBR_CBRFILE に指定されたファイル。
3. AppDomain の BaseDirectory 内の"COBOL85.CBR" ファイル。
4. システム環境変数@CBR_CBRFILE に指定されたファイル。

AppDomain(アプリケーション ドメイン)とは、 アプリケーションが実行される分離された環境です。AppDomain および AppDomain.BaseDirectory の詳細については、 .NET Framework クラス ライブラリの AppDomain クラスを参照してください。

NCOBU00103

エントリ情報ファイル

エントリ情報ファイルとは

エントリ情報ファイルとは、副プログラムのエントリ情報の定義の開始を示す"ENTRY"セクションを持ち、そのセクションにエントリ情報が指定されているファイルのことをいいます。

このセクションの内容

エントリ情報ファイルの形式

エントリ情報ファイルの内容を以下に示します。

```
[ENTRY]
エントリ情報
```

- [ENTRY]セクションは、副プログラムのエントリ情報の定義の開始を意味し、エントリ情報ファイルに1つしか記述できません。
- エントリ情報については、エントリ情報の形式を参照してください。

エントリ情報ファイルの指定方法

エントリ情報ファイルを使用する場合、環境変数情報@CBR_ENTRYFILE にエントリ情報ファイル名を指定します。

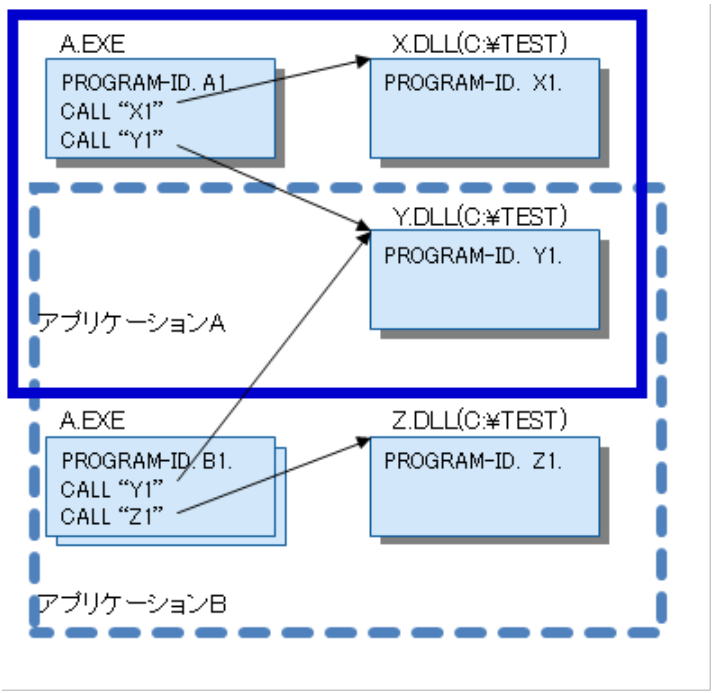
たとえば、エントリ情報ファイル名が `C:¥TEST¥FILE.ENT` である場合、エントリ情報ファイルは次のように設定します。

```
@CBR_ENTRYFILE = C:¥TEST¥FILE.ENT
```

エントリ情報ファイルの使用例

以下の例を使って、エントリ情報ファイルの使用方法について説明します。

アプリケーション A およびアプリケーション B はともに動的プログラム構造で、X.DLL、Y.DLL および Z.DLL を呼び出します。 この中で、Y.DLL はアプリケーション A およびアプリケーション B から呼び出される共通の DLL です。



この例の場合、エントリ情報ファイルの内容は以下のようになります。

エントリ情報ファイル (C:¥TEST¥FILE.ENT)

```
[ENTRY]
X1=C:¥TEST¥X.DLL
Y1=C:¥TEST¥Y.DLL
Z1=C:¥TEST¥Z.DLL
```

[ENTRY]セクションに指定された内容は、すべてのアプリケーションに有効です。つまり、アプリケーション A およびアプリケーション B で共通に使用する Y.DLL は、ここで 1 回定義するだけで、両方のアプリケーションで有効になります。

- エントリ情報では、英小文字と英大文字が区別されますので、指定時には注意してください。
- 同一の副プログラム名を複数個指定しないでください。同一の副プログラム名を複数個指定した場合の動作については保証されません。

SET コマンドでの設定

ここでは、コマンドプロンプトまたはバッチファイルで、SET コマンドを使って環境変数情報を設定する方法について説明します。

コマンドプロンプトで環境変数情報を設定すると、そのコマンドプロンプトから起動されたプログラムで、その環境変数情報が有効になります。

また、バッチファイルを使うことによって、環境設定から実行までを 1 回の作業で行うこともできます。バッチファイルで設定した環境変数情報は、そのバッチファイルで起動したプログラムにだけ有効になります。

コマンドの形式

```
SET 環境変数=環境変数値
```

環境変数と環境変数値については、実行環境変数 - 詳細を参照してください。

NCOBU00106

OS のユーザ環境変数に設定

プログラムの実行前に、コントロールパネルのシステムで環境変数情報を設定します。設定方法については、コントロールパネルのヘルプを参照してください。

実行時オプションの設定

実行時オプションは、実行時に COBOL ソースプログラムに対して、いくつかの情報や動作を指示します。NetCOBOL for .NET における実行時オプションは、外部スイッチおよび CHECK 機能の抑止を指定する場合に使用します。

このセクションの内容

NCOBU00108

実行時オプションの形式

実行時オプションは、実行時に COBOL ソースプログラムに対していくつかの情報や動作を指示します。

COBOL プログラムで使用している機能や、COBOL ソースプログラムを翻訳するときに指定したオプションによっては、実行時オプションを指定する必要があります。

実行時オプションの種類と指定形式を以下に示します。

機能	オプション
CHECK 機能抑止の指定	[noc]、[nocb] 、[nocn]
外部スイッチの値の指定	[s 値]
PowerSORT が使用するメモリ容量の指定	[smsize 値 k]

以下に、それぞれのオプションについて説明します。

[noc] (CHECK 機能抑止の指定)

このオプションは、翻訳オプション CHECK を指定したプログラムの CHECK 機能を抑止する場合に指定します。 CHECK 機能については、CHECK (CHECK 機能の使用の可否)を参照してください。

[nocb] (CHECK(BOUND)機能抑止の指定)

このオプションは、翻訳オプション CHECK(ALL)または CHECK(BOUND)を指定したプログラムの CHECK(BOUND)機能を抑止する場合に指定します。CHECK 機能については、CHECK (CHECK 機能の使用の可否)を参照してください。

[nocn] (CHECK(NUMERIC)機能抑止の指定)

このオプションは、翻訳オプション CHECK(ALL)または CHECK(NUMERIC)を指定したプログラムの CHECK(NUMERIC)機能を抑止する場合に指定します。CHECK 機能については、CHECK (CHECK 機能の使用の可否)を参照してください。

[s 値] (外部スイッチの値の指定)

COBOL プログラムの特殊名段落で指定した外部スイッチ SWITCH-0～SWITCH-7 に値を設定したい場合に指定します。 値には、連続した 8 個のスイッチ値を、一番左が SWITCH-0 に、 その右が SWITCH-1、さらにその右が SWITH-2 と順に一番右の SWITCH-7 まで指定します。 なお、SWITCH-8 を指定した場合、SWITCH-8 は SWITCH-0 に等しいため、 スイッチ値の一番左が SWITCH-8 に、その隣が SWITCH-1 と順に SWITCH-7 に対応します。

外部スイッチには、それぞれ 0 または 1 が指定できます。 省略した場合は、"s00000000" が指定されたとみなされます。

外部スイッチ 1,3,5,7,8 を設定した例を以下に示します。

```
s11010101
```

[smsize 値_k](PowerSORT が使用するメモリ容量の指定)

SORT 文および MERGE 文から呼び出される PowerSORT が使用するメモリ空間の容量を限定したい場合に指定します。 指定する値は、キロバイト単位の数字です。 指定された値が実際に有効になるかについては、PowerSORT の"オンラインマニュアル"をお読みください。 このオプションは、翻訳オプション SMSIZE および特殊レジスタ SORT-CORE-SIZE と同じ意味を持ちますが、 同時に指定された場合の優先順位は、特殊レジスタ SORT-CORE-SIZE が一番強く、 以降、実行時オプション smsize、翻訳オプション SMSIZE の順で弱くなります。

```
smsize300k
```

NCOBU00109

実行時オプションの指定方法

実行時オプションは、以下の方法で指定できます。

- コマンドラインで指定する方法
- 実行環境変数@GOPT で指定する方法

指定方法による実行時オプションの優先順位は以下のようになります。

1. コマンドラインに指定された値
2. アプリケーション構成ファイルに指定された@GOPT の値
3. 環境変数に指定された@GOPT の値

ここでは、それぞれの実行時オプションの指定方法について説明します。

コマンドラインで指定する方法

実行時オプションは、識別子/CBL または-CBL の後ろに指定します。

実行時オプションの指定形式については、実行時オプションの形式を参照してください。

```
例 : PROG1.EXE -CBL s11101101
```

s11101101 は、実行時オプションです。 外部スィッチ 0(もしくは 8)、1, 2, 4, 5,7 を設定しています。

実行環境変数@GOPT に指定する方法

実行環境設定ユーティリティを使用して、実行環境変数@GOPT を指定します。これにより、アプリケーション構成ファイルの内容は、以下のように設定されます。

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="@GOPT" value="s01010101" />
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

アプリケーションの実行

以下の方法で、NetCOBOL for .NET アプリケーションを実行することができます。

Visual Studio IDE から実行する場合

[デバッグ] メニューから [開始] または [デバッグなしで開始] を選択します。

コマンドラインから実行する場合

コマンドラインからアプリケーションを実行する場合、以下の形式で実行環境を引数として指定できます。

```
実行可能ファイル [実行時パラメタ]
                 [/CBR 実行用の初期化ファイル名]
                 [/CBL 実行時オプション]
```

- 実行用の初期化ファイルは、識別子/CBR または-CBR の後ろに指定します。
- 実行時オプションは、識別子/CBL または-CBL の後ろに指定します。
- /CBR および/CBL は順不同です。

エクスプローラーから実行する場合

エクスプローラー上で、アプリケーションの.EXE ファイルをダブルクリックします。

関連トピックス

実行用の初期化ファイル

実行用の初期化ファイルについて説明しています。

実行時オプションの設定

実行時オプションについて説明しています。

NCOBU00111

アプリケーションの配置

多くのプログラミング言語と同様に、NetCOBOL for .NET はランタイムシステムを持っています。 NetCOBOL for .NET で開発したアプリケーションを配置して実行するには、あらかじめそのマシンに NetCOBOL for .NET のランタイムシステムがインストールされている必要があります。 NetCOBOL for .NET のランタイムシステムは、NetCOBOL for .NET 製品によりインストールされます。

作成したアプリケーションの配置については、.NET Framework のドキュメントのアプリケーションの配置を参照してください。

NetCOBOL for .NET V4.0 以前のバージョンで作成したアプリケーションを実行する場合の注意事項

NetCOBOL for .NET V4.0 以前のバージョンで作成したアプリケーションを実行する場合の注意事項について説明します。

.NET Framework について

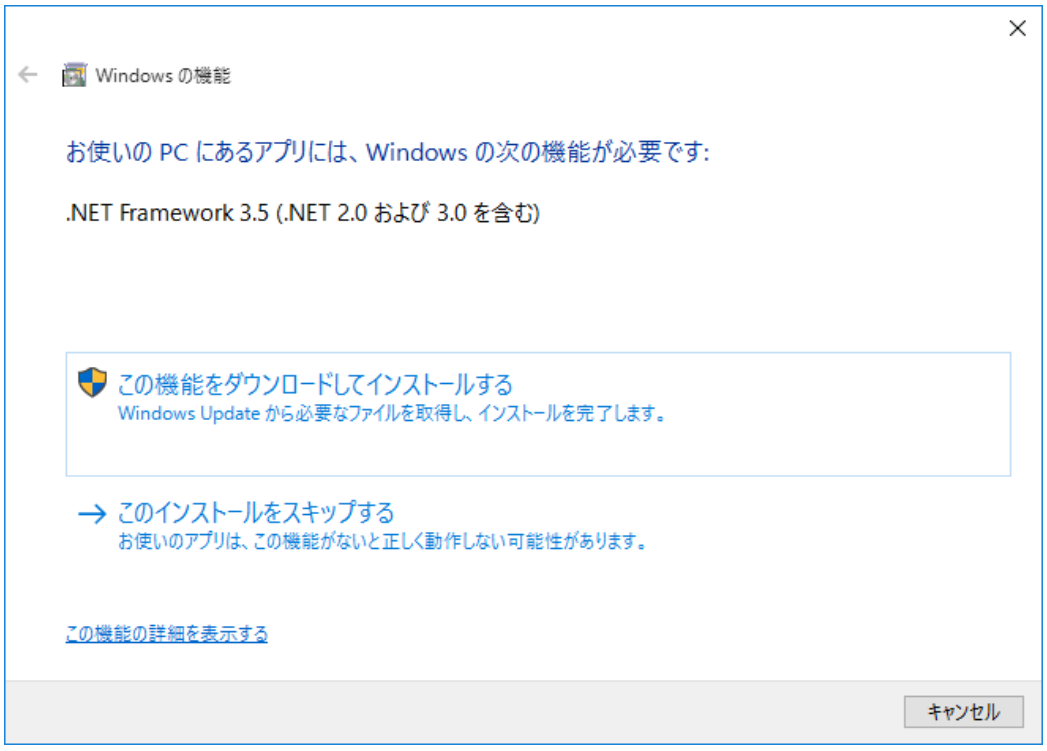
NetCOBOL for .NET にはターゲットとしている.NET Framework があります。 各.NET Framework には実行エンジンである共通言語ランタイム(CLR:Common Language Runtime)が存在します。 CLR は実行時のコード管理をするエージェントのようなもので、CLR のバージョンアップはアプリケーション実行環境のさまざまな変更を伴います。 そのため、新しい実行エンジン上でアプリケーションを実行するには、アプリケーションをリビルドする必要があります。

詳細は MSDN ライブラリの.NET Framework の概要を参照してください。

NetCOBOL for .NET V4.1 以降の使用

NetCOBOL for .NET V4.0 以前のバージョンで作成したアプリケーションを NetCOBOL for .NET V4.1 以降を使用して実行することはできません。 NetCOBOL for .NET V4.1 以降 を使用してアプリケーションを実行するには、アプリケーションを NetCOBOL for .NET V4.1 以降でリビルドする必要があります。 NetCOBOL for .NET V4.1 以降で翻訳するためのプロジェクトの移行については「プロジェクトのアップグレード」を参照してください。

- NetCOBOL for .NET V4.0 以前のバージョンで作成したアプリケーションを、.NET Framework 4 以降の.NET Framework のみがインストールされた環境で実行すると、以下のエラーが発生します。



- 複数の.NET Framework がインストールされている環境で、NetCOBOL for .NET V4.0 以前のバージョンで作成したアプリケーションを動作させると、「アセンブリ"Fujitsu.COBOL"が見つからない」という意味のエラーメッセージが表示され、 アプリケーションが動作しない場合があります。 これは、そのアプリケーションが古いバージョンの.NET Framework 上で動作した場合に発生します。アプリケーションが利用する.NET Framework のバージョンは、アプリケーション開始時に決定します。 アプリケーション開始時に、EXE ファイルがリンクしている.NET Framework のバージョンと同じバージョンの.NET Framework がその環境に存在すれば、そのバージョンの.NET Framework が利用されます。

- NetCOBOL for .NET V4.0 以前のバージョンで作成されたアプリケーションを実行する場合は、NetCOBOL for .NET V4.0 をご使用ください。

デバッグ

Visual Studio でアプリケーションを開発した場合は、 Visual Studio デバッガーを使用できます。 このデバッガーは、COBOL および混合言語で開発されたアプリケーションに適用し、 優れた機能を提供します。

Visual Studio デバッガーの機能については、 Visual Studio のドキュメントの Visual Studio でのデバッグを参照してください。

ここでは、COBOL プログラマの視点から見た、デバッグ作業に関する情報を提供します。

NetCOBOL for .NET プロジェクトをデバッグするには、[ツール]メニューから[オプション]を選択し、[デバッグ] > [全般]の[マネージ互換モードの使用]をオンにする必要があります。

このセクションの内容

NCOBU00113

デバッグオプションの設定

NetCOBOL for .NET プロジェクトにデバッグオプションを設定する場合

アプリケーションのデバッグ時に、そのアプリケーションに渡す引数や作業用のフォルダを指定するには、以下の手順で行います。

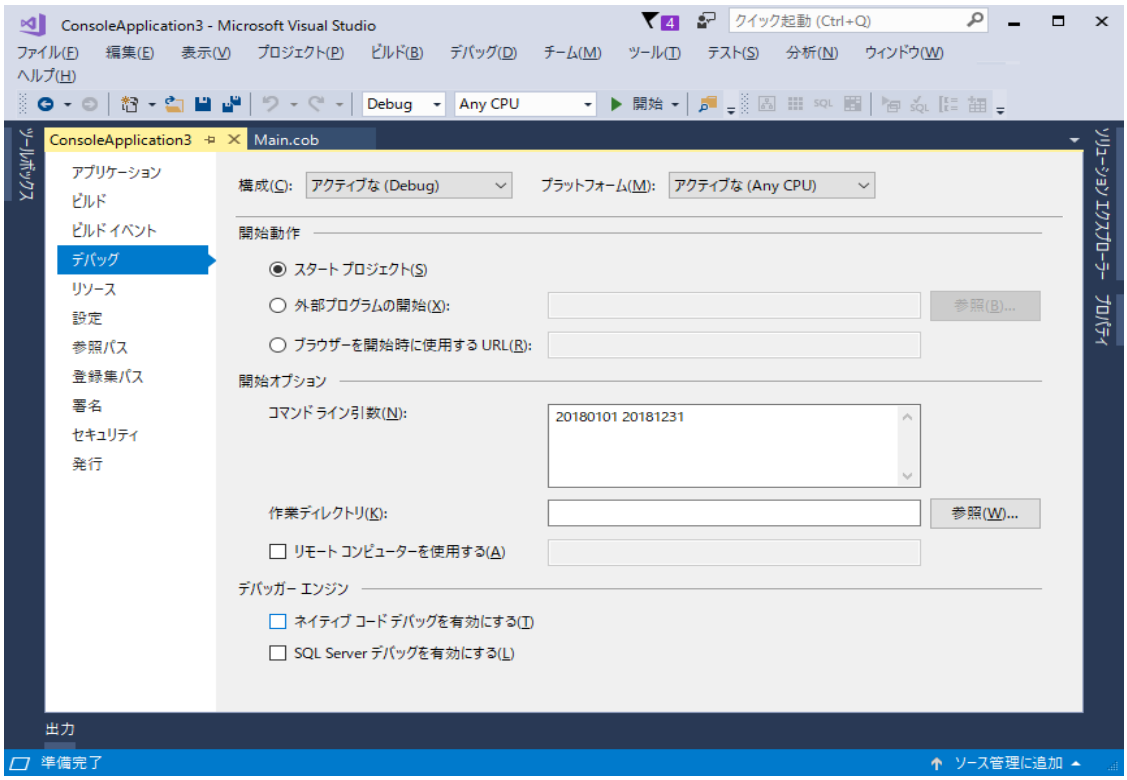
1. ソリューションエクスプローラーのプロジェクト名を右クリックし、コンテキストメニューから[プロパティ]を選択します。

   ソリューションエクスプローラーが表示されない場合は、[表示]メニューを選択し、ソリューションエクスプローラーをクリックすることで表示されます。

   これにより、プロジェクトの[プロパティページ]ダイアログボックスが表示されます。

2. [プロパティページ]ダイアログボックスの左ペインにある [デバッグ]をクリックします。

3. [開始オプション]の[コマンドライン引数]プロパティおよび[作業ディレクトリ]プロパティに、デバッグセッションを開始するときにスタートアプリケーションに渡す作業フォルダと任意のコマンドライン引数を指定します。



ASP.NET Web サイトにデバッグオプションを設定する場合

ASP.NET Web サイトのデバッグ時の設定を編集するには、以下の手順で行います。

1. ソリューションエクスプローラーの Web サイト名を右クリックし、コンテキストメニューから[プロパティ]を選択します。

   ソリューションエクスプローラーが表示されない場合は、[表示]メニューを選択し、ソリューションエクスプローラーをクリックすることで表示されます。

これにより、プロジェクトの[プロパティページ]ダイアログボックスが表示されます。

2. [プロパティページ]ダイアログボックスの左ペインにある、[開始オプション]を選択します。

NCOBU00114

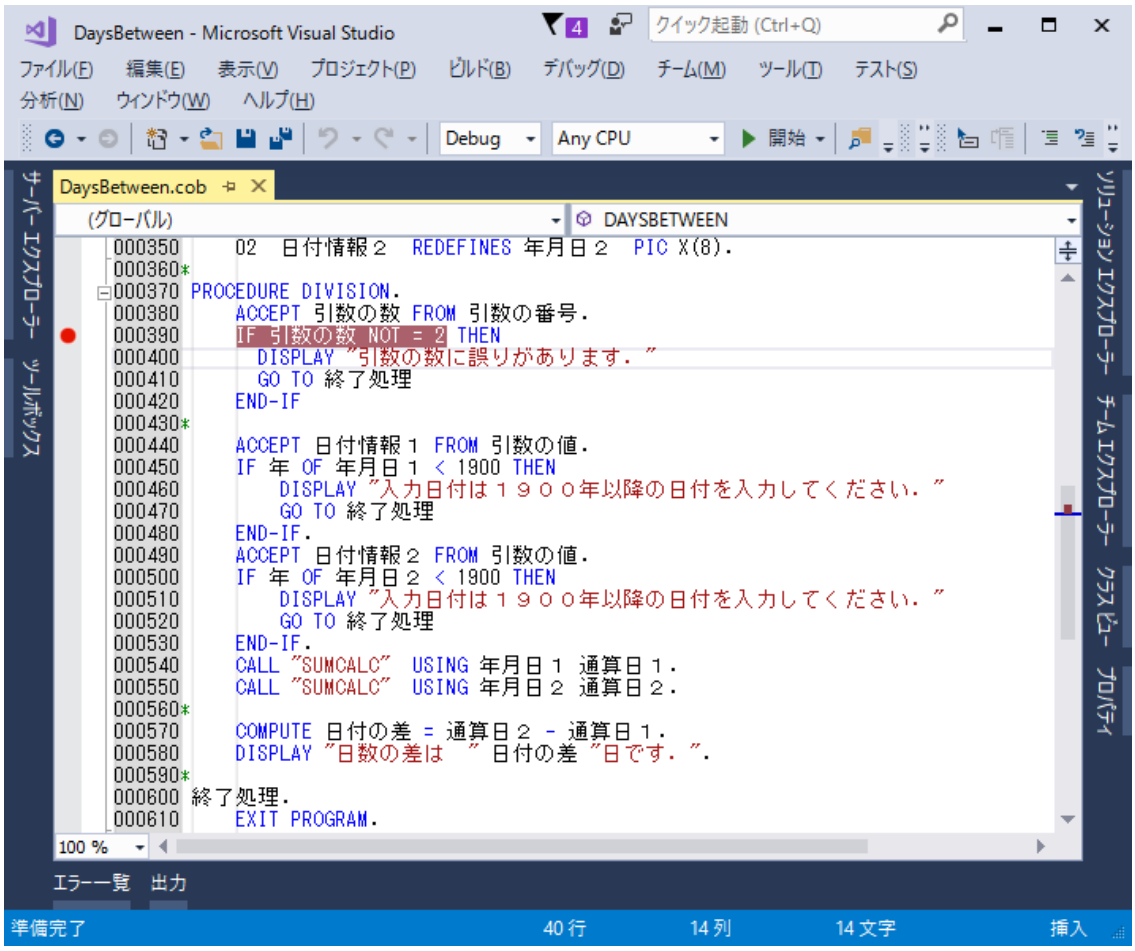
デバッグの開始

デバッグの前に

NetCOBOL for .NET プロジェクトをデバッグするには、 デバッグモードで、プロジェクトをビルドしておく必要があります。

デバッグモードでビルドするには、ツールバーの[ソリューション構成]、 または [ビルド] メニューから [構成マネージャー] を選択することで、アクティブな構成を保存することができます。

ツールバーの[ソリューションの構成]は、標準ツールバーとして表示されています。



ここでは、デバッグの開始方法として、 「ステップイン」と「ブレークポイントまで実行する」方法について説明します。

ステップイン

コードの最初の行を実行できる状態にあるデバッガーを開始する場合、 [デバッグ] メニューまたは [デバッグ] ツールバーから [ステップ イン] を選択します。

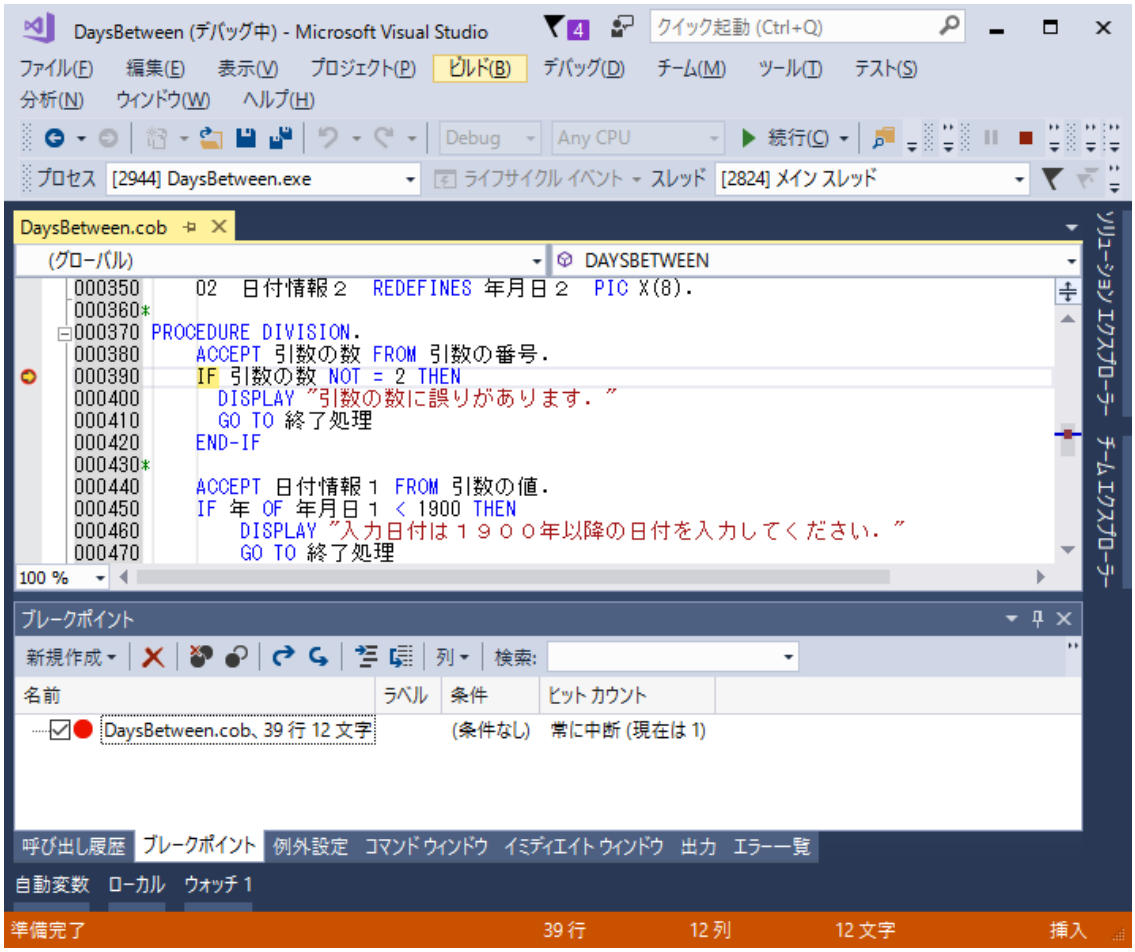
[ステップ イン]を実行すると、ソースファイルが開き、 プログラムの入り口である PROCEDURE DIVISION に黄色のポインタが設定されます。 黄色いポインタは、次に実行するコードの行を示しています。 もう一度、 [ステップ イン] を実行すると、プログラムの最初の文に移動します。

コンソールアプリケーションのデバッグ時には、コンソール画面も表示されます。

ブレークポイントまで実行する

設定したブレークポイントまでプログラムを実行したい場合、 [デバッグ]メニューから [デバッグの開始]を選択して、プログラムを実行します。 この場合はブレークポイントに達するまで、プログラムは実行されます。 ブレークポイントに達した場合、 停止したソースコードの位置に黄色のポインタが表示されます。

停止中のブレークポイントは、[ブレークポイント]ウィンドウの一覧に太字で示されます。



ブレークポイントに達しない場合、プログラムが終了するまで、 またはエラーが発生するまで実行し続けます。

NCOBU00115

ブレークポイントの設定

プログラムをデバッグする場合、ブレークポイントは必要不可欠なツールです。 Visual Studio は、多種多様なブレークポイントを提供します。 これらのブレークポイントについては、 Visual Studio のドキュメントのブレークポイントの使用を参照してください。

ここでは、COBOL ソースにブレークポイントを設定する方法について説明します。

ブレークポイントの種類には、以下があります。

- 文にブレークポイントを設定する

  COBOL の文にブレークポイントを設定できます。

- 中断条件を設定する

  ある位置でブレークポイントを設定し、その位置に達した時に評価される条件を設定できます。条件式は、COBOL の式で入力します。

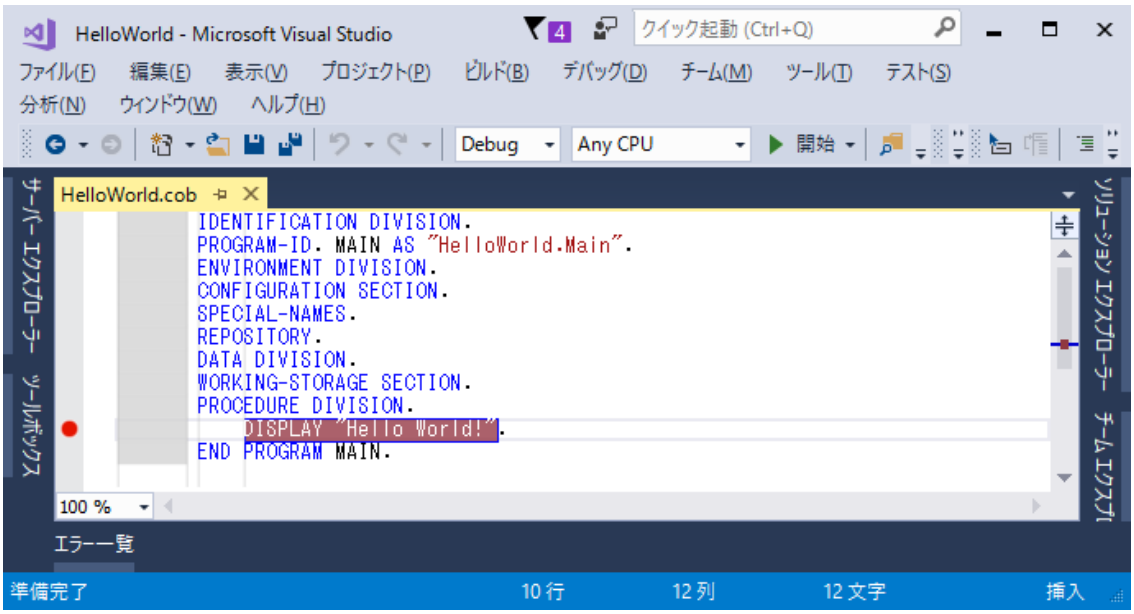
- 関数ブレークポイントを設定する

  実行が指定の COBOL 関数の位置に達したときに、プログラムを中断させます。ここでいう COBOL 関数とは、手続き部を持つプログラム定義またはメソッド定義を指します。

文にブレークポイントを設定する

文にブレークポイントを設定する方法には、以下の方法があります。

- 文にブレークポイントを設定する最も簡単な方法は、ソースコードの左側にある余白部分(マージン)をクリックすることです。赤い丸形のグリフが左のマージンに表示されます。これは、その部分に、ブレークポイントが設定されたことを意味します。



- 2 つ目の方法は、文にカーソルを合わせて右クリックし、コンテキストメニューから[ブレークポイント]-[ブレークポイントの挿入]を選択します。

NCOBU00117

中断条件を設定する

ある位置でブレークポイントを設定し、その位置に達した時に評価される条件を設定できます。この条件を中断条件といいます。

例えば、ループカウンタのようなデータ項目が、特定の値に達したときだけ、実行を中断させることができます。入力可能な条件の種類についての詳細は、COBOL の式を参照してください。

中断条件を設定するには、以下の方法があります。

- 文に通常のブレークポイントを設定してから、その文を右クリックし、コンテキストメニューから[ブレークポイント]-[条件]を選択します。

  [ブレークポイントの条件]ダイアログで中断条件を入力します。

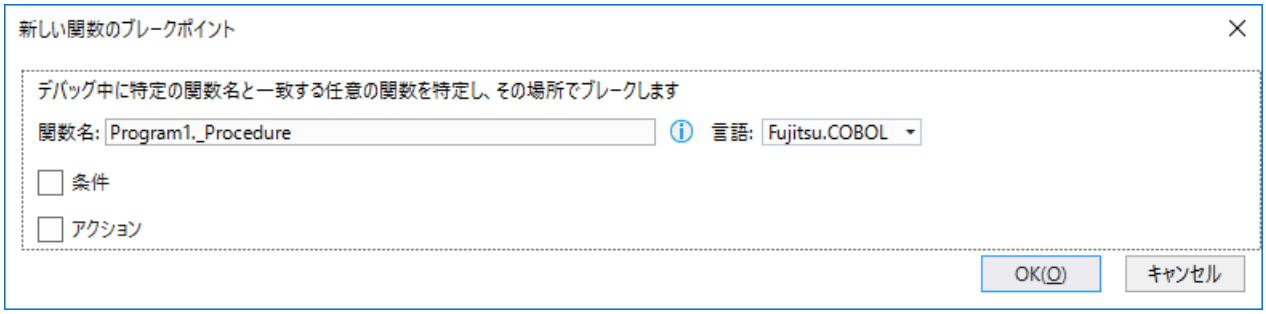
- 文に通常のブレークポイントを設定してから、[ブレークポイント]ウィンドウにあるその文を右クリックし、コンテキストメニューから[条件]を選択します。

  [ブレークポイントの条件]ダイアログで中断条件を入力します。

関数ブレークポイントを設定する

関数ブレークポイントは、実行が指定の COBOL 関数の位置に達したときに、プログラムを中断させます。ここでいう COBOL 関数とは、手続き部を持つプログラム定義またはメソッド定義を指します。

COBOL 関数ブレークポイントを設定するには、[デバッグ]メニューから [ブレークポイントの作成]-[関数のブレークポイント] を選択し、 [新しい関数のブレークポイント]ダイアログ の[関数名]フィールドにプログラム名またはメソッド名を以下の形式で入力します。



プログラム定義の場合

以下の書き方で記述します。

- プログラム名._Procedure

プログラム名と"_Procedure"は、ピリオド(.)で区切ります。

メソッド定義の場合

以下の書き方で記述します。

- クラス名.メソッド名

クラス名は、そのメソッド定義が含まれるクラスのクラス名です。 クラス名とメソッド名は、ピリオド(.)で区切ります。

プログラム名段落、クラス名段落、またはメソッド名段落に AS 指定がある場合、外部名を使用します。

例 1:

```
PROGRAM-ID. Program1 AS "HelloWorld.Program1".
```

このケースでは、以下の書き方で記述します。

- HelloWorld.Program1._Procedure

例 2:

```
CLASS-ID. class1 AS "HelloWorld.Class1".
METHOD-ID. m1 AS "Method1".
```

このケースでは、以下の書き方で記述します。

- HelloWorld.Class1.Method1

[関数のブレークポイント]は、[デバッグ]メニューに表示されていない場合があります。 この場合は、[ツール]メニューから[カスタマイズ]を選択し、 [カスタマイズ]ダイアログから[関数のブレークポイント]を追加してください。 手順の詳細については、Visual Studio ドキュメントの「方法: Visual Studio でメニューおよびツール バーをカスタマイズする」を参照してください。

データ値の監視

Visual Studio デバッガーには、 データの値を検査する方法がいくつかあります。

データチップ

デバッガーがプログラムを実行した場合、マウスポインタでデータ名に近づけると、ツールチップと同じように、データ項目に含まれる値が表示されます。

テーブル項目の値、または、2 語以上からなるデータ項目の値を参照したい場合は、参照するデータ項目を構成する語をすべて（例えば、テーブル項目名とそのインデックス、または添字）選択して、マウスポインタを選択したテキストに近づけます。デバッガーが、選択された式を評価して、それに対応するデータ項目の値を表示します。

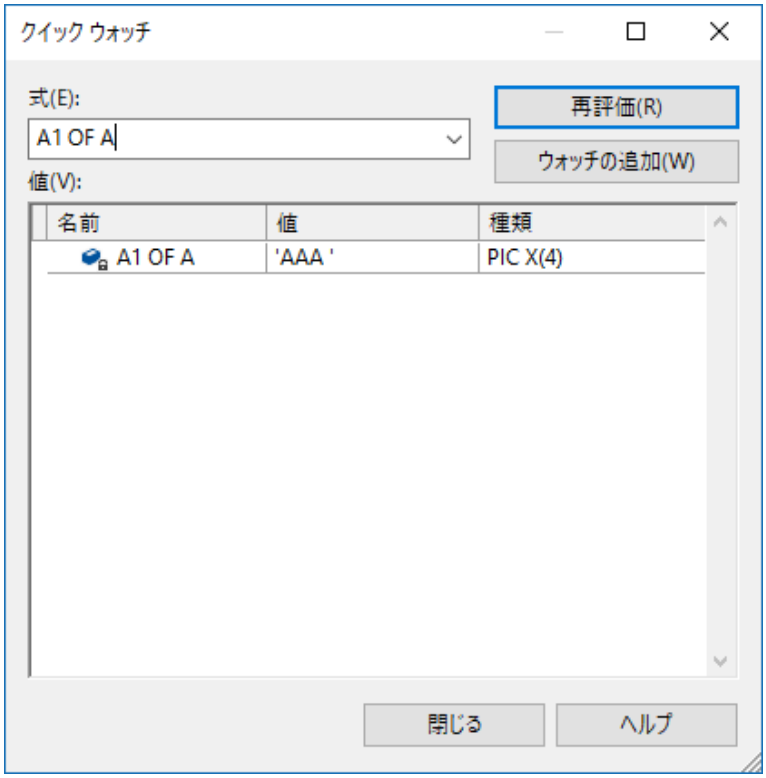
クイック ウォッチ

基本項目でないデータ項目の値を検査したい、またはソースコードで利用できない式を入力する必要がある場合、「クイック ウォッチ」機能を使用します。

この機能を使用するには、テキスト（または、参照したい値を持つ、選択されたテキスト）を右クリックして、コンテキストメニューから[クイック ウォッチ]を選択します。

[クイック ウォッチ]ダイアログボックスでは、選択されたデータ項目の値が表示され、また、以下の操作が可能になります。

- 従属項目の値を表示する
- 評価中の式を編集する
- [ウォッチ式の追加]ボタンを押すことで、ウォッチ ウィンドウに、表示している式を追加する



式を監視する

データ項目の値と同様、式も監視できます。入力できる式の詳細については、 Visual Studio のドキュメントのデバッガー内の式を参照してください。

式を監視するには、[クイック ウォッチ]ダイアログボックスを表示して、[式]フィールドに式を入力します。

例えば、現在のインデックスの 2 つ先にあるテーブル項目の値を監視したい場合、以下のような式を入力できます。

```
MyTableItem (MyTableIndex + 2)
```

中断条件

データ値を監視する別の方法は、文に条件付きのブレークポイントを設定するというものです。

条件は、データ項目または式の値をテストできます。条件を true にした場合、条件を満たした時デバッガーは実行を中断します。詳細については、中断条件を設定するを参照してください。

デバッグウィンドウの種類

デバッグウィンドウには、以下の種類があります。

ウィンドウの種類	使用目的と使用方法	ウィンドウの表示方法
ウォッチウィンドウ	ウォッチ ウィンドウでは、デバッグセッションを通じて、データ項目の値を監視できます。データ項目、または式がウォッチ ウィンドウに追加された場合、その値は監視されます。そして、値に変更があると、変更部分が赤字で表示されます。 以下の方法のいずれかを使用して、ウォッチ ウィンドウに項目を追加します。 ・ 監視の対象となるデータ項目、または式を選択します。右クリックをして、コンテキストメニューから[ウォッチ式の追加]を選択します。 ・ 監視の対象となるデータ項目、または式を選択してから、[クイック ウォッチ]を選択します。必要に応じて、式を編集します。[クイック ウォッチ]ダイアログから[ウォッチ式の追加]を選択します。 ・ [デバッグ]メニュー、またはコンテキストメニューから[クイック ウォッチ]を選択し、監視の対象となるデータ項目名、または式を入力します。	左記のいずれかの方法、または、[デバッグ]メニューから[ウィンドウ]を選択し、[ウォッチ]をクリックします。
ローカルウィンドウ	ローカルウィンドウは、現在実行されている手続きを含むメソッド定義またはプログラム定義で宣言されている、データ項目を表示します。 現在実行されている手続きがスタティックメソッドの場合、すべてのスタティックデータが表示されます。	[デバッグ]メニューから[ウィンドウ]を選択し、[ローカル]をクリックします。

	現在実行されている手続きがプログラム定義の場合、作業場所節や連結節のデータがすべて表示されます。	
自動変数ウィンドウ	自動変数ウィンドウは、現在使用している文、および前に表示されたデータ項目を表示します。	[デバッグ]メニューから[ウィンドウ]を選択し、[自動変数]をクリックします。

メニュー項目が、[デバッグ]メニューに表示されていない場合があります。この場合は、[ツール]メニューから[ユーザ設定]を選択し、[ユーザ設定]ダイアログの[コマンド]タブで、"デバッグ"分類から必要なコマンドを[デバッグ]メニューに追加する必要があります。

データ項目の表示における注意事項

- 値がそのデータ項目で無効な場合、例えば、PIC 9 項目に設定できない、x"00"のような数字でない値が含まれているといった場合、値またはメッセージは表示されません。
- データ項目が基本項目ではない場合、デバッガーは項目の長さを表示します。
- データ項目がオブジェクト参照の場合は、次のいずれかを表示します。
  - オブジェクト固有の表示文字列
  - オブジェクトのクラス名
  - “NULL”(オブジェクトが null の場合)
- 英字データ項目、英数字データ項目、英数字編集データ項目、日本語データ項目および日本語編集データ項目の場合、 制御文字のように文字として表示できない文字が含まれているときは、16 進文字リテラルで表示します。
- 「16 進数で表示」を設定して数字項目を表示する場合、 以下の数字項目は 16 進数字リテラル(H’…’のビッグエンディアン形式)で表示します。
  - COMP-5
  - BINARY
  - BINARY-CHAR
  - BINARY-SHORT
  - BINARY-LONG
  - BINARY-DOUBLE

NCOBU00120

COBOL の式

NetCOBOL for .NET 製品には、 Visual Studio デバッガーと統合された式エバリュエータが備わっています。Visual Studio のドキュメントのデバッガー内の式で記述してあるように、式は、デバッガーのさまざまな場所で使用されます。

ここでは、デバッガーの式エバリュエータでサポートされる COBOL の式について説明します。

COBOL プログラムをデバッグする場合、式エバリュエータに入力することができる式は、 値を必要としたときだけ評価されるので、注意してください。 たとえば、式が文の中断点と関連する場合、文を実行する前に、その式は評価されます。 適切なコンテキストが使用可能なときに式が評価される場合だけ、 式の構文検査が実施されます。 式にエラーが発生した場合、初めて式を評価しようとした時にエラーが通知されます。

式の作用対象

データ名

COBOL データ識別子は、データ名、インデックス付きまたは添字付き名前、部分参照の名前、および修飾された名前を含む、式で使用することができます。

定数

以下の定数が含まれます。

- '....'、または、".....": 英数字定数
- N'....'、または、N".....": 日本語文字定数
- NC'....'、または、NC".....": 日本語文字定数
- X'....'、または、X".....": 16 進文字定数
- NX'....'、または、NX".....": 日本語 16 進定数
- B'....'、または、B".....": ブール定数 (1 or 0)
- 123..: 数字定数
- H'....'、または、H".....": 16 進数字定数

表意定数

以下の表意定数が含まれます。

- SPACE (SPACES)
- ZERO (ZEROS, ZEROES)
- HIGH-VALUE
- LOW-VALUE
- QUOTE (QUOTES)
- ALL

ブール（System.Boolean）定数(デバッガー拡張定数)

以下のブール定数が含まれます。この定数は、デバッガーだけで使用できます。

- TRUE
- FALSE

条件式

条件式には、以下の条件演算子が含まれます。

- [IS] [NOT] EQUAL [TO]
- [IS] [NOT] =
- [IS] [NOT] GREATER [THAN]
- [IS] [NOT] >
- [IS] [NOT] LESS [THAN]
- [IS] [NOT] <
- [IS] GREATER [THAN] OR EQUAL [TO]
- [IS] >=
- [IS] LESS [THAN] OR EQUAL [TO]
- [IS] <=

複合条件演算子

条件式には、以下の条件演算子が含まれます。

- AND
- OR

算術演算子

式には、以下の算術演算子が含まれます。

- + （加算）
- - （減算）
- * （乗算）
- / （除算）
- ** （べき乗）

単項演算子

式には、以下の単項演算子が含まれます。

- + （プラス）
- - （マイナス）
- NOT

論理（ブール）演算子

式には、以下の論理演算子が含まれます。

- AND
- OR
- EXOR

リモートデバッグ

リモートデバッグを有効にするには、以下の方法で行います。

1. リモートデバッグを行う場合は、リモートマシンにリモートデバッグのための環境を構築する必要があります。 詳細については、リモートデバッグのセットアップ手順を参照してください。
2. ソリューション エクスプローラーでプロジェクトが選択されている状態で、[プロジェクト] メニューから [プロパティ] を選択します。
3. [デバッグ] タブをクリックします。
4. [リモート コンピューターを使用する] チェック ボックスをオンにします。
5. [リモート コンピューターを使用する] フィールドに、¥¥domain¥machinename の形式でリモート コンピューターの名前を入力します。

NCOBU01160

デバッグ時の注意事項

ここではデバッグする際の注意事項について説明します。

Visual Studio ホスティングプロセスを有効にする

プロジェクトデザイナーの[デバッグ]ページで、[Visual Studio ホスティングプロセスを有効にする]をチェックしても、この機能は有効になりません。

システムクラスのオブジェクト参照項目

システムクラスのオブジェクト参照項目のプロパティまたはフィールドの値を評価した場合、正しく評価できない場合があります。

例外処理アシスタント

デバッガの例外処理アシスタント機能はサポートしていません。

エディットコンティニュ

エディットコンティニュ機能はサポートしていません。

オブジェクト参照のメンバの表示

常にオブジェクトの生の構造を変数ウィンドウに表示します。

デバッガ上での算術式の評価

算術式をデバッガ上で評価する場合、以下の理由により演算結果は実際のプログラムの実行結果と異なる場合があります。

- 式が 2 進項目だけからなる場合は 2 進ネイティブ演算を行います。
- 式に 2 進項目以外の 10 進データが含まれている場合は、10 進データをパック 10 進形式に変換して式を評価します。
- べき乗および浮動小数点項目/定数が含まれる場合は浮動小数演算となる場合があります。
- 演算式の中間結果が 30 桁を超える場合は、正しい演算結果を得られない場合があります。

自由形式

ソースプログラムが自由形式の場合、データ部と手続き部をひとつの行に書くことができます。データ部と手続き部がひとつの行に書かれている場合、エントリポイントに正しく停止できない場合があります。

プロパティの評価およびその他の暗黙な関数の呼び出し

デバッガでデータが評価される場合、プロパティの自動評価と暗黙的な関数の呼び出しは常に有効となります。

プロキシオブジェクトのメンバ

プロキシオブジェクトのメンバを評価することはできません。

デバッガ上でのデータ項目の表示

- デバッガ上で文字データ項目を表示する場合は Unicode(UCS2)文字列に変換して表示します。Unicode 文字として表示できない文字が含まれている場合は 16 進文字列で表現します。 また、Unicode 文字に変換できたとしてもシステムのロケールの違いによって適切に表示できない場合があります。
- デバッガは、翻訳オプション RCS によって文字データ項目の扱いが決まります。 データ項目に翻訳オプション RCS で指定されたコード系と異なる文字が含まれる場合、正しく表示されません。

フィールド名またはプロパティ名の評価

デバッガ上で評価するフィールドまたはプロパティの名前が大文字または小文字の違いだけでクラス内で重複している場合、予期しない評価結果が得られることがあります。

ブレークポイント

デバッガ起動後は、ブレークポイントの位置および範囲はデバッガが補正します。

COPY 登録集

1 つの手続きにおいて、複数の COPY 文を使用して同じ登録原文集を複写し、この登録集原文にブレークポイントを設定した場合、ブレークポイントが有効になるのは最初に複写された登録集原文のみとなます。 2 回目以降の複写された登録集原文にはブレークポイントを設定できません。

基底場所節データ、ポインタデータ項目

ポインタデータ項目を評価することはできません。 また、基底場所節に定義されたデータ項目は、ポインタ修飾子または暗黙でポインタ付けされたデータ項目であっても評価することはできません。

同名のデータ項目が複数存在する場合

プログラム中に同名のデータ項目が複数存在する場合は、一意名修飾を含めた状態で評価してください。

NCOBU00122

アプリケーション開発時の注意事項

NetCOBOL for .NET では、 Visual Studio のビジュアルなデザイナーを利用して ASP.NET Web アプリケーション、XML Web サービス、および Windows アプリケーションを作成することができます。

ここでは、COBOL でこれらのアプリケーションを作成する場合の注意事項について説明します。

このセクションの内容

アセンブリ情報設定時の注意事項

Visual Studio でアプリケーションを開発する際、プロジェクトオプションの[アプリケーション]タブからアセンブリ情報を設定する場合、アセンブリ情報は Properties フォルダの AssemblyInfo.cob という COBOL ソースファイルに記述された、".ASSEMBLY"という名前の特殊なクラス定義を自動的に更新します。

ここではプロジェクトオプションでアセンブリ情報を設定する際の注意事項について説明します。

アセンブリ情報の文字コード系

アセンブリ情報はクラス定義で表現されるため、翻訳オプションの設定で RCS(SJIS)翻訳オプションを指定する場合は、翻訳エラーが発生します。

この場合は AssemblyInfo.cob の先頭に翻訳指示文(@OPTIONS)を追加し、RCS(UTF8-UCS2)または RCS(SJIS-UCS2)を指定します。

```
000000 @OPTIONS RCS(SJIS-UCS2)
```

アセンブリ情報の正書法

アセンブリ情報は可変形式の正書法で生成されます。このため、翻訳オプションの設定で SRF 翻訳オプションに可変形式以外を指定する場合は、翻訳エラーが発生する場合があります。

この場合は AssemblyInfo.cob の先頭に翻訳指示文(@OPTIONS)を追加し、SRF 翻訳オプションで可変形式を指定します。

```
000000 @OPTIONS SRF(VAR)
```

アセンブリ情報の文字列制限

アセンブリ情報を設定する場合、以下の項目の文字列の長さは 50 文字以内でなければなりません。

- タイトル
- 説明
- 会社
- 製品
- 著作権
- 商標

NCOBU00123

ASP.NET アプリケーション開発の注意事項

Web サイトの説明は互換のために残されています。Visual Studio では、Web サイトの作成を推奨していません。NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ(http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

Web サイト(ASP.NET Web アプリケーション、XML Web サービスおよび WCF サービス)において、 対象とする Framework の選択を変更した場合 や NetCOBOL for .NET 以外(Visual C#や Visual Basic)で Web サイトを作成した場合は、 Web 構成ファイル(Web.config)の変更が必要になる場合があります。 詳細については、Web 構成ファイル(Web.config)の変更を参照してください。

プロジェクト作成時の注意事項

Visual Studio 2005 以降では ASP.NET アプリケーションの編集方法が大きく変更されました。 これに伴い、V3.0 以降では、 「新しいプロジェクト」ダイアログの COBOL のフォルダから ASP.NET アプリケーションを作成するためのテンプレートが削除されています。 Visual Studio 2017 で ASP.NET アプリケーションを作成するには、 開発環境の「ファイル」-「新規追加」-「Web サイト」メニューによって表示される「新しい Web サイト」ダイアログからテンプレートを選択する際に、 言語として「Fujitsu NetCOBOL for .NET」を選択します。

IIS アプリケーションの削除

IIS アプリケーション（IIS アプリケーション化されたプロジェクトを含む）を削除する場合は「インターネットサービスマネージャー」を使って削除を行うことを推奨します。エクスプローラーからフォルダを削除しただけでは IIS の内部設定（metabase）が削除されません。

アクセス制限

Web サイトを IIS 上に直接作成した場合、COBOL ソースファイルにはアクセス制限が設定されないため、 外部から HTTP によって参照できてしまいます。 そのため、IIS アプリケーションの設定を変更して、 拡張子が.cob、.cobx、.cbl などのファイルを外部から HTTP によってアクセスできないようにすることを推奨します。アクセス制限を行う方法は IIS のドキュメントを参照してください。

デザイナー使用時の注意事項

NetCOBOL for .NET で ASP.NET 関連のデザイナーを使用する場合の注意事項は、以下を参照してください。

- デザイナーを利用する場合の共通注意事項
- ASP.NET 関連のデザイナーを利用する場合の注意事項

デバッグ時の注意事項

ASP.NET アプリケーションをデバッグするためには、 ASP.NET アプリケーション構成ファイル（web.config）中の compilation 要素の debug 属性を true に設定する必要があります。 構成ファイルの設定方法に関しては、Visual Studioのドキュメントの ASP.NET 構成設定および system.web 要素 (ASP.NET 設定スキーマ)を参照してください。

開発終了時の注意事項

[ASP.NET Web アプリケーション]または[ASP.NET Web サービス]テンプレートによって作成されたプロジェクトには、 構成ファイル（web.config）が自動的に追加されています。 その構成ファイル中の compilation 要素の debug 属性の初期値は true に設定されています。開発が完了し、ASP.NET アプリケーションを運用する際には、debug 属性の値を false にしてください。

NCOBU01132

Web 構成ファイル(Web.config)の変更

Web サイト(ASP.NET Web アプリケーション、XML Web サービスおよび WCF サービス)において、 対象とする Framework の選択を変更した場合 や NetCOBOL for .NET 以外(Visual C#や Visual Basic)で Web サイトを作成した場合は、Web 構成ファイル(Web.config)の変更が必要になる場合があります。 以下の説明に従って Web 構成ファイルを変更してください。

NetCOBOL for .NET V4.0 以前のバージョンで作成された Web サイトのアップグレード時に「古い.NET Framework を対象とする Web サイトが見つかりました」というタイトルのメッセージに対して「はい」を選択した場合 または Web サイト作成後に対象とする Framework を.NET Framework 2.0、3.0、 または 3.5 から .NET Framework 4.x に変更した場合

Web 構成ファイルを開き、 "Fujitsu.COBOL"という文字列を含む language 属性を検索し、 その language 属性を含む compiler 要素の type 属性に記述されている Version の値を 「 8.0.124.0 」に変更します。 Version の値を変更しない場合の動作は保証できません。

以下は、Web 構成ファイルの変更例です。

```
<compilers>
  <compiler
    extension=".cobx;.cob" language="Fujitsu.COBOL;NetCOBOL;cobol"
    type="Fujitsu.COBOL.COBOLCodeProvider, Fujitsu.COBOL.CodeDom,
    Version=8.0.124.0, Culture=neutral,
    PublicKeyToken=fac0fe3cab973246,
    processorArchitecture=MSIL" compilerOptions="/wc:SCS(ACP)" />
</compilers>
```

NetCOBOL for .NET V4.0 以前のバージョンで作成された Web サイトのアップグレード時に「古い.NET Framework を対象とする Web サイトが見つかりました」というタイトルのメッセージに対して「いいえ」を選択した場合

Web 構成ファイルを開き、<system.codedom>の下位要素である<compilers>要素から、language 属性が「c#;cs;csharp」または「vb;vbs;visualbasic;vbscript」である<compiler>要素をすべて削除します。

以下は、Web 構成ファイルから削除する要素の例です。

```
<compiler language="c#;cs;csharp" extension=".cs"
  type="Microsoft.CSharp.CSharpCodeProvider, System,
  Version=2.0.0.0, Culture=neutral,
  PublicKeyToken=b77a5c561934e089" warningLevel="4">
  <providerOption name="CompilerVersion" value="v3.5"/>
  <providerOption name="WarnAsError" value="false"/>
</compiler>
<compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
  type="Microsoft.VisualBasic.VBCodeProvider, System,
  Version=2.0.0.0, Culture=neutral,
  PublicKeyToken=b77a5c561934e089" warningLevel="4">
  <providerOption name="CompilerVersion" value="v3.5"/>
  <providerOption name="OptionInfer" value="true"/>
  <providerOption name="WarnAsError" value="false"/>
</compiler>
```

Web サイト作成後に対象とする Framework を .NET Framework 4.x から.NET Framework 2.0、3.0、または 3.5 に変更した場合

Web 構成ファイルを開き、次の 2 箇所を変更してください。

1. "Fujitsu.COBOL"という文字列を含む language 属性を検索し、 その language 属性を含む compiler 要素の type 属性に記述されている Version の値を「4.0.111.0」に変更します。 Version の値を変更しない場合の動作は保証できません。

   以下は、Web 構成ファイルの変更例です。

```
<compilers>
  <compiler
    extension=".cobx;.cob" language="Fujitsu.COBOL;NetCOBOL;cobol"
    type="Fujitsu.COBOL.COBOLCodeProvider, Fujitsu.COBOL.CodeDom,
    Version=4.0.111.0, Culture=neutral,
    PublicKeyToken=fac0fe3cab973246,
    processorArchitecture=MSIL" compilerOptions="/wc:SCS(ACP)" />
</compilers>
```

2. <system.codedom> の下位要素である<compilers> 要素から、language 属性が「c#;cs;csharp 」または「vb;vbs;visualbasic;vbscript」である<compiler>要素をすべて削除します。

Visual C#や Visual Basic で作成した Web サイトに NetCOBOL for .NET のソースファイルや Web フォーム等を追加する場合

Web 構成ファイルを開き、<system.codedom>の下位要素である<compilers>要素に、 NetCOBOL for .NET のための<compiler>要素を追加します。

以下は Web 構成ファイルの変更例です。

```
<system.codedom>
  <compilers>
    <compiler extension=".cobx;.cob" language="Fujitsu.COBOL;NetCOBOL;cobol"
              type="Fujitsu.COBOL.COBOLCodeProvider, Fujitsu.COBOL.CodeDom,
                    Version=8.0.124.0, Culture=neutral,
                    PublicKeyToken=fac0fe3cab973246,
                    processorArchitecture=MSIL"
             compilerOptions="/wc:SCS(ACP)"/>
    <compiler language="c#;cs;csharp" extension=".cs"
      type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
                  Microsoft.CodeDom.Providers.DotNetCompilerPlatform,
                  Version=1.0.7.0, Culture=neutral,
                  PublicKeyToken=31bf3856ad364e35"
            warningLevel="4"
            compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>
    <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
      type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
                  Microsoft.CodeDom.Providers.DotNetCompilerPlatform,
                  Version=1.0.7.0, Culture=neutral,
                  PublicKeyToken=31bf3856ad364e35"
            warningLevel="4"
            compilerOptions="/langversion:default /nowarn:41008
                             /define:_MYTYPE=¥"Web¥" /optionInfer+"/>
  </compilers>
</system.codedom>
```

NCOBU00124

Windows アプリケーション開発の注意事項

NetCOBOL for .NET では、 プロジェクトの新規作成時に[Windows アプリケーション]テンプレートを選択することによって、 Windows フォームアプリケーションを簡単に作成することができます。

コントロールライブラリ開発時の注意事項

Windows フォーム向けのコントロールライブラリを開発したい場合は、 [クラスライブラリ]テンプレートを使ってクラスライブラリプロジェクトを作成し、 そこへ「ユーザコントロール」または「カスタムコントロール」項目を追加してください。 その際、 プロジェクトのプロパティとして、翻訳オプション NOALPHAL を忘れずに指定してください。

デザイナー使用時の注意事項

NetCOBOL for .NET で Windows フォームデザイナーを使用する場合の注意事項は、以下を参照してください。

- デザイナーを利用する場合の共通注意事項
- Windows フォームデザイナーを利用する場合の注意事項

特定のバージョンの.NET Framework 向けアプリケーションを開発する際の注意事項

このバージョンの NetCOBOL for .NET では.NET Framework 4 以降を対象としたアプリケーションを開発、運用することができます。

特定のバージョンの.NET Framework を対象としたアプリケーションを開発する際には、以下の点に注意する必要があります。

必要ソフトウェア

以前のバージョンの.NET Framework 向けアプリケーションを開発するためには、別途ソフトウェアが必要な場合があります。

翻訳と実行に使用される NetCOBOL for .NET コンパイラ・ランタイムのバージョン

NetCOBOL for .NET プロジェクトまたは ASP.NET Web サイトのプロパティで、対象のフレームワークとして以前のバージョンの.NET Framework を指定した場合、COBOL ソースの翻訳と COBOL プログラムの実行に、このバージョン(V8.0)とは異なるバージョンの NetCOBOL for .NET コンパイラ・ランタイムが使用される場合があります。

対象とする.NET Framework のバージョンと使用される NetCOBOL for .NET コンパイラ・ランタイムのバージョンの対応は以下のとおりです。

.NET Framework と NetCOBOL for .NET コンパイラ・ランタイムの対応

.NET Framework のバージョン	NetCOBOL for .NET コンパイラ・ランタイムのバージョン
2.0	V4.0
3.0	
3.5	
4	V8.0
4.5	
4.5.1	
4.5.2	
4.6	
4.6.1	
4.6.2	
4.7	
4.7.1	

開発時に使用する NetCOBOL for .NET の選択は統合開発環境が自動的に行います。 ただし、 COBOL ソースの記述や実行環境の設定は、 実際に使用されるコンパイラ・ランタイムを対象としたものにする必要があります。

プロファイル

NetCOBOL for .NET では.NET Framework の特定のプロファイルを対象としたアプリケーションを開発することはできません。 プロファイル指定のない.NET Framework を対象として指定してください。

ツール

ここでは、NetCOBOL for .NET が提供するツールについて説明します。

このセクションの内容

NCOBU00924

開発パッケージに含まれるツール

ここでは、開発パッケージと共にインストールされるツールについて説明します。

ツールの名称	ツールの概要
プログラム原型定義 ウィザード	Windows 版 NetCOBOL で作成されたプログラムを呼び出すためのプログラム原型定義を作成するウィザード
.NET リモート処理プロキシ作成ツール（genrp.exe）	アセンブリの.NET リモート処理プロキシを作成するコンソールコマンド

プログラム原型定義 ウィザード

プログラム原型定義 ウィザードは、Windows 版 NetCOBOL で出力した SAI ファイルの情報を利用し、NetCOBOL for .NET で作成したプログラムから Windows 版 NetCOBOL で作成されたプログラムを呼び出すためのプログラム原型定義を作成します。

プログラム原型定義 ウィザードを起動する方法については、プログラム原型定義の追加を参照してください。また、プログラム原型定義については、COBOL からアンマネージコードの呼出しを参照してください。

このウィザードは、3 つのページで構成されています。これらのページは、ウィザードの下側にある[次へ]ボタン（または、[前へ]ボタン)をクリックすると表示されます。

このセクションの内容

[基本オプション]ページ

プログラム原型定義を作成する場合に必須となるオプションを指定します。

[プログラム] ページ

プログラム原型定義を作成するプログラムを選択します。

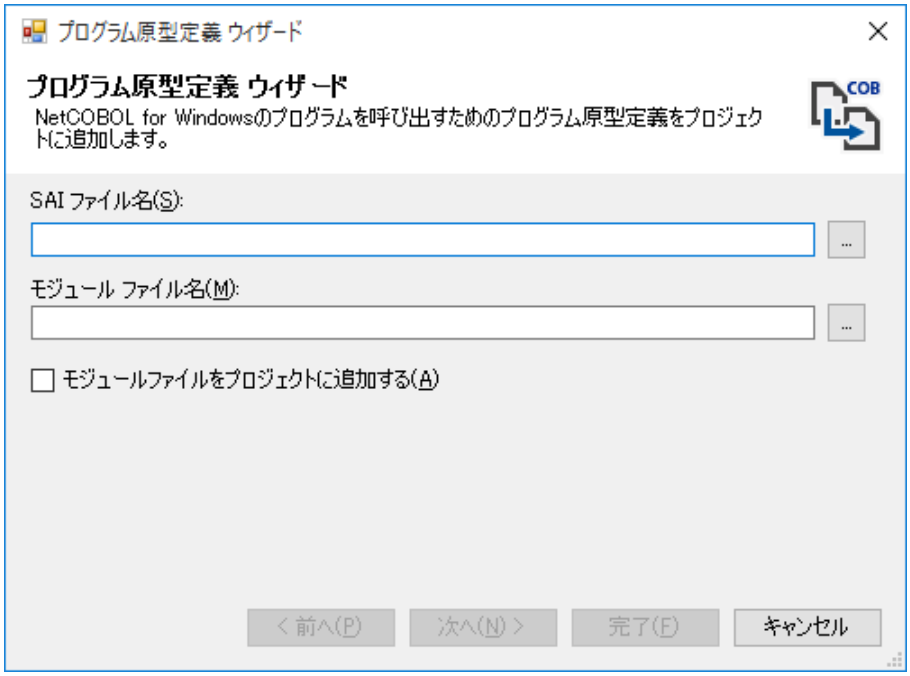
[カスタム オプション] ページ

プログラムごとにカスタマイズする情報を設定します。

NCOBU00152

[基本オプション]ページ

このページでは、プログラム原型定義を作成する上で、必須となるオプションを指定します。



[SAI ファイル名]

Windows 版 NetCOBOL で作成された SAI ファイルを指定します。 SAI ファイル名は必ず指定しなければなりません。SAI ファイルについては、「Windows 版 NetCOBOL ユーザーズガイド」を参照してください。

[モジュール ファイル名]

Windows 版 NetCOBOL で作成されたモジュールファイル名(DLL ファイル名)を指定します。 モジュール ファイル名は必ず指定しなければなりません。

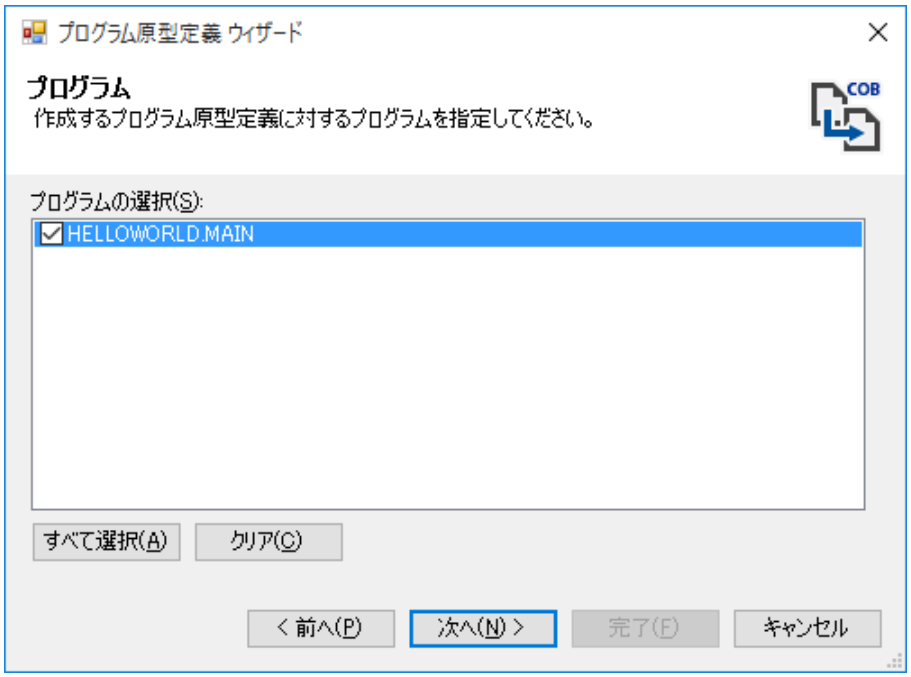
指定されたモジュールファイル名は、 DllImportAttribute カスタム属性のコンストラクタに渡されるパラメタとして出力されます。

[モジュールファイルをプロジェクトに追加する]

[モジュール ファイル名] で指定されているモジュールファイルをプロジェクトに追加します。

[プログラム] ページ

このページでは、プログラム原型定義を作成するプログラムを選択します。指定した SAI ファイル内に複数のプログラム定義がある場合に、作成対象となるプログラムを選択します。



[プログラムの選択]

リストに表示されているプログラムの中からプログラム原型定義を作成するプログラムを、選択（チェックボックスがチェックされた状態）します。 プロラム名をダブルクリック、またはチェックボックスをクリックすることにより、選択状態を変更することができます。

[すべて選択]

すべてのプログラムを選択します。

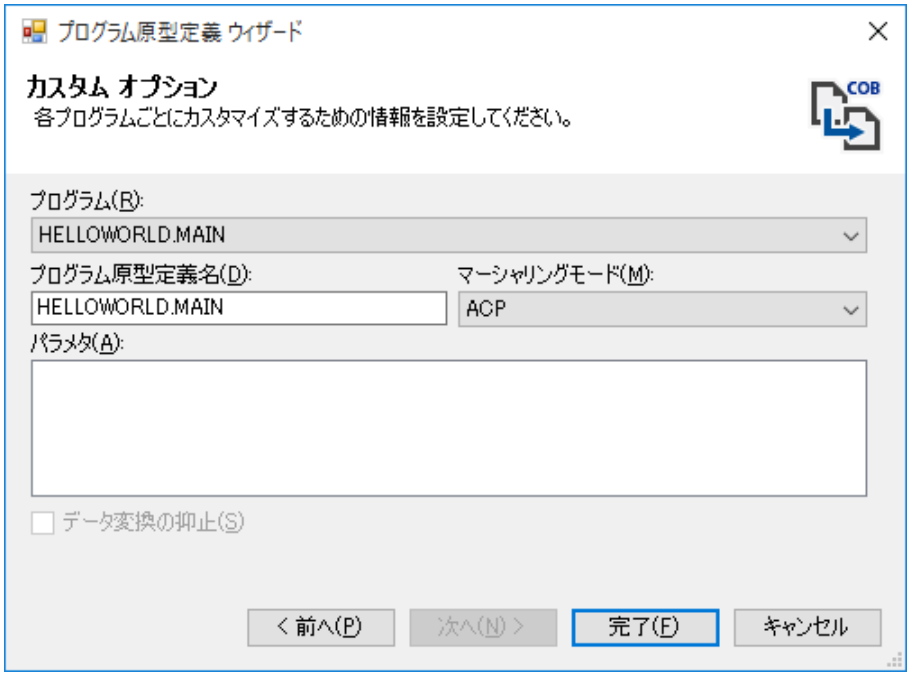
[クリア]

すべてのプログラムの選択を解除します。

NCOBU00154

[カスタム オプション] ページ

このページでは、プログラムごとにカスタマイズするための情報を設定します。



[プログラム]

プログラム原型定義を新規に作成するプログラムの一覧を表示します。選択されたプログラムに対して各オプションを設定することができます。

[プログラム原型定義名]

新規に作成するプログラム原型定義のプログラム名を指定します。 デフォルトでは、Windows 版 NetCOBOL で作成したプログラムの プログラム名が使用されます。

[マーシャリングモード]

実行時の文字列データの変換方法を指定します。 デフォルトは、ACP です。

詳細については、COBOL データのマーシャリングの「文字コード系を指定する」を参照してください。

値	説明
ACP	Windows 版 NetCOBOL で作成した プログラムの実行時文字コードが、ANSI コードページの場合に指定します。
Unicode	Windows 版 NetCOBOL で作成した プログラムの実行時文字コードが、UCS-2 符号化の Unicode の場合に指定します。ただし、半角空白を日本語項目の空白とします。
UnicodeZenkakuPadding	Windows 版 NetCOBOL で作成した プログラムの実行時文

	字コードが、UCS-2 符号化の Unicode の場合に指定します。ただし、日本語の全角空白を日本語項目の空白とします。
ACPTripleSized	Windows 版 NetCOBOL で作成した プログラムの実行時文字コードが、ANSI コードページの場合に指定します。 ただし、プログラムを呼び出す際に、 データ長を 3 分の 1 にします。

[パラメタ]

選択されたプログラムのパラメタ一覧です。

[データ変換の抑止]

Windows版 NetCOBOL で作成した プログラムに渡されるパラメタデータのデータ変換を抑止するかをパラメタごとに指定します。

詳細については、COBOL データのマーシャリングの「特定のパラメタの文字コード変換を抑止する」を参照してください。

NCOBU00167

.NET リモート処理プロキシ作成ツール（genrp.exe）

.NET リモート処理プロキシ作成ツールを利用することによって、 アセンブリの.NET リモート処理プロキシを簡単に作成することができます。

.NET リモート処理プロキシ作成ツールは、 アセンブリからリモート呼出しができる型、 およびリモート呼出しができるメソッドとプロパティの型情報を抜き出したプロキシを生成します。 それ以外の型やメンバーはプロキシに出力されません。

.NET リモート処理プロキシ作成ツールはコンソールコマンドであり、 NetCOBOL for .NET コマンドプロンプトから実行することができます。

```
genrp [オプション] assembly-file
```

引数	説明
assembly-file	.NET リモート処理プロキシを作成したいアセンブリファイルのパスを指定します。

オプション	説明
/TargetFolder:*path*	*path* 部分には生成した.NET リモート処理プロキシを出力するフォルダを指定します。 .NET リモート処理プロキシは元アセンブリと同じ名前になるため、 元アセンブリと同じフォルダを指定することはできません。 このオプションを省略した場合、 元アセンブリが格納されているフォルダの'Proxy'サブフォルダにプロキシが出力されます。
/Report	このオプションが指定されている場合、 ツールが.NET リモート処理プロキシに含めなかった型やメンバーと、 その理由を出力します。
/KeyName:*name*	元アセンブリが厳密な名前をもつ場合に必要です。*name* 部分に元アセンブリを作成するときに利用したキーペアを保持するコンテナを指定します。
/KeyFile:*path*	元アセンブリが厳密な名前をもつ場合に必要です。*path* 部分に元アセンブリを作成するときに利用したキーペアを保持するファイルを指定します。

運用パッケージに含まれるツール

ここでは、運用パッケージと共にインストールされるツールについて説明します。 開発パッケージは運用パッケージのコンポーネントも含むため、 これらのツールは開発パッケージをインストールした際にもインストールされます。

これらのツールは、ランタイムシステムがインストールされるフォルダの下にインストールされます。 ランタイムシステムは Windows がインストールされているドライブの以下の場所にインストールされます。

Windows の種類	ランタイムシステムの種類	インストールフォルダ （Windows がインストールされているドライブ上のパス）
32-bit	32-bit	¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0
64-bit	32-bit	¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0
	64-bit	¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0

運用パッケージと共にインストールされるツールは、以下のプラットフォーム毎に提供される専用のコマンドプロンプトから実行することができます。

x86 プラットフォーム向けのツールを実行する場合

[スタート] > お使いの NetCOBOL for .NET 製品名 > [NetCOBOL for .NET Utilities コマンドプロンプト (x86)]を選択します。

x64 プラットフォーム向けのツールを実行する場合

[スタート] > お使いの NetCOBOL for .NET 製品名 > [NetCOBOL for .NET Utilities コマンドプロンプト (x64)]を選択します。

- 実際には、上記のコマンドプロンプトがすべてインストールされるわけではありません。 インストール先の Windows のバージョンと、インストールされた運用パッケージのエディションに応じて、運用環境上でサポートされるプラットフォーム向けのコマンドプロンプトがインストールされます。
- 管理者の権限を必要とする操作を行う場合は、マウスの右クリックでコンテキストメニューを開き、[管理者として実行]を選択して起動する必要があります。

運用パッケージと共にインストールされるツールは以下のとおりです。

ツールの名称	ツールの概要
COBOL ファイルユーティリティ	COBOL ファイル(レコード順/行順/相対/索引ファイル)を処理するユーティリティ
実行環境設定ユーティリティ	アプリケーション構成ファイルの COBOL の実行環境設定情報を編集するツール

実行環境設定ツール(注 1)	実行用の初期化ファイルの内容を編集するツール
ODBC 情報設定ユーティリティ(注 2)	SQL 情報(ODBC 情報)を ODBC 情報ファイルに設定するユーティリティ
アプリケーション構成ファイル作成コマンド (CBRtoConfig.exe)	実行用の初期化ファイルの情報を読み込んで、アプリケーション構成ファイルを作成するコンソールコマンド

注 1:

NetCOBOL for .NET では、 実行環境情報は、実行環境設定ユーティリティを使用して、アプリケーション構成ファイルに設定することを推奨しています。 従来からある実行用の初期化ファイルは、このツールを使用せずに、実行環境設定ユーティリティを使用して、実行用の初期化ファイルの内容をアプリケーション構成ファイルに変換して使用してください。

注 2:

NetCOBOL for .NET では、 SQL 情報は、実行環境設定ユーティリティを使用して、アプリケーション構成ファイルに設定することを推奨しています。

COBOL ファイルユーティリティ

COBOL ファイルユーティリティは、COBOL ファイルの作成、および保守に役立つ機能を提供します。

COBOL ファイルユーティリティとは

COBOL ファイルユーティリティは、COBOL ファイルシステムが持つファイル操作機能を、COBOL のアプリケーションを介することなくユーティリティのコマンドによって行うためのものです。(以降では、COBOL ファイルシステムが扱うファイル(レコード順/行順/相対/索引ファイル)のことを単に COBOL ファイルといいます。)

COBOL ファイルユーティリティは、COBOL ファイルを処理するためのユーティリティです。具体的には以下の機能を持ちます。

- 各種テキストエディターを使って作成したデータから COBOL ファイルを作成する。
- COBOL ファイルに対する以下の操作を行う。
  - ファイルの複写/移動/削除
  - ファイル構造の変換
  - 索引ファイルの再編成/復旧/属性の表示
- COBOL ファイルのレコードの操作(表示/編集/整列など)を行う。

COBOL ファイルユーティリティでは、 これらの操作を、ウィンドウを使ったメニュー選択で簡単に行うことができます。 また、ウィンドウを使わずに、 コマンドや関数を使用して COBOL ファイルを操作することもできます。 これらの使用方法については、COBOL ファイルユーティリティコマンドおよび COBOL ファイルユーティリティ関数を参照してください。

- COBOL ファイルユーティリティでは、処理の対象となるファイル以外に、ファイルの属性などを定義するファイルは存在しません。
- 各コマンドに指定するファイルの属性およびレコードの属性を誤って指定した場合、内容不良のファイルが作成されることがあります。
- ファイルアクセス中にリセットや電源切断など、強制的に処理を中断させないでください。リセットや電源切断などでファイル処理を中断した場合、処理中のファイルの内容は保証されません。また、このような場合、元のファイルが破壊されたり、作業用ファイルが削除されないことがあります。作業用ファイルが削除されない場合は手動で削除してください。(作業用ファイルについては、COBOL ファイルユーティリティの概要の「環境設定」を参照してください。)
- COBOL ファイルユーティリティで各 COBOL ファイルに対して行うことのできる処理は、COBOL プログラムでのファイル処理と同様に、ファイル編成ごとに異なります。

  たとえば、レコード順ファイルおよび行順ファイルは順呼出しで処理されるため、一定の順序による処理しか行うことができなかったり、レコードの挿入や削除を行うことができなかったりします。 [参照]ファイルの種類
- COBOL ファイルユーティリティのウィンドウを使い、出力ファイルに指定したファイルが既に存在していた場合、そのファイルを上書きするかどうか確認を求めますが、コマンドや関数を使用した場合、上書き確認をせずにエラーとします。
- 以下の環境下において、COBOL ファイルユーティリティを使用する場合、実行環境変数@CBR_FILE_LFS_ACCESS (COBOL ファイルのサイズを拡張する指定)を有効にしてください。
  - 実行環境変数@CBR_FILE_LFS_ACCESS を指定して作成した COBOL ファイルを操

作する

- 実行環境変数@CBR_FILE_LFS_ACCESS 指定が有効な COBOL アプリケーションと COBOL ファイルを共有する

・ 本ユーティリティは、ファイルの高速処理に対応していません。ファイル名に続き「,BSAM」を指定した場合、エラーとします。

・ 本ユーティリティは､操作対象となるファイルが COBOL プログラムや他のユーティリティからアクセスされている場合、エラーとします。 このため、同じファイルに対するユーティリティの同時実行はできません。 なお、索引ファイルの復旧処理では、同時実行した場合の動作は保証されません。 直前の操作が完了したことを確認してから再度実行してください。

このセクションの内容

COBOL ファイルユーティリティの機能

COBOL ファイルユーティリティの機能と操作方法について説明します。

COBOL ファイルユーティリティの使用方法

COBOL ファイルユーティリティの使用方法について説明します。

COBOL ファイルユーティリティコマンド

COBOL ファイルユーティリティ コマンドの使用方法について説明します。

COBOL ファイルユーティリティの機能

ここでは、COBOL ファイルユーティリティの機能について説明します。

以下の表では、COBOL ファイルユーティリティ(*1)の機能概要、ならびに COBOL ファイルユーティリティ関数(*2)、COBOL ファイルユーティリティコマンド(*3)との機能差について示します。

COBOL ファイルユーティリティの機能

機能		概要	*1	*2	*3	備考
COBOL ファイルの操作	創成	テキストエディターで作成したデータを入力して、COBOL ファイルを新規に作成します。	○	○	○	[変換]+[ロード]
	削除	COBOL ファイルを削除します。	○	○	×	[削除]
	移動	COBOL ファイルを別のフォルダに移動します。	○	○	×	[移動]
	複写	COBOL ファイルを複写します。	○	○	×	[複写]
	印刷	COBOL ファイルの内容を印刷します。	○	×	×	[印刷]
内容の編集	レコードの追加	COBOL ファイルにレコードを 1 件ずつ追加します。	○	×	×	[拡張]
		COBOL ファイルに別の COBOL ファイルのレコードを追加します。	○	○	○	[変換]+[ロード](拡張指定)[アンロード]+[ロード](拡張指定)
	レコードの編集	COBOL ファイルのレコードを更新、挿入、削除します。	○	×	×	[編集]
	レコードの整列	キーとなるデータ項目を指定して、COBOL ファイルのレコードを整列します。	○	○	○	[整列]
内容の表示	レコードの表示	COBOL ファイルのレコードの内容を表示します。	○	×	○	[表示]
その他	ファイル構造の変換	COBOL ファイルの構造を別の種類のファイル構造に変換します。	○	○	○	[アンロード]+[ロード]
	索引ファイルの属性情報の表示	索引ファイルの属性情報を表示します。	○	×	○	[属性]
	索引ファイルの属性情報の復旧	アクセスできなくなった索引ファイルを復旧します。	○	○	○	[復旧]

	索引ファイルの再編成	索引ファイル内の未使用空間を削除して、ファイルサイズを小さくします。	○	○	○	[再編成]

操作手順

ここでは、複数のコマンドを合わせて使用する機能について、操作手順を説明します。

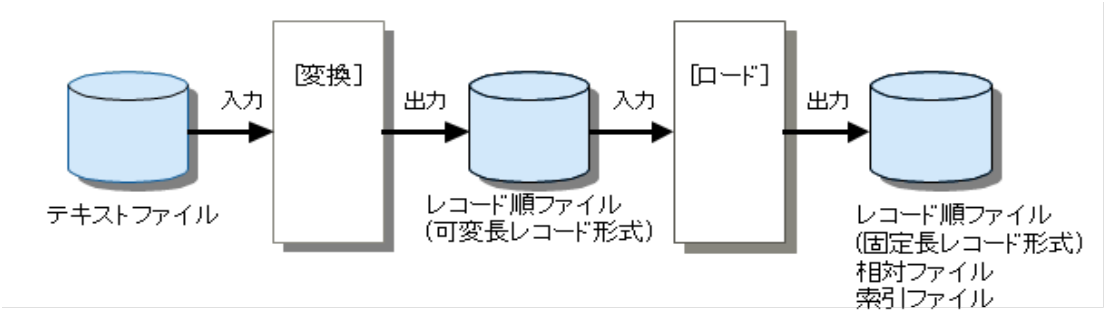
ファイルの創成

1. テキストエディターを使用して、テキスト形式のファイルを作成します。

   テキスト形式のファイルは、COBOL の行順ファイルに相当し、文字表現のデータと 16 進表現のデータで構成されます。テキスト形式の 1 行(改行文字で区切られるデータ)が COBOL ファイルの 1 レコードになります。

2. [変換]コマンドを使用して、テキストファイルからレコード順ファイル(可変長レコード形式)を作成します。

3. レコード順ファイル(固定長レコード形式)、相対ファイル、または、索引ファイルを作成したい場合、可変長レコード形式のレコード順ファイルを入力として、[ロード]コマンドを実行します。



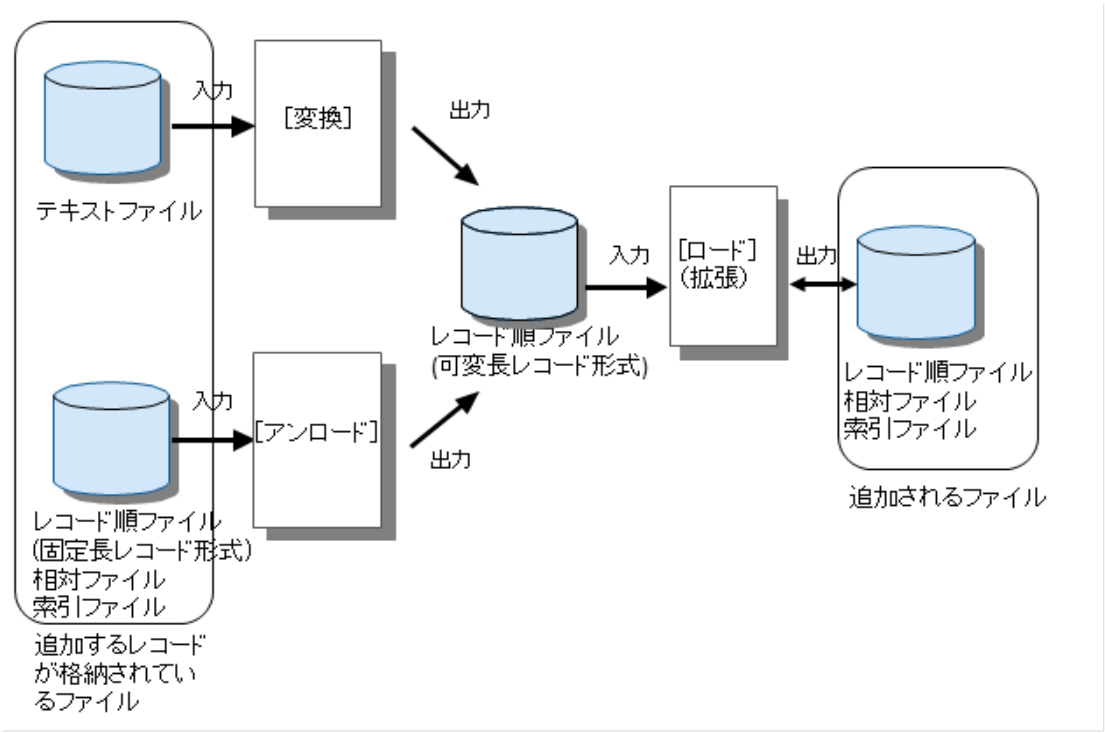
ファイルの拡張(レコードの追加)

1. [変換]コマンド、または[アンロード]コマンドを使用します。

   追加するレコードが格納されているファイルをレコード順ファイル(可変長レコード形式)に変換します。対象ファイルがテキストファイルの場合は、[変換]コマンドを使用し、レコード順ファイル(固定長レコード形式)、相対ファイル、索引ファイルの場合は[アンロード]コマンドを使用します。

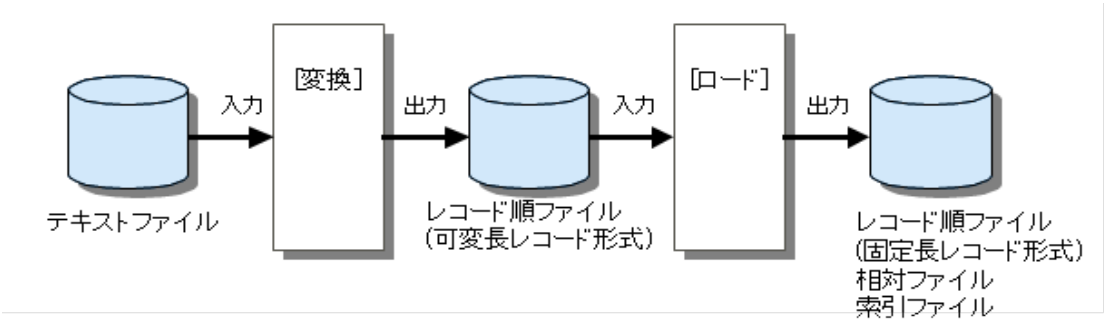
2. [ロード]コマンドの拡張指定を使用します。

   1.で作成したレコード順ファイル(可変長レコード形式)を入力ファイルとします。[ロード]コマンドの拡張指定を使用して、既存ファイルにレコードを追加することができます。



ファイル構造の変換

1. [アンロード]コマンドを使用して、対象ファイルをレコード順ファイル(可変長レコード形式)に変換します。
2. 1.で作成したレコード順ファイル(可変長レコード形式)を入力ファイルとします。[ロード]コマンドを使用してファイル構造の変換を行います。



NCOBU00127

COBOL ファイルユーティリティの使用方法

ここでは、COBOL ファイルユーティリティの概要と各コマンドメニューについて説明します。

このセクションの内容

COBOL ファイルユーティリティの概要

COBOL ファイルユーティリティを使用するための環境設定と起動方法について説明します。

[変換]コマンド

テキストファイルから可変長レコード形式のレコード順ファイルを作成します。また、可変長レコード形式のレコード順ファイルからテキストファイルを作成します。

[ロード]コマンド

可変長レコード形式のレコード順ファイルから、任意の COBOL ファイルを作成します。また、可変長レコード形式のレコード順ファイルの全レコードを、任意の属性のファイルに拡張(レコードの追加)することができます。

[アンロード]コマンド

任意の COBOL ファイルから、可変長レコード形式のレコード順ファイルを作成します。

[表示]コマンド

任意の COBOL ファイルのレコードを 1 件ずつ、画面に表示します。

[印刷]コマンド

任意の COBOL ファイルのレコードを印刷します。

[編集]コマンド

任意の COBOL ファイルのレコードを 1 件ずつ、画面に表示して、レコード内容を編集することができます。

[拡張]コマンド

任意の COBOL ファイルにレコードを 1 件ずつ、追加することができます。

[整列]コマンド

任意の COBOL ファイルのレコードを整列し、その整列結果を可変長レコード形式のレコード順ファイルに格納します。

[属性]コマンド

索引ファイルの属性情報(レコード長など)を画面に表示します。

[復旧]コマンド

アクセスできなくなった索引ファイルを復旧します。

[再編成]コマンド

索引ファイルの未使用空間を削除し、ファイルサイズを小さくします。

[複写]コマンド

ファイルを複写します。

[削除]コマンド

ファイルを削除します。

[移動]コマンド

ファイルを別の場所に移動します。

NCOBU00128

COBOL ファイルユーティリティの概要

ここでは、COBOL ファイルユーティリティを使用するための環境設定と起動方法について説明します。

環境設定

COBOL ファイルユーティリティを起動する前に、以下の設定を行う必要があります。

- 環境変数 TEMP に、作業用一時ファイルを作成するフォルダを、ドライブ名から始まるパス名で設定してください。このフォルダには、COBOL ファイルユーティリティで、[復旧]コマンドを実行した場合、処理するファイルと同じ大きさの一時的な作業用のファイル(-UTYnnnn.TMP(nnnn は英数字を示す))を作成します。
- 環境変数 BSORT_TMPDIR に、整列併合用ファイルを作成するフォルダを、ドライブ名から始まるパス名で設定してください。 このフォルダには、COBOL ファイルユーティリティで、[整列]コマンドを実行した場合、一時的な作業用のファイル(-SRTnnnn.TMP(nnnn は英数字を示す))を作成します。

  環境変数 BSORT_TMPDIR を設定していない場合、環境変数 TEMP に指定されたフォルダに整列併合用ファイルを作成します。

COBOL ファイルユーティリティの起動

Windows 7

COBOL ファイルユーティリティは、[スタート] - [すべてのプログラム] - お使いの NetCOBOL for .NET 製品名 - [NetCOBOL]-[COBOL ファイルユーティリティ]を選択すると表示されます。

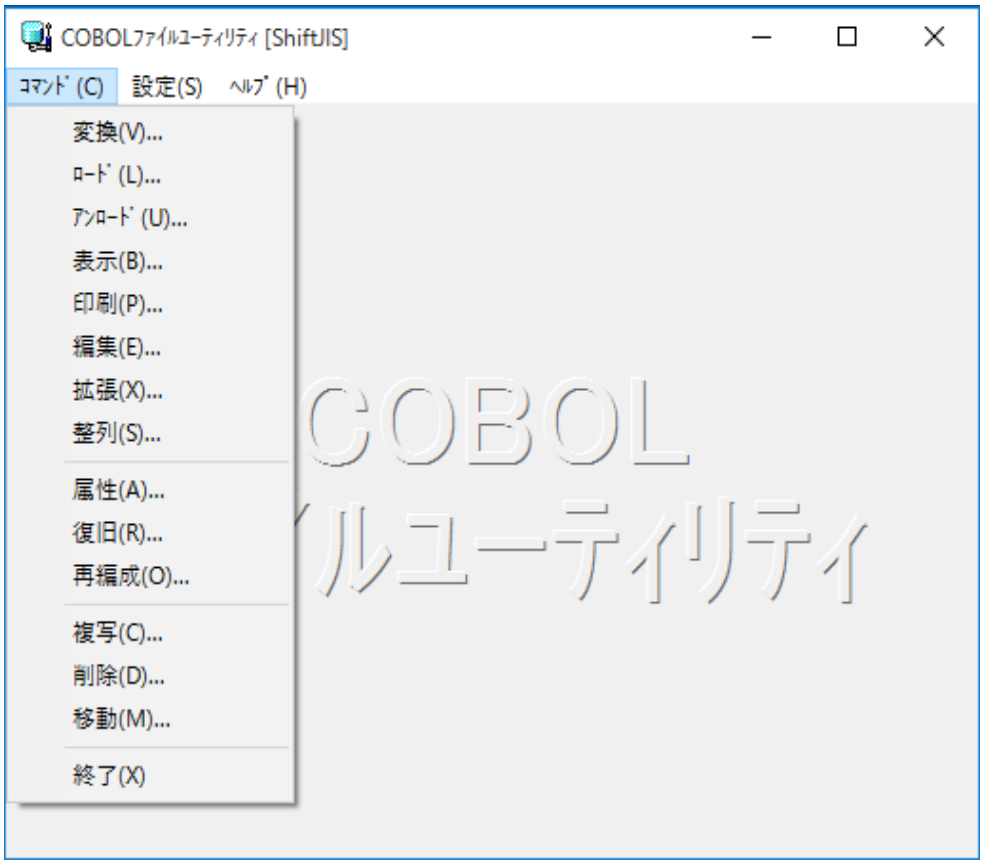
Windows 7 以外

COBOL ファイルユーティリティは、[スタート] - アプリ一覧(*) - お使いの NetCOBOL for .NET 製品名 - [COBOL ファイルユーティリティ]を選択すると表示されます。

* : Windows 8.1 および Windows Server 2012 R2 の場合、[スタート]画面 - [↓] - [アプリ]を操作をすることで同様の画面になります。

処理の選択

[コマンド]メニューから、実行するコマンドを選択します。選択すると、各コマンドを実行するためのダイアログボックスが表示されます。



COBOL ファイルユーティリティの終了

[コマンド]メニューから、[終了]を選択します。

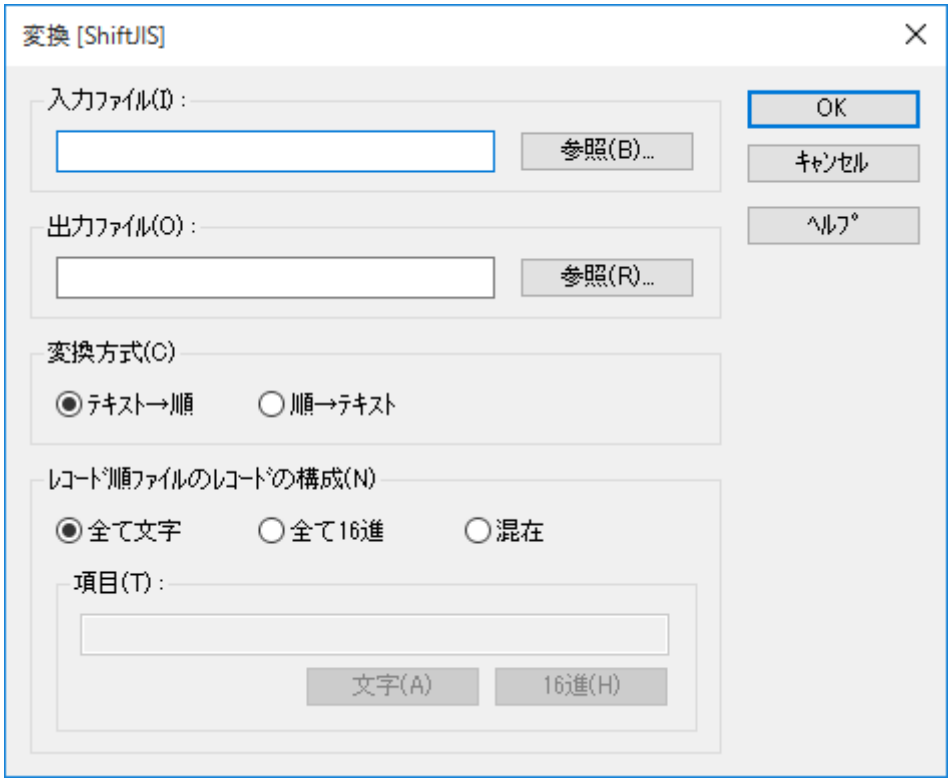
NCOBU00965

[変換]

テキスト形式のファイルから可変長レコード形式のレコード順ファイルを新しく作成(創成)します。また、逆に可変長レコード形式のレコード順ファイルからテキスト形式のファイルを新しく作成(創成)することもできます。

テキスト形式のファイルとは、COBOL の行順ファイルであり、文字表現のデータと 16 進表現のデータで構成されるファイルです。ファイル中の 1 行(改行文字で区切られるデータ)が 1 レコードの意味を持ちます。

文字コードに Unicode を指定した場合、テキスト形式のファイルの文字コードは、UCS2 コード体系として扱います。



[入力ファイル]フィールド

入力ファイル名を指定します。 変換方式にて選択する指定が[テキスト→順]の場合はテキスト形式のファイルのパス名を、[順→テキスト]の場合はレコード順ファイルのパス名を指定してください。

[出力ファイル]フィールド

出力ファイル名を指定します。 変換方式にて選択する指定が[テキスト→順]の場合はレコード順ファイルのパス名を、[順→テキスト]の場合はテキスト形式のファイルのパス名を指定してください。

[変換方式]

変換方式を選択(クリック)します。 テキスト形式のファイルからレコード順ファイルを創成する場合、[テキスト→順]を選択してください。レコード順ファイルからテキスト形式のファイルを創成する場合、[順→テキスト]を選択してください。

[レコード順ファイルのレコードの構成]

レコードの構成を選択します。 文字コードの指定により、選択画面が異なります。

文字コード	ラジオボタン	説明
ShiftJIS JEF(EBCDIC/KANA) JEF(EBCDIC/ASCII)	[全て文字]	レコード中の項目全てが文字表現(*1)のデータの場合に選択してください。
	[全て 16 進]	レコード中の項目全てが 16 進表現(*2)のデータの場合に選択してください。
	[混在]	レコード中に文字表現のデータと 16 進表現のデータが混在する場合に選択してください。
Unicode	[全て UCS2]	レコード中の項目全てが UCS2 のデータの場合に選択してください。
	[全て UTF8]	レコード中の項目全てが UTF8 のデータの場合に選択してください。
	[混在]	レコード中に UCS2 のデータ、UTF8 のデータ、16 進表現のデータが混在する場合に選択してください。

*1: 文字表現のデータとは、文字そのものを表現するデータ(文字データ)です。

*2: 16 進表現のデータとは、バイナリデータを 16 進数字で表現したデータです。バイト境界に注意して指定してください。

【例】

- 3 バイトの文字列「abc」の文字表現は、「abc」です。
- 2 バイトのバイナリ値「5」の 16 進表現は、「0005」です。
- 4 バイトのバイナリ値「3456」の 16 進表現は、「00000d80」です。
- 3 バイトの文字列「abc」、2 バイトのバイナリ値「5」、3 バイトの文字列「999」の連続したデータの、テキスト形式ファイルでの表現は、「abc0005999」となります。

[項目]フィールド

レコードの構成において[混在]ボタンを選択した場合、レコード内の個々の項目を指定します。レコードの先頭からのデータの記述形式が変わるたびに、キーワード文字列とデータ長を指定します。

ボタンをクリックすると、以下のキーワード文字列が表示されます。

文字コード	ボタン	キーワード文字列	説明
ShiftJIS EBCDIC/KANA EBCDIC/ASCII	[文字]	char()	レコード中の項目が文字表現の場合に指定します。
	[16 進]	txtbin()	レコード中の項目が 16 進表現の場合に指定します。
Unicode	[UCS2]	ucs2()	レコード中の項目が UCS2 の場合に指定します。

	[UTF8]	utf8()	レコード中の項目が UTF8 の場合に指定します。
	[16 進]	txtbin()	レコード中の項目が 16 進表現の場合に指定します。

データ長は、キーワード文字列の()内にバイト数で指定してください。16 進表現の場合、バイナリデータとして格納されるデータの長さであり、16 進で表現した文字データとしての数ではありません。

データ形式が混在する場合、指定できるデータ形式の最大個数は 256 です。

JEF 環境について

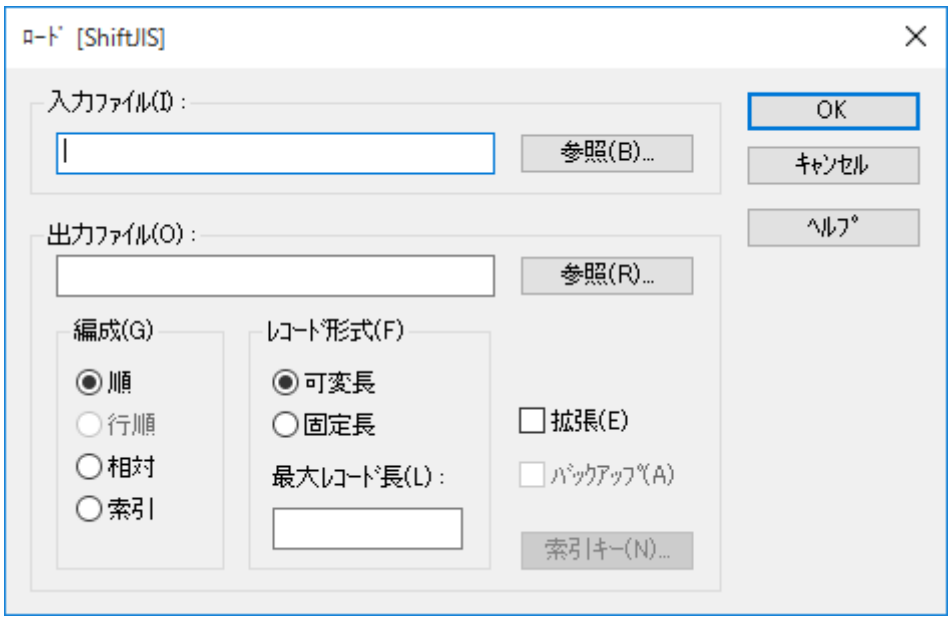
文字コードに EBCDIC/KANA または EBCDIC/ASCII を選択した場合、テキスト形式のファイル中の文字コードを ShiftJIS コード体系として扱い、順ファイル中の文字を JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)のコード体系として扱います。また、[項目]フィールドにキーワード文字列 kanji()を挿入するための[漢字]ボタンが表示されるようになります。

コードの変換は、以下の変換方式およびレコード項目によって、それぞれ変換されます。

変換方式	キーワード文字列	コードの変換
テキスト→順	char()	データを 1 バイト表記の文字コードとして扱い、ASCII から JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)に変換します。
	txtbin()	データを 16 進数文字表記として扱い、特別な文字コード変換はしません。
	kanji()	データを 2 バイト表記の漢字コードとして扱い、SJIS から JEF に変換します。
順→テキスト	char()	データを 1 バイト表記の文字コードとして扱い、JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)から ASCII に変換します。
	txtbin()	データを 16 進数文字表記として扱い、特別な文字コード変換はしません。
	kanji()	データを 2 バイト表記の漢字コードとして扱い、JEF から SJIS に変換します。

[ロード]

可変長レコード形式のレコード順ファイルからレコード順ファイル(固定長レコード形式)/相対ファイル/索引ファイルを新しく作成(創成)します。また、既に存在するレコード順ファイル/相対ファイル/索引ファイルに、可変長レコード形式のレコード順ファイルのデータを追加(拡張)します。



[入力ファイル]フィールド

入力ファイル名を指定します。 創成または拡張するためのデータを格納しているファイルのパス名を指定してください。指定するファイルは、可変長レコード形式のレコード順ファイルでなければなりません。

[出力ファイル]フィールド

出力ファイル名を指定します。 創成または拡張するファイルのパス名を指定してください。

[編成]

出力ファイルのファイル編成を選択します。 ファイル編成の[順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

出力ファイルのレコード形式を選択します。 レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、[拡張]指定でファイル編成が[索引]の場合は指定できません。

[最大レコード長]フィールド

出力ファイルの最大レコード長を指定します。 創成または拡張するファイルの最大レコード長を指定してください。入力ファイルに出力ファイルの最大レコード長より大きいレコードが存在する場合、コマンドの実行時にエラーとなります。なお、[拡張]指定でファイル編成が[索引]の場合は指定できません。

[拡張]チェックボックス

ファイルの拡張を選択します。 ファイルを拡張する場合、チェックボックス[拡張]をクリックしてください。

[バックアップ]チェックボックス

バックアップ作成を選択します。 ファイルを拡張する場合、処理するファイルのバックアップファイルを作成することができます。バックアップファイルを作成する、または作成しないの選択は、[バックアップ]をクリックして行います。チェックしている場合、バックアップファイルを作成します。バックアップファイルは、処理前のファイルと同じフォルダに、ファイル名の拡張子を BAK として作成します。

[ロード]コマンドで[拡張]を指定した場合、直接ファイルにレコードを追加するため、バックアップファイルを作成することをお勧めします。

[索引キー]

ファイル編成で[索引]を選択した場合、索引キー情報を指定します。 [索引キー...]ボタンをクリックして[索引キー指定]ダイアログで索引キー情報を指定します。なお、[拡張]を指定した場合は指定できません。

[索引キー指定]ダイアログ

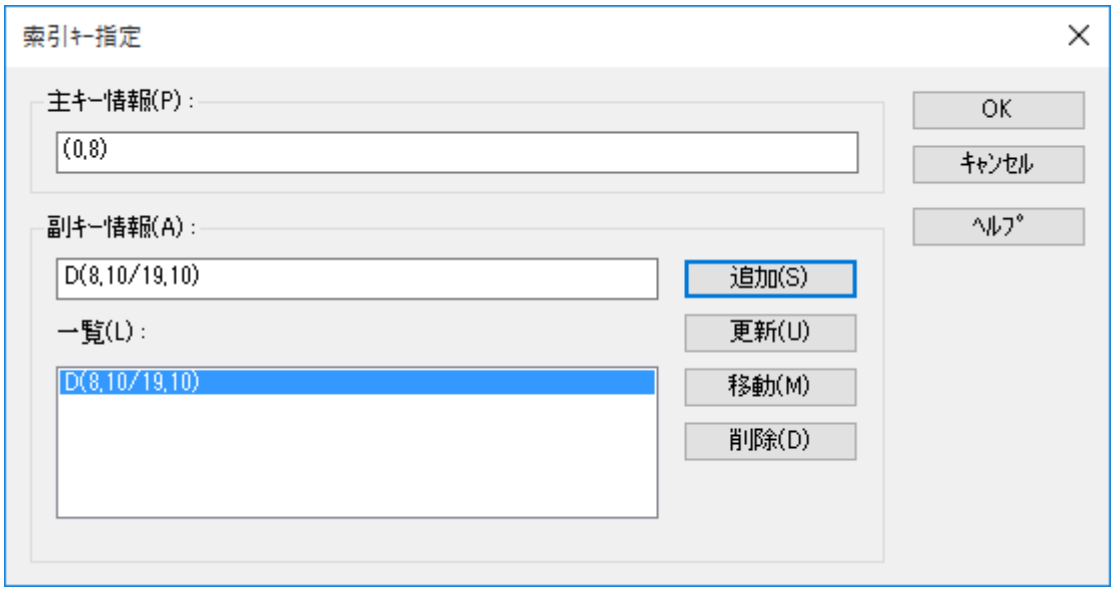
索引ファイルのレコードキー情報を指定します。指定する情報は、キーとするデータ項目のレコード内の位置と長さ、および他のキーと重複するデータの有無です。

キー情報の記述形式

[D](オフセット,長さ[,N][/オフセット,長さ[,N]]…)

注意:“[]”は省略可能。“…”は繰り返し指定可能。

D	キーとするデータ項目が他のキーと重複する場合に指定します。WITH DUPLICATES 指定に相等します。
オフセット	キーとするデータ項目のレコードの先頭からの相対位置(0 バイトから始まるバイト数)を指定します。
長さ	キーとするデータ項目の長さをバイト数で指定します。
N	UCS2 データをキーとする場合に指定します。(文字コードに Unicode を指定した場合にだけ指定できます。)
/	1 つのキーとして、非連続な複数のデータ項目を指定する場合、「/」で区切って指定します。



[主キー情報]フィールド

主キーの設定･更新を行います。 キー情報の記述形式に従い、キー情報を直接入力してください。

[副キー情報]フィールド

副キー情報の入力を行います。

[一覧]リストボックス

指定された副キーの一覧が表示されます。

[追加]ボタン

副キーの設定を行います。 キー情報の記述形式に従い、[副キー情報]フィールドにキー情報を直接入力し、[追加]ボタンをクリックします。 設定したキー情報は[一覧]リストボックスに表示されます。

[更新]ボタン

副キーの更新を行います。 [一覧]リストボックス内の対象とするキー情報をクリックして[副キー情報]フィールドに表示させます。副キー情報]フィールドで値を更新後、[更新]ボタンをクリックします。

[移動]ボタン

副キーの移動を行います。 [一覧]リストボックス内の対象とするキー情報をクリックして[副キー情報]フィールドに表示させます。[移動]ボタンをクリックし、[一覧]リストボックス内の先頭 0 から始まる相対値で移動先を指定します。

[削除]ボタン

副キーの削除を行います。 [一覧]リストボックス内の対象とするキー情報をクリックして[副キー情報]フィールドに表示させます。[削除]ボタンをクリックします。

キー情報の記述例

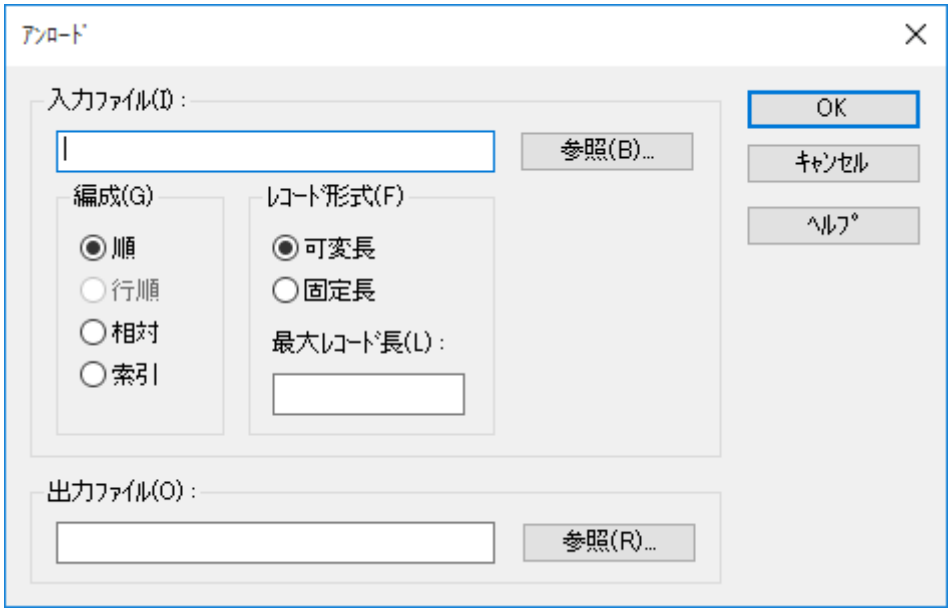
以下の COBOL アプリケーションで使用される索引ファイルのキー情報を設定する方法を説明します。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. SAGYOU.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
         SELECT IXDFILE ASSIGN TO FILENAME
           ORGANIZATION IS INDEXED
           ACCESS MODE  IS DYNAMIC
           RECORD KEY   IS SERIAL-NUMBER
           ALTERNATE RECORD KEY IS F-NAME M-NAME WITH DUPLICATES.
      DATA DIVISION.
      FILE SECTION.
      FD IXDFILE.
      01 IXDFILE-RECORD.
        02 SERIAL-NUMBER PIC 9(8).
         02 F-NAME       PIC X(10).
        02 FILLER       PIC X.
        02 M-NAME       PIC X(10).
```

- 主キー情報]フィールドに (0,8) を記述します。
- [副キー情報]フィールドに D(8,10/19,10) を記述し、[追加]ボタンをクリックして、[一覧]リストボックスに追加します。

[アンロード]

レコード順ファイル/相対ファイル/索引ファイルから可変長レコード形式のレコード順ファイルを新しく作成(創成)します。



[入力ファイル]フィールド

入力ファイル名を指定します。 アンロードするファイルのパス名を指定してください。 指定するファイルは、レコード順ファイル、相対ファイルまたは索引ファイルでなければなりません。

[編成]

入力ファイルのファイル編成を選択します。 ファイル編成の[順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

ファイル編成で[順]または[相対]を選択した場合、入力ファイルのレコード形式を選択します。レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、索引ファイルの場合は指定できません。

[最大レコード長]フィールド

ファイル編成で[順]または[相対]を選択した場合、入力ファイルの最大レコード長を指定します。入力ファイルの最大レコード長を指定してください。 固定長レコード形式の場合、レコード長を指定します。 指定したレコード長が実際のファイルと異なると、正しく処理できません。なお、索引ファイルの場合は指定できません。

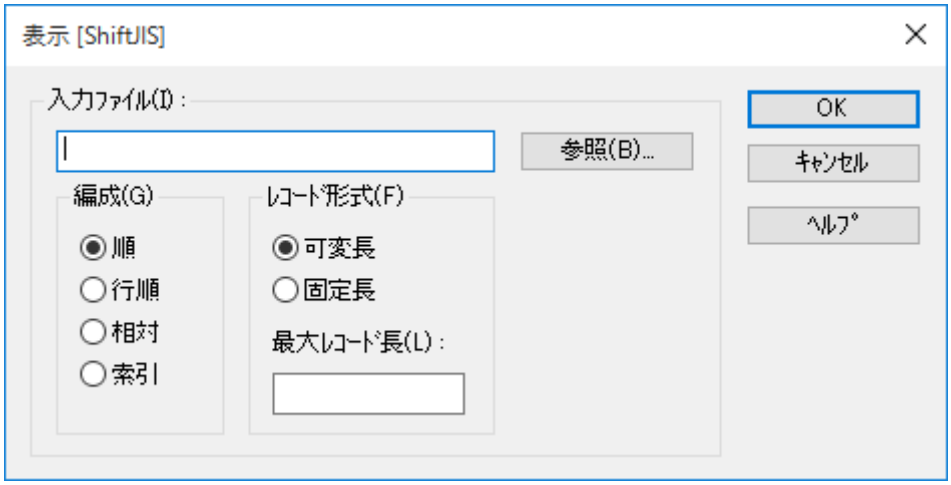
[出力ファイル]フィールド

出力ファイル名を指定します。 新しく作成(創成)するファイルのパス名を指定してください。

NCOBU00968

[表示]

ファイルの内容をレコード単位に文字と 16 進数で表示します。



[入力ファイル]フィールド

入力ファイル名を指定します。 表示するファイルのパス名を指定してください。指定するファイルは、レコード順ファイル、行順ファイル、相対ファイルまたは索引ファイルでなければなりません。

[編成]

入力ファイルのファイル編成を選択します。 ファイル編成の[順]/[行順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

ファイル編成で[順]、[行順]または[相対]を選択した場合、入力ファイルのレコード形式を選択します。レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、索引ファイルの場合は指定できません。

[最大レコード長]フィールド

ファイル編成で[順]、[行順]または[相対]を選択した場合、入力ファイルの最大レコード長を指定します。入力ファイルの最大レコード長を指定してください。固定長レコード形式の場合、レコード長を指定します。指定したレコード長が実際のファイルと異なると、正しく表示できません。なお、索引ファイルの場合は指定できません。

[OK]ボタン

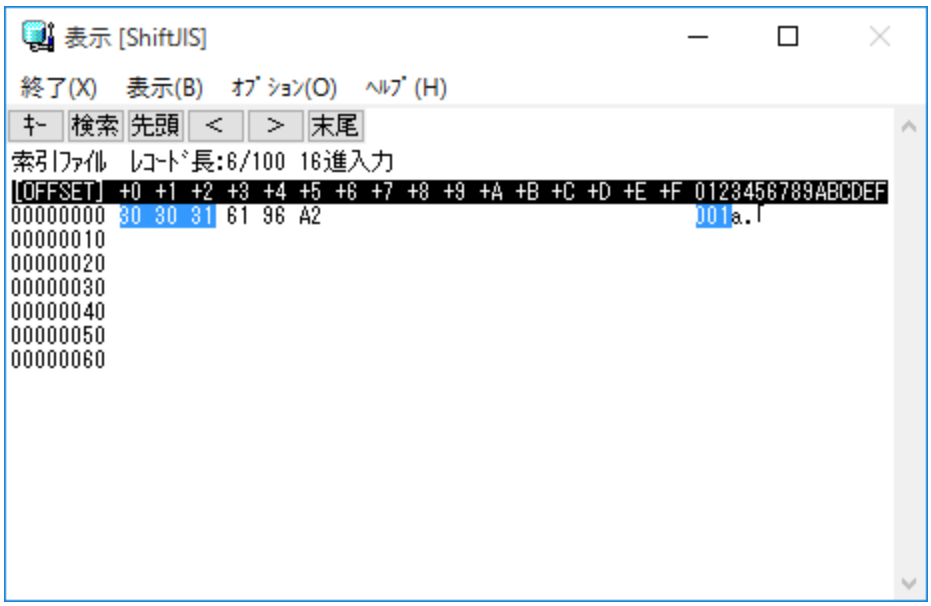
レコードを表示します。 指定完了後、[OK]ボタンをクリックして、レコードを表示します。操作方法については、画面の操作(表示)を参照してください。

画面の操作(表示)

メニューバーまたはボタンから、操作を選択することができます。

メニューバー		ボタン	機能
終了		―	表示ウィンドウを終了します。
表示	索引キー選択	キー	レコードを検索するための索引キーを指定します。キーの指定は、ダイアログボックスで選択します。指定されたキーは、画面上に青色で表示されます。画面に表示される順番は、選択した索引キーの順番に依存するようになります。この指定は、索引ファイルに指定できます。
	レコード検索	検索	レコードを検索するためのキー値を指定します。 相対ファイルは、ダイアログボックスで相対レコード番号をキー値として入力します。索引ファイルは、画面に表示されているレコードのキー領域(青色で表示されている領域)に値を入力後、選択(クリック)してください。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	先頭レコード	先頭	ファイル中に存在する論理的な先頭レコードを表示します。相対ファイルは、相対レコード番号が最小のレコードを表示します。索引ファイルは、指定されているキーのキー値が最小のレコードを表示します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	前レコード	<	現在表示されているレコードの論理的に前に位置するレコードを表示します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	次レコード	>	現在表示されているレコードの論理的に次に位置するレコードを表示します。
	末尾レコード	末尾	ファイル中に存在する論理的に最後のレコードを表示します。相対ファイルは、相対レコード番号が最大のレコードを表示します。索引ファイルは、指定されているキーのキー値が最大のレコードを表示します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
オプション	16 進入力	―	データを入力する位置にカーソルを位置付けて 16 進文字で入力します。
	文字入力	―	データを入力する位置にカーソルを位置付けて英数字および 1 バイトカナ文字で入力します。
	フォント指定	―	画面のフォントを変更できます。ダイアログボックスを使用します。

レコード操作画面



索引ファイル(上記画面の場合)

表示中のファイルの属性を示しています。(順ファイル/行順ファイル/相対ファイル/索引ファイル)

レコード長

表示中のレコードのレコード長/最大レコード長を示しています。

16 進入力

レコード操作画面へのデータ入力方法を示しています。(16 進入力/文字入力)

[OFFSET]の下の文字列および“+0 +1 … +E +F”

画面に表示されているレコードデータ内の位置のオフセットを 16 進数で示しています。

“+0 +1 … +E +F”の下の 2 個の文字列

レコードデータを 1 バイトごとに 16 進数で表示しています。

“0123456789ABCDEF”の下の文字列

レコードデータを 1 バイト表記の文字で表示しています。 なお、英数字および 1 バイトカナ文字(0x20～0x7E、0xA1～0xDF)以外のデータは、ピリオドに置き換えて表示されます。

JEF 環境について

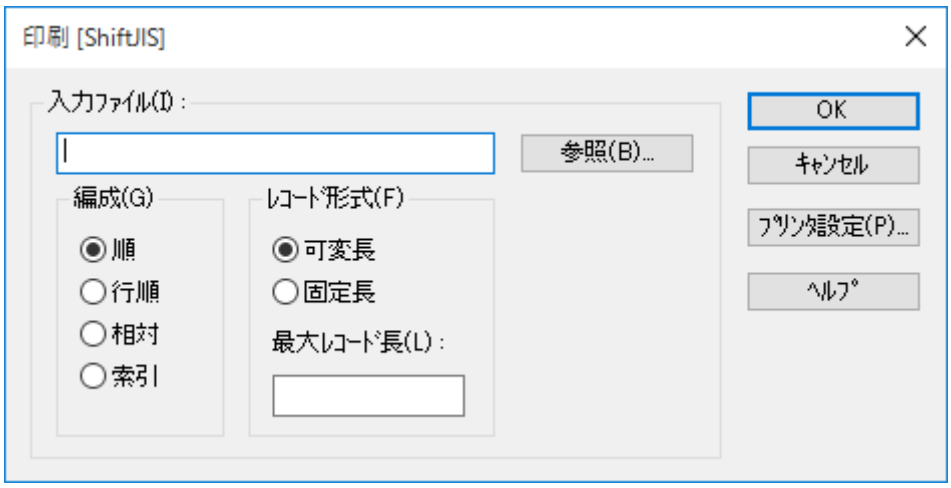
ファイル中のコードを JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)のコードとして扱います。

文字コード	表示
JEF(EBCDIC/KANA)	英数字および 1 バイトカナ文字以外のデータは、ピリオドに置き換えて表示されます。
JEF(EBCDIC/ASCII)	英数字および 1 バイト英小文字以外のデータは、ピリオドに置き換えて表示されます。

NCOBU00969

[印刷]

ファイルの内容をレコード単位に文字と 16 進数で印刷します。英数字および 1 バイトカナ文字(0x20～0x7E、0xA1～0xDF)以外のデータは、ピリオドに置き換えて印刷されます。また、印刷範囲をレコード単位で指定することができます。



[入力ファイル]フィールド

入力ファイル名を指定します。 印刷するファイルのパス名を指定してください。指定するファイルは、レコード順ファイル、行順ファイル、相対ファイルまたは索引ファイルでなければなりません。

[編成]

入力ファイルのファイル編成を選択します。 ファイル編成の[順]/[行順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

入力ファイルのレコード形式を選択します。 レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、索引ファイルの場合は指定できません。

[最大レコード長]フィールド

入力ファイルの最大レコード長を指定します。 入力ファイルの最大レコード長を指定してください。固定長レコード形式の場合、レコード長を指定します。指定したレコード長が実際のファイルと異なると、正しく表示できません。なお、索引ファイルの場合は指定できません。

[プリンタ設定]ボタン

プリンタの設定を行います。 [プリンタ設定]ボタンをクリックして、ダイアログボックスでプリンタの設定を行います。設定方法は、システムにおけるプリンタの設定と同様です。

[OK]ボタン

印刷範囲を指定します。 [OK]ボタンをクリックして、表示される[印刷範囲指定]ダイアログで印刷範囲を指定してください。

JEF 環境について

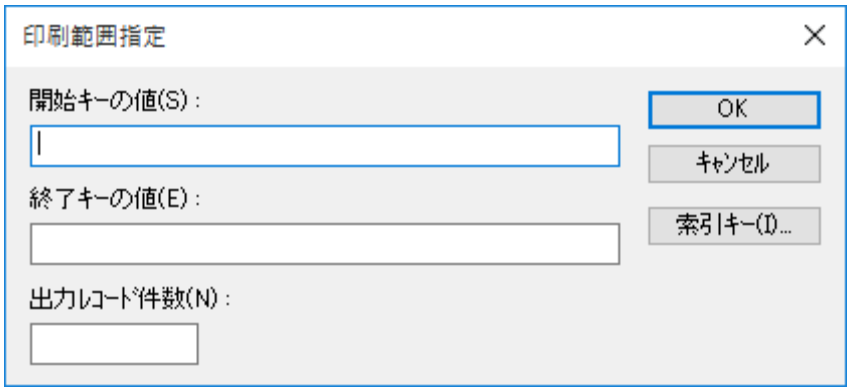
ファイル中のコードを JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)のコードとして扱います。

文字コード	表示
JEF(EBCDIC/KANA)	英数字および 1 バイトカナ文字以外のデータは、ピリオドに置き換えて表示されます。
JEF(EBCDIC/ASCII)	英数字および 1 バイト英小文字以外のデータは、ピリオドに置き換えて表示されます。

[印刷範囲指定]ダイアログ

レコード位置と印刷件数で印刷範囲を指定します。両方指定した場合、最初に成立する条件の範囲で印刷します。印刷範囲の指定を省略した場合、ファイル中の全レコードを印刷します。

索引ファイルの場合



[索引キー]ボタン

印刷範囲として指定する索引キーを選択します。 [索引キー]ボタンをクリックし、表示されたキー情報リストから適切なキーを選択してください。

[開始キーの値]フィールド

印刷を開始するキーの値(全角文字は不可)を指定します。

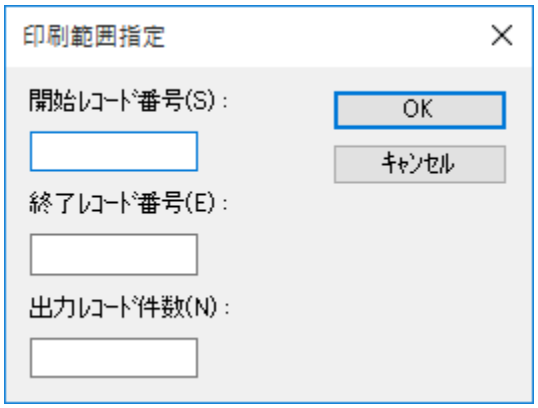
[終了キーの値] フィールド

印刷を終了するキーの値(全角文字は不可)を指定します。

[出力レコード件数] フィールド

印刷するレコード件数を指定します。

レコード順ファイル、行順ファイル、相対ファイルの場合



[開始レコード番号]フィールド

印刷を開始するレコード番号を指定します。

[終了レコード番号] フィールド

印刷を終了するレコード番号を指定します。

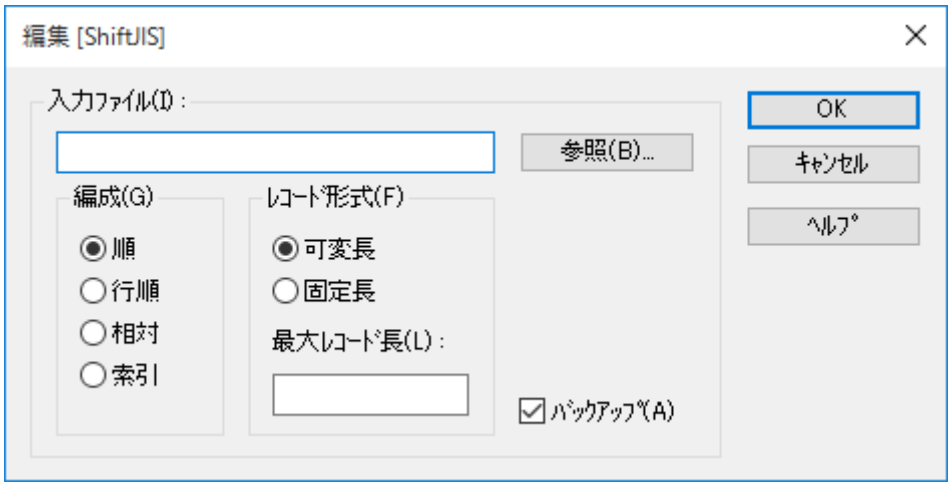
[出力レコード件数] フィールド

印刷するレコード件数を指定します。

開始レコード番号および終了レコード番号に指定するレコード番号は、レコード順ファイル／行順ファイルの場合、ファイル中のレコードの格納順序で指定します。相対ファイルの場合、相対レコード番号で指定します。

[編集]

ファイルの内容をレコード単位で編集します。文字と 16 進数で表示されるレコードを編集し、挿入/削除/更新を行います。



[入力ファイル]フィールド

入力ファイル名を指定します。 編集するファイルのパス名を指定してください。指定するファイルは、レコード順ファイル、行順ファイル、相対ファイルまたは索引ファイルでなければなりません。

[編成]

入力ファイルのファイル編成を選択します。 ファイル編成の[順]/[行順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

入力ファイルのレコード形式を選択します。 レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、索引ファイルの場合は指定できません。

[最大レコード長]フィールド

入力ファイルの最大レコード長を指定します。 入力ファイルの最大レコード長を指定してください。固定長レコード形式の場合、レコード長を指定します。指定したレコード長が実際のファイルと異なると、正しく表示/編集できません。なお、索引ファイルの場合は指定できません。

[バックアップ]チェックボックス

バックアップ作成を選択します。 処理するファイルのバックアップファイルを作成することができます。バックアップファイルを作成する、または作成しないの選択は、[バックアップ]をクリックして行います。チェックしている場合、バックアップファイルを作成します。バックアップファイルは、処理前のファイルと同じフォルダに、ファイル名の拡張子を.BAK として作成します。

[編集]コマンドは、直接ファイルにレコードを編集するため、バックアップファイルを作成することをお勧めします。

[OK]ボタン

レコードを編集します。 指定完了後、[OK]ボタンをクリックしてください。レコードが表示されるので、編集作業を行います。操作方法については、画面の操作(編集)を参照してください。

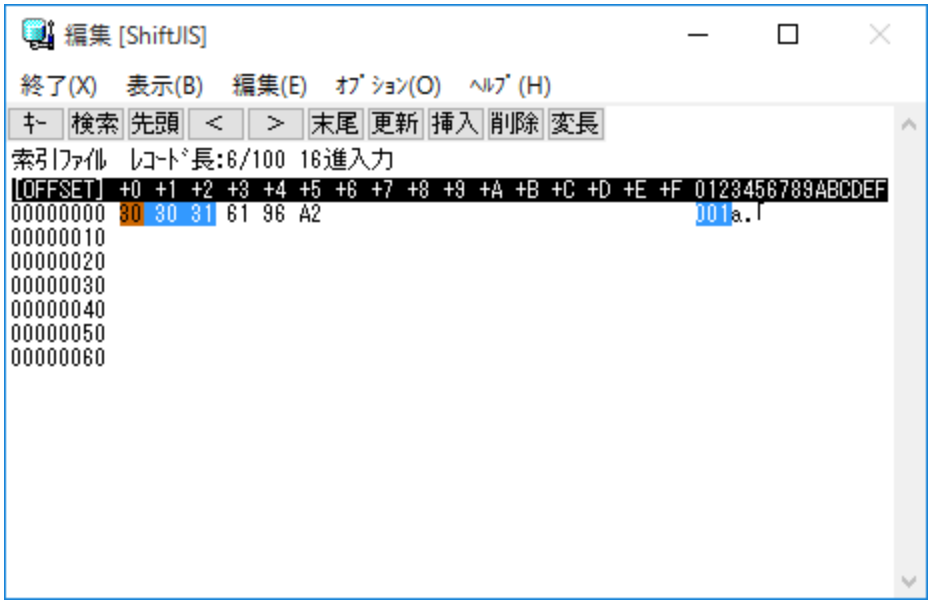
画面の操作(編集)

メニューバーまたはボタンから、操作を選択することができます。

メニューバー		ボタン	機能
終了		―	編集ウィンドウを終了します。
表示	索引キー選択	キー	レコードを検索するための索引キーを指定します。キーの指定は、ダイアログボックスで選択します。指定されたキーは、画面上に青色で表示されます。画面に表示される順番は、選択した索引キーの順番に依存するようになります。この指定は、索引ファイルに指定できます。
	レコード検索	検索	レコードを検索するためのキー値を指定します。相対ファイルは、ダイアログボックスで相対レコード番号をキー値として入力します。索引ファイルは、画面に表示されているレコードのキー領域(青色で表示されている領域)に値を入力後、選択(クリック)してください。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	先頭レコード	先頭	ファイル中に存在する論理的な先頭レコードを表示します。相対ファイルは、相対レコード番号が最小のレコードを表示します。索引ファイルは、指定されているキーのキー値が最小のレコードを表示します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	前レコード	<	現在表示されているレコードの論理的に前に位置するレコードを表示します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	次レコード	>	現在表示されているレコードの論理的に次に位置するレコードを表示します。
	末尾レコード	末尾	ファイル中に存在する論理的に最後のレコードを表示します。相対ファイルは、相対レコード番号が最大のレコードを表示します。索引ファイルは、指定されているキーのキー値が最大のレコードを表示します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
編集	レコード更新	更新	画面に表示、変更したレコードを更新する場合に選択します。この指定は、レコード順ファイル、相対ファイルと索引ファイルで行えます。ただし、レコード順ファイルは、1 レコードに対して一度だ

			け更新が可能です。一度更新したレコードを再度更新する場合、対象レコードを再度表示し直してください。
	レコード挿入	挿入	画面に表示､変更したレコードをファイルに追加する場合に選択します。相対ファイルは、挿入する位置として相対レコード番号をダイアログボックスに指定します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	レコード削除	削除	画面に表示したレコードを削除します。この指定は、相対ファイルと索引ファイルの場合に指定できます。
	レコード変長	変長	画面に表示したレコードのレコード長を変更します。レコード長をダイアログボックスに指定します。この指定は、可変長レコード形式の相対ファイルと索引ファイルの場合に指定できます。
オプション	16 進入力	―	データを入力する位置にカーソルを位置付けて 16 進文字で入力します。
	文字入力	―	データを入力する位置にカーソルを位置付けて英数字および 1 バイトカナ文字で入力します。
	フォント指定	―	画面のフォントを変更できます。ダイアログボックスを使用します。

レコード操作画面



索引ファイル(上記画面の場合)

表示中のファイルの属性を示しています。(順ファイル/行順ファイル/相対ファイル/索引ファイル)

レコード長

表示中のレコードのレコード長/最大レコード長を示しています。

16 進入力

レコード操作画面へのデータ入力方法を示しています。(16 進入力/文字入力)

[OFFSET]の下の文字列および“+0 +1 … +E +F”

画面に表示されているレコードデータ内の位置のオフセットを 16 進数で示しています。

“+0 +1 … +E +F”の下の 2 個の文字列

レコードデータを 1 バイトごとに 16 進数で表示しています。

“0123456789ABCDEF”の下の文字列

レコードデータを 1 バイト表記の文字で表示しています。 なお、英数字および 1 バイトカナ文字(0x20～0x7E、0xA1～0xDF)以外のデータは、ピリオドに置き換えて表示されます。

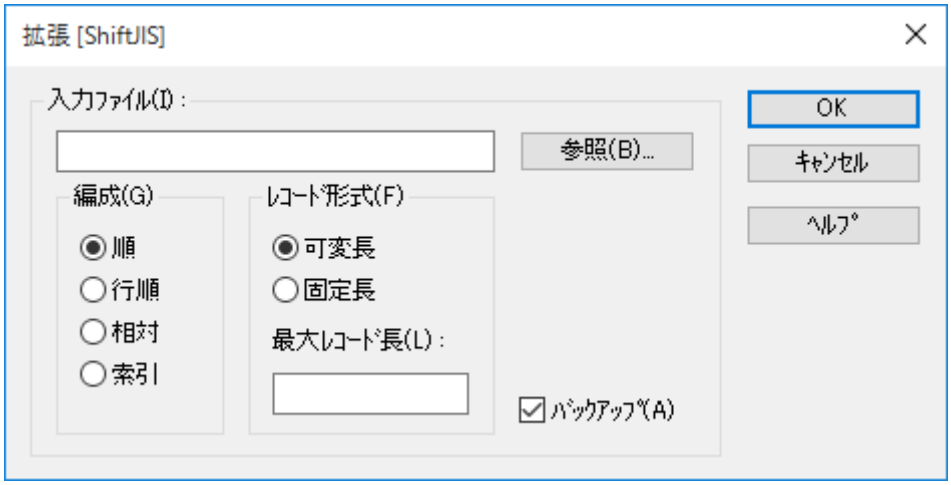
JEF 環境について

ファイル中のコードを JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)のコードとして扱います。

文字コード	表示
JEF(EBCDIC/KANA)	英数字および 1 バイトカナ文字以外のデータは、ピリオドに置き換えて表示されます。
JEF(EBCDIC/ASCII)	英数字および 1 バイト英小文字以外のデータは、ピリオドに置き換えて表示されます。

[拡張]

ファイルにレコードを追加します。レコードは英数字、1 バイトカナ文字および 16 進数で作成します。



[入力ファイル]フィールド

入力ファイル名を指定します。 拡張するファイルのパス名を指定してください。指定するファイルは、レコード順ファイル、行順ファイル、相対ファイルまたは索引ファイルでなければなりません。

[編成]

入力ファイルのファイル編成を選択します。 ファイル編成の[順]/[行順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

入力ファイルのレコード形式を選択します。 レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、索引ファイルの場合は指定できません。

[最大レコード長]フィールド

入力ファイルの最大レコード長を指定します。 固定長レコード形式の場合、レコード長を指定します。指定したレコード長が実際のファイルと異なると、正しく表示できません。なお、索引ファイルの場合は指定できません。

[バックアップ]チェックボックス

処理するファイルのバックアップファイルを作成することができます。 バックアップファイルは、処理前のファイルと同じフォルダに、ファイル名の拡張子を BAK として作成します。

[拡張]コマンドは、直接ファイルにレコードを追加するため、バックアップファイルを作成することをお勧めします。

[OK]ボタン

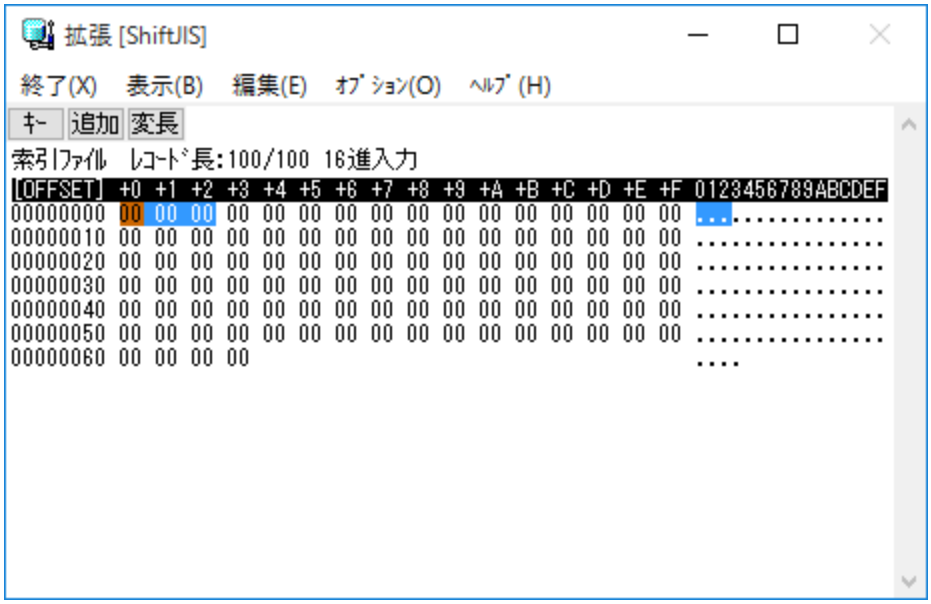
ファイルを拡張します。 指定完了後、[OK]ボタンをクリックすると、レコード操作画面が表示されます。操作方法については、画面の操作(拡張)を参照してください。

画面の操作(拡張)

メニューバーまたはボタンから、操作を選択することができます。

メニューバー		ボタン	機能
終了		―	拡張ウィンドウを終了します。
表示	索引キー選択	キー	キーの指定は、ダイアログボックスで選択します。指定されたキーは、画面上に青色で表示されます。この指定は、索引ファイルの場合に指定できます。
編集	レコード追加	追加	画面上のレコードをファイルに追加します。索引ファイルは、最大キー値より小さいキー値のレコードを追加できます。
	レコード変長	変長	画面に表示したレコードのレコード長を変更します。レコード長をダイアログボックスに指定します。この指定は、可変長レコード形式の相対ファイルと索引ファイルの場合に指定できます。
オプション	16 進入力	―	データを入力する位置にカーソルを位置付けて 16 進文字で入力します。
	文字入力	―	データを入力する位置にカーソルを位置付けて英数字および 1 バイトカナ文字で入力します。
	フォント指定	―	画面のフォントを変更できます。ダイアログボックスを使用します。

レコード操作画面



索引ファイル

表示中のファイルの属性を示しています。(順ファイル/行順ファイル/相対ファイル/索引ファイル)

レコード長

表示中のレコードのレコード長/最大レコード長を示しています。

16 進入力

レコード操作画面へのデータ入力方法を示しています。(16 進入力/文字入力)

[OFFSET]の下の文字列および“+0 +1 … +E +F”

画面に表示されているレコードデータ内の位置のオフセットを 16 進数で示しています。

“+0 +1 … +E +F”の下の 2 個の文字列

レコードデータを 1 バイトごとに 16 進数で表示しています。

“0123456789ABCDEF”の下の文字列

レコードデータを 1 バイト表記の文字で表示しています。 なお、英数字および 1 バイトカナ文字(0x20～0x7E、0xA1～0xDF)以外のデータは、ピリオドに置き換えて表示されます。

JEF 環境について

ファイル中のコードを JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)のコードとして扱います。

文字コード	表示
JEF(EBCDIC/KANA)	英数字および 1 バイトカナ文字以外のデータは、ピリオドに置き換えて表示されます。
JEF(EBCDIC/ASCII)	英数字および 1 バイト英小文字以外のデータは、ピリオドに置き換えて表示されます。

NCOBU00972

[整列]

レコード中の任意に指定したデータ項目をキーにして、ファイル内のレコードを昇順または降順に整列し、可変長レコード形式のレコード順ファイルを新しく作成(創成)します。

[入力ファイル]フィールド

入力ファイル名を指定します。 整列するファイルのパス名を指定してください。指定するファイルは、レコード順ファイル、行順ファイル、相対ファイルまたは索引ファイルでなければなりません。

[編成]

入力ファイルのファイル編成を選択します。 ファイル編成の[順]/[行順]/[相対]/[索引]のいずれかをクリックしてください。

[レコード形式]

入力ファイルのレコード形式を選択します。 レコード形式の[固定長]または[可変長]のどちらかをクリックしてください。なお、索引ファイルの場合は指定できません。

[最大レコード長]フィールド

入力ファイルの最大レコード長を指定します。 固定長レコード形式の場合、レコード長を指定します。指定したレコード長が実際のファイルと異なると、正しく動作しません。なお、索引ファイルの場合は指定できません。

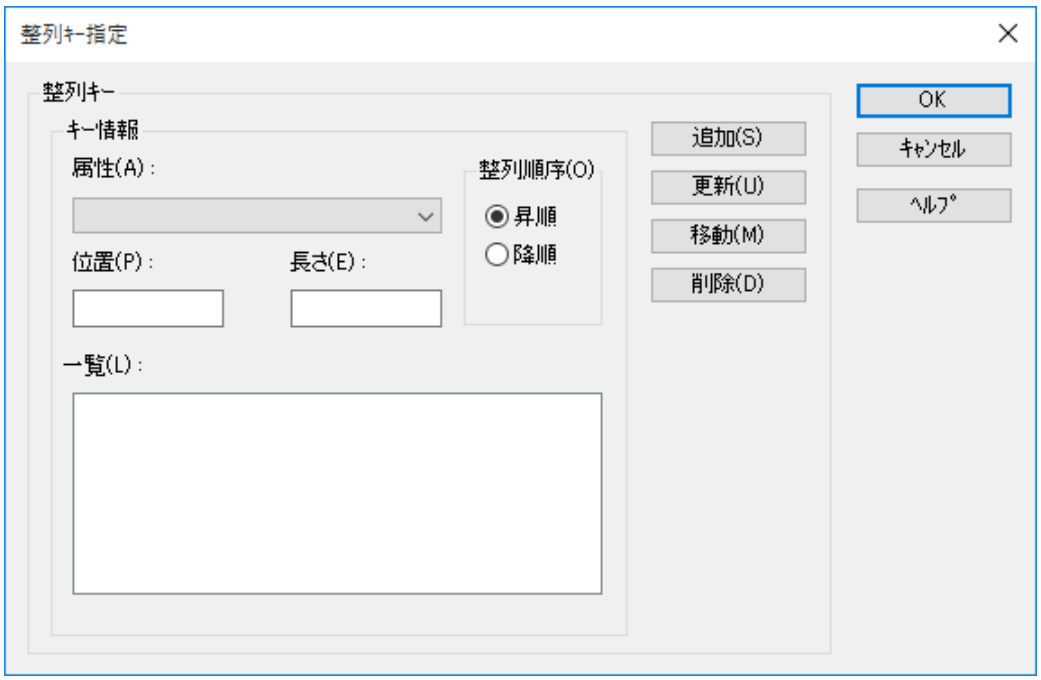
[整列キー]ボタン

整列キーを指定します。 [整列キー]ボタンをクリックし、[整列キー指定]ダイアログで整列キーを指定します。

[出力ファイル]フィールド

出力ファイル名を指定します。 新しく作成(創成)する可変長レコード形式のレコード順ファイルのパス名を指定してください。

[整列キー指定]ダイアログ



[キー情報]

整列キーの情報として、以下の項目を指定します。

- [属性]リストボックス

  整列キーの属性を指定します。 整列キーとして指定する属性を[属性]リストボックスから選択してください。属性は、COBOL のデータ属性として表現されています。

- [位置]フィールド

  整列キーのレコード中の位置を指定します。

- [長さ]フィールド

  整列キーの長さを指定します。 属性に「PIC 1() BIT」を指定した場合、整列キーの長さは、無条件に 1 バイトになります。整列キーの長さは指定できません。1 バイトのマスク値を 10 進数で指定してください。

  [補足] データ属性に「PIC 1() BIT」を指定した場合、整列キーの値は、指定されたレコード中の位置から1バイトのデータとマスク値に指定された値との論理積になります。例えば、データ属性に「PIC 1() BIT」とレコード中の位置に「1」とマスク値に「227」を指定した場合、レコードの内容が「05 AD」(16 進表現)であれば、整列キーの値はレコード中の 1 バイト「AD」(16 進表現)とマスク値「E3」(16 進表現)との論理積「A1」(16 進表現)となります。

- [整列順序]

  整列順序を指定します。 [昇順]または[降順]のどちらかをクリックしてください。

[一覧]リストボックス

指定された整列キーの一覧が表示されます。

[追加]ボタン

整列キーの設定を行います。 [キー情報]の各項目を入力後、[追加]ボタンをクリックします。設定したキー情報は[一覧]リストボックスに表示されます。

[更新]ボタン

整列キーの更新を行います。[一覧]リストボックス内の対象とするキー情報をクリックして、[キー情報]の各項目に表示させます。[キー情報]の各項目の値を更新後、[更新]ボタンをクリックします。

[移動]ボタン

整列キーの移動を行います。[一覧]リストボックス内の対象とするキー情報をクリックして、[キー情報]の各項目に表示させます。[移動]ボタンをクリックし、[一覧]リストボックス内の先頭 0 から始まる相対値で移動先を指定します。

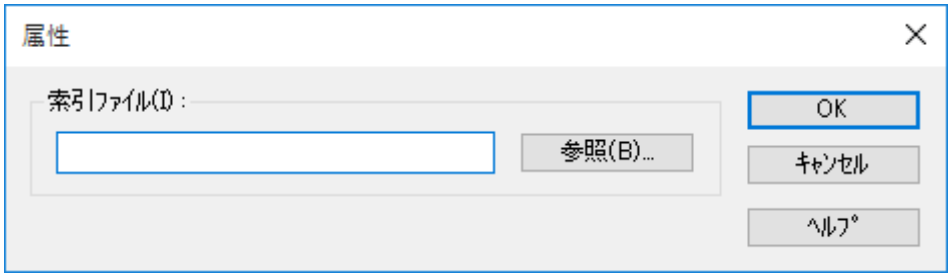
[削除]ボタン

整列キーの削除を行います。[一覧]リストボックス内の対象とするキー情報をクリックして、[キー情報]の各項目に表示させます。[削除]ボタンをクリックします。

- 指定できる整列キーの最大個数は 64 です。
- 整列キーに指定した属性が以下の場合、JEF(EBCDIC/KANA)または JEF(EBCDIC/ASCII)のコード体系として整列します。
  - PIC X()
  - PIC 9()
  - PIC 9() LEADING
  - PIC 9() TRAILING
  - PIC 9() LEADING SEPARATE
  - PIC 9() TRAILING SEPARATE

[属性]

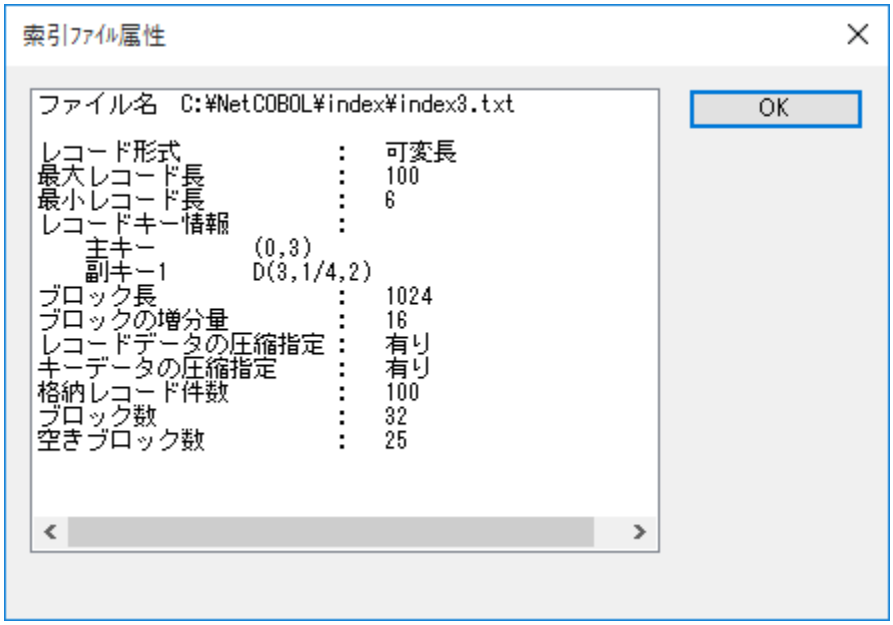
索引ファイルの属性情報を表示します。



[索引ファイル]フィールド

索引ファイル名を指定します。 属性情報を表示させたい索引ファイルのパス名を指定してください。

画面の説明



ファイル名

属性を表示している索引ファイル名を示しています。

レコード形式

レコード形式を示しています。(固定長/可変長)

最大レコード長

固定長の場合、レコード長を示しています。可変長の場合、最大レコード長を示しています。

最小レコード長

固定長の場合、表示されません。可変長の場合、最小レコード長を示しています。

レコードキー情報

索引ファイルのキー情報を示しています。 キー情報の形式は、[ロード]で索引ファイルを創成する際に指定するキー情報の形式と同じです。詳細については、[ロード]の[索引キー指定]ダイアログを参照してください。

ブロック長

索引ファイルの 1 ブロックの大きさを示しています。

ブロックの増分量

索引ファイルの増分単位をブロックの数で示しています。

レコードデータの圧縮指定

格納されているレコードデータが圧縮されているかどうかを示しています。

キーデータの圧縮指定

格納されているキーデータが圧縮されているかどうかを示しています。

格納レコード件数

索引ファイル中に格納されているレコードの件数を示しています。

ブロック数

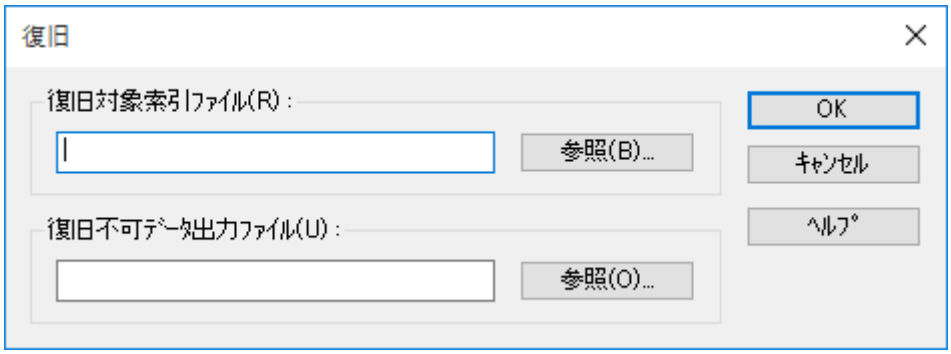
索引ファイルを構成するブロックの数を示しています。

空きブロック数

索引ファイルを構成するブロックのうち、未使用ブロックの数を示しています。

[復旧]

アクセスできなくなった索引ファイルを復旧します。また、復旧できないデータは「復旧不可データ出力ファイル」にバイナリ形式で出力されます。



[復旧対象索引ファイル]フィールド

復旧する索引ファイルのパス名を指定してください。

[復旧不可データ出力ファイル]フィールド

復旧できないデータを格納するためのファイルのパス名を指定してください。なお、復旧できないデータが存在しない場合、復旧不可データ出力ファイルは作成されません。

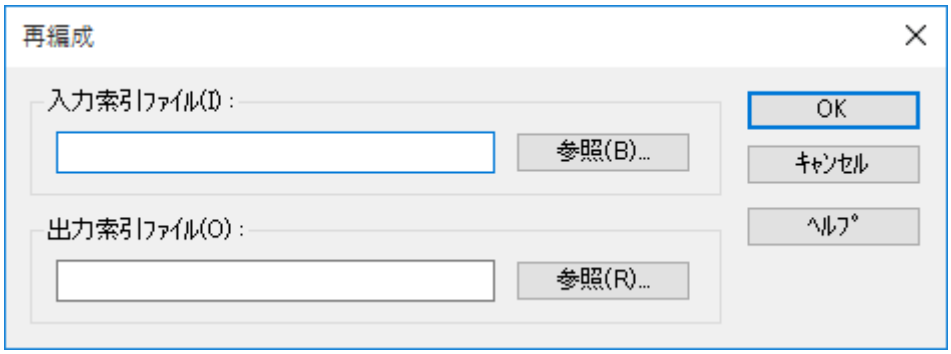
注意

- [復旧]コマンドは復旧前のファイルを書き換えます。復旧前のファイルを残す場合、コマンドの実行前にファイルを複写しておいてください。
- データの状態によっては、復旧不可データ出力ファイルに出力されない場合があります。

NCOBU00975

[再編成]

索引ファイルの空きブロックをできる限り削除することでファイルの大きさを最小にします。なお、空きブロックの削除により、ファイルアクセスの性能が低下する場合があります。



[入力索引ファイル]フィールド

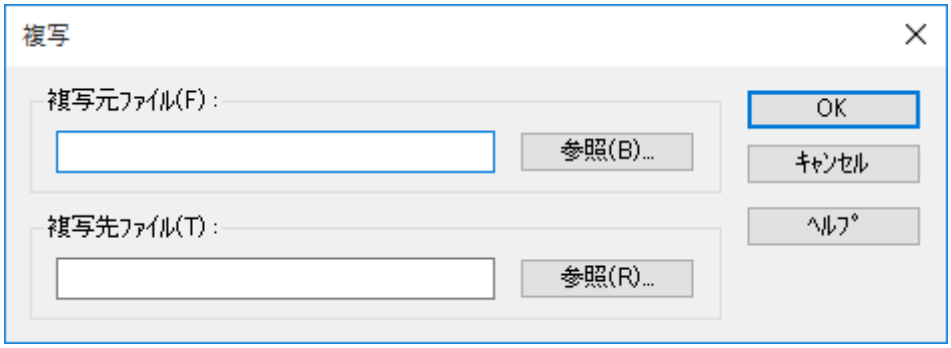
入力索引ファイル名を指定します。 再編成を行う索引ファイルのパス名を指定してください。

[出力索引ファイル]フィールド

出力索引ファイル名を指定します。 再編成後の索引ファイル(新しく作成される)のパス名を指定してください。

[複写]

ファイルを複写します。



[複写元ファイル]フィールド

複写元ファイル名を指定します。 複写元ファイルのパス名を指定してください。ワイルドカード「*」を指定することもできます。

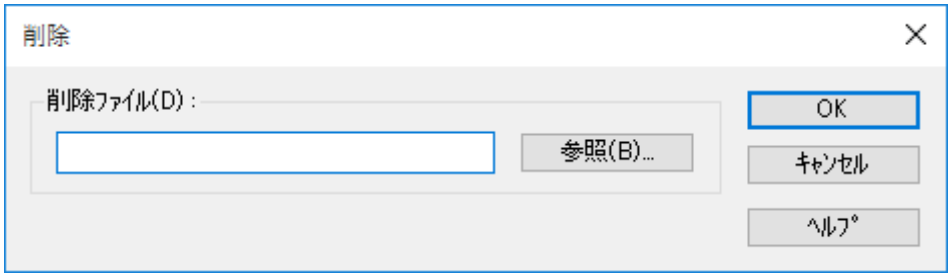
[複写先ファイル]フィールド

複写先ファイル名を指定します。複写先ファイルのパス名を指定してください。

NCOBU00977

[削除]

ファイルを削除します。

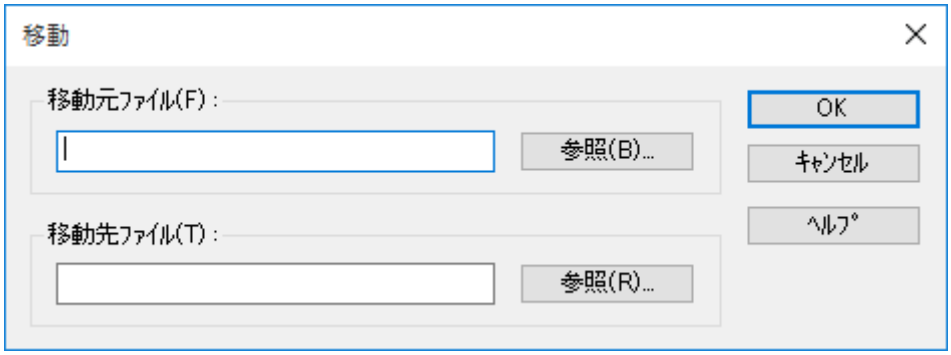


[削除ファイル]フィールド

削除ファイル名を指定します。 削除するファイルのパス名を指定してください。ワイルドカード「*」を指定することもできます。

[移動]

ファイルを移動します。



[移動元ファイル]フィールド

移動元ファイル名を指定します。 移動元ファイルのパス名を指定してください。ワイルドカード「*」を指定することもできます。

[移動先ファイル]フィールド

移動先ファイル名を指定します。 移動先ファイルのパス名を指定してください。

NCOBU00142

COBOL ファイルユーティリティコマンド

COBOL ファイルユーティリティコマンドモードを使用して、コマンドラインから COBOL ファイルユーティリティを使用することができます。

このコマンドは、NetCOBOL for .NET ランタイムシステムのインストールフォルダの"Utilities"サブフォルダにインストールされます。NetCOBOL for .NET ランタイムシステムがインストールされる場所については、「運用パッケージに含まれるツール」を参照してください。

既定の設定では、COBOL ファイルユーティリティコマンドは以下の場所にインストールされます。 （Windows が C:ドライブにインストールされているとしています）

COBOL ファイルユーティリティコマンドの位置
C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥

戻り値

COBOL ファイルユーティリティコマンドの実行が成功した場合は 0 を返します。 失敗した場合は-1 を返します。 コマンドの実行が失敗した場合は、出力されるメッセージに従って対処してください。

このセクションの内容

ファイル復旧コマンド(cobfrcov)

索引ファイルを復旧する操作を説明します。

ファイル再編成コマンド(cobfreog)

索引ファイルを再編成する操作を説明します。

NCOBU00143

ファイル変換コマンド(cobfconv)

テキストファイルから可変長形式のレコード順ファイルの作成、または、可変長形式のレコード順ファイルからテキストファイルの作成を行います。

テキストファイルでは、バイナリデータは 16 進表記の文字列として表現されます。

指定方法

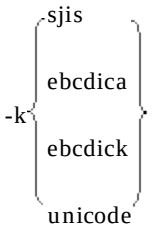
```
cobfconv [-k 文字コード] -o 出力ファイル名 -c 変換方法,データ形式 入力ファイル名
```

インタフェース

インタフェースについて説明します。

文字コード

ファイルのコード体系を以下の形式で指定します。



sjis	ASCII/シフト JIS
ebcdica	JEF (EBCDIC/ASCII)
ebcdick	JEF (EBCDIC/KANA)
unicode	Unicode

- -k オプションを省略した場合または-ksjis を指定した場合、テキストファイルおよびレコード順ファイルの文字コードは ASCII/シフト JIS コード体系として扱います。
- -kebcdica または-kebcdick を指定した場合、テキストファイルの文字コードは ASCII/シフト JIS コード体系として扱います。レコード順ファイルの文字コードは JEF(EBCDIC/KANA)コード体系または JEF(EBCDIC/ASCII)コード体系として扱います。
- -kunicode を指定した場合、テキストファイルの文字コードは UCS-2 コード体系として扱います。レコード順ファイルの内容はレコードデータ項目の定義に従って、UCS-2 コード体系、UTF-8 コード体系または 16 進表記のバイナリ値として扱います。

出力ファイル名

作成するテキストファイルまたはレコード順ファイルのパス名を指定します。既に存在し

ているファイルを指定した場合、エラーとなります。

変換方法,データ形式

変換方法およびレコード順ファイルのレコード構成(データ形式)を以下の形式で指定します。

-k オプションを省略した場合または-ksjis/-kebcdica/-kebcdick を指定した場合

-kunicode を指定した場合

変換方法

変換方法を以下の文字で指定します。

b	テキストファイルからレコード順ファイルに変換します。
t	レコード順ファイルからテキストファイルに変換します。

データ形式

レコード順ファイルの 1 レコードを構成するデータ項目を、データ形式で指定します。データ形式の定義を以下に示します。

- ASCII/シフト JIS コード体系の場合、「用途が表示用のデータ項目(注 1)」を文字形式、そのほかのデータ項目を 16 進形式といいます。
- JEF コード体系の場合、日本語項目および日本語編集項目を日本語形式、日本語項目および日本語編集項目を除いた「用途が表示用のデータ項目」を文字形式、そのほかのデータ項目を 16 進形式といいます。
- Unicode コード体系の場合、日本語項目および日本語編集項目を UCS-2 形式、日本語項目および日本語編集項目を除いた「用途が表示用のデータ項目」を UTF-8 形式、そのほかのデータ項目を 16 進形式といいます。

注 1: 「用途が表示用のデータ項目」については、COBOL 文法書の「5.4.17 USAGE 句」を参照してください。

データ形式を以下の文字で指定します。

allchar	レコード中の全データを文字形式とみなします。
alltxtbin	レコード中の全データを 16 進形式とみなします。

"{c 長さ ｜t 長さ ｜k 長さ} [;…]": レコード中にデータ形式が混在する場合、キーワード文字に続けて、そのデータ形式の長さ(日本語形式の場合は文字数)を指定します。それぞれのキーワード文字の意味を以下に示します。

c	文字形式
t	16 進形式
k	日本語形式

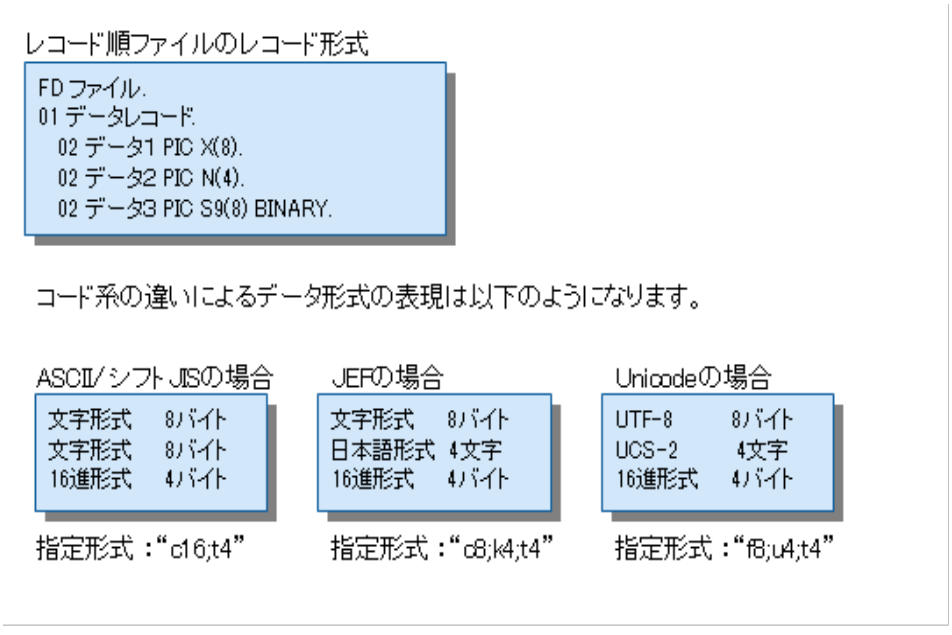
ただし、日本語形式は、JEF コード体系の場合しか指定できません。

allucs2	レコード中の全データを UCS-2 形式とみなします。
allutf8	レコード中の全データを UTF-8 形式とみなします。

"{u 長さ ｜f 長さ ｜t 長さ} [;…]": レコード中にデータ形式が混在する場合、キーワード文字に続けて、そのデータ形式の長さ(UCS-2 形式の場合は文字数)を指定します。それぞれのキーワード文字の意味を以下に示します。

u	UCS-2 形式
f	UTF-8 形式
t	16 進形式

例: データ形式が混在する場合の指定



注意

レコード中にデータ形式が混在する場合、指定できるデータ形式の最大個数は 256 です。

入力ファイル名

変換元のテキストファイルまたはレコード順ファイルのパス名を指定します。

指定例

例１:レコード順ファイルからテキストファイルを作成する

文字コード	ASCII/シフト JIS
入力ファイル名	infile
出力ファイル名	outfile.txt
変換方法	レコード順ファイル → テキストファイル
データ形式	全て文字形式

```
cobfconv -ooutfile.txt -ct,allchar infile
```

例２:テキストファイルからレコード順ファイルを作成する(JEF コード体系)

文字コード	JEF(EBCDIC/ASCII)
入力ファイル名	infile.txt
出力ファイル名	outfile
変換方法	テキストファイル → レコード順ファイル
データ形式	混在(文字形式 3 バイト、16 進形式 2 バイト、文字形式 3 バイト)

```
cobfconv -kebcdica -ooutfile -cb,"c3;t2;c3" infile.txt
```

例３:テキストファイルからレコード順ファイルを作成する(Unicode コード体系)

文字コード	Unicode
入力ファイル名	infile.txt
出力ファイル名	outfile
変換方法	テキストファイル → レコード順ファイル
データ形式	全て UCS-2 形式

```
cobfconv -kunicode -ooutfile -cb,"allucs2" infile.txt
```

NCOBU00144

ファイル属性コマンド(cobfattr)

索引ファイルの属性情報(レコード長、レコード形式、キー情報など)を表示します。

索引ファイル以外の属性情報は表示できません。

指定方法

```
cobfattr 入力ファイル名
```

インタフェース

インタフェースについて説明します。

入力ファイル名

属性を表示する索引ファイルのパス名を指定します。

指定例

例: 索引ファイルの属性を表示する

入力ファイル名	ixdfile

```
cobfattr ixdfile
```

ファイルロードコマンド(cobfload)

可変長形式のレコード順ファイルからレコード順ファイル/相対ファイル/索引ファイルを作成します。また、可変長形式のレコード順ファイルのレコードデータを、すでに存在するレコード順ファイル/相対ファイル/索引ファイルに追加することもできます。これをファイルの拡張といいます。

ファイルの拡張でエラーが発生した場合、出力ファイルはコマンドの実行前の状態に戻されます。

指定方法

```
cobfload [-k 文字コード] -o 出力ファイル名 [-e] -d ファイル属性 入力ファイル名
```

インタフェース

インタフェースについて説明します。

文字コード

```
-kunicode
```

Unicode 形式の索引ファイルを作成する場合、上記の形式で指定します。

出力ファイル名

作成または拡張するファイルのパス名を指定します。

作成時に、既に存在しているファイルを指定した場合、エラーとなります。また、拡張時に、指定したファイルが存在していない場合、エラーとなります。

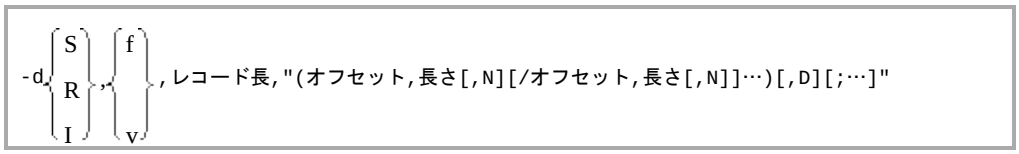
拡張指定

```
-e
```

ファイルの拡張を行う場合、-e オプションを指定します。

ファイル属性

作成または拡張するファイルの属性を以下の形式で指定します。



ファイル編成

ファイル編成を以下の文字で指定します。

S	レコード順ファイル
R	相対ファイル
I	索引ファイル

索引ファイルの拡張を行う場合、レコード形式、レコード長およびレコードキー情報を指定することはできません。

レコード形式

レコード形式を以下の文字で指定します。

f	固定長
v	可変長

レコード長

レコード長を指定します。可変長の場合は、最大レコード長で指定します。

レコードキー情報

索引ファイルを作成する場合、以下のレコードキー情報を指定します。

オフセット	レコードキーとするデータ項目のレコード内位置を、レコードの先頭を 0 とした相対バイト数で指定します。
長さ	レコードキーとするデータ項目の長さをバイト数で指定します。
N	レコードキーとするデータ項目が UCS-2 形式である場合に指定します。
/	1 つのレコードキーとして、連続しない複数のデータ項目を指定する場合、それぞれのデータ項目のオフセットと長さを”/”で区切って指定します。
D	レコードキーの重複を許す場合に指定します。
;	副レコードキーを定義する場合、それぞれの副レコードキー情報を”;”で区切って指定します。

入力ファイル名

変換元のレコード順ファイルのパス名を指定します。

指定例

例 1: レコード順ファイルから索引ファイルを作成する

入力ファイル名	infile	
出力ファイル名	ixdfile	
ファイル属性	ファイル編成	索引ファイル
	レコード形式	可変長
	レコード長	80

	レコードキー情報	データ項目１ (レコード内位置：0／長さ：5)
		データ項目２ (レコード内位置：10／長さ：5)
		重複指定：あり
		副レコードキー(レコード内位置：5／長さ：5)

```
cobfload -oixdfile -dI,v,80,"(0,5/10,5),D;(5,5)" infile
```

例 2: レコード順ファイルの内容を相対ファイルに追加する(拡張)

入力ファイル名	infile	
出力ファイル名	relfile	
ファイル属性	ファイル編成	相対ファイル
	レコード形式	固定長
	レコード長	80

```
cobfload -orelfile -e -dR,f,80 infile
```

入力ファイルに出力ファイルの最大レコード長より大きいレコードが存在した場合、コマンド実行時にエラーとなります。

NCOBU00146

ファイルアンロードコマンド(cobfulod)

レコード順ファイル/相対ファイル/索引ファイルから可変長形式のレコード順ファイルを作成します。

指定方法

```
cobfulod -o 出力ファイル名 -i ファイル属性 入力ファイル名
```

インタフェース

インタフェースについて説明します。

出力ファイル名

作成するレコード順ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

ファイル属性

変換元のファイルの属性を以下の形式で指定します。

ファイル編成

ファイル編成を以下の文字で指定します。

S	レコード順ファイル
R	相対ファイル
I	索引ファイル

レコード形式

レコード形式を以下の文字で指定します。ただし、ファイル編成が索引ファイルの場合は指定できません。

f	固定長
v	可変長

レコード長

レコード長を指定します。可変長の場合は、最大レコード長で指定します。ただし、ファイル編成が索引ファイルの場合は指定できません。

入力ファイル名

変換元のレコード順ファイル/相対ファイル/索引ファイルのパス名を指定します。

指定例

例: 相対ファイルからレコード順ファイルを作成する

入力ファイル名	relfile	
出力ファイル名	outfile	
ファイル属性	ファイル編成	相対ファイル
	レコード形式	固定長
	レコード長	80

```
cobfulod -ooutfile -iR,f,80 relfile
```

NCOBU00147

ファイル表示コマンド(cobfbrws)

ファイルの内容をレコード単位に文字形式と16進形式で表示します。なお、英数字(0x20～0x7e)以外のデータは、ピリオドに置き換えて表示します。また、表示範囲をレコード単位で指定することができます。表示範囲を指定しなかった場合は、ファイル中の全レコードを表示します。

指定方法

```
cobfbrws [-k 文字コード] -i ファイル属性 [-ps 開始位置]
        [-pe 終了位置] [-po 表示順序]
        [-pk 検索キー番号] 入力ファイル名
```

インタフェース

インタフェースについて説明します。

文字コード

表示するファイルのコード体系を以下の形式で指定します。

```
-k {sjis
    ebcdica
    ebcdick
    unicode}
```

sjis	ASCII/シフト JIS
ebcdica	JEF (EBCDIC/ASCII)
ebcdick	JEF (EBCDIC/KANA)
unicode	Unicode

- -k オプションを省略した場合または-ksjis を指定した場合、ファイルの文字コードを ASCII/シフト JIS コード体系として扱います。
- -kebcdica または-kebcdick を指定した場合、ファイルの文字コードを JEF(EBCDIC/KANA)コード体系または JEF(EBCDIC/ASCII)コード体系として扱います。
- -kunicode を指定した場合、ファイルの文字コードを Unicode のコード体系として扱います。

ファイル属性

表示するファイルの属性を以下の形式で指定します。

ファイル編成

ファイル編成を以下の文字で指定します。

S	レコード順ファイル
L	行順ファイル
R	相対ファイル
I	索引ファイル

レコード形式

レコード形式を以下の文字で指定します。ただし、ファイル編成が索引ファイルの場合は指定できません。

f	固定長
v	可変長

レコード長

レコード長を指定します。可変長の場合は、最大レコード長で指定します。ただし、ファイル編成が索引ファイルの場合は指定できません。

開始位置

表示を開始するレコード位置を以下の形式で指定します。

ファイル編成によって指定する内容が異なります。

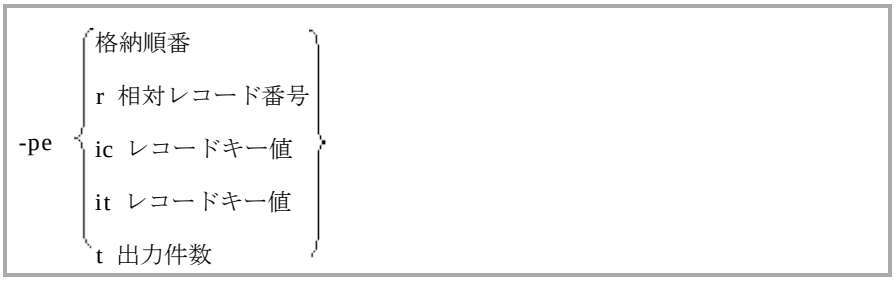
- レコード順ファイルおよび行順ファイルでは、レコードの格納順番を指定します。
- 相対ファイルでは、キーワード文字"r"に続けて相対レコード番号を指定します。
- 索引ファイルでは、レコードキー値を文字形式で指定するか 16 進形式で指定するかによって指定方法が異なります。文字形式の場合は、キーワード文字列"ic"に続けてレコ

ードキー値を指定します。16 進形式の場合は、キーワード文字列"it"に続けてレコードキー値を指定します。

開始位置を省略した場合、ファイルの先頭レコードから表示します。

終了位置

表示を終了するレコード位置を以下の形式で指定します。



出力件数の指定以外は、開始位置と同じです。終了位置を出力件数で指定する場合は、キーワード文字"t"に続けて出力レコード件数を指定します。

終了位置を省略した場合、ファイルの最後のレコードまで表示します。

表示順序

開始位置と終了位置の範囲に複数のレコードが存在する場合、それらのレコードの表示順序を以下の形式で指定します。

A	レコードを昇順に表示します。 レコード順ファイルおよび行順ファイルの場合は、先に格納されたレコードから表示します。 相対ファイルの場合は、相対レコード番号の小さいレコードから表示します。 索引ファイルの場合は、レコードキー値が小さいレコードから表示します。
D	レコードを降順に表示します。 相対ファイルの場合は、相対レコード番号の大きいレコードから表示します。 索引ファイルの場合は、レコードキー値が大きいレコードから表示します。 なお、レコード順ファイルおよび行順ファイルでは降順は指定できません。

表示順序を省略した場合、昇順を指定したとみなします。

検索キー番号

索引ファイルを表示する場合、検索キーになるレコードキーの番号を指定します。レコードキーの番号は、主レコードキーを 0、最初の副レコードキーを 1 として、それ以降の副レコードキーを 2 以上の追番で表現した数字です。

検索キー番号を省略した場合、主レコードキーが指定されたとみなします。

入力ファイル名

表示するファイルのパス名を指定します。

指定例

例 1: レコード順ファイルの内容を表示する

入力ファイル名	seqfile	
ファイル属性	ファイル編成	レコード順ファイル
	レコード形式	固定長
	レコード長	80
開始位置(格納順番)	5	
終了位置(出力件数)	10	

```
cobfbrws -iS,v,80  -ps5  -pet10  seqfile
```

例 2: 相対ファイルの内容を表示する

入力ファイル名	relfile	
文字コード	JEF(EBCDIC/ASCII)	
ファイル属性	ファイル編成	相対ファイル
	レコード形式	固定長
	レコード長	50
開始位置(相対レコード番号)	20	
終了位置(相対レコード番号)	10	
表示順序	降順	

```
cobfbrws -kebcdica -iR,f,50  -psr20  -per10  -poD  relfile
```

例 3: 索引ファイルの内容を表示する

入力ファイル名	relfile	
文字コード	JEF(EBCDIC/KANA)	
ファイル属性	ファイル編成	索引ファイル
開始位置(レコードキー値)	0001(16 進)	
終了位置(レコードキー値)	0010(16 進)	
表示順序	昇順	
検索キー番号	副レコードキー	

```
cobfbrws -kebcdick -iI  -psit0001  -peit0010  -poA  -pk1 ixdfile
```

- 文字コードが JEF(EBCDIC/KANA)の場合、英数字および 1 バイトカナ文字以外のデータは、ピリオドに置き換えて表示します。
- 文字コードが JEF(EBCDIC/ASCII)の場合、英数字および 1 バイト英小文字以外のデータは、ピリオドに置き換えて表示します。
- 文字コードが Unicode の場合、UTF-8 コード体系の 1 バイト表記可能な英数字以外のデータは、ピリオドに置き換えて表示します。
- -psic および-peic に指定するレコードキー値には、ASCII/シフト JIS コード体系で表記可能な値だけを設定できます。

ファイル整列コマンド(cobfsort)

レコード中の任意のデータ項目をキーとして、ファイル中のレコードを昇順または降順に整列し、整列された結果を可変長形式のレコード順ファイルに出力します。

指定方法

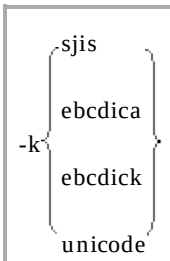
```
cobfsort [-k 文字コード] -o 出力ファイル名 -s 整列条件 -i ファイル属性 入力ファイル名
```

インタフェース

インタフェースについて説明します。

文字コード

文字コードのコード体系を以下の形式で指定します。



sjis	ASCII/シフト JIS
ebcdica	JEF (EBCDIC/ASCII)
ebcdick	JEF (EBCDIC/KANA)
unicode	Unicode

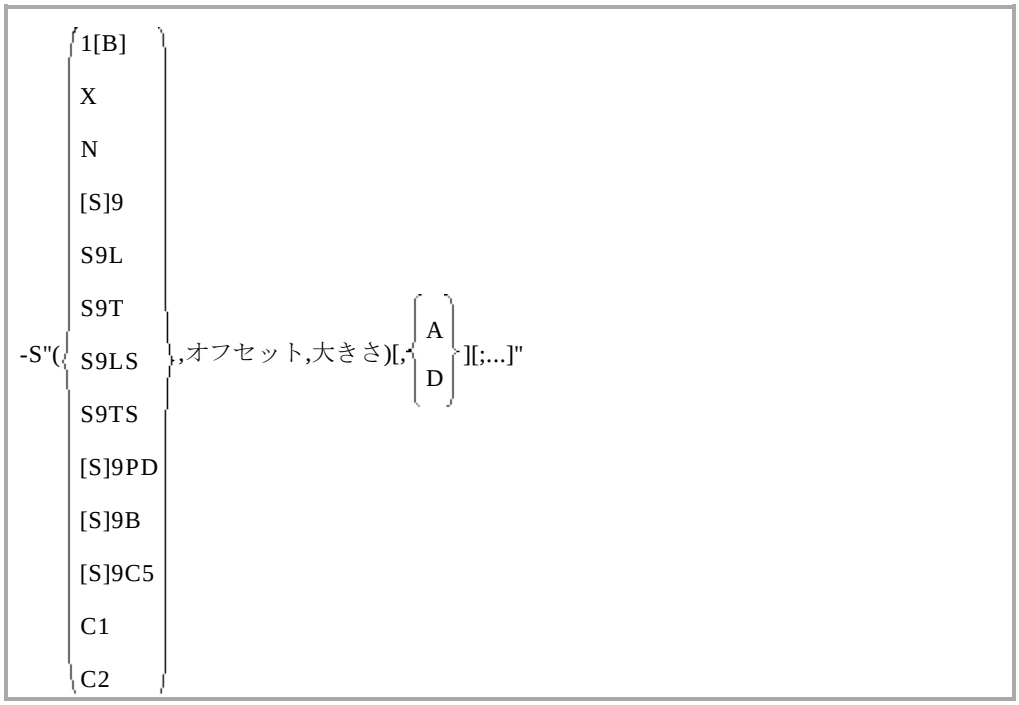
- -k オプションを省略した場合または-ksjis を指定した場合、整列するファイルの文字コードを ASCII/シフト JIS コード体系として扱います。
- -kebcdica または-kebcdick を指定した場合、整列するファイルの文字コードを JEF(EBCDIC/KANA)コード体系または JEF(EBCDIC/ASCII)コード体系として扱います。
- -kunicode を指定した場合、整列するファイルの文字コードを Unicode コード体系として扱います。

出力ファイル名

整列結果を出力するレコード順ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

整列条件

キーとするデータ項目の属性と整列順序を以下の形式で指定します。



キーの項目属性

キーの項目属性を以下の値で指定します。

1[B]	PIC 1() [BIT]
X	PIC X()
N	PIC N()
[S]9	PIC [S]9()
S9L	PIC S9() LEADING
S9T	PIC S9() TRAILING
S9LS	PIC S9() LEADING SEPARATE
S9TS	PIC S9() TRAILING SEPARATE
[S]9PD	PIC [S]9() PACKED-DECIMAL
[S]9B	PIC [S]9() BINARY
[S]9C5	PIC [S]9() COMP-5
C1	COMP-1
C2	COMP-2

オフセット

キー項目のレコード内オフセットを、 レコードの先頭を 0 とした相対バイト位置で指定します。

大きさ

キー項目の大きさを、PICTURE 句での数字項目の桁数または英数字項目/日本語項目の文字数で指定します。

- キーの項目属性に "1[B]" を指定した場合、ここには、整列の対象になるビットのマスク値を 10 進数で指定します。キーの大きさは無条件に 1 バイトになります。
- キーの項目属性に "C1" または "C2" を指定した場合、大きさを省略することができます。ただし、大きさを指定する場合は、項目属性 "C1" に対しては 4、項目属性 "C2" に対しては 8 を指定します。

整列順序

整列順序には、レコードをキーの項目属性に従って昇順に整列するか、降順に整列するかを指定します。

A	昇順
D	降順

整列順序を省略した場合は昇順となります。

指定できる整列条件の最大個数は 64 です。

ファイル属性

整列するファイルの属性を以下の形式で指定します。

ファイル編成

ファイル編成を以下の文字で指定します。

S	レコード順ファイル
L	行順ファイル
R	相対ファイル
I	索引ファイル

レコード形式

レコード形式を以下の文字で指定します。ただし、ファイル編成が索引ファイルの場合は指定できません。

f	固定長
v	可変長

レコード長

レコード長を指定します。可変長の場合は、最大レコード長で指定します。ただし、ファイル編成が索引ファイルの場合は指定できません。

入力ファイル名

整列するファイルのパス名を指定します。

指定例

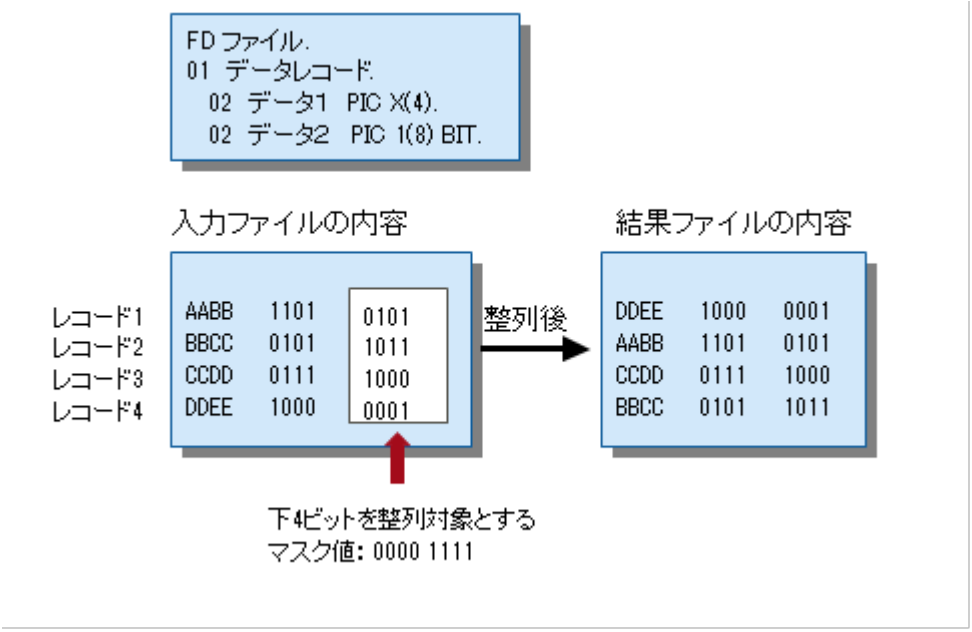
例 1: 相対ファイルを整列してレコード順ファイルに出力する

入力ファイル名	relfile		
出力ファイル名	outfile		
整列条件	[第 1 キー]	項目属性	日本語(N)
		レコード内オフセット	0
		大きさ	2
		整列順序	昇順
	[第 2 キー]	項目属性	英数字
		レコード内オフセット	10
		大きさ	5
		整列順序	降順
ファイル属性	ファイル編成	相対ファイル	
	レコード形式	可変長	
	レコード長	80	

```
cobfsort  -ooutfile  -s"(N,0,2),A;(X,10,5),D"  -iR,v,80  relfile
```

例 2: レコード順ファイルを整列してレコード順ファイルに出力する

レコード順ファイルのレコード構成



入力ファイル名	infile		
出力ファイル名	outfile		
整列条件	[第 1 キー]	項目属性	ブール(1B)
		レコード内オフセット	4
		マスク値(00001111)	15
		整列順序	昇順
ファイル属性	ファイル編成	レコード順ファイル	
	レコード形式	固定長	
	レコード長	5	

```
cobfsort  -ooutfile  -s"(1B,4,15),A"  -iS,f,5  infile
```

NCOBU00149

ファイル復旧コマンド(cobfrcov)

プロセスの異常終了などで、クローズ処理が正常に行われなかった索引ファイルを、再度正常にアクセスできるように復旧します。ただし、データに異常があるために復旧できないレコードについては、それらのデータを未復旧データファイルとして可変長形式のレコード順ファイルに出力します。

指定方法

```
cobfrcov 復旧ファイル名 未復旧データファイル名
```

インタフェース

インタフェースについて説明します。

復旧ファイル名

復旧処理を行う索引ファイルのパス名を指定します。

未復旧データファイル名

復旧できないレコードのデータを出力するファイルのパス名を指定します。復旧できないデータがなければ、未復旧データファイルは作成されません。

指定例

索引ファイルを復旧する。

復旧ファイル名	ixdfile
未復旧データファイル名	seqfile

```
cobfrcov ixdfile seqfile
```

- 同じ索引ファイルに対し、同時にファイル復旧コマンドを実行した場合の動作は保証されません。直前のコマンド実行の完了を確認後、実行してください。
- データの状態によっては、未復旧データファイルに出力されない場合があります。

ファイル再編成コマンド(cobfreog)

索引ファイルの空きブロックを削除し、再編成した内容を別の索引ファイルに出力します。再編成した索引ファイルのファイルサイズは、再編成前のファイルサイズよりも小さくなります。

指定方法

```
cobfreog -o 出力ファイル名 入力ファイル名
```

インタフェース

インタフェースについて説明します。

出力ファイル名

再編成後の索引ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

入力ファイル名

再編成を行う索引ファイルのパス名を指定します。

指定例

索引ファイルを再編成してファイルに出力する。

入力ファイル名	ixdfile
出力ファイル名	outfile

```
cobfreog -ooutfile ixdfile
```

空きブロックの削除によって、ファイルアクセス性能が低下する場合があります。

NCOBU00155

実行環境設定ユーティリティ

ここでは、実行環境設定ユーティリティ(EnvironmentSetup.exe)の使い方について説明します。

はじめに

実行環境設定ユーティリティを使用するためには、 .NET Framework 4 以上が必要となります。

概要

実行環境設定ユーティリティは、アプリケーション構成ファイルの COBOL の実行環境設定情報を編集する GUI ツールです。 実行環境設定ユーティリティで設定および変更できる情報は、COBOL の実行環境情報(エントリ情報設定、SQL 情報設定も含みます)だけです。

アプリケーション構成ファイルに他のアプリケーション設定を行う方法については、.NET Framework のドキュメント の構成ファイルを参照してください。

実行環境設定ユーティリティの起動

このツールはプラットフォームごとに異なる exe ファイルが提供されます。 アプリケーション構成ファイルの対象となるアプリケーションの実行モードに応じて適切なコマンドを実行してください。

アプリケーションのターゲットプラットフォーム	使用する実行環境設定ユーティリティ
x86	x86 プラットフォーム向け
x64	x64 プラットフォーム向け
AnyCPU	実際に実行環境上でアプリケーションが動作するモードに応じて、x86, x64 プラットフォーム向けのいずれか

ツールを起動する手順は以下のとおりです。

x86 プラットフォーム向けの実行環境設定ユーティリティを起動する場合

[スタート] > お使いの NetCOBOL for .NET 製品名 > [実行環境設定ユーティリティ(x86)]を選択します。

x64 プラットフォーム向けの実行環境設定ユーティリティを起動する場合

[スタート] > お使いの NetCOBOL for .NET 製品名 > [実行環境設定ユーティリティ(x64)]を選択します。

- 実際には、上記の実行環境設定ユーティリティがすべてインストールされるわけではありません。 インストール先の Windows のバージョンと、インストールされた運用パッケージのエディションに応じて、 運用環境上でサポートされるプラットフォーム向けの実行環境設定ユーティリティがインストールされます。
- 管理者の権限を必要とする設定を行う場合は、 マウスの右クリックでコンテキストメニューを開き、 [管理者として実行]を選択して起動する必要があります。

また、Visual Studio でプロジェクトを開いている場合は、 プロジェクトから起動することもできます。 詳細については、プロジェクトの実行環境設定を参照してください。

このセクションの内容

画面の説明

実行環境設定ユーティリティの画面について説明します。

[セクションの選択] ダイアログ

[セクションの選択]ダイアログについて説明します。

[プリンタフォントの選択]ダイアログ

[プリンタフォントの選択]ダイアログについて説明します。

[接続文字列ビルダ] ダイアログ (SqlClient データプロバイダ)

SqlClient データプロバイダー用の[接続文字列ビルダ]ダイアログについて説明します。

[接続文字列ビルダ] ダイアログ (OLE DB データプロバイダー)

OLE DB データプロバイダー用の[接続文字列ビルダ]ダイアログについて説明します。

[接続文字列ビルダ] ダイアログ (ODBC データプロバイダ)

ODBC データプロバイダー用の[接続文字列ビルダ]ダイアログについて説明します。

[接続文字列ビルダ] ダイアログ (OracleClient データプロバイダ)

OracleClient データプロバイダー用の[接続文字列ビルダ]ダイアログ について説明します。

[接続文字列ビルダ] ダイアログ (データプロバイダ汎用)

データプロバイダー汎用の[接続文字列ビルダ]について説明します。

[詳細設定] ダイアログ

接続文字列ビルダの[詳細設定]ダイアログについて説明します。

[接続文字列設定] ダイアログ

[接続文字列設定] ダイアログについて説明します。

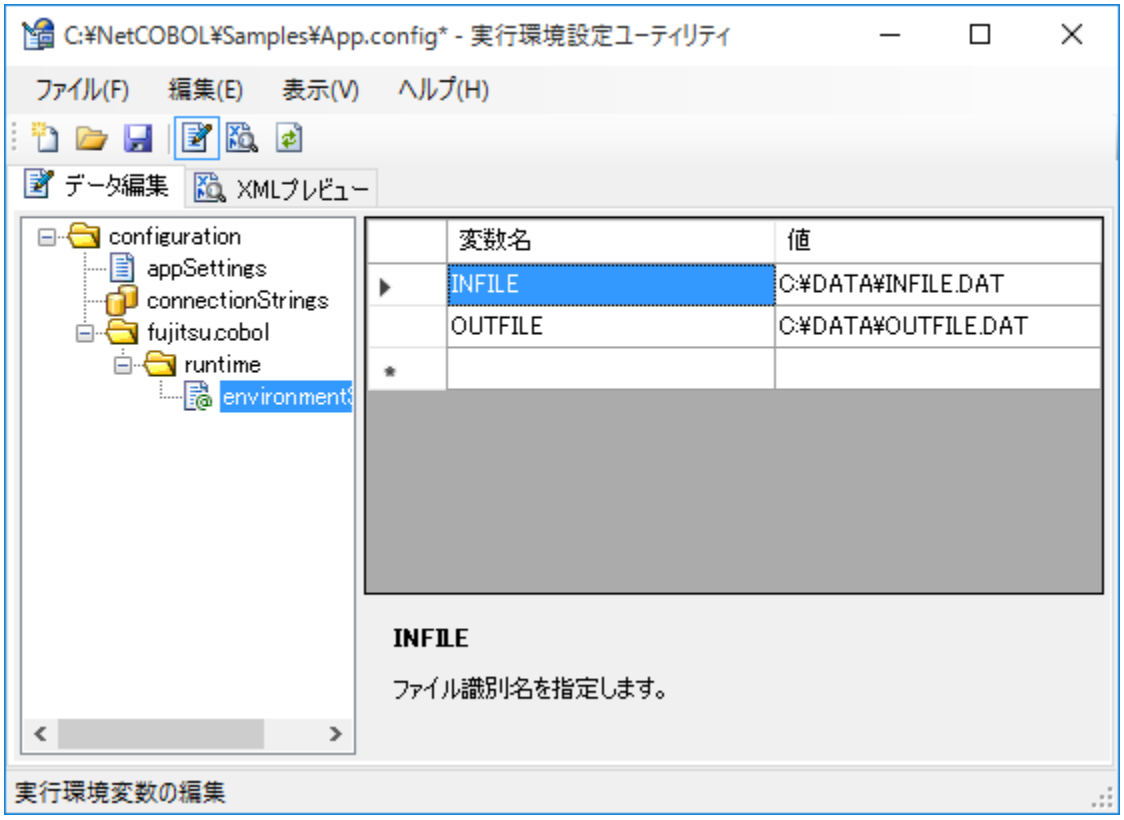
実行環境設定ユーティリティの操作

実行環境設定ユーティリティの操作方法について説明します。

NCOBU00156

画面の説明

ここでは、実行環境設定ユーティリティの画面について説明します。



[画面の説明]

[ファイル]メニュー

新規作成	新規にアプリケーション構成ファイルを作成します。
開く	編集したいアプリケーション構成ファイルを選択します。
インポート▸実行用初期化ファイル	アプリケーション構成ファイルにインポートしたい実行用初期化ファイルを選択します。
インポート▸エントリ情報ファイル	アプリケーション構成ファイルにインポートしたいエントリ情報ファイルを選択します。
インポート▸ODBC 情報ファイル	アプリケーション構成ファイルにインポートしたい ODBC 情報ファイルを選択します。
保存	アプリケーション構成ファイルに編集中の情報を保存します。
名前を付けて保存	アプリケーション構成ファイルを選択したファイル名で編集中の情報を保存します。
終了	実行環境設定ユーティリティを終了します。

[編集]メニュー

設定の追加	各セクションに対応した設定情報を新規に追加します。実行環境変数の編集を行っている場合は、環境変数情報を追加します。
エントリ情報の追加	エントリ情報を追加します。
SQL 設定の追加	SQL のサーバ情報を追加します。
SQL オプション情報の追加	SQL のオプション情報を追加します。
削除	選択された情報を削除します。情報の種類によっては削除できないものもあります。
名前の変更	選択された情報の名前を変更します。情報の種類によっては名前を変更することができないものもあります。

[表示]メニュー

データ編集	データを編集するための画面を表示します。
XML プレビュー	編集中の情報を XML 形式で表示します。この画面で編集することはできません。
最新の情報に更新	編集中の情報を最新のアプリケーション構成ファイルの状態に更新します。

[ヘルプ]メニュー

トピック	ヘルプのトピックを表示します。
インデックス	ヘルプのインデックスを表示します。
検索	ヘルプの検索画面を表示します。
バージョン情報	バージョン情報を表示します。

ツールバー

	[ファイル]-[新規作成]メニューの操作を行います。
	[ファイル]-[開く]メニューの操作を行います。
	[ファイル]-[保存]メニューの操作を行います。
	[表示]-[データ編集]メニューの操作を行います。
	[表示]-[XML プレビュー]メニューの操作を行います。
	[表示]-[最新の情報に更新]メニューの操作を行います。

データ編集タブ

アプリケーション構成ファイルの内容を編集するためのタブです。各セクションを表示するツリービューと設定情報の編集を行うデータビューから構成されます。

- ツリービュー(左側のペイン)

  アプリケーション構成ファイルに設定される各セクションをツリー形式で表示します。

configuration	アプリケーション構成設定のルートセクションを示します。このセクションには編集画面はありません。
appSettings	カスタムアプリケーションの構成情報が格納されるセクション(appSettings 要素)を示します。 選択した場合は、構成情報が右側のデータビューに表示され、構成情報の編集が可能になります。 appSettings 要素については、.NET Framework のドキュメントの appSettings 要素 (全般設定スキーマ)を参照してください。なお、実行環境設定ユーティリティでは、appSettings 要素の file 属性に指定されているファイルを編集することはできません。
connectionStrings	データベース接続文字列のコレクションが格納されるセクション(connectionStrings 要素)を示します。 選択した場合は、接続文字列情報が右側のデータビューに表示され、接続文字列情報の編集が可能になります。 connectionStrings 要素については、.NET Framework のドキュメントの connectionStrings 要素 (ASP.NET 設定スキーマ)を参照してください。
fujitsu.cobol	NetCOBOL for .NET のための設定が格納されるセクショングループを示します。 このセクションには編集画面はありません。
runtime	NetCOBOL for .NET のための実行環境設定が格納されるセクショングループを示します。 このセクションには編集画面はありません。
environmentSettings	実行環境情報のセクションを示します。選択した場合は、実行環境情報が右側のデータビューに表示され、実行環境変数の編集が可能になります。
entrySettings	エントリ情報のセクションを示します。選択した場合は、実行環境情報が右側のデータビューに表示され、エントリ情報の編集が可能になります。
sqlSettings	データベース機能ための SQL 情報が格納されるセクショングループを示します。 このセクションには編集画面はありません。
serverList	SQL 情報の"serverList"セクションを示します。選択した場合は、SQL 情報のサーバ一覧が右側のデータビューに表示され、サーバ一覧の編集が可能になります。
サーバ情報名	SQL 情報のサーバ情報セクション("server"要素)を示します。"サーバ情報名"は"serverList"セクションのサーバ一覧で表示される名前です。選択した場合は、SQL サーバ情報の詳細情報が右側のデータビューに表示され、サーバ情報の編集が

	可能になります。
sqlOptionInf	SQL 情報の"sqlOptionInf"セクションを示します。選択した場合は、SQL オプション情報の一覧が右側のデータビューに表示され、SQL オプション情報一覧の編集が可能になります。
オプション名	SQL 情報のオプション情報セクション("option"要素)を示します。"オプション名"は"sqlOptionInf"セクションのオプション情報一覧で表示される名前です。選択した場合は、SQL オプション情報の詳細情報が右側のデータビューに表示され、オプション情報の編集が可能になります。
sqlDefaultInf	SQL 情報の"sqlDefaultInf"セクションを示します。選択した場合は、デフォルトコネクション情報の詳細情報が右側のデータビューに表示され、デフォルトコネクション情報の編集が可能になります。
connectionScope	SQL 情報の"connectionScope"セクションを示します。選択した場合は、コネクションの有効範囲に関する設定情報が右側のデータビューに表示され、コネクション有効範囲の編集が可能になります。

- データビュー(右側のペイン)

  選択されたセクションの詳細な情報をデータビューで表示します。このビューから各セクションの編集を行うことができます。

XML プレビュータブ

アプリケーション構成ファイルの内容を XML 形式で表示します。このタブでは編集することはできません。

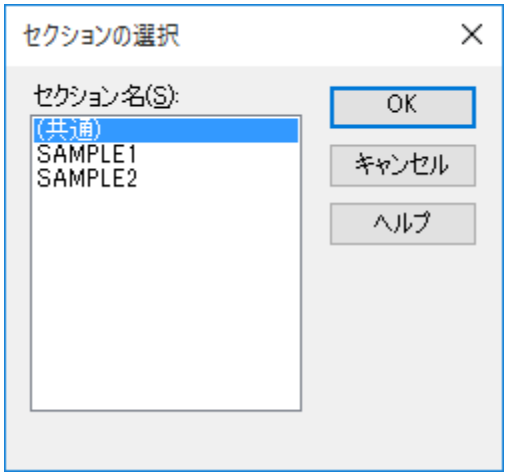
ステータスバー

実行環境設定ユーティリティの状態を表示します。

NCOBU00157

[セクションの選択] ダイアログ

実行用の初期化ファイルを開く(またはインポートする)場合に表示されるダイアログです。



[セクションの選択]ダイアログの各要素について以下に説明します。

[セクション名]リストボックス

実行用の初期化ファイル内に存在するセクション名の一覧が表示されます。"(共通)"を選択した場合は、 セクション名が設定されていない共通セクションが選択されたことを示します。

[OK]ボタン

選択したセクションの情報をアプリケーション構成ファイルに取込みたい場合は、[OK]ボタンをクリックします。

[キャンセル]ボタン

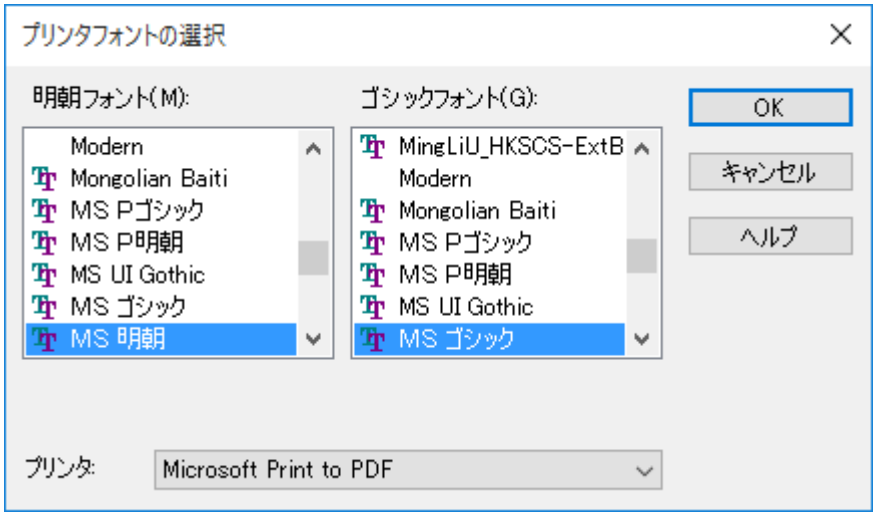
実行用の初期化ファイルのインポートを中止したい場合は、[キャンセル]ボタンをクリックします。

[ヘルプ]ボタン

このドキュメントを表示したい場合は、[ヘルプ]ボタンをクリックします。

[プリンタフォントの選択]ダイアログ

実行環境変数@PrinterFontName の値を編集する場合、...ボタンをクリックすると表示されるダイアログです。



[プリンタフォントの選択]ダイアログの各要素について以下に説明します。

[明朝フォント]リストボックス

フォント名の一覧が表示されます。 明朝フォントとして使用したいフォント名を選択します。

[ゴシックフォント]リストボックス

フォント名の一覧が表示されます。 ゴシックフォントとして使用したいフォント名を選択します。

[OK]ボタン

"明朝フォント"リストボックスと"ゴシックフォント"リストボックスでそれぞれ選択されたフォントをプリンタフォントとして使う場合、[OK]ボタンをクリックします。

[キャンセル]ボタン

プリンタフォントの設定を中止したい場合は、[キャンセル]ボタンをクリックします。

[プリンタ]フィールド

プリンタ名を選択します。

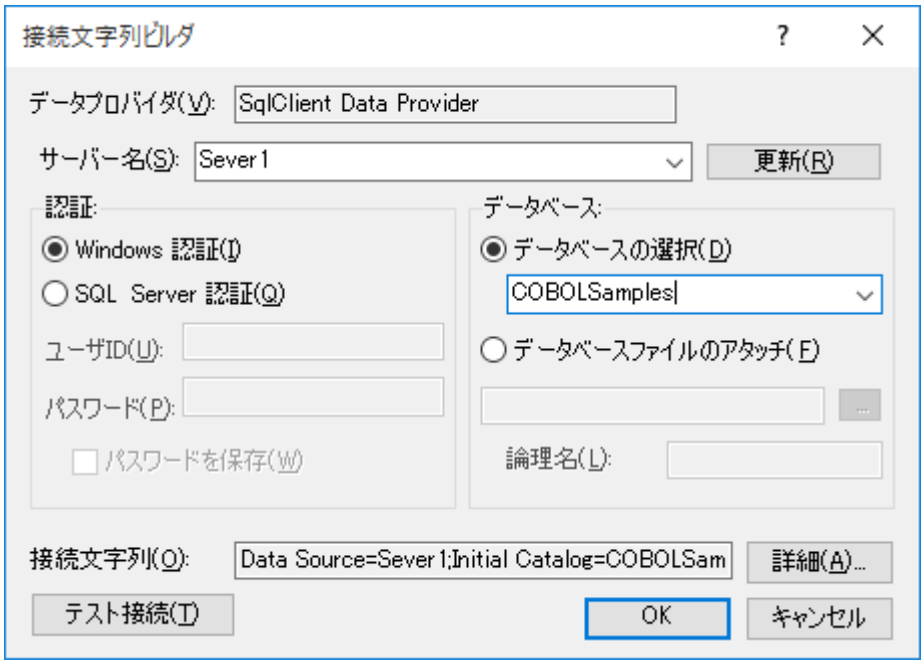
[ヘルプ]ボタン

このドキュメントを表示したい場合は、[ヘルプ]ボタンをクリックします。

NCOBU00817

[接続文字列ビルダ] ダイアログ (SqlClient データプロバイダ)

"connectionStrings"セクションで、接続文字列情報をプロバイダ名として"System.Data.SqlClient"を選択して、接続文字列を編集する場合、...ボタンをクリックすると表示されるダイアログです。このダイアログは、SqlClient データプロバイダーを使用して、SQL Server または Azure SQL データベースに接続するための接続文字列を生成します。



[接続文字列ビルダ] ダイアログ (SqlClient データプロバイダ)の各要素について以下に説明します。

[データプロバイダ]フィールド

データプロバイダー名が表示されます。

[サーバー名]フィールド

接続する SQL Server のホスト名または IP アドレスを指定します。 Azure SQL データベースに接続する場合は、Azure SQL データベースサーバを作成した時に通知された名前を指定します。 指定した値は接続文字列の"Data Source"に対応します。

[Windows 認証]ラジオボタン

SQL Server に Windows 認証でアクセスする場合は、オンにします。 設定した値は接続文字列の"Integrated Security=True"に対応します。

[SQL Server 認証]ラジオボタン

SQL Server に SQL Server 認証でアクセスする場合は、オンにします。 設定した値は接続文字列の"Integrated Security=False"に対応します。

Azure SQL データベースにアクセスする場合は、SQL Server 認証でアクセスします。

[ユーザ ID]フィールド

SQL Server 認証でアクセスする場合のユーザ ID を指定します。 設定した値は接続文字列の"User ID"に対応します。

[パスワード]フィールド

SQL Server 認証でアクセスする場合のパスワードを指定します。 設定した値は接続文字列の"Password"に対応します。

[パスワードを保存]チェックボックス

SQL Server 認証でアクセスする場合にパスワードを接続文字列に含めるときはチェックします。 設定した値は接続文字列の"Persist Security Info"に対応します。

[データベースの選択]ラジオボタン

データベース名を選択する場合はオンにし、データベース名を選択します。 設定した値は接続文字列の"Initial Catalog"に対応します。

[データベースファイルのアタッチ]ラジオボタン

データベースとしてプライマリデータベースファイルを指定する場合はオンにし、アクセスするデータベースファイル名を指定します。 設定した値は接続文字列の"AttachDBFilename"に対応します。

[論理名]フィールド

データベースとしてプライマリデータベースファイルを指定した場合のデータベース名を指定します。 設定した値は接続文字列の"Initial Catalog"に対応します。

[接続文字列]フィールド

生成される接続文字列を表示します。

[詳細]ボタン

接続文字列の詳細設定をする場合、クリックすると[詳細設定] ダイアログが表示されます。

[テスト接続]ボタン

接続文字列が設定された場合に有効になります。クリックすると生成された接続文字列を使用してデータベースに接続します。

[OK]ボタン

接続文字列の編集を完了したい場合は、[OK]ボタンをクリックします。

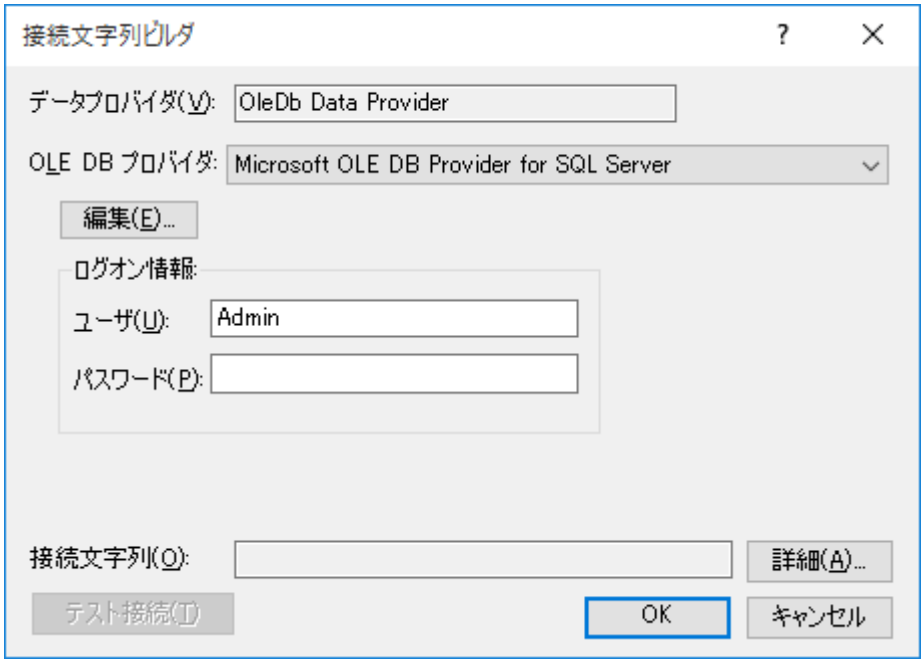
[キャンセル]ボタン

接続文字列の編集を中止したい場合は、[キャンセル]ボタンをクリックします。

NCOBU00818

[接続文字列ビルダ] ダイアログ (OLE DB データプロバイダー)

"connectionStrings"セクションで、接続文字列情報をプロバイダ名として"System.Data.OleDb"を選択して、接続文字列を編集する場合、...ボタンをクリックすると表示されるダイアログです。このダイアログは、OLE DB データプロバイダーを使用して、OLE DB 経由でデータベースに接続するための接続文字列を生成します。



[接続文字列ビルダ] ダイアログ (OLE DB データプロバイダ)の各要素について以下に説明します。

[データプロバイダ]フィールド

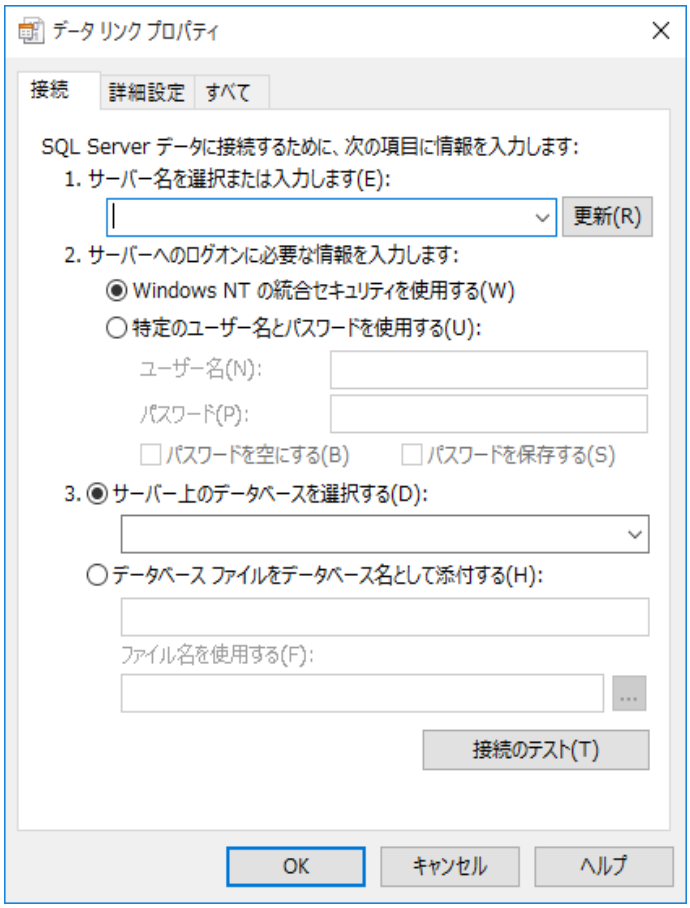
データプロバイダー名が表示されます。

[OLE DB プロバイダ]フィールド

接続するデータベースに対応した OLE DB プロバイダーを選択します。 指定した値は接続文字列の"Provider"に対応します。

[編集]ボタン

選択した OLE DB プロバイダーに対応した接続文字列の編集を行います。 クリックすると以下の[データ リンク プロパティ]ダイアログが表示されます。



実際の接続文字列の設定は[データ リンク プロパティ]ダイアログで行われます。 [データ リンク プロパティ]ダイアログは、選択されている OLE DB プロバイダーによって異なります。

[ユーザ]フィールド

接続時に認証が必要な場合はユーザ ID を指定します。

[パスワード]フィールド

接続時に認証が必要な場合はパスワードを指定します。

[接続文字列]フィールド

生成される接続文字列を表示します。

[詳細]ボタン

接続文字列の詳細設定をする場合、クリックすると[詳細設定] ダイアログが表示されます。

[テスト接続]ボタン

接続文字列が設定された場合に有効になります。クリックすると生成された接続文字列を使用してデータベースに接続します。

[OK]ボタン

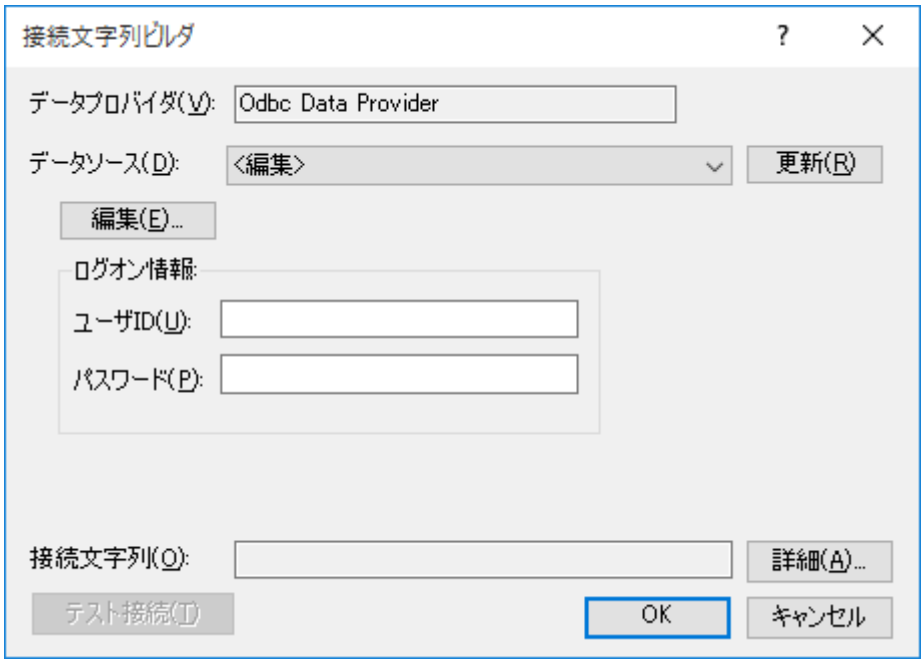
接続文字列の編集を完了したい場合は、[OK]ボタンをクリックします。

[キャンセル]ボタン

接続文字列の編集を中止したい場合は、[キャンセル]ボタンをクリックします。

[接続文字列ビルダ] ダイアログ (ODBC データプロバイダ)

"connectionStrings"セクションで、接続文字列情報をプロバイダ名として"System.Data.Odbc"を選択して、接続文字列を編集する場合、...ボタンをクリックすると表示されるダイアログです。このダイアログは、ODBC データプロバイダーを使用して、ODBC 経由でデータベースに接続するための接続文字列を生成します。



[接続文字列ビルダ] ダイアログ (ODBC データプロバイダー)の各要素について以下に説明します。

[データプロバイダ]フィールド

データプロバイダー名が表示されます。

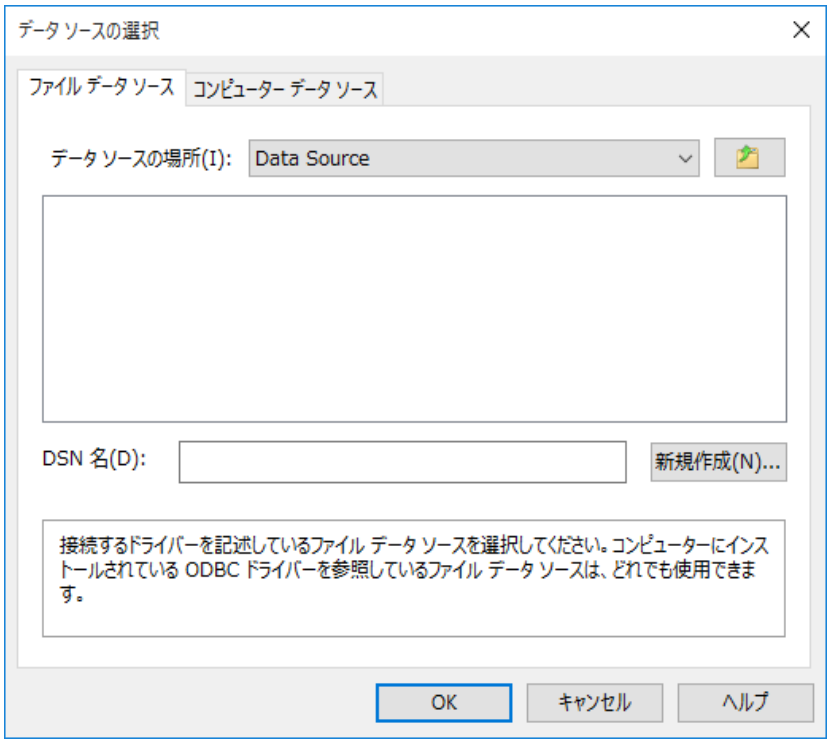
[データソース]フィールド

ODBC データソース名を指定します。 ODBC データソース名が選択された場合は、接続文字列の"DSN"に対応します。

[編集]ボタン

クリックすると[データソース]フィールドに設定されている ODBC データソースに対応したダイアログが表示されます。ここで表示されるダイアログは ODBC ドライバによって異なります。

[データソース]フィールドに"<新規作成>"が選択されている場合は、以下の[データ ソースの選択]ダイアログが表示されます。



[データ ソース]フィールドに"<編集>"が選択されている場合は、接続文字列に対応したダイアログが表示されます。ここで表示されるダイアログは ODBC ドライバによって異なります。接続文字列が設定されていない場合は、"<新規作成>"と同じです。

注意

実行環境設定ユーティリティを管理者として実行している場合のみ、 システムデータソースを作成することができます。

[ユーザ]フィールド

接続時に認証が必要な場合はユーザ ID を指定します。

[パスワード]フィールド

接続時に認証が必要な場合はパスワードを指定します。

[接続文字列]フィールド

生成される接続文字列を表示します。

[詳細]ボタン

接続文字列の詳細設定をする場合、クリックすると[詳細設定] ダイアログが表示されます。

[テスト接続]ボタン

接続文字列が設定された場合に有効になります。クリックすると生成された接続文字列を使用してデータベースに接続します。

[OK]ボタン

接続文字列の編集を完了したい場合は、[OK]ボタンをクリックします。

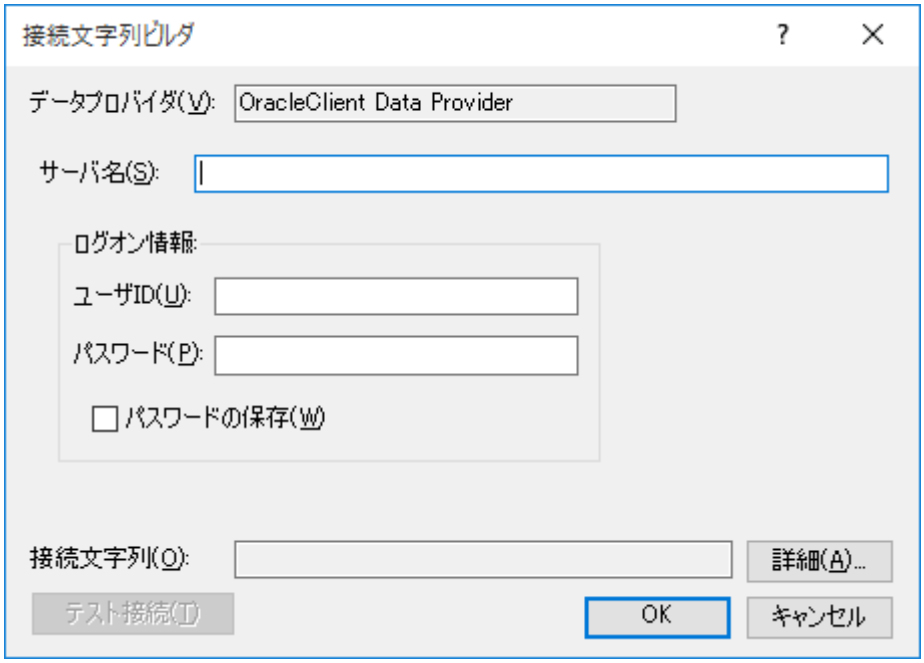
[キャンセル]ボタン

接続文字列の編集を中止したい場合は、[キャンセル]ボタンをクリックします。

NCOBU00820

[接続文字列ビルダ] ダイアログ (OracleClient データプロバイダ)

"connectionStrings"セクションで、接続文字列情報をプロバイダ名として"System.Data.OracleClient"を選択して、接続文字列を編集する場合、...ボタンをクリックすると表示されるダイアログです。このダイアログは、OracleClient データプロバイダーを使用して、Oracle データベースに接続するための接続文字列を生成します。



[接続文字列ビルダ] ダイアログ (OracleClient データプロバイダ)の各要素について以下に説明します。

[データプロバイダ]フィールド

データプロバイダー名が表示されます。

[サーバー名]フィールド

接続する Oracle データベースの SID やサービス名を指定します。 指定した値は接続文字列の"Data Srource"に対応します。

[ユーザ ID]フィールド

Oracle データベースにアクセスする場合のユーザ ID を指定します。 設定した値は接続文字列の"User ID"に対応します。

[パスワード]フィールド

Oracle データベースにアクセスする場合のパスワードを指定します。 設定した値は接続文字列の"Password"に対応します。

[パスワードの保存]チェックボックス

パスワードを接続文字列に含める場合はチェックします。 設定した値は接続文字列の"Persist Security Info"に対応します。

[接続文字列]フィールド

生成される接続文字列を表示します。

[詳細]ボタン

接続文字列の詳細設定をする場合、クリックすると[詳細設定] ダイアログが表示されます。

[テスト接続]ボタン

接続文字列が設定された場合に有効になります。クリックすると生成された接続文字列を使用してデータベースに接続します。

[OK]ボタン

接続文字列の編集を完了したい場合は、[OK]ボタンをクリックします。

[キャンセル]ボタン

接続文字列の編集を中止したい場合は、[キャンセル]ボタンをクリックします。

NCOBU00996

[接続文字列ビルダ] ダイアログ (データプロバイダ汎用)

"connectionStrings"セクションで、接続文字列情報を"System.Data.SqlClient"、"System.Data.OleDb"、"System.Data.Odbc"および"System.Data.OracleClient"以外のプロバイダ名を選択して接続文字列を編集する場合、  ボタンをクリックすると表示されるダイアログです。このダイアログは、指定されたデータプロバイダーを使用して、データベースに接続するための接続文字列を生成します。

[接続文字列ビルダ] ダイアログ (データプロバイダ汎用)の各要素について以下に説明します。

[データプロバイダ]フィールド

データプロバイダー名が表示されます。

プロパティグリッド

接続文字列に設定する詳細な属性の一覧を表示します。 表示される内容は、データプロバイダーによって異なります。

[接続文字列]フィールド

生成される接続文字列を表示します。

[テスト接続]ボタン

接続文字列が設定された場合に有効になります。クリックすると生成された接続文字列を使用してデータベースに接続します。

[OK]ボタン

接続文字列の編集を完了したい場合は、[OK]ボタンをクリックします。

[キャンセル]ボタン

接続文字列の編集を中止したい場合は、[キャンセル]ボタンをクリックします。

[詳細設定] ダイアログ

[接続文字列ビルダ]ダイアログで、[詳細]ボタンをクリックすると表示されるダイアログです。

[詳細設定]ダイアログの各要素について以下に説明します。

プロパティグリッド

接続文字列に設定する詳細な属性の一覧を表示します。 表示される内容は、データプロバイダーによって異なります。

[OK]ボタン

接続文字列の詳細設定を完了したい場合は、[OK]ボタンをクリックします。

[キャンセル]ボタン

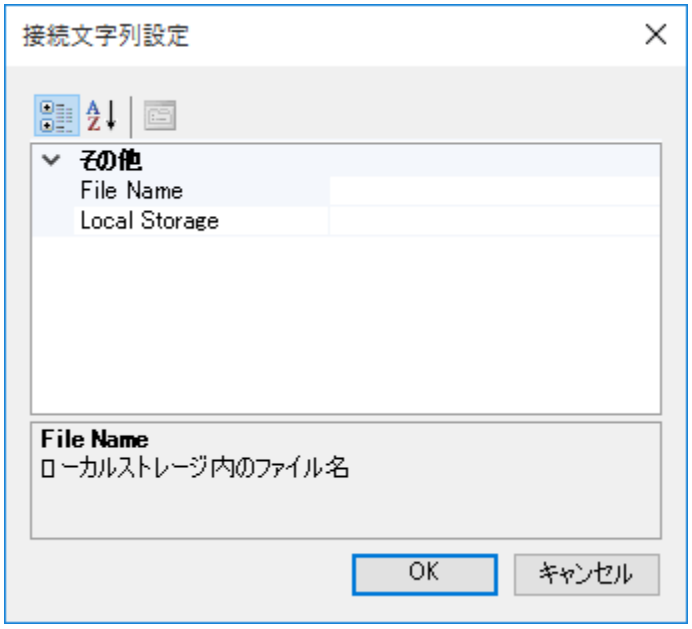
接続文字列の詳細設定を中止したい場合は、[キャンセル]ボタンをクリックします。

NCOBU01124

[接続文字列設定] ダイアログ

[接続文字列設定] ダイアログは、以下の操作で表示されるダイアログです。

- "environmentSettings"セクションで、以下のいずれかの実行環境変数を選択し、値の編集で"<ローカルストレージ 設定>"を選択したとき
  - ファイル識別名
  - SYSIN のアクセス名
  - SYSOUT のアクセス名
  - @MessOutFile



[接続文字列設定]ダイアログの各要素について以下に説明します。

プロパティグリッド

接続文字列に設定する属性の一覧を表示します。 表示される内容は、環境変数によって異なります。

[OK]ボタン

接続文字列設定を完了したい場合は、[OK]ボタンをクリックします。

[キャンセル]ボタン

接続文字列設定を中止したい場合は、[キャンセル]ボタンをクリックします。

実行環境設定ユーティリティの操作

実行環境設定ユーティリティの以下の操作について説明します。

- アプリケーション構成ファイルの操作
  - アプリケーション構成ファイルの新規作成
  - アプリケーション構成ファイルのオープン
  - アプリケーション構成ファイルの保存
- データビューの操作(各セクション共通)
  - 設定情報の追加
  - 設定情報の変更
  - 設定情報の削除
- 環境変数設定の操作
  - 実行用の初期化ファイルの取り込み
- エントリ情報設定の操作
  - エントリ情報の追加
  - エントリ情報の削除
  - エントリ情報ファイルの取り込み
- SQL 情報設定の操作
  - SQL 情報の追加
  - SQL オプション情報の追加
  - SQL 情報のセクション名の変更
  - SQL 情報の削除
  - ODBC 情報ファイルの取り込み

  SQL 情報の設定操作についての詳細は、実行環境設定ユーティリティの使い方を参照してください。

NCOBU00822

アプリケーション構成ファイルの新規作成

アプリケーション構成ファイルを新規に作成する場合、以下の操作を行います。

1. [ファイル]-[新規作成]メニューを選択します。(アプリケーション構成ファイルを指定せずにユーティリティを起動した場合、 起動直後はこの状態になっています。)

アプリケーション構成ファイルのオープン

編集を行うアプリケーション構成ファイルをオープンする場合、以下の操作を行います。

1. [ファイル]メニューから[開く]を選択します。
2. [開く]ダイアログが表示されます。
3. 編集したいファイルを選択し、[開く]ボタンをクリックします。
4. アプリケーション構成ファイルの COBOL 実行環境情報の内容が表示されます。

[開く]ダイアログから、従来の実行用の初期化ファイルを開いてアプリケーション構成ファイルに変換することもできます。この場合は、以下の操作を行います。

1. [ファイル]メニューから[開く]を選択します。
2. [開く]ダイアログが表示されます。
3. アプリケーション構成ファイルに変換したい実行用初期化ファイルを選択し、[開く]ボタンをクリックします。
4. セクションが複数存在する場合は、[セクションの選択] ダイアログが表示されます。この場合は変換したいセクション名を選択し[OK]ボタンをクリックしてください。
5. 実行用の初期化ファイルの内容が展開されます。

変換中に実行環境変数@CBR_ENTRYFILE を検出した場合は、変数値に指定されているエントリ情報ファイルを取り込みます。 同様に、実行環境変数@ODBC_Inf を検出した場合は、変数値に指定されている ODBC 情報ファイルを取り込みます。 いずれも、絶対パスで指定されている場合はそのまま変換を行いアプリケーション構成ファイルに取り込まれます。 相対パスで指定されている場合は、[開く]ダイアログが表示され、取り込みたいファイル名を選択することができます。 正常に変換できた場合は、@CBR_ENTRYFILE または@ODBC_Inf の情報は"environmentSettings"セクションから削除されます。

NCOBU00824

アプリケーション構成ファイルの保存

アプリケーション構成ファイルを保存する場合、以下の操作を行います。

1. [ファイル]-[保存]または[名前をつけて保存]メニューを選択します。
2. 新規作成時または、[名前をつけて保存]メニューを選択した場合は、 [名前を付けて保存]ダイアログが表示されます。この場合は、保存するアプリケーション構成ファイルの名前を指定して、[保存]ボタンをクリックします。
3. 実行環境設定ユーティリティで編集した内容がアプリケーション構成ファイルに保存されます。

設定情報の追加

各セクションに対応した情報を新規に追加する場合、以下の操作を行います。

1. ツリービューから編集を行うセクションを選択します。
2. データビューにフォーカスを移し、[編集]-[設定の追加]メニューまたはコンテキストメニューから[設定の追加]を選択します。
3. 新たに追加されたキー名(変数名や情報名など)のセルを直接編集するか、キー名の候補一覧から追加する情報を選択します。
4. 入力中のセルの編集状態を確定するために Enter キーを押すか、他のセルをクリックします。
5. 追加された行の各セルに必要な情報を設定します。

NCOBU00826

設定情報の変更

データビューに表示されている設定情報のキー(変数名や情報名など)や値を変更する場合、以下の操作を行います。

1. データビュー上で変更したいセルを選択し、選択したセルをクリックするか、Enter キー、英数字キー、F2 キーのいずれかを押します。
2. 選択されたセルが編集可能になるので、値を入力します。 文字列を入力するには以下の方法があります。
   - テキストボックスに直接文字列を入力する
   - ボタンが表示されている場合は、 ボタンをクリックして値の候補リストを表示し、候補リストから値を選択する (選択した文字列によっては、引き続きダイアログが表示される場合があります)
   -  ボタンが表示されている場合は、 ボタンをクリックしてダイアログを表示し、 ダイアログの操作を行う (環境変数名の種類により、表示されるダイアログは異なります)
3. 入力中のセルを確定する場合は、Enter キーを押します。キャンセルする場合は、Esc キーを押します。

設定情報の削除

データビューに表示されている設定情報を削除する場合、以下の操作を行います。

1. データビュー上で削除したい設定情報行または任意のセルを選択します。
2. Del キーを押すか、[編集]-[削除]メニューまたはコンテキストメニューから[設定の削除]を選択します。
3. 選択した環境変数情報がデータビューから削除されます。

NCOBU00828

実行用の初期化ファイルの取り込み

従来の実行用の初期化ファイルの内容をアプリケーション構成ファイルに取り込む場合、以下の操作を行います。

1. [ファイル]メニューから[インポート]-[実行用初期化ファイル]を選択します。
2. [開く]ダイアログが表示されます。
3. 取り込む実行用の初期化ファイルを選択し、[開く]ボタンをクリックします。
4. セクションが複数存在する場合は、[セクションの選択] ダイアログが表示されますので、インポートしたいセクション名を選択し[OK]ボタンをクリックしてください。
5. 実行用の初期化ファイルの内容が取り込まれます。

取り込み中に実行環境変数@CBR_ENTRYFILE を検出した場合は、変数値に指定されているエントリ情報ファイルも同時に取り込まれます。 同様に、実行環境変数@ODBC_Inf を検出した場合は、変数値に指定されているODBC 情報ファイルも取り込まれます。 いずれも、絶対パスで指定されている場合はそのまま変換を行いアプリケーション構成ファイルに取り込まれます。 相対パスで指定されている場合は、[開く]ダイアログが表示され、取り込みたいファイル名を選択することができます。 正常に変換できた場合は、@CBR_ENTRYFILE または@ODBC_Inf の情報は"environmentSettings"セクションから削除されます。

実行用の初期化ファイルの取り込みを行うと、同名の変数が既に存在している場合、後から取り込まれた実行用の初期化ファイルの内容に書き換えられます。

エントリ情報の追加

エントリ情報の追加は以下の手順で行います。

1. "entrySettings"セクションが存在しない場合は、[編集]メニューから[エントリ情報の追加]を選択 または"runtime"セクションのコンテキストメニューから[エントリ情報の追加]を選択し、 "entrySettings" セクションを追加します。 "entrySettings"セクションが存在する場合は、"entrySettings"セクションを選択し、 [編集]メニューから[設定の追加]を選択またはコンテキストメニューから[設定の追加]を選択します。
2. データビューにエントリ情報が表示されます。
3. 新たに追加された副プログラム名を直接変更します。
4. [副プログラム名]セルの編集状態を確定するために Enter キーを押すか、他のセルをクリックします。
5. [DLL またはアセンブリ]セルに情報を設定します。

NCOBU00830

エントリ情報の削除

すべてのエントリ情報を削除する場合は、以下の操作を行います。

1. "entrySettings"セクションをツリービューから選択します。
2. Del キーを押すか、[編集]-[削除]メニューまたはコンテキストメニューから[削除]を選択します。
3. すべての"entrySettings"セクションが削除されます。

エントリ情報ファイルの取り込み

エントリ情報ファイルを取り込む場合、以下の操作を行います。

1. [ファイル]メニューから[インポート]-[エントリ情報ファイル]を選択します。
2. [開く]ダイアログが表示されます。
3. 取り込みたいエントリ情報ファイルを選択し、[開く]ボタンをクリックします。
4. エントリ情報ファイルの内容がアプリケーション構成ファイルに取り込まれます。

ツリービューに"entrySettings"セクションがすでに存在する場合は、エントリ情報ファイルを取り込むことはできません。

NCOBU00832

SQL 情報の追加

ツリービューに"sqlSettings"セクションが存在しない場合は、以下の操作を行います。

1. [編集]-[SQL 設定の追加]メニューまたは"runtime"セクションを選択してコンテキストメニューから[SQL 設定の追加]を選択します。
2. "サーバ情報名"セクション、"sqlDefaultInf"セクションおよび"connectionScope"セクションが追加され、データビューにサーバ情報が表示されます。
3. 必要な各 SQL 情報の編集を行います。

ツリービューに"sqlSettings"セクションが存在する場合は、以下の操作を行います。

1. [編集]-[SQL 設定の追加]メニューまたは"sqlSettings"セクションを選択してコンテキストメニューから[SQL 設定の追加]を選択します。 または、"serverList"セクションを選択し、コンテキストメニューから[設定の追加]を選択します。
2. 新しい"サーバ情報名"セクションが追加され、データビューにサーバ情報が表示されます。
3. 必要なサーバ情報の編集を行います。

SQL オプション情報の追加

"sqlSettings"セクションに SQL オプション情報を追加する場合は、以下の操作を行います。

1. [編集]-[SQL オプション情報の追加]メニューまたは"sqlSettings"セクションを選択してコンテキストメニューから[SQL オプション情報の追加]を選択します。
2. 新しい"オプション名"セクションが追加され、データビューにオプション情報が表示されます。
3. 必要なオプション情報の編集を行います。

NCOBU00834

SQL 情報のセクション名の変更

SQL 情報のサーバ情報名やオプション名を変更する場合、以下の操作を行います。

1. 名前を変更したいサーバ情報名またはオプション名のセクションをツリービューから選択します。
2. 選択したセクション名部分をクリックするか、[編集]-[名前の変更]メニューまたはコンテキストメニューから[名前の変更]を選択します。
3. 編集可能となるので、名前の変更を行います。

ツリービューの"serverList"セクションが選択されている場合は、データビュー上でサーバ情報名の変更を行った場合も、ツリービューのサーバ情報のセクション名が変更されます。 同様に、"sqlOptionInf"セクションが選択されている場合は、データビュー上でオプション名の変更を行うことができます。

SQL 情報の削除

サーバ情報名およびオプション名の各セクションは削除することができます。 削除する場合は、以下の操作を行います。

1. 削除したいセクションをツリービューから選択します。
2. Del キーを押すか、[編集]-[削除]メニューまたはコンテキストメニューから[削除]を選択します。
3. 選択したセクションが削除されます。

ツリービューの"serverList"が選択されている場合、データビュー上でサーバ情報を選択してコンテキスメニューから[設定の削除]メニューを選択した場合も、SQL 情報からサーバ情報が削除されます。 同様に、"sqlOptionInf"が選択されている場合は、データビュー上でオプション情報を削除できます。

すべての SQL 情報を削除する場合は、以下の操作を行います。

1. "sqlSettings"セクションをツリービューから選択します。
2. Del キーを押すか、[編集]-[削除]メニューまたはコンテキストメニューから[削除]を選択します。
3. "sqlSettings"セクションが削除されます。

NCOBU00836

ODBC 情報ファイルの取り込み

ODBC 情報ファイルを取り込む場合、以下の操作を行います。

1. [ファイル]メニューから[インポート]-[ODBC 情報ファイル]を選択します。
2. [開く]ダイアログが表示されます。
3. 取り込みたい ODBC 情報ファイルを選択し、[開く]ボタンをクリックします。
4. ODBC 情報ファイルの内容がアプリケーション構成ファイルに取り込まれます。

ツリービューに"sqlSettings"セクションがすでに存在している場合は、ODBC 情報ファイルを取り込むことはできません。

実行環境設定ツール

このツールは互換のために残されています。NetCOBOL for .NET では、実行環境設定ユーティリティを使用することをお勧めします。

実行環境設定ツールは、実行用の初期化ファイルの内容を編集するためのツールです。

COBOL プログラムの実行時に実行用の初期化ファイルを使用する場合は、プログラムの実行前にあらかじめこのツールで実行用の初期化ファイルの内容を編集してください。このとき、ファイルのパス名やプリンタ名の指定は、動作環境によって異なる場合があるため、注意が必要です。

なお、実行用の初期化ファイルを格納する位置については、実行用の初期化ファイルの検索順序を参照してください。

実行環境設定ツールの起動

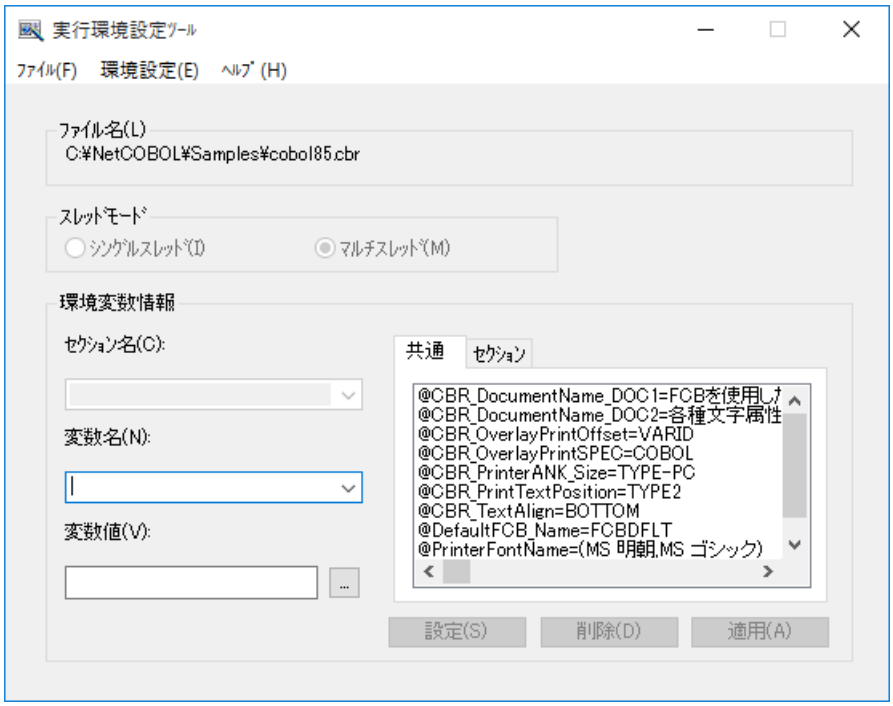
このコマンドは、NetCOBOL for .NET ランタイムシステムのインストールフォルダの"Utilities"サブフォルダにインストールされます。NetCOBOL for .NET ランタイムシステムがインストールされる場所については、「運用パッケージに含まれるツール」を参照してください。

既定の設定では、実行環境設定ツールは以下の場所にインストールされます。 （Windows が C:ドライブにインストールされているとしています）

Windows の種類	実行環境設定ツールの位置
32-bit	C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥F5FHENVU.exe
64-bit	C:¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥F5FHENVU.exe

設定する実行環境情報の詳細については、実行環境変数 - トピック順一覧を、ツールの使い方の詳細についてはツールのヘルプを参照してください。

実行環境設定ツールは、32-bit アプリケーションです。 実行環境設定ツールは x86 プラットフォーム、または x64 プラットフォーム上の WOW64 環境で使用してください。



注意

NetCOBOL for .NET の全てのアプリケーションはマルチスレッドです。マルチスレッドのアプリケーションでは、共通の環境変数しか指定できません。

実行環境設定ツールの画面の説明

メニューバー

ファイル

実行用の初期化ファイルのオープン/クローズとツールの終了を行います。(初期化ファイルを作成するには、オープンの機能を使用します。)

環境設定

プリンタ名の取込み、文字コードの設定および各種フォント指定を行います。

ヘルプ

実行環境設定ツールのヘルプを表示します。

エディットボックス

セクション名

NetCOBOL for .NET では、使用できません。

変数名

環境変数名を入力します。COBOL ランタイムシステムで予約されている環境変数名については、リストに表示されます。

変数値

環境変数の値を入力します。

リストボックス

共通タブ

現在設定されている、実行用の初期化ファイルの共通部の設定内容を表示します。

セクションタブ

NetCOBOL for .NET では、使用できません。

ラジオボタン

NetCOBOL for .NET では、使用できません。

ボタン

設定

選択されているリストボックスに、変数名と変数値に記述された情報を設定するときに押します。 実行用の初期化ファイルに設定内容を反映させるには、さらに[適用]ボタンを押す必要があります。

削除

リストボックスで選択されている変数名と変数値をリストボックスから削除するときに押します。

適用

選択されているリストボックスの内容を実行用の初期化ファイルに反映するときに押します。

実行用の初期化ファイルのオープン

編集を行う実行用の初期化ファイルをオープンする場合、以下の操作を行います。

1. [ファイル]メニューから[開く]を選択します。
2. [実行用の初期化ファイルの指定]ダイアログが表示されるので、既存のファイルを開く場合は、ファイルを選択します。新規に実行用の初期化ファイルを作成する場合は、そのファイル名をエディットボックスに入力します。
3. [実行用の初期化ファイルの指定]ダイアログの[開く]ボタンをクリックすると、リストボックスに実行用の初期化ファイルの内容が表示されます。

環境変数情報の追加

環境変数情報を追加する場合、以下の操作を行います。

1. [変数名]コンボボックスのリストから追加する環境変数情報を選択します。
2. [変数値]に情報を設定します。
3. [設定]ボタンをクリックします。リストボックスに環境変数情報が追加されます。

エントリ情報については、あらかじめエントリ情報ファイルを作成し、環境変数情報@CBR_ENTRYFILEに、エントリ情報ファイル名を指定してください。 なお、指定形式については、@CBR_ENTRYFILE (エントリ情報ファイルの指定)を参照してください。

環境変数情報の変更

リストボックスに表示されている環境変数情報を変更する場合、以下の操作を行います。

1. リストボックスから変更する環境変数情報を選択します。選択した情報は、変数名および変数値のエディットボックスに表示されます。
2. 内容を変更します。
3. [設定]ボタンをクリックします。変更した環境変数情報がリストボックスに表示されます。

環境変数情報の取消し

リストボックスに表示されている環境変数情報を取り消す場合、以下の操作を行います。

1. リストボックスから削除する環境変数情報を選択します。
2. [削除]ボタンをクリックします。選択した環境変数情報がリストボックスから削除されます。

実行用の初期化ファイルのクローズ

現在オープンしている実行用の初期化ファイルをクローズする場合、[ファイル]メニューから[閉じる]を選択します。その時点で未適用の情報は実行用の初期化ファイルに反映されないため、注意してください。

実行用の初期化ファイルへの保存

[適用]ボタンをクリックすると、リストボックス内の情報が実行用の初期化ファイルに保存されます。前回の実行用の初期化ファイルの保存内容は書き換えられるため、注意してください。

実行環境設定ツールの終了

実行環境設定ツールの[ファイル]メニューから[終了]を選択すると、実行環境設定ツールが終了します。その時点で未適用の情報は実行用の初期化ファイルに反映されないため、注意してください。

ODBC 情報設定ユーティリティ

このツールは互換のために残されています。NetCOBOL for .NET では、実行環境設定ユーティリティを使用することをお勧めします。

ODBC 情報設定ユーティリティは、ODBC 接続のデータベースアクセスに必要となる情報を ODBC 情報ファイルに設定するためのユーティリティです。

ODBC 情報ファイルは、実行環境変数@ODBC_Inf に指定したファイルです。実行環境変数@ODBC_Inf の詳細については、@ODBC_Inf (ODBC 情報ファイルの指定)を参照してください。

ODBC 情報設定ユーティリティには、以下の機能があります。

- ODBC 情報ファイルの選択
- サーバ情報の設定
- デフォルトコネクション情報の設定
- コネクション有効範囲の設定

ODBC 情報設定ユーティリティは、3 つのページで構成されています。

このセクションの内容

ODBC 情報設定ユーティリティの起動

ODBC 情報設定ユーティリティの起動方法について説明します。

[サーバ情報] ページ

[サーバ情報]ページについて説明します。

[デフォルトコネクション情報] ページ

[デフォルトコネクション情報]ページについて説明します。

[コネクション有効範囲] ページ

[コネクション有効範囲]ページについて説明します。

NCOBU00162

ODBC 情報設定ユーティリティの起動

このツールは互換のために残されています。 NetCOBOL for .NET では、実行環境設定ユーティリティを使用することをお勧めします。

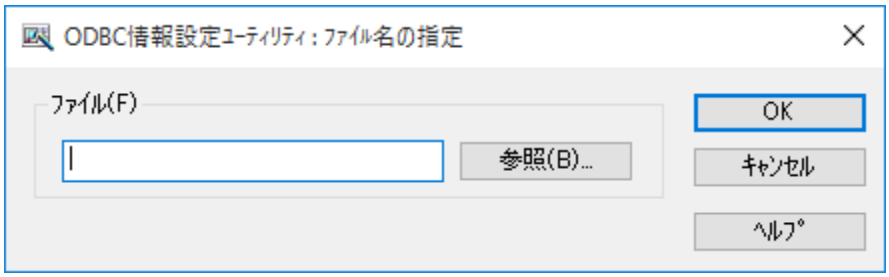
ODBC 情報設定ユーティリティは、NetCOBOL for .NET ランタイムシステムのインストールフォルダの"Utilities"サブフォルダにインストールされます。 NetCOBOL for .NET ランタイムシステムがインストールされる場所については、 「運用パッケージに含まれるツール」を参照してください。

既定の設定では、実行環境設定ツールは以下の場所にインストールされます。 （Windows が C:ドライブにインストールされた場合）

Windows の種類	実行環境設定ツールの位置
32-bit	C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥F5FHSQL.exe
64-bit	C:¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥F5FHSQL.exe

ODBC 情報設定ユーティリティは、32-bit アプリケーションです。 ODBC 情報設定ユーティリティは x86 プラットフォーム、または x64 プラットフォーム上の WOW64 環境で使用してください。

ODBC 情報設定ユーティリティが起動されると、以下の画面が表示されます。

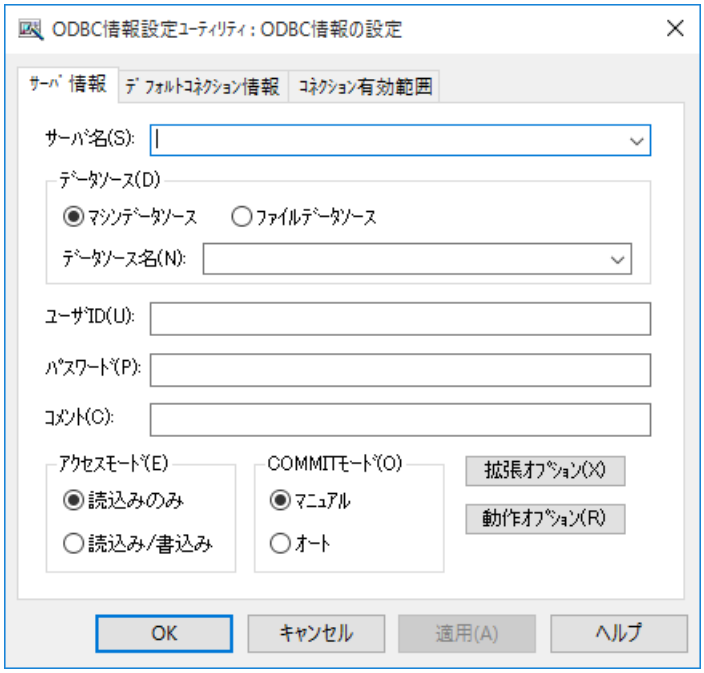


[ファイル]フィールドに、ODBC 情報ファイル名を入力します。入力されたファイル名が存在しない場合は、新規にファイルが作成されます。

[サーバ情報] ページ

このツールは互換のために残されています。NetCOBOL for .NET では、実行環境設定ユーティリティを使用することをお勧めします。

このページでは、CONNECT 文またはデフォルトコネクション情報に指定したサーバ名に対して、必要な情報を設定します。



[サーバ名] フィールド (必須)

コネクションを確立するサーバの名前を指定します。次の箇所で指定したサーバ名を指定してください。

- CONNECT 文に指定したサーバ名(サーバ名指定の CONNECT 文によってコネクションを確立する場合)
- ODBC 情報ファイルのデフォルトコネクション情報に指定したサーバ名(DEFAULT 指定の CONNECT 文によってコネクションを確立する場合)

なお、すでにサーバ名が ODBC 情報ファイルに定義されている場合は、サーバ名の一覧から選択することもできます。

注意

ここに指定するサーバ名は、CONNECT 文とデータソースの対応を一意に行うための名前です。実際のデータベースが存在するマシンの名前ではないことに注意してください。

[データソース] (必須)

[データソース名]フィールドに、アクセスする ODBC データソースの名前を指定します。すでに ODBC データソースが定義されている場合は、ODBC データソースの一覧から選択することもできます。(ファイルデータソースは一覧に表示されません。)

ODBC データソースにはデータプロバイダーへの接続に関する情報が格納されています。ODBC データソースの定義は、コントロールパネルの[管理ツール]-[データソース(ODBC)]で設定できます。

ODBC 情報設定ユーティリティで指定できる ODBC データソースは、マシンデータソース(ユーザデータソースおよびシステムデータソース)およびファイルデータソースです。

[補足]: ファイルデータソースを指定した場合、かつ、ODBC 情報ファイルのサーバ情報にユーザ ID とパスワードを設定している場合は、サーバ情報に設定したユーザ ID とパスワードが使用されます。

64-bit Windows 上でデータソース名を指定する場合

- 64-bit Windows 上では、 WOW64(32-bit 互換)環境のマシンデータソースの一覧のみ表示されます。 64-bit プログラム向けの ODBC データソースを指定する場合は、 [データソース名]フィールドに ODBC データソース名を直接入力してください。
- 64-bit Windows 上で 32-bit プログラム向けのデータソースを定義する場合は、WOW64 システムフォルダ(`%WINDIR%¥SysWOW64¥`)の `odbcad32.exe` コマンドを直接実行し、WOW64 環境の ODBC アドミニストレータを起動してください。

  例:

```
C:¥> C:¥WINDOWS¥SysWOW64¥odbcad32.exe
```

[ユーザ ID]フィールド

データベースの接続に認証が必要な場合、サーバ情報に定義されたデータソースを操作するためのユーザ ID を指定します。

[パスワード]フィールド

データベースの接続に認証が必要な場合、サーバ情報に定義されたデータソースを操作するユーザ ID のパスワードを指定します。

[注意]: パスワードはセキュリティのために入力時にアスタリスク(*)で表示されます。 入力には十分注意してください。

[補足]: パスワードはセキュリティのために暗号化されて ODBC 情報ファイルに格納されます。

[コメント]フィールド

指定したサーバ名に対するコメントを記述します。

[アクセスモード]

データソースに対するアクセスモードを指定します。

読込みのみ	読込み専用でアクセスすることを指定します。
読込み/書込み	読込みおよび書込みの両方ができる状態でアクセスすることを指定します。

[補足]: デフォルトは「読込みのみ」です。アクセスモードが指定されていない場合は、「読込みのみ」として扱います。

[COMMIT モード]

データソースに対する COMMIT モード(トランザクションの動作)を指定します。

マニュアル	COBOL ソースプログラムに COMMIT 文または ROLLBACK 文を記述することで、SQL 文の操作を確定します。
オート	COBOL ソースプログラムの記述に関係なく、SQL 文ごとにその操作が確定(COMMIT)します。

- オートを指定した場合、SQL 文を実行した時点でデータベースへ処理が反映されるため、ROLLBACK 文でデータベースを元の状態に戻すことができません。したがって、マニュアルを指定することをお勧めします。
- データソースによって、指定に対する動作が異なる場合があります。

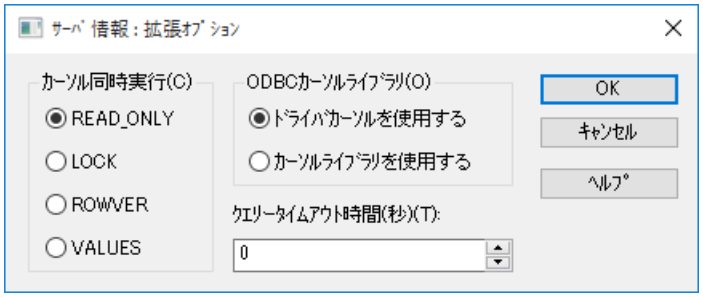
[拡張オプション]ボタン

クリックすると、[拡張オプション]ダイアログが表示されます。詳細については、[拡張オプション]ダイアログを参照してください。

[動作オプション]ボタン

クリックすると、[動作オプション]ダイアログが表示されます。詳細については、[動作オプション]ダイアログを参照してください。

[拡張オプション]ダイアログ



[カーソル同時実行]

カーソルの同時実行を指定します。同時実行とは、複数のクライアント(複数のコネクションでも同様)が、同一データを同時に使用する機能を指します。

READ_ONLY	カーソルは読込み専用です。カーソル系データ操作文の UPDATE 文(位置付け)、DELETE 文(位置付け) は実行できません。
LOCK	カーソル系データ操作文の UPDATE 文(位置付け)、DELETE 文(位置付け) は実行できます。カーソルは最下位レベルのロックを使用することによって、他のユーザが同時に同一リソースを更新または削除することを防ぎます。
ROWVER	カーソル系データ操作文の UPDATE 文(位置付け)、DELETE 文(位置付け) は実行できます。カーソルは行の変更履歴を使用したオプティミスティック同時実行制御を使用して、更新または削除を行います。オプティミスティック同時実行では、行が更新または削除されるまでロックされません。したがって、他のユーザが同時に同一リソースを更新または削除した場合、処理が失敗することがあります。
VALUES	カーソル系データ操作文の UPDATE 文(位置付け)、DELETE 文(位置付け) は実行できます。カーソルは行の値を使用したオプティミスティック同時実行制御を使用して、更新または削除を行います。

- データソースによって、指定に対する動作が異なる場合があります。
- データソースによって、指定に対する動作がエラーとなる場合があります。この場合、「READ_ONLY」を指定してください。
- カーソルのロックレベルはデータソースに依存します。

[ODBC カーソルライブラリ]

ODBC カーソルライブラリの使用の有無を指定します。ODBC カーソルライブラリは、通常データソースが行うカーソル処理を代替する機能を持ちます。ODBC カーソルライブラリを使用することによって、データソースが UPDATE 文 (位置付け)、DELETE 文 (位置付け) をサポートしない場合でも、これらを実現することができます。

ドライバカーソルを使用する	ODBC カーソルライブラリを使用しません。データソースがカーソル処理を行います。
カーソルライブラリを使用する	ODBC カーソルライブラリを使用します。ODBC カーソルライブラリがカーソル処理を行います。

カーソルライブラリを使用するを指定した場合

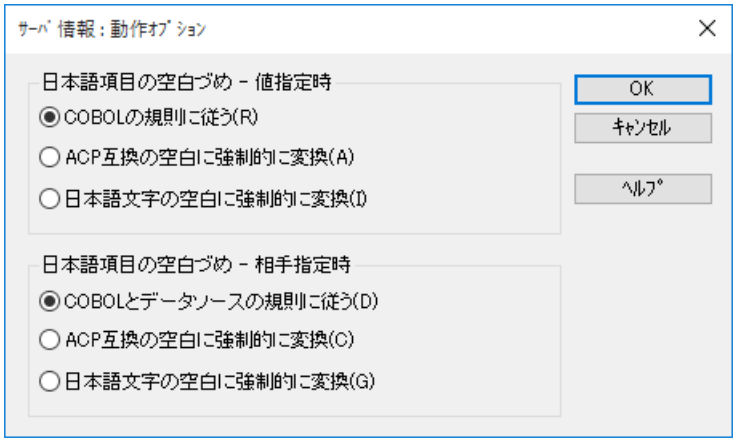
- UPDATE 文(位置付け)または DELETE 文(位置付け)を使用する場合は、カーソル宣言において値が一意な列を必ず 1 つ以上選択してください。ODBC カーソルライブラリは、UPDATE 文(位置付け)および DELETE 文(位置付け)をそれぞれ UPDATE 文(探索)および DELETE 文(探索)にシミュレートして実行します。そのため、一意な値を持つ列が選択されていない場合、複数行に処理が影響する場合があります。
- ODBC カーソルライブラリを使用しない場合と比較して、実行性能が若干劣化する場合があります。
- カーソル同時実行には、「VALUES」を指定してください。「VALUES」以外が指定された場合、カーソル処理がエラーになる場合があります。

[クエリータイムアウト時間]

データソースに対するクエリーのタイムアウト時間を秒単位で指定します。指定範囲は、0～4294967285 です。タイムアウト時間に 0 が指定された場合は、タイムアウトしません。

- データソースによって、指定に対する動作が異なる場合があります。
- データソースによって、指定に対する動作がエラーとなる場合があります。この場合、0 を指定してください。

[動作オプション]ダイアログ



日本語項目のホスト変数とデータソースとのデータの受け渡しの際の後続空白の種別を指定します。これには値指定時と相手指定時の 2 つの設定項目があります。

日本語項目の空白づめ - 値指定時

日本語項目のホスト変数が持つ値をデータソースに渡す場合の、後続空白の種別を指定します(例:INSERT 文)。

COBOL の規則に従う	COBOL システムの仕様に従った後続空白を入れます。
ACP 互換の空白に強制的に変換	後続空白として半角空白を入れます。
日本語文字の空白に強制的に変換	後続空白として全角空白を入れます。

日本語項目の空白づめ - 相手指定時

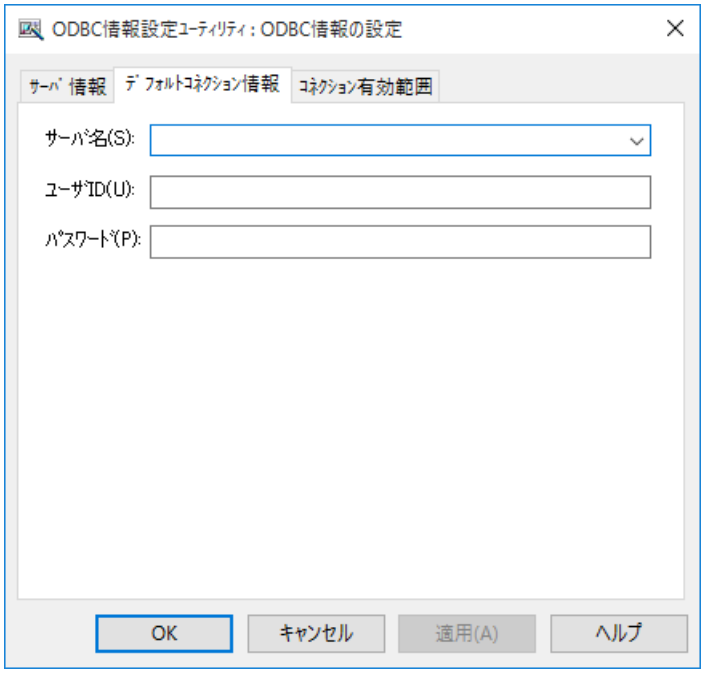
データソースから日本語項目のホスト変数に値を受け取る場合の、後続空白の種別を指定します(例:FETCH 文)。

COBOL の規則に従う	データソースや COBOL システムの仕様に従った後続空白が入ります。
ACP 互換の空白に強制的に変換	後続空白として半角空白が入ります。
日本語文字の空白に強制的に変換	後続空白として全角空白が入ります。

[デフォルトコネクション情報] ページ

このツールは互換のために残されています。NetCOBOL for .NET では、実行環境設定ユーティリティを使用することをお勧めします。

このページでは、CONNECT 文に DEFAULT を記述した場合に、コネクションを確立するために必要な情報を設定します。



[サーバ名] フィールド

デフォルトコネクションを確立する対象のサーバ名を指定します。COBOL ランタイムシステムは、このサーバ名をもとにしてサーバ情報を検索し、サーバ情報で定義されたデータソースに対してコネクションを確立します。

ここで指定するサーバ名は、サーバ情報としてすでに定義されているものを指定してください。

[ユーザ ID]フィールド

サーバ情報に定義されたデータソースを操作するためのユーザ ID を指定します。

[パスワード]フィールド

サーバ情報に定義されたデータソースを操作するユーザ ID のパスワードを指定します。

[注意]: パスワードはセキュリティのために入力時にアスタリスク(*)で表示されます。 入力には十分注意してください。

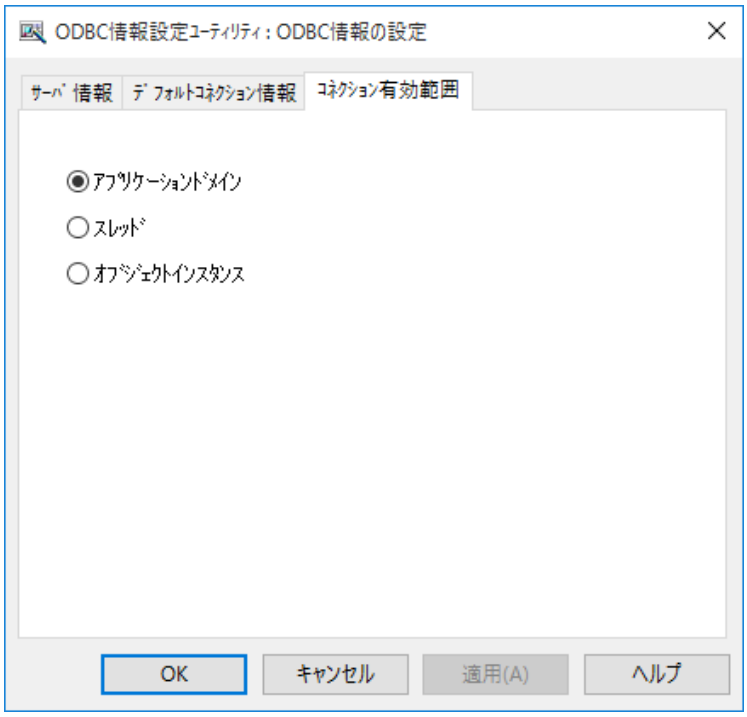
[補足]: パスワードはセキュリティのために暗号化されて ODBC 情報ファイルに格納されます。

NCOBU00165

[コネクション有効範囲] ページ

このツールは互換のために残されています。NetCOBOL for .NET では、実行環境設定ユーティリティを使用することをお勧めします。

このページでは、コネクションの有効範囲を指定します。コネクションの有効範囲とは、接続したコネクションの利用可能範囲です。



[アプリケーションドメイン]

接続したコネクションは、実行環境内（アプリケーションドメイン内）で利用できます。通常、シングルスレッドプログラムを動作させる場合に指定します。

注意

NetCOBOL for .NET で作成したプログラムは、すべてのマルチスレッドプログラムです。このため、アプリケーションドメインを指定してプログラムを動作させる場合、以下の注意が必要です。

- 複数のマルチスレッドプログラムが１つのコネクションを共有して利用する場合、いずれかのマルチスレッドプログラムでトランザクション処理を行った時、コネクションを共有する全てのマルチスレッドプログラムのデータ操作に影響します。
- 複数のマルチスレッドプログラムが複数のコネクションを利用する場合、各マルチスレッドプログラムにおいて最初に実行される埋込み SQL 文は、CONNECT 文、または、SET CONNECTION 文でなければなりません。 CONNECT 文、または、SET CONNECTION 文以外の埋込み SQL 文を実行した場合は、どのコネクションに対して操作をするか保証できません。これは、各マルチスレッドプログラムが各々の現コネクションを利用するためです。なお、複数のマルチスレッドプログラムが１つのコネクションを共有して利用する場合は必要ありません。
- カーソルを使用したマルチスレッドプログラムが複数のスレッドで実行された場合、

それぞれのスレッドが自分自身のカーソルを持ちます。したがって、カーソルをスレッド間で共有することはできません。

[スレッド]

接続したコネクションは、実行単位内（スレッド内）で利用できます。通常、プログラム定義のシングルスレッドプログラムをマルチスレッドプログラムに移行する場合に指定します。実行単位内で複数のコネクションを接続した場合、最後に実行された CONNECT 文、または、SET CONNECTION 文で指定されたコネクションが、実行単位の現コネクションになります。

クラス定義（オブジェクト指向プログラミング機能）に記述した埋込み SQL 文は動作しません。

[オブジェクトインスタンス]

接続したコネクションは、オブジェクトインスタンス内で利用できます。通常、オブジェクト指向機能を利用したシングルスレッドプログラムをマルチスレッドプログラムに移行する場合に指定します。 オブジェクトコネクション内で複数のコネクションを接続した場合、最後に実行された CONNECT 文または SET CONNECTION 文で指定されたコネクションが、オブジェクトインスタンスの現コネクションになります。

プログラム定義に記述した埋込み SQL 文は動作しません。

NCOBU00166

アプリケーション構成ファイル作成コマンド (CBRtoConfig.exe)

アプリケーション構成ファイル作成コマンド(CBRtoConfig.exe)は、従来の実行用の初期化ファイルの情報を読み込んで、アプリケーション構成ファイルを作成するコンソールコマンドです。 このコマンドは、従来の初期化ファイルをバッチファイルなどを使って変換する場合に利用します。一方、実行環境設定ユーティリティを使うと、GUI を使って従来の初期化ファイルをアプリケーション構成ファイルに変換することができます。

このコマンドを使用するには、.NET Framework 4 以上が必要となります。

このコマンドはプラットフォームごとに異なる exe ファイルが提供されます。 アプリケーション構成ファイルが設定しようとしている NetCOBOL for .NET アプリケーションの実行モードに応じて適切なコマンドを実行してください。

アプリケーションのターゲットプラットフォーム	使用する CBRToConfig.exe のプラットフォーム
x86	x86 プラットフォーム向け
x64	x64 プラットフォーム向け
AnyCPU	実際に実行環境上でアプリケーションが動作するモードに応じて、x86, x64 プラットフォーム向けのいずれか

各プラットフォーム向けのコマンドがインストールされる場所は以下のとおりです。

対象プラットフォーム	ランタイムシステムのインストールフォルダからの相対パス
x86	`Utilities¥CBRToConfig.exe`
x64	`Utilities¥CBRToConfig.exe`

ランタイムシステムのインストール場所については、「運用パッケージに含まれるツール」を参照してください。

実際には、上記の CBRToConfig.exe がすべてインストールされるわけではありません。 インストール先の Windows のバージョンと、インストールされた運用パッケージのエディションに応じて、 運用環境上でサポートされるプラットフォーム向けの CBRToConfig.exe がインストールされます。

例えば、x64 プラットフォーム向け Windows に NetCOBOL Enterprise Edition サーバ運用パッケージ for .NET をインストールした場合、 以下の CBRtoConfig.exe がインストールされます。 （Windows が C:ドライブにインストールされているとします）

対象プラットフォーム	CBRToConfig.exe のインストール場所
x86	`C:¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥CBRToConfig.exe`
x64	`C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥CBRToConfig.exe`

コマンド構文

```
CBRToConfig.exe [オプション] 実行用の初期化ファイル
```

オプション

CBRtoConfig コマンドは以下のオプションを指定することができます。

オプション	説明
/out:*config-file*	*config-file* には出力先となる構成ファイル名を指定します。出力されるファイルのエンコードは UTF-8 になります。このオプションおよび/utf8output オプションを省略した場合は、システムの現在の ANSI コードページのエンコードで標準出力に出力されます。
/utf8output	UTF-8 エンコードで標準出力に出力します。/out オプションが指定されている場合、このオプションは無視されます。
/section:*section-name*	実行用の初期化ファイル中に複数のセクションが存在する場合は、このオプションを指定します。*section-name* は情報を読み込むセクション名です。このオプションを省略した場合は、共通セクションから情報を読み込みます。
/entry:remove	実行用の初期化ファイルから情報を読み込んだ場合に、実行環境変数 @CBR_ENTRYFILE を検索した場合、変数値のエントリ情報ファイルが絶対パスで指定されている場合に限りエントリ情報も読み込まれます。正常に読み込まれた場合は、@CBR_ENTRYFILE の情報は"environmentSettings"セクションから削除されます。後述の/entry:noremove オプション、/entry:ignore オプションおよびこのオプションのいずれも指定されなかった場合、この動作が規定の動作となります。
/entry:noremove	実行環境変数@CBR_ENTRYFILE を検索した場合、変数値のエントリ情報ファイルが絶対パスで指定されている場合に限りエントリ情報も読み込まれます。@CBR_ENTRYFILE の情報は"environmentSettings"セクションから削除されません。
/entry:ignore	実行環境変数@CBR_ENTRYFILE を検索しても、エントリ情報は読み込みません。@CBR_ENTRYFILE の情報は"environmentSettings"セクションから削除されません。
/odbcinf:remove	実行用の初期化ファイルから情報を読み込んだ場合に、実行環境変数 @ODBC_Inf を検索した場合、変数値の ODBC 情報ファイルが絶対パスで指定されている場合に限り ODBC 情報も読み込まれます。正常に読み込まれた場合は、@ODBC_Inf の情報は"environmentSettings"セクションから削除されます。後述の/odbcinf:noremove オプション、/odbcinf:ignore オプションおよびこのオプションのいずれも指定されなかった場合、この動作が規定の動作となります。
/odbcinf:noremove	実行環境変数@ODBC_Inf を検索した場合、変数値の ODBC 情報ファイルが絶対パスで指定されている場合に限り ODBC 情報も読み込まれます。@ODBC_Inf の情報は"environmentSettings"セクションから削除されません。
/odbcinf:ignore	実行環境変数@ODBC_Inf を検索しても、ODBC 情報は読み込みません。@ODBC_Inf の情報は"environmentSettings"セクションから削除されません。
/nologo	このコマンドの著作権情報を表示しないようにします。

NCOBU00168

COBOL プログラミング

ここでは、.NET プラットフォームで、 従来の COBOL プログラミングの知識や機能を使用する場合のテクニックについて説明します。

.NET 特有の機能の使い方については、.NET プラットフォームでの COBOL プログラミングを参照してください。

このセクションの内容

オブジェクト指向プログラミング

オブジェクト指向プログラミングについて説明します。

ACCEPT 文および DISPLAY 文の使い方

ACCEPT 文および DISPLAY 文の使い方について説明します。

ファイルの処理

レコード順ファイル、行順ファイル、相対ファイル、索引ファイルなどの ファイルの処理について説明します。

印刷処理

印刷処理について説明します。

整列併合機能（SORT 文および MERGE 文の使い方）

SORT 文および MERGE 文の使い方について説明します。

CSV 形式データの操作

STRING 文および UNSTRING 文を使用した CSV 形式データの操作について説明します。

データベースアクセス

データベースへのアクセスについて説明します。

マルチスレッドアプリケーション

NetCOBOL for .NET によるマルチスレッドアプリケーションの作成方法と マルチスレッドの使用方法について説明します。

Unicode

ソースプログラムや登録集などのリソースのコード系および NetCOBOL for .NET アプリケーションの実行時データのコード系について説明します。

組込み関数の使用

組込み関数の使い方やコーディング時の注意点について説明します。

ポインタデータ項目の使用方法

獲得したメモリのハンドルをポインタデータ項目に格納して使用する方法および メモリの獲得・解放について説明します。

特殊な定数の書き方

プログラム名やファイル名などのシステムで定められた名前を指定する、 定数の書き方について説明します。

オブジェクト指向と従来機能の組合せ

クラス定義やメソッド定義などで使用できない機能について説明します。

広域最適化

コンパイラが行う広域最適化の内容および 使用上の注意事項について説明します。

例外処理

NetCOBOL for .NET における例外処理について説明します。

動的構造

NetCOBOL for .NET における動的構造の考え方について説明します。

他形式の外部 10 進互換モード

他形式の表現形式を扱うための翻訳オプションと変換関数について説明します。

セキュリティ

知っておくべきセキュリティ問題の概要を示します。

NCOBU00169

オブジェクト指向プログラミング

ここでは、オブジェクト指向プログラミングの概念について説明します。

このセクションの内容

プロパティ

プロパティについて説明します。

NCOBU00170

なぜオブジェクト指向 COBOL なのか

なぜ COBOL にオブジェクト指向プログラミングの機能が取り込まれたのか、また、それが何をもたらすのかについて簡単に紹介します。

進化する COBOL

COBOL は、最新の技術や開発手法を取り入れるため、定期的に規格が改定されています。オブジェクト指向プログラミングは、広く受け入れられる開発技術であり、COBOL でも最新の規格でその機能が追加されました。

ソフトウェア開発者の挑戦

多くの開発者は次のことを目指します。

- ユーザが本当に欲しいものを提供する
- ユーザが必要とする時期に提供する
- わかりやすく、理解しやすい機能を提供する
- 機能性の向上
- 技術的進歩への追随

これらを実現するために、開発プロセスをどうするか、開発のための製品、技術に何を採用するかという課題があります。この課題を解決する手段として、新しいソフトウェア開発技術である、オブジェクト指向の技法とそれに基づく.NET Framework があります。

ソフトウェア開発技法の進歩

1985 年の COBOL 規格で構造化プログラミングのための機構が COBOL 言語に導入されました。構造化プログラミングとは、プログラムの制御の流れとデータ構造を、わかりやすく記述するための手法です。これにより、ソフトウェア工学上の技術が生産性、品質、保守性に大きく貢献することが立証されました。また、これと合わせて以下の手法も採り入れられました。

- モジュール化
- 抽象データ型
- オブジェクト指向プログラミング

それぞれの手法について、簡単に紹介します。

モジュール化

モジュール化とは、プログラムを独立性の高いモジュールに分割し、さらにそのモジュールを、より小さいモジュールに分割する手法です。この手法を使うと、プログラムの規模が大きくても、全体の見通しは非常に良くなります。また、大規模プログラムの共同開発が可能になります。

モジュール化とは、見方を変えれば、プログラムの持つ機能を抽象化することです。プログラミングする対象を分析し、それを抽象的な機能単位に分けます。もちろん、内部コードがどうなっているかを考える必要はありません。これを段階的に繰り返すと、全体の見通しの良いプログラムを作ることができます。

COBOL では、PROGRAM-ID 段落と CALL 文でモジュール化を実現できます。

抽象データ型

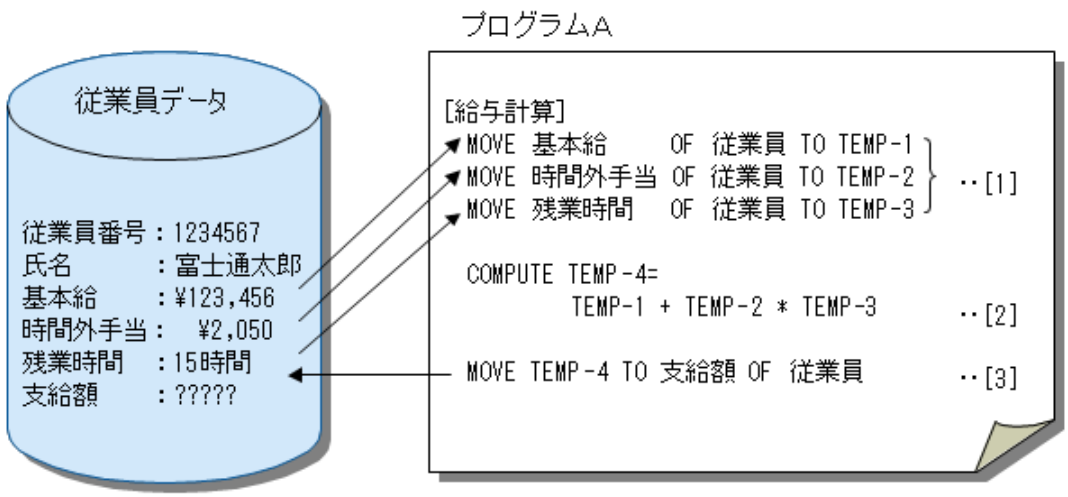
従来、プログラムで扱うデータは、数値、文字列などのコンピュータで表現しやすいものが中心でし

た。しかし、プログラムの対象となる現実の世界は、そんなに単純ではありません。たとえば、従業員管理システムで従業員データをモデル化しようとすると、従業員番号、氏名、住所、基本給など、さまざまな情報が集まった複合的なデータになります。データの構造化により、そのようなデータを定義できるようになりましたが、データ操作は、あいかわらず個々のデータごとに行なっていました。

例 1 に示すように、プログラム A から従業員データの"基本給"、"時間外手当"、"残業時間" および"支給額"を直接アクセスします。この場合、以下の問題があります。

- そのデータを扱う全てのプログラムは、データの構造を知っていなければならない。
- データ構造の個々の情報へのアクセスが制限されないため、誤った操作により破壊される危険性がある。
- データ構造を変更した場合、そのデータを利用していたモジュールを全て修正しなければならない。

例 1: データの直接アクセスの例



[1]:"基本給"、"時間外手当"、"残業時間"を作業域に転記する。

[2]:給与計算を行なう。

[3]:結果を"支給額"に転記する。

この問題を解決するために登場したのが、抽象データ型という考えです。

抽象データ型では、現実世界の"物"の持つ性質を、抽象的なデータ型としてモデル化します。抽象データ型には、以下の特徴があります。

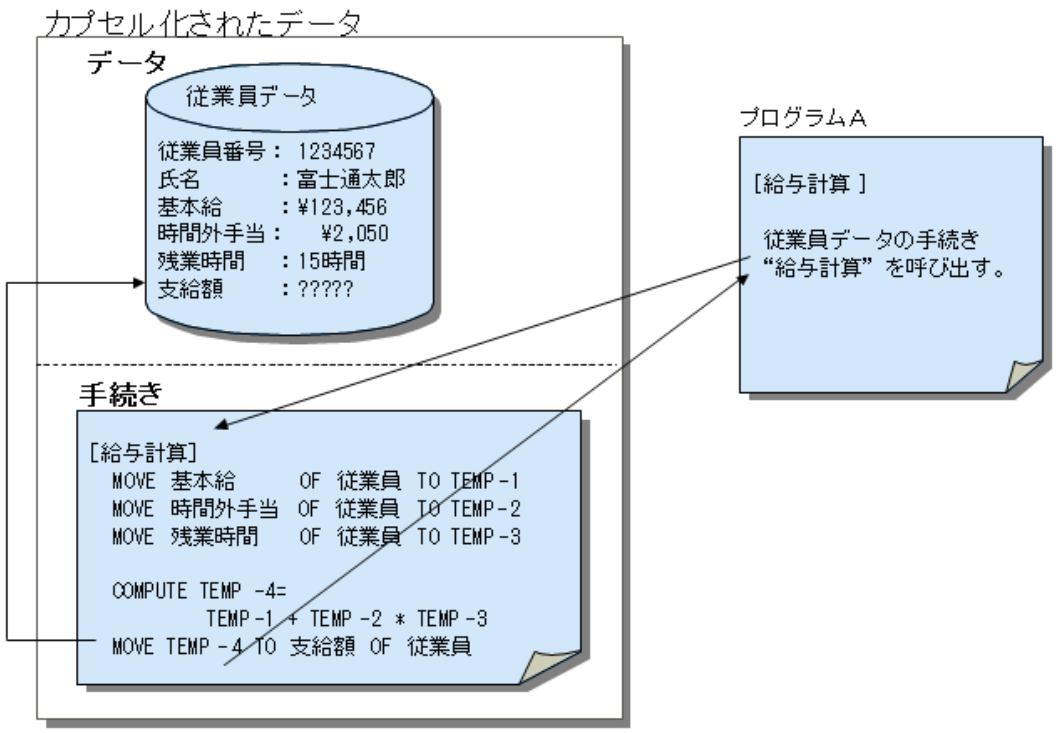
- 内部に、データとそれをアクセスする手続きコードを持つ。
- 内部のデータと手続きコードは、外部からは全く見えない。
- データをアクセスするためには、あらかじめ用意された手続きを呼び出さなければならない。
- 外部(そのデータを使用するプログラム)に公開されているのは、手続きのインタフェースだけ。

つまり、内部のデータおよび手続きコードは外部から完全に隠されており、外部仕様である手続きのインタフェースを通じてだけアクセスできます。このような考え方を「情報隠蔽」といいます。また、データと手続きを一体化することを「カプセル化」といいます。

以下の例 2 は、従業員データの中に"給与計算"という手続きを持たせています。プログラムから見えるのは"給与計算"手続きだけで、この手続きを呼ばないかぎり、給与関連の情報にはアクセスできません。

また、"給与計算"手続きは、従業員データの一部として管理されます。

例 2: 抽象データ型を使用した例



このような構成にしておくと、それぞれのデータ型との接点は公開されたインタフェースだけになります。そのため、内部の実現方法(例えば、内部のデータ構造や手続きコード)を変更しても、インタフェースの変更がなければ、それを利用するプログラムには影響ありません。つまり、変更に対する影響を局所化でき、保守作業の負荷を大幅に減らすことができます。また、公開されているのはインタフェースだけであるため、あらかじめ決められた正しい方法以外ではアクセスできません。そのため、自然にプログラムの信頼性が向上します。

オブジェクト指向プログラミングの誕生

オブジェクト指向プログラミングは抽象データ型の延長として現れました。

開発されるプログラムがより大規模になるにしたがって、開発効率、保守効率を大幅に向上させることが望まれました。そのため、あらかじめ部品として用意されたプログラムを組み合わせて、新しいプログラムを作ることが考えられました。そして、モジュール化、抽象データ型の手法を用いた部品化が試みられましたが、これらの手法による部品化には、以下の欠点がありました。

- 再利用しにくい
- 柔軟性がない

これはプログラミングモデルの設計上の問題と、それを実現するプログラミング言語の機能上の問題からくる欠点です。オブジェクト指向プログラミングは、この欠点を容易に補うことができます。

オブジェクト指向プログラミングは、オブジェクト指向設計とそれを容易に実現するためのオブジェクト指向言語からなります。

オブジェクト指向設計は、プログラミングのための設計モデルを、より現実世界にあったものとして設計します。例えば、銀行が顧客と取引をしている状況をオブジェクト指向で考える場合、顧客オブジェクトと取引オブジェクトが存在することになります。そして、そのオブジェクトは、顧客と取引の操作をモデル化していると説明することができます。オブジェクト指向デザインは、より現実的な形で定型化するものであり、それによりコードも理解しやすくなります。

オブジェクト指向設計を利用するためには、必ずしもオブジェクト指向言語を使用する必要はありませんが、オブジェクト指向言語を使用すると、デザインからコーディングまでの作業を簡略化することができます。また、柔軟に再利用しやすいコードを生み出すことができます。

オブジェクト指向プログラミングにより、開発効率、保守効率を大幅に向上することができます。

NCOBU00172

最良のプログラム言語 ～オブジェクト指向 COBOL～

COBOL は、長い間ビジネスアプリケーションとして最適な言語であると認識されてきました。いくつもの言語がある中で、コードの理解しやすさ、保守性、ファイルの操作性に加え、ビジネスデータの構造をサポートする点が評価されてきました。これらの利点を維持しつつ、オブジェクト指向プログラミングの技法に基づく.NET Framework 技術を採用し、COBOL 言語は今後もビジネスアプリケーション開発のための最適な言語であり続けます。

オブジェクト指向プログラミングの指針

オブジェクト指向は、プログラムの保守性を高めるために誕生しました。

- 保守性向上の決め手は、一度だけ本当によいものを作ってそれを何度も使うこと。「再利用」がポイント。
- 「再利用」がしやすい設計を議論する上での考え方(継承、情報隠蔽、ポリモフィズム、パターンなど)を提供してくれるのが「オブジェクト指向」。

ここで重要なのは、「オブジェクト指向プログラミング」がプログラムの保守性を高めるわけではなく、「オブジェクト指向」の考え方に基づいて設計されたプログラミングがプログラム保守性を高め、生産性や品質を向上させるという点です。

オブジェクト指向プログラミングの指針

オブジェクト指向の考え方に基づいて、以下の指針でプログラムを設計することで、「再利用」しやすいプログラムを作成します。

- 実在する"もの"および概念上の"もの"を全てオブジェクトとして扱い、オブジェクトを中心にしてプログラムを作成する。
- データ構造を変更した場合に、そのデータを利用していたモジュールを全て変更しなくてもいいように、データをカプセル化する。(抽象データ型)
- 一部の手続きを変更した場合、他のコードを変更しなくてもいいように、手続きをメソッドとしてモジュール化する。
- 継承を用いて関連するオブジェクトの属性と振る舞いを共用化することで、コードの再利用性を向上させる。
- コーディングの生産性を高めるため、クラスのコードを再利用する際に特異なコードだけを追加できるような構造にしておく。
- 1 つの操作が対象となるオブジェクトによって異なる振る舞いができるように、プログラムに柔軟性を持たせる。(これを多態性(ポリモフィズム)といいます)

NCOBU00174

オブジェクト指向プログラミングの概念

ここでは、以下のオブジェクト指向プログラミングの概念について説明します

- オブジェクト
- クラス
- オブジェクトインスタンス
- メソッド
- スタティックメンバー
- カプセル化
- 継承
- 多態
- 適合

オブジェクト

オブジェクト指向プログラミングにおいて核となる概念がオブジェクトです。オブジェクトは、実在する"もの"や概念上の"もの"をモデル化したものです。プログラム上では、オブジェクトは、"データ"と"手続き"をカプセル化したものとして表現されます。

例えば、銀行と顧客の場合、顧客は、名前、住所、銀行口座を持っています。そして、口座からお金を入金したり引き出したりします。それをオブジェクト指向の観点からみると、顧客オブジェクトと口座オブジェクトが存在し、 顧客オブジェクトは、名前と住所を、口座オブジェクトは、差し引き金額と前回の取引金額を、属性としてもっていることになります。 顧客オブジェクトは、お金の入金と引き出しするために口座オブジェクトにメッセージを投げます。この場合は、オブジェクトは顧客ごと、口座ごとに作られます。

NCOBU00176

クラス

オブジェクト指向プログラミングにおいて、同じ属性と振る舞いを持つオブジェクトをグループとして識別する必要があります。これらのグループをクラスと呼びます。各クラスには、属性(データ)と振る舞い(手続きコード)を定義します。

オブジェクト指向のアプリケーションを作成する時は、各クラスを定義します。クラスには、属性を表現するためのデータとオブジェクトの振る舞いを実装するための手続き(メソッド)のコードが含まれます。アプリケーションを実行する際に、これらのクラス定義はオブジェクトインスタンスを生成するために用いられます。

オブジェクトインスタンス

あるオブジェクト指向のシステムにおいて、新規に顧客オブジェクトを作成するとします。この場合、顧客の属性と振る舞いを定義した顧客クラスからメモリ上にオブジェクトインスタンスを生成します。１つのクラスから、複数のオブジェクトインスタンスを作成することができます。これらは同じ形式(データ構造と手続き)を持つものですが、互いに独立しています。それぞれのオブジェクトインスタンスは異なる値をデータとして持つことができます。

オブジェクトインスタンスは、インスタンスコンストラクタによって生成されます。NetCOBOL for .NET では、"NEW" というメソッド名がコンストラクタとして予約されています。もし、コンストラクタで特定の初期化処理などを行う必要がある場合は、"NEW"というオブジェクトメソッドを定義します。その場合は、デフォルトコンストラクタに代わり、定義した"NEW"メソッドが実行されます。

NCOBU00178

メソッド

それぞれのクラスに関連付けられた手続きは、メソッドと呼ばれます。メソッドは、モジュールやサブルーチンの一種と考えることができます。クラスには、そのデータを操作するに必要なだけのメソッドを定義することができます。

メソッドは、以下の情報で識別されます。

- メソッドの名前
- パラメタ個数と各パラメタのデータ型
- メソッドが属するクラス

クラスが異なれば、同じ名前のメソッドを使用することができます。また、同じクラスの中でもパラメタ個数やパラメタのデータ型が異なる場合、同じ名前のメソッドを定義すること(オーバロード)ができます。

メソッドの呼出しでは、メソッドの呼出し時に指定された、パラメタ個数やパラメタのデータ型と一致するメソッドが、複数の同名メソッドの中から選び出されます。

メソッドを呼び出すには、メソッド名やパラメタを正確に指定する必要があります。

スタティックメンバー

それぞれのクラスで共通なデータなどをスタティックメンバーといいます。オブジェクトインスタンスは、1 つのクラスから複数作成することは可能ですが、スタティックメンバーは、クラスに 1 つしか存在しません。また、オブジェクトインスタンスは、生成後、破棄することができますが、スタティックメンバーは、そのクラスを使用しているアプリケーションが終了するまで存在しつづけます。

スタティックメンバーには、そのクラスの全てのオブジェクトインスタンスに関連するデータを管理します。 たとえば、オブジェクトの総数をカウントするような場合に使用されます。

NCOBU00180

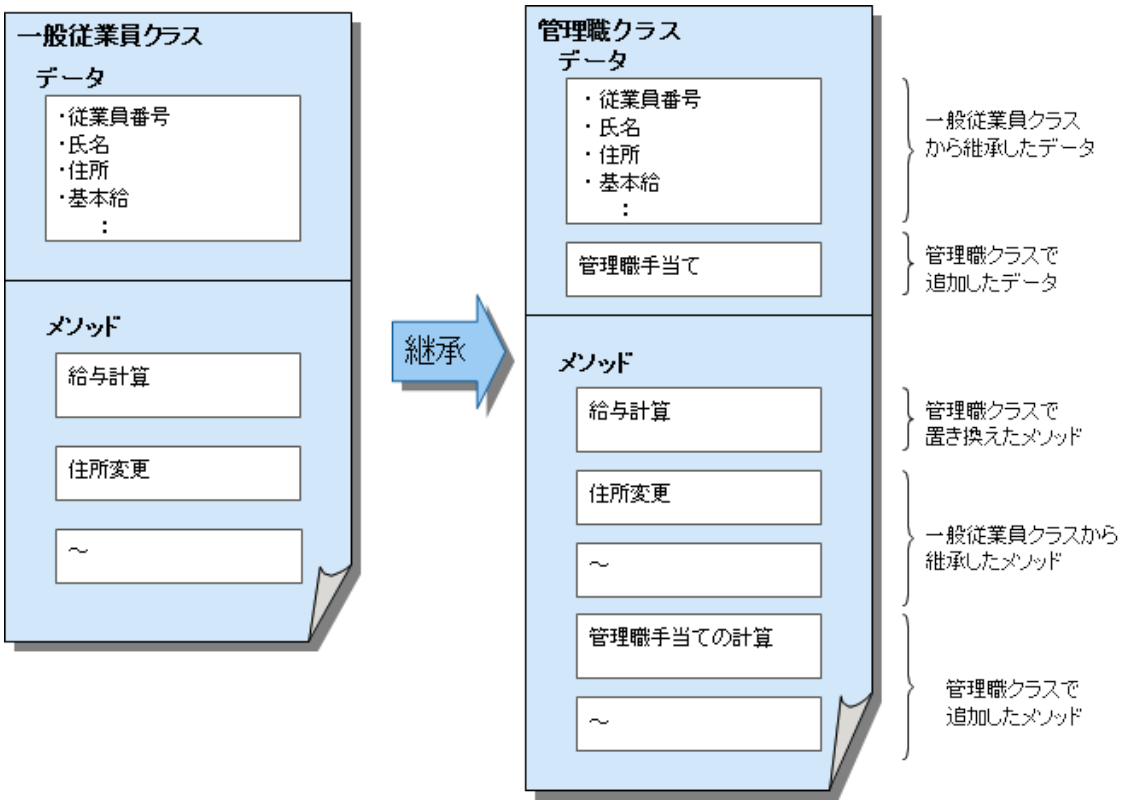
カプセル化

一般にカプセル化は、オブジェクト指向において情報隠蔽のために使用される用語です。情報隠蔽とは、あるオブジェクトに対するオブジェクトデータの参照や更新は、そのオブジェクトの持つメソッドだけから可能にすることです。そのオブジェクトの適切なメソッドを呼ぶことによって、他のプログラムや他のオブジェクトに属するメソッドから、オブジェクトのデータをアクセスすることができます。

継承

継承とは、他のクラスの性質をそのまま受け継ぐことです。新しいクラスを作る場合に継承を使用すると、既存のクラスの性質をそのまま引き継ぐことができます。さらに新しいデータを追加したり、新しいメソッドを追加したり、メソッドを置き換えたり、さまざまな改造ができます。つまり、既存のクラスを継承すれば、そこからの差分をコーディングするだけで新しいクラスを作ることができます。継承を使用することによって、オブジェクト指向の長所の１つである既存プログラムの再利用を実現できます。

たとえば、一般従業員クラスと管理職手当ての処理だけが異なる管理職クラスを作る場合、一般従業員クラスを継承して、新たに"管理職"クラスを作ります。管理職クラスでは、一般従業員クラスに対して、管理職手当ての処理を追加します。そのためには、それに関するデータ(管理職手当て)とメソッド(管理職手当て計算)の追加が必要です。また、管理職手当てを加算するために給与計算処理も変更する必要があります。一般従業員クラスと管理職クラスの関係は以下の図のようになります。



親クラス、子クラス、サブクラス、スーパークラス

ある親クラスから特性を引き継いだクラスを子クラス、またはサブクラスといいます。また、継承されるクラスのことを、親クラスまたはスーパークラスといいます。

上記の例では、一般従業員クラスが親クラス、管理職クラスが子クラスになります。

サブやスーパーという用語は、あるクラスから見て、上位クラス(スーパー)か下位クラス(サブ)かという意味です。

単一継承

1 つのクラスに対して親クラスを 1 つだけ継承する場合を単一継承といいます。NetCOBOL for .NET では、単一継承モデルをサポートしています。

多重継承

多重継承とは、1 つのクラスに対して、複数の親クラスを継承していることをいいます。
NetCOBOL for .NET では多重継承をサポートしていません。

NetCOBOL for .NET では、複数のインタフェースを実装する機能をサポートしています。

多態

多態とは、"1 つのものが複数の形態をとることができる" ことです。オブジェクト指向では、同じ名前のメソッドを呼び出したときに、オブジェクトの違いによって、異なる振る舞いをする時に使われます。

例えば、いくつかの異なるオブジェクトがそれぞれ"SaveData"というメソッドを持っているとします。これはオブジェクトの内容を保存するための一般的なメソッドとして使うことができますが、その処理の詳細はオブジェクトにより異なるかもしれません。

この場合、"SaveData"メソッドを呼び出すオブジェクトを変更するだけで、そのオブジェクトに最適な処理を行うことができます。

NCOBU00183

適合

オブジェクト指向プログラミングには、適合と呼ばれる概念があります。適合は、

- あるクラスが別のクラスに適合する
- あるインタフェースが別のインタフェースに適合する
- 送出し側のパラメタが受取り側のパラメタに適合する

というように使われます。適合とは、クラス間、インタフェース間あるいはパラメタ間の関係を表現するものです。

プログラマはオブジェクトインスタンスを操作したり、メソッドを呼び出したりする場合、適合を意識する必要があります。インタフェースおよびオブジェクトの適合を確実にするために、コンパイラは次のようなチェックを行います。

- メソッドに渡す引数が、メソッドの手続き部見出しの USING 指定に記述されているパラメタに適合する場合だけ受け渡しを行う
- 参照が有効な場合だけ、オブジェクト参照に代入する

たとえば、パラメタが３つあるメソッドを呼び出そうとしているのに、パラメタを２つしか渡さない場合、適合チェックでエラーとなります。また、呼び出すメソッドが要求するものと異なるデータ型のパラメタを渡した場合も、適合チェックでエラーになります。

オブジェクト指向 COBOL で追加された要素

オブジェクト指向 COBOL では、以下の要素が追加されています。また、.NET 固有の要素については、【.NET 固有】を表示しています。

見出し部

要素	要素の説明	COBOL ソース構造の要約
クラス定義	クラス内には、スタティック定義とオブジェクト定義を書くことができます。また、INHERITS 句を使用して他のクラスを継承することができます。 クラスは、クラス自身のデータ部と手続き部を持つことはできません。	[IDENTIFICATION DIVISION.] CLASS-ID. クラス名 [環境部] [スタティック定義] [オブジェクト定義] END CLASS クラス名.
スタティック定義	スタティック定義では、スタティックなデータやメソッドを定義します。	[IDENTIFICATION DIVISION.] STATIC. [環境部] [データ部] [手続き部] [メソッド定義] END STAIC.
オブジェクト定義	オブジェクト定義は、オブジェクトのテンプレートとして使われます。ここでは、オブジェクトデータの定義およびオブジェクトを操作するためのメソッドを定義します。	[IDENTIFICATION DIVISION.] OBJECT. [環境部] [データ部] [手続き部] [メソッド定義] END OBJECT.
メソッド定義	オブジェクトまたはクラスの振る舞いを実現するための手続きを記述します。OVERRIDE 句を利用することで、メソッドをオーバーライドすることもできます。	[IDENTIFICATION DIVISION.] METHOD-ID. メソッド名. [環境部] [データ部] [手続き部] END METHOD メソッド名.
インタフェース定義 【.NET 固有】	インタフェース定義では、実体を持たない抽象型を宣言します。	[IDENTIFICATION DIVISION.] INTERFACE-ID. インタフェース名. [環境部] [手続き部] [メソッド定義] END INTERFACE インタフェース名.
列挙型定義 【.NET 固有】	列挙型(ENUM)定義では、列挙型を定義します。列挙型は、名前の付いた整数を列挙したデータ型です。	[IDENTIFICATION DIVISION.] ENUM-ID. ENUM 名. [環境部] [データ部] END ENUM ENUM 名.
デリゲート定義 【.NET 固有】	デリゲート定義では、手続き(メソッド)とその手続きが実行されるオブジェクトとをカプセル化するデリゲートを宣言します。	[IDENTIFICATION DIVISION.] DELEGATE-ID. デリゲート名. [環境部] [データ部] [手続き部] END DELEGATE デリゲート名

上記以外に、アンマネージコードを呼び出す場合のプログラム原型定義もあります。

- IDENTIFICATION DIVISION ヘッダは省略できます。そのため、これらの構造では、"CLASS-ID"、"STATIC"、"OBJECT"、"METHOD-ID"、"INTERFACE-ID"、"ENUM-ID"または"DELEGATE-ID"で始めることもできます。
- これらの構造の詳細については、COBOL 文法書の「11.4 プログラム構造」を参照してください。

データ部

要素	要素の説明
オブジェクトデータ/スタティックデータ	オブジェクト定義のデータ部に記述すればオブジェクトデータとしての属性を持ち、スタティック定義のデータ部に記述すればスタティックデータとしての属性を持ちます。データのカプセル化を実現するため、他のクラスから直接オブジェクトデータ、スタティックデータへの参照や設定はできないようにしています。

手続き部

要素	要素の説明
INVOKE 文	INVOKE 文は、メソッドを呼び出す時に使われます。また、多態を実現することができます。

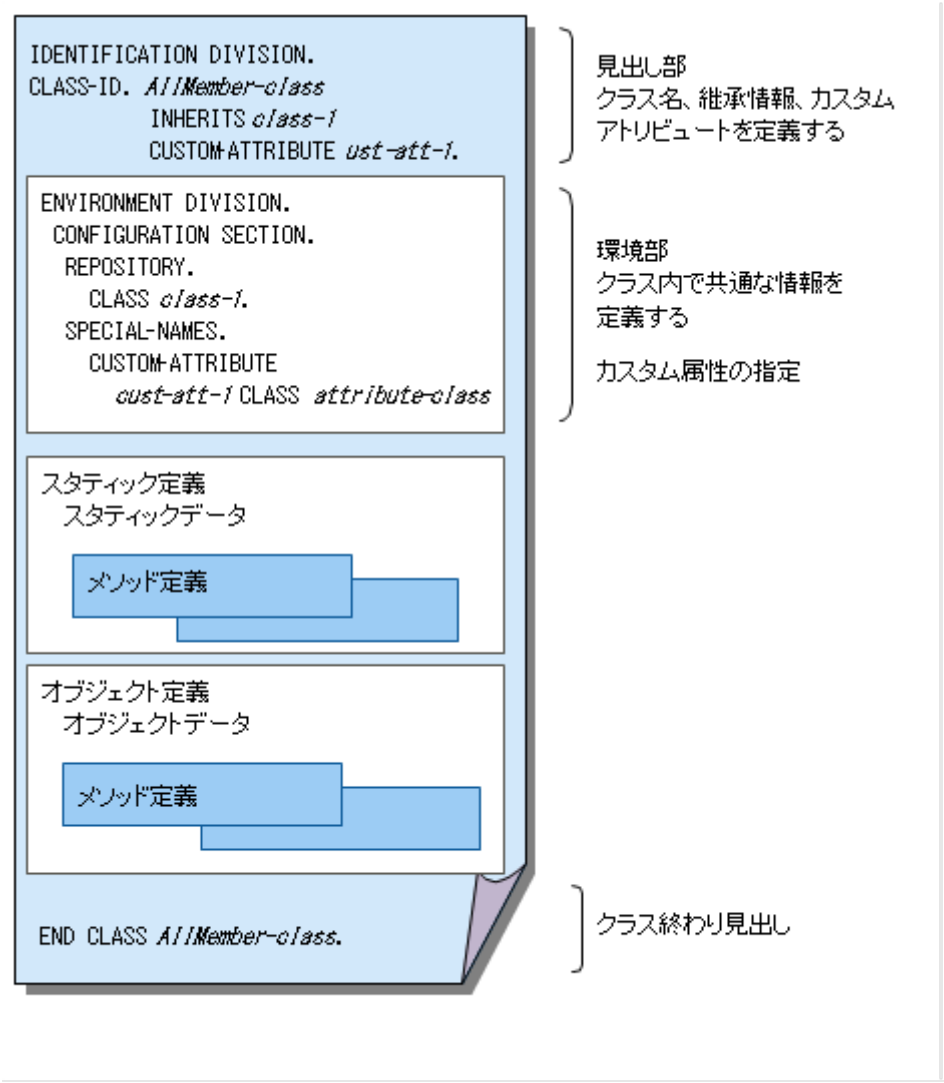
クラス定義

クラス定義は、オブジェトを定義するときに基本となる定義で、データとデータを操作するための手続き(メソッド)を 1 つにカプセル化したものです。

クラス定義は、スタティック定義とオブジェクト定義で構成されており、これらをいれておく入れ物(枠)のようなものです。そのため、クラスの継承を定義するための見出し部や、クラス内で共通な情報を定義するための環境部は定義できますが、データ部や手続き部を記述することはできません。

クラス定義の構造

クラス定義は、クラス名段落からはじまり、クラス終わり見出しで終わります。 クラス定義の一般的構造を以下に示します。



環境部で定義した情報の有効範囲

クラス定義の環境部で宣言された情報の有効範囲は、クラス定義内の全てのソース定義(スタティック定義、オブジェクト定義)です。環境部で宣言する情報には以下があります。

- リポジトリ段落で宣言されたクラス名
- 特殊名段落で宣言された機能名や呼び名、 記号定数など

NCOBU00186

スタティック定義

スタティック定義には、すべてのオブジェクトインスタンスで共通なデータ(スタティックデータと呼びます)を定義したり、スタティックデータを扱うためのメソッド(スタティックメソッドと呼びます)を定義したりします。

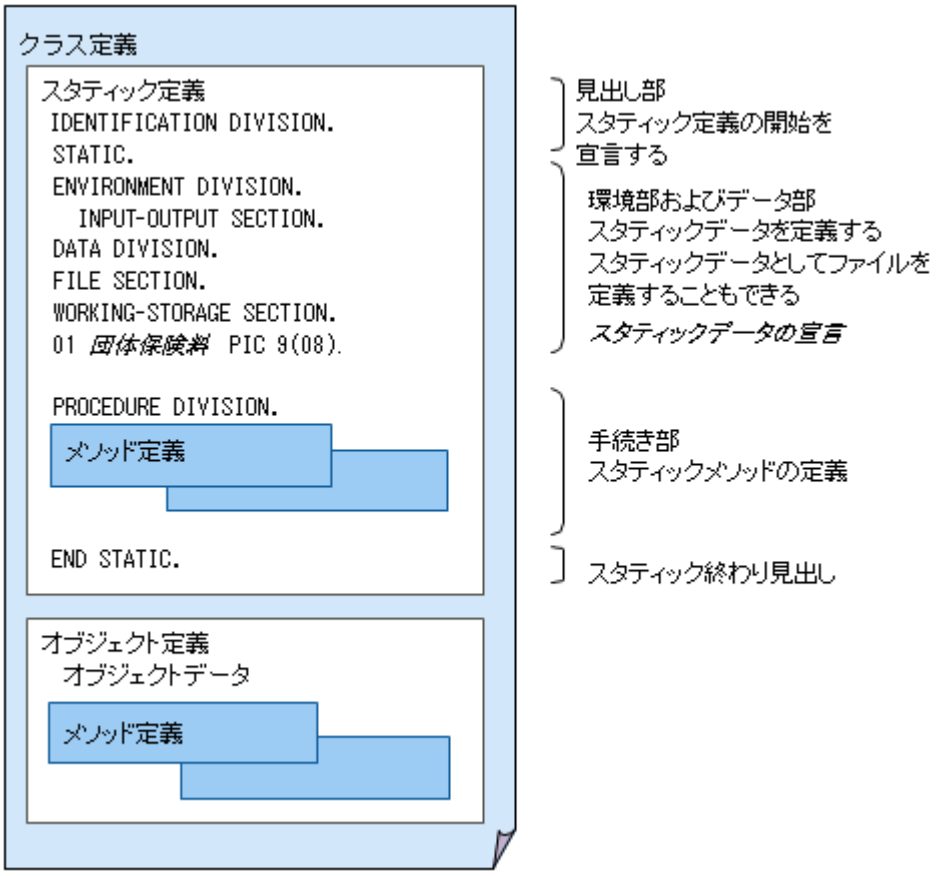
スタティックメンバーは、アプリケーションの実行が開始されてから終了するまで存在します。つまり、アプリケーションが実行されている間はスタティックメンバーは常に利用可能なので、各インスタンスに共通なデータを保持することができます。

スタティック定義の構造

スタティック定義は、スタティック段落で始まり、スタティック終わり見出しで終わります。

スタティック定義は、スタティックデータを定義するための環境部およびデータ部、スタティックメソッドを定義するための手続き部から構成されます。

これらの定義を必要としないクラスの場合、スタティック定義を省略することもできます。



スタティック定義の環境部およびデータ部で定義されたデータは、スタティックメソッドからアクセスすることができます。

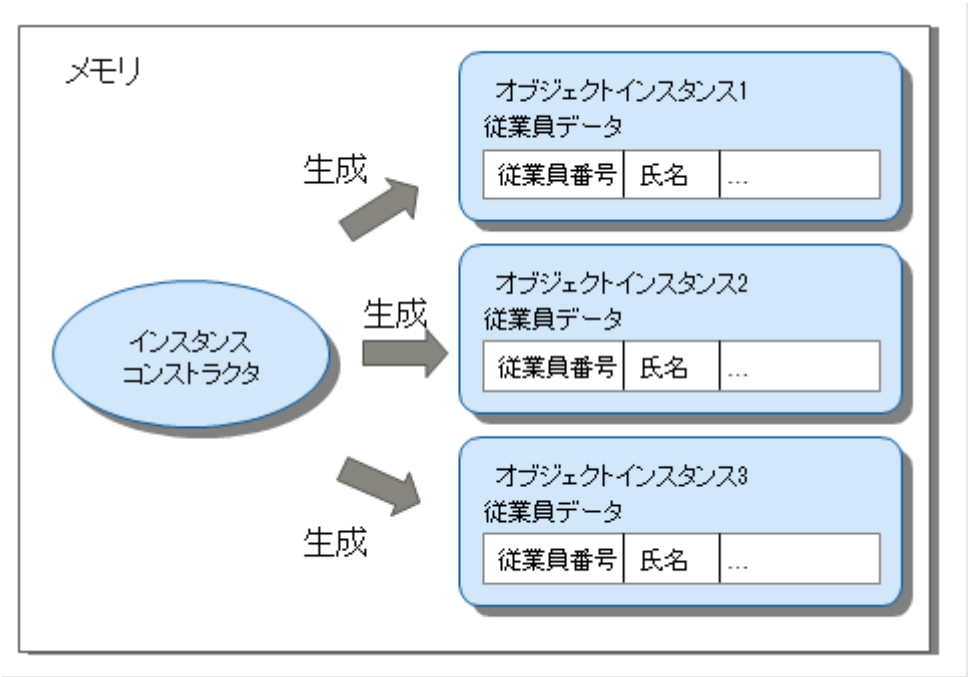
注意

スタティック定義のデータ部で宣言されたデータを、オブジェクトインスタンスのメソッドから参照するには、クラス名で修飾をすることで参照できます。

オブジェクト定義

オブジェクト定義には、オブジェクトインスタンスのデータ(オブジェクトデータと呼びます)を定義したり、オブジェクトデータを扱うためのメソッド(オブジェクトメソッドと呼びます)を定義したりします。

以下の図は、オブジェクトインスタンスの生成の様子を示しています。どのオブジェクトインスタンスも同じ構造の"従業員データ"を持っていることがわかります。"従業員データ"には、インスタンスごとに異なるデータ値を保持できます。

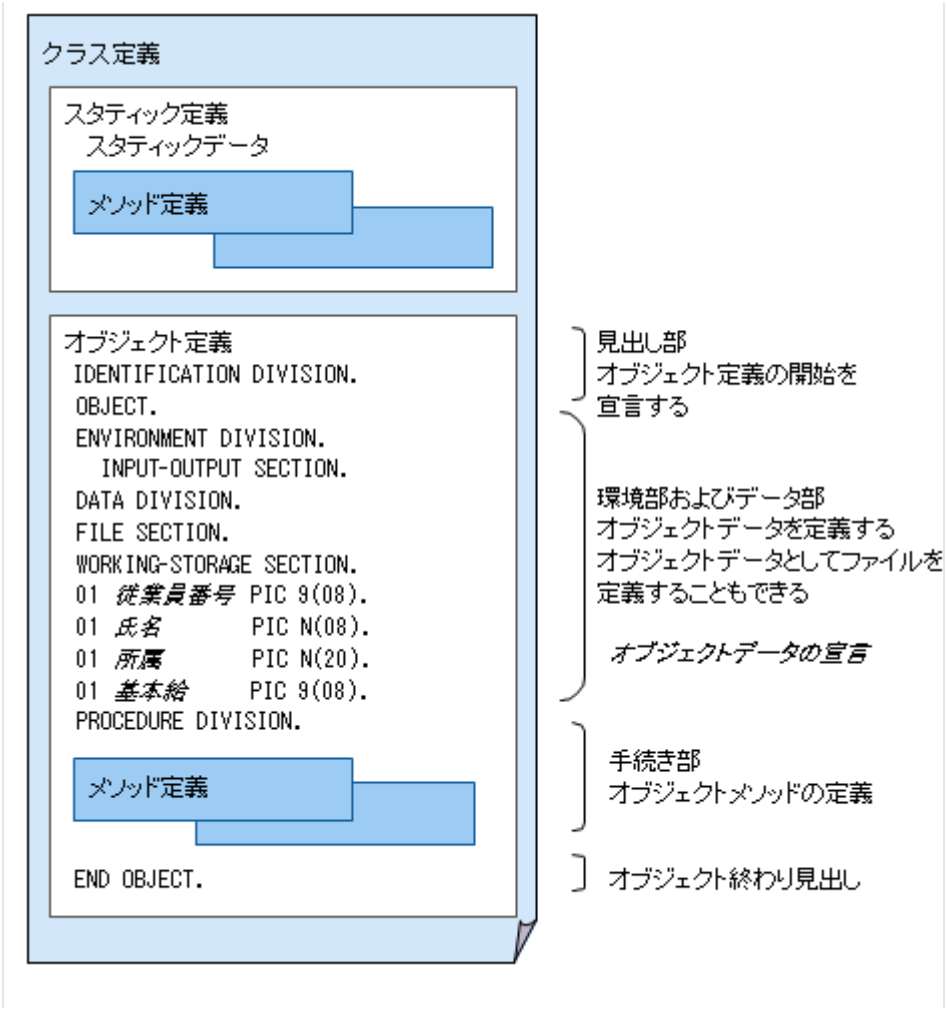


オブジェクト定義の構造

オブジェクト定義は、オブジェクト段落で始まり、オブジェクト終わり見出しで終わります。

オブジェクト定義は、オブジェクトデータを定義するための環境部およびデータ部、オブジェクトメソッドを定義するための手続き部から構成されます。

これらの定義を必要としないクラスの場合、オブジェクト定義を省略することもできます。



オブジェクト定義の環境部およびデータ部で宣言されたデータは、オブジェクトメソッドからアクセスすることができます。

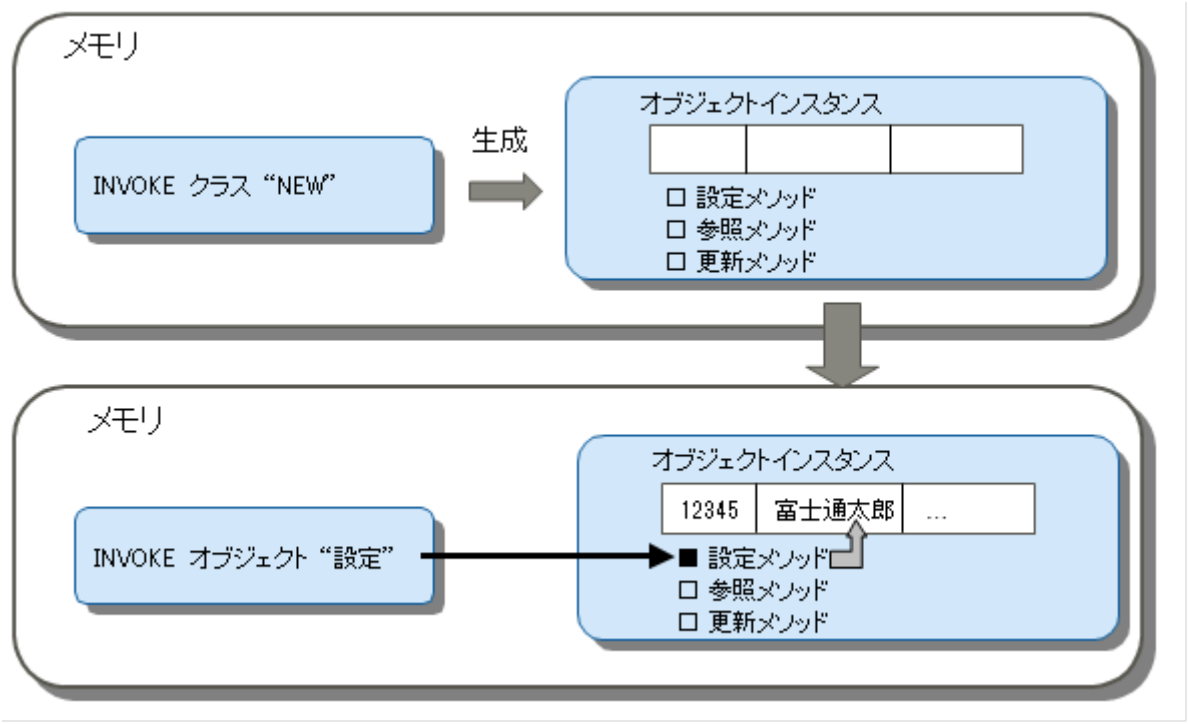
メソッド定義

メソッドには、以下 2 つの役割があります。

- オブジェクトやアプリケーションの機能を利用する
- オブジェクト内に隠蔽されたデータをアクセスする

オブジェクトデータやスタティックデータは、外部からデータを取得できないように隠蔽されています。このため、これらのデータを操作(取り出しや更新など)するためには、それぞれにメソッドを用意する必要があります。つまり、オブジェクトデータは、オブジェクトメソッドを介してしかアクセスできず、スタティックデータは、スタティックメソッドを介してしかアクセスできません。

たとえば、生成されたオブジェクトインスタンス内のデータは、そのインスタンス内のメソッドを使って設定します。



メソッド定義の構造

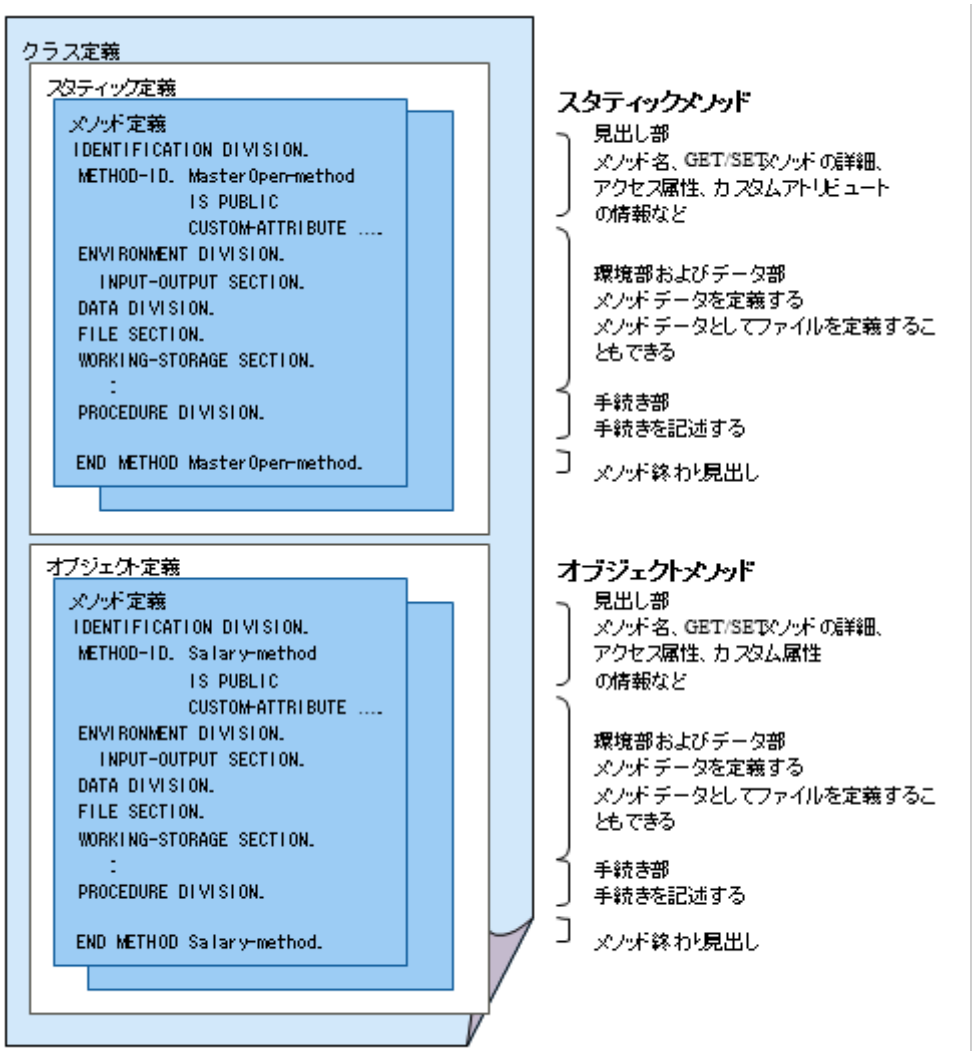
メソッドには、以下の 2 つがあります。

- スタティックメソッド
- オブジェクトメソッド

スタティックメソッドは、スタティック定義の中で、オブジェクトメソッドは、オブジェクト定義の中で設定します。

メソッド定義は、メソッドを定義するための見出し部、データを定義するための環境部およびデータ部、手続きを記述するための手続き部から構成されます。つまり、メソッド定義は従来の内部プログラムと同じ構造を持つことができます。また、メソッドは、それぞれのオブジェクトデータまたはスタティックデータに直接アクセスでき、再帰的に起動することも可能です。

スタティックメソッドとオブジェクトメソッドの構造は同じです。



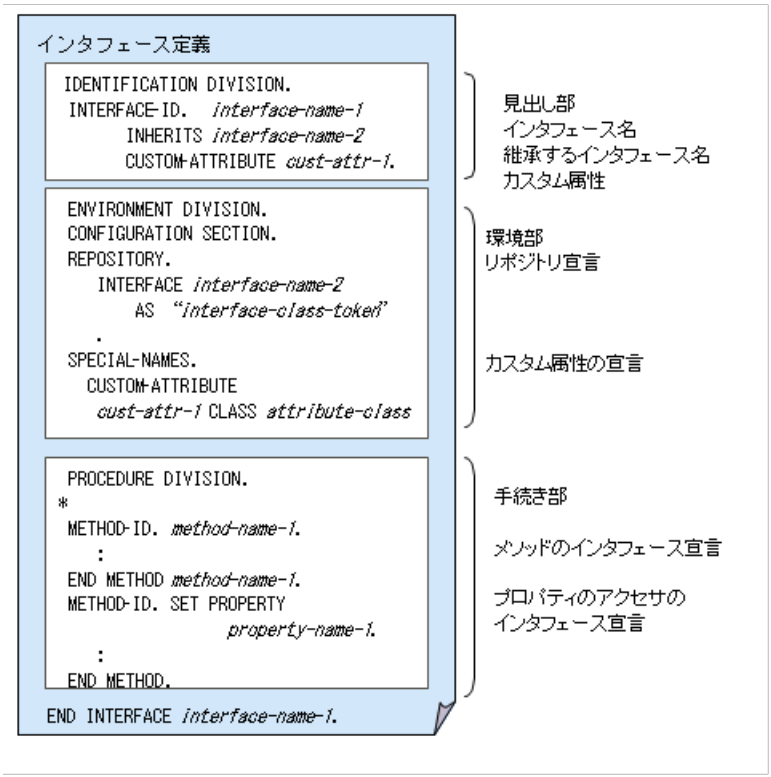
メソッド定義で宣言されたデータは、そのメソッド内の手続きからだけアクセスすることができます。メソッドの呼出し方法については、メソッド呼出しを参照してください。

インタフェース定義

インタフェース定義は、実体を持たない抽象型の宣言です。

インタフェース定義の構造

インタフェース定義は、インタフェース名段落から始まり、インタフェース終わり見出しで終わります。インタフェース定義の一般的な構造を以下に示します。



インタフェース定義の詳細については、 COBOL 文法書の 「11.5.1.7 インタフェース名段落(INTERFACE-ID)」を参照してください。

例: インタフェース定義

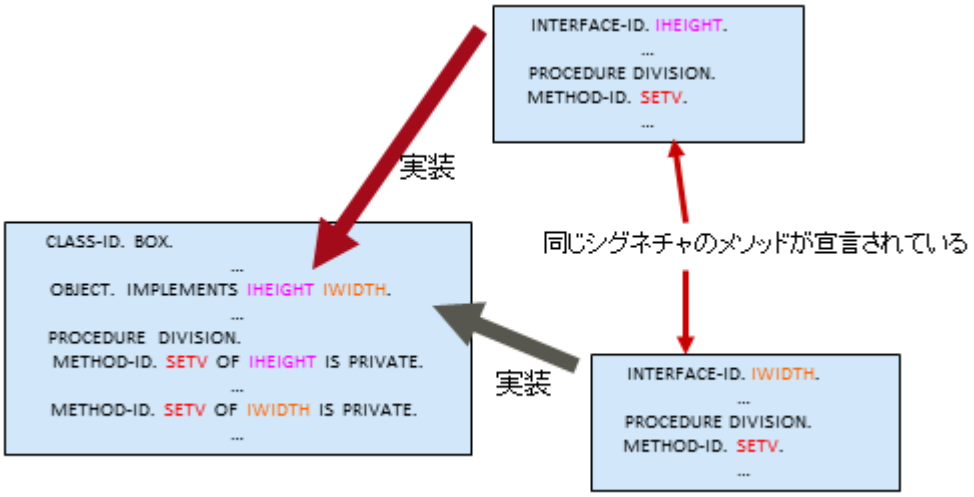
```
      INTERFACE-ID.  IFILEIO.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      REPOSITORY.
           CLASS STR AS "System.String"
           .
      PROCEDURE DIVISION.
     *
      METHOD-ID.  WRITEPROC.
      DATA       DIVISION.
      LINKAGE     SECTION.
      01 LSTR     USAGE OBJECT REFERENCE STR.
      PROCEDURE   DIVISION RETURNING LSTR.
      END METHOD  WRITEPROC.
     *
      METHOD-ID.  SET PROPERTY FILENAME.
      DATA       DIVISION.
      LINKAGE     SECTION.
      01 LSTR     USAGE OBJECT REFERENCE STR.
      PROCEDURE DIVISION USING BY VALUE  LSTR.
      END METHOD.
     *
      END INTERFACE IFILEIO.
```

注意

以下の条件を満たす場合、インタフェースのメソッドを実装する“METHOD-ID”段落に指定されたメソッド名の後に“OF インタフェース名 IS PRIVATE”を指定する必要があります。

- 複数のインタフェースに定義されている同じシグネチャのメソッドを実装する場合、かつ
- 同じシグネチャのメソッドで異なる実装をする必要がある場合

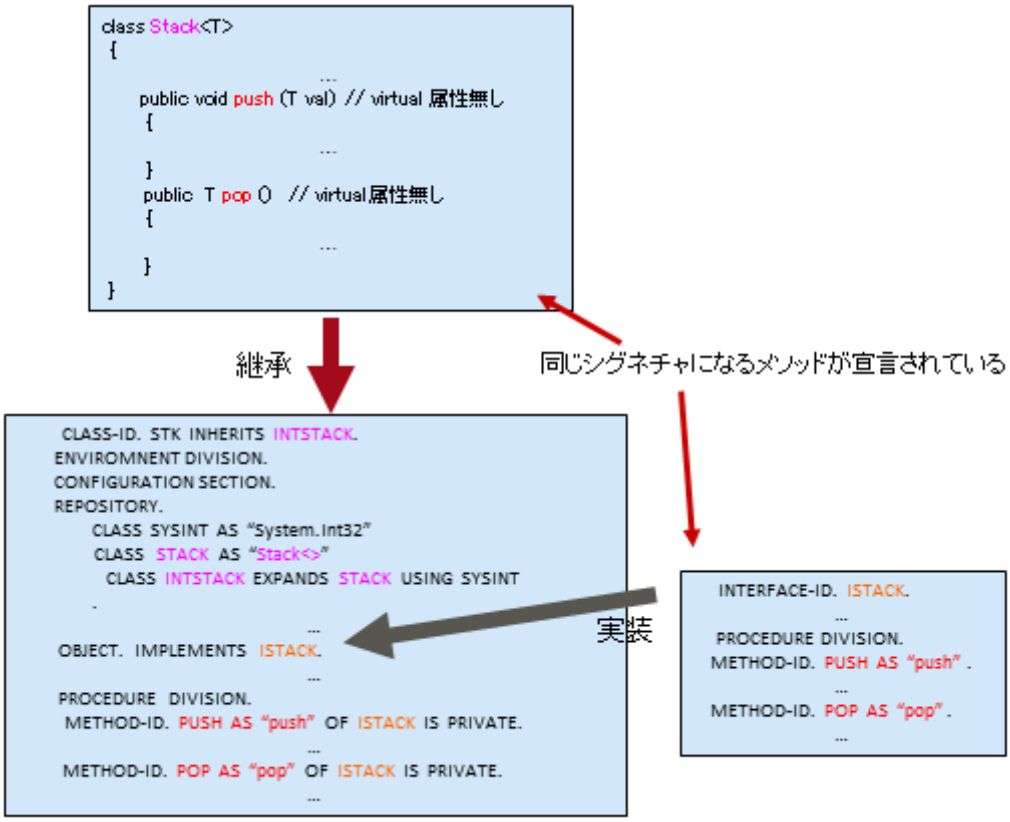
※ 名前の衝突による制限で、同じシグネチャをもつメソッドを複数定義できないため、インタフェース名で修飾する必要があります。



または

- 親クラスにインタフェースで定義されているメソッド名と同じシグネチャを持つメソッドが定義されている、かつ
- そのメソッドに virtual 属性が付いていない(非 virtual メソッド)

※ CLR の規則で、インタフェースのメソッドは virtual 属性を持つ必要がありますが、名前の衝突による制限からメソッド名をインタフェース名で修飾する必要があります。


class Stack<T>
{
...
public void push (T val) // virtual 属性無し
{
...
}
public T pop () // virtual属性無し
{
...
}
}
継承
同じシグネチャになるメソッドが宣言されている
CLASS-ID. STK INHERITS INTSTACK.
ENVIROMNENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
CLASS SYSINT AS "System.Int32"
CLASS STACK AS "Stack<>"
CLASS INTSTACK EXPANDS STACK USING SYSINT
.
...
OBJECT. IMPLEMENTS ISTACK.
...
PROCEDURE DIVISION.
METHOD-ID. PUSH AS "push" OF ISTACK IS PRIVATE.
...
METHOD-ID. POP AS "pop" OF ISTACK IS PRIVATE.
...
実装
INTERFACE-ID. ISTACK.
...
PROCEDURE DIVISION.
METHOD-ID. PUSH AS "push".
...
METHOD-ID. POP AS "pop".
...

NCOBU00190

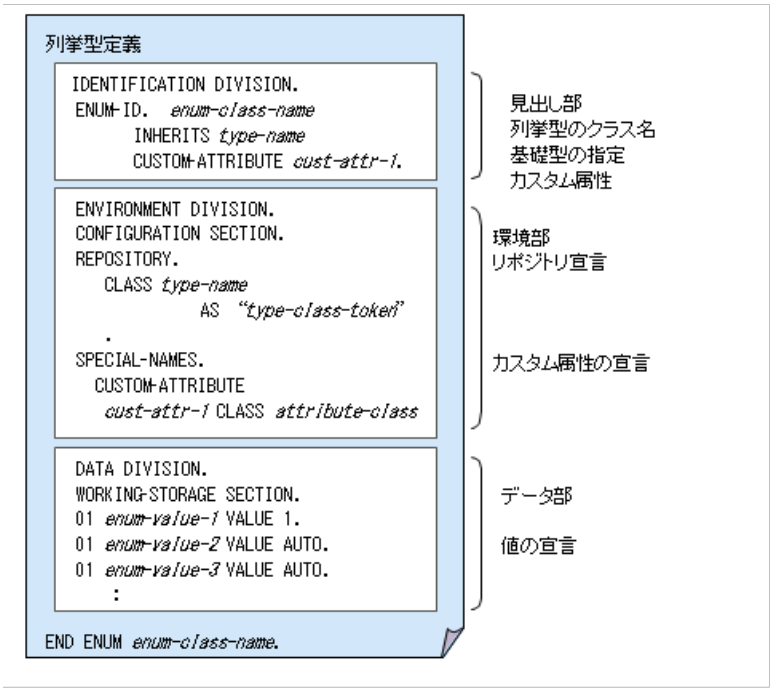
列挙型定義

列挙型(ENUM)定義は、名前の付いた整数からなるデータの定義です。

列挙型では、名前で参照できる整数値の集合を定義できます。また、コンパイラが暗黙的な値を割り当てることもできます。

列挙型定義の構造

列挙型定義は、ENUM 名段落から始まり、ENUM 終わり見出しで終わります。列挙型定義の一般的な構造を以下に示します。



列挙型定義については、COBOL 文法書の「11.5.1.8 ENUM 名段落(ENUM-ID)」も参考にしてください。

また、列挙型の詳細については、 Visual C#の enum および Visual Basic の列挙型の概要を参照してください。

例: ENUM 定義

```
IDENTIFICATION DIVISION.
ENUM-ID. WEEK INHERITS INT16.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
  REPOSITORY.
    CLASS INT16 AS "System.Int16"
    .
DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SUN VALUE 0.
 01 MON VALUE AUTO.
 01 TUE VALUE AUTO.
 01 WED VALUE AUTO.
 01 THU VALUE AUTO.
 01 FRI VALUE AUTO.
 01 SAT VALUE AUTO.
END ENUM WEEK.
```

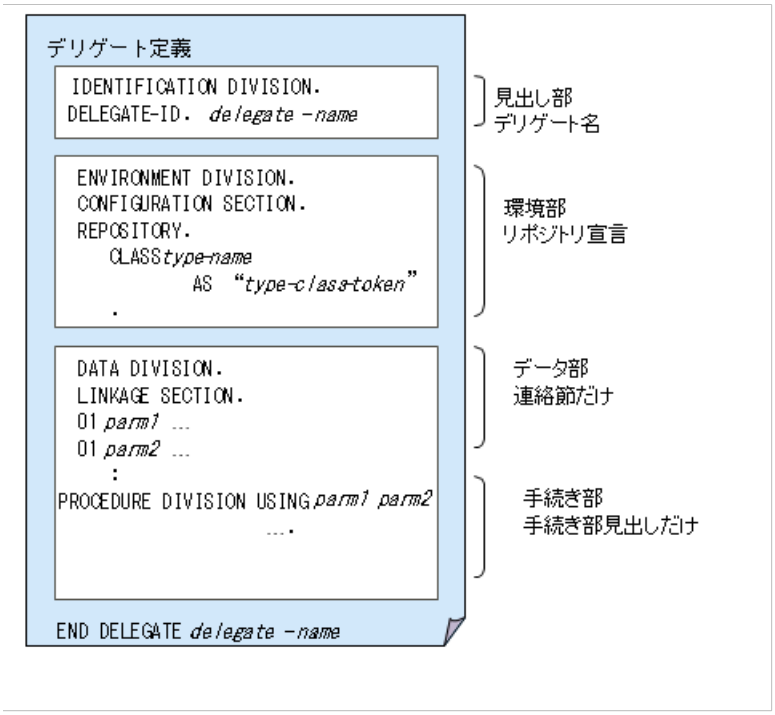
NetCOBOL for .NET で列挙型を利用する方法については、列挙体を使うを参照してください。

デリゲート定義

デリゲート定義は、手続き(メソッド)とその手続きが実行されるオブジェクトとをカプセル化するデリゲートの宣言です。

デリゲート定義の構造

デリゲート定義は、デリゲート名段落から始まり、デリゲート終わり見出しで終わります。デリゲート定義にはデータ部の連絡節と手続き部見出しだけを記述できます。処理手続きは記述できません。また、連絡節と手続き部見出しには、呼び出されるメソッドと同じインタフェースを持つように、データを宣言します。



デリゲート定義の詳細については、COBOL 文法書の「11.5.1.6 デリゲート名段落(DELEGATE-ID)」を参照してください。

また、デリゲートの一般的な情報については、.NET Framework のドキュメントのデリゲートを参照してください。

たとえば、以下のメソッド呼出しに使用するデリゲートの定義は以下のようになります。

【呼び出されるメソッド】

```
CLASS-ID. ShopClass.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
METHOD-ID. Fruit.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
 01 P1 BINARY-SHORT.
PROCEDURE DIVISION USING BY VALUE P1.
   EVALUATE P1
   WHEN 1
     DISPLAY "Applie"
```

```
          WHEN 2
            DISPLAY "Orange"
          WHEN 3
            DISPLAY "banana"
          END-EVALUATE
      END METHOD Fruit.
      END OBJECT.
      END CLASS ShopClass.
```

【デリゲート定義】

```
      IDENTIFICATION  DIVISION.
      DELEGATE-ID. ShopDelegate.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
      DATA DIVISION.
      LINKAGE SECTION.
       01 TYP BINARY-SHORT.
      PROCEDURE DIVISION USING BY VALUE TYP.
      END DELEGATE shopDelegate.
```

BINARY-SHORT 型の引数を持つメソッドに合わせて、デリゲートを定義します。

NetCOBOL for .NET でデリゲートを利用する方法については、デリゲートを使うを参照してください。

継承

ここでは、オブジェクト指向 COBOL における継承の利用方法について説明します。

オブジェクト指向プログラミングの概念でも説明したように、継承は、オブジェクト指向の主な特徴の 1 つです。この継承を利用することで、以下のメリットを得ることができます。

- 既存の部品の流用(再利用)が容易にできる
- システムの変更に対して、柔軟に対応できる
- システムクラスの属性や振る舞いを引き継ぐことができる

オブジェクト指向プログラミングの概念の継承も参考にしてください。

このセクションの内容

継承の定義方法

クラスを継承する場合の書き方について説明します。

継承の使用方法

継承を使用する場面について説明します。

アクセス可能なデータ、メソッドの範囲

クラスを継承して作成したクラスについて、クラス内におけるアクセス可能なデータおよびメソッドの範囲について説明します。

メソッドのオーバライド

メソッドのオーバライドについて説明します。

NCOBU00193

継承の定義方法

継承モデルを実現するには、最初に親クラスを作成しておきます。次に、子クラスのクラス名段落(CLASS-ID) に親クラス名を指定した INHERITS 句を指定します。このとき、子クラスの環境部のリポジトリ段落には、必ず親クラス名を指定します。

リポジトリ段落については、リポジトリを参照してください。

以下に、一般従業員クラス(EMPLOYEE)を継承した、管理職クラス(MANAGER)の例を示します。

```
 IDENTIFICATION DIVISION.
 CLASS-ID. MANAGER
    INHERITS EMPLOYEE.                                  *> [1]
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
   REPOSITORY.
     CLASS EMPLOYEE AS "Application.Employee".          *> [2]
**
** オブジェクト定義
**
 IDENTIFICATION DIVISION.
 OBJECT.
 DATA DIVISION.
**
** オブジェクトデータの宣言を追加                      [3]
**
 PROCEDURE DIVISION.
**
** オブジェクトメソッド定義
**
 IDENTIFICATION DIVISION.
 METHOD-ID. Compute-Bonus.
**
** メソッドのデータ、手続きを追加                      [4]
**
 END METHOD Compute-Bonus.
 END OBJECT.
 END CLASS MANAGER.
```

[1]: INHERITS 句に親クラス名(EMPLOYEE)を指定します。

[2]: リポジトリ段落のクラス指定子に親クラス名を指定します。この例では、Application.Employee というクラス名(外部名)を、EMPLOYEE という名前(内部名)に関連付けます。AS 指定で外部名を指定した場合、そのクラス名を参照する場合は、ソースプログラム中では内部名で参照します。(AS 指定のある場合、[1]の INHERITS 句に指定する親クラス名も内部名となります。)AS 指定の詳細については、COBOL 文法書の「11.3.4 外部名と内部名」および「11.6.1.3 リポジトリ段落」を参照してください。

[3]: オブジェクトデータを宣言します。継承する親クラスとの差分だけを追加します。

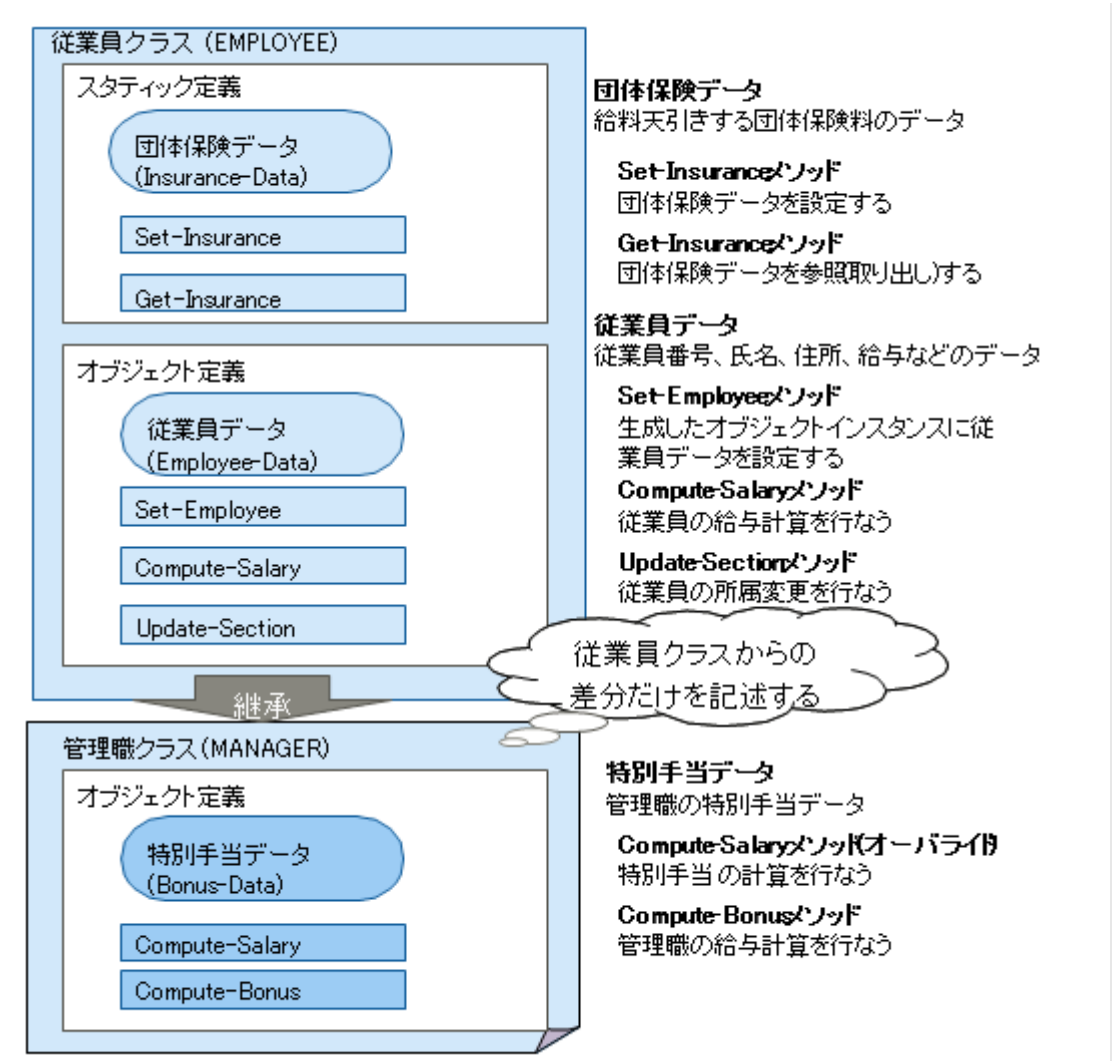
[4]: オブジェクトメソッドを定義します。継承する親クラスとの差分となる処理を追加します。

継承の使用方法

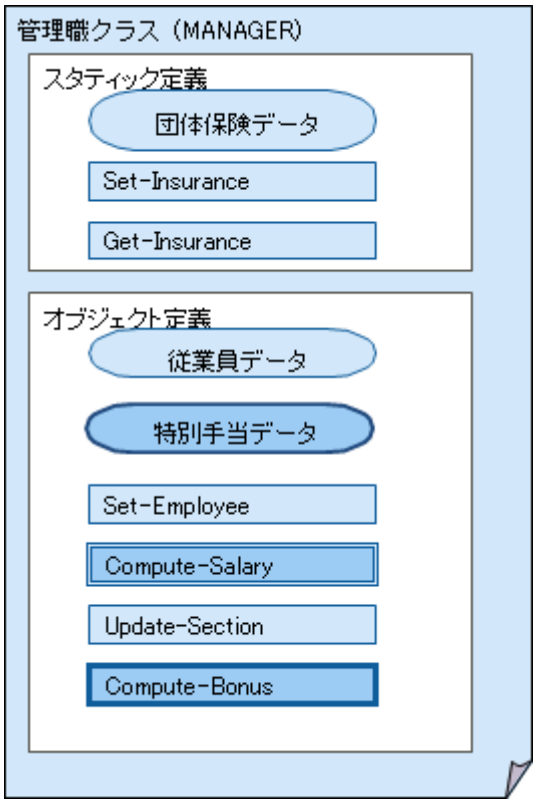
従来のプログラミング手法では、既存の部品(ルーチン)とよく似た機能を持った部品を作成する場合、既存のソースプログラムを複写し、複写したソースプログラムをベースにしてプログラミングする方法をとっていました。

ところが、オブジェクト指向の場合、既存の部品(クラス)との差分をコーディングするだけで同様のことが実現できます。つまり、「あるクラスが持つ機能をすべて引き継ぐ」ことが簡単にできるのです。これを継承と呼びます。

ここでは、具体例として、従業員クラス(EMPLOYEE)を継承した管理職クラス(MANAGER)を作成します。



この場合、管理職クラスの論理的な構成は、以下のようになります。



上図の太線は明示的に定義されたデータおよびメソッドを、細線は継承によって暗黙的に定義されたデータおよびメソッドを表しています。また、二重線はオーバライドされたメソッドを表しています。

メソッド呼出しの場合には、明示的に定義または暗黙的に定義を区別する必要はありません。暗黙的に定義されたメソッドも明示的に定義されたメソッドと同じように呼び出すことができます。

また、継承の階層(深さ)に制限はないので、必要に応じて継承を利用してください。ただし、あまり階層が深すぎると資源の管理が大変になるので、極端に深くならないように設計することをおすすめします。

なお、継承関係にあるクラスを表現する場合、あるクラスから派生したクラス(継承したクラス)を子クラス、継承されたクラスを親クラスと呼びます。

上図では、従業員クラス(EMPLOYEE)が親クラスで、管理職クラス(MANAGER)が子クラスになります。

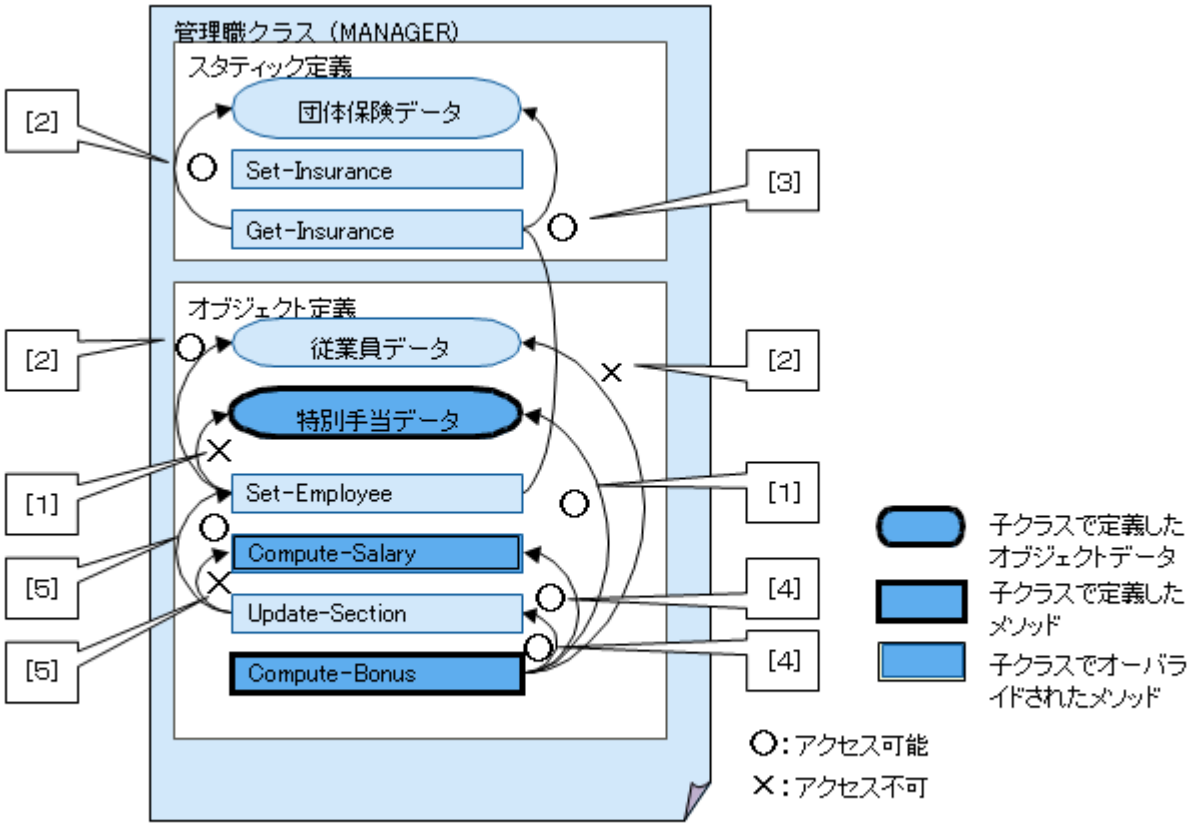
アクセス可能なデータ、メソッドの範囲

ここでは、クラスを継承して作成したクラスについて、クラス内におけるデータとメソッドのアクセス規則について説明します。ここでいうデータは、スタティックデータおよびオブジェクトデータを指します。

クラス内のデータとメソッドは、以下のように分けられます。

- 親クラスで定義されたデータとメソッド
- 子クラスで定義されたデータとメソッド(子クラスでオーバライドされたメソッドを含みます)

以下の図は、メソッドからデータへのアクセス範囲、メソッドからメソッドへのアクセス範囲について示しています。



メソッドからデータへのアクセス範囲

オブジェクトデータおよびスタティックデータは、そのデータが明に定義されたクラスの中で定義されたメソッドからだけアクセスできます。

[1] 子クラスで定義されたスタティックデータおよびオブジェクトデータは、子クラスで定義されたメソッドからだけアクセスできます。上の例の場合、子クラスで定義された「特別手当データ」は、子クラスで定義されたオブジェクトメソッド(Compute-Bonus メソッドおよび Compute-Salary メソッド)からだけアクセスできます。親クラスで定義されたメソッド(Set-Employee メソッドおよび Update-Section メソッド)からはアクセスできません。

[2] 親クラスで定義されたスタティックデータおよびオブジェクトデータは、親クラスで定義されたメソッドからだけアクセスできます。上の例の場合、親クラスで定義された「従

業員データ」は、親クラスで定義されたメソッド(Set-Employee メソッドおよび Update-Section メソッド)からだけアクセスできます。子クラスで定義されたオブジェクトメソッド(Compute-Bonus メソッドおよび Compute-Salary メソッド)からはアクセスできません。

[3] 直接データをアクセスできない場合、そのデータをアクセスできるメソッドを利用して、データにアクセスします。

- 子クラスで定義されたメソッドから、親クラスで定義したデータをアクセスしたい場合
- オブジェクトメソッドからスタティックデータをアクセスしたい場合
- 外部のクラスからデータをアクセスしたい場合

メソッドからメソッドへのアクセス範囲

[4] 子クラスで定義されたメソッドは、親クラスと子クラスのどちらで定義されたメソッドも呼び出すことができます。

[5] 親クラスで定義されたメソッドは、親クラスで定義されたメソッドしか呼び出せません。また、子クラスでオーバライドされたメソッドは呼び出すことができません。

メソッドのオーバライド

クラスを継承する場合、「メソッド名やインタフェースは同じでよいが、処理を少し変更(追加)したい」ということが多くあります。このようなとき、継承をあきらめて新規に似たようなクラスを作成する必要はありません。OVERRIDE 句を利用することによって、メソッドを再定義することができます。これを「メソッドをオーバライドする」と言います。

従業員クラスを継承した管理職クラスの場合を考えてみましょう。管理職クラスの場合、給与計算メソッド(Compute-Salary)で、「特別手当を加算する」必要があります。つまり、従業員クラスから継承した Compute-Salary メソッドに処理を加える必要があります。このような場合、OVERRIDE 句を利用してメソッドをオーバライドすることができます。

```
      IDENTIFICATION DIVISION.
      OBJECT.
      DATA DIVISION.
      PROCEDURE DIVISION.
      IDENTIFICATION DIVISION.
      METHOD-ID. Compute-Salary OVERRIDE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 給料  PIC 9(08).
      PROCEDURE DIVISION.
     **
     ** New code calulating:
     **    給料 = 基本給 + 特別手当 - 団体保険料
     **
      END METHOD Compute-Salary.
      END OBJECT.
```

メソッドのオーバライドは、直接の親クラスで明示的または暗黙的に定義されたメソッドのどちらに対しても行うことができます。また、親クラスでオーバライドされているメソッドをさらにオーバライドすることも可能です。ただし、インタフェース(パラメタ)はオーバライドされるメソッドと同じでなければなりません。

NCOBU00197

リポジトリ

リポジトリには、そのクラス内のメソッドを利用するために必要なインタフェース情報やプロパティの情報が含まれています。外部のクラスを使用するには、このリポジトリが必要になります。ここでは、リポジトリについて説明します。

このセクションの内容

リポジトリの概要

リポジトリとは

リポジトリには、クラス定義やインタフェース定義内のメソッドを利用するためのメソッドインタフェース情報やプロパティの情報が集められます。外部のクラスを使用する場合は、このリポジトリを使用するという宣言をリポジトリ段落に記述する必要があります。

リポジトリの格納場所

.NET Framework では、リポジトリはアセンブリファイルに保持されます。リポジトリ段落で、そのアセンブリファイルで使われている名前と、COBOL ソースプログラムの中で使われる内部的な名前とを結びつけて、クラスを利用することができます。

アセンブリファイルを利用する場合は、ビルド時に、/reference オプションでアセンブリファイルの名前を指定する必要があります。

リポジトリの内容

COBOL ソースプログラムでクラスを定義する場合、CLASS-ID 段落に記述した名前が、そのクラスの名前になります。クラスをコンパイルすると、そのクラスを参照、継承するのに必要な情報とその親クラスの全ての情報がアセンブリファイルに格納されます。

別の COBOL ソースプログラムからクラスを参照する場合、その COBOL ソースプログラムの中のリポジトリ段落に参照するクラスを指定します。この時、参照するクラスの親クラスまで指定する必要はありません。親クラスの情報は、子クラスのリポジトリ情報の中に含まれています。

NCOBU00199

リポジトリ段落

リポジトリ段落には、翻訳単位内で使われる、クラス名、プロパティ名、フィールド名、インタフェース名、プログラム原型定義名、デリゲート名および ENUM 名を指定します。

```
REPOSITORY.
    CLASS  クラス名 [AS "クラス名"]
    PROPERTY  プロパティ名 [AS "プロパティ名"]
    PROPERTY  フィールド名 [AS "フィールド名"]
    INTERFACE  インタフェース名 [AS "インタフェース名"]
    PROGRAM  プログラム原型名-1 [AS "プログラム名"]
    DELEGATE  デリゲート名 [AS "デリゲート名"]
    ENUM  ENUM 名 [AS "ENUM 名"]
   .
```

AS 指定は、AS に指定された外部名を、翻訳単位内で参照するための名前(内部名)と対応付けるために使用します。

リポジトリの使用方法

クラスの継承

継承は、親クラスのリポジトリを入力することで実現します。

例えば、従業員クラスを継承して管理職クラスを作成する場合、 管理職クラスの翻訳時に従業員クラスのリポジトリを入力します。この場合、以下のように CLASS-ID 段落の INHERITS 句と REPOSITORY 段落のクラス指定子に従業員クラスを指定します。

```
IDENTIFICATION DIVISION.
CLASS-ID. 管理職クラス INHERITS 従業員クラス.
ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  REPOSITORY.
    CLASS 従業員クラス.
```

リポジトリ段落に指定する親クラスは、直接の親クラスだけを指定します。例えば、従業員クラスに親クラスがあったとしても、管理職クラスの直接の親クラスでないため指定する必要はありません。

クラス、プロパティ、デリゲートなどの参照

もしプログラムの中で外部で定義された項目(クラス、プロパティ、フィールド、デリゲートなど)を参照する場合は、リポジトリ段落にその情報を記述する必要があります。

例えば、サンプルの"Walk the Dog"では、外部で定義されている項目を参照する場合は、リポジトリ段落に記述する必要があります。

```
CLASS-ID. WALK INHERITS FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FORM AS "System.Windows.Forms.Form"
    CLASS CONTAINER AS "System.ComponentModel.Container"
    CLASS BUTTON AS "System.Windows.Forms.Button"
    CLASS TIMER AS "System.Windows.Forms.Timer"
    DELEGATE EVENTHANDLER AS "System.EventHandler"
    PROPERTY PROP-INTERVAL AS "Interval"
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 COMPONENTS OBJECT REFERENCE CONTAINER.
01 BUTTON1 OBJECT REFERENCE BUTTON.
01 TIMER1 OBJECT REFERENCE TIMER.
METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TICK-EVENT OBJECT REFERENCE EVENTHANDLER.
01 CLICK-EVENT OBJECT REFERENCE EVENTHANDLER.
PROCEDURE DIVISION.
    INVOKE BUTTON "NEW" RETURNING BUTTON1.
    INVOKE TIMER "NEW" USING BY VALUE COMPONENTS RETURNING TIMER1.
    MOVE 7000 TO PROP-INTERVAL OF TIMER1.
```

NCOBU00201

オブジェクトインスタンスの操作

オブジェクトインスタンスの操作は、オブジェクトメソッドを呼び出して行います。オブジェクトメソッドを呼び出すには、「どのオブジェクトインスタンス」の「どのメソッド」かを指定します。「どのオブジェクトインスタンス」を表現するのに、オブジェクト参照項目と呼ばれるデータ項目を利用します。

ここでは、オブジェクトインスタンスの生成、参照および呼出しといったオブジェクトインスタンスの操作方法について説明します。

このセクションの内容

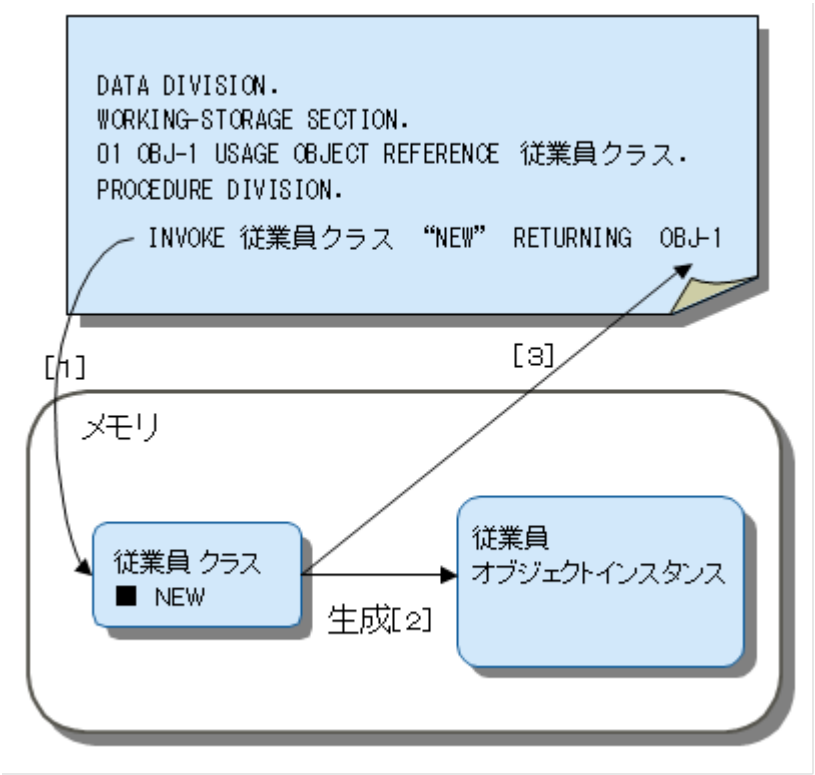
オブジェクトインスタンスの生成と参照

オブジェクトインスタンスを生成するには、NEW メソッドを呼び出します。これにより、生成されたオブジェクトのオブジェクト参照が返却されます。以降、このオブジェクトインスタンスを操作する場合には、このオブジェクト参照を使用します。

NEW メソッド

NEW メソッドを定義することで、すべての値型を論理的な ZERO の値に初期化することができます。NEW メソッドは、オブジェクト定義の中で独自に定義して使用する方法と、システムに備わっている NEW メソッドを使用する方法があります。どちらの場合でも、新しいオブジェクトインスタンスを生成し、オブジェクト参照を返します。NEW メソッドは、通常コンストラクタと呼ばれます。

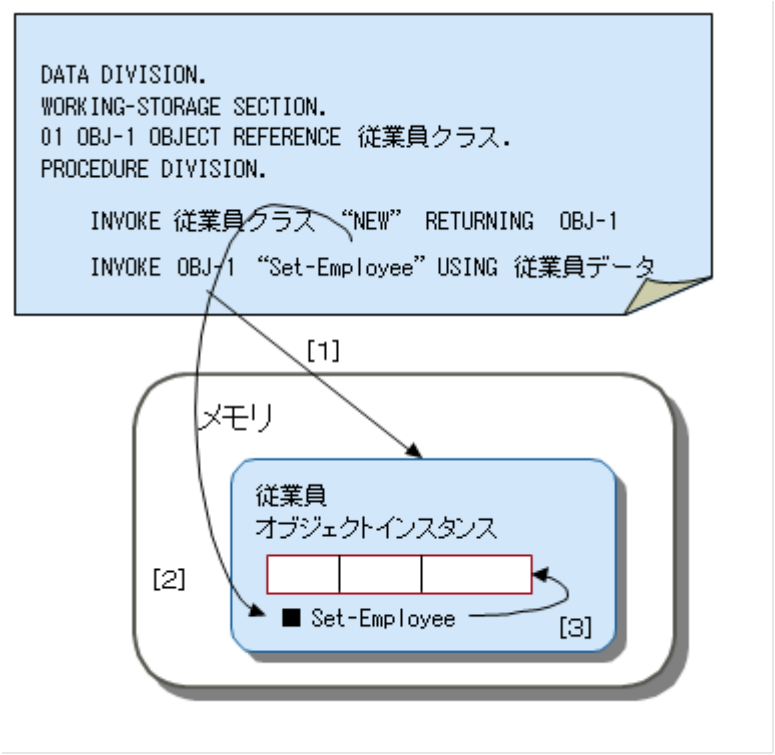
従業員クラスのオブジェクトインスタンスを生成するコードを以下に示します。



1. プログラム内で NEW メソッド、つまりコンストラクタを呼び出します。 独自の NEW メソッドを定義している場合には、独自の NEW メソッドのインタフェースに合わせて、INVOKE 文の USING 指定でパラメタを渡すことも可能です。
2. NEW メソッドが呼び出された時、NetCOBOL for .NET ランタイムシステムは、そのクラスで定義されている新しいオブジェクトインスタンスを生成します。インスタンスの生成の後、NEW メソッドが独自に定義されたものなら、NEW メソッドの手続き（例えばオブジェクトを初期化するルーチンの呼出しなど）が実行されます。
3. NEW メソッドは、それを呼び出す INVOKE 文の RETURNING に指定されたデータ項目にオブジェクト参照を返します。これは NetCOBOL for .NET ランタイムシステムによって実行されるため、NEW メソッドの中で書いたコードでは実行されません。すなわち、NEW メソッドの中で復帰項目を使うことはできません。

オブジェクトインスタンスの操作

生成したオブジェクトインスタンスを操作する場合は、処理対象となるオブジェクトインスタンスのオブジェクト参照を使って、オブジェクトメソッドを呼び出します。

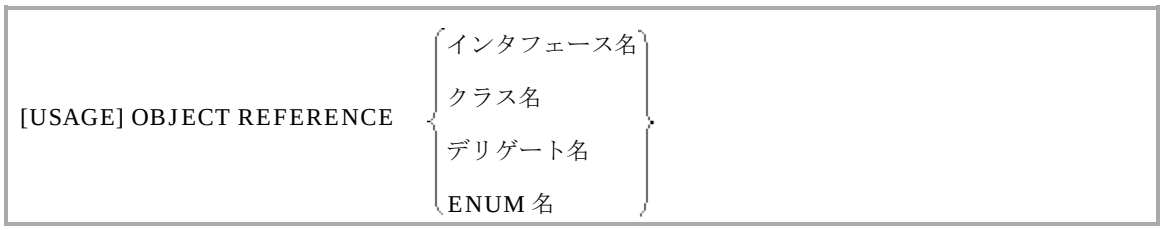


1. オブジェクト参照項目 OBJ-1 は、従業員オブジェクトインスタンスのオブジェクト参照を保持します。
2. OBJ-1 が指すオブジェクトインスタンスのオブジェクトメソッド Set-Employee を呼び出します。INVOKE 文でメソッド名は、文字定数として指定します。
3. Set-Employee メソッドは、USING 指定の従業員データを、そのオブジェクトインスタンスに設定します。

オブジェクト参照

オブジェクト参照を保持する項目を、オブジェクト参照項目といいます。オブジェクト参照項目は、データの取り扱いで型という概念が利用されています。このため、オブジェクト参照項目には型を示す必要があります。

NetCOBOL for .NET での、USAGE OBJECT REFRENCE 句の書き方を以下に示します。



異なるクラス、インタフェース、デリゲートおよび ENUM は、異なった型としてみなします。(インタフェース、デリゲートおよび ENUM は、.NET 固有の要素です。)

また、.NET では、型を指定していないオブジェクト参照は許されていません。従来のオブジェクト指向 COBOL とは異なるため注意してください。詳細については、COBOL 文法書の「11.7.3.9 USAGE 句」を参照してください。

オブジェクト参照の型は、翻訳時または実行時の適合に影響します。 詳細については、適合を参照してください。

NCOBU00204

定義済みオブジェクト一意名

定義済みオブジェクト一意名とは、特殊なオブジェクトを識別するために使用する予約語です。定義済みオブジェクト一意名には、以下があります。

NULL	NULL が設定されているオブジェクト参照は、何のオブジェクトも示していません。
SUPER	メソッドを呼び出すときに使用されます。束縛されるメソッドは、そのメソッド呼出し構文が記述されているクラスのものではなく、親クラスのものとなります。
SELF	メソッドを呼び出すときに使用されます。自分自身のクラスにあるメソッドが束縛されます。
EXCEPTION-OBJECT	例外オブジェクトを参照するときに使用します。

定義済みオブジェクト一意名の詳細については、以下を参照してください。

- 定義済みオブジェクト一意名 SUPER
- 定義済みオブジェクト一意名 SELF
- COBOL 文法書の「11.3.3.4.1 EXCEPTION-OBJECT」
- COBOL 文法書の「11.3.3.4.2 NULL」
- COBOL 文法書の「11.3.3.4.3 SELF と SUPER」

オブジェクト参照の転記

オブジェクト参照項目は、SET 文を用いて他のオブジェクト参照項目に値を代入することができます。たとえば、オブジェクト参照 OBJ-1 を OBJ-X に代入する場合、以下のコードのようになります。

```
DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 OBJ-1 USAGE OBJECT REFERENCE EMPLOYEE.
 01 OBJ-X USAGE OBJECT REFERENCE EMPLOYEE.
 01 従業員データ.
   02 従業員番号  PIC 9(8).
   02 氏名        PIC N(10).
PROCEDURE DIVISION.
    INVOKE EMPLOYEE "NEW" RETURNING OBJ-1
    SET OBJ-X TO OBJ-1
    INVOKE OBJ-X "Set-Employee" USING 従業員データ
```

- オブジェクト参照項目の初期値は NULL です。VALUE 句によって初期値を与えることはできません。
- オブジェクト参照項目の内部形式は、意識しないようにコーディングしてください。NetCOBOL for .NET ランタイムシステムが管理するデータのため、無理に内容を変更した場合、正常に動作できなくなります。
- CALL 文でオブジェクト参照項目を受け渡す場合、USAGE 句が一致していないと正しく受け渡すことができません。

NCOBU00206

メソッド呼出し

メソッドには、スタティックメソッドとオブジェクトメソッドがあります。これらのメソッドは、INVOKE 文で呼び出すことができます。

INVOKE 文の一般的な形式を以下に示します。構文の詳細については、COBOL 文法書の「11.8.3.9 INVOKE 文」を参照してください。

```
INVOKE{オブジェクト一意名 | クラス名}"メソッド名"

[USING {パラメタ}…]

[RETURNING 復帰値]
```

INVOKE 文には、以下の情報が含まれます。

- 呼び出すオブジェクト
- 呼び出すメソッド
- メソッドに渡すパラメタおよびメソッドからの復帰値

INVOKE 文の USING 指定と RETURING 指定については、以下を参照してください。

- INVOKE 文の USING 指定
- INVOKE 文の RETURNING 指定

また、メソッドの呼出しには、INVOKE 文のほかに行内呼出しを使った呼出し方法もあります。

スタティックメソッドの呼出し

スタティックメソッドを含むスタティックメンバーは、NetCOBOL for .NET ランタイムシステムによって、アプリケーションの開始時にメモリ上に生成され、アプリケーションが終了した時にメモリ上から削除されます。

スタティックメソッドを呼び出す場合は、INVOKE 文にクラス名を指定します。NetCOBOL for .NET は、その INVOKE 文を、適切なスタティックメソッドへの参照として翻訳します。たとえば、従業員クラス中のスタティックメソッド"Set-Insurance"を呼び出すには以下のように記述します。

```
INVOKE EMPLOYEE "Set-Insurance" RETURNING OBJ-1
```

オブジェクトメソッドの呼出し

オブジェクトメソッドを呼び出す場合は、INVOKE 文にオブジェクト参照または定義済みオブジェクト一意名(NULL を除く)を指定します。たとえば、従業員クラスのオブジェクトメソッド"Set-Employee"を呼び出すには以下のように記述します。

```
01 OBJ-1 USAGE OBJECT REFERENCE EMPLOYEE.
   INVOKE OBJ-1 "Set-Employee" USING 従業員データ
```

USING 指定

従来の CALL 文の USING 指定と同じように使用します。INVOKE 文の USING 指定に記述したデータを送出し側として、受取り側のデータを、呼び出されるメソッドの連絡節および手続き部見出しに記述することで、データの受渡しが可能になります。

Set-Employee メソッドに従業員データを渡す場合は、以下のように記述します。

【呼出し側】

```
WORKING-STORAGE SECTION.
01 OBJ-1 USAGE OBJECT REFERENCE EMPLOYEE.
01 従業員データ.
  02 従業員番号  PIC 9(8).
  02 氏名        PIC N(10).
    INVOKE OBJ-1 "Set-Employee" USING 従業員データ
```

【呼び出される側】

```
METHOD-ID.  Set-Employee.
DATA DIVISION.
LINKAGE SECTION.
01 従業員データ.
  02 従業員番号  PIC 9(8).
  02 氏名        PIC N(10).
PROCEDURE DIVISION USING 従業員データ.
```

パラメタを渡す時には「BY REFERENCE」、「BY CONTENT」または「BY VALUE」を正しく指定してください。いずれも指定されていない場合は、呼び出されるメソッドの手続き部の見出しの指定に合わせられます。あるいは、パラメタの属性によって決定されます。これらの使い方と詳細については、以下を参照してください。

- COBOL 文法書の「11.8.3.9 INVOKE 文」
- 参照型と値型

INVOKE 文のパラメタに BY VALUE を記述した場合、手続き部の見出しにも BY VALUE が書かれていなければなりません。

NCOBU00208

RETURNING 指定

RETURNING 指定は、呼出し先からの復帰値を受け取るために指定します。また、USING 指定に複数個のパラメタが指定できるのに対し、RETURNING 指定には、1 つだけ指定できます。

【呼出し側】

```
 WORKING-STORAGE SECTION.
 01 給料  PIC 9(8).
    INVOKE OBJ-1 "Compute-Salary" RETURNING 給料
```

【呼び出される側】

```
METHOD-ID. Compute-Salary.
DATA DIVISION.
 LINKAGE SECTION.
 01 給料  PIC 9(8).
PROCEDURE DIVISION RETURNING 給料.
```

RETURNING 指定は、復帰値であるため、呼出し側で設定された値を呼び出される側で参照することはできません。つまり、一方通行の関係となります。呼出し側の処理イメージは、以下のとおりです。

【ソースコード】

```
    INVOKE OBJ-1 "Compute-Salary" RETURNING 給料
```

【実行されたコード】

```
    INVOKE OBJ-1 "Compute-Salary" USING temp
    MOVE temp TO 給料
```

行内呼出し

通常、メソッドの呼出しには INVOKE 文を使用しますが、INVOKE 文を使用しない方法(書き方)があります。これを「メソッドの行内呼出し」と呼びます。

行内呼出しは、メソッドからの復帰値を参照する場合だけ利用できます。復帰値を持たないメソッドに対しては利用することができません。

行内呼出しの形式を以下に示します。構文の詳細については、COBOL 文法書の「11.3.3.2 メソッドの行内呼出し」を参照してください。

```
{オブジェクト一意名}
{                  } :: "メソッド名" [ ( {パラメタ} … ) ]
{クラス名          }
```

Compute-Salary メソッドを呼び出し、受け取った復帰値を"給与欄"に転記したい場合、INVOKE 文で記述すると以下のようになります。

```
INVOKE OBJ-1 "Compute-Salary" RETURNING 給料
MOVE 給料 TO 給与欄
```

上記を行内呼出しで記述すると以下のようになります。

```
MOVE OBJ-1 :: "Compute-Salary" TO 給与欄
```

NCOBU00210

オブジェクトの寿命

オブジェクトを生成する方法は、オブジェクトインスタンスの生成と参照で説明しました。ここでは、そのオブジェクトがいつメモリから削除されるか、または取り除かれるかを説明します。

スタティックメンバーの寿命

スタティックメンバー(スタティックデータやメソッド)は、アプリケーションの寿命と同じです。つまり、実行環境が閉鎖されるまでメモリ上に存在します。また、アプリケーションの動作中に削除することはできません。

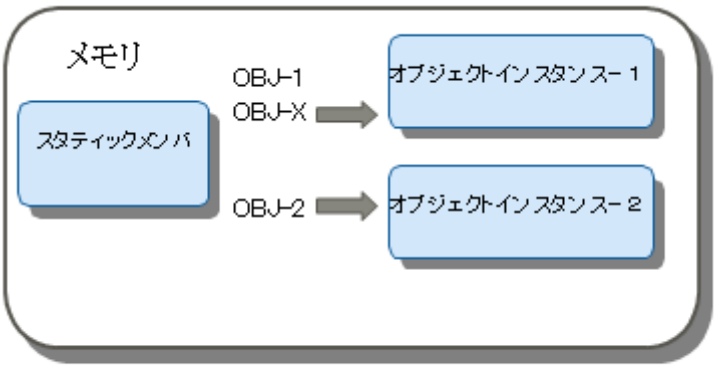
オブジェクトインスタンスの寿命

オブジェクトインスタンスは、どこからも参照されなくなったとき、つまり、そのオブジェクトインスタンスのオブジェクト参照を持つオブジェクト参照項目がなくなったときに削除されます。

詳細については、Visual Studio のドキュメントのガベージ コレクションを参照してください。

以下に、オブジェクト参照が２つのデータ項目に設定される例を示します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 OBJ-1 OBJECT REFERENCE EMPLOYEE.
 01 OBJ-2 OBJECT REFERENCE EMPLOYEE.
 01 OBJ-X OBJECT REFERENCE EMPLOYEE.
PROCEDURE DIVISION.
   INVOKE EMPLOYEE "NEW" RETURNING OBJ-1
   SET OBJ-X TO OBJ-1
   INVOKE EMPLOYEE "NEW" RETURNING OBJ-2
```



オブジェクトインスタンスが不要になったときは、そのオブジェクトインスタンスのオブジェクト参照を持つオブジェクト参照項目に、NULL オブジェクトを代入して初期化します。

```
        SET OBJ-2 TO NULL     *> [1]
        SET OBJ-1 TO NULL     *> [2]
        SET OBJ-X TO NULL     *> [3]
```

1. OBJ-2 はオブジェクトインスタンス-2 のオブジェクト参照を保持しています。[1]の実行により、オブジェクトインスタンス-2 のオブジェクト参照は削除され、これと同時に参照のなくなったオブジェクトインスタンス-2 も削除されます。
2. OBJ-1 と OBJ-X はオブジェクトインスタンス-1 のオブジェクト参照を保持しています。[2]の実行により、OBJ-1 に NULL オブジェクトを代入しても、OBJ-X がオブジェクトインスタンス-1 のオブジェクト参照を持っているため、この時点ではオブジェクトインスタンス-1 はまだ削除されません。
3. [3]の実行により、さらに OBJ-X に NULL オブジェクトを代入すると、オブジェクトインスタンス-1 の

オブジェクト参照はなくなるため、オブジェクトインスタンス-1 は削除されます。

ただし、ここでいう「削除」とは、アプリケーションから論理的に見えなくなるだけであり、メモリ上から解放されるわけではありません。メモリ上からの解放は、.NET システムが最適なタイミングで自動的に行います。これをガベージ コレクションと呼びます。

NCOBU00211

適合

ほとんどのオブジェクト指向プログラミングシステムでは、「適合チェック」が行われます。 これは、より早い段階での問題検出を実現するもので、翻訳時にチェックできるものは翻訳時に、実行時にしかチェックできないものは実行時に行います。NetCOBOL for .NET では、以下の項目ついてチェックしています。

- オブジェクト参照項目が扱うことができるオブジェクトインスタンスか？
- クラス中に存在しないメソッドを呼び出そうとしていないか？
- OVERRIDE 句が書かれたメソッドの引数の型が親クラスのメソッドの引数の型と一致しているか？
- メソッドに渡すパラメタが異なっていないか？オーバロードされているメソッドを呼び出す場合、そのパラメタが正しいか？

オブジェクト指向では、メソッドを呼び出す場合、呼出し時のインタフェースが正しいかどうかチェックします。ここでは、これらのチェックがいつ、どのように実行されるのかについて説明します。

- オブジェクト指向プログラミングの概念の適合も参考にしてください。
- オーバロードとは、1 つのクラスの中で、名前が同じで内容が異なるメソッドを複数定義できる機能をいいます。これらの複数の同名メソッドは引数の型や個数で区別されます。

このセクションの内容

クラスの適合

クラスの適合について説明します。

オブジェクト参照項目と適合チェック

オブジェクト参照項目へ代入する場合の適合チェックの働きについて説明します。

メソッド定義時の適合チェック

メソッド定義時の適合チェックについて説明します。

メソッド呼出し時の適合チェック

メソッド呼出し時の適合チェックについて説明します。

オブジェクト指定子

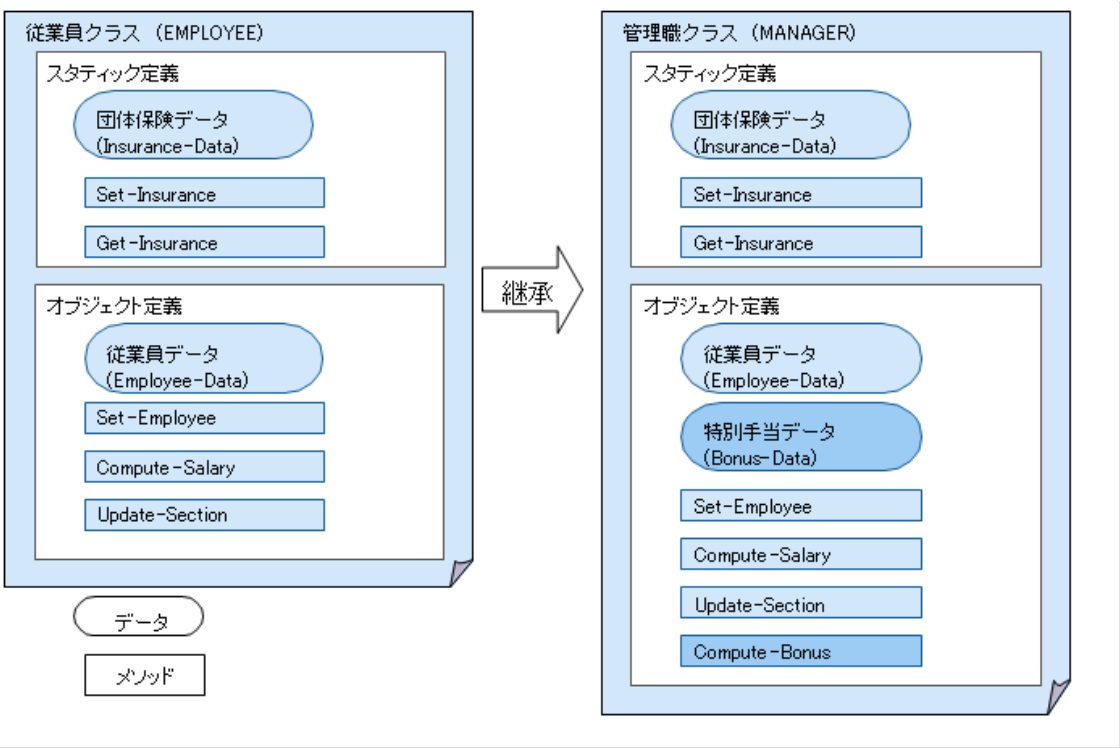
オブジェクト指定子の使い方について説明します。

翻訳時と実行時の適合チェック

翻訳時と実行時の適合チェックについて説明します。

クラスの適合

クラス A のインタフェースを完全に含むクラス B があった時、「B は A に適合する」と表現します。この時のインタフェースとは、クラスで定義されたメソッド(暗黙定義も含む)とそのメソッドのパラメタを指します。つまり、継承で親子関係にあるクラスにおいて適合関係は成立します。



上の図の場合、管理職クラスは従業員クラスの持つ全機能を包含しています。このとき、「管理職クラスは従業員クラスに適合している」と表現します。

逆に、従業員クラスは、管理職クラスの持つ全機能を包含していません。したがって、「従業員クラスは、管理職クラスに適合していない」ことになります。

つまり、適合の関係は相互に成立するものではなく、単一方向に成立するものといえます。

NCOBU00213

オブジェクト参照項目と適合チェック

オブジェクト参照項目へ代入する場合の適合チェックの働きを以下に示します。オブジェクト参照項目については、オブジェクト参照を参照してください。

管理職クラス(MANAGER)は従業員クラス(EMPLOYEE)を継承しています。

```
WORKING-STORAGE SECTION.
 01 OBJ-1 OBJECT REFERENCE EMPLOYEE.
 01 OBJ-2 OBJECT REFERENCE EMPLOYEE.
 01 OBJ-X OBJECT REFERENCE MANAGER.
 01 OBJ-Y OBJECT REFERENCE MANAGER.
PROCEDURE DIVISION.
   SET OBJ-1 TO OBJ-X    *> [1]
   SET OBJ-Y TO OBJ-2    *> [2]
   SET OBJ-1 TO OBJ-2    *> [3]
   SET OBJ-2 TO OBJ-1    *> [4]
```

上記のコードを翻訳した結果を以下に示します。

[1]:エラーなし

管理職クラスは従業員クラスに適合するので、管理職クラスのオブジェクト参照を従業員クラス型のオブジェクト参照に代入することができます。管理職クラスが従業員クラスに適合するということは、従業員クラスのすべてのメソッドを管理職クラスのオブジェクト参照で呼び出せるということです。つまり、以下の書き方が可能です。

```
INVOKE OBJ-1 "従業員クラスのメソッド"
```

これは、OBJ-1 が管理職クラスのオブジェクト参照を保持するため、正しく動作します。

[2]:翻訳エラー

管理職クラスのオブジェクト参照である OBJ-Y に、従業員クラスのオブジェクト参照である OBJ-2 を代入するこはできません。これは、管理職クラスのメソッドは、管理職クラスのオブジェクト参照からしか呼び出せないことを意味します。

```
INVOKE OBJ-Y "管理職クラスのメソッド"
```

また、OBJ-Y が従業員クラスのオブジェクト参照を保持する場合は、管理職クラスのメソッドを利用できません。

[3]、[4]:エラーなし

オブジェクト参照の型が一致するので適合エラーになりません。

メソッド定義時の適合チェック

以下の例は、子クラスで新規に定義したメソッドと、親クラスから継承したメソッドをオーバライドしたメソッドの例です。

【親クラス(従業員クラス:EMPLOYEE)】

```
METHOD-ID. Compute-Salary.
DATA DIVISION.
LINKAGE SECTION.
01 給与 PIC S9(9) COMP-5.
PROCEDURE DIVISION RETURNING 給与.
```

【翻訳するクラス(管理職クラス:MANAGER)】

```
METHOD-ID. Compute-Salary.   *>[1]
DATA DIVISION.
LINKAGE SECTION.
01 給与 PIC S9(9) COMP-5.
01 月   PIC S9(4) COMP-5.
PROCEDURE DIVISION USING BY REFERENCE 月
                RETURNING 給与.
```

```
METHOD-ID. Compute-Salary OVERRIDE.   *>[2]
DATA DIVISION.
LINKAGE SECTION.
01 給与 PIC S9(9) COMP-5.
PROCEDURE DIVISION RETURNING 給与.
```

上記のメソッドは、両方とも同じ名前のメソッドとして適合チェックされます。 チェックは、翻訳するクラスから親クラスまでの関係をさかのぼって行われます。この適合チェックは、次の説明するメソッド呼出しの適合チェックよりも厳しく行われます。メソッドの引数の型や数が正確に一致しない限り、適合しません。

メソッド名と引数の型が一致するか、そのメソッドがどのクラスに属するかによって結果が異なります。

[1]:エラーなし

このメソッドは親クラスのメソッドと適合していません。親クラスのメソッドと引数の型と数が異なるため、メソッドのオーバロードと見なされます。この場合、エラーにはなりません。ただし、親クラスに同名で引数の型と数が一致するメソッドがある場合、このメソッド定義はエラーになります。

[2]:エラーなし

このメソッドは親クラスのメソッドと適合しています。OVERRIDE 指定で親クラスのメソッドをオーバライドしているため、正しく処理されます。ただし、OVERRIDE 指定がある場合、親クラスに同名で引数の型と数が一致するメソッドがない時はエラーになります。

NCOBU00215

メソッド呼出し時の適合チェック

ここでは、メソッドを呼び出すときに適合チェックがどのように働いているかについて説明します。

従業員クラスに以下のような Compute-Salary メソッドが 2 つ含まれているとします。

【従業員クラス】

```
METHOD-ID. Compute-Salary. *>[a]
DATA DIVISION.
LINKAGE SECTION.
01 給与 PIC S9(9) COMP-5.
PROCEDURE DIVISION RETURNING 給与.
*
METHOD-ID. Compute-Salary. *>[b]
DATA DIVISION.
LINKAGE SECTION.
01 給与 PIC S9(9) COMP-5.
01 月   PIC S9(4) COMP-5.
PROCEDURE DIVISION USING BY REFERENCE 月
                RETURNING 給与.
```

従業員オブジェクトのメソッドを呼び出す、以下のコードを翻訳します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 OBJ-1 OBJECT REFERENCE EMPLOYEE.
 01 給与  PIC S9(9) COMP-5.
 01 月    PIC S9(4).
PROCEDURE DIVISION.
   INVOKE OBJ-1 "Compute-Salary" RETURNING 給与.       *>[1]
   INVOKE OBJ-1 "Compute-Salary" USING BY REFERENCE 月
                            RETURNING 給与.       *>[2]
```

OBJ-1 は、従業員クラスまたは従業員クラスを継承する子クラスのオブジェクト参照を格納することができます。コンパイラは、Compute-Salary メソッドが従業員クラスに存在していて、INVOKE 文に指定されたパラメタが正しいかどうかをチェックします。この時、従業員クラスの情報を含むアセンブリファイルを使用して適合チェックをします。

USING パラメタは左から順にチェックされます。すべての USING パラメタのチェックが終了したら、RETURNING パラメタがチェックされます。

翻訳した結果は、以下のようになります。

[1]: エラーなし

[a]の Compute-Salary メソッドが呼び出されます。

[2]: 翻訳エラー

[b]の Compute-Salary メソッドとパラメタの型が一致しません。

- [b]の USING パラメタの宣言 ：PIC S9(4) COMP-5.
- [2]の USING パラメタの宣言 ：PIC S9(4).

このため、翻訳エラーが発生します。

USING パラメタの適合規則は、「BY VALUE」、「BY CONTENT」または「BY REFERENCE」のどれを渡すかによっても異なります。一般に、「BY REFERENCE」で渡されるパラメタは、厳しくチェックが行われます。そ

れに比べ、「BY VALUE」もしくは「BY CONTENT」で渡される値は、「BY REFERENCE」のようにチェックが厳しくありません。したがって、オーバロードされた複数のメソッドが INVOKE 文のパラメタに一致するという現象が起こります。

より適切な適合

以下の例では、.NET Framework のクラスライブラリに含まれる Max メソッドを呼び出しています。Max メソッドは、２つの引数をもち、「Max(int32,int32)」、「Max(single,single)」などの引数の型の異なるメソッドが多数存在します。

```
REPOSITORY.
   CLASS MATH AS "System.Math".
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 int_param1    PIC S9(9) COMP-5.  *> int32
 01 int_param2    PIC S9(9) COMP-5.  *> int32
 01 single_param1 COMP-1.           *> single
 01 cobnum_param1 PIC S9(18).
 01 cobnum_param2 PIC S9(18).
 01 ans1         PIC S9(9) COMP-5.
 01 ans2         COMP-1.
 01 ans3         COMP-2.
PROCEDURE DIVISION.
   INVOKE MATH "Max" USING BY VALUE int_param1     *>[1]
                           BY VALUE int_param2
                     RETURNING     ans1.
   INVOKE MATH "Max" USING BY VALUE int_param1     *>[2]
                           BY VALUE single_param1
                     RETURNING     ans2.
   INVOKE MATH "Max" USING BY VALUE single_param1   *>[3]
                           BY VALUE cobnum_param1
                     RETURNING     ans2.
   INVOKE MATH "Max" USING BY VALUE cobnum_param1   *>[4]
                           BY VALUE cobnum_param2
                     RETURNING     ans3.
```

[1]: エラーなし

「Max(int32, int32)」メソッドの呼出しとして正確に指定されているため、エラーになりません。

[2],[3]: エラーなし

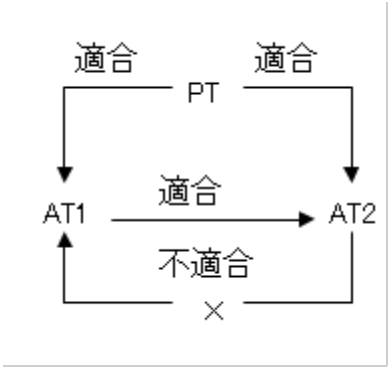
この 2 つ INVOKE 文は、「Max(single, single)」と「Max(double, double)」の両方のメソッドを呼び出せます。しかし、パラメタ single_param1 に対して「Max(single, single)」は「Max(double, double)」よりも、より適切な適合を持つため、「Max(single, single)」が呼び出されます。

[4]: 翻訳エラー

この INVOKE 文では「Max(single, single)」、「Max(double, double)」および「Max(decimal, decimal)」の 3 つを呼び出すことができます。「Max(single, single)」は「Max(double, double)より適切な適合を持ちますが、「Max(single, single)」と「Max(decimal, decimal)」の間ではどちらも他方より適切な適合を持っていません。このため、コンパイラがどちらの Max メソッドを呼び出すか決定できずエラーが発生します。

メソッドの引数と INVOKE 文に指定されるパラメタの『より適切な適合』は次のように決められます。

1. パラメタの型 PT と一致する引数の型 AT1 があるなら、AT1 は他の引数の型より PT により適切な適合を持ちます。
2. パラメタの型 PT と異なる型であるが、適合する引数の型 AT1 と AT2 があるとき、AT1 は AT2 に適合するが、AT2 は AT1 に適合しないとします。このとき、AT1 は AT2 より PT により適切に適合します。(下図参照)



3. これ以外の場合は AT1 と AT2 のどちらも、PT に対してより適切な適合ではありません。

また、次のような場合、コンパイラは呼び出すメソッドを決定することができません。

- オーバロードされたメソッドの中で他のメソッドよりも適切な適合を持つメソッドが存在しない。
- オーバロードされたメソッドの中で、引数の順次位置ごとに他のメソッドよりも適切な適合を持つメソッドが異なる。

上記の 2 つの場合は、引数を正しい型で渡すことで翻訳エラーを避けることができます。型をあわせるには、一時変数を使用します。例えば[4]の場合、パラメタを double 型に一度転記してからパラメタに指定すると「MAX(double,double)」が呼び出せます。

```
      01 work_double_val1 COMP-2.
      01 work_double_val2 COMP-2.
    *
          MOVE cobnum_param1 TO work_double_val1
          MOVE cobnum_param2 TO work_double_val2
          INVOKE MATH "Max" USING BY VALUE work_double_val1
                                  BY VALUE work_double_val2
                        RETURNING     ans3.
```

オブジェクト指定子

オブジェクト参照項目に対してコンパイラが行う適合チェックをゆるめ、翻訳エラーが出ないようにすることができます。COBOL では、オブジェクト指定子を利用してこれを実現しています。

以下のコードで、管理職クラスは従業員クラスを継承しているものとします。

```
WORKING-STORAGE SECTION.
 01 OBJ-1 OBJECT REFERENCE EMPLOYEE.
 01 OBJ-2 OBJECT REFERENCE MANAGER.
PROCEDURE DIVISION.
   INVOKE MANAGER "NEW" RETURNING OBJ-2.
   SET OBJ-1 TO OBJ-2.  *>[1]
   SET OBJ-2 TO OBJ-1.  *>[2]
```

管理職クラスは従業員クラスに適合していますが、従業員クラスは管理職クラスに適合していないため、[2]で翻訳エラーが発生します。

しかし、OBJ-1 が従業員クラスのオブジェクト参照ではなく、管理職クラスのオブジェクト参照を保持するような場合、翻訳エラーを出さずにこの代入を行う必要があります。このような場合、OBJ-1 にオブジェクト指定子を指定します。

```
   SET OBJ-1 TO OBJ-2.
   SET OBJ-2 TO OBJ-1 AS MANAGER.
```

オブジェクト指定子は、コンパイラが OBJ-1 を管理職クラスのオブジェクト参照とみなすように指示しています。コンパイラは、オブジェクト指定子で指定された型をもとに適合チェックを行ないます。また、システムは実行時に、OBJ-1 が保持するオブジェクト参照が管理職クラスに適合するかをチェックします。

オブジェクト指定子について、COBOL 文法書の「11.3.3.3 オブジェクト指定子」を参照してください。他の言語（Visual Basic や Visual C#など）では、キャストによってオブジェクトの型の変更を行います。

NCOBU00217

翻訳時と実行時の適合チェック

コンパイラは、どのオブジェクトのメソッドが実行されているかわからないような場合は、適合チェックをすることができません。システムが適合チェックを正しく行えるように、オブジェクト参照項目に指定した格納可能なオブジェクト参照を正しく設定してください。もし、正しく設定していなければ、実行時エラーが発生します。

メソッドの束縛

概要

メソッドを呼び出す場合、呼び出すメソッドは、以下の２つの情報によって決定されます。

- どのオブジェクトのメソッドなのか？
- 何という名前のメソッドなのか？

この「呼び出すメソッドを決定する」ことを、オブジェクト指向では、「メソッドの束縛」と呼びます。

束縛の過程は、オブジェクト指向において、振る舞いや性能に作用する大変重要な要素です。束縛には以下の 2 種類があります。

- 静的束縛
- 動的束縛

静的束縛は、メソッドの決定が翻訳時に行われ、動的束縛は、オブジェクトとメソッドの決定が実行時に行われます。動的束縛は、従来のオブジェクト指向 COBOL でも使用されてきたものです。 静的束縛は、NetCOBOL for .NET では、以下のようなメソッド呼出しの時に使われます。

- スタティックメソッド呼出し
- NEW メソッド呼出し(コンストラクタ呼出し)

COBOL 言語以外で定義されたメソッドの束縛は、その言語で定義されたメソッドによって決まります。

Visual C++などのオブジェクト指向言語では、「動的束縛」は「レイトバインディング」といわれています。

このセクションの内容

動的束縛と多態

動的束縛と多態について説明します。

定義済みオブジェクト一意名 SELF

定義済みオブジェクト一意名 SELF の使い方について説明します。

定義済みオブジェクト一意名 SUPER

定義済みオブジェクト一意名 SUPER の使い方について説明します。

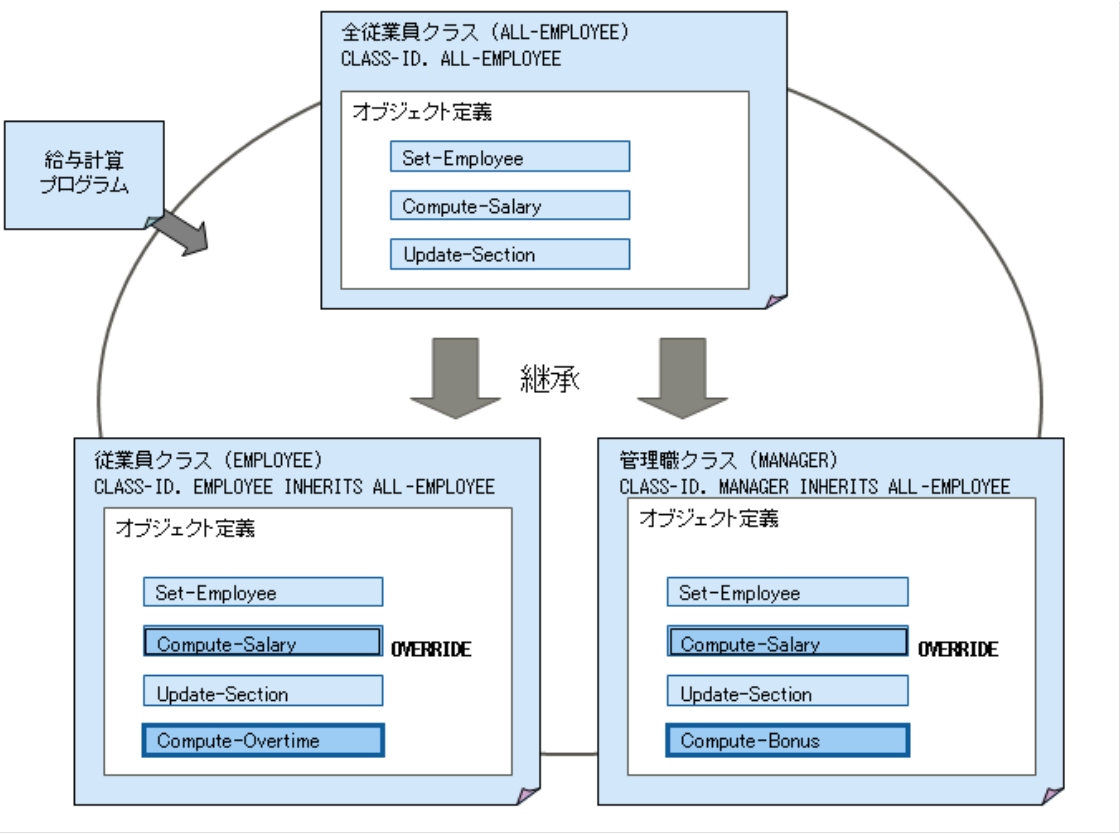
NCOBU00219

動的束縛と多態

動的束縛

動的束縛は、メソッドを呼び出すオブジェクトと呼び出されるメソッドが、実行時まで決定することができない時に使われます。

ここでは、以下の構成のクラスをアクセスする給与計算プログラムを例にして、説明します。



以下のコードは動的束縛が必要になります。

```
PROGRAM-ID. MAIN.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS EMPLOYEE
   CLASS MANAGER
   CLASS ALL-EMPLOYEE
   .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 OBJ-1          OBJECT REFERENCE ALL-EMPLOYEE.

PROCEDURE DIVISION.
   INVOKE OBJ-1 "Compute-Salary" RETURNING 給与. *>[1]
```

[1]: OBJ-1 は、全従業員クラスに適合するクラスのオブジェクト参照が格納されます。つまり、全従業員クラス、従業員クラスおよび管理職クラスのどのオブジェクト参照も格納できます。この場合、どのクラスのオブジェクトを呼び出すかが実行時までわからないため、動的束縛になります。

```
CLASS-ID. ALL-EMPLOYEE.
OBJECT.
PROCEDURE DIVISION.
METHOD-ID. Set-Employee.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 給与 PIC 9(8).
PROCEDURE DIVISION.
   INVOKE SELF  "Compute-Salary" RETURNING 給与. *>[2]
```

[2]: SELF は、実行中のオブジェクトを表しています。実行されるメソッドは、そのメソッドが定義されているクラスのオブジェクトが実行されているか、メソッドが定義されているクラスを継承しているクラスのオブジェクトが実行されているかで異なります。この例の場合、従業員クラスのオブジェクトが実行されていれば、SELF によって従業員クラスで OVERRIDE して定義された Compute-Salary メソッドが実行されますし、管理職クラスのオブジェクトが実行されていれば、管理職クラスで定義された Compute-Salary メソッドが実行されます。このため、この例も動的束縛になります。

多態

以下のように記述すると、ALL-EMPLOYEE-OBJ に格納されているオブジェクトが、従業員クラスのオブジェクトか、管理職クラスのオブジェクトかを意識せずに、給料計算処理を行うことができます。

このように、１つのオブジェクト参照項目によって、複数の異なるオブジェクトを操作することを、「多態」といいます。多態の概念については、多態を参照してください。

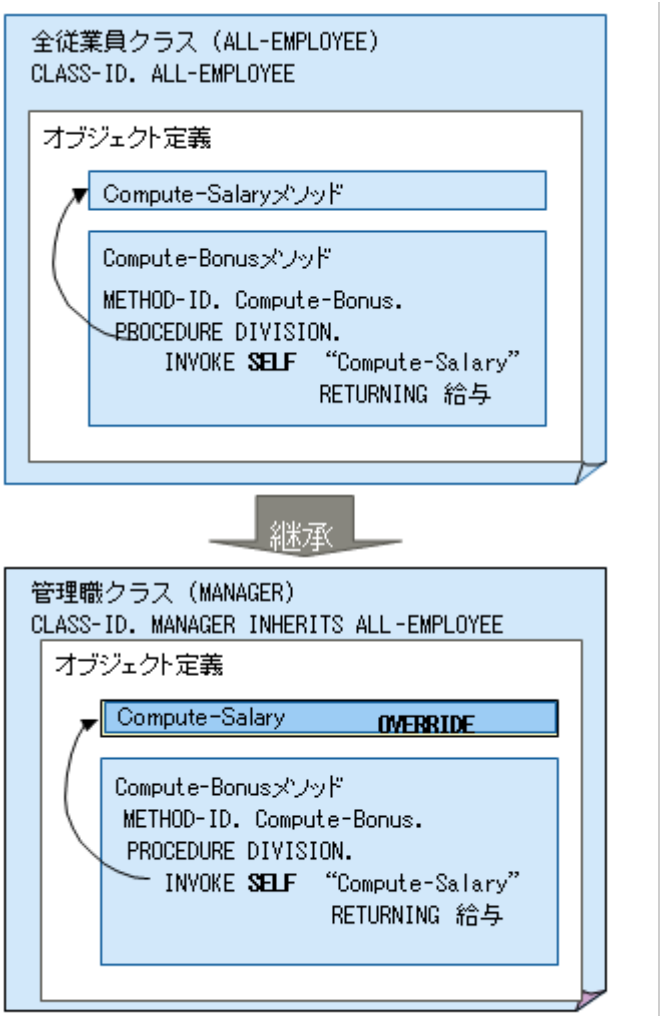
```
PROGRAM-ID. MAIN.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS EMPLOYEE
   CLASS MANAGER
   CLASS ALL-EMPLOYEE
   .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 OBJ-1          OBJECT REFERENCE ALL-EMPLOYEE.
01 EMPLOYEE-OBJ    OBJECT REFERENCE EMPLOYEE.
01 MANAGER-OBJ     OBJECT REFERENCE MANAGER.
01 ALL-EMPLOYEE-OBJ OBJECT REFERENCE ALL-EMPLOYEE.
01 給与 PIC 9(8).
PROCEDURE DIVISION.
   SET ALL-EMPLOYEE-OBJ TO EMPLOYEE-OBJ
   PERFORM 給与計算.
*   :
   SET ALL-EMPLOYEE-OBJ TO MANAGER-OBJ
   PERFORM 給与計算.
*   :
 給与計算 SECTION.
   INVOKE ALL-EMPLOYEE-OBJ "Compute-Salary" RETURNING 給与.
```

NCOBU00220

定義済みオブジェクト一意名 SELF

定義済みオブジェクト一意名 SELF は、現在実行中のオブジェクト参照を表すために使用します。

たとえば、動的束縛と多態と同じクラスモデルで、全従業員クラスの中に賞与を計算するメソッド(Compute-Bonus メソッド)があったとします。Compute-Bonus メソッドで給料を求める処理が必要だった場合、定義済みオブジェクト一意名 SELF を使用して以下のように記述することができます。

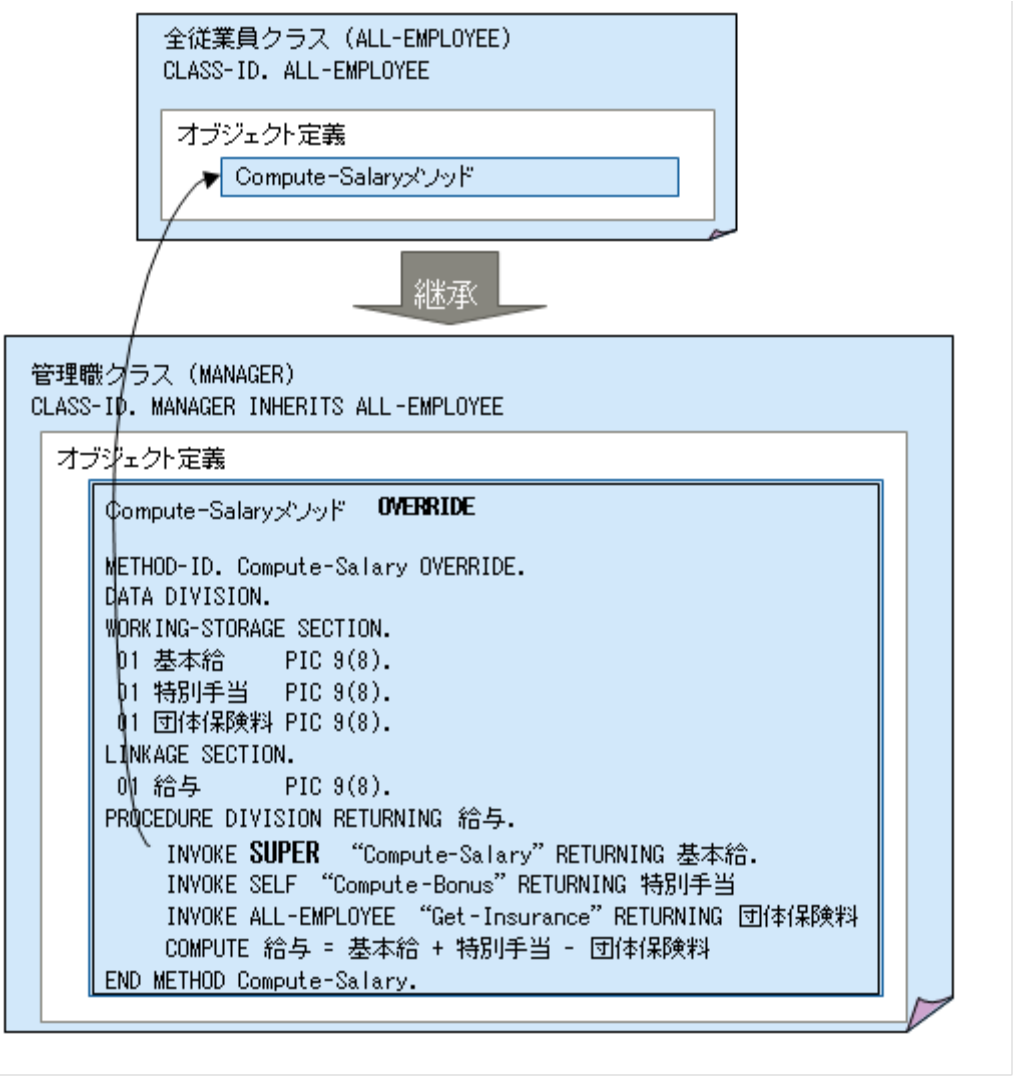


これは、全従業員クラスのオブジェクトの実行中なら、全従業員クラスの中で定義された Compute-Salary メソッドが実行され、管理職クラスのオブジェクトの実行中なら、管理職クラスの中で OVERRIDE して定義された Compute-Salary メソッドが実行されることを表しています。

定義済みオブジェクト一意名 SUPER

定義済みオブジェクト一意名 SUPER は、親クラスを表すために使用します。

たとえば、動的束縛と多態と同じクラスモデルでは、管理職クラスの Compute-Salary メソッドは、特別手当の加算処理があるため、親クラスの Compute-Salary メソッドを上書きして定義しています。しかし、特別手当て以外の処理は、親クラスで定義した処理と同じだったとします。このような場合に、上書きしたメソッドから親クラスで定義しているメソッドを呼び出すことができます。



管理職クラスの Compute-Salary メソッドでは、手続きの最初に親クラス(全従業員クラス)の Compute-Salary メソッドを実行します。

NCOBU00222

プロパティ

プロパティを利用すると、オブジェクトデータやスタティックデータを簡単に設定・参照することができます。ここでは、プロパティの使用方法について説明します。

このセクションの内容

PROPERTY 句

データ項目に PROPERTY 句を指定して、プロパティとして利用する方法について説明します。

プロパティのアクセス

オブジェクトプロパティを利用してプロパティにアクセスする方法について説明します。

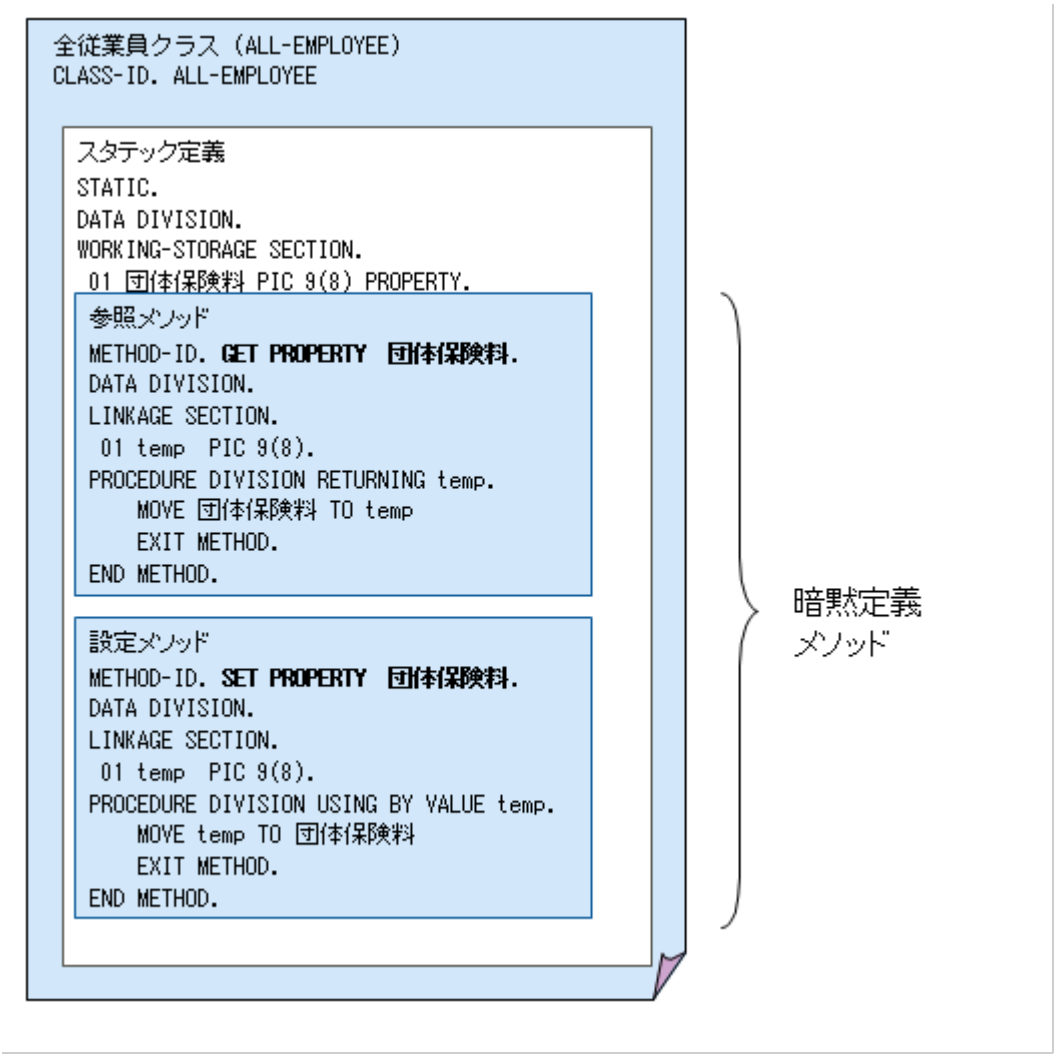
明示的な GET/SET プロパティメソッドの定義

プロパティメソッドに特別な処理を持たせたい場合には、利用者がプロパティメソッドを明示的に定義することができます。プロパティメソッドを明示的に記述する方法について説明します。

PROPERTY 句

オブジェクデータまたはスタティックデータに PROPERTY 句を指定して、そのデータ項目をプロパティとして利用することができます。PROPERTY 句を指定すると、これと同じ名前を持つプロパティメソッドが自動的に生成されます。生成されるメソッドは、PROERTY 句が指定されたデータ項目に対して、 値を参照するメソッド(GET PROPERTY メソッド)と値を設定するメソッド(SET PROPERTY メソッド)です。

例えば、全従業員クラスのスタティック"団体保険料"に PROPERTY 句を指定すると、上で述べた 2 つのメソッドが自動生成されます。自動生成されたメソッドを、暗黙定義メソッドといいます。暗黙定義メソッドとは、ソースプログラムには記述されていないが、論理的に存在するメソッドのことをいいます。



PROPERTY 句の規則

- PROPERTY 句は、スタティック定義またはオブジェクト定義の作業場所節だけに書けます。
- PROPERTY 句は、レベル番号 01 または 77 の基本項目だけに指定できます。

その他の規則については、COBOL 文法書の「11.7.3.7 PROPERTY 句」を参照してください。

NCOBU00224

プロパティのアクセス

プロパティメソッドを呼び出す時は、オブジェクトプロパティと呼ばれる一意参照を利用します。INVOKE 文を利用して呼び出すことはできません。

オブジェクトプロパティの構文を以下に示します。

以下は、従業員クラスと管理職クラスが全従業員クラスを継承しているクラスモデルにおいて、全従業員クラスのスタティック定義に宣言された"団体保険料"に PROPERTY 句を指定した場合の、データの参照、設定方法について示しています。

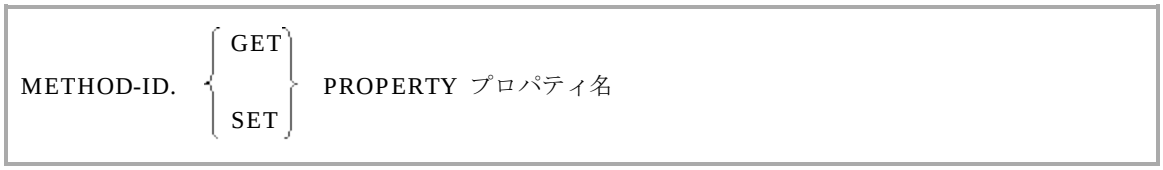
```
       PROGRAM-ID. MAIN.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       REPOSITORY.
           CLASS EMPLOYEE
           CLASS MANAGER
           CLASS ALL-EMPLOYEE
           .
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 保険料  PIC 9(8).
       PROCEDURE DIVISION.
      *-------------------------------------
      * プロパティデータの設定
      *-------------------------------------
           MOVE 保険料 TO 団体保険料 OF EMPLOYEE
           MOVE 保険料 TO 団体保険料 OF MANAGER
      *-------------------------------------
      * プロパティデータの参照
      *-------------------------------------
           MOVE 団体保険料 OF EMPLOYEE TO 保険料
           MOVE 団体保険料 OF MANAGER  TO 保険料
```

参照メソッドを呼び出すか、設定メソッドを呼び出すかは、オブジェクトプロパティが送出し側に指定されたか、受取り側に指定されたかによって決まります。

また、上の例ではオブジェクトプロパティにクラス名を指定していますが、これはプロパティデータがスタティック定義で宣言されているためです。プロパティデータがオブジェクト定義で宣言されている場合は、オブジェクト一意名を指定します。

明示的な GET/SET プロパティメソッドの定義

プロパティメソッドに特別な処理を持たせたい場合には、利用者がプロパティメソッドを明示的に定義することができます。プロパティメソッドを明示的に記述する場合は、以下のように記述します。



以下は、スタティックデータである"団体保険料"が更新されると簡単なメッセージを出力するものです。

```
CLASS-ID. ALL-EMPLOYEE.
STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 保険データ.
  02 団体保険料 PIC 9(8).
PROCEDURE DIVISION.
*-----------------------------------------------
*  利用者定義のプロパティメソッド(参照メソッド)
*-----------------------------------------------
METHOD-ID. GET PROPERTY 団体保険料.
DATA DIVISION.
LINKAGE SECTION.
01 保険額 PIC 9(8).
PROCEDURE DIVISION RETURNING 保険額.
    MOVE 団体保険料 TO 保険額
    EXIT METHOD.
END METHOD.
*-----------------------------------------------
*  利用者定義のプロパティメソッド(設定メソッド)
*-----------------------------------------------
METHOD-ID. SET PROPERTY 団体保険料.
DATA DIVISION.
LINKAGE SECTION.
01 保険額 PIC 9(8).
PROCEDURE DIVISION USING BY VALUE 保険額.
    MOVE 保険額 TO 団体保険料
    DISPLAY "団体保険料が更新されました"  *> [1]
    EXIT METHOD.
END METHOD.
```

設定メソッドが書かれていることに注意してください。プロパティ名と同名のデータ名(団体保険料)に PROPERTY 句は指定できないため、設定および参照の両方のメソッドが必要な場合は、両方のメソッドを明示的に定義しなければなりません。

明示的なプロパティメソッドの定義の規則

- プロパティメソッドを明示的に定義した場合は、データの宣言で PROPERTY 句指定をしてはいけません。
- プロパティ名は、スタティック定義またはオブジェクト定義の作業場所節の 01 または 77 レベル項目として定義したデータ名と同じ名前であってはなりません。(上の例で、"団体保険料"を 02 レベルの項目として宣言しているのは、この規則のためです。)

その他の規則については、COBOL 文法書の「11.5.1.5 メソッド名段落(METHOD-ID)」の構文規則を参照してください。

NCOBU00226

ACCEPT 文および DISPLAY 文の使い方

ここでは、ACCEPT 文および DISPLAY 文を使った機能について説明します。

このセクションの内容

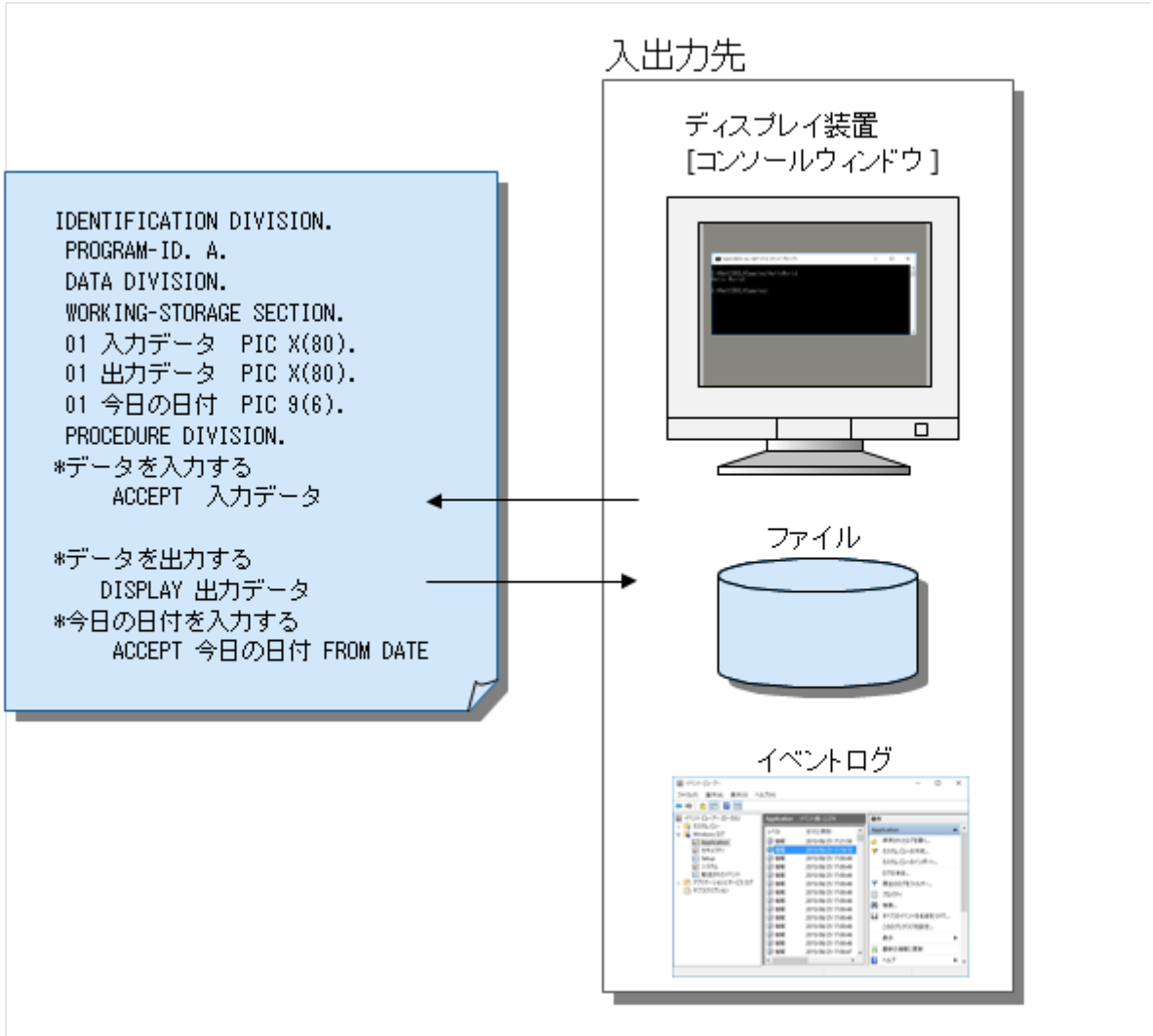
小入出力機能

ここでは、ACCEPT 文および DISPLAY 文を使ってデータの入出力を行う、小入出力機能について説明します。なお、小入出力機能を使用した例題プログラムをサンプルプログラムとして提供しています。NetCOBOL for .NET Visual Studio プロジェクトサンプルアプリケーションの"3. Alphabet Table Lookup"を参考にしてください。

概要

小入出力機能では、コンソールウィンドウ、イベントログ、およびファイルを使って、データの入出力を簡単に行うことができます。また、システムから現在の日付や時刻を読み込むこともできます。

小入出力機能を使ったデータの入力には ACCEPT 文を、データの出力には DISPLAY 文を使います。



入出力先の種類と指定方法

小入出力機能を使ってデータの入出力を行うときの入出力先は、ACCEPT 文の FROM 指定および DISPLAY 文の UPON 指定の記述や、翻訳オプションの指定、環境変数情報の設定内容によって異なります。

これらの指定と入出力先の関係を以下に示します。

小入出力機能の入出力先

FROM 指定または UPON 指定の記述	指定する翻訳オプション	環境変数情報の設定内容	入出力先
記述なし、または、機能名 SYSIN/SYSOUT に対応付けた呼び名	なし	－	ディスプレイ装置(コンソールウィンドウ)[注 2]
	SSIN(環境変数情報名) SSOUT(環境変数情報名)	環境変数情報名にファイル名を設定	ファイル[注 1]
	－	@CBR_DISPLAY_SYSOUT_OUTPUT に EVENTLOG を設定	イベントログ[注 4][注 5]
機能名 SYSERR に対応付けた呼び名	－	－	ディスプレイ装置(コマンドプロンプトまたはシステムのコンソール)[注 3]
		@MessOutFile にファイル名を設定	ファイル
		@CBR_DISPLAY_SYSERR_OUTPUT に EVENTLOG を設定	イベントログ[注 5]
機能名 CONSOLE に対応付けた呼び名	－		ディスプレイ装置(コンソールウィンドウ)[注 2]
		@CBR_DISPLAY_CONSOLE_OUTPUT に EVENTLOG を設定	イベントログ[注 4][注 5]

[注 1]

翻訳オプション SSIN/SSOUT に、環境変数情報名 SYSIN/SYSOUT を指定した場合、ACCEPT 文/DISPLAY 文の入出力先はコンソールウィンドウになります。

[注 2]

コンソールウィンドウには、以下があります。

- コマンドプロンプト

  Windows のコマンドプロンプト からアプリケーションを起動した場合、 起動したウィンドウをコンソールウィンドウとして使用することができます。 また、この場合、リダイレクションを使用することができます。

  ACCEPT 文は標準入力、機能名 SYSOUT および CONSOLE に対応付けた呼び名を指定した DISPLAY 文は標準出力、 機能名 SYSERR に対応付けた呼び名を指定した DISPLAY 文は標準エラーに、 それぞれの入出力先が対応付けられます。

コマンドプロンプトの終了ボタンをクリックすると、ウィンドウが閉じるのと同時に、 オープンされているファイルの CLOSE 処理などの資産の回収処理が行われない状態で COBOL アプリケーションが強制的に終了します。

- システムのコンソールウィンドウ

  システムのコンソールウィンドウは、 アプリケーションの起動と同時または実行中に生成される、コマンドプロンプト と同様なデザインのウィンドウのことです。

COBOL アプリケーションの動作中に、 システムのコンソールウィンドウの終了ボタンをクリックすると、 ウィンドウが閉じるのと同時に、COBOL アプリケーションが強制的に終了されます。 この場合、オープンされているファイルの CLOSE 処理などの資産の回収処理が行われません。

これらのウィンドウは、プログラムの実行開始から終了までを通して、複数同時に使うことはできません。

[注 3]

コマンドプロンプトまたはシステムのコンソールウィンドウの標準エラーに出力します。

[注 4]

イベントログはデータの入力はできません。

[注 5]

イベントログに出力する場合、他の出力先の指定は無効になり出力されません。

NCOBU00228

コンソールウィンドウを使ったデータの入出力

コンソールウィンドウとは、小入出力機能によってディスプレイ装置からデータを読み込んだり、データを表示したりするときに使われるウィンドウです。コンソールウィンドウは、COBOL プログラムの 1 つの実行単位について 1 つです。

ここでは、コンソールウィンドウを使うプログラムの記述方法のうち、最も簡単な記述方法について、プログラムの書き方、ビルドおよび実行方法について説明します。

プログラムの記述

ここでは、プログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
 01  データ名  PIC X(80).
PROCEDURE DIVISION.
   ACCEPT   データ名.
   DISPLAY  データ名.
   DISPLAY   "文字定数".
   EXIT PROGRAM.
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

とくに必要な記述はありません。

データ部(DATA DIVISION)

入力したデータを格納するためのデータ項目および出力するデータを設定するためのデータ項目を定義します。これらのデータ項目は、次のいずれかの属性で定義します。

集団項目/英字項目/英数字項目/2 進項目/内部 10 進項目/外部 10 進項目/ 英数字編集項目/数字編集項目/日本語項目[注]/日本語編集項目[注]

注：ACCEPT 文では使用できません。

手続き部(PROCEDURE DIVISION)

コンソールウィンドウからデータを入力するには、ACCEPT 文を使います。入力データは、ACCEPT 文に指定したデータ名に、そのデータ名に定義した長さ(たとえば、01 入力データ PIC X(80).と定義した場合、英数字が 80 文字)の分だけ格納されます。なお、入力データの長さが格納するデータの長さより短い場合、文字のときには長さ分のデータが入力されるまで、数字のときには実行キーまたはリターンキーが入力されるまで入力要求が行われます。

Micro Focus COBOL 仕様による ACCEPT 文動作の指定については、@CBR_ACCEPT_COMPATIBILITY を参照してください。

コンソールウィンドウにデータを出力するには、DISPLAY 文を使います。DISPLAY 文にデータ名を指定した場合には、データ名に格納されているデータが出力されます。また、DISPLAY 文に文字定数を指定した場合には、指定した文字列が出力されます。

プログラムのビルド

翻訳オプション SSIN および SSOUT を指定してはなりません。

コマンド行からビルドする場合、コンパイラオプション/target に winexe を指定してはなりません。

Visual Studio プロジェクトを使用してビルドする場合、 プロジェクト デザイナーの[アプリケーション]ページで、「出力の種類」に"Windows アプリケーション"を指定してはなりません。

プログラムの実行

通常のプログラムを実行するときと同様に実行してください。ただし、コマンドプロンプトを使用する場合は、環境が正しく設定されたコマンドプロンプトから実行してください。コマンドプロンプトの起動については、コマンドプロンプトの起動を参照してください。

プログラム中の ACCEPT 文が実行されると、コンソールウィンドウにデータの入力が要求されます。

データの入力が要求されたら、必要なデータを入力してください。入力データは ACCEPT 文に指定した長さ分(80バイト)だけそのデータに設定されます。入力したデータの長さがデータ項目の長さより短い場合、文字のときには長さ分のデータが入力されるまで、数字のときには実行キーまたはリターンキーが入力されるまで入力要求が行われます。

プログラム中の DISPLAY 文が実行されると、コンソールウィンドウにデータが出力されます。

備考

1 回の入力データを 1 回の ACCEPT 文に対応付ける場合は、機能名 CONSOLE を使用します。 機能名 CONSOLE を使用した場合、入力したデータの長さがデータ項目の長さより短いときには空白詰めが行われます。

```
IDENTIFICATION DIVISION.
PROGRAM-ID. A.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
 SPECIAL-NAMES.
    CONSOLE IS CONS.
DATA DIVISION.
WORKING-STORAGE SECTION.
 01  データ１     PIC  X(80).
PROCEDURE DIVISION.
    ACCEPT   データ１  FROM  CONS.
    DISPLAY  データ１  UPON  CONS.
    EXIT PROGRAM.
END PROGRAM A.
```

コンソールウィンドウの[閉じる]ボタンをクリックした場合、およびコンソールウィンドウのポップアップメニューから[閉じる]コマンドを選択した場合は、プログラムを強制終了するかどうかを確認するダイアログが表示されます。このダイアログで強制終了を選択した場合は、プログラムを終了します。

NCOBU00229

標準エラー出力にメッセージを出力する

ここでは、小入出力機能を使って標準エラー出力にメッセージを出力するプログラムの書き方、ビルドおよび実行方法について説明します。

プログラムの記述

ここでは、標準エラー出力にメッセージを出力するプログラムの記述内容について、COBOL の部ごとに説明します。

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
 SPECIAL-NAMES.
    SYSERR IS 呼び名.
DATA DIVISION.
WORKING-STORAGE SECTION.
 01  データ名  PIC X(80).
PROCEDURE DIVISION.
    DISPLAY  データ名  UPON  呼び名.
    EXIT PROGRAM.
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

機能名 SYSERR に呼び名を対応付けます。

データ部(DATA DIVISION)

出力するデータを設定するためのデータ項目を定義します。これらのデータ項目は、次の属性のいずれかで定義します。

集団項目/英字項目/英数字項目/外部 10 進項目/英数字編集項目/数字編集項目/日本語項目/日本語編集項目

手続き部(PROCEDURE DIVISION)

標準エラー出力にメッセージを出力するには、UPON 指定に機能名 SYSERR に対応付けた呼び名を記述した DISPLAY 文を使います。DISPLAY 文にデータ名を指定した場合には、指定したデータ名に格納されているデータが出力され、文字定数を指定した場合には、指定した文字列が出力されます。

プログラムのビルド

コマンド行からビルドする場合、コンパイラオプション/target に winexe を指定してはなりません。

Visual Studio プロジェクトを使用してビルドする場合、 プロジェクト デザイナーの[アプリケーション]ページで、「出力の種類」に"Windows アプリケーション"を指定してはなりません。

プログラムの実行

通常のプログラムを実行するときと同じです。

ファイルを使うプログラム

ここでは、小入出力機能を使ってファイル処理を行うプログラムのうち、最も簡単な記述方法について、プログラムの書き方、ビルドおよび実行方法を説明します。

プログラムの記述

ここでは、小入出力機能を使ってファイル処理を行うプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  プログラム名.
DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  入力データ名  PIC X(80).
 01  出力データ名  PIC X(80).
PROCEDURE DIVISION.
   ACCEPT   入力データ名.
   DISPLAY  出力データ名.
   EXIT PROGRAM.
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

とくに必要な記述はありません。

データ部(DATA DIVISION)

入力したデータを格納するためのデータ項目および出力するデータを設定するためのデータ項目を定義します。これらのデータ項目は、次の属性のいずれかで定義します。

集団項目/英字項目/英数字項目/外部 10 進項目/英数字編集項目/数字編集項目/ 日本語項目[注]/日本語編集項目[注]

注：ACCEPT 文では使用できません。

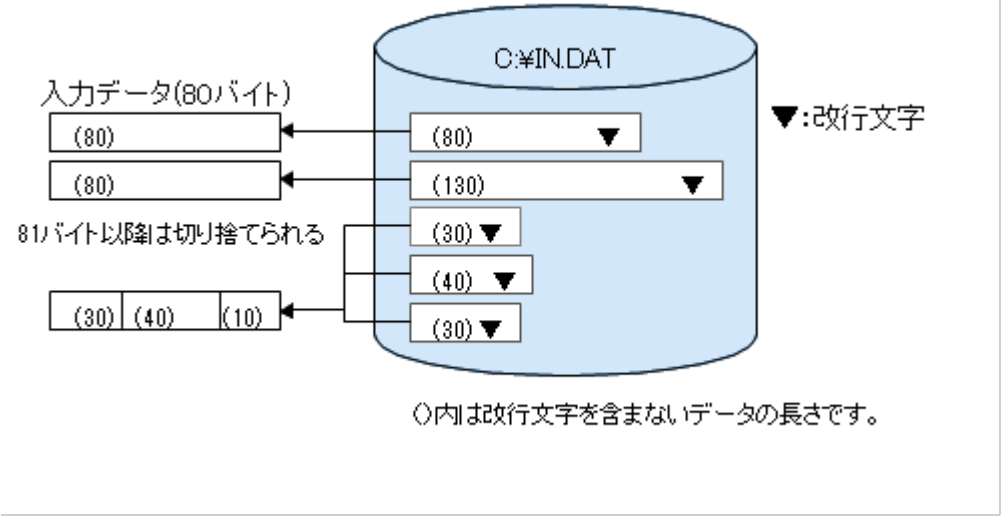
手続き部(PROCEDURE DIVISION)

プログラム実行時には、プログラムの実行開始から終了までに動作する実行文の中で、最初の ACCEPT 文で入力ファイルが開き、最初の DISPLAY 文で出力ファイルが開かれます。2 回目以降の ACCEPT 文および DISPLAY 文では、データの読込みおよびデータの出力だけが行われます。入力ファイルおよび出力ファイルは、プログラムの実行が終了するときに閉じられます。

[データの入力]

ファイルからデータを入力するには、ACCEPT 文を使います。ファイル中のデータは改行文字までを 1 レコードとし、入力データは 1 レコードずつ読み込まれます。

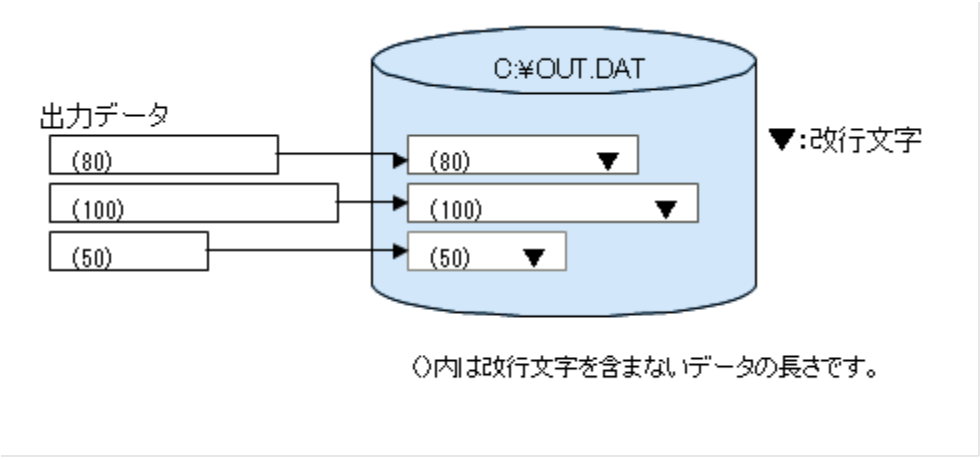
入力データは、ACCEPT 文に指定したデータ名に定義した長さ(上記の例では、01 入力データ PIC X(80).と定義しているので、英数字が 80 文字)の分だけ格納されます。なお、入力データの長さが格納するデータの長さより短い場合、次のレコードが読み込まれ、前に読み込まれたデータの後に連結されます。このとき、改行文字はデータとして扱われません。長さ分のデータが入力されるまでレコードの読込みが行われます。



ACCEPT 文に指定したデータ項目の長さ分のデータを入力するまで、レコードを読み込みます。3 枚目のレコードは 10 バイトだけが設定され、残りは切り捨てられます。

[データの出力]

ファイルにデータを出力するには、DISPLAY 文を使います。DISPLAY 文にデータ名を指定した場合にはデータ名に格納されている内容が、DISPLAY 文に文字定数を指定した場合には指定した文字列が、出力データとなります。 1 回の DISPLAY 文では、出力データの長さに改行文字の長さを加えた長さのデータが出力されます。



プログラムのビルド

小入出力機能のデータの入力先をファイルにする場合には、翻訳オプション SSIN を指定します。 また、データの出力先をファイルにする場合には、翻訳オプション SSOUT を指定します。

例：翻訳オプションの指定方法

```
SSIN(INDATA),SSOUT(OUTDATA)
```

- 翻訳オプション SSIN に環境変数情報名として SYSIN を指定した場合、ACCEPT 文のデータ入力先は、環境変数情報 SYSIN の設定にかかわらず、コンソールウィンドウとなります。
- 翻訳オプション SSOUT に環境変数情報名として SYSOUT を指定した場合、DISPLAY 文のデータ出力先は、環境変数情報 SYSOUT の指定にかかわらず、コンソールウィンドウとなります。

プログラムの実行

翻訳オプション SSIN および SSOUT に指定した環境変数情報名に、入出力処理を行うファイルの名前を設定します。

例：環境変数情報の指定方法

```
INDATA=C:¥IN.DATOUTDATA=C:¥OUT.DAT
```

- 入力ファイルは入力モードでオープンされ、共用モードで使用します。また、レコード読込み時にレコードがロックされることはありません。
- 出力ファイルは出力モードでオープンされ、排他モードで使用します。
- 出力先にすでに存在するファイルを指定した場合、そのファイルは作り直されます。(以前の情報は消去されます。)
- 改行文字はデータとして扱われません。
- ファイルをオープンした後(最初の ACCEPT 文および DISPLAY 文を実行した後)に、環境変数操作機能を使って入出力先を変更することはできません。

NCOBU00231

現在の日付および時刻の取得

ここでは、小入出力機能を使ってシステム時計による現在の日付や時刻を取得するプログラムの書き方、ビルドおよび実行方法について説明します。

プログラムの記述

ここでは、小入出力機能を使ってシステム時計による現在の日付や時刻を入力するプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  プログラム名.
DATA DIVISION.
WORKING-STORAGE SECTION.
 01  年月日    PIC 9(6).
 01  年日      PIC 9(5).
 01  曜日      PIC 9(1).
 01  時刻      PIC 9(8).
PROCEDURE DIVISION.
   ACCEPT   年月日    FROM DATE.
   ACCEPT   年日      FROM DAY.
   ACCEPT   曜日      FROM DAY-OF-WEEK.
   ACCEPT   時刻      FROM TIME.
   EXIT PROGRAM.
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

とくに必要な記述はありません。

データ部(DATA DIVISION)

入力したデータを格納するためのデータ項目を定義します。

手続き部(PROCEDURE DIVISION)

現在の日付および時刻を入力するには、FROM 指定に DATE(年月日)、DAY(年日)、DAY-OF-WEEK(曜日)または TIME(時刻)を指定した ACCEPT 文を使います。ACCEPT 文の書き方については、COBOL 文法書の「6.4.1 ACCEPT 文(中核)」を参照してください。

プログラムのビルド

とくに必要な翻訳オプションおよび追加ライブラリはありません。

プログラムの実行

通常のプログラムを実行するときと同様に実行します。

プログラム中の ACCEPT 文が実行されると、ACCEPT 文に指定したデータ名にそのときの日付および時刻が設定されます。

例：2002 年 12 月 25 日(水) 14 時 15 分 45 秒

FROM 句の書き方	データに設定される内容
FROM DATE	０２１２２５
FROM DAY	０２３５９
FROM DAY-OF-WEEK	３
FROM TIME	１４１５４５００

CURRENT-DATE 関数を利用すれば、西暦を 4 桁で取得することもできます。 詳細については、CURRENT-DATE 関数を利用した西暦の取得を参照してください。

NCOBU00837

任意の日付の入力

ここでは、小入出力を使って任意の日付を入力するプログラムの書き方、翻訳・リンク時の注意事項およびプログラムの実行方法について説明します。

プログラムの記述

ここでは、小入出力を使って任意の日付を入力するプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. プログラム名.
 DATA DIVISION.
  WORKING-STORAGE SECTION.
  01  年月日    PIC 9(6).
 PROCEDURE DIVISION.
    ACCEPT   年月日    FROM DATE.
    EXIT PROGRAM.
 END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

特に必要な記述はありません。

データ部(DATA DIVISION)

入力したデータを格納するためのデータ項目を定義します。

手続き部(PROCEDURE DIVISION)

任意日付を入力するには、FROM 指定に DATE(年月日)を指定した ACCEPT 文を使います。

プログラムのビルド

特に必要な翻訳・リンクオプションはありません。

プログラムの実行

任意日付を入力するプログラムを実行するときには、以下の環境変数情報の設定が必要です。

```
@CBR_JOBDATE=年.月.日
```

年.月.日は以下のように指定します。

- 年：(00-99)または(1900-2099)
- 月：(01-12)
- 日：(01-31)

“年”の値は、西暦 1900 年代ならば、西暦年の下 2 けたまたは 4 けたの西暦年です。西暦 2000 年代ならば、4 けたの西暦年です。

例

任意日付として、1990 年 10 月 1 日を指定します。

@CBR_JOBDATE=90.10.01

FROM 句の書き方	データ名に設定される内容
FROM DATE	901001

例

任意日付として、2004 年 10 月 1 日を指定します。

@CBR_JOBDATE=2004.10.01

FROM 句の書き方	データ名に設定される内容
FROM DATE	041001

NCOBU01078

イベントログにメッセージを出力する

ここではイベントログにメッセージを出力するプログラムの書き方、ビルド、および実行方法について説明します。

プログラムの記述

イベントログにメッセージを出力するプログラムの記述内容について、COBOL の部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. プログラム名.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     CONSOLE IS 呼び名.
DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 データ名 PIC X(80).
PROCEDURE DIVISION.
      DISPLAY データ名 UPON 呼び名.
      DISPLAY "文字定数" UPON 呼び名
      EXIT PROGRAM.
END PROGRAM プログラム名.
```

環境部(ENVIRONMENT DIVISION)

機能名 SYSOUT、SYSERR、または CONSOLE に呼び名を対応付けます。

データ部(DATA DIVISION)

出力するデータを設定するためのデータ項目を定義します。これらのデータ項目は、次の属性のいずれかで定義します。

集団項目/英字項目/英数字項目/外部 10 進/英数字編集項目/数字編集項目/日本語項目/日本語編集項目

手続き部(PROCEDURE DIVISION)

DISPLAY 文を使ってイベントログにメッセージを出力するには、UPON 指定に機能名 SYSOUT、SYSERR、または CONSOLE に対応付けた呼び名を指定します。DISPLAY 文にデータ名を指定した場合には、データ名に格納されているデータがイベントログに出力されます。また、DISPLAY 文に文字定数を指定した場合には、指定した文字列がイベントログに出力されます。

プログラムのビルド

特に必要な翻訳・リンクオプションはありません。

プログラムの実行

DISPLAY 文の UPON 指定ごとに以下の環境変数情報の設定が必要です。

UPON 指定なしまたは機能名 SYSOUT の出力先をイベントログにする場合

@CBR_DISPLAY_SYSOUT_OUTPUT=EVENTLOG[(コンピュータ名)]

機能名 SYSERR の出力先をイベントログにする場合

```
@CBR_DISPLAY_SYSERR_OUTPUT=EVENTLOG[(コンピュータ名)]
```

機能名 CONSOLE の出力先をイベントログにする場合

```
@CBR_DISPLAY_CONSOLE_OUTPUT=EVENTLOG[(コンピュータ名)]
```

- 一度にイベントログへ出力できるデータは 1024 文字です。それ以降は出力されません。
- イベントログに出力する場合、他の出力先(ファイル、コンソールなど)の指定は無効となり出力されません。
- イベントログ出力時のイベントソース名は"NetCOBOL for .NET"となります。
- イベントログ出力時のイベント種類は"情報"となります。イベント種類を変更したい場合は、以下の環境変数で変更してください。
  - UPON 指定なしまたは機能名 SYSOUT のイベント種類を変更する場合：

    ```
    @CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL
    ```

  - 機能名 SYSERR のイベント種類を変更する場合:

    ```
    @CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL
    ```

  - 機能名 CONSOLE のイベント種類を変更する場合:

    ```
    @CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL
    ```

- 指定したコンピュータのイベントログに出力対象メッセージを出力できない場合、実行中のコンピュータのイベントログに、エラーメッセージと出力対象メッセージを出力します。
- 出力先の Windows システム上でイベントログへの書き込み権限の設定(ユーザアカウント、ファイアウォールなど)が必要な場合があります。設定方法の詳細については、Microsoft のドキュメントを参照してください。

NCOBU00232

コマンド行引数の取り出し

ここでは、プログラムを呼び出したコマンドに指定した引数の数および値を参照する方法について説明します。

概要

プログラム実行中に、プログラムを呼び出したコマンドに指定された引数の数を求めたり、引数の値を参照することができます。

引数の数を求めるには、機能名 ARGUMENT-NUMBER に対応付けた呼び名を指定した ACCEPT 文を使います。

引数の値を参照するには、機能名 ARGUMENT-NUMBER に対応付けた呼び名を指定した DISPLAY 文と機能名 ARGUMENT-VALUE に対応付けた呼び名を指定した ACCEPT 文を使います。

なお、空白または二重引用符(")で囲まれた文字列が 1 つの引数として数えられます。

例：SAMPLE.exe の呼出しコマンドの例

```
SAMPLE  M.HORIUCHI  901234  SHIZUOKA

引数の数：3
引数の値：値(1)="M.HORIUCHI"
         値(2)="901234"
         値(3)="SHIZUOKA"
```

プログラムの記述例

コマンド行引数の操作機能を使うときのプログラムの記述について、COBOL の各部ごとに説明します。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID.  プログラム名.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
       SPECIAL-NAMES.
          ARGUMENT-NUMBER IS  呼び名１
           ARGUMENT-VALUE  IS  呼び名２.
      DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  引数の数     PIC 9(2)  COMP-5.
       01  引数の位置   PIC 9(2)  COMP-5.
       01  引数.
           02  引数の値 PIC X(10)  OCCURS 1 TO 10 TIMES
                              DEPENDING ON  引数の数.
       01  誤り番号     PIC 9(2)  COMP-5.
      PROCEDURE DIVISION.
          ACCEPT   引数の数    FROM 呼び名１         *>[1]
          DISPLAY  5           UPON 呼び名１         *>[2]
          ACCEPT   引数の値(5) FROM 呼び名２         *>[3]
             ON EXCEPTION MOVE  5 TO  誤り番号       *>[4]
                        GO TO 誤り処理
          END-ACCEPT.
      誤り処理.
          DISPLAY "引数の処理で例外が発生しました"
      END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

次の機能名を呼び名に対応付けます。

機能名	意味
ARGUMENT-NUMBER	コマンド行の引数を入力するため、または、コマンド行の引数を位置付けるために使います。
ARGUMENT-VALUE	コマンド行の引数の値を入力するために使います。

これらの機能名の詳細については、COBOL 文法書の「4.2.3.3 機能名-3 句」を参照してください。

データ部(DATA DIVISION)

値の受渡しを行うためのデータ項目を定義します。

内容	属性
引数の数	符号なし整数項目
引数の位置(定数で指定する場合は不要)	符号なし整数項目
引数の値	固定長集団項目または英数字項目

手続き部(PROCEDURE DIVISION)

[1]：引数の数を求めるには、機能名 ARGUMENT-NUMBER に対応付けた呼び名を指定した ACCEPT 文を使います。引数の数は、ACCEPT 文に指定したデータ名に設定されます。

[2]：引数の値を参照するには、機能名 ARGUMENT-NUMBER に対応付けた呼び名を指定した DISPLAY 文を使って、引数の位置を指定します。この例では、定数で指定していますが、以下のようにデータ名で指定することもできます。

```
MOVE 5 TO 引数の位置
DISPLAY   引数の位置  UPON 呼び名 1
```

[3]：機能名 ARGUMENT-VALUE に対応付けた呼び名を ACCEPT 文に指定して、値を取り出します。引数の値は、ACCEPT 文に指定したデータ名に設定されます。

[4]：ここで、存在しない引数の位置が指定されていた(引数の数が 3 つなのに引数の位置に 4 が指定されるなど)場合、例外条件が発生します。例外条件が発生すると、ACCEPT 文の ON EXCEPTION に指定された文が実行されます。

引数の位置付けには、0～99 を指定することができ、0 はコマンド名に位置付けられます。

機能名 ARGUMENT-NUMBER および機能名 ARGUMENT-VALUE を使用した ACCEPT 文および DISPLAY の書き方については、COBOL 文法書の「6.4.3 ACCEPT 文(コマンド行引数と環境変数の操作)」、「6.4.14 DISPLAY 文(コマンド行引数と環境変数の操作)」を参照してください。

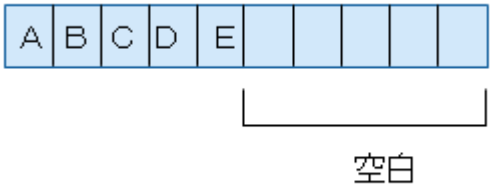
注意

- 位置付けのための DISPLAY 文を実行しないで引数の値を参照した場合、プログラム実行開始時に引数の位置は 1 に位置付けられ、その後 ACCEPT 文を実行するごとに、次の引数に位置付けられます。
- 引数の値の長さを得ることはできません。
- 引数の数および値のデータ項目への設定は、COBOL の MOVE 文の規則が適用されます。

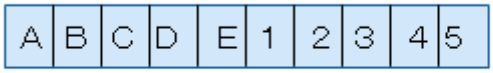
```
01  個数   PIC 9.
01  引数   PIC X(10).
     :
   ACCEPT 個数 FROM 呼び名１ *>[1]
   ACCEPT 引数 FROM 呼び名２ *>[2]
```

上記では、コマンドに指定された引数の数が 10 の場合、個数の定義が PIC 9 であるため、[1]を実行すると個数の内容は 0 となります。

また、取り出す引数の値が"ABCDE"の場合、[2]を実行すると引数の内容は以下のようになります。



取り出す引数の値が"ABCDE12345FGHIJ"の場合、[2]を実行すると引数の内容は以下のようになります。



プログラムのビルド

とくに必要な翻訳オプションおよび追加ライブラリはありません。

プログラムの実行

通常のプログラムを実行するときと同様に実行してください。

注意

- コマンド行引数の取り出しは、システムから起動したプログラムが COBOL プログラムの場合だけ使用することができます。
- COBOL プログラムから呼び出された COBOL プログラムの場合、参照する引数の値は、システムから起動されたコマンド行に指定した引数の値となります。

環境変数の操作機能

ここでは、環境変数の値を参照・更新する方法について説明します。なお、ここで説明する環境変数とは、プログラム実行時に設定する実行環境情報のことをいいます。

概要

プログラムの実行中に、環境変数の値を参照したり、更新したりすることができます。

環境変数の値を参照するには､機能名 ENVIRONMENT-NAME に対応付けた呼び名を指定した DISPLAY 文と、機能名 ENVIRONMENT-VALUE に対応付けた呼び名を指定した ACCEPT 文を使います。

環境変数の値を更新するには､機能名 ENVIRONMENT-NAME に対応付けた呼び名を指定した DISPLAY 文と、機能名 ENVIRONMENT-VALUE に対応付けた呼び名を指定した DISPLAY 文を使います。

プログラムの記述例

環境変数の操作機能を使うときのプログラムの記述について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
 SPECIAL-NAMES.
    ENVIRONMENT-NAME  IS   呼び名１
     ENVIRONMENT-VALUE IS   呼び名２.
DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  環境変数名    PIC X(8).
 01  環境変数の値  PIC X(240).
PROCEDURE DIVISION.
    DISPLAY "TMP1"       UPON   呼び名１.    *>[1]
    ACCEPT   環境変数の値 FROM   呼び名２      *>[2]
           ON EXCEPTION                       *>[3]
             GO TO エラー発生
    END-ACCEPT.
    DISPLAY "TMP2"       UPON   呼び名１     *>[4]
    DISPLAY  環境変数の値 UPON   呼び名２     *>[5]
           ON EXCEPTION                      *>[6]
             GO TO エラー発生
    END-DISPLAY
    EXIT PROGRAM.
エラー発生.
    DISPLAY "環境変数の処理でエラーが発生しました".
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

次の機能名を呼び名に対応付けます。

機能名	意味
ENVIRONMENT-NAME	環境変数を位置付けるために使います。
ENVIRONMENT-VALUE	環境変数の値を入力するため、または環境変数に値を設定するために使います。

これらの機能名の詳細については、COBOL 文法書の「4.2.3.3 機能名-3 句」を参照してください。

データ部(DATA DIVISION)

値の受け渡しを行うためのデータ項目を定義します。

内容	属性
環境変数名(定数で指定する場合は不要)	固定長集団項目または英数字項目
環境変数の値(定数で指定する場合は不要)	固定長集団項目または英数字項目

手続き部(PROCEDURE DIVISION)

[1] : 環境変数の値を参照するには、機能名 ENVIRONMENT-NAME に対応付けた呼び名を指定した DISPLAY 文を使って、参照する環境変数名を指定します。この例では、定数で指定していますが、以下のようにデータ名で指定することもできます。

```
MOVE "TMP1" TO 環境変数名
DISPLAY  環境変数名   UPON   呼び名 1
```

[2] : 機能名 ENVIRONMENT-VALUE に対応付けた呼び名を指定した ACCEPT 文で、環境変数の値を参照します。環境変数の値は、ACCEPT 文に指定したデータ名に設定されます。ただし、参照する環境変数名が指定されていなかったり、存在しない環境変数名が指定されていた場合、例外条件が発生します。

[3] : 例外条件が発生すると、ACCEPT 文の ON EXCEPTION に指定された文が実行されます。

[4] : 環境変数の値を更新するには、機能名 ENVIRONMENT-NAME に対応付けた呼び名を指定した DISPLAY 文を使って、更新する環境変数名を指定します。

[5] : 機能名 ENVIRONMENT-VALUE に対応付けた呼び名を指定した DISPLAY 文で環境変数の値を更新します。更新する環境変数の値は、DISPLAY 文に指定したデータ名に設定しておきます。この例では、環境変数 TMP1 の値で、環境変数 TMP2 の値を更新しようとしています。

[6]ここで、更新する環境変数名が指定されていなかったり、環境変数の値を設定する領域を割り付けることができなかった場合、例外条件が発生します。例外条件が発生すると、DISPLAY 文の ON EXCEPTION に指定された文が実行されます。

[注意] : 環境変数の値の長さを得ることはできません。

機能名 ENVIRONMENT-NAME および機能名 ENVIRONMENT-VALUE を使用した ACCEPT 文および DISPLAY の書き方については、COBOL 文法書の「6.4.3 ACCEPT 文(コマンド行引数と環境変数の操作)」、「6.4.14 DISPLAY 文(コマンド行引数と環境変数の操作)」を参照してください。

プログラムのビルド

とくに必要な翻訳オプションおよび追加ライブラリはありません。

プログラムの実行

通常のプログラムを実行するときと同様に実行してください。

- プログラムの実行中に変更した環境変数の値は、そのプログラムの実行しているプロセス内でだけ有効となります。
- DISPLAY 文による環境変数の値の更新では、現在のアプリケーション ドメインのアプリケーション ドメイン プロパティに、環境変数名をプロパティの名前、環境変数の文字列値をプロパティの値として設定します。 アプリケーションドメイン プロパティについては、.NET Framework のドキュメントの AppDomain クラスを参照してください。
- ACCEPT 文による環境変数の値の参照では、システムの環境変数の値を取得し、値が設定されていなければ、現在のアプリケーション ドメインのアプリケーション ドメイン プロパティから、環境変数名をプロパティの名前とし、プロパティの値を取得します。
- アプリケーション構成ファイルおよび実行用の初期化ファイルに指定された環境変 数は、現在のアプリケーション ドメインの環境変数に設定されるため、ACCEPT 文によって値を参照できます。

NCOBU00234

ファイルの処理

ここでは、ファイルからデータを読み込んだり、ファイルにデータを書き出したりする処理について説明します。

このセクションの内容

[他のファイルシステムの使用法]

利用できる他のファイルシステムについて説明します。

[COBOL ファイルユーティリティ関数]

COBOL ファイルユーティリティ関数について説明します。

関連トピックス

[COBOL ファイルユーティリティの使用方法]

COBOL ファイルユーティリティの使用方法について説明します。

[COBOL ファイルユーティリティコマンド]

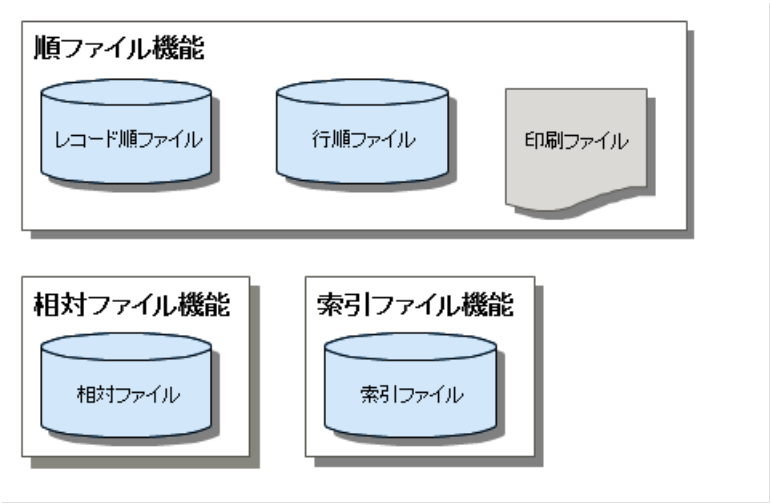
COBOL ファイルユーティリティコマンドモードの使用方法について説明します。

NCOBU00235

ファイルの種類

ファイルの種類と特徴

順ファイル機能、相対ファイル機能および索引ファイル機能を使って、以下のファイルを処理することができます。



ファイルの種類と特徴

ファイルの種類	レコード順ファイル	行順ファイル	印刷ファイル	相対ファイル	索引ファイル
レコードの処理	レコードの格納されている順番			相対レコード番号	レコードキーの値
利用できる媒体	ハードディスク(*2)フロッピーディスク	ハードディスク(*2)フロッピーディスク	印刷装置	ハードディスク(*2)フロッピーディスク	ハードディスク(*2)フロッピーディスク
利用例	データ退避作業ファイル	テキストファイル	データの印刷	作業ファイル	マスタファイル
ファイルの最大サイズ(バイト数)	1G (*1)	1G (*1)	－	1G	1.7G

***1：**

ファイルの高速処理を指定した場合、ファイルの最大サイズはシステム制限までになります。

***2：**

仮想デバイス(NUL など)は指定できません。

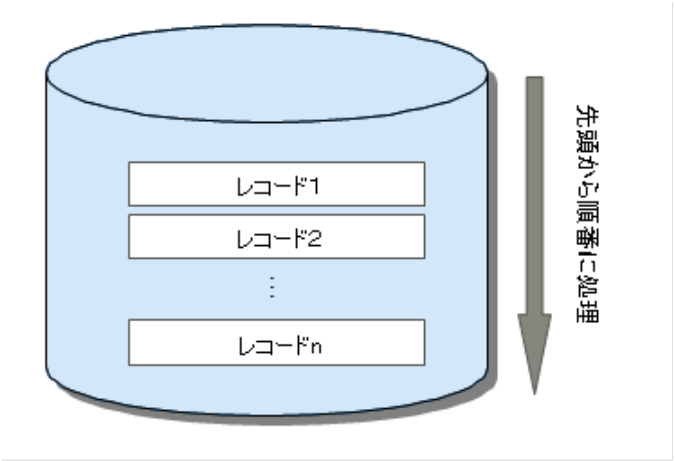
ファイルの種類は、ファイルを作成するときに決定され、あとで変更することはできません。ファイルを作成するときには、ファイルの特徴を十分に理解し、用途に合ったファイルの種類を選択してください。

NCOBU00236

レコード順ファイル

レコード順ファイルでは、ファイルの先頭レコードからファイルに書き出した順番でレコードを読んだり、更新したりできます。

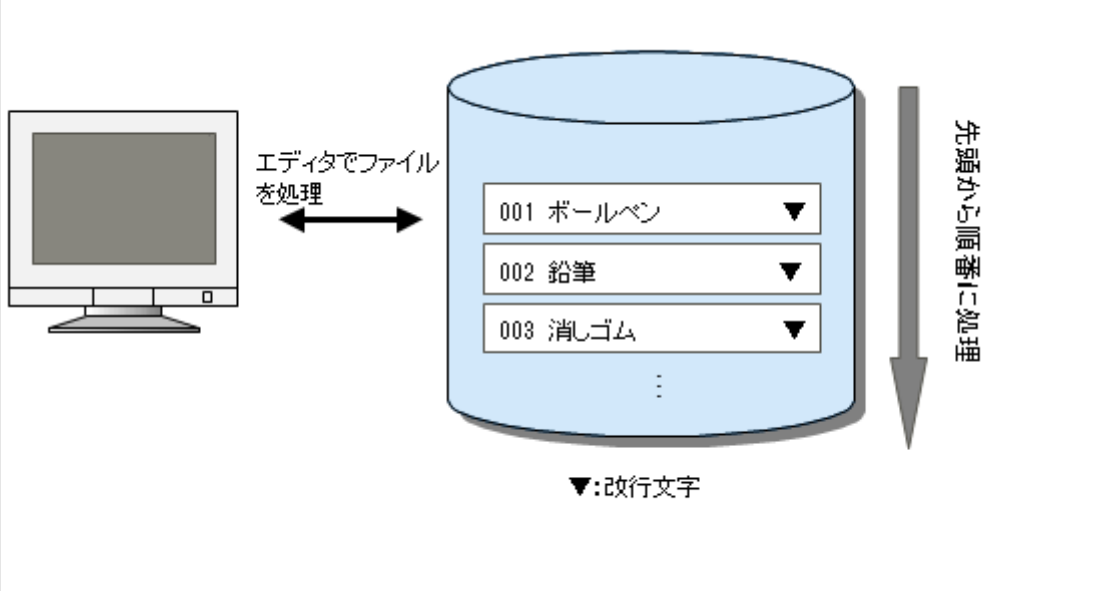
レコード順ファイルは、最も簡単に扱うことのできるファイルで、データを順次蓄積する場合や、大量データを保存する場合などに効果的です。



行順ファイル

行順ファイルでは、ファイルの先頭レコードからファイルに書き出した順番でレコードを読むことができます。
行順ファイルでは、改行文字をレコードの区切りとします。

行順ファイルは、エディターで作成したテキストファイルを操作するときなどに利用します。



改行文字は、2 バイトの大きさです。以下に、改行文字の内容を 16 進数表記で示します。

0x0D	0x0A

注意

- レコード読み込み時の改行文字の扱いについては、行順ファイルの処理の“レコード内の制御文字の扱い”を参照してください。
- ADVANCING 指定付きの WRITE 文を実行した場合、改行文字以外の制御文字が出力されます。詳細は、行順ファイルの処理の“ADVANCING 指定時の動作”を参照してください。
- CSV 形式データの操作については、CSV 形式データの操作を参照してください。

NCOBU00238

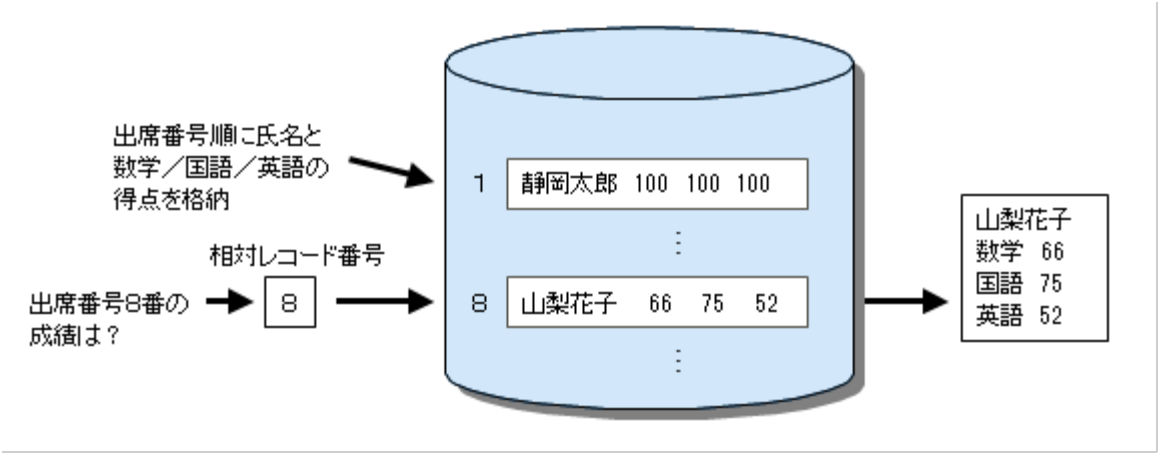
印刷ファイル

印刷ファイルとは、順ファイル機能を使用した印刷するためのファイルのことで、印刷ファイルという特別なファイルがあるわけではありません。

相対ファイル

相対ファイルでは、ファイルの先頭のレコードを 1 とする相対レコード番号を指定することによって、レコードを読んだり、更新したりできます。

相対ファイルは、相対レコード番号をキーとしてアクセスする作業ファイルなどに利用します。

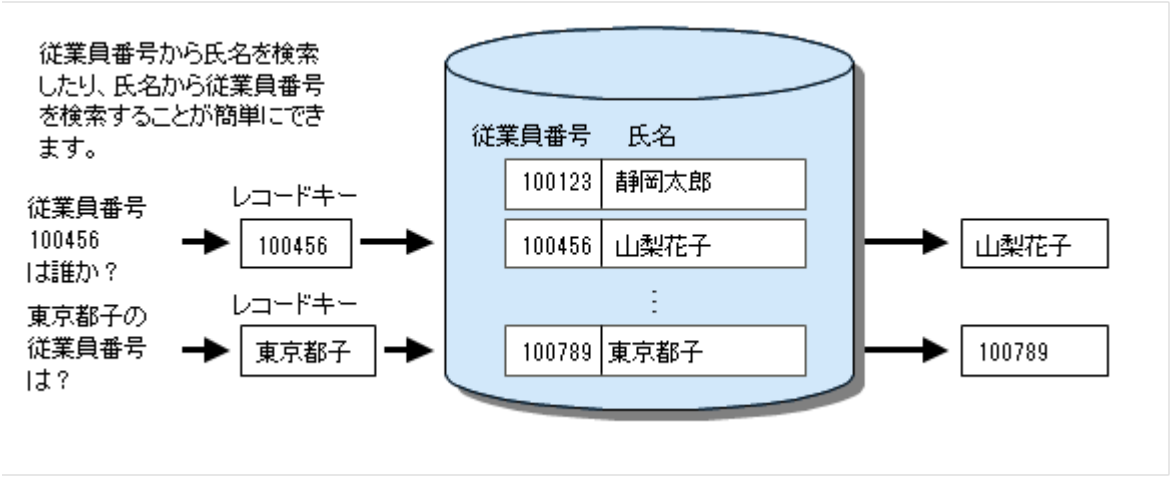


NCOBU00240

索引ファイル

索引ファイルでは、レコード中のある項目の値(レコードキー)を指定することによって、レコードを読んだり、更新したりできます。

索引ファイルは、レコード中のある項目の値から他の情報を引き出すマスタファイルなどに利用します。



レコードの設計

ここでは、レコード形式の種類と特徴および索引ファイルを使用するときのレコードキーについて説明します。

レコード形式

レコード形式には、固定長レコード形式と可変長レコード形式があります。それぞれのレコード形式について、以下に説明します。

- 固定長レコード形式

  固定長レコード形式では、1 つのレコードは一定のレコード長で区切られ、ファイル中のレコードの大きさはすべて同じになります。

- 可変長レコード形式

  可変長レコード形式では、レコードごとにレコードの大きさが異なります。1 つのレコードの大きさは、そのレコードがファイルに書き出されたときの大きさとなります。

  可変長レコード形式は、必要な大きさでレコードを書き出すことができるため、ファイル容量を小さくしたい場合に有効です。

索引ファイルのレコードキー

索引ファイルのレコードの設計では、レコードキーを決定する必要があります。

レコードキーは、レコード中の項目で、複数個指定することもできます。レコードキーには、主レコードキー(主キー)と副レコードキー(副キー)があり、ファイル中のレコードは主キーの昇順に格納されます。

ファイル中のどのレコードを処理するかは、主キーおよび副キーの両方またはどちらかの値を指定することにより決定します。また、あるレコードから昇順に処理していくこともできます。

レコードキーを決定するときには、以下の注意が必要です。

- 同一ファイルを複数のレコード構成で処理したい場合、主キーは、すべてのレコード構成で同じ位置と大きさである必要があります。
- 可変長レコード形式の場合、レコードキーの位置は固定部になければなりません。

NCOBU00242

ファイルの処理方法の種類

ファイルに対する処理は、以下の 6 種類があります。

ファイルの創成	ファイルにレコードを書き出します。
ファイルの拡張	ファイルの最後のレコードの後にレコードを書き出します。
レコードの挿入	ファイルの任意の位置にレコードを書き出します。
レコードの参照	ファイルの中のレコードを読み込みます。
レコードの更新	ファイルの中のレコードを書き換えます。
レコードの削除	ファイルの中のレコードを削除します。

これらの処理が可能かどうかは、ファイルに対するアクセス形態で異なります。アクセス形態には、以下の 3 種類があります。

順呼出し	一連のレコードを一定の順序で処理します。
乱呼出し	任意のレコードを単独で処理します。
動的呼出し	順呼出しと乱呼出しの両方の処理ができます。

各ファイル編成で行うことのできる処理を以下に示します。

ファイルの種類と処理

ファイルの種類	アクセス形態	処理					
		創成	拡張	挿入	参照	更新	削除
レコード順ファイル	順呼出し	○	○	×	○	○	×
行順ファイル	順呼出し	○	○	×	○	×	×
印刷ファイル	順呼出し	○	○	×	×	×	×
相対ファイル/索引ファイル	順呼出し	○	○	×	○	○	○
	乱呼出し	○	×	○	○	○	○
	動的呼出し	○	×	○	○	○	○

○: 処理可能、×: 処理不可能

また、ファイル処理では、ファイル自体を排他モードにしたり、使用中のレコードを排他状態にしたりすることによって、他からのアクセスを不可能にすることができます。 これをファイルの排他制御といいます。ファイルの排他制御については、"ファイルの排他制御" で説明します。

実行時の動作の指定

ここでは、ファイルを使ったプログラムの実行時の動作を指定する、ファイルの割当て、ファイルの排他制御、およびファイル処理の結果について説明します。

このセクションの内容

NCOBU00244

ファイルの割当て

プログラムの実行時に入出力処理の対象となるファイルの決定方法は、ファイル管理記述項の ASSIGN 句の記述内容によって異なります。

ASSIGN 句には、以下の内容を記述できます。記述内容とファイルの関係については、それぞれの説明を参照してください。

- ASSIGN 句にファイル識別名を記述した場合
- ASSIGN 句にデータ名を記述した場合
- ASSIGN 句にファイル識別名定数を記述した場合
- ASSIGN 句に DISK を記述した場合

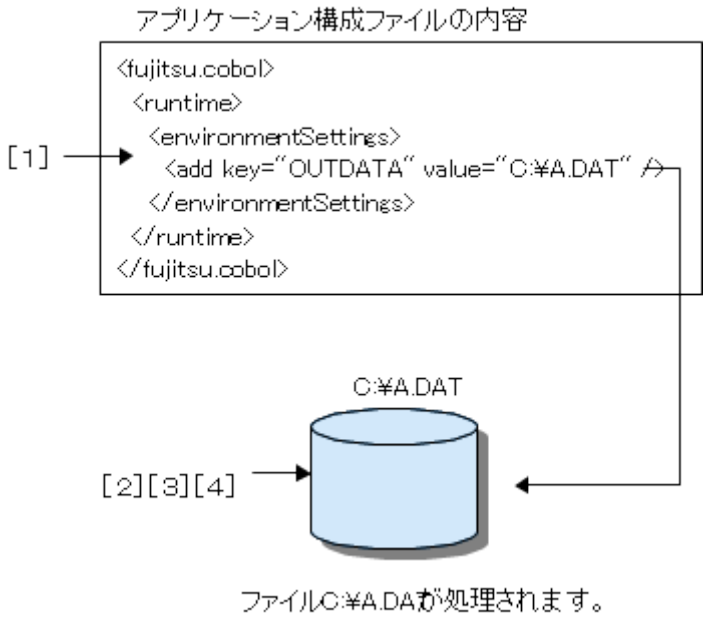
ASSIGN 句にファイル識別名を記述した場合

ファイル識別名を環境変数情報名として、プログラム実行時に、入出力処理の対象となるファイルの名前を設定します。

環境変数情報の設定方法については、実行環境の設定を参照してください。

以下に、アプリケーション構成ファイルを使った環境変数情報を設定する場合の例を示します。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. A.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.
          SELECT ファイル１
            ASSIGN TO OUTDATA.    *>[1]
      DATA DIVISION.
      FILE SECTION.
       FD ファイル１.
       01 レコード１ PIC X(80).
      PROCEDURE DIVISION.
          OPEN OUTPUT ファイル１. *>[2]
          WRITE レコード１.      *>[3]
          CLOSE ファイル１.      *>[4]
          EXIT PROGRAM.
      END PROGRAM A.
```



注意

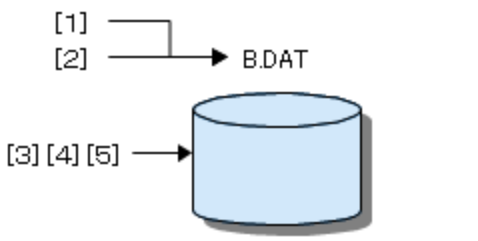
- ASSIGN 句に英小文字のファイル識別名を指定した場合、以下の注意が必要です。
  - 翻訳オプション NOALPHAL を指定して翻訳すると、翻訳エラーとなります。
  - 環境変数情報を設定するときには、英大文字で指定してください。
- 環境変数情報の内容が空白の場合、ファイルの割当てエラーとなります。

NCOBU00246

ASSIGN 句にデータ名を記述した場合

データ名に設定されたファイル名のファイルに対して入出力処理が行われます。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. B.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.
          SELECT ファイル１
            ASSIGN TO データ名１.     *>[1]
      DATA DIVISION.
      FILE SECTION.
       FD ファイル１.
       01 レコード１ PIC X(80).
      WORKING-STORAGE SECTION.
       01 データ名１ PIC X(30).
      PROCEDURE DIVISION.
          MOVE "B.DAT" TO データ名１. *>[2]
          OPEN OUTPUT ファイル１.     *>[3]
          WRITE レコード１.          *>[4]
          CLOSE ファイル１.          *>[5]
          EXIT PROGRAM.
      END PROGRAM B.
```



プログラムBを実行すると、ファイルB.DATが処理されます。

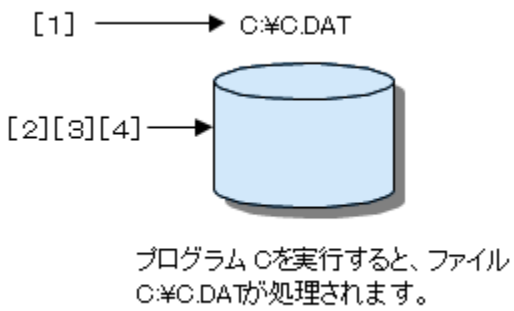
注意

- プログラムに記述されたファイル名が相対パス名の場合、カレントフォルダを先頭に付加したファイルが入出力処理の対象となります。
- データ名の内容が空白の場合、ファイルの割当てエラーとなります。

ASSIGN 句にファイル識別名定数を記述した場合

データ名に設定されたファイル名のファイルに対して入出力処理が行われます。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. C.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.
          SELECT ファイル１
            ASSIGN TO "C:¥C.DAT".     *>[1]
      DATA DIVISION.
      FILE SECTION.
       FD ファイル１.
       01 レコード１ PIC X(80).
      PROCEDURE DIVISION.
          OPEN OUTPUT ファイル１.      *>[2]
          WRITE レコード１.            *>[3]
          CLOSE ファイル１.            *>[4]
          EXIT PROGRAM.
      END PROGRAM C.
```



注意

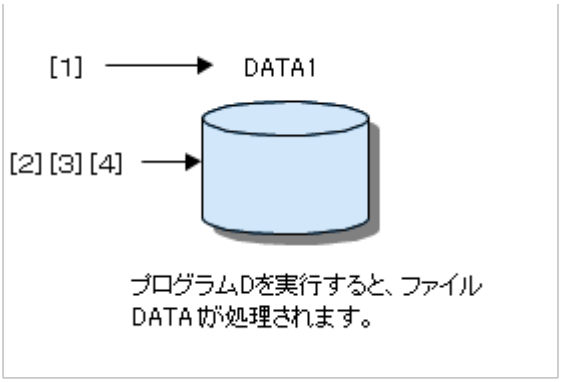
プログラムに記述されたファイル名が相対パス名の場合、カレントフォルダを先頭に付加したファイルが入出力処理の対象となります。

NCOBU00248

ASSIGN 句に DISK を記述した場合

SELECT 句に記述したファイル名のファイルに対して入出力処理が行われます。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. D.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.
          SELECT DATA1
            ASSIGN TO DISK.          *>[1]
      DATA DIVISION.
      FILE SECTION.
       FD DATA1.
       01 レコード１ PIC X(80).
      PROCEDURE DIVISION.
          OPEN OUTPUT DATA1.         *>[2]
          WRITE レコード１.          *>[3]
          CLOSE DATA1.               *>[4]
          EXIT PROGRAM.
      END PROGRAM D.
```



注意

ファイルはカレントフォルダが対象となります。

ファイルの排他制御

ファイル処理では、ファイル自体を排他モードにしたり、使用中のレコードを排他状態にしたりすることで、他からアクセスできないようにすることができます。これをファイルの排他制御といいます。

ファイルの排他制御は、COBOL プログラムまたは COBOL ファイルユーティリティを使用したアクセスに対して有効です。他言語や各種ツールからのアクセスではファイルの排他制御は無効となり、同時にアクセスした場合の動作は保証されません。

ここでは、ファイル処理とファイルの排他制御の関係について説明します。

ファイルを排他モードにする方法

ファイルを排他モードで開くと、他の利用者はそのファイルをアクセスすることができません。

以下の場合、ファイルは排他モードで開かれます。

- ファイル管理記述項の LOCK MODE 句に EXCLUSIVE を指定したファイルに対して、OPEN 文を実行した場合
- ファイル管理記述項の LOCK MODE 句を指定しないファイルに対して、INPUT モード以外の OPEN 文を実行した場合
- WITH LOCK 指定の OPEN 文を実行した場合
- OUTPUT モードの OPEN 文を実行した場合

上記の組み合わせによる、ファイルのモードの状態を以下の表に示します。

ファイルのモードの状態

	LOCK MODE 句が指定されていない		LOCK MODE 句に EXCLUSIVE が指定されている		LOCK MODE 句に AUTOMATIC または MANUAL が指定されている	
OPEN 文のモード	OPEN 文の WITH LOCK 指定あり	OPEN 文の WITH LOCK 指定なし	OPEN 文の WITH LOCK 指定あり	OPEN 文の WITH LOCK 指定なし	OPEN 文の WITH LOCK 指定あり	OPEN 文の WITH LOCK 指定なし
INPUT	排他	共用	排他	排他	排他	共用
I-O	排他	排他	排他	排他	排他	共用
EXTEND	排他	排他	排他	排他	排他	共用
OUTPUT	排他	排他	排他	排他	排他	排他

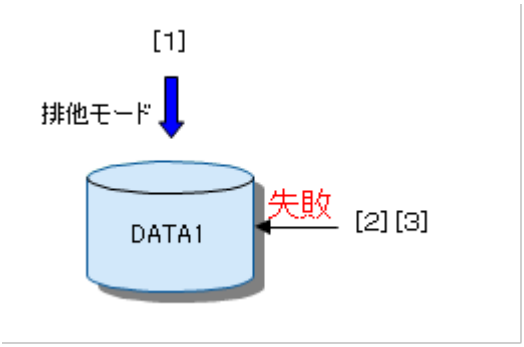
たとえば、以下のようなプログラム A とプログラム B から 1 つのファイルをアクセスした場合、プログラム A では排他モードでファイルをオープンしているため、プログラム B からそのファイルをアクセスしようとした時、エラーになります。

【プログラム A】

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. A.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.
          SELECT ファイル１
            ASSIGN TO "DATA1"
            LOCK MODE IS EXCLUSIVE.
      DATA DIVISION.
      FILE SECTION.
       FD ファイル１.
       01 レコード１ PIC X(80).
      PROCEDURE DIVISION.
          OPEN I-O ファイル１.       *>[1]
```

【プログラム B】

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. B.
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.
          SELECT ファイル２
            ASSIGN TO データ名
            FILE STATUS IS FS.
      DATA DIVISION.
      FILE SECTION.
       FD ファイル２.
       01 レコード１ PIC X(80).
      WORKING-STORAGE SECTION.
       01 データ名  PIC X(12).
       01 FS       PIC X(2).
      PROCEDURE DIVISION.
          MOVE "DATA1" TO データ名.
          OPEN I-O ファイル２.       *>[2]
          IF FS = "93" THEN          *>[3]
            MOVE "DATA2" TO データ名
            OPEN I-O ファイル２
          END-IF.
```



[図の説明]

[1] : 排他モードでファイル(DATA1)を開きます。

[2] : プログラム A で排他モードで使用されているファイル(DATA1)に対して、OPEN 文を実行してもエラーとなります。

[3] : FILE STATUS 句に指定したデータ名に入出力状態値”93”(ファイルの排他によるエラー発生)が設定されます。

レコードを排他状態にする方法

任意のレコードを排他状態にすると、他の利用者はそのレコードを処理することができません。任意のレコードを排他状態にするためには、まず、ファイルを共用モードで開きます。

次の場合、ファイルは共用モードで開かれます。

- ファイル管理記述項の LOCK MODE 句に AUTOMATIC または MANUAL を指定したファイルに対して、OUTPUT モード以外の WITH LOCK 指定なしの OPEN 文を実行した場合
- LOCK MODE 句を指定しないファイルに対して INPUT モードの OPEN 文を実行した場合

共用モードで開かれたファイルは、他の利用者と共用して使用することができます。ただし、すでに他の利用者がそのファイルを排他モードで使用しているとき、OPEN 文は失敗します。共用モードで開かれたファイルのレコードは、排他処理を指定した入出力文の実行により排他状態になります。

次の場合、レコードが排他状態となります。

- ファイル管理記述項の LOCK MODE 句に AUTOMATIC を指定したファイルを入出力モードで開き、WITH NO LOCK 指定なしの READ 文を実行した場合
- ファイル管理記述項の LOCK MODE 句に MANUAL を指定したファイルを入出力モードで開き、WITH LOCK 指定ありの READ 文を実行した場合

上記の組み合わせによる、レコードの状態を以下の表に示します。

レコードの状態(排他/共用)

LOCK MODE 句の記述	AUTOMATIC			MANUAL		
READ 文の記述	記述なし	WITH LOCK	WITH NO LOCK	記述なし	WITH LOCK	WITH NO LOCK
レコードの排他状態	する	する	しない	しない	する	しない

また、次の場合、レコードの排他状態が解除されます。

- LOCK MODE 句に AUTOMATIC を指定したファイルの場合
  - READ 文/REWRITE 文/WRITE 文/DELETE 文/START 文を実行します。
  - UNLOCK 文を実行します。
  - CLOSE 文を実行します。
- LOCK MODE 句に MANUAL を指定したファイルの場合
  - UNLOCK 文を実行します。
  - CLOSE 文を実行します。

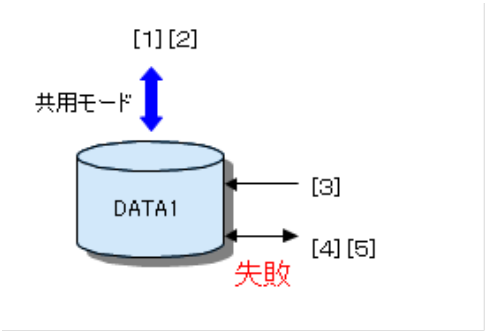
たとえば、以下のようなプログラム C とプログラム D から 1 つのファイルをアクセスした場合、プログラム C ではファイルの先頭レコードを排他状態にしているため、プログラム D からそのファイルのレコードを読み込もうとした時、エラーになります。

【プログラム C】

```
IDENTIFICATION DIVISION.
PROGRAM-ID. C.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT ファイル１
      ASSIGN TO SAMPLE
      LOCK MODE IS AUTOMATIC.
DATA DIVISION.
FILE SECTION.
 FD ファイル１.
 01 レコード１ PIC X(80).
PROCEDURE DIVISION.
    OPEN I-O ファイル１.        *>[1]
    READ ファイル１.            *>[2]
```

【プログラム D】

```
IDENTIFICATION DIVISION.
PROGRAM-ID. D.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT ファイル２
      ASSIGN TO SAMPLE
      LOCK MODE IS AUTOMATIC
      FILE STATUS IS FS.
DATA DIVISION.
FILE SECTION.
 FD ファイル２.
 01 レコード１ PIC X(80).
WORKING-STORAGE SECTION.
 01 FS       PIC X(2).
PROCEDURE DIVISION.
    OPEN I-O ファイル２.        *>[3]
    READ ファイル２.            *>[4]
    IF FS = "99" THEN          *>[5]
      GO TO 排他エラー発生
    END-IF.
```



[図の説明]

[1] : 共用モードでファイルを開きます。

[2] : READ 文の実行により、ファイルの先頭レコードは排他状態となります。

[3] : 共用モードでファイルを開きます。

[4] : 排他状態となっているレコードに対する READ 文はエラーとなります。

[5] : FILE STATUS 句に指定したデータ名に入出力状態値 "99" (レコードの排他によるエラー発生)が設定されます。

ファイル処理の結果

ファイル処理を行うことによって、新しいファイルが作られたり、ファイル中の内容が書き換えられたりします。ここでは、ファイル処理を行ったときのファイルの状態について説明します。

ファイルの創成処理を行ったとき

創成処理では、OPEN 文の実行により、新しいファイルが作られます。このとき、同じファイル名のファイルがすでに存在している場合、そのファイルは新しく作り直され、元の内容は失われます。

ファイルの拡張処理を行ったとき

既存ファイルに対しての拡張処理では、WRITE 文の実行によってファイルが拡張されます。プログラム実行時に存在しないファイル(不定ファイル)に対しての拡張処理では、OPEN 文の実行により、新しいファイルが作られます。

レコードの参照処理を行ったとき

参照処理では、参照したファイルの内容は変更されません。不定ファイルに対しての参照処理では、最初の READ 文でファイル終了条件が発生します。

レコードの更新/削除/挿入処理を行ったとき

既存ファイルに対しての更新/削除/挿入処理では、REWRITE 文/DELETE 文/WRITE 文の実行により、ファイルの内容が変更されます。

不定ファイルに対しての更新/削除/挿入処理では、OPEN 文の実行により、新しいファイルが作られます。ただし、このファイルにはデータが存在しないので、最初の READ 文でファイル終了条件が発生します。

- CLOSE 文を実行しないでプログラムを終了すると、そのファイルは強制的に閉じられます。 これを強制クローズといいます。強制クローズは、以下の場合に行なわれます。
  - STOP RUN 文の実行
  - 主プログラムでの EXIT PROGRAM 文の実行
  - 外部プログラムに対する CANCEL 文の実行

  もし、強制クローズに失敗した場合、メッセージが出力され、 そのファイルは開かれた状態のまま使用不可能となります。
- ファイル識別名を環境変数情報名としてファイルを割り当て、 ファイルが開かれた状態のまま環境変数操作機能によってファイルの割当て先を変更しても、 入出力文はファイル識別名で割り当てたファイルに対して行われます。 ファイル識別名で割り当てたファイルを CLOSE 文で閉じて、OPEN 文を実行すると、 その後の入出力文は環境変数操作機能で変更したファイルに対して行われます。 ファイルの一連の処理は、OPEN 文で開始し、CLOSE 文で終了するようにしてください。
- ファイルの創成・拡張処理またはレコードの更新・挿入処理で領域不足が発生した場合、 それ

以降のそのファイルに対する処理の正常な動作は保証できません。 また、領域不足発生時に書き出そうとしたレコードの内容が、 ファイル内にどのように格納されるかは規定できません。

- 索引ファイルを OUTPUT、I-O、または EXTEND モードで開いているとき、 ファイルを閉じる前にプログラムが異常終了すると、 そのファイルが使用できなくなることがあります。 異常終了する可能性のあるプログラムを実行する前には、 事前にバックアップすることをおすすめします。 なお、使用できなくなったファイルは、 COBOL ファイルユーティリティの［復旧］コマンドによって、 再び使用できる状態に復旧することができます。 また、そのコマンドと同じ機能を持つ関数をアプリケーションから呼び出すこともできます。 COBOL ファイルユーティリティについては、COBOL ファイルユーティリティの使用方法"を、 同じ機能を持つ関数(索引ファイルの復旧関数)については、索引ファイルの復旧関数を参照してください。

ファイルの高速処理

レコード順ファイルおよび行順ファイルでは、使用範囲を限定することでアクセス性能を高速化することができます。

本機能は、以下のような場合に使用すると効果的です。

- ファイルを排他モードで OPEN し、出力ファイルとして書き込みのみを行う場合
- 入力専用ファイルの読込みを行う場合

ここでは、ファイルの高速処理を行うために必要な指定方法および注意事項について説明します。

- ファイル単位の指定方法
- 一括の指定方法
- ファイル共用時の注意事項

NCOBU01088

ファイル単位の指定方法

ファイルの高速処理をファイル単位に指定する方法について説明します。

プログラム中のファイル参照子にファイル識別名を指定する場合

環境変数情報の設定時に、割り当てるファイルのファイル名に続き ",BSAM" を指定します。環境変数情報の設定方法については、ファイル識別名 (プログラムで使用するファイルの指定)を参照してください。

```
ファイル識別名=[パス名]ファイル名,BSAM
```

プログラム中のファイル参照子にデータ名を指定する場合

プログラム中のデータ名の指定時に、割り当てるファイルのファイル名に続き ",BSAM" を指定します。

ドライブ名を省略した場合、カレントドライブとみなされます。

```
MOVE "[パス名]ファイル名,BSAM"   TO データ名
```

プログラム中のファイル参照子にファイル識別名定数を指定する場合

プログラム中のファイル識別名定数の指定時に、割り当てるファイルのファイル名に続き ",BSAM"を指定します。

```
ASSIGN TO "[パス名] ファイル名,BSAM"
```

- レコードの更新(REWRITE 文)は、できません。レコードの更新を行った場合は、実行時にエラーとなります (レコード順ファイルだけ)。
- ファイル共用する場合には、以下の注意が必要です。
  - 他プロセス間でのファイル共用は、すべてのファイルが共用モードで、かつ INPUT 指定で開かれている必要があります。 INPUT 指定以外で開いたファイルがある場合、動作は保証されません。 また、同一プロセス内はファイル共用できません。同一プロセス内でファイル共用した場合、動作は保証されません。 なお、OPEN モードが INPUT 指定以外の場合は、常に排他モードで OPEN します。このため、他のプログラムからアクセスした際、OPEN エラーとなる場合があります。[参照]ファイル共用時の注意事項
- ファイル参照子に DISK を指定した場合、この機能は使用できません。
- 行順ファイルの場合、読み込んだレコードにタブが存在していても、そのタブコードを空白に置き換えません。 また、制御文字（0x0C(改頁)、0x0D(復帰)、0x1A(データ終了記号)）が含まれていても、レコードの区切り文字やファイルの終端として扱いません。 ファイルの高速処理を指定しない場合のタブや制御文字の扱いについては、行順ファイルの処理の“レコード内のタブ

の扱い”および“レコード内の制御文字の扱い”を参照してください。

- レコードの書込み(WRITE 文)における ADVANCING 指定は有効となりません。指定した場合は、ADVANCING 指定のない WRITE 文と同じ結果となります。

NCOBU01089

一括の指定方法

ファイルの高速処理を一括して有効とする指定方法について説明します。

使い方

環境変数情報@CBR_FILE_SEQUENTIAL_ACCESS に、"BSAM"を指定します。

```
@CBR_FILE_SEQUENTIAL_ACCESS=BSAM
```

ファイルの高速処理を有効とする場合、BSAM を指定します。本環境変数は、以下のファイルに対して有効となります。

ファイル編成	レコード順ファイル
	行順ファイル
ASSIGN 句の指定	ファイル識別名
	ファイル識別名定数
	データ名
	DISK

以下の機能を指定したファイルに対しては、有効となりません。

機能	ファイル機能名
ファイルの追加書き(*1)	MOD
ファイルの連結(*1)	CONCAT
他のファイルシステム	BTRV
	RDM
	EXFH

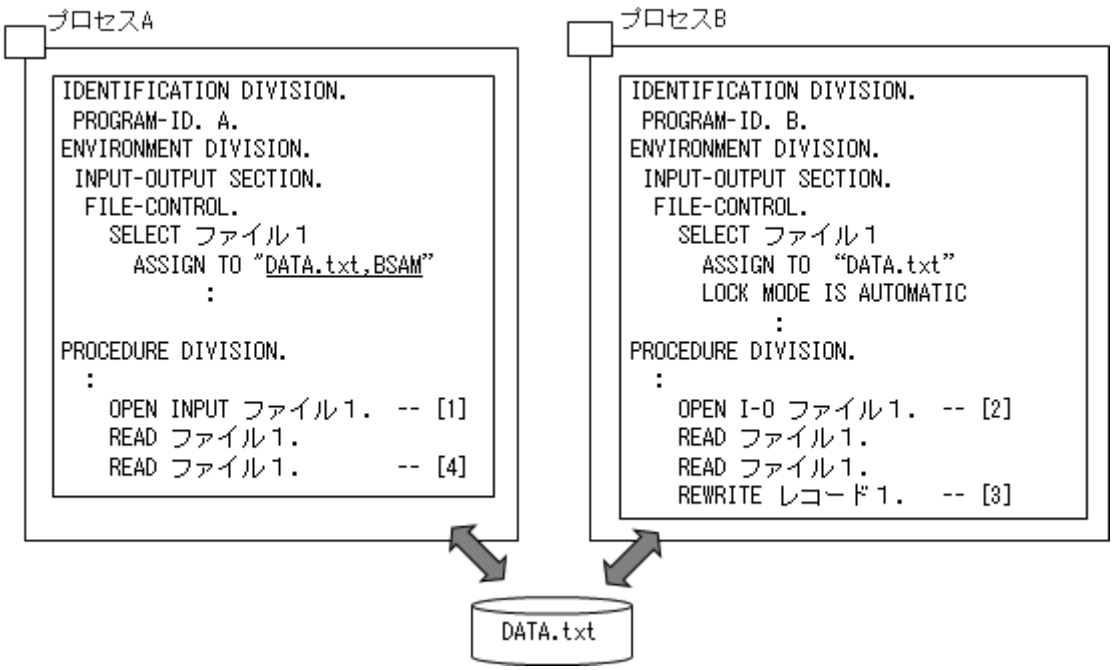
(*1)ファイルの高速処理を同時に有効としたい場合は、ファイル単位に指定してください。ファイル単位に指定する方法については、注意事項の“同時に指定可能なファイル機能の組合せ”を参照してください。

本環境変数の指定により、ファイルの高速処理の制限事項が適用されます。 特にファイルを共用モードで OPEN している場合、アプリケーションの動作が変わる場合があります。制限事項に該当するファイルがある場合、本環境変数を指定しないでください。[参照]ファイル共用時の注意事項

ファイル共用時の注意事項

ファイル共用する際に問題が発生する例について、以下に示します。

例 1：他プロセスからレコードの更新が行われる運用環境の場合、本機能は使用できません。



[図の説明]

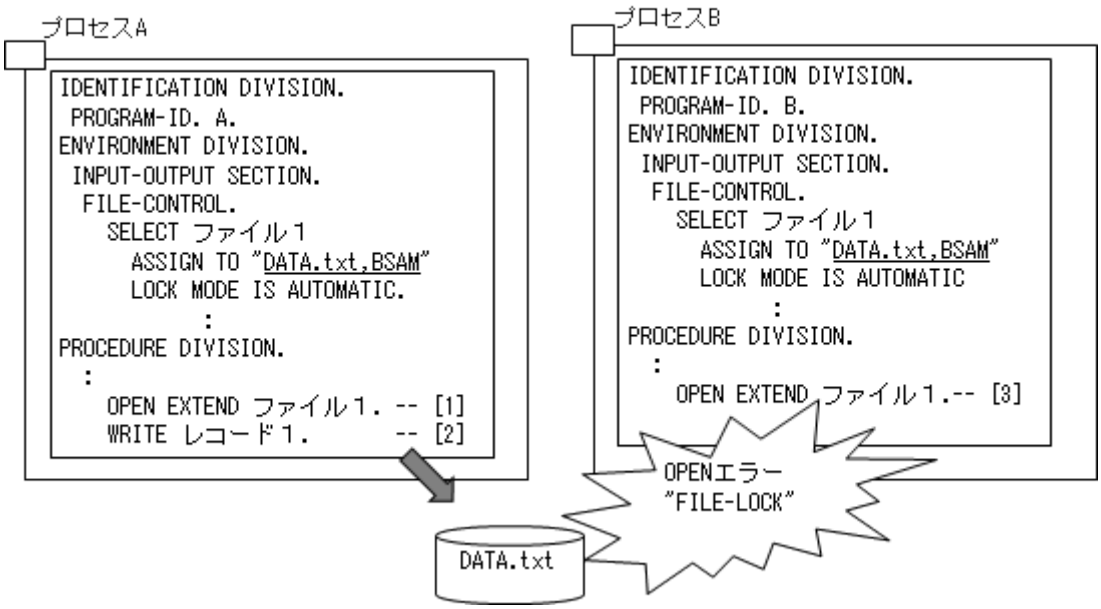
[1] プロセス A：共用モード、INPUT 指定でファイルを開きます。(ファイルの高速処理)

[2] プロセス B：共用モード、I-O 指定でファイルを開きます。

[3] プロセス B：2 件目のレコードを更新します。

[4] プロセス A：2 件目のレコードを読み込みます。ここで、更新前のデータが読み込まれる可能性があります。

例 2：ファイルを共用して書き込みを行う場合、後続の OPEN がエラーとなります。



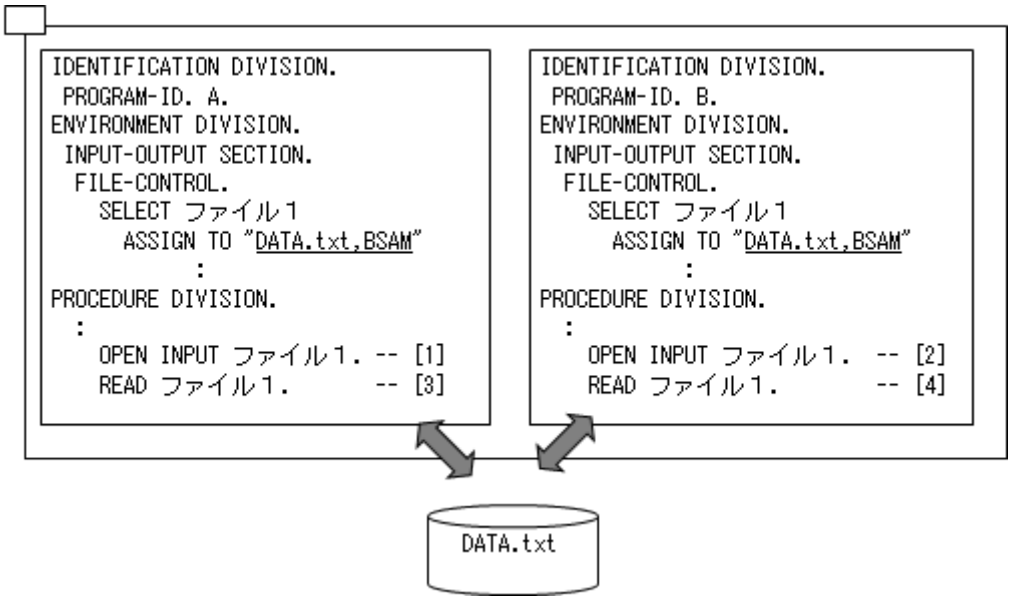
[図の説明]

[1] プロセス A：共用モード、EXTEND 指定でファイルを開きます。

[2] プロセス A：レコードを書き出します。

[3] プロセス B：共用モード、EXTEND 指定でファイルを開きます。ここで、OPEN 文がエラーとなります。

例 3：同一プロセス内でファイルを共用する場合、本機能は使用できません。



[図の説明]

[1] プログラム A：共用モード、INPUT 指定でファイルを開きます。

[2] プログラム B：共用モード、INPUT 指定でファイルを開きます。

[3] プログラム A：1 件目のレコードを読み込みます。

[4] プログラム B：1 件目のレコードを読み込みます。ここで、2 件目以降のレコードデータが読み込まれる、または、ファイル終了条件が発生する場合があります。

ファイル追加書き

OPEN OUTPUT 文の実行で、既に存在するファイルにレコードを追加することができます。以下に本機能の使い方について説明します。

使い方

ファイル識別名に、割り当てるファイルのファイル名に続き、“,MOD”を指定します。

```
ファイル識別名=ファイル名,,MOD
```

COBOL ファイルのレコード順ファイルにだけ有効となります。他のファイル編成および他のファイルシステムに指定することはできません。

NCOBU00839

ファイルの連結

複数のファイルを連結して、レコードを参照・更新することができます。以下に本機能の使い方について説明します。

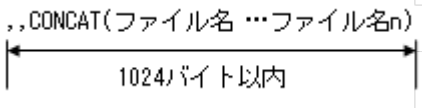
使い方

ファイル識別名に、“,CONCAT”を指定します。

```
ファイル識別名=,,CONCAT(ファイル名１ ファイル名２･･)
```

注意

- ファイル名は空白文字で区切ります。ファイル名に空白文字を含む場合は、二重引用符(")で囲む必要があります。
- COBOL ファイルのレコード順ファイルにだけ有効となります。他のファイル編成および他のファイルシステムに指定することはできません。
- OUTPUT 指定または EXTEND 指定の OPEN 文は実行時にエラーになります。
- 同一ファイルが複数指定された場合は、別ファイルを指定した場合と同じ動作となります。
- ファイル識別名に 1024 バイトを超える文字列を指定することはできません。したがって、連結可能なファイル数は、ファイル連結機能で指定するファイル名の長さに依存します。 ファイル識別子にファイル連結機能だけを指定する場合は、以下のように指定してください。



ファイル連結機能の指定の途中で文字列が 1024 バイトを超えた場合は、OPEN 文実行時にエラーになります。

注意事項

ファイル識別名に指定できる文字列のバイト数

ファイル識別名には､1024 バイト以内の文字列を指定してください。文字列が 1024 バイトを超えた場合は､1024 バイトまでの文字列を有効とみなして処理します。 ただし、ファイル連結機能の指定の途中で文字列が 1024 バイトを超えた場合は、OPEN 文実行時にエラーになります。

同時に指定可能なファイル機能の組合せ

ファイル機能は、以下の３つの種別に分類することができます。

種別	ファイル機能名	機能
(1) アクセス種別	BSAM	ファイルの高速処理
(2) ファイルシステム種別	BTRV	Btrieve ファイル
	RDM	RDM ファイル
	EXFH	外部ファイルハンドラ
(3) その他	MOD	ファイル追加書き
	CONCAT	ファイルの連結

ファイル機能は、次の形式で指定してください。指定順序が異なる場合、ファイル機能は有効になりません。

```
                ì (1)アクセス種別          ü
ファイル名,     í                          ý, (3)その他
                ï (2)ファイルシステム種別  þ
```

同時に指定可能な組合せとその動作は、以下のとおりです。

- (1)アクセス種別のファイル高速処理(BSAM)と(3)その他は、同時に指定することができます。また、いずれも有効になります。 例:ファイル高速処理(BSAM)とその他を同時に指定

```
ファイル名,BSAM,MOD
```

```
,BSAM,CONCAT(ファイル名 1 ファイル名 2 …)
```

- (3)その他の機能は、ASSIGN 句の指定がファイル識別名以外の場合、OPEN 文実行時にエラーになります。
- 指定が有効にならない組み合わせを指定した場合の動作は、以下のとおりです。
    - (2)ファイルシステム種別を先に指定した場合、後に指定した機能は無効にし、処理を続行します。
    - 上記以外の場合、OPEN 文実行時にエラーとなります。

NCOBU00251

レコード順ファイルの使い方

ここでは、レコード順ファイルの設計方法および処理方法について説明します。

レコード順ファイルを使用する COBOL プログラム

```
IDENTIFICATION DIVISION.
PROGRAM-ID. プログラム名.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT ファイル名
          ASSIGN TO ファイル参照子
         [ORGANIZATION IS SEQUENTIAL].
DATA DIVISION.
FILE SECTION.
 FD ファイル名
    [RECORD レコードの大きさ].
 01 レコード名.
    レコード記述項
     :
PROCEDURE DIVISION.
    OPEN オープンモード ファイル名.
   [READ ファイル名.]
   [REWRITE レコード名.]
   [WRITE レコード名.]
    CLOSE ファイル名.
END PROGRAM プログラム名.
```

このセクションの内容

[レコード順ファイルの定義方法]

レコード順ファイルを定義する場合の、環境部のファイル管理段落(FILE-CONTROL)の書き方について説明しています。

[レコード順ファイルのレコードの定義方法]

レコード順ファイルのレコードを定義する場合の、データ部のファイル記述項の書き方について説明しています。

[レコード順ファイルの処理]

レコード順ファイルを使用する場合の、用途と手続き部の書き方について説明しています。

レコード順ファイルの定義方法

ここでは、レコード順ファイルを使うときに必要なファイルの定義について説明します。

ファイル名とファイル参照子

```
SELECT  ファイル名  ASSIGN  TO  ファイル参照子 …
```

1. まず、COBOL プログラムで使用するファイル名を決定し、SELECT 句に記述します。このファイル名は、COBOL の利用者語の規則に従った名前にします。
2. 次に、ファイル参照子を決定し、ASSIGN 句に記述します。ファイル参照子には、ファイル識別名、ファイル識別名定数、データ名または文字列 DISK のどれかを指定します。ファイル参照子は、SELECT 句に指定した COBOL プログラムのファイル名と実際の入出力媒体のファイルとを関連付けるために使用します。ファイル参照子に何を指定したかによって、COBOL プログラムのファイル名と実際の入出力媒体のファイルを関連付ける方法が異なります。

ファイル参照子に何を指定するかは、実際の入出力媒体のファイル名がいつ決まるかにより、以下のように決定することをおすすめします。

- COBOL プログラム作成時に実際の入出力媒体のファイル名が決定し、その後変更されない場合には、ファイル識別名定数または文字列 DISK を指定します。
- COBOL プログラム作成時に実際の入出力媒体のファイル名が決定しなかったり、毎回のプログラム実行時にファイル名を決定したい場合には、ファイル識別名を指定します。
- プログラムの中でファイル名を決定したい場合には、データ名を指定します。
- プログラム終了後に不必要となる一時的なファイルである場合には、文字列 DISK を指定します。

文字列 DISK を指定した場合、プログラムの終了時に、使用したファイルが削除されるわけ ではありません。

また、実際の入出力媒体のファイル名には、以下の文字を含むことができます。

```
空白, "+", ",", ";", "=", "[", "]", "(", ")", "'"
```

なお、コンマ(,)を含む場合には、ファイル名を二重引用符(”)で囲む必要があります。

備考

レコード順ファイルおよび行順ファイルでは、ファイルを高速に処理することができます。 指定方法については、ファイルの高速処理を参照してください。

SELECT 句および ASSIGN 句の記述例を、以下の表に示します。

SELECT 句および ASSIGN 句の記述例

ファイル参照子の種類	記述例	備考
ファイル識別名	SELECT ファイル名１ ASSIGN TO INFILE	プログラム実行時に実際の入出力媒体と結び付ける必要があります。
データ名	SELECT ファイル名２ ASSIGN TO データ名	データ名はデータ部の作業場所節で定義する必要があります。
ファイル識別名定数	SELECT ファイル名３ ASSIGN TO "C.dat"	－
文字列 DISK	SELECT DATA1 ASSIGN TO DISK	ファイル名を絶対パス名で指定することはできません。

ファイル編成

ORGANIZATION 句に SEQUENTIAL を指定します。なお、ORGANIZATION 句を省略した場合には、SEQUENTIAL を指定したものとみなされます。

レコード順ファイルのレコードの定義方法

ここでは、レコード形式とレコード長およびレコードの構成について説明します。

レコード形式とレコード長

レコード順ファイルのレコード形式には、固定長レコード形式と可変長レコード形式があります。

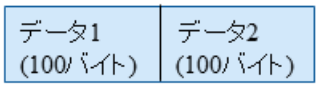
固定長レコード形式のレコード長は、RECORD 句に指定した値、または RECORD 句が省略された場合はレコード記述項の最大値となります。

可変長レコード形式では、レコードを書き出したときのレコードの長さが、そのレコードのレコード長になります。書き出すレコードの長さは、RECORD 句の“DEPENDING ON データ名”に記述したデータ名に設定することができます。また利用者は、このデータ名を使って、レコードの入力時にレコード長を得ることもできます。

レコードの構成

レコード中のデータの属性、位置および大きさは、レコード記述項で定義します。以下に、レコードの定義例を示します。

【固定長レコード形式のレコードの定義例】




```
FD データ保存用ファイルＦ.
01 データレコード.
  02 データ１ PIC X(100).
  02 データ２ PIC X(100).
```

OCCURS 句を指定した可変長データ項目を含む複数のデータ項目から構成される可変長レコード形式の場合、以下の注意が必要です。

- レコードの書き出し時

  レコード項目にデータを転記するとき、最初の可変長データ項目から順にデータおよびデータ長を転記しなければなりません。データを転記する順番によっては、意図しない転記結果となる場合があります。

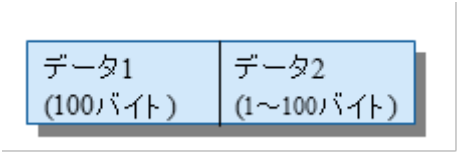
- レコードの読込み時

  レコードの読込みでは、読み込んだレコードの全体の長さは返却されますが、レコード内に含まれる可変長データ項目の長さは返却されません。

  この場合、全体のレコード長から固定部の長さを引いて可変長データ項目の長さを求めてください。

  だだし、複数の可変長データ項目が定義されている場合、計算では長さを求めることはできません。この場合、レコード内に可変部分の終わりを示す情報を埋め込むか、または、各可変部分の長さを持つなどして、個々の可変長データ項目の長さを求めてください。

【可変長レコード形式のレコードの定義例】




```
 FD データ保存用ファイルＶ
    RECORD IS VARYING IN SIZE FROM 101 TO 200 CHARACTERS
                      DEPENDING ON レコード長.
 01 データレコード.
    02  データ１  PIC X(100).
    02  データ２.
        03  PIC X OCCURS 1 TO 100 TIMES DEPENDING ON  長さ.
WORKING-STORAGE SECTION.
 01  レコード長    PIC 9(3) BINARY.
 01  長さ         PIC 9(3) BINARY.
```

レコード順ファイルの処理

レコード順ファイルの処理では、入出力文を使って、創成、拡張、参照、更新を行うことができます。ここでは、レコード順ファイルの処理で使用する入出力文の種類と使い方およびそれぞれの処理の概要について説明します。

入出力文の種類

- OPEN 文
- CLOSE 文
- READ 文
- REWRITE 文
- WRITE 文

入出力文の使い方

OPEN 文および CLOSE 文

OPEN 文はファイル処理の最初に、CLOSE 文はファイル処理の最後にそれぞれ 1 回だけ必ず実行します。

OPEN 文に指定するオープンモードは、ファイルに対してどのような処理を行うかによって決定されます。たとえば、創成処理を行う場合には、出力モード(OUTPUT 指定)の OPEN 文を実行します。

その他の文

READ 文は、ファイル中のレコードを読み込むときに使用します。

REWRITE 文は、ファイル中のレコードを更新するときに使用します。

WRITE 文は、ファイルにレコードを書き出すときに使用します。

処理の概要

創成

レコード順ファイルを創成するには、ファイルを出力モードで開いて、WRITE 文を使ってファイルにレコードを書き出していきます。

```
OPEN OUTPUT ファイル名.
レコードの編集処理
WRITE レコード名 ～ .
CLOSE ファイル名.
```

すでに存在するファイルに対して創成処理を行った場合、そのファイルは新しく作り直され、元の内容は失われます。

拡張

レコード順ファイルの拡張は、ファイルを拡張モードで開いて、WRITE 文を使ってファイルにレコードを書き出していきます。このとき、ファイルの最後のレコードの後ろにレコードが追加されます。

```
OPEN EXTEND ファイル名.
レコード編集処理
WRITE レコード名 ～ .
CLOSE ファイル名.
```

参照

レコードの参照では、ファイルを入力モードで開いて、READ 文を使ってファイルのレコードを先頭から順番に読み込みます。

```
OPEN INPUT ファイル名.
READ ファイル名 ～ .
CLOSE  ファイル名.
```

ファイル管理記述項の SELECT 句に OPTIONAL を指定した場合、OPEN 文実行時にファイルが存在しなくても OPEN 文は成功となり、最初の READ 文の実行でファイル終了条件が成立します。

ファイル終了条件については、入出力エラー処理を参照してください。

更新

レコードの更新では、ファイルを入出力モードで開いて、まず READ 文を使ってファイルのレコードを読込み、次に REWRITE 文を使ってレコードを書き換えます。

```
OPEN I-O ファイル名.
READ ファイル名 ～ .
レコードの編集処理
REWRITE レコード名 ～.
CLOSE ファイル名.
```

- REWRITE 文を実行した場合、直前の READ 文により読み込まれたレコードの内容が更新されます。
- レコード形式が可変長の場合、レコードの長さを変更することはできません。

行順ファイルの使い方

ここでは、行順ファイルの設計方法および処理方法について説明します。

行順ファイルを使用する COBOL プログラム

```
IDENTIFICATION DIVISION.
PROGRAM-ID. プログラム名.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT ファイル名
          ASSIGN TO ファイル参照子
          ORGANIZATION IS LINE SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
 FD ファイル名
    [RECORD レコードの大きさ].
 01 レコード名.
    レコード記述項
PROCEDURE DIVISION.
    OPEN オープンモード ファイル名.
    [READ ファイル名.]
    [WRITE レコード名.]
    CLOSE ファイル名.
END PROGRAM プログラム名.
```

このセクションの内容

NCOBU00256

行順ファイルの定義方法

ここでは、行順ファイルを使うときに必要なファイルの定義について説明します。

ファイル名とファイル参照子

行順ファイルでは、レコード順ファイルと同様に、ファイル名とファイル参照子を指定します。

指定方法については、レコード順ファイルのレコード順ファイルの定義方法を参照してください。

ファイル編成

ORGANIZATION 句に LINE SEQUENTIAL を指定します。

行順ファイルのレコードの定義方法

ここでは、レコード形式とレコード長およびレコードの構成について説明します。

レコード形式とレコード長

行順ファイルのレコード形式とレコード長の説明は、レコード順ファイルの説明と同じです。レコード順ファイルのレコードの定義方法を参照してください。

なお、行順ファイルの 1 つのレコードは改行文字で区切られるので、レコード形式に関係なく 1 つのレコードの最後は改行文字になります。 ただし、レコード長にはこの改行文字の長さは含まれません。

レコードの構成

レコード中のデータの属性、位置および大きさは、レコード記述項で定義します。なお、レコードを区切るための改行文字は、レコードを書き出すときに付加されるため、レコード記述項で定義する必要はありません。

【可変長レコード形式のレコードの定義例】


```
      FD テキストファイル
         RECORD IS VARYING IN SIZE FROM 1 TO 80 CHARACTERS
         DEPENDING ON レコード長.
       01 テキストコード.
         02 テキスト文字列.
           03 文字 PIC X OCCURS 1 TO 80 TIMES
                              DEPENDING ON レコード長.
     WORKING-STORAGE SECTION.
       01 レコード長 PIC 9(3) COMP-5.
```

NCOBU00258

行順ファイルの処理

行順ファイルの処理では、入出力文を使って、創成、拡張、参照を行うことができます。 ここでは、行順ファイルの処理で使用する入出力文の種類と使い方およびそれぞれの処理の概要について説明します。

入出力文の種類

- OPEN 文
- CLOSE 文
- READ 文
- WRITE 文

入出力文の使い方

OPEN 文および CLOSE 文

OPEN 文はファイル処理の最初に、CLOSE 文はファイル処理の最後にそれぞれ 1 回だけ必ず実行します。

OPEN 文に指定するオープンモードは、ファイルに対してどのような処理を行うかによって決定されます。たとえば、創成処理を行う場合には、出力モード(OUTPUT 指定)の OPEN 文を実行します。

その他の文

READ 文は、ファイル中のレコードを読み込むときに使用します。

WRITE 文は、ファイルにレコードを書き出すときに使用します。

処理の概要

創成

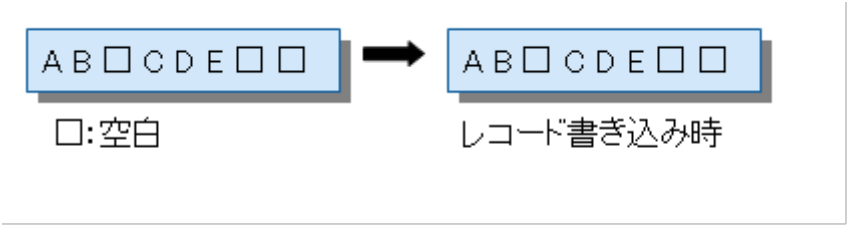
レコード順ファイルを創成するには、ファイルを出力モードで開いて、WRITE 文を使ってファイルにレコードを書き出します。

```
OPEN OUTPUT ファイル名.
レコードの編集処理
WRITE レコード名 ～ .
CLOSE ファイル名.
```

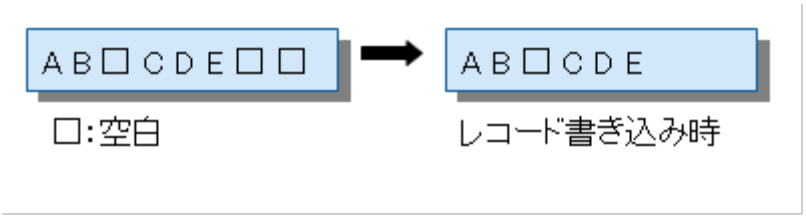
- すでに存在するファイルに対して創成処理を行った場合、そのファイルは新しく作り直され、元の内容は失われます。

- レコードの書出しでは、レコードの領域の内容と改行文字が書き出されます。
- レコード内の後置空白を取り除いてレコードを書き出す場合は、環境変数情報@CBR_TRAILING_BLANK_RECORD (行順ファイルのレコード内後置空白を取り除くまたは有効にする指定)に文字列 "REMOVE" を指定します。

〔後置空白を削除する機能を無効にした場合（デフォルト)〕



〔後置空白を削除する機能を有効にした場合〕



レコードの後ろに設定されている空白を削除して、ファイルに書込みます。

拡張

行順ファイルの拡張は、ファイルを拡張モードで開いて、WRITE 文を使って順番にレコードを書き出していきます。 このとき、ファイルの最後のレコードの後ろにレコードが追加されます。

```
OPEN EXTEND ファイル名.
レコード編集処理
WRITE レコード名 ～ .
CLOSE ファイル名.
```

レコード内の後置空白を取り除いてレコードを書き出す場合は、環境変数情報@CBR_TRAILING_BLANK_RECORD (行順ファイルのレコード内後置空白を取り除くまたは有効にする指定)に文字列“REMOVE”を指定します。

参照

レコードの参照では、ファイルを入力モードで開いて、READ 文を使ってファイルのレコードを先頭から順番に読み込みます。

```
OPEN INPUT ファイル名.
READ ファイル名 ～ .
CLOSE  ファイル名.
```

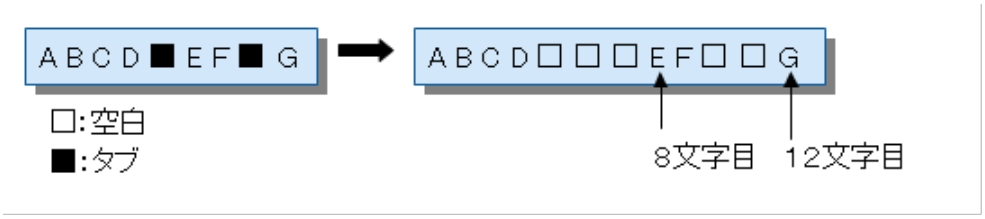
注意

- ファイル管理記述項の SELECT 句に OPTIONAL を指定した場合、OPEN 文実行時にファイルが存在しなくても OPEN 文は成功となり、最初の READ 文の実行でファイル終了条件が成立します。ファイル終了条件については、入出力エラー処理を参照してください。
- 読み込んだレコードの大きさがレコード長より大きいときには、1 回の READ 文の実行でレコード長と同じ長さのデータがレコード領域に設定されます。 次の READ 文の実行では、同じレコードのデータの続きから、データがレコード領域に設定されます。 設定するデータがレコード長より小さい場合には、レコード領域の残りの領域に、空白が設定されます。

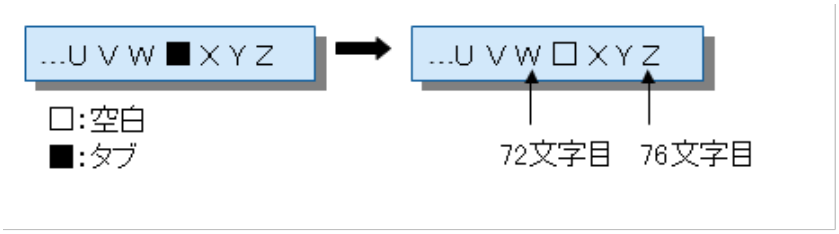
レコード内のタブの扱い

読み込んだレコードにタブが存在した場合、先頭の文字位置を 1 として、8、12、16、20、24、28、32、36、40、44、48、52、56、60、64、68、72 の文字位置にタブの次の文字が配置されるように、空白が設定されます。 なお、72 を超える文字位置にタブが存在する場合には、1 文字の空白が設定されます。

[７２文字以内にタブが存在する場合]



[７２文字を超えた位置にタブが存在する場合]



レコード内の制御文字の扱い

読み込むレコードに制御文字が含まれている場合の動作について、以下に説明します。

制御文字	動作
0x0C(改頁)	レコードの区切り文字として扱います。
0x0D(復帰)	レコードの区切り文字として扱います。
0x1A(データ終了記号)	ファイルの終端として扱います。

〔補足〕()内は、制御文字の意味を示します。

ADVANCING 指定時の動作

ADVANCING 指定付きの WRITE 文を実行した場合、改行文字以外の制御文字が出力されます。 ADVANCING 指定によりファイルに出力されるデータを以下に示します。

ADVANCING 指定		出力されるデータ
なし		{レコードデータ}0x0D 0x0A
BEFORE	n LINES (*1)	{レコードデータ}0x0D 0x0A … 0x0A n個
	PAGE	{レコードデータ}0x0D 0x0C
AFTER	n LINES (*1)	0x0A … 0x0A{レコードデータ}0x0D n個
	PAGE (*2)	0x0C{レコードデータ}0x0D

***1:**

行数に 0 を指定した場合、レコードの後ろに 0x0D(復帰)のみを付加したレコードが出力されます。 0x0A(改行)は出力されません。

***2:**

OPEN OUTPUT 直後の WRITE 文では、0x0C(改頁)は出力されません。

ADVANCING 指定の WRITE 文で書き出されたレコードを読み込む場合、特に前後のレコードの制御文字が連続している場合は、意図したレコード区切りで読み込まれないことがあります。

- 以下の連続する制御文字は、1 つのレコード区切り文字として扱います。 － 0x0D(復帰)に続き、0x0A(改行)や 0x0C(改頁)が存在する － 0x0A(改行)や 0x0C(改頁)に続き、0x0D(復帰)が存在する
- 0x0A(改行)や 0x0C(改頁)が連続する場合、1 つの制御文字として扱います。
- レコードの先頭にある 0x0A(改行)や 0x0C(改頁)は、読み飛ばされます。

制御文字が連続しない場合の動作については、“レコード内の制御文字の扱い”を参照してください。

NCOBU00259

相対ファイルの使い方

ここでは、相対ファイルの設計方法および処理方法について説明します。

相対ファイルを使用する COBOL プログラム

```
IDENTIFICATION DIVISION.
PROGRAM-ID. プログラム名.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT ファイル名
         ASSIGN TO ファイル参照子
         ORGANIZATION IS RELATIVE
        [ACCESS MODE IS アクセス形態]
        [RELATIVE KEY IS 相対レコード番号格納域].
DATA DIVISION.
FILE SECTION.
FD ファイル名
   [RECORD レコードの大きさ].
01 レコード名.
  レコード記述項
  02 データ名 ～ .
    :
WORKING-STORAGE SECTION.
[01 相対レコード番号格納域  PIC 9(5) BINARY.]
PROCEDURE DIVISION.
   OPEN オープンモード ファイル名.
   [MOVE 相対レコード番号 TO 相対レコード番号格納域.]
   [READ ファイル名.]
   [REWRITE レコード名.]
   [DELETE ファイル名.]
   [START ファイル名.]
   [WRITE レコード名.]
   CLOSE ファイル名.
END PROGRAM プログラム名.
```

このセクションの内容

相対ファイルの定義方法

ここでは、相対ファイルを使うときに必要なファイルの定義について説明します。

ファイル名とファイル参照子

相対ファイルでは、レコード順ファイルと同様に、ファイル名とファイル参照子を指定します。指定方法については、レコード順ファイルの定義方法を参照してください。

ファイル編成

ORGANIZATION 句に RELATIVE を指定します。

アクセス形態

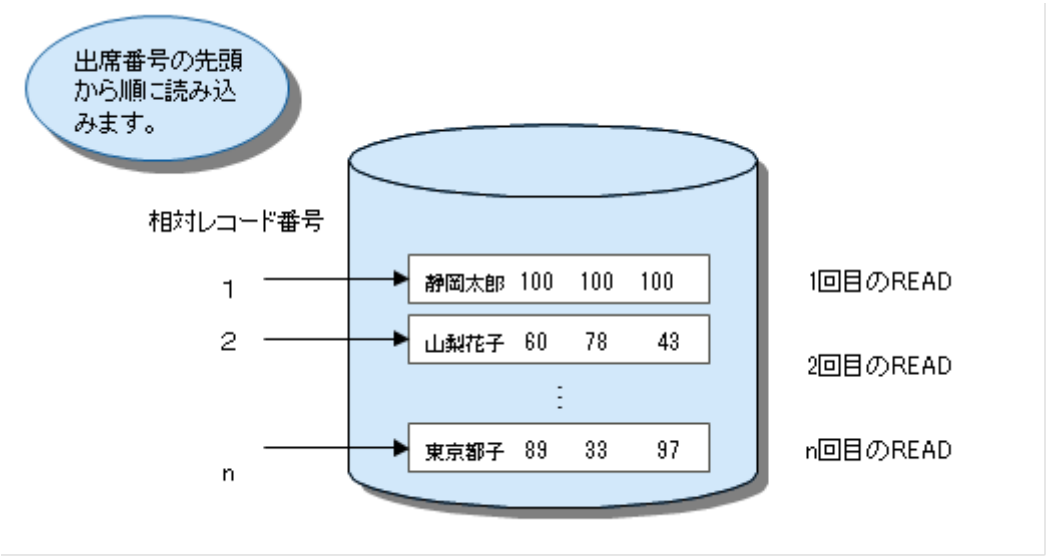
ACCESS MODE 句に以下のアクセス形態のどれかを指定します。

順呼出し法 (SEQUENTIAL)	ファイルの先頭またはある相対レコード番号のレコードから、相対レコード番号の昇順にレコードを処理することができる、順呼出しの処理を行うことができます。
乱呼出し法 (RANDOM)	ある相対レコード番号のレコードだけを単独に処理することができる、乱呼出しの処理を行うことができます。
動的呼出し法 (DYNAMIC)	順呼出しと乱呼出しの両方の処理を行うことができます。

以下に、レコードの参照処理を例に、順呼出しの場合と乱呼出しの場合での処理の違いを示します。

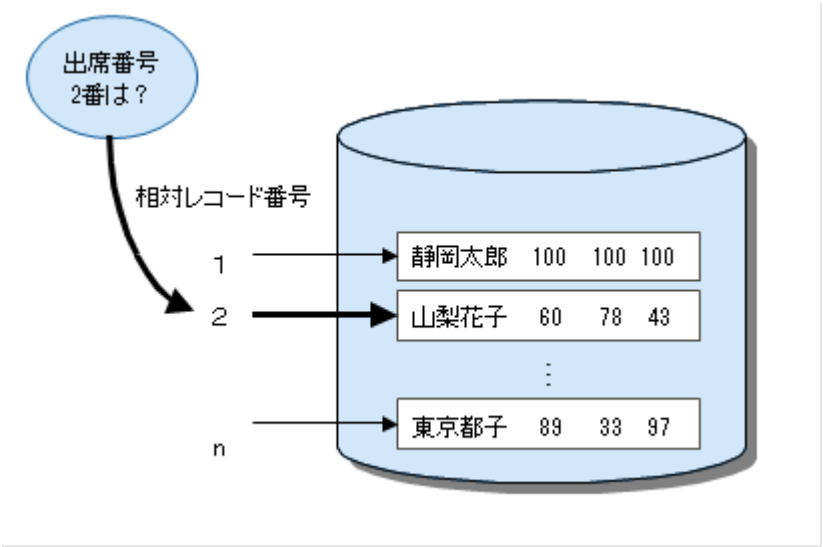
順呼出し

〔出席番号順に格納されているレコードを先頭から順番に処理する場合〕



乱呼出し

〔出席番号順に格納されているレコードの、ある出席番号の者のデータを処理する場合〕



相対レコード番号

相対レコード番号を設定するデータ名を、RELATIVE KEY 句に指定します。ただし、順呼出しの場合は、この句を省略することができます。このデータ名には、レコードの入力時には入力したレコードの相対レコード番号が設定され、出力時には書き出すレコードの相対レコード番号を利用者が設定します。

ただし、レコードの出力を順呼出しでアクセスする場合、利用者が設定した相対レコード番号は無視されます。なお、このデータ名は、符号なし整数項目として作業場所節に定義する必要があります。

相対ファイルのレコードの定義方法

ここでは、レコード形式とレコード長およびレコードの構成について説明します。

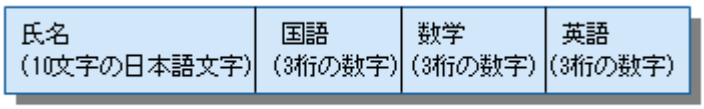
レコード形式とレコード長

相対ファイルのレコード形式とレコード長の説明は、レコード順ファイルの説明と同じです。レコード順ファイルのレコードの定義方法を参照してください。

レコードの構成

レコード中のデータの属性、位置および大きさは、レコード記述項で定義します。なお、相対レコード番号を設定するための領域を定義する必要はありません。

〔固定長レコード形式のレコードの定義例〕




```
FD  学級ファイル.
01  成績レコード.
    02  氏名       PIC N(10).
    02  国語       PIC 9(3).
    02  数学       PIC 9(3).
    02  英語       PIC 9(3).
```

NCOBU00262

相対ファイルの処理

相対ファイルの処理では、入出力文を使って、創成、拡張、挿入、参照、更新、削除を行うことができます。ここでは、相対ファイルの処理で使用する入出力文の種類と使い方およびそれぞれの処理の概要について説明します。

入出力文の種類

- OPEN 文
- CLOSE 文
- DELETE 文
- READ 文
- START 文
- REWRITE 文
- WRITE 文

入出力文の使い方

OPEN 文および CLOSE 文

OPEN 文はファイル処理の最初に、CLOSE 文はファイル処理の最後にそれぞれ 1 回だけ必ず実行します。

OPEN 文に指定するオープンモードは、ファイルに対してどのような処理を行うかによって決定されます。たとえば、創成処理を行う場合には、出力モード(OUTPUT 指定)の OPEN 文を実行します。

その他の文

DELETE 文は、ファイル中のレコードを削除するときに使用します。

READ 文は、ファイル中のレコードを読み込むときに使用します。

START 文は、処理を開始するレコードを指示するときに指定します。

REWRITE 文は、ファイル中のレコードを更新するときに使用します。

WRITE 文は、ファイルにレコードを書き出すときに使用します。

処理の概要

創成〔順/乱/動的〕

相対ファイルを創成するには、ファイルを出力モードで開いて、WRITE 文でファイルにレコードを書き出していきます。レコードは、WRITE 文に指定したレコードの長さで書き出されます。順呼出し法の場合、書き出したレコードの順番に、相対レコード番号が 1、2、3…となります。乱呼出し法または動的呼出し法の場合、相対レコード番号で指定した位置にレコードが書き出されます。

```
OPEN OUTPUT  ファイル名.
レコードの編集処理
[相対レコード番号の設定]
WRITE  レコード名  ～.
CLOSE  ファイル名.
```

すでに存在するファイルに対して創成処理を行った場合、そのファイルは新しく作り直され、元の内容は失われます。

拡張〔順〕

相対ファイルの拡張は、ファイルを拡張モードで開いて、WRITE 文で順番にファイルにレコードを書き出します。このとき、ファイルの最後のレコードの後ろにレコードが追加されます。書き出すレコードの相対レコード番号は、ファイルの最大の相対レコード番号から 1 つ大きい値となります。ファイルの拡張が可能なアクセス形態は、順呼出し法だけです。

```
        OPEN EXTEND  ファイル名.
        レコードの編集処理
        WRITE  レコード名  ～.
        CLOSE  ファイル名.
```

参照〔順/乱/動的〕

レコードの参照では、ファイルを入力モードで開いて、READ 文でファイルのレコードを読み込みます。順呼出しの場合、START 文を使って読み込むレコードの開始位置を指定し、そのレコードから相対レコード番号の順番にレコードを読み込んでいきます。乱呼出しの場合、READ 文実行時に設定されている相対レコード番号のレコードが読み込まれます。

```
【順呼出し】
        OPEN INPUT  ファイル名.
       [相対レコード番号の設定
        START  ファイル名  ～.]
        READ   ファイル名 [NEXT] ～.
        CLOSE  ファイル名.
```

```
【乱呼出し】
        OPEN INPUT  ファイル名.
        相対レコード番号の設定
        READ   ファイル名  ～.
        CLOSE  ファイル名.
```

- ファイル管理記述項の SELECT 句に OPTIONAL を指定した場合、OPEN 文実行時にファイルが存在していなくても OPEN 文は成功となり、最初の READ 文の実行でファイル終了条件が成立します。 ファイル終了条件については、入出力エラー処理を参照してください。
- 乱呼出し法または動的呼出し法で、指定した相対レコード番号のレコードが存在しないとき、無効キー条件が成立します。 無効キー条件については、入出力エラー処理を参照してください。

更新〔順/乱/動的〕

レコードの更新では、ファイルを入出力モードで開きます。順呼出しの場合、まず READ 文でレコードを読込み、次に REWRITE 文でレコードを書き換えます。REWRITE 文の実行で、直前の READ 文により読み込まれたレコードの内容が変更されます。乱呼出しの場合、内容を変更したいレコードの相対レコード番号を設定して REWRITE 文を実行します。

```
【順呼出し】
         OPEN I-O  ファイル名.
        [相対レコード番号の設定
         START  ファイル名  ～.]
         READ   ファイル名 [NEXT] ～.
         レコードの編集処理
         REWRITE  レコード名  ～.
         CLOSE    ファイル名.
```

```
【乱呼出し】
         OPEN I-O  ファイル名.
         相対レコード番号の設定
        [READ   ファイル名  ～.]
         レコードの編集処理
         REWRITE  レコード名  ～.
         CLOSE    ファイル名.
```

乱呼出し法または動的呼出し法で、指定した相対レコード番号のレコードが存在しないときに無効キー条件が成立します。無効キー条件については、入出力エラー処理を参照してください。

削除〔順/乱/動的〕

レコードの削除では、ファイルを入出力モードで開きます。順呼出しの場合、まず READ 文でレコードを読込み、次に DELETE 文でレコードを削除します。DELETE 文の実行で、直前の READ 文により読み込まれたレコードが削除されます。乱呼出しの場合、削除したいレコードの相対レコード番号を設定して DELETE 文を実行します。

```
【順呼出し】
         OPEN I-O  ファイル名.
         [相対レコード番号の設定
         START  ファイル名  ～.]
         READ   ファイル名 [NEXT] ～.
         DELETE ファイル名  ～.
         CLOSE  ファイル名.
```

```
【乱呼出し】
         OPEN I-O  ファイル名.
         相対レコード番号の設定
         DELETE ファイル名  ～.
         CLOSE  ファイル名.
```

乱呼出し法または動的呼出し法で、指定した相対レコード番号のレコードが存在しないときに無効キー条件が成立します。無効キー条件については、入出力エラー処理を参照してください。

挿入〔乱/動的〕

レコードの挿入では、ファイルを入出力モードで開いて、挿入したい位置の相対レコード番号を設定して WRITE 文を実行します。レコードは、設定した相対レコード番号の位置に挿入されます。

```
OPEN I-O ファイル名.
レコードの編集処理
相対レコード番号の設定
WRITE  レコード名  ～.
CLOSE  ファイル名.
```

指定した相対レコード番号のレコードがすでに存在するとき、無効キー条件が成立します。無効キー条件については、入出力エラー処理を参照してください。

［備考］：相対レコード番号は、RELATIVE KEY 句に指定したデータ名に設定します。

```
例:        MOVE  1  TO  データ名.
```

NCOBU00263

索引ファイルの使い方

ここでは、索引ファイルの設計方法および処理方法について説明します。

索引ファイルを使用する COBOL プログラム

```
IDENTIFICATION DIVISION.
PROGRAM-ID. プログラム名.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT ファイル名
         ASSIGN TO ファイル参照子
         ORGANIZATION IS INDEXED
        [ACCESS MODE IS アクセス形態]
         RECORD KEY IS 主キー名１
        [主キー名ｎ]…[WITH DUPLICATES]
        [[ALTERNATE RECORD KEY IS 副キー名１
        [副キー名ｎ]…[WITH DUPLICATES]]…].
DATA DIVISION.
 FILE SECTION.
 FD ファイル名
   [RECORD レコードの大きさ].
01 レコード名.
  02 主キー名１ ～.
  [02 主キー名ｎ ～.]
  [02 副キー名１ ～.]
  [02 副キー名ｎ ～.]
  [02 キー以外のデータ ～.]
PROCEDURE DIVISION.
   OPEN オープンモード ファイル名.
   [MOVE 主キーの値 TO 主キー名ｎ.]
   [MOVE 副キーの値 TO 副キー名ｎ]
   [READ ファイル名.]
   [REWRITE レコード名.]
   [DELETE ファイル名.]
   [START ファイル名.]
   [WRITE レコード名.]
   CLOSE ファイル名.
END PROGRAM プログラム名.
```

このセクションの内容

索引ファイルの定義方法

ここでは、索引ファイルを使うときに必要なファイルの定義について説明します。

ファイル名とファイル参照子

索引ファイルでは、レコード順ファイルと同様に、ファイル名とファイル参照子を指定します。指定方法については、レコード順ファイルの定義方法を参照してください。

ファイル編成

ORGANIZATION 句に INDEXED を指定します。

アクセス形態

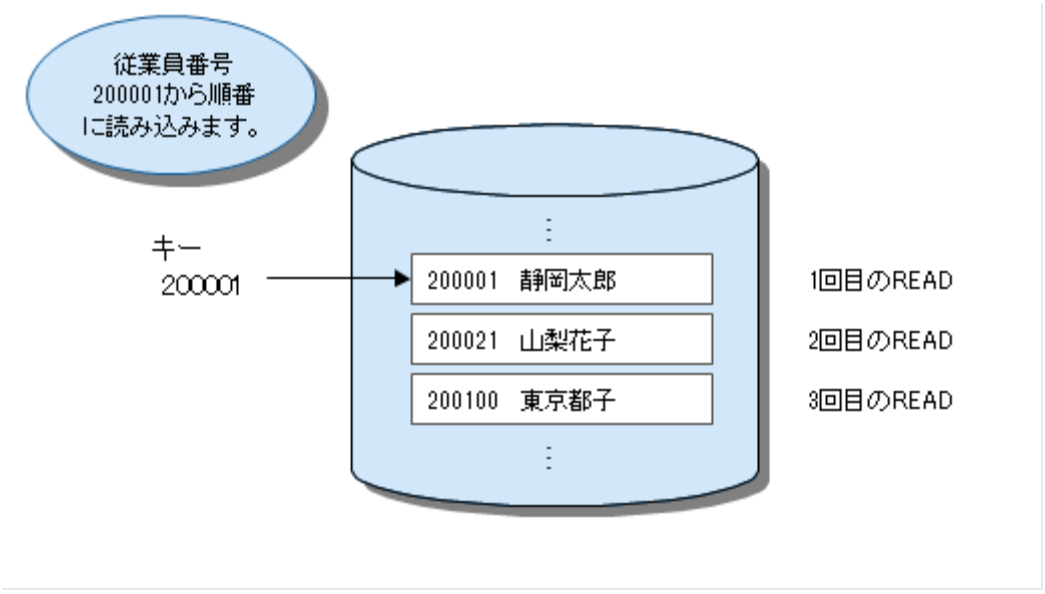
ACCESS MODE 句に以下のアクセス形態のどれかを指定します。

順呼出し法 (SEQUENTIAL)	ファイルの先頭またはある特定の値のキーを持つレコードから、キーの昇順にレコードを処理することができる、順呼出しの処理を行うことができます。
乱呼出し法 (RANDOM)	ある特定の値のキーを持つレコードだけを単独に処理することができる、乱呼出しの処理を行うことができます。
動的呼出し法 (DYNAMIC)	順呼出しと乱呼出しの両方の処理を行うことができます。

以下に、レコードの参照処理を例に、順呼出しの場合と乱呼出しの場合での処理の違いを示します。

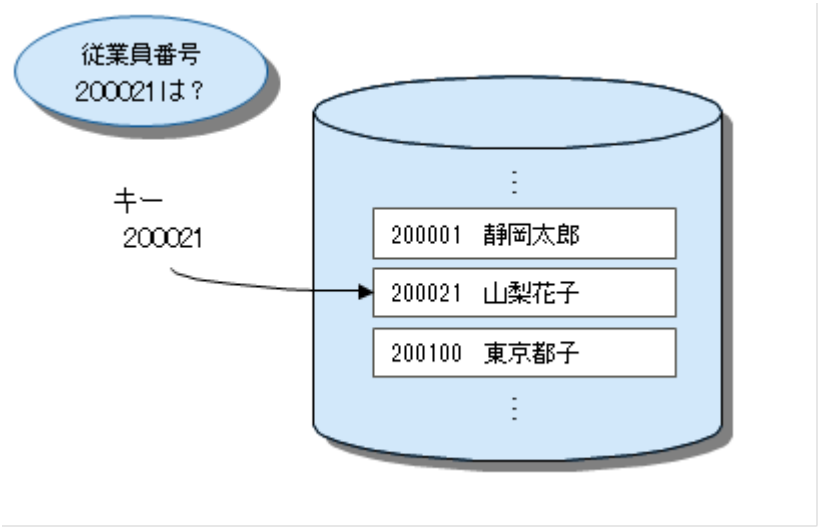
順呼出し

〔従業員番号の先頭が 2 に該当する人のデータを順番に取り出す場合〕



乱呼出し

〔ある従業員番号に該当する人のデータを取り出す場合〕



主キーと副キー

キーには主レコードキー(主キー)と副レコードキー(副キー)があり、キーの個数やレコード中での位置および大きさはファイル創成時に決定され、以後変更することはできません。

ファイル中のレコードは、論理的に主キーの値の昇順に並んでいて、主キーの値によって特定のレコードを選択することができます。

索引ファイルの定義では、必ず主キーとなるデータ項目の名前を RECORD KEY 句に指定します。副キーは、主キーと同様にファイル中の特定のレコードを選択するための情報となります。副キーとなるデータ項目の名前は、必要に応じて ALTERNATE RECORD KEY 句に指定します。

RECORD KEY 句および ALTERNATE RECORD KEY 句には、複数のデータ項目を指定することができます。RECORD KEY 句に複数のデータ項目を指定した場合、それらのデータ項目をつなげたものが主キーとなります。RECORD KEY 句に指定するデータ項目は、連続している必要はありません。RECORD KEY 句および ALTERNATE RECORD KEY 句に DUPLICATES を指定することにより、複数のレコードが同じキーの値を持つこと(キーの値の重複)ができます。DUPLICATES が指定されていない場合、キーの値が重複するとエラーとなります。

索引ファイルのレコードの定義方法

ここでは、レコード形式とレコード長およびレコードの構成について説明します。

レコード形式とレコード長

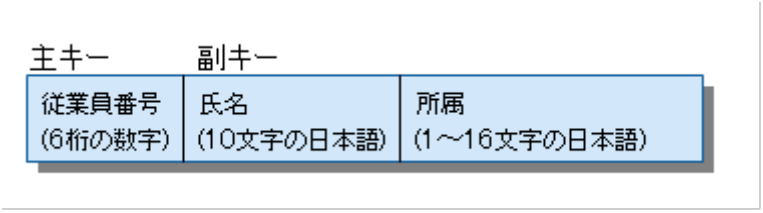
索引ファイルのレコード形式とレコード長の説明は、レコード順ファイルの説明と同じです。レコード順ファイルのレコードの定義方法を参照してください。

レコードの構成

レコード中のキーおよびキー以外のデータの属性、位置および大きさは、レコード記述項で定義します。キーの定義については、以下の点に注意する必要があります。

- 既存のファイルを処理する場合、主キーまたは副キーに指定する項目の数および位置と大きさは、ファイルが創成されたときとすべて同じでなければなりません。また、主キー、副キーの指定順序および個数が一致していなければなりません。
- 1 つのファイルに対してレコード記述項を 2 つ以上記述する場合、主キーとなるデータ項目は、これらのレコード記述項のうち 1 つにだけ記述します。そのレコード記述項以外のレコード記述項でも、主キーを定義した文字位置および大きさが主キーとして使用されます。
- 可変長レコード形式の場合、キーの位置は固定部(レコードの先頭からの位置がつねに一定のところ)になければなりません。

〔可変長レコード形式のレコードの定義例〕




```
*          :
           RECORD KEY IS 従業員番号
           ALTERNATE RECORD KEY IS 氏名.
*          :
  FD 従業員ファイル
     RECORD IS VARYING IN SIZE FROM 27 TO 58 CHARACTERS
     DEPENDING ON レコード長.
*          :
 01 従業員レコード.
   02 従業員番号       PIC 9(6).
   02 氏名             PIC N(10).
   02 所属.
     03 PIC N OCCURS 1 TO 16 TIMES
            DEPENDING ON 所属の長さ.
*          :
 WORKING-STORAGE SECTION.
 01 レコード長         PIC 9(3) COMP-5.
 01 所属の長さ         PIC 9(3) COMP-5.
```

NCOBU00266

索引ファイルの処理

索引ファイルの処理では、入出力文を使って、創成、拡張、挿入、参照、更新、削除を行うことができます。ただし、これらの処理はアクセス形態によっては使用不可能な場合があります。ここでは、索引ファイルの処理で使用する入出力文の種類と使い方およびそれぞれの処理の概要について説明します。

入出力文の種類

- OPEN 文
- CLOSE 文
- DELETE 文
- READ 文
- START 文
- REWRITE 文
- WRITE 文

入出力文の使い方

OPEN 文および CLOSE 文

OPEN 文はファイル処理の最初に、CLOSE 文はファイル処理の最後にそれぞれ 1 回だけ必ず実行します。

OPEN 文に指定するオープンモードは、ファイルに対してどのような処理を行うかによって決定されます。たとえば、創成処理を行う場合には、出力モード(OUTPUT 指定)の OPEN 文を実行します。

その他の文

DELETE 文は、ファイル中のレコードを削除するときに使用します。

READ 文は、ファイル中のレコードを読み込むときに使用します。

START 文は、処理を開始するレコードを指示するときに指定します。

REWRITE 文は、ファイル中のレコードを更新するときに使用します。

WRITE 文は、ファイルにレコードを書き出すときに使用します。

処理の概要

創成〔順/乱/動的〕

索引ファイルを創成するには、ファイルを出力モードで開いて、WRITE 文でファイルにレコードを書き出していきます。

```
OPEN OUTPUT  ファイル名.
レコードの編集処理
MOVE 主キー値 TO 主キー名.
WRITE  レコード名  ～.
CLOSE  ファイル名.
```

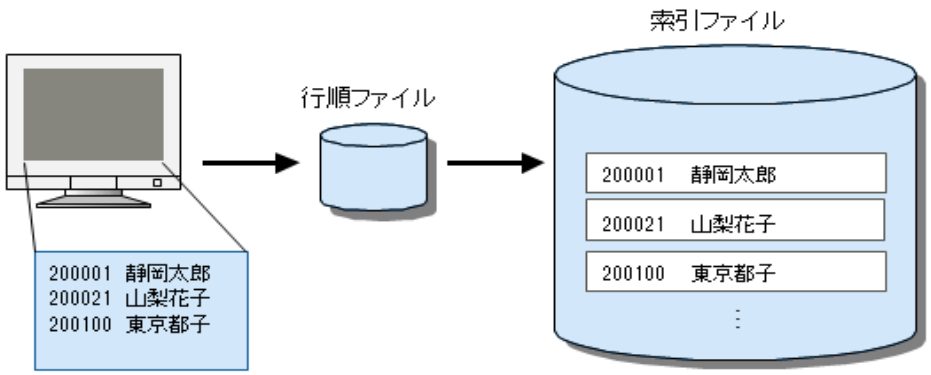
注意

- すでに存在するファイルに対して創成処理を行った場合、そのファイルは新しく作り直され、元の内容は失われます。
- レコードを書き出すとき、主キーの値を設定する必要があります。また、順呼出しの場合、主キーの内容が昇順になるように書き出す必要があります。

参考

索引ファイルを作成するときのデータの収集方法として次の機能を使うと便利です。

- エディターでデータを作成し、行順ファイル機能を使ってそのデータを読込み、索引ファイルに書き出します。



拡張〔順〕

索引ファイルの拡張は、ファイルを拡張モードで開いて、順番にレコードを書き出します。このとき、ファイルの最後のレコードの後ろにレコードが追加されます。ファイルの拡張が可能なアクセス形態は、順呼出し法だけです。

```
OPEN EXTEND  ファイル名.
レコードの編集処理
WRITE  レコード名  ～.
CLOSE  ファイル名.
```

注意

書き出すレコードの内容を編集するときに、主キーの値が昇順となるように設定する必要があります。 また、ファイル管理記述項の RECORD KEY 句に DUPLICATES 指定がない場合、最初に処理する主キーの値は、そのファイルに存在する最大の主キーの値より大きくなければなりません。DUPLICATES

指定がある場合、そのファイルに存在する最大の主キーの値と等しいか、それより大きくなければなりません。

参照〔順/乱/動的〕

レコードの参照では、ファイルを入力モードで開いて、READ 文でファイルのレコードを読み込みます。順呼出しの場合、START 文を使って読み込むレコードの開始位置を指定し、そのレコードから主キーまたは副キーの値で昇順に読み込んでいきます。乱呼出しの場合、読み込まれるレコードは、主キーまたは副キーの値により決定されます。

```
【順呼出し】
        OPEN INPUT  ファイル名.
        MOVE   キーの値 TO キー名.
        START  ファイル名.
        READ   ファイル名 [NEXT] ～.
        CLOSE  ファイル名.
```

```
【乱呼出し】
        OPEN INPUT  ファイル名.
        MOVE   キーの値 TO キー名.
        READ   ファイル名  ～.
        CLOSE  ファイル名.
```


注意

- ファイル管理記述項の SELECT 句に OPTIONAL を指定した場合、OPEN 文実行時にファイルが存在していなくても OPEN 文は成功となり、最初の READ 文の実行でファイル終了条件が成立します。 ファイル終了条件については、入出力エラー処理を参照してください。
- 乱呼出し法または動的呼出し法で、指定したキーの値を持つレコードが存在しないときに無効キー条件が成立します。 無効キー条件については、入出力エラー処理を参照してください。
- 複数のキーを指定して START 文または READ 文を実行することもできます。

更新〔順/乱/動的〕

レコードの更新では、ファイルを入出力モードで開いて、ファイルのレコードを書き換えます。順呼出しの場合、直前の READ 文により読み込まれたレコードの内容が変更されます。乱呼出しの場合、設定したキー値を持つ主キーのレコードの内容が変更されます。

```
【順呼出し】
        OPEN I-O  ファイル名.
        MOVE   キーの値 TO キー名.
        START  ファイル名.
        READ   ファイル名 [NEXT] ～.
        レコードの編集処理
        REWRITE  レコード名  ～.
        CLOSE  ファイル名.
```

```
【乱呼出し】
        OPEN I-O  ファイル名.
        MOVE   キーの値 TO キー名.
       [READ   ファイル名  ～.]
        レコードの編集処理
        REWRITE  レコード名  ～.
        CLOSE  ファイル名.
```

- 乱呼出し法または動的呼出し法で、指定したキーの値を持つレコードが存在しないときに無効キー条件が成立します。 無効キー条件については、入出力エラー処理を参照してください。
- 副キーの内容を変更することはできますが、主キーの内容を変更することはできません。

削除〔順/乱/動的〕

レコードの削除では、ファイルを入出力モードで開いて、ファイルのレコードを削除します。 順呼出しの場合、直前の READ 文により読み込まれたレコードが削除されます。 乱呼出しの場合、設定したキー値を持つ主キーのレコードが削除されます。

```
【順呼出し】
        OPEN I-O  ファイル名.
        MOVE   キーの値 TO キー名.
        START  ファイル名.
        READ   ファイル名 [NEXT] ～.
        DELETE ファイル名  ～.
        CLOSE  ファイル名.
```

```
【乱呼出し】
        OPEN I-O   ファイル名.
        MOVE   キーの値 TO キー名.
        DELETE ファイル名  ～.
        CLOSE  ファイル名.
```

乱呼出し法または動的呼出し法で、指定したキーの値を持つレコードが存在しないときに無効キー条件が成立します。 無効キー条件については、入出力エラー処理を参照してください。

挿入〔乱/動的〕

レコードの挿入では、ファイルを入出力モードで開いて、ファイルにレコードを挿入します。 レコードの挿入される位置は、主キーの値によって決定されます。

```
OPEN I-O  ファイル名.
レコードの編集処理
MOVE   キーの値 TO キー名.
WRITE  レコード名  ～.
CLOSE  ファイル名.
```

ファイル管理記述項の RECORD KEY 句または ALTERNATE RECORD KEY 句に DUPLICATES 指定がない場合、指定したキーの値を持つレコードがすでに存在するとき、無効キー条件が成立します。 無効キー条件については、入出力エラー処理を参照してください。

入出力エラー処理

ここでは、入出力エラーの検出方法および入出力エラーが発生したときの実行結果について説明します。

入出力エラーの検出方法には、次の 4 つがあります。

- AT END 指定(ファイル終了条件発生の検出)
- INVALID KEY 指定(無効キー条件発生の検出)
- FILE STATUS 句(入出力状態のチェックによる入出力エラーの検出)
- 誤り処理手続き(入出力エラーの検出)

また、入出力エラーが発生したときの実行結果についても説明します。

NCOBU00268

AT END 指定

ファイル中のレコードを順番に読み、すべてのレコードを読み終わったときに、次に読み込むレコードが存在しないとファイル終了状態になります。これを、ファイル終了条件の発生といいます。ファイル終了条件の発生を検出するには、READ 文に AT END 指定を記述します。AT END 指定には、ファイル終了条件が発生したときに行う処理を記述することができます。

以下に、AT END 指定の READ 文の記述例を示します。

```
READ 順ファイル
    AT END GO TO ファイルの終了処理
END-READ.
```

ファイルの最後のレコードを読んだ次の READ 文の実行で、ファイル終了条件が発生し、GO TO 文が実行されます。

INVALID KEY 指定

索引ファイルまたは相対ファイルの入出力処理で、指定したキーや相対レコード番号を持つレコードが存在しなかった場合、入出力エラーとなります。これを、無効キー条件の発生といいます。無効キー条件の発生を検出するには、READ 文、WRITE 文、REWRITE 文、START 文および DELETE 文に INVALID KEY 指定を記述します。INVALID KEY 指定には、無効キー条件が発生したときに行う処理を記述することができます。

以下に、INVALID KEY 指定の READ 文の記述例を示します。

```
MOVE "Z" TO 主キー.
READ 索引ファイル
   INVALID KEY GO TO 無効キーの処理
END-READ.
```

主キーの値に”Z”を持つレコードが存在しないとき、無効キー条件が発生し、GO TO 文が実行されます。

NCOBU00270

FILE STATUS 句

ファイル管理記述項に FILE STATUS 句を記述すると、入出力文実行時に、FILE STATUS 句に指定したデータ名に入出力状態が通知されます。入出力文の後に、このデータ名の内容(入出力状態値)をチェックする文(IF 文または EVALUATE 文)を記述することによって、プログラムで入出力文の結果に応じた処理手続きを実行することができます。ただし、入出力文の後に入出力状態値をチェックしない場合、入出力エラーが発生してもプログラムの後続処理の実行が続けられる場合があるため、その後の動作は保証されません。

通知される入出力状態値については、ファイル入出力状態一覧を参照してください。入出力状態の成功の分類には、入出力文の実行は成功していても、その入出力の結果について何らかの情報が通知されるものが含まれます。

以下に、FILE STATUS 句の記述例を示します。

```
      SELECT ファイル ASSIGN TO SYS001
           FILE STATUS IS  入出力状態値
WORKING-STORAGE SECTION.
 01  入出力状態値  PIC X(2).
PROCEDURE DIVISION.
   OPEN INPUT ファイル.
   IF 入出力状態値 NOT = "00" THEN
      GO TO オープン失敗の処理
   END-IF
```

ファイルのオープンに失敗すると、入出力状態値に"00"以外の値が設定されます。IF 文でこの値をチェックしているので、オープンに失敗した場合には GO TO 文が実行されます。

誤り処理手続き

手続き部の宣言節部分で、USE AFTER ERROR/EXCEPTION 文を記述することにより、誤り処理手続きを指定することができます。誤り処理手続きを記述すると、入出力エラー発生時に誤り処理手続きに記述した処理が実行されます。誤り処理を実行後、入出力エラーが発生した入出力文の直後の文に制御が渡るので、入出力文の直後には、入出力エラーが発生したファイルの処理の制御を指示する文を記述する必要があります。ここでの入出力エラーとは、入出力状態の成功の分類に含まれるもの以外の条件が発生したことを示します。

以下の場合には、誤り処理手続きに制御は渡りません。

- ファイル終了条件が発生した READ 文に AT END 指定がある場合
- 無効キー条件が発生した入出力文に INVALID KEY 指定がある場合
- ファイルが開かれる前に入出力文を実行した場合(オープンモードの指定をした場合)

以下の場合、誤り処理手続き中から手続き中へ GO TO 文を使って分岐してください。

- 誤り処理手続きの中で、入出力エラーが発生したファイルに対して入出力文を実行した場合
- 誤り処理手続きが終了する前に、その誤り処理手続きが再び実行された場合

```
PROCEDURE DIVISION.
 DECLARATIVES.
  誤り処理 SECTION.
    USE AFTER ERROR PROCEDURE ON ファイル.
    MOVE エラー発生 TO ファイルの状態.          *>[1]
 END DECLARATIVES.
    OPEN INPUT ファイル.
    IF ファイルの状態 = エラー発生 THEN         *>[2]
      GO TO  オープン失敗の処理
    END-IF
```

ファイルのオープンに失敗すると、誤り処理手続き([1]の MOVE 文)が実行され、OPEN 文の直後の文([2]の IF 文)に制御が渡ります。

NCOBU00272

入出力エラーが発生したときの実行結果

入出力エラーが発生したときの実行結果は、AT END 指定の有無、INVALID KEY 指定の有無、FILE STATUS 句の有無および誤り処理手続きの有無によって異なります。入出力エラーが発生したときの実行結果を以下の表に示します。

ここでの入出力エラーとは、入出力状態の成功の分類に含まれるもの以外の条件が発生したことを示します。

表:入出力エラーが発生したときの実行結果

エラータイプ		誤り処理手続きあり		誤り処理手続きなし	
		FILE STATUS 句あり	FILE STATUS 句なし	FILE STATUS 句あり	FILE STATUS 句なし
AT END 指定 または INVALID KEY 指定	あり	AT END 指定／INVALID KEY 指定に記述した文が実行されます。			
	なし	誤り処理手続きが実行された後、エラーが発生した入出力文の直後の文から処理が続行されます。		エラーが発生した入出力文の直後の文から処理が続行されます。	Ｕレベルメッセージが出力されて、プログラムが異常終了します。
その他の入出力エラー発生時の処理		誤り処理手続きが実行された後、エラーが発生した入出力文の直後の文から処理が続行されます。		Ｉレベルメッセージが出力されて、エラーが発生した入出力文の直後の文から処理が続行されます。	Ｕレベルメッセージが出力されて、プログラムが異常終了します。

有効な誤り処理手続きがある場合、環境変数情報@CBR_FILE_USE_MESSAGE=YES を指定すると、Ｉレベルメッセージが出力されます。

```
@CBR_FILE_USE_MESSAGE=YES
```

索引ファイルの復旧関数

NetCOBOL for .NET では、索引ファイルの復旧関数を提供しています。

索引ファイルの復旧関数には、以下の 2 種類があります。

- 索引ファイル復旧関数 (CFURCOV)
- 索引ファイル簡易復旧関数 (CFURCOVS)

ここでは、以下についても説明します。

- COBOL から復旧関数を呼び出す場合のプログラム例
- メッセージの内容とコード

索引ファイルの復旧関数を使用する場合の注意事項

- 索引ファイル簡易復旧関数(CFURCOVS)のプログラム原型定義には、以下の DLL ファイル名を指定してください。

対象プラットフォーム	ファイル名
x86	f5fhfutc.dll
x64	f4fhfutc.dll

- C プログラムから使用する場合、 以下のファイルをインクルードしてください。

対象プラットフォーム	ファイル名
x86	f5fhfutc.h
x64	f4fhfutc.h

- LINK コマンドを使用してファイルをリンクする場合は、以下のファイルをリンクしてください。

対象プラットフォーム	ファイル名
x86	f5fhfutc.lib
x64	f4fhfutc.lib

これらのファイルは、NetCOBOL for .NET ランタイムシステムのインストールフォルダの“Utilities”サブフォルダに格納されています。既定の設定では、以下の場所に格納されます。(Windows が C:ドライブにインストールされているとします。)

Windows の種類	ランタイムシステムの種類	ファイルの格納場所
32-bit	32-bit	C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥
64-bit	32-bit	C:¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥
	64-bit	C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥

- 実行する際、環境変数 PATH に上記フォルダ名を設定してください。 ただし、COBOL プログラムから索引ファイル復旧関数(CFURCOV)を呼び出す場合は、設定不要です。

索引ファイル復旧関数 (CFURCOV)

索引ファイル復旧関数は、正常に閉じることができなかったために使用できない状態にある索引ファイルを使用可能にするため、ファイルの先頭から正常な部分を再生し、異常部分を別ファイルに出力します。つまり、COBOLファイルユーティリティの[復旧]コマンドと同じ機能を持っています。

再度使用できない状態にある索引ファイルを開いた場合は、入出力状態に"39"または"90"が返却されます。

記述形式

【COBOL 言語からの呼出し】

```
CALL "CFURCOV" USING BY REFERENCE IXDFILE-NAME
                     BY REFERENCE BLKFILE-NAME
                     BY REFERENCE MESSAGE-AREA
               RETURNING RET-CODE
```

【C 言語からの呼出し】

```
#include "f5fhfutc.h" /*  関数宣言  */
 signed long int CFURCOV( char far*ixdfilename,
          char far *blkdatname,
          char far *message);
```

パラメタの説明

IXDFILE-NAME および ixdfilename

復旧する索引ファイルの名前(文字列)が格納されている領域のアドレスを指定します（COBOL 言語の場合、BY REFERENCE 指定)。領域は 1024 バイト確保してください。文字列の終端位置には、NULL 文字(0x00)または空白文字(0x20)を設定する必要があります。また、ファイル名に空白またはコンマ(,)を含む場合は、ファイル名を二重引用符で囲む必要があります。

ファイル名に空白を含む例を以下に示します。

【COBOL 文字列】

```
"""C:¥ixd file """
```

【C 文字列】

```
"¥"C:¥¥ixd file¥"¥0"
```

BLKFILE-NAME および blkdatname

復旧不可能であったレコードを書き込むファイルの名前(文字列)が格納されている領域のアドレスを指定します（COBOL 言語の場合、BY REFERENCE 指定)。領域は 1024 バイト確保してください。文字列の終端位置には、NULL 文字(0x00)または空白文字(0x20)を設定する必要があります。また、ファイル名に空白またはコンマ(,)を含む場合は、ファイル名を二重引用符で囲む必要があります。

すでに存在するファイルの名前を指定した場合、エラーとなります。

データの状態によっては、ファイルに出力されない場合があります。

MESSAGE-AREA および message

索引ファイル復旧関数の実行結果を示すメッセージを格納する領域のアドレスを指定します（COBOL 言語の場合、BY REFERENCE 指定）。格納する領域は、呼出し元で確保(1024 バイト必要)してください。

RET-CODE および 関数の復帰値

索引ファイル復旧関数の結果として、メッセージに対応するコードを返却します。 コードの値とメッセージの内容については、メッセージの内容とコードの “表:索引ファイル復旧関数(簡易復旧関数)が返却するメッセージの内容とコード” を参照してください。

関数の使用例

COBOL 言語の場合

COBOL から呼び出す場合の使用例を参照してください。

C 言語の場合

索引ファイルの復旧関数の使用方法を以下に示します。

```
#include "f5fhfutc.h"
void callcobfrcov(void)
 {
   char ixdfilename[1024] = "C:¥¥ixdfile¥0";
   char blkdatname[1024] = "C:¥¥blkdat¥0";
   char message[1024];
   CFURCOV(ixdfilename,blkdatname,message);
      return;
 }
```

索引ファイル簡易復旧関数 (CFURCOVS)

索引ファイル簡易復旧関数は、再度使用できない状態にある索引ファイルを使用可能にするため、索引ファイル中のフラグをリセットします。 索引ファイル復旧関数とは異なり、索引ファイル中の異常な箇所について修復しないため、復旧後の索引ファイルに対するアクセスで、データの矛盾でエラーになる場合があります。

再度使用できない状態にある索引ファイルを開いた場合は、入出力状態に"39"または"90"が返却されます。

記述形式

【COBOL 言語からの呼出し】

```
CALL CFURCOVS USING BY REFERENCE IXDFILE-NAME
                    BY REFERENCE MESSAGE-AREA
           RETURNING RET-CODE
```

NetCOBOL for .NET のプログラムからこの関数を呼び出す場合、プログラム原型定義を用意する必要があります。

指定する DLL ファイルの名前および格納場所については、索引ファイルの復旧関数の“索引ファイルの復旧関数を使用する場合の注意事項”を参照してください。

【C 言語からの呼出し】

```
#include "f5fhfutc.h" /* 関数宣言 */
 signed long int CFURCOVS( char far*ixdfilename,
        char far *message);
```

パラメタの説明

IXDFILE-NAME および ixdfilename

復旧する索引ファイルの名前(文字列)が格納されている領域のアドレスを指定します（COBOL 言語の場合、BY REFERENCE 指定)。領域は 1024 バイト確保してください。文字列の終端位置には、NULL 文字(0x00)または空白文字(0x20)を設定する必要があります。また、ファイル名に空白またはコンマ(,)を含む場合は、ファイル名を二重引用符で囲む必要があります。

ファイル名に空白を含む場合の指定例は、索引ファイル復旧関数 (CFURCOV)の記述形式を参照してください。

MESSAGE-AREA および message

索引ファイル簡易復旧関数の実行結果を示すメッセージを格納する領域のアドレスを指定します（COBOL 言語の場合、BY REFERENCE 指定)。格納する領域は、呼出し元で確保(512 バイト必要)してください。

RET-CODE および 関数の復帰値

索引ファイル簡易復旧関数の結果として、メッセージに対応するコードを返却します。 コードの値とメッセージの内容については、メッセージの内容とコードの “表:索引ファイル

復旧関数(簡易復旧関数)が返却するメッセージの内容とコード”を参照してください。

関数の使用例

COBOL 言語の場合

CFURCOV で使用する COBOL から呼び出す場合の使用例を参照し、関数名を CFURCOVS に置き換え、BLK-FILENAME パラメタを削除したものが、索引ファイル簡易復旧関数の使用例になります。

C 言語の場合

索引ファイルの簡易復旧関数の使用方法を以下に示します。

```
#include "f5fhfutc.h"
void callcobfrcovs(void)
 {
    char ixdfilename[512] = "c:¥¥ixdfile¥0;"
    char message[512];
    CFURCOVS(ixdfilename,message);
    return;
 }
```

COBOL から復旧関数を呼び出す場合のプログラム例

ここでは、COBOL から復旧関数(CFURCOV)を呼び出す場合のプログラム例を示します。

[補足]: NetCOBOL for .NET V2.0 以前では、復旧関数を呼び出すには、プログラム原型定義が必須でしたが、V2.1 以降ではプログラム原型定義は必須ではありません。通常の CALL 呼出しで呼び出すことができます。もちろん、これまでどおりプログラム原型定義を使って呼び出すこともできます。

復旧できなかったデータおよびメッセージの返却を必要とする場合

索引ファイルの復旧関数の呼出しにおいて、復旧できなかったデータの格納およびメッセージの返却を必要とする場合の例を以下に示します。

BLKFILE-NAME パラメタには、復旧できなかったデータを格納するファイル名を設定し、MESSAGE-AREA パラメタには、1024 バイトの領域を確保して指定します。

【復旧関数(CFURCOV)の呼出し】

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. RECOVER2.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
          ENVIRONMENT-NAME ENV-NAME
          ENVIRONMENT-VALUE ENV-VALUE.
      REPOSITORY.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
          SELECT IXDFILE ASSIGN TO FILE1
          ORGANIZATION IS INDEXED
          RECORD KEY IS IXDFILE-RECKEY
          FILE STATUS IS FSDATA.
      DATA DIVISION.
      FILE SECTION.
      FD IXDFILE.
      01 IXDFILE-REC.
        03 IXDFILE-RECKEY PIC X(8).
        03 IXDFILE-DATA   PIC X(20).
      WORKING-STORAGE SECTION.
      77 IXDFILE-NAME     PIC X(1024).
      77 BLKFILE-NAME     PIC X(1024) VALUE "C:¥BLKNDAT ".
      77 MESSAGE-AREA     PIC X(1024).
      77 RET-CODE        PIC S9(9) COMP-5.
      77 FSDATA         PIC XX.
      77 OPEN-FSDATA     PIC XX.
      PROCEDURE DIVISION.
          OPEN I-O IXDFILE
          MOVE FSDATA TO OPEN-FSDATA
          CLOSE IXDFILE
          EVALUATE OPEN-FSDATA
            WHEN "00"
              GO TO MAIN-PROCESS
            WHEN "39"
            WHEN "90"
      * ファイル名を取り出す
              MOVE ALL SPACE TO IXDFILE-NAME
              DISPLAY "FILE1" UPON ENV-NAME
              ACCEPT IXDFILE-NAME FROM ENV-VALUE
      * 索引ファイル復旧関数を呼び出す
              CALL "CFURCOV" USING BY REFERENCE IXDFILE-NAME
                                   BY REFERENCE BLKFILE-NAME
                                   BY REFERENCE MESSAGE-AREA
                             RETURNING RET-CODE
              EVALUATE TRUE
                WHEN RET-CODE <= 1
                  GO TO MAIN-PROCESS
                WHEN OTHER
                  GO TO PROCESS-END
              END-EVALUATE
            WHEN OTHER
```

```
            GO TO PROCESS-END
          END-EVALUATE.
      MAIN-PROCESS.
         OPEN I-O IXDFILE
         CLOSE IXDFILE.
      PROCESS-END.
         EXIT PROGRAM.
      END PROGRAM RECOVER2.
```

メッセージの内容とコード

ここでは、索引ファイルの復旧で出力されるメッセージの内容とコードについて示します。

表：索引ファイル復旧関数が返却するメッセージの内容とコード

コード(10 進)	メッセージの内容
0	レコードを n 件復旧しました。(注 1)
1	レコードを n 件復旧しました。復旧できないレコードが n 件存在しました。(注 2)
10	復旧すべきファイルが存在しません。
11	復旧すべきファイルが索引ファイルではありません。
12	復旧すべきファイルへのアクセス権がありません。
13	復旧不可データ出力ファイルが既に存在します。(注 2)
14	復旧すべきファイルを他プロセスがアクセスしています。
15	復旧すべきファイルを他プロセスがリカバリしています。
16	ファイルの書き込み領域が不足しました。(注 2)
128	メモリ領域が不足しました。
131	指定された索引ファイル中にレコードが存在しませんでした。(注 2)
132	ファイル情報に矛盾がありました。

注 1：簡易復旧関数ではコードだけが返却されます。

注 2：簡易復旧関数ではコード、メッセージとも返却されません。

NCOBU00279

他のファイルシステムの使用法

NetCOBOL for .NET では、製品に同梱されている COBOL ファイルシステム以外にも、ファイル名に特定の文字列を指定することで、他のファイルシステムを使用することができます。

利用できるファイルシステムを以下に示します。

- Btrieve

  Btrieve は、 Actian 社 のレコードマネージメントシステムです。

  詳細については、Btrieve ファイルを参照してください。

- PowerRDBconnector

  PowerRDBconnector と連携することで、入出力機能を使用してデータベース(Microsoft SQL Server, Oracle)をアクセスすることが可能となります。PowerRDBconnector の使用方法、機能範囲については、"PowerRDBconnector 説明書"を参照してください。

- 外部ファイルハンドラ

  外部ファイルハンドラを利用して、Micro Focus COBOL が公開している FCD 構造を持ったファイルシステムを呼び出すことができます。 外部ファイルハンドラで利用できる入出力機能の範囲は、結合するファイルシステムの仕様に依存します。

  詳細については、外部ファイルハンドラを参照してください。

ここでは、COBOL ファイルシステム、Btrieve および PowerRDBconnector 経由のデータベース のそれぞれのファイルシステムで利用できる入出力機能の範囲を以下の表に示します。

表：各ファイルシステムの機能差

分類	項目		COBOL ファイル システム	Btrieve (注 1)	PowerRDB connector SQL Server(注 1)	PowerRDB connector Oracle(注 1)
ファイル	ファイルの最大サイズ (バイト数)	レコード順ファイル	1G(注 10)	128G	32T	無制限
		レコード順ファイル (BSAM 指定)	システム制限まで	―	―	―
		行順ファイル	1G(注 10)	―	―	―
		行順ファイル (BSAM 指定)	システム制限まで	―	―	―
		相対ファイル	1G	―	―	―
		索引ファイル	約 1.7G (注 10)	128G	32T	無制限
レコード	レコード形式	固定長レコード形式	○	○	○	○
		可変長レコード形式	○	○	○	○

	レコードの最大長(バイト数)	固定長レコード形式	32,760	32,760	8,060	32,760
		可変長レコード形式	32,760	32,760	8,060	32,760
	レコードの最小長(バイト数)	レコード順	1	4	1	1
		行順	0	－	－	－
		相対	1	－	1	
		索引	キーを構成する項目までの大きさ	キーを構成する項目までの大きさ	キーを構成する項目までの大きさ	キーを構成する項目までの大きさ
ファイル管理記述項	SELECT句	OPTIONAL指定	○	○	－ (注2)	－ (注2)
	ASSIGN句	ファイル識別名指定	○	○	○	○
		ファイル識別名定数指定	○	○	○	○
		データ名指定	○	○	○	○
		DISK指定	○	－	－	－
	FILE STATUS句	ファイル状態	○	○ (注3)	○ (注3)	○ (注3)
	LOCK MODE句	AUTOMATIC	○	○	－ (注4)	－ (注4)
		EXCLUSIVE	○	○	－ (注4)	－ (注4)
		MANUAL	○	○	－ (注4)	－ (注4)
	RECORD KEY句	指定できるデータの最大個数	254 (注5)	119	16	32
		指定できるデータ総長の最大(バイト数)	254	255	255	データ・ブロックのサイズに依存
	ALTERNATE RECORD KEY句	指定できるデータの最大個数	254 (注5)	119	16	32
		指定できるデータ総長の最大(バイト数)	254	255	255	データ・ブロックのサイズに依存
	レコードキーの項目	英数字	○	○	○	○
		日本語	○	○	○	○

		符号なし外部 10 進数	○	○ (注 6)	○	○
		符号付き外部 10 進数	－	○ (注 6)	○	○
		符号なし内部 10 進数	○	○	○	○
		符号付き内部 10 進数	－	○	○	○
		符号なし 2 進数 (正順格納形式)	○	○	○	○
		符号付き 2 進数 (正順格納形式)	－	○	○	○
		符号なし 2 進数 (逆順格納形式)	－	○	○	○
		符号付き 2 進数 (逆順格納形式)	－	○	○	○
文	READ 文	WITH LOCK 指定	○	○	－	－
	START 文	キー全体を指定	○	○	○	○
		キーの一部を指定	○	○	－	－
	DELETE 文	キー値が重複しているレコードを削除	○	○(注 7)	○	○
	UNLOCK 文		○	○	－	－
機能	スレッド	シングルスレッド	○	○	○	○
		マルチスレッド	○	○	○	○
	トランザクション管理	トランザクション開始指示	－	○ (注 8)	○ (注 8)	○ (注 8)
		COMMIT 指示	－	○ (注 8)	○ (注 8)	○ (注 8)
		ROLLBACK 指示	－	○ (注 8)	○ (注 8)	○ (注 8)
	リカバリ		○ (注 9)	○ (注 9)	○ (注 9)	○ (注 9)

○: サポート

－: 未サポート

注 1 : Btrieve については、 " Actian PSQL V11 "の情報をもとにしています。 PowerRDBconnector については、“PowerRDBconnector”の情報をもとにしています。

注 2 : PowerRDBconnector では、ファイルが存在しない場合、ファイルなしのエラーとなります。自動生成は行いません。

注 3:Btrieve および PowerRDBconnector は、FILE STATUS=02 を返却しません。また、PowerRDBconnector の場合、排他エラー(ファイル排他/レコード排他とも)を示す FILE STATUS 値は"92"です。

注 4：エラーとなりませんが、PowerRDBconnector では、LOCK MODE 句で指定したロック機構は動作しません。

注 5: COBOL ファイルシステムでは、RECORD KEY 句と ALTERNATE RECORD KEY 句に指定できるデータ個数の総和が最大 255 です。

注 6: Btrieve ファイルの NUMERIC タイプは、COBOL では SEPARATE 指定のない符号付き外部 10 進項目のデータ型に相当しますが、NUMERIC の内部形式は 88 コンソーシアム形式と呼ばれる内部形式になっており、NetCOBOL for .NET で扱う内部形式とは異なります。そのため、Btrieve ファイルを使用して、SEPARATE 指定のない符号付き外部 10 進項目を入力または出力項目として扱う場合、書込み前および読込み後にデータを変換する必要があります。データ変換の詳細については、Btrieve ファイルの「外部 10 進項目のデータ形式変換」を参照してください。

注 7: Btrieve では、ファイル位置指示子が確定しているレコードを乱呼出し法の DELETE 文で削除した場合、直前までのファイル位置指示子は不定になります。また、副キーで順呼出し法の READ 文を実行し、重複している副キーの先頭から 2 番目以降のレコードを読み込んだ後、以下のどれかを実行した場合も、ファイル位置指示子は不定になります。

- 乱呼出し法の REWRITE 文
- 乱呼出し法の DELETE 文
- 乱呼出し法の WRITE 文

注 8:Btrieve および PowerRDBconnector に用意されている機能(関数)を、CALL 文で呼び出す必要があります。Btrieve でトランザクション操作を行うためには、 別途 AG-TECH 社から発売されているプログラムサポートを購入する必要があります。 PowerRDBconnector でトランザクション操作を行う方法については、“PowerRDBconnector 説明書”を参照してください。

注 9：COBOL ファイルシステムでは、アクセスできなくなった索引ファイルを復旧する機能を提供しています。Btrieve では、 トランザクション管理の機能と合わせたリカバリ機構や各種ツールが提供されています。PowerRDBconnector 経由でアクセスするデータベース（SQL Server、Oracle）で、トランザクション管理の機能と合わせたリカバリ機構や各種ツールが提供されています。

注 10: 実行環境変数@CBR_FILE_LFS_ACCESS=YES を指定した場合、システム制限まで拡張することができます。

NCOBU00280

Btrieve ファイル

Btrieve ファイルは、順ファイルおよび索引ファイルとして利用できます。Btrieve で利用できる機能については、他のファイルシステムの使用法の"表：各ファイルシステムの機能差"を参照してください。

ここでは、Btrieve ファイルの使用方法について説明します。

ファイル環境の指定

COBOL ファイルシステムを使用するか、Btrieve ファイルシステムを使用するかは、ファイル管理記述項の ASSIGN 句の記述によって決まります。

ASSIGN 句にファイル識別名定数を記述した場合

ファイル識別名定数は、以下に示す形式で指定します。

ファイル名	入出力対象となるファイルのファイル名を指定します。
指定なし	COBOL ファイルシステムを使用する場合には何も指定しません。
BTRV	Btrieve レコードマネージャーを使用します。

ASSIGN 句にデータ名を記述した場合

データ名に設定するファイル名は、ファイル識別名定数の場合と同じ形式で指定します。

ASSIGN 句に DISK を記述した場合

Btrieve レコードマネージャーを使用することはできません。COBOL ファイルシステムが使用されます。

ASSIGN 句にファイル識別名を記述した場合

ファイル識別名を環境変数情報名とした環境変数に、ファイル識別名定数の場合と同じ形式でファイル名を指定します。

環境変数の指定形式

外部 10 進項目のデータ形式変換

SEPARATE 指定のない符号付き外部 10 進項目を扱うために、内部形式を変換する 2 種類の方法を用意しています。

- NetCOBOL for .NET で扱う形式から Btrieve ファイルで扱う形式(88 コンソーシアム形式)に変換する方法
- 88 コンソーシアム形式から NetCOBOL for .NET で扱う形式に変換する方法

このデータ変換は、Btrieve ファイルを使用して SEPARATE 指定のない符号付き外部 10 進項目を入力または出力項目として扱う場合、書込み前および読込み後に行う必要があります。

Btrieve の NUMERIC タイプのデータの内部形式については、"Pervasive PSQL" のマニュアルをお持ちの方は、そちらを参照してください。お持ちでない方は、技術員(SE)に連絡してください。

記述形式

- 88 コンソーシアム形式から NetCOBOL for .NET の内部形式への変換

```
CALL "#DEC88TOFJ" USING [BY REFERENCE] 一意名
```

- NetCOBOL for .NET の内部形式から 88 コンソーシアム形式への変換

```
CALL "#DECFJTO88" USING [BY REFERENCE] 一意名
```

機能概要

- CALL 文で呼ぶプログラム名が”#DEC88TOFJ”の場合には、一意名または一意名中に含まれるすべての SEPARATE 指定のない符号付き外部 10 進項目の内部形式を、88 コンソーシアム形式から NetCOBOL for .NET の内部形式に変換します。
- CALL 文で呼ぶプログラム名が”#DECFJTO88”の場合には、一意名または一意名中に含まれるすべての SEPARATE 指定のない符号付き外部 10 進項目の内部形式を、NetCOBOL for .NET の内部形式から 88 コンソーシアム形式に変換します。
- 一意名が基本項目の場合、その項目が SEPARATE 指定のない符号付き外部 10 進項目でなければ、変換は行われません。
- 一意名が集団項目で、かつ従属する項目に以下の項目が含まれている場合、その項目は変換の対象とはなりません。
  - SEPARATE 指定のない符号付き外部 10 進項目ではない項目
  - REDEFINES 句が指定された項目、およびその項目に従属する項目
  - RENAMES 句が指定された項目
- 以下の場合、実行結果は保証されません。
  - CALL 文で呼ぶプログラム名が一意名で指定されており、その一意名により"#DEC88TOFJ"または"#DECFJTO88"が指定されている。
  - 上記の記述形式とは異なる形式で、"#DEC88TOFJ"または"#DECFJTO88"を呼び出している(USING 句に複数の一意名が指定されている、BY CONTENT 句が指定されているなど)。
  - 一意名が集団項目であり、従属する項目に DEPENDING 指定付きの OCCURS 句を持つ項目が存在する。

- 一意名が定数節に定義されたデータ項目である。
- 一意名が部分参照付けされたデータ項目である。

使用上の注意

- 本関数の呼出しは、レコード(キーデータ項目)を Btrieve ファイルに渡す直前または受け取った直後に行ってください。Btrieve ファイルから入力した直後や本関数で変換済みの 88 コンソーシアム形式の外部 10 進項目を COBOL で扱うことはできません。 なお、「COBOL で扱うことができない」とは、演算したり比較したり、または、外部 10 進項目として転記した場合の結果が保証されないことを意味します。88 コンソーシアム形式の外部 10 進項目を含むレコードを他のレコードへ集団項目転記した場合は、転記されたレコード内で 88 コンソーシアム形式の外部 10 進項目としての値が保証されます。
- Btrieve ファイルに対しては、整列併合機能は利用できません。
- 本関数呼出しを使用して、ビルド時に"#DECFJTO88"または”#DEC88TOFJ”の外部参照エラーとなった場合は、呼び出す関数名が正しくないか、本関数の使い方を機能概要で示した規則に違反して使用している場合です。

使用例

```
FILE-CONTROL.
    SELECT ファイル    ASSIGN TO  "ファイル名,BTRV"
                    ORGANIZATION IS INDEXED
                    ACCESS MODE SEQUENTIAL
                    RECORD KEY IS 主キー OF レコード.
DATA DIVISION.
FILE SECTION.
FD  ファイル.
01  レコード.
  02  主キー       PIC 9(4).
  02  データ１     PIC S9(8).
  02  データ２     PIC X(50).
WORKING-STORAGE SECTION.
01  作業域.
  02  主キー       PIC 9(4).
  02  データ１     PIC S9(8).
  02  データ２     PIC X(50).
PROCEDURE DIVISION.
*Btrieve ファイルへの書込み
    OPEN  OUTPUT ファイル.
    CALL  "#DECFJTO88" USING 作業域.                    *> [1]
    WRITE  レコード FROM 作業域.
    CLOSE  ファイル.
*Btrieve ファイルからの読込み
    OPEN INPUT   ファイル.
    MOVE 6  TO   主キー OF レコード.
    START  ファイル.
    READ   ファイル INTO 作業域.
    CALL "#DEC88TOFJ" USING  作業域.                    *> [2]
    DISPLAY   "データ１：" データ１ OF 作業域.
    CLOSE  ファイル.
```

[プログラムの説明]

[1]: Btrieve ファイルへの書き込みの前に、書き込まれるデータとなる"作業域"内のすべての SEPARATE 指定のない符号付き外部 10 進項目を、NetCOBOL for .NET の内部形式から 88 コンソーシアム形式に変換します。

[2]: Btrieve ファイルからのレコード読込み後、読込み先である"作業域"内のすべての

SEPARATE 指定のない符号付き外部 10 進項目を、88 コンソーシアム形式から NetCOBOL for .NET の内部形式に変換します。

LOCK MODE 句に AUTOMATIC を指定した場合、ロックされているレコード以外のレコードに対して、WRITE 文/REWRITE 文/DELETE 文/START 文を実行しても、ロックは解除されません。ただし、ロックされているレコードに対して、REWRITE 文/DELETE 文を実行した場合、ロックは解除されます。

NCOBU00281

PowerRDBconnector

PowerRDBconnector は、 順ファイル、相対ファイルおよび索引ファイルとして利用できます。RDM で利用できる機能については、他のファイルシステムの使用法の"表：各ファイルシステムの機能差"を参照してください。

ここでは、PowerRDBconnector の使用方法について説明します。

［備考］

PowerRDBconnector の文字コードの取り扱いや各種固有機能の詳細な使用方法については、“PowerRDBconnector” のマニュアルを 参照してください。

ファイル環境の指定

COBOL ファイルシステムを使用するか、 PowerRDBconnector を使用するかは、 ファイル管理記述項の ASSIGN 句の記述によって決まります。

ASSIGN 句にファイル識別名定数を記述した場合

ファイル識別名定数は、以下に示す形式で指定します。

```
ASSIGN TO "[パス名]ファイル名{指定なし|,RDM}"
```

ファイル名	入出力対象となるファイルのファイル名を指定します。
指定なし	COBOL ファイルシステムを使用する場合には何も指定しません。
RDM	PowerRDBconnector を使用します。

ASSIGN 句にデータ名を記述した場合

データ名に設定するファイル名は、ファイル識別名定数の場合と同じ形式で指定します。

```
MOVE "[パス名]ファイル名{指定なし|,RDM}" TO データ名
```

ASSIGN 句に DISK を記述した場合

PowerRDBconnector を使用することはできません。COBOL ファイルシステムが使用されます。

ASSIGN 句にファイル識別名を記述した場合

ファイル識別名を環境変数情報名とした環境変数には、ファイル識別名定数の場合と同じ形式でファイル名を指定します。

環境変数の指定形式

- ディスク容量不足が発生した場合、CLOSE 文の実行時に以下のエラーメッセージが出力されます。

  JMP0310I-I/U 'アクセス名またはファイル名'ファイルで'CLOSE'エラーが発生しました.'ERFLD=1C'

- PowerRDBconnector の排他制御は、COBOL ファイルシステムと比較して動作が異なる場合があります。詳細については、PowerRDBconnector 説明書を参照してください。

NCOBU01048

外部ファイルハンドラ

外部ファイルハンドラを使用して、Micro Focus COBOL が公開している FCD 構造を持ったファイルシステムを呼び出すことができます。 外部ファイルハンドラは、レコード順ファイル、行順ファイル、相対ファイル、索引ファイルに対応しています。 ここでは、外部ファイルハンドラの使い方と指定方法について説明します。

使い方

ファイル参照子にファイル識別名を指定する場合

ファイル識別名を環境変数として設定する時に、割り当てるファイルのファイル名に続き、“,EXFH”を指定します。

```
ファイル識別名=ファイル名,EXFH
```

ファイル参照子にファイル識別名定数を指定する場合

ファイル識別名定数の指定で、割り当てるファイルのファイル名に続き、“,EXFH”を指定します。

```
ASSIGN TO "ファイル名,EXFH".
```

ファイル参照子にデータ名を指定する場合

手続き部でデータ名に値(ファイル名)を設定する時に、割り当てるファイルのファイル名に続き、“,EXFH”を指定します。

```
MOVE "ファイル名,EXFH" TO データ名.
```

ファイル参照子については、レコード順ファイルの定義方法の“ファイル名とファイル参照子”を参照してください。行順ファイル、相対ファイル、索引ファイルの場合も、レコード順ファイルの場合と同様です。

指定方法

外部ファイルハンドラの指定方法には、以下の 2 つの方法があります。 どちらも、DLL 名と入口名の指定が必要です。

- 実行環境で有効になる指定方法
- ファイル単位で有効になる指定方法

2 つの方法が同時に指定されている場合は、ファイル単位の指定が優先されます。

実行環境で有効になる指定方法

実行時に実行環境変数@CBR_EXFH_API と実行環境変数@CBR_EXFH_LOAD を設定します。

```
@CBR_EXFH_API=入口名
```

入口名	結合するファイルシステムの入口名を指定します。(必須)

```
@CBR_EXFH_LOAD=DLL 名
```

DLL 名	結合するファイルシステムの DLL 名を指定します。

DLL 名には、絶対パス、相対パスのどちらも指定することができます。相対パスを用いた場合、カレントディレクトリからの相対パスとなります。

実行環境変数@CBR_EXFH_LOAD が設定されていない場合、実行環境変数@CBR_EXFH_API に指定された入口名を用いて、“入口名.DLL”としたものを DLL 名とみなして処理します。

[例 1]：結合するファイルシステムの入口名を“flsys”、DLL 名を“filesys.dll”とする場合

```
@CBR_EXFH_API=flsys @CBR_EXFH_LOAD=filesys.dll
```

[例 2]：結合するファイルシステムの入口名を“file”、DLL 名を“file.dll”とする場合

```
@CBR_EXFH_API=file
```

ファイル単位で有効になる指定方法

外部ファイルハンドラ情報ファイルを作成し、以下のように割り当てます。

```
ファイル識別名=ファイル名,EXFH,INF(外部ファイルハンドラ情報ファイル名)
```

外部ファイルハンドラ情報ファイルは、以下の内容のテキストファイルです。

```
[EXFH] @CBR_EXFH_API=入口名  @CBR_EXFH_LOAD=DLL 名
```

入口名	結合するファイルシステムの入口名を指定します。（必須）
DLL 名	結合するファイルシステムの DLL 名を指定します。

実行環境変数@CBR_EXFH_LOAD が設定されていない場合、実行環境変数@CBR_EXFH_API に指定された入口名を用いて、“入口名.DLL”としたものを DLL

名とみなして処理します。

[例] :外部ファイルハンドラ情報ファイル名を“aflsys.inf”、ファイル“Afile”に対する結合するファイルシステムの入口名を“aflsys”、DLL 名を“afilesys.dll”とする場合

```
ファイル識別名=Afile,EXFH,INF(aflsys.inf)
```

外部ファイルハンドラ情報ファイル(aflsys.inf)の内容

```
[EXFH] @CBR_EXFH_API=aflsys @CBR_EXFH_LOAD=afilesys.dll
```

注意事項

- 使用できる外部ファイルハンドラは DLL ファイルのみです。Micro Focus COBOL と異なり、オブジェクトファイルを使用することはできません。
- 外部ファイルハンドラは、Unicode 環境でコンパイルされた COBOL アプリケーションでは使用できません。
- 複数スレッドで COBOL アプリケーションを動作させる場合、結合するファイルシステムもマルチスレッドに対応していなければなりません。
- FILE STATUS 句を使う場合、外部ファイルハンドラから返される入出力状態値が返却されます。このため、“ファイル入出力状態一覧”の値とは異なる場合があります。
- 索引ファイルでは FIRST の指定された START 文はサポートされていません。

COBOL ファイルユーティリティ関数

COBOL ファイルユーティリティ関数には次の機能があります。

機能	関数名	
変換	ファイル変換関数	COB_FILE_CONVERT
ロード	ファイルロード関数	COB_FILE_LOAD
アンロード	ファイルアンロード関数	COB_FILE_UNLOAD
整列	ファイル整列関数	COB_FILE_SORT
再編成	ファイル再編成関数	COB_FILE_REORGANIZE
コピー	ファイルコピー関数	COB_FILE_COPY
移動	ファイル移動関数	COB_FILE_MOVE
削除	ファイル削除関数	COB_FILE_DELETE

[補足]:NetCOBOL for .NET V2.0 以前では、COBOL ファイルユーティリティ関数を呼び出すには、プログラム原型定義が必要でしたが、V2.1 以降ではプログラム原型定義は必要ありません。通常の CALL 呼出しで呼び出すことができます。

COBOL ファイルユーティリティ関数呼出しの準備

COBOL ファイルユーティリティ関数を呼び出すプログラムには、以下の記述が必要です。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
WORKING-STORAGE SECTION.
 COPY COBF-INF.              *>[1]
 01 RET-CODE PIC S9(9) COMP-5.  *>[2]
```

1. 作業場所節で登録集 COBF-INF.cbl を取り込む。

   この登録集は、NetCOBOL for .NET ランタイムシステムのインストールフォルダの"Utilities"サブフォルダに格納されています。COBOL ファイルユーティリティ関数を呼び出すプログラムをコンパイルする時には、登録集パスに以下のフォルダを指定してください。 （Windows が C:ドライブにインストールされているとしています）

COBOL ファイルユーティリティ関数の登録集の位置
C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Utilities¥

   また、この登録集を使用する場合、以下に注意してください。

   - COBOL ファイルユーティリティ関数の登録集を取り込む COPY 文の REPLACING 指定、またはこの COPY 文が置き換えの対象となるような REPLACE 文の記述はしないでください。
   - 登録集を変更した場合、動作は保証されません。
   - 各 COBOL ファイルユーティリティ関数を呼び出す前に、必ずインタフェース領域の COBF-INF を LOW-VALUE で初期化してください。

2. 作業場所節で復帰コードを受け取るデータを宣言する。

   COBOL ファイルユーティリティ関数の復帰コードを受け取るデータを宣言します。

ファイル変換関数

テキストファイルから可変長形式のレコード順ファイルの作成、または、可変長形式のレコード順ファイルからテキストファイルの作成を行います。

テキストファイルでは、バイナリデータは 16 進表記の文字列として表現されます。

呼出しの記述

```
CALL "COB_FILE_CONVERT" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

本関数では、レコード順ファイルの 1 レコードを構成するデータ項目を、データ形式で指定します。

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

変換元のテキストファイルまたはレコード順ファイルのパス名を指定します。

COBF-OUTPUT-FILENAME

作成するテキストファイルまたはレコード順ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

COBF-CONVERT-COND

変換方法を、以下の値で指定します。

値	意味
"B"	テキストファイルからレコード順ファイルに変換する
"T"	レコード順ファイルからテキストファイルに変換する

COBF-CONVERT-NUM

レコード順ファイルの 1 レコードを構成するデータ形式の個数を指定します。

指定できる値の最大は 256 です。

COBF-CONVERT-TYPE

レコード順ファイルの 1 レコードを構成するデータ項目を、データ形式で指定します。データ形式の指定には、以下の条件名を使用します。

条件名	値	意味
COBF-CONVERT-TYPE-ALLCHAR	"1"	レコード中の全データを ASCII/シフト JIS 形式とみなします
COBF-CONVERT-TYPE-ALLTXTBIN	"2"	レコード中の全データを 16 進形式とみなします
COBF-CONVERT-TYPE-CHAR	"3"	指定長だけ ACSII/シフト JIS 形式とみなします
COBF-CONVERT-TYPE-TXTBIN	"4"	指定長だけ 16 進形式とみなします
COBF-CONVERT-TYPE-ALLUCS2	"6"	レコード中の全データを UCS-2 形式とみなします。
COBF-CONVERT-TYPE-ALLUTF8	"7"	レコード中の全データを UTF-8 形式とみなします。
COBF-CONVERT-TYPE-UCS2	"8"	指定長だけ UCS-2 形式とみなします。
COBF-CONVERT-TYPE-UTF8	"9"	指定長だけ UTF-8 形式とみなします

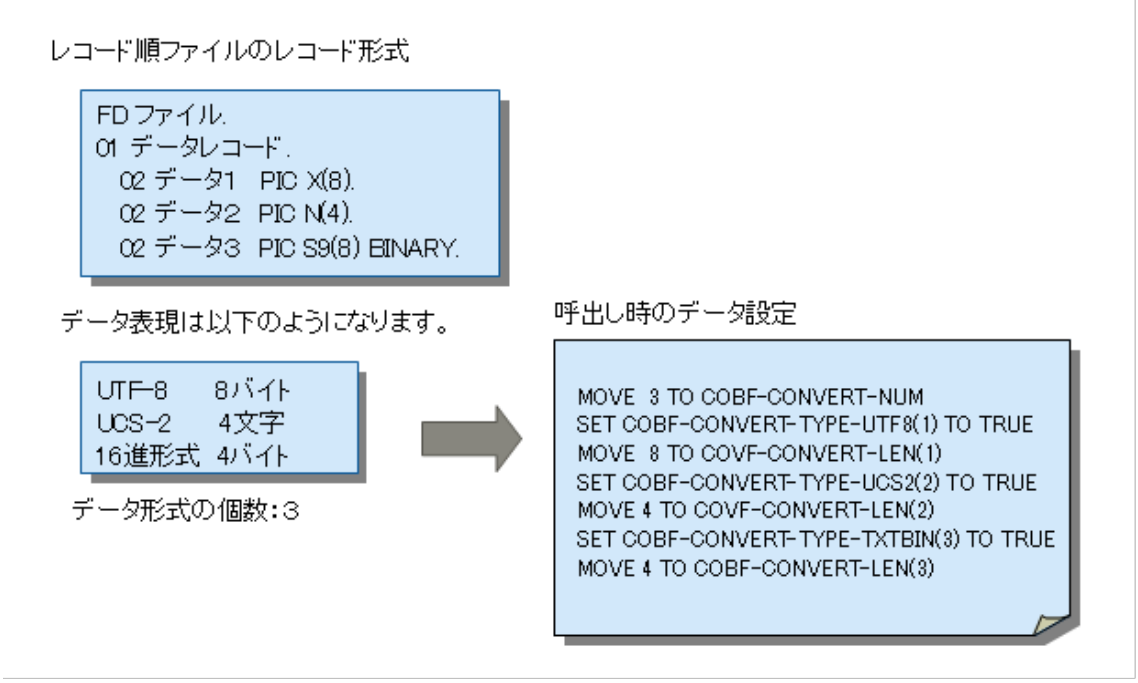
COBF-CONVERT-LEN

データ形式の長さ(UCS-2 形式の場合は文字数)を指定します。COBF-CONVERT-TYPE が以下の場合、必ず指定してください。

- COBF-CONVERT-TYPE-CHAR
- COBF-CONVERT-TYPE-TXTBIN
- COBF-CONVERT-TYPE-UCS2
- COBF-CONVERT-TYPE-UTF8

なお、COBF-CONVERT-TYPE が以下の場合、この指定は無視されます。

- COBF-CONVERT-TYPE-ALLCHAR
- COBF-CONVERT-TYPE-ALLTXTBIN
- COBF-CONVERT-TYPE-ALLUCS2
- COBF-CONVERT-TYPE-ALLUTF8



復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルの変換に成功しました。
-1	ファイルの変換に失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

指定例

例 1: レコード順ファイルからテキストファイルを作成する

入力ファイル名	C:¥INFILE
出力ファイル名	C:¥OUTFILE.TXT
変換方法	レコード順ファイル→テキストファイル
データ形式	すべて UTF-8 形式

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥INFILE" TO COBF-INPUT-FILENAME
        MOVE "C:¥OUTFILE.TXT" TO COBF-OUTPUT-FILENAME
        MOVE "T" TO COBF-CONVERT-COND
        MOVE 1 TO COBF-CONVERT-NUM
        SET  COBF-CONVERT-TYPE-ALLUTF8(1) TO TRUE
        CALL "COB_FILE_CONVERT" USING BY REFERENCE COBF-INF
                             RETURNING RET-CODE
```

例 2: テキストファイルからレコード順ファイルを作成する

入力ファイル名	C:¥INFILE
出力ファイル名	C:¥OUTFILE.TXT
変換方法	テキストファイル→レコード順ファイル
データ形式	混在(UTF-8 形式 3 バイト、16 進形式 2 バイト、UTF-8 形式 3 バイト)

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥INFILE.TXT" TO COBF-INPUT-FILENAME
        MOVE "C:¥OUTFILE" TO COBF-OUTPUT-FILENAME
        MOVE "B" TO COBF-CONVERT-COND
        MOVE 3 TO COBF-CONVERT-NUM
        SET  COBF-CONVERT-TYPE-UTF8(1) TO TRUE
        MOVE 3 TO COBF-CONVERT-LEN(1)
        SET  COBF-CONVERT-TYPE-TXTBIN(2) TO TRUE
        MOVE 2 TO COBF-CONVERT-LEN(2)
        SET  COBF-CONVERT-TYPE-UTF8(3) TO TRUE
        MOVE 3 TO COBF-CONVERT-LEN(3)
        CALL "COB_FILE_CONVERT" USING BY REFERENCE COBF-INF
                          RETURNING RET-CODE
```

ファイルロード関数

可変長形式のレコード順ファイルからレコード順ファイル/相対ファイル/索引ファイルを作成します。また、可変長形式のレコード順ファイルのレコードデータを、すでに存在するレコード順ファイル/相対ファイル/索引ファイルに追加することもできます。これをファイルの拡張といいます。

ファイルの拡張でエラーが発生した場合、出力ファイルはファイルの拡張を行う前の状態に戻されます。

呼出しの記述

```
CALL "COB_FILE_LOAD" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

変換元のレコード順ファイルのパス名を指定します。

COBF-OUTPUT-FILENAME

作成または拡張するファイルのパス名を指定します。作成時に、既に存在しているファイルを指定した場合、エラーとなります。また、拡張時に、指定したファイルが存在していない場合、エラーとなります。

COBF-OUTPUT-ATTR

作成または拡張するファイルのファイル編成を、以下の文字で指定します。

値	意味
"S"	レコード順ファイル
"R"	相対ファイル
"I"	索引ファイル

COBF-OUTPUT-RECFM

作成または拡張するファイルのレコード形式を、以下の文字で指定します。

値	意味
"F"	固定長
"V"	可変長

COBF-OUTPUT-RECLEN

作成または拡張するファイルのレコード長を指定します。可変長の場合は、最大レコード長を指定します。

COBF-OUTPUT-KEYNUM

索引ファイルを作成または拡張する場合に、レコードキーとして扱うデータ項目の個数を指定します。

COBF-OUTPUT-USE-EXTKEY

索引ファイルを作成するまたは拡張する場合、レコードキーとして扱うデータ項目に属性を指定するかしないかを以下の文字で指定します。属性の指定は、COBF-OUTPUT-EXT-KEY-TYPE で行います。

値	意味
"Y"	データ項目の属性を使用する
" "	データ項目の属性を使用しない

UCS-2 形式のデータ項目をキーにする場合、"Y"を指定します。

COBF-LOAD-COND

ファイルを作成するか、ファイルを拡張するかを、以下の文字で指定します。

値	意味
"C"	ファイルを作成する
"E"	ファイルを拡張する

レコードキーの個数分、以下の情報を指定します。

COBF-OUTPUT-OFFSET

レコードキーとするデータ項目のレコード内位置を、レコードの先頭を 0 とした相対位置で指定します。

COBF-OUTPUT-KEYLEN

レコードキーとするデータ項目の長さをバイト数で指定します。

COBF-OUTPUT-KEYCONT

1 つのレコードキーの構成を、以下の文字で指定します。

値	意味
"C"	キーの連続性を示す
"E"	１つのキー定義の終了を示す

COBF-OUTPUT-KEYDUP

S レコードキーの重複を許す場合に"D"を指定します。なお、この項目は COBF-OUTPUT-KEYCONT に"E"が指定された場合だけ有効です。

COBF-OUTPUT-EXT-KEY-TYPE

COBF-OUTPUT-USE-EXTKEY に"Y"を指定した場合、レコードキーとするデータ項目の属性を、以下の条件名を使用して指定します。

条件名	値	意味
COBF-EXT-KEY-TYPE-PICX	1	PIC N 以外の項目
COBF-EXT-KEY-TYPE-PICN	2	PIC N の項目

復帰コード

本関数からの復帰コードは、特殊レジスタ PROGRAM-STATUS を使用して受け取ります。

復帰値	意味
0	ファイルのロードに成功しました。
-1	ファイルのロードに失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

- 出力ファイルの最大レコード長より大きいレコードが入力ファイルに存在した場合、実行時にエラーとなります。
- 索引ファイルを拡張する場合、最大キー値より小さいキー値のレコードを書き出すことができます。

指定例

例 1: レコード順ファイルから索引ファイルを作成する

入力ファイル名	C:¥INFILE
出力ファイル名	C:¥IXDFILE
ファイル編成	索引ファイル
レコード形式	可変長
レコード長	80 バイト
主レコードキー	2 つのデータ項目で構成する。また、レコードキーの重複を許す。
	［キー１］0 バイトオフセットから始まる、5 バイトの項目
	［キー２］10 バイトオフセットから始まる、5 バイトの項目
副レコードキー	5 バイトオフセットから始まる、5 バイトの項目

```
MOVE LOW-VALUE TO COBF-INF
MOVE "C:¥INFILE" TO COBF-INPUT-FILENAME
MOVE "C:¥IXDFILE" TO COBF-OUTPUT-FILENAME
MOVE "I" TO COBF-OUTPUT-ATTR
MOVE "V" TO COBF-OUTPUT-RECFM
MOVE 80 TO COBF-OUTPUT-RECLEN
MOVE 3 TO COBF-OUTPUT-KEYNUM
MOVE 0 TO COBF-OUTPUT-OFFSET(1)
MOVE 5 TO COBF-OUTPUT-KEYLEN(1)
MOVE "C" TO COBF-OUTPUT-KEYCONT(1)
MOVE 5 TO COBF-OUTPUT-OFFSET(2)
MOVE 10 TO COBF-OUTPUT-KEYLEN(2)
MOVE "E" TO COBF-OUTPUT-KEYCONT(2)
MOVE "D" TO COBF-OUTPUT-KEYDUP(2)
MOVE 5 TO COBF-OUTPUT-OFFSET(3)
MOVE 5 TO COBF-OUTPUT-KEYLEN(3)
MOVE "E" TO COBF-OUTPUT-KEYCONT(3)
MOVE "C" TO COBF-LOAD-COND
CALL "COB_FILE_LOAD" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

例 2: レコード順ファイルの内容を相対ファイルに追加する(拡張)

入力ファイル名	C:¥INFILE
出力ファイル名	C:¥IXDFILE
ファイル編成	相対ファイル
レコード形式	固定長
レコード長	80 バイト

```
MOVE LOW-VALUE TO COBF-INF
MOVE "C:¥INFILE" TO COBF-INPUT-FILENAME
MOVE "C:¥RELFILE" TO COBF-OUTPUT-FILENAME
MOVE "R" TO COBF-OUTPUT-ATTR
MOVE "F" TO COBF-OUTPUT-RECFM
MOVE 80 TO COBF-OUTPUT-RECLEN
MOVE "E" TO COBF-LOAD-COND
CALL "COB_FILE_LOAD" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

ファイルアンロード関数

レコード順ファイル/相対ファイル/索引ファイルから可変長形式のレコード順ファイルを作成します。

呼出しの記述

```
CALL "COB_FILE_UNLOAD" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

変換元のレコード順ファイル/相対ファイル/索引ファイルのパス名を指定します。

COBF-INPUT-ATTR

変換元のファイルのファイル編成を以下の文字で指定します。

値	意味
"S"	レコード順ファイル
"R"	相対ファイル
"I"	索引ファイル

COBF-INPUT-RECFM

変換元のファイルのレコード形式を以下の文字で指定します。

値	意味
"F"	固定長
"V"	可変長

COBF-INPUT-RECLEN

変換元のファイルのレコード長を指定します。可変長の場合は、最大レコード長を指定します。ただし、入力ファイルが索引ファイルの場合は指定できません。

COBF-OUTPUT-FILENAME

作成するレコード順ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルのアンロードに成功しました。
-1	ファイルのアンロードに失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

指定例

例: 相対ファイルからレコード順ファイルを作成する

入力ファイル名	C:¥RELFILE
出力ファイル名	C:¥OUTFILE
ファイル編成	相対ファイル
レコード形式	固定長
レコード長	80 バイト

```
        MOVE LOW-VALUE TO COBF-INF.
        MOVE "C:¥RELFILE" TO COBF-INPUT-FILENAME.
        MOVE "R" TO COBF-INPUT-ATTR.
        MOVE "F" TO COBF-INPUT-RECFM.
        MOVE 80 TO COBF-INPUT-RECLEN.
        MOVE "C:¥OUTFILE" TO COBF-OUTPUT-FILENAME.
        CALL "COB_FILE_UNLOAD" USING BY REFERENCE COBF-INF
                            RETURNING RET-CODE
```

ファイル整列関数

レコード中の任意のデータ項目をキーとして、ファイル内のレコードを昇順または降順に整列し、整列された結果を可変長形式のレコード順ファイルに出力します。

呼出しの記述

```
CALL "COB_FILE_SORT" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

整列するファイルのパス名を指定します。指定するファイルは、レコード順ファイル、行順ファイル、相対ファイルまたは索引ファイルのどれかでなければなりません。

COBF-INPUT-ATTR

整列するファイルのファイル編成を、以下の文字で指定します。

値	意味
"S"	レコード順ファイル
"L"	行順ファイル
"R"	相対ファイル
"I"	索引ファイル

COBF-INPUT-RECFM

整列するファイルのレコード形式を、以下の文字で指定します。

値	意味
"F"	固定長
"V"	可変長

COBF-INPUT-RECLEN

整列するファイルのレコード長を指定します。可変長の場合は、最大レコード長を指定します。整列するファイルが索引ファイルの場合は指定できません。

COBF-OUTPUT-FILENAME

整列結果を出力するレコード順ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

COBF-SORT-KEYNUM

整列キーの個数を指定します。

指定できる値の最大は 64 です。

COBF-SORT-OFFSET

キーとする任意のデータ項目の、レコードの先頭からの相対位置を指定します。

COBF-SORT-KEYLEN

キーとする任意のデータ項目の大きさを指定します。なお、キー項目に PIC 1()BIT を指定した場合、1 バイトの大きさに対するマスク値を指定します。

COBF-SORT-KEYATTR

任意のキーの項目属性を指定します。項目属性の指定には以下の条件名を使用します。

条件名	値	意味
COBF-SORT-KEYATTR-PIC1	1	PIC 1()
COBF-SORT-KEYATTR-PIC1B	2	PIC 1() BIT
COBF-SORT-KEYATTR-PICX	3	PIC X()
COBF-SORT-KEYATTR-PICN	4	PIC N()
COBF-SORT-KEYATTR-PIC9	5	PIC 9()
COBF-SORT-KEYATTR-PICS9	6	PIC S9()
COBF-SORT-KEYATTR-PIC9L	7	PIC 9() LEADING
COBF-SORT-KEYATTR-PICS9L	8	PIC S9() LEADING
COBF-SORT-KEYATTR-PICS9T	10	PIC S9() TRAILING
COBF-SORT-KEYATTR-PICS9LS	12	PIC S9() LEADING SEPARATE

COBF-SORT-KEYATTR-PICS9TS	14	PIC S9() TRAILING SEPARATE
COBF-SORT-KEYATTR-PIC9PD	15	PIC 9() PACKED-DECIMAL
COBF-SORT-KEYATTR-PICS9PD	16	PIC S9() PACKED-DECIMAL
COBF-SORT-KEYATTR-PIC9B	17	PIC 9() BINARY
COBF-SORT-KEYATTR-PICS9B	18	PIC S9() BINARY
COBF-SORT-KEYATTR-PIC9C5	19	PIC 9() COMP-5
COBF-SORT-KEYATTR-PICS9C5	20	PIC S9() COMP-5
COBF-SORT-KEYATTR-PICC1	21	COMP-1
COBF-SORT-KEYATTR-PICC2	22	COMP-2

COBF-SORT-KEYSEQ

整列順序を、以下の文字で指定します。

値	意味
"A"	昇順にレコードを整列する
"D"	降順にレコードを整列する

復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルの整列に成功しました。
-1	ファイルの整列に失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

指定例

例: 相対ファイルを整列し、結果を OUTFILE に出力する

入力ファイル名	C:¥RELFILE	
出力ファイル名	C:¥OUTFILE	
整列条件	［第１キー］項目属性	日本語
	レコード内オフセット	0
	大きさ	2
	整列順序	昇順
	［第２キー］項目属性	英数字
	レコード内オフセット	10
	大きさ	5
	整列順序	降順
ファイル属性	ファイル編成	相対ファイル
	レコード形式	可変長
	レコード長	80 バイト

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥RELFILE" TO COBF-INPUT-FILENAME
        MOVE "R" TO COBF-INPUT-ATTR
        MOVE "V" TO COBF-INPUT-RECFM
        MOVE 80 TO COBF-INPUT-RECLEN
        MOVE "C:¥OUTFILE" TO COBF-OUTPUT-FILENAME
        MOVE 2 TO COBF-SORT-KEYNUM
        SET COBF-SORT-KEYATTR-PICN(1) TO TRUE
        MOVE 0 TO COBF-SORT-OFFSET(1)
        MOVE 2 TO COBF-SORT-KEYLEN(1)
        MOVE "A" TO COBF-SORT-KEYSEQ(1)
        SET COBF-SORT-KEYATTR-PICX(2) TO TRUE
        MOVE 10 TO COBF-SORT-OFFSET(2)
        MOVE 5 TO COBF-SORT-KEYLEN(2)
        MOVE "D" TO COBF-SORT-KEYSEQ(2)
        CALL "COB_FILE_SORT" USING BY REFERENCE COBF-INF
                             RETURNING RET-CODE
```

ファイル再編成関数

索引ファイルの空きブロックを削除し、再編成した内容を別の索引ファイルに出力します。再編成した索引ファイルのファイルサイズは、再編成前のファイルサイズよりも小さくなります。

呼出しの記述

```
CALL "COB_FILE_REORGANIZE" USING BY REFERENCE COBF-INF
                         RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

再編成を行う索引ファイルのパス名を指定します。

COBF-OUTPUT-FILENAME

再編成後の内容を出力する索引ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルの再編成に成功しました。
-1	ファイルの再編成に失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

指定例

例: 索引ファイルを再編成し、OUTFILE に出力する

入力ファイル名	C:¥IXDFILE
出力ファイル名	C:¥OUTFILE

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥IXDFILE" TO COBF-INPUT-FILENAME
        MOVE "C:¥OUTFILE" TO COBF-OUTPUT-FILENAME
        CALL "COB_FILE_REORGANIZE" USING BY REFERENCE COBF-INF
                              RETURNING RET-CODE
```

ファイルコピー関数

ファイルを複写します。

呼出しの記述

```
CALL "COB_FILE_COPY" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

複写元ファイルのパス名を指定します。

COBF-OUTPUT-FILENAME

複写先ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルの複写に成功しました。
-1	ファイルの複写に失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

ファイル名にワイルドカード(?,*)を指定することはできません。

指定例

例: INFILE を OUTFILE にコピーする

複写元ファイル名	C:¥INFILE
複写先ファイル名	C:¥OUTFILE

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥INFILE" TO COBF-INPUT-FILENAME
        MOVE "C:¥OUTFILE" TO COBF-OUTPUT-FILENAME
        CALL "COB_FILE_COPY" USING BY REFERENCE COBF-INF
                        RETURNING RET-CODE
```

ファイル移動関数

ファイルを移動します。

呼出しの記述

```
CALL "COB_FILE_MOVE" USING BY REFERENCE COBF-INF
                  RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

移動元ファイルのパス名を指定します。

COBF-OUTPUT-FILENAME

移動先ファイルのパス名を指定します。既に存在しているファイルを指定した場合、エラーとなります。

復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルの移動に成功しました。
-1	ファイルの移動に失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

ファイル名にワイルドカード(?,*)を指定することはできません。

指定例

例: INFILE を OUTFILE に移動する

移動元ファイル名	C:¥INFILE
移動先ファイル名	F:¥OUTFILE

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥INFILE" TO COBF-INPUT-FILENAME
        MOVE "F:¥OUTFILE" TO COBF-OUTPUT-FILENAME
        CALL "COB_FILE_MOVE" USING BY REFERENCE COBF-INF
                         RETURNING RET-CODE
```

ファイル削除関数

ファイルを削除します。

呼出しの記述

```
CALL "COB_FILE_DELETE" USING BY REFERENCE COBF-INF
                    RETURNING RET-CODE
```

COBOL ファイルユーティリティ関数を呼び出すプログラムでは、作業場所節でこれらの関数を使用するための記述が必要です。これらの詳細については COBOL ファイルユーティリティ関数の"COBOL ファイルユーティリティ関数呼出しの準備"を参照してください。

呼出し時のデータ設定

以下に、関数を呼び出す前に設定するデータについて説明します。

COBF-INPUT-FILENAME

削除するファイルのパス名を指定します。

復帰コード

本関数からの復帰コードは、RET-CODE に設定されます。

復帰値	意味
0	ファイルの削除に成功しました。
-1	ファイルの削除に失敗しました。

復帰値-1 が返却された場合、エラーの詳細は、COBF-MESSAGE に文字列で設定されます。

ファイル名にワイルドカード(?,*)を指定することはできません。

指定例

例:C:¥INFILE を削除する

```
        MOVE LOW-VALUE TO COBF-INF
        MOVE "C:¥INFILE" TO COBF-INPUT-FILENAME
        CALL "COB_FILE_DELETE" USING BY REFERENCE COBF-INF
                            RETURNING RET-CODE
```

NCOBU00291

印刷処理

ここでは、1 行単位のデータや帳票形式のデータを印刷装置に出力する方法について説明します。

このセクションの内容

印刷方法の種類

COBOL プログラムでデータを印刷するには、印刷ファイルまたは表示ファイルを使用します。

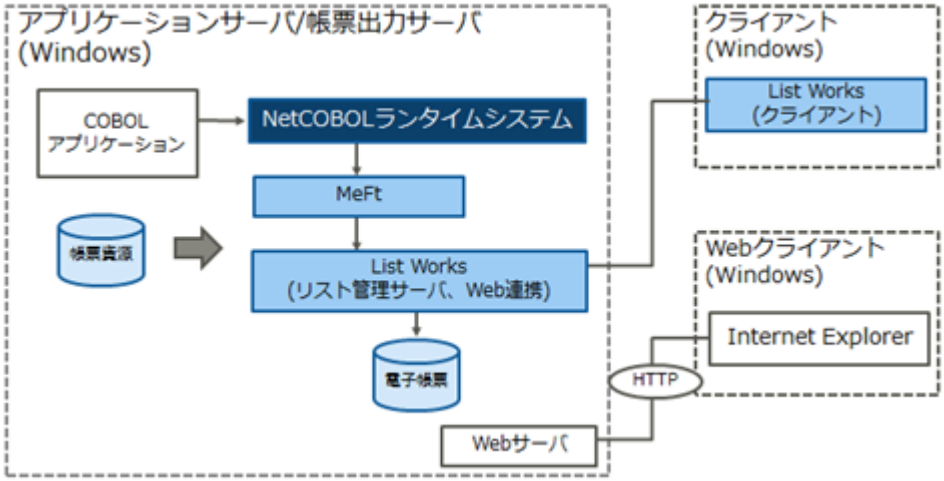
ここでは、これらのファイルを使った印刷方法の概要、印字文字の種類、フォームオーバレイパターン、FCB および帳票定義体について説明します。なお、印刷機能は、使用する印刷装置により異なる場合があるので、各印刷装置の説明書を参照してください。

注意

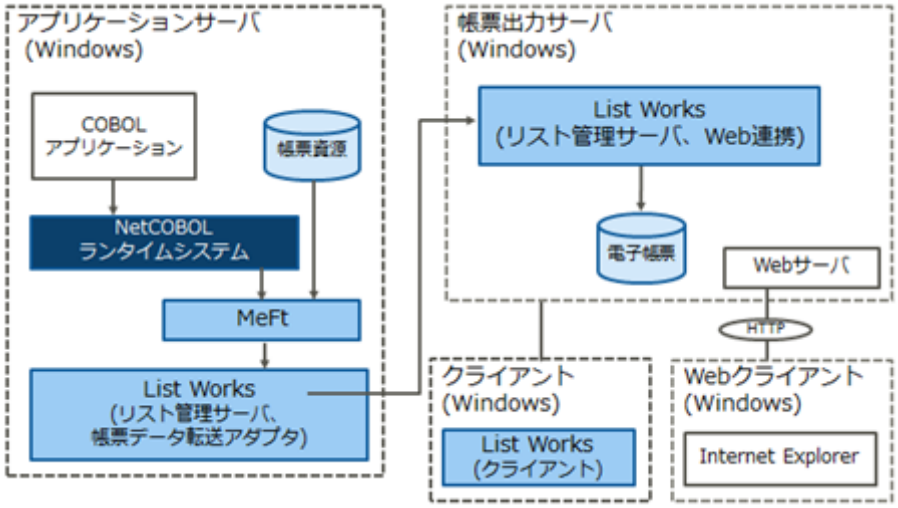
印刷ファイルで出力するデータは、表示用(USAGE IS DISPLAY)のデータ項目で定義されていなければなりません。データ中にバイナリの不正なコードが含まれる場合、正しく印字されないことがあります。

印刷ファイルまたは表示ファイルを使用して、直接プリンタに出力することもできますが、 帳票製品と連携することで多様な運用環境に対応することができます。

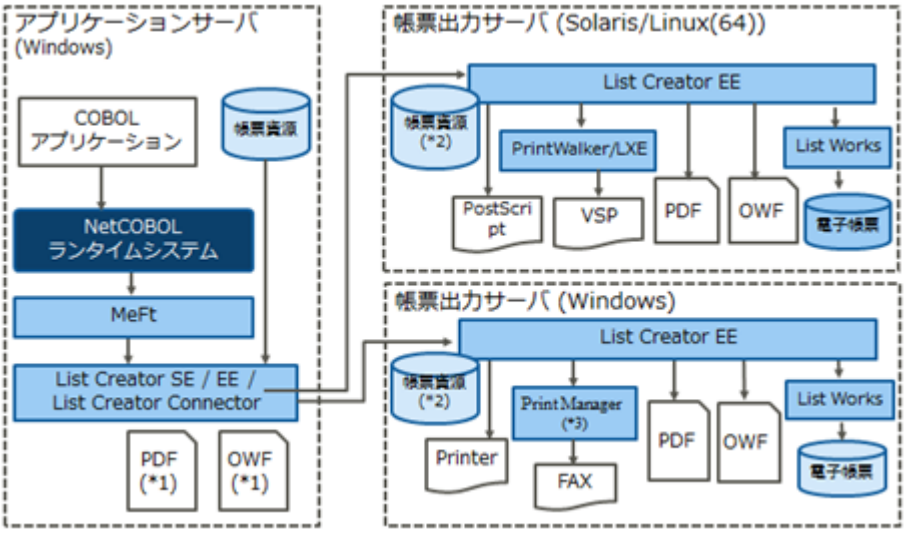
図：List Works 連携（電子帳票出力機能)アプリケーションサーバ/帳票出力サーバが同一筐体の場合



図：List Works 連携（電子帳票出力機能)アプリケーションサーバ/帳票出力サーバが別筐体の場合



図：List Creator 連携(COBOL アプリケーション連携機能)



*1) 帳票サーバで出力した PDF/OWF ファイルをアプリケーションサーバへ転送(配置)することも可能です。 *2) アプリケーションサーバから転送します。 *3) Print Manager/FAX 出力連携製品です。Windows(64)では未サポートです。

このセクションの内容

特殊レジスタ

EDIT-MODE、EDIT-OPTON、EDIT-COLOR などの特殊レジスタについて説明します。

印字文字の配置座標

印字文字の配置座標について説明します。

印刷情報ファイル

印刷情報ファイルについて説明します。

印刷ファイル/表示ファイルの決定方法

印刷ファイル/表示ファイルの決定方法について説明します。

サービス配下の注意事項(印刷時)

サービス配下の注意事項(印刷時)について説明します。

帳票設計

帳票設計について説明します。

印刷不可能な領域

印刷不可能な領域について説明します。

フォントテーブル

フォントテーブルについて説明します。

NCOBU00293

各印刷方法の概要

印刷ファイルには、FORMAT 句なし印刷ファイルと FORMAT 句付き印刷ファイルがあります。

- FORMAT 句なし印刷ファイルは、行単位のデータを印刷したり、行単位のデータをフォームオーバレイパターンと合成したり、FCB を使って行間隔などの印刷情報を設定してデータを印刷するときに使用します。
- FORMAT 句付き印刷ファイルは、FORMAT 句なし印刷ファイルの機能に加えて、帳票定義体を使った帳票形式のデータの印刷を行うことができます。

ここでは、印刷ファイルおよび表示ファイルを以下のように分類して説明します。

- [1] FORMAT 句なし印刷ファイル(行単位のデータを印刷する)
- [2] FORMAT 句なし印刷ファイル(フォームオーバレイおよび FCB を使用して印刷する)
- [3] FORMAT 句付き印刷ファイル
- [4] 表示ファイル

[1]～[4]の印刷方法の特徴、利点および用途を"表:印刷方法の特徴・利点・用途"に、必要な関連製品を"表:関連製品"に示します。

表: 印刷方法の特徴・利点・用途

使用するファイルの種類		[1]	[2]	[3]	[4]
特徴	行単位のデータの印刷ができる	○	○	○	×
	フォームオーバレイパターンと合成した印刷ができる	×	○	○	○ (注)
	帳票定義体を使った帳票印刷ができる	×	×	○	○
利点	プログラムの記述が簡単	◎	○	○	◎
	帳票形式の印刷が簡単	○	◎	◎	◎
	他システムで作成した既存の帳票定義体を使用できる	×	×	◎	◎
	プログラム中で各種印刷情報を指示することができる	×	◎	◎	○
用途	帳票を印刷する	○	◎	◎	◎

◎: 使用可能であり適している

○: 使用可能である

×: 使用不可能

注: 帳票定義体にオーバレイパターン名が指定されている場合またはプリンタ情報ファイルにオーバレイパターン名を指定した場合だけ印刷可能です。詳細については、"MeFt ユーザーズガイド"を参照してください。

表: 関連製品

使用するファイルの種類	[1]	[2]	[3]	[4]
FORM (注 1)	―	○	○	○
FORM オーバレイオプション	―	○	○	○
PowerFORM (注 2)	―	○	○	○
MeFt	―	―	○	○
Interstage List Works (注 3)	―	―	○	○
Interstage List Creator Enterprise Edition (注 4)	―	―	○	○

注 1: オーバレイを作成するために使用する場合は、FORM オーバレイオプションが必要です。

注 2: FORM に同梱されています。

注 3: 帳票を電子化したり、電子化された帳票に対するさまざまな操作・管理を行う場合に必要となります。このセクションでは、特に断りがない限り、Interstage List Works を ListWORKS と記述します。

注 4: 帳票を PDF ファイルとして出力する場合に必要となります。

以下に、各印刷方法の概要を説明します。

[1] FORMAT 句なし印刷ファイル(行単位のデータを印刷する)

FORMAT 句なし印刷ファイルでは、行単位のデータを印刷装置に出力することができます。このとき、論理ページの大きさを指定したり、行送りやページ替えを指定することもできます。

行単位のデータを印刷するときの印刷ファイルの使い方については、行単位のデータを印刷する方法を参照してください。

[2] FORMAT 句なし印刷ファイル(フォームオーバレイおよび FCB を使用して印刷する)

FORMAT 句なし印刷ファイルでは、1 ページ分の出力データをフォームオーバレイパターンと合成したり、FCB を使って印刷情報を指示することができます。1 ページ分の出力データをフォームオーバレイパターンと合成することを指示するには、制御レコードを使います。FCB を使って印刷情報を指示するには、FCB 制御文をアプリケーション構成ファイルに記述します。

フォームオーバレイパターンについてはフォームオーバレイパターンを、FCB については FCB を、使い方についてはフォームオーバレイおよび FCB を使う方法を参照してください。

[3] FORMAT 句付き印刷ファイル

FORMAT 句付き印刷ファイルとは、プログラムのファイル定義で FORMAT 句を指定した印刷ファイルのことです。FORMAT 句付き印刷ファイルでは、パーティション形式の帳票定義体を使って、帳票形式のデータを印刷することができます。また、前述したフォームオーバレイパターンを使った帳票印刷および FCB を使った行単位のデータ印刷を行うこともできます。ただし、FORMAT 句付き印刷ファイルによる帳票印刷では、MeFt が必要となります。

帳票定義体については帳票定義体を、帳票定義体を使った印刷ファイルについては帳票定

義体を使う印刷ファイルの使い方を参照してください。

[4] 表示ファイル

表示ファイルでは、帳票定義体に定義した帳票形式のデータを印刷することができます。前述した FORMAT 句付き印刷ファイルとの相違点は、パーティション形式以外の帳票定義体も使用できることです。ただし、行レコードの印刷やフォームオーバレイパターンおよび FCB をプログラムから変更することはできません。

帳票印刷を行う表示ファイルの使い方については、表示ファイル(帳票印刷)の使い方を参照してください。

表示ファイルによる帳票印刷では、MeFt が必要になります。

印字文字

印字文字の印字属性(大きさ、書体、スタイル、形態、方向および間隔)を、データ記述項の CHARACTER TYPE 句で指定します。CHARACTER TYPE 句には、MODE-n、呼び名および印字モード名が指定できます。それぞれの書き方から指定可能な印字属性を以下に示します。

なお、日本語項目を印字する場合には、CHARACTER TYPE 句の記述が必須です。

指定方法	印字文字の属性					
	大きさ	書体	スタイル	形態	方向	間隔
CHARACTER TYPE MODE-n	○	－	－	○ *2	－	○ *3
CHARACTER TYPE 呼び名	○	○	－	○	○	○ *3
CHARACTER TYPE 印字モード名	○	○	○ *1	○	○	○

*1: PRINTING MODE 句の FONT 指定に FONT-nnn を指定しなければなりません。

*2: MODE-n の後に BY 呼び名 を指定しなければなりません。

*3: 印字文字の大きさと形態から決定されます。

- CHARACTER TYPE 句に MODE-1、MODE-2、MODE-3 を指定した場合、それぞれ印字文字の大きさを 12 ポイント、9 ポイント、7 ポイントとすることができます。
- CHARACTER TYPE 句に呼び名を指定した場合、特殊名段落の機能名句でその呼び名と関連付けられた機能名の示す印字属性で印字することができます。機能名については、COBOL 文法書の「5.4.3 CHARACTER TYPE 句」を参照してください。
- CHARACTER TYPE 句に印字モード名を指定した場合、特殊名段落の PRINTING MODE 句で印字モード名に関連付けて印字属性を定義します。定義された印字属性で印字することができます。PRINTING MODE 句の書き方については、COBOL 文法書の「4.2.3.11 PRINTING MODE 句」を参照してください。

指定できる印字属性について、以下に説明します。

印字文字の大きさ

3.0～300.0 ポイントの文字サイズを指定できます。

◆指定方法

文字サイズの指定方法を以下に示します。ただし、ラスタフォントタイプ(固定サイズ)のデバイスフォントを選択した場合、各プリンタ装置に搭載されているフォントのポイント数(文字サイズ)で印字されます。

指定方法	文字サイズ
MODE-1/ MODE-2/ MODE-3	12 ポイント/ 9 ポイント/ 7 ポイント
呼び名と関連付ける機能名で指定	12 ポイント/ 9 ポイント/ 7 ポイント
印字モード名 (PRINTING MODE 句の SIZE 指定)	3.0～300.0 ポイントを 0.1 ポイント単位で指定できます。文字サイズの指定を省略した場合、文字間隔の指定に合わせた文字サイズで印字します。文字サイズと文字間隔を両

	方省略した場合、以下の文字サイズで印字します。 ・ 日本語項目：12 ポイント ・ 英数字項目： 環境変数情報@CBR_PrinterANK_Size 指定が有効な場合(注)、指定された文字サイズとなります。指定されていない場合 7.0 ポイントで印字します。

注 : CHARACTER TYPE 句および PRINTING POSITION 句を使用していない項目だけに有効です。

指定方法の詳細については、COBOL 文法書の「5.4.3 CHARACTER TYPE 句」および「4.2.3.11 PRINTING MODE 句」を参照してください。

◆ラスタフォントタイプのデバイスフォントを選択した場合

以下に、富士通製のプリンタ装置にラスタフォントタイプのデバイスフォントを選択して印字した場合の、プログラムで指定した文字サイズと、実際に印字される文字サイズの対応を示します。

プログラムに指定した文字サイズ	プリンタ種別	FMLBP		FMPR
	漢字 ROM カートリッジ	あり	なし	―
3.0 ～ 8.9		7.0	10.5	10.5
9.0 ～ 10.4		9.0		
10.5 ～ 11.9		10.5		
12.0 ～ 300.0		12.0		

(単位:ポイント)

- FORMAT 句なし印刷ファイルのフォントは、環境変数情報@PrinterFontName (印刷ファイルで使用するフォントの指定)で指定します。フォントの指定を省略した場合、通常"明朝"や"ゴシック"などのデバイスフォントが選択されます。
- 以下の場合、スケーラブルフォントとなり 0.1 ポイントきざみで 3.0～300.0 ポイントの範囲で印字することができます。
  - デバイスフォントがアウトラインフォントである場合
  - 環境変数情報に"MS 明朝"や"MS ゴシック"などの trueType フォントを指定した場合
- 上記表の漢字 ROM カートリッジは、7 ポイント/9 ポイント/12 ポイントすべてのカートリッジを装着した場合を想定しています。
- 他社のプリンタ装置を使用する場合、デバイスフォントの扱いは各社プリンタ装置および各社プリンタドライバに依存します。プリンタ装置の取扱い説明書などを参照してください。

印字文字の書体

明朝体/明朝体半角/ゴシック体/ゴシック体半角/ゴシック DP/書体番号を指定できます。

◆指定方法

指定方法	書体
MODE-1/ MODE-2/ MODE-3	明朝体
呼び名と関連付ける機能名で指定	明朝体／ゴシック体 書体の指定を省略した場合、“明朝体”で印字します。
印字モード名 (PRINTING MODE 句の FONT 指定)	明朝体／明朝体半角／ゴシック体／ゴシック体半角／ゴシック DP／書体番号 書体の指定を省略した場合、以下の書体で印字します。 ・ 日本語項目：明朝体 ・ 英数字項目：ゴシック体

指定方法の詳細については、COBOL 文法書の「5.4.3 CHARACTER TYPE 句」および「4.2.3.11 PRINTING MODE 句」を参照してください。

◆印字の規則

- 書体番号の指定がある場合、環境変数情報@CBR_PrintFontTable (印刷ファイルで使用するフォントテーブルの指定)またはファイル識別名に指定されたフォントテーブル内のそれぞれの書体番号に対応付けたフォントフェース名の文字書体で印字されます。書体番号とは、PRINTING MODE 句に指定された"FONT-nnn"のことを指します。

  フォントテーブル名を指定しない場合またはフォントテーブル内に書体番号に対応付けたフォントフェース名の指定がない場合は、書体番号の値にかかわらず、すべて明朝体(通常の書体)で印字されます。

  ファイル識別名にフォントテーブルを指定する方法については以下を参照してください。

  - ファイル識別名 (プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定)
  - ファイル識別名 (プログラムで使用するプリンタおよび各種パラメタの指定)

- FORMAT 句なし印刷ファイルでは、環境変数情報@PrinterFontName (印刷ファイルで使用するフォントの指定)にフォントフェース名が指定されている場合、指定されているフォントフェース名の文字書体で印字されます。フォントフェース名の指定がない場合、以下に示す順番で検索します。検索の結果、どのフォントもシステムにインストールされていない場合には実行時エラーとなります。

  【明朝体/明朝体半角の場合】

  1. 明朝
  2. MS 明朝

【ゴシック体/ゴシック体半角/ゴシック DP の場合】

1. ゴシック(ゴシック体をサポートするプリンタにだけ有効)
2. MS ゴシック

・ FORMAT 句付き印刷ファイルおよび表示ファイルの場合に選択されるフォントについては、"MeFt ユーザーズガイド"を参照してください。

"MS P 明朝"や"MS P ゴシック"など、プロポーショナルフォントを選択した場合でも、文字間隔はプログラムで指定した固定ピッチとなります。この場合、横幅の狭い文字は、左端に寄せられます。

印字文字のスタイル

標準/太字/斜体/太字・斜体を指定できます。文字スタイルは、書体番号の指定がある文字書体に対して指定できます。

◆指定方法

文字スタイルの指定方法については、フォントテーブルを参照してください。文字スタイルの指定を省略した場合、“標準”が指定されたものと解釈します。

・ 印字文字の書体にアウトラインフォント以外の文字書体が指定または選択された場合、スタイルの指定にかかわらず、文字書体の持つスタイルで印字されます。

・ 文字スタイルはフォントテーブル内に指定するため、印字文字の書体には書体番号を指定しなければなりません。印字文字の書体に書体名を指定した場合、印字スタイルはすべて標準になります。

印字文字の形態

全角/全角長体/全角平体/全角倍角/半角/半角長体/半角平体/半角倍角を指定できます。

◆指定方法

指定方法	文字形態
MODE-n の呼び名に関連付ける機能名で指定	全角長体／全角平体／全角倍角／半角／半角倍角
呼び名と関連付ける機能名で指定	全角長体／全角平体／全角倍角／半角 文字形態の指定を省略した場合、“全角”で印字します。
印字モード名 (PRINTING MODE 句の FORM 指定)	全角長体／全角平体／全角倍角／全角／半角長体／半角平体／半角倍角／半角 文字形態の指定を省略した場合、“全角”で印字します。

指定方法の詳細については、COBOL 文法書の「5.4.3 CHARACTER TYPE 句」および「4.2.3.11 PRINTING MODE 句」を参照してください。

ラスタフォントタイプのデバイスフォントを選択した場合、指定にかかわらず、すべて全角で印字されます。また、他社のプリンタ装置を使用した場合、そのプリンタ装置やプリンタドライバの持つ機能に依存します。

印字文字の方向

縦書き/横書きを指定できます。

◆指定方法

指定方法	文字方向
MODE-1/ MODE-2/ MODE-3	横書き
呼び名と関連付ける機能名で指定	縦書き／横書き 文字方向の指定を省略した場合、“横書き”で印字します。
印字モード名 (PRINTING MODE 句の ANGLE 指定)	縦書き／横書き 文字方向の指定を省略した場合、“横書き”で印字します。

指定方法の詳細については、COBOL 文法書の「5.4.3 CHARACTER TYPE 句」および「4.2.3.11 PRINTING MODE 句」を参照してください。

縦書きは、日本語項目に対してだけ有効となります。

印字文字の間隔

0.01～24.00cpi の文字間隔を指定できます。

◆指定方法

指定方法	文字間隔
MODE-1/ MODE-2/ MODE-3	印字文字の大きさと形態によって決定されます。(注)
呼び名と関連付ける機能名で指定	
印字モード名 (PRINTING MODE 句の PITCH 指定)	0.01～24.00cpi の文字間隔を cpi 単位で指定します。文字間隔の指定を省略した場合、文字サイズの指定に合わせた文字間隔で印字します。文字間隔と文字サイズを両方省略した場合、以下の文字間隔で印字します。 ・ 日本語項目：6.00cpi ・ 英数字項目：10.00cpi

注: 印字文字の大きさと形態によって決定される間隔を以下の表に示します。

指定方法の詳細については、COBOL 文法書の「5.4.3 CHARACTER TYPE 句」および「4.2.3.11 PRINTING MODE 句」を参照してください。

表: 印字文字の大きさ/形態と文字間隔の関係

文字の大きさ	文字の形態							
	全角	半角	長体	半長体	平体	半平体	倍角	半倍角
MODE-1 (12 ポイント)	5	10	5	―	2.5	―	2.5	5
MODE-2 (9 ポイント)	8	16	8	―	4	―	4	8
MODE-3 (7 ポイント)	10	―	10	―	5	―	5	―
A (9 ポイント)	5	10	5	10	2.5	5	2.5	5
B (9 ポイント)	20/3	40/3	20/3	40/3	10/3	20/3	10/3	20/3
X-12P (12 ポイント)	5	10	5	10	2.5	5	2.5	5
X-9P (9 ポイント)	8	16	8	16	4	8	4	8
X-7P (7 ポイント)	10	20	10	20	5	10	5	10
C (9 ポイント)	7.5	15	7.5	15	3.75	7.5	3.75	7.5
D-12P (12 ポイント)	6	12	6	12	3	6	3	6
D-9P (9 ポイント)	6	12	6	12	3	6	3	6

フォームオーバレイパターン

フォームオーバレイパターンは、あらかじめ罫線や見出し文字など帳票の固定部分を設定するときに使用します。1 ページ分の出力データとフォームオーバレイパターンを合成して印字することにより、帳票印刷を簡単に行うことができます。

フォームオーバレイパターンは、FORM オーバレイオプションを使って画面イメージで簡単に作成することができます。また、1 つのフォームオーバレイパターンを複数のプログラムで使用したり、他システムで作成したものを使用したりすることもできます。

フォームオーバレイパターンの作成方法については、"FORM ユーザーズガイド"、"FORM ヘルプ"および "PowerFORM ヘルプ"（注）を参照してください。また、フォームオーバレイパターンを使った帳票印刷については、フォームオーバレイおよび FCB を使う方法を参照してください。

注: PowerFORM でオーバレイを作成時に参照してください。

NCOBU00296

FCB

FCB は、1 ページ分の行数、行間隔および印字開始位置を定義します。FCB を使用することにより、1 ページの行数、行間隔および印字開始位置を変更することができます。

デフォルト FCB 名が指定された場合(@DefaultFCB_Name=FCBxxxx)、または I 制御レコードで FCB 名が指定された場合、FCB 情報を外部から指定できます。

FCB 情報は、アプリケーション構成ファイルに以下の形式で指定します。

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="FCBxxxx" value="FCB 制御文" />
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

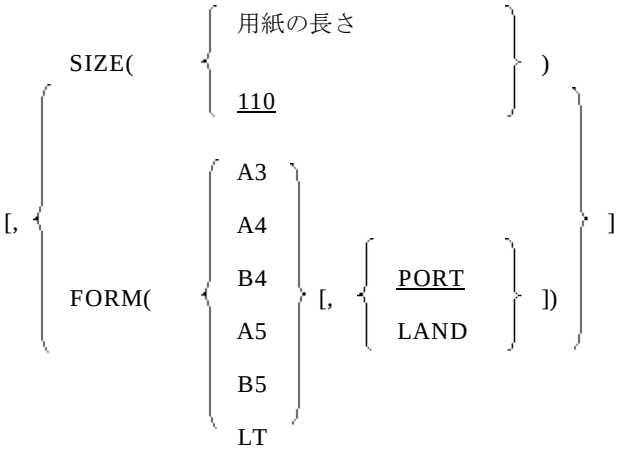
xxxx

環境変数情報@DefaultFCB_Name=FCBxxxx に指定した FCB 名(xxxx の部分)または I 制御レコードに指定した FCB 名

FCB 制御文の形式を以下に示します。

[LPI((行スペース[,行数])[,(行スペース[,行数])]…)]

[,CHm(行番号[,行番号]…)[,CHm(行番号[,行番号]…)]…]



LPI 情報

用紙の先頭から順に、(行スペース,行数)の形式で指定できます。行スペースには、LPI の単位で、6、8、12 のどれかを指定します。

CH 情報

CHANNEL-01～12 までに対して、行番号を指定できます。CHANNEL-01～12 は、それぞれチャネル 1 からチャネル 12 に割り当てた行番号にスキップすることを意味します。通常、チャネル 1(CHANNEL-01)は、印字可能な最初の行を識別するために使用されます。

SIZE 情報

SIZE オペランドでは、帳票の縦方向のサイズを 1/10 インチ単位で指定します。指定可能な値は、35～159(3.5 インチ～15.9 インチ)です。省略した場合、110(11 インチ)が有効になります。

FORM 情報

FORM オペランドでは、帳票を定型サイズで指定します。定型サイズでは、PORT(ポートレート:向きが縦)と LAND(ランドスケープ:向きが横)の用紙の向きごとに、縦方向のサイズが一意に決定します。

なお、デフォルト FCB 名の指定方法については、@DefaultFCB Name (デフォルト FCB 名の指定)を参照してください。また、I 制御レコードでの FCB 名の指定方法については フォームオーバレイおよび FCB を使う方法を参照してください。

デフォルト FCB の指定および I 制御レコードの FCB の指定が省略された場合、NetCOBOL for .NET ランタイムシステムは以下のデフォルト情報をもとに動作します。

```
LPI ((6,66)),CH1(4),SIZE(110)
```

6LPI×11 インチ(66 行)

チャネル番号	01	02	03	04	05	06	07	08	09	10	11	12
行位置	4	10	16	22	28	34	40	46	66	52	58	64

このデフォルト情報と利用者が使用する物理用紙が合致していない場合、デフォルト FCB または I 制御レコードを使用して物理用紙に合致した FCB を指定しなければなりません。

FCB の指定は、プリンタから供給される用紙のサイズや向きを決定するものではありません。 用紙サイズおよび印刷形式の指定は、I 制御レコードを使用して決定します。詳細については、I 制御レコードを参照してください。

FCB の必要性(重要)

FCB は、物理ページの用紙サイズに合わせて NetCOBOL for .NET ランタイムシステムがページ制御および行制御を行うために必ず必要な情報です。

ページ制御および行制御とは、COBOL プログラムの印刷要求(WRITE 文の ADVANCING 句の指定)に応じて出力する印刷データをページ内のどの位置に配置させるのか、改ページが必要かどうかをて NetCOBOL for .NET ランタイムシステムが自動的に制御する処理を意味しています。

例えば、利用者が実際に次のような帳票印刷を行うものと仮定した場合、

物理用紙サイズ	A4
印刷形式	縦向き(ポートレートモード)
ページ内の先頭行位置	1 行目
行間隔	6LPI(1 インチに 6 行印刷できる間隔)

FCB 制御文の指定は以下のように行います。

```
LPI((6)),CH1(1),FORM(A4,PORT)
```

この場合、て NetCOBOL for .NET ランタイムシステムは、この FCB 制御文を解析し COBOL の WRITE～ADVANCING 指定の要求にしたがって次のような制御を行います。

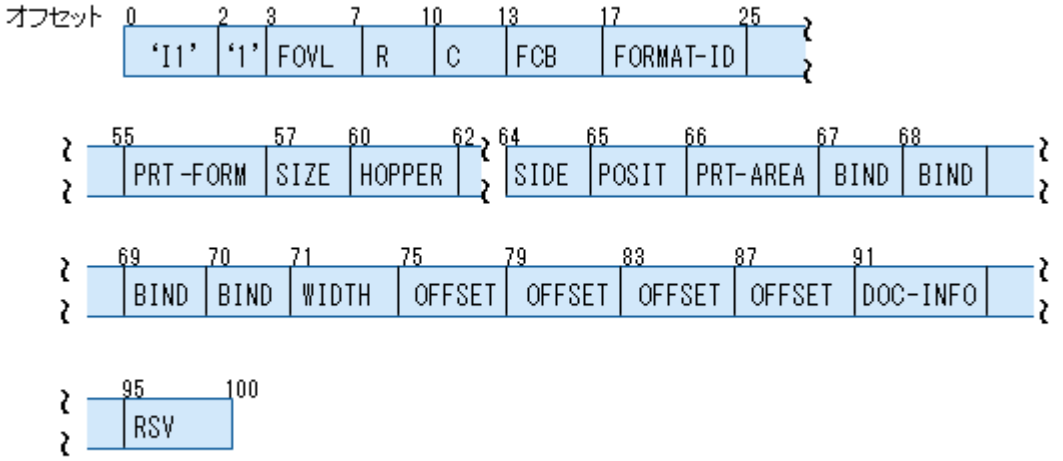
- WRITE 文で ADVANCING PAGE(改ページ)が要求された場合、1 枚ページを送って CH1 オペランドで指定されたページの先頭行である 1 行目に印刷行を位置付けます。
- WRITE 文で ADVANCING n LINES が要求された場合、LPI オペランドで指定された行間隔である 6LPI 単位で行間を計算し指定された行数分だけ行送りを行います。
- FORM オペランドの情報から A4 用紙を縦向き(ポートレートモード)に使用した場合のページ内に印刷可能な行数を計算します。このため、COBOL の WRITE 文で明に ADVANCING PAGE(改ページ要求)が行われない場合でも、ページ内の印刷可能な行数を超える印刷要求に対して自動的に改ページする制御を行います。

このように、COBOL プログラムで帳票印刷を行う場合、実際に使用する物理用紙サイズに対応した FCB の指定が必ず必要です。物理用紙サイズと FCB の指定が合致していない場合、意図した設計どおりの印刷結果が得られないことがありますので十分注意してください。

I 制御レコード

レコード内の各領域への設定値の組合せ条件や各フィールドの詳細な説明およびここで説明していない領域については、COBOL 文法書の「F.1 I 制御レコード」を参照してください。なお、有効となる機能が、使用する印刷装置によって異なる場合があるため、各印刷装置の説明書を参照してください。

I 制御レコードの形式を以下に示します。




```
01  I制御レコード.
    03  識別子                 PIC X(2) VALUE "I1".
    03  形式                   PIC X(1) VALUE "1" .
    03  オーバレイ名           PIC X(4).                → FOVL
    03  オーバレイ焼付け回数   PIC 9(3).                → R
    03  複写数                 PIC 9(3).                → C
    03  ＦＣＢ名               PIC X(4).                → FCB
    03  画面帳票定義体名       PIC X(8).                → FORMAT-ID
    03                         PIC X(30).
    03  印刷形式               PIC X(2).                → PRT-FORM
    03  用紙サイズ             PIC X(3).                → SIZE
    03  用紙供給口             PIC X(2).                → HOPPER
    03                         PIC X(2).
    03  印刷面                 PIC X.                  → SIDE
    03  印刷面位置付け         PIC X.                  → POSIT
    03  印字禁止域             PIC X.                  → PRT-AREA
    03  とじしろ方向.                                 → BIND
        04                     PIC X    OCCURS 4 TIMES.
    03  印字位置情報.
        04  とじしろ幅         PIC 9(4).                → WIDTH
        04  印刷原点位置.
            05                 PIC 9(4) OCCURS 4 TIMES.    → OFFSET
    03  文書名識別情報         PIC X(4).                → DOC-INFO
    03                         PIC X(5) VALUE SPACE.      → RSV
```

以降の説明中で"ファイルオープン時の指定"とある場合、そのファイルがオープンされたときのシステムが持つ印刷環境のことを指します。

FORMAT 句付き印刷ファイルの場合、帳票定義体の指定値が有効になります。帳票定義体の指定がない場合は、プリンタ情報ファイルに指定された情報が有効になります。プリンタ情報ファイルの詳細については、"MeFt ユーザーズガイド"を参照してください。

FOVL (フォームオーバレイパターン名)

使用するフォームオーバレイパターン名を指定します。オーバレイグループが指定された場合、先頭のオーバレイパターンに対して単一オーバレイの処理を行います。このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

R (フォームオーバレイパターン焼付け回数)

フォームオーバレイパターンの焼付け回数(0～255)を指定します。
ただし、本システムでは複写数と同じ値が指定されたものと扱われます。

C (複写数)

ページ単位の複写数(0～255)を指定します。0 を指定するとファイルオープン時の指定が有効となります。

両面印刷指定時は、複写数の指定は有効となりません。1 が指定されたものとみなします。また、複写数の指定はプリンタドライバが複写機能を持っている場合だけ有効となります。

FCB (FCB 名)

FCB 名を指定します。このフィールドが空白の場合、環境変数情報@DefaultFCB_Name に指定された FCB 名が有効となります。なお、環境変数情報の指定については、@DefaultFCB_Name (デフォルト FCB 名の指定)を参照してください。デフォルト FCB 名が指定されていない場合、NetCOBOL for .NET ランタイムシステムのデフォルト情報が有効となります。デフォルト情報の詳細については、FCB を参照してください。FCB 名は帳票定義体名と同時に指定することはできません。

FORMAT-ID (帳票定義体名)

I 制御レコードで指定する情報を適用する帳票定義体名を指定します。この指定により、固定形式ページとなります。このフィールドが空白の場合、不定形式ページとなります。FORMAT-ID は FCB 名と同時に指定することはできません。

帳票定義体名の指定は、FORMAT 句付き印刷ファイルの場合だけ有効となります。 使用方法については、プログラムの記述を参照してください。

PRT-FORM (印刷形式)

印刷形式を指定します。設定可能な値を以下に示します。

- "P"(ポートレートモード)
- "L"(ランドスケープモード)
- "LP"(ラインプリンタモード)
- "PZ"(縮小印刷のポートレートモード)

- "LZ"(縮小印刷のランドスケープモード)

このフィールドが空白の場合、用紙サイズの指定(SIZE)にも空白が指定されていなければなりません。この場合、ファイルオープン時の指定が有効となります。

- FORMAT 句なし印刷ファイルの場合、"PZ"、"LZ"指定は、80%縮刷をサポートしているプリンタおよびプリンタドライバに対してだけ有効となります。80%縮刷をサポートしていないプリンタおよびプリンタドライバでは、"PZ"、"LZ"の縮小指定は無視され、"P"、"L"が指定されたものとみなします。また、"LP"指定は、連続用紙(ストックフォーム)サイズで設計された帳票を、A4 用紙を横向きに使用したときのサイズに文字間および行間を縮小して印刷します。この場合、文字サイズ自身は縮小されませんので、文字の大きさや形態(平体や倍角など)によっては、隣接する文字同士が重なって印刷されることがあります。

  なお、"PZ"、"LZ"、"LP"指定で、フォームオーバレイは縮小されません。

- プリンタから供給される用紙の印刷形式(用紙の方向)は、本フィールドの指定により決定します。本フィールドの指定を省略した場合、以下の解釈になります。

  - FORMAT 句なし印刷ファイル

    プリンタドライバでの設定値

  - FORMAT 句付き印刷ファイル

    帳票定義体での設定値→プリンタ情報ファイルでの設定値→プリンタドライバでの設定値

  なお、実際に使用される印刷形式に合わせて、FCB 制御文を定義・指定する必要があります。詳細については、FCB を参照してください。

SIZE (用紙サイズ)

用紙サイズを指定します。指定可能なサイズを以下に示します。

- "A3"
- "A4"
- "A5"
- "B4"
- "B5"
- "LTR"

このフィールドが空白の場合、印刷形式(PRT-FORM)にも空白が指定されていなければなりません。この場合、ファイルオープン時の指定が有効となります。

また、FORMAT 句なし印刷ファイルでは、上記用紙サイズに加えて任意の 3 文字以内の文字列を指定することができます。任意の 3 文字以内の文字列は、連帳の用紙サイズなど上記以外の用紙サイズを動的に変更する場合に、環境変数情報@PRN_FormName_xxx に用紙名を対応付けて指定します。なお、環境変数情報の指定方法については、@PRN_FormName_xxx (用紙名の指定) を参照してください。

- 任意の 3 文字以内の文字列は、FORMAT 句付き印刷ファイルには指定できません。FORMAT 句付き印刷ファイルを利用して、連帳の用紙サイズなどあらゆる用紙サイズを指定する場合、本フィールドに空白を指定し、プリンタ情報ファイルで実際の用紙サイズを指定します。詳細については、"MeFt ユーザーズガイド"を参照してください。
- プリンタから供給される用紙サイズ(用紙の種類)は、本フィールドの指定により決定します。本フィールドの指定を省略した場合、以下の解釈になります。
  - FORMAT 句なし印刷ファイル

    プリンタドライバでの設定値
  - FORMAT 句付き印刷ファイル

    帳票定義体での設定値→プリンタ情報ファイルでの設定値→プリンタドライバでの設定値

  なお、実際に使用される用紙サイズに合わせて、FCB 制御文を定義・指定する必要があります。詳細は、FCB を参照してください。

HOPPER (用紙供給口)

用紙供給時の用紙供給口を指定します。設定可能な値を以下に示します。

- "P1"(主供給口 1)
- "P2"(主供給口 2)
- "S"(副供給口)
- "P"(任意の供給口)

このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

用紙供給口は用紙サイズ指定に従ってシステムが自動的に選択するため、アプリケーションから I 制御レコードを利用して、用紙供給口を変更することはできません。

SIDE (印刷面)

印刷面を指定します。設定可能な値を以下に示します。

- "F"(片面印刷)
- "B"(両面印刷)

このフィールドが空白の場合、ファイルオープン時の指定が有効となります。

両面印刷指定時は、複写数の指定は有効となりません。1 が指定されたものとみなします。

POSIT (印刷面位置付け)

両面印刷時の印刷開始面を指定します。指定可能な値を以下に示します。

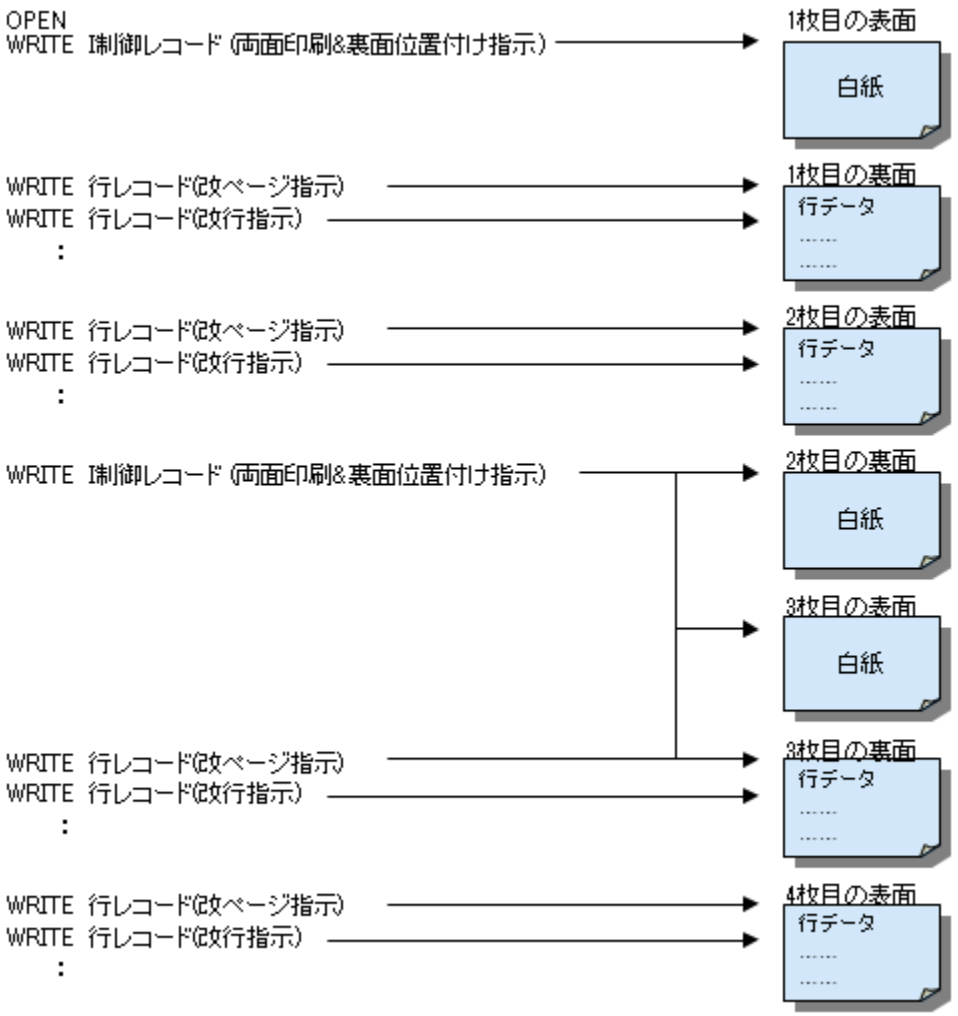
- "F"(表面位置付け)
- "B"(裏面位置付け)

空白が指定された場合は、表面に位置付けられます。

注意

- プリンタまたはプリンタドライバで白紙ページの扱いが指定できる場合、この指定に"白紙を印刷しない"を設定すると、印刷面位置付け(POSIT)機能は、正しく制御されないことがあります。印刷面位置付け(POSIT)機能を利用する場合は、必ず"白紙を印刷する"を設定してください。
- I 制御レコードを出力すると、それまでの印刷ジョブが終了し、新しい印刷ジョブが開始されます。したがって、I 制御レコード出力直前の印刷面が表面だったのか裏面だったのかに関係なく、それまでの用紙が出力され、新しい用紙への印刷が開始されます。

例:



PRT-AREA (印字禁止域)

印字禁止領域の設定を行う("L")か、行わない("N")かを指定します。このフィールドが空白の場合、ファイルオープン時の指定が有効になります。

ただし、帳票定義体を使用する場合は無効となります。

BIND (とじしろ方向)

連続して出力される複数ページを製本するときのとじしろ方向を指定します。とじしろ方向は複数ページに対して意味を持つ指定であるため、このフィールドが空白の場合、直前のページでのとじしろ方向がそのまま引き継がれます。

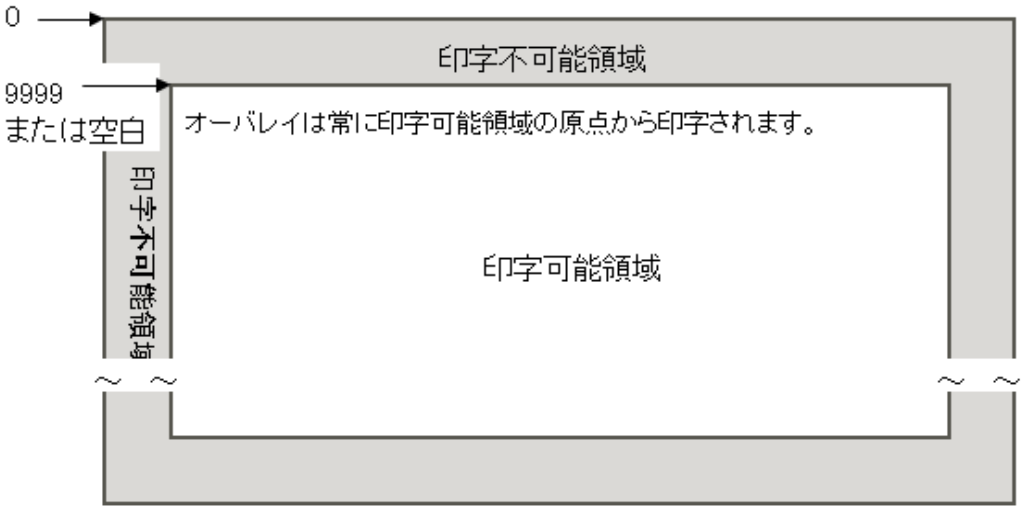
とじしろ方向の指定は、左とじ指定('L')および上とじ指定('U')だけ有効となります。したがって、右とじ指定('R')および下とじ指定('D')は無視されます。右方向および下方向にとじしろ幅を設ける場合、アプリケーションで意識する必要があります。

WIDTH (とじしろ幅)

とじしろ幅を 0～9999(単位:1/1440 インチ)で指定します。

FORMAT 句なし印刷ファイルの場合、本指定は、環境変数情報@CBR_OverlayPrintOffset の指定値により有効範囲が異なります。環境変数情報@CBR_OverlayPrintOffset に"VALID"が指定されている場合は、フォームオーバレイおよび行レコードに対して有効となります。"INVALID"が指定された場合および指定が省略された場合は、行レコードに対してだけ有効となります。この場合、フォームオーバレイパターンには有効となりませんので注意してください(行レコードとオーバレイが正しく合成されず、ずれて印刷されることがあります)。

また、0 が指定された場合の印刷開始位置は、用紙の左端角となります。この場合、プリンタの印刷不可能な領域に該当する可能性があり、印刷データが欠落するおそれがありますので注意してください。印字可能領域の左端角を原点として印字する場合は、本フィールドに 9999 または空白を指定してください。詳細については、印刷不可能な領域を参照してください。



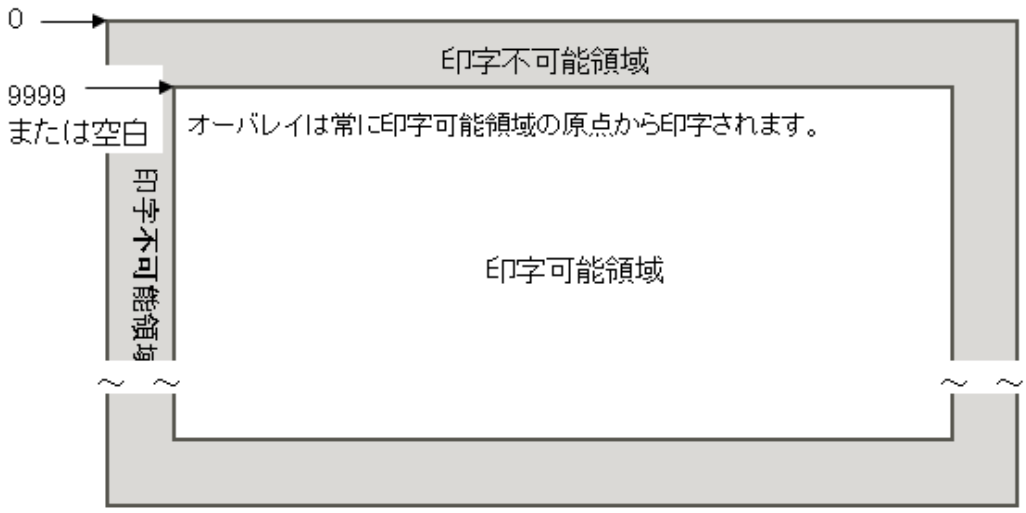
FORMAT 句付き印刷ファイルの場合、このフィールドが 9999 または空白のとき、ファイルオープン時の指定が有効となります。

OFFSET (印刷原点位置)

印刷原点位置を 0～9999(単位:1/1440 インチ)で指定します。

FORMAT 句なし印刷ファイルの場合、本指定は、環境変数情報@CBR_OverlayPrintOffset の指定値により有効範囲が異なります。環境変数情報@CBR_OverlayPrintOffset に"VALID"が指定されている場合は、フォームオーバレイおよび行レコードに対して有効となります。"INVALID"が指定された場合および指定が省略された場合は、行レコードに対してだけ有効となります。この場合、フォームオーバレイパターンには有効となりませんので注意してください(行レコードとオーバレイが正しく合成されず、ずれて印刷されることがあります)。

また、0 が指定された場合の印刷開始位置は、用紙の左端角となります。この場合、プリンタの印刷不可能な領域に該当する可能性があり、印刷データが欠落するおそれがありますので注意してください。印字可能領域の左端角を原点として印字する場合は、本フィールドに 9999 または空白を指定してください。詳細については、印刷不可能な領域を参照してください。



FORMAT 句付き印刷ファイルの場合、このフィールドが 9999 または空白のとき、ファイルオープン時の指定が有効となります。なお、PRT-FORM(印刷形式)と同時に指定できますが、LP(ラインプリンタモード)を指定した場合、OFFSET の指定は無効となります。また、縮小印刷形式を指定した場合の OFFSET は縮小前の長さで指定します。

DOC-INFO (文書名識別情報)

文書名を識別するための情報を指定します。指定可能な値は、4 文字以内の任意の英数字です。ここで指定した値は、実行時に環境変数情報@CBR_DocumentName_xxxx の xxxx 部分に置き換えて設定しておく必要があります。また、本環境変数情報名には、128 バイト以内の英字/数字/カナ/日本語の組合せで構成される文書名を対応付けておかなければなりません。なお、環境変数情報の指定形式については、@CBR_DocumentName_xxxx (I制御レコードによる文書名の指定)を参照してください。

DOC-INFO フィールドの指定は、FORMAT 句付き印刷ファイルには指定できません。 FORMAT 句付き印刷ファイルを利用して、プリントマネージャーに任意の文書名を表示させる場合、本フィールドに空白を指定し、プリンタ情報ファイルで文書名を指定します。 詳細については、"MeFt ユーザーズガイド"を参照してください。

RSV (システム使用領域)

システムの使用する領域です。空白を設定しておきます。

制御レコードの有効範囲

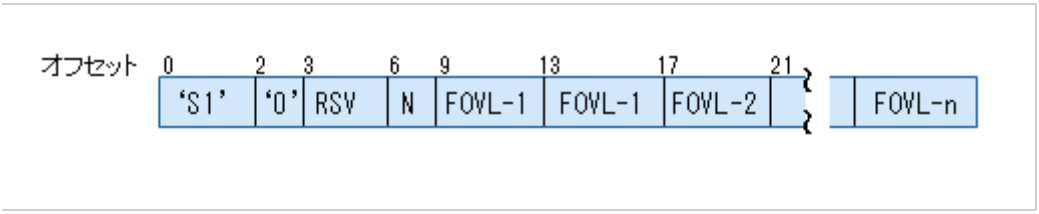
I 制御レコードが有効となる範囲は、そのレコードが出力されてから、次の I 制御レコードが出力されるまでです。ただし、I 制御レコードで指定するフォームオーバレイパターン名は、次の I 制御レコードまたは S 制御レコードが出力されるまで有効です。

制御レコードの各フィールドに指定された値に誤りがあると、FILE STATUS 句または誤り手続きの指定に関係なく、プログラムの実行を中断し、終了処理を行います。

S 制御レコード

レコード内の各領域への設定値の組合せ条件や各フィールドの詳細な説明およびここで説明していない領域については、COBOL 文法書の「F.2 S 制御レコード」を参照してください。なお、有効となる機能が使用する印刷装置により異なる場合があるため、各印刷装置の説明書を参照してください。

S 制御レコードの形式を以下に示します。




```
01  S制御レコード.
  03  識別子              PIC X(2) VALUE "S1".
  03  形式                PIC X(1) VALUE "0" .
  03                       PIC X(3) VALUE SPACE.      → RSV
  03  個数                PIC 9(3).                  → N
  03  オーバレイ名１      PIC X(4).                  → FOVL-n
      :
```

RSV (システム使用領域)

システムの使用する領域です。空白を設定しておきます。

N (フォームオーバレイパターン名の個数)

指定するフォームオーバレイパターン名の数を指定します。

FOVL-n (フォームオーバレイパターン名)

フォームオーバレイパターン名を設定します。ただし、本システムでは、先頭に指定されたフォームオーバレイパターン名だけが有効になります。

I 制御レコードが有効となる範囲は、そのレコードが出力されてから、次の I 制御レコードが出力されるまでです。ただし、I 制御レコードで指定するフォームオーバレイパターン名は、次の I 制御レコードまたは S 制御レコードが出力されるまで有効です。

S 制御レコードが有効となる範囲は、そのレコードが出力されてから、次の S 制御レコードまたは I 制御レコードが出力されるまでです。

制御レコードの有効範囲

S 制御レコードが有効となる範囲は、そのレコードが出力されてから、次の S 制御レコードまたは I 制御レコードが出力されるまでです。

制御レコードの各フィールドに指定された値に誤りがあると、FILE STATUS 句または誤り手続きの指定に関係なく、プログラムの実行を中断し、終了処理を行います。

NCOBU00299

帳票定義体

FORM を使って帳票を設計すると、帳票定義体が作成されます。NetCOBOL for .NET では、帳票定義体に定義したデータ項目をプログラムに取り込み、そのデータ項目に値を設定して出力することにより、帳票を印刷することができます。また、帳票定義体に、フォームオーバレイパターンを取り込むこともできます。

帳票定義体を使って帳票の印刷を行う場合は、MeFt が必要です。MeFt を使用する場合、MeFt が使用するプリンタ情報ファイルが必要となります。プリンタ情報ファイルの詳細については、"MeFt ユーザーズガイド"を参照してください。

帳票定義体の作成方法については"FORM ユーザーズガイド"、"FORM ヘルプ"または"PowerFORM ヘルプ"を参照してください。

帳票定義体を使った帳票印刷については、帳票定義体を使う印刷ファイルの使い方または表示ファイル(帳票印刷)の使い方を参照してください。

NetCOBOL for .NET で使う帳票定義体の拡張子を除いたファイル名は、英字で始まる 8 文字以内の英数字となるようにしてください。

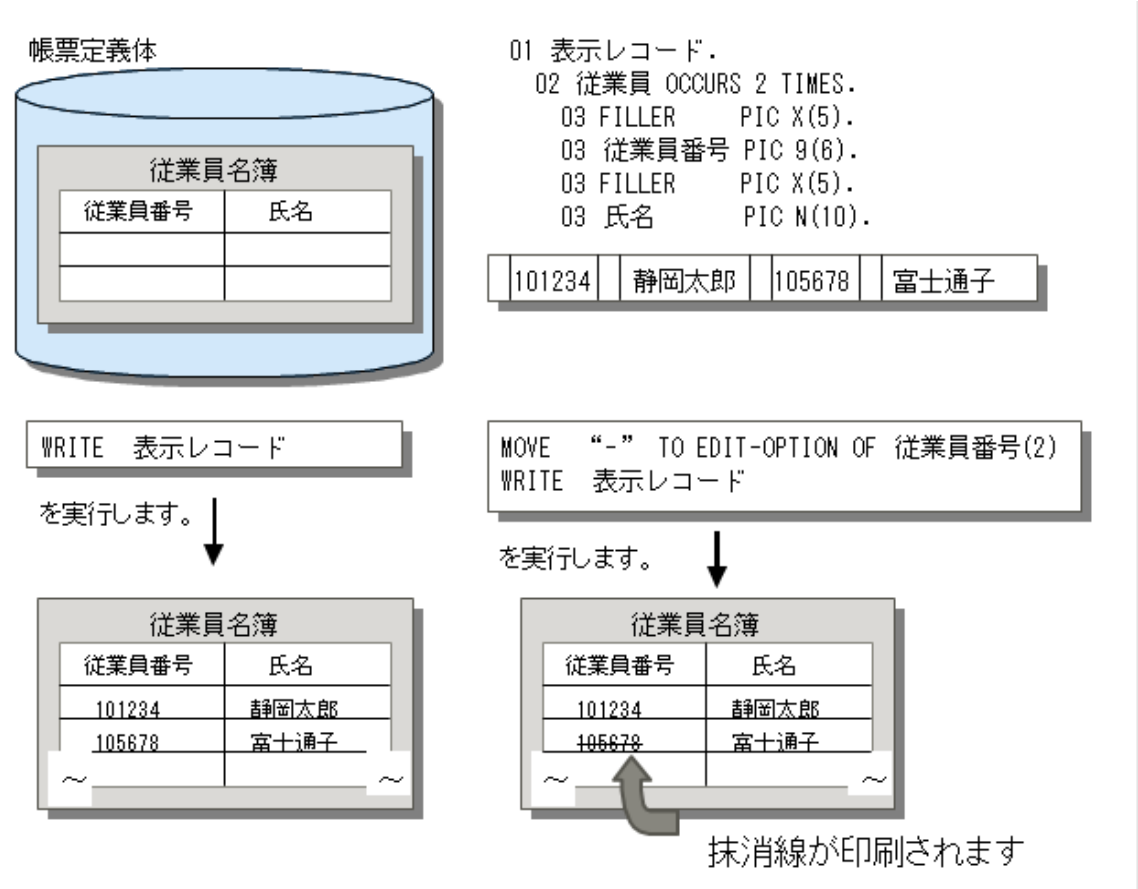
特殊レジスタ

FORMAT 句付き印刷ファイルおよび表示ファイル(帳票印刷)では、帳票定義体で定義されている出力データの属性を、COBOL の特殊レジスタを使って、プログラムの実行中に変更することができます。

帳票機能の特殊レジスタには、次の 3 種類があります。

EDIT-MODE	出力処理の対象にする/しないなどを指定します。
EDIT-OPTION	下線付き、抹消線付きなどを指定します。
EDIT-COLOR	色を指定します。

これらの特殊レジスタは、帳票定義体で定義したデータ名で修飾して使います。たとえば、データ名 A の色属性の設定は、"EDIT-COLOR OF A"のように記述します。各特殊レジスタに設定する値については、"MeFt ユーザーズガイド"を参照してください。



注意

項目制御部なしを指定した帳票定義体では、特殊レジスタを使用することはできません。 また、1 つのプログラム中で項目制御部なしを指定した帳票定義体と項目制御部を指定した帳票定義体を混在して使うことはできません。

NCOBU00301

印字文字の配置座標

配置座標は、プリンタ解像度(DPI)をアプリケーションで指定された文字間隔(cpi)または行間隔(LPI)で除算し、除算した値をもとに求めます。

FORMAT 句なし印刷ファイルを使用して帳票印刷を行う場合、除算した余りを切り捨てた座標に配置する印字方法と、除算して割り切れないときに 1 インチ単位内で補正した座標に配置する印字方法があります。

除算して余りを切り捨てる方法では、切り捨てたドット数の分だけ文字間隔(cpi)や行間隔(LPI)がそれぞれ左方向、上方向に詰めて印字されます。出力するプリンタの解像度に依存しない印刷結果を得るために、1 インチ単位内で補正した座標に配置する方法を選択することをおすすめします。

印字文字の配置座標の指定方法については、@CBR_PrintTextPosition (文字配置座標の計算方法の指定)を参照してください。

印刷情報ファイル

印刷情報ファイルとは、FORMAT 句なし印刷ファイルを利用して帳票出力を行う場合に、出力する帳票の状態制御情報を設定するテキスト形式のファイルです。

印刷情報ファイルの書式および出力される帳票の状態を制御する情報は、次のとおりです。

- 印刷情報ファイルに指定する絶対パスのファイル名および絶対パス名は、特に断りのない限り二重引用符(")で囲まないでください。
- 行の先頭に;(セミコロン)がある場合、セミコロンから改行までの間はコメントとして認識されます。

```
[PrintInformation]

TextAlign=      { TOP
                { BOTTOM

DocumentName= 文書名


OverlayPrintOffset= { VALID
                    { INVALID

PRTOUT=         { LPTn:
                { COMn:
                { PRTNAME: プリンタ名

PAPERSIZE={A3 | A4 | A5 | B4 | B5 | LTR | 任意}

PRTFORM={ P | L | PZ | LZ | LP }

FOVLDIR= オーバレイファイルのフォルダ名

FOVLTYPE= オーバレイファイルの形式

FOVLNAME= オーバレイファイル名

OVD_SUFFIX= オーバレイファイル(拡張子)名
```

セクション名(固定)[必須]

セクション名"[PrintInformation]"は、印刷情報ファイルを識別するための名前で、常に固定です。

TextAlign(上端/下端合わせを指定)[省略可]

行内に配置する印字文字の位置を指定します。 印字する文字を文字セルの左上端を基準点として行内に上端合わせで配置する(TOP)か、左下端を基準点として行内に下端合わせで配置する(BOTTOM)かを指定します。 なお、本指定は、実行環境変数@CBR_TextAlign の指定よりも優先されます。

DocumentName(文書名指定)[省略可能]

Windows システムが提供するプリントマネージャーに文書名を表示したい場合に指定します。 指定可能な文書名は、128 バイト以内の英字/数字/カナ/日本語の組合せでなければなりません。 なお、実行環境変数@CBR_DocumentName_xxxx の指定が有効な場合、本指定は無視されます。

OverlayPrintOffset(オーバレイ原点移動機能を有効または無効にする指定)[省略可能]

FORMAT 句なし印刷ファイルで、I 制御レコードに指定されたとじしろ方向(BIND)、とじしろ幅(WIDTH)および印刷原点位置(OFFSET)機能をフォームオーバレイに対しても有効にする(VALID)か、無効にする(INVALID)かを指定します。 本指定が省略された場合、"INVALID"が指定されたものとみなされます。 なお、本指定は、実行環境変数@CBR_OverlayPrintOffset の指定よりも優先されます。

以下に、原点オフセットを設けない通常の印刷結果と、上方向および左方向にそれぞれ 1 インチの原点オフセットを設けた場合の印刷結果を例にとり、VALID/INVALID 指定時(または省略時)の印刷結果の相違について図示します。

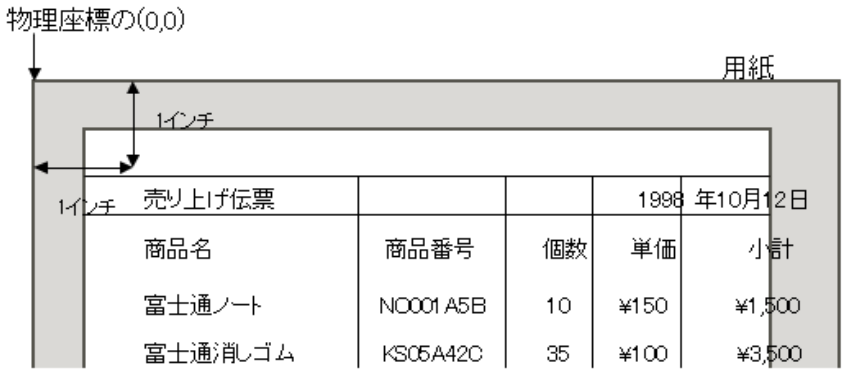
※ 以下の図では、罫線=オーバレイ、文字=行レコードを示しています。

[印刷原点位置を指定しない場合の印刷結果]

[X,Y 座標にそれぞれ 1 インチの原点オフセットを設けた場合の印刷結果]

- VALID 指定の場合

・ INVALID 指定の場合



PRTOUT(出力プリンタ名を指定)[省略可能]

プログラムで使用するプリンタを指定します。 以下のような場合に指定します。

- ファイル識別名に対して割り当てた出力先("LPTn:"/"COMn:"/"PRTNAME:プリンタ名")を変更したい場合。
- ファイル識別名に対する出力先の割り当てを省略した場合に有効となる出力先を指定したい場合。

本指定は、プログラムを変更することなく出力先を変更することが可能になるため、以下の場合は特に便利な指定となります。

- ASSIGN 句にデータ名指定または定数指定を使用して、プログラム中に直接出力先を指定している場合

指定可能な出力先は、ファイル識別名の割り当て方法の場合と同じです。 実際に帳票出力を行うプリンタが接続されているローカルプリンタポート名(LPTn:)/通信ポート名(COMn:)/プリンタの名前(PRTNAME:プリンタ名)を指定します。 ただし、ここでは印刷情報ファイル名(INF(～))およびフォントテーブル名(FONT(～))は指定できません。

PRTOUT 指定は、省略することも可能です。 PRTOUT 指定を省略した場合、実行環境変数 "ファイル識別名" に対する指定が有効となります。ファイル識別名および印刷情報ファイルの PRTOUT 指定の両方にそれぞれ異なるローカルプリンタポート名(LPTn:)/通信ポート名(COMn:)/プリンタの名前(PRTNAME:プリンタ名)を指定した場合、印刷情報ファイルの PRTOUT 指定が優先されます。両方とも省略された場合または指定に誤りがある場合は、実行時エラーとなります。

PAPERSIZE

デフォルト用紙サイズを指定します。指定可能なサイズは、A3/A4/A5/B4/B5/LTR です。

FORMAT 句なし印刷ファイルでは、上記用紙サイズに加えて任意の 3 文字以内の文字列を指定することができます。 任意の 3 文字以内の文字列は、連帳の用紙サイズなど上記以外の用紙サイズを動的に変更する場合に、 環境変数情報@PRN_FormName_xxx に用紙名を対応付けて指定します。 環境変数情報の指定方法については、@PRN_FormName_xxx (用紙名の指定)を参照してください。

なお、本指定が省略された場合、プリンタドライバの設定値が有効となります。

[注意] デフォルト用紙サイズを変更する場合は、実際に使用される用紙サイズに合わせて、 FCB 制御文を定義・指定する必要があります。 詳細は、FCB を参照してください。

PRTFORM

デフォルトの印刷形式を指定します。指定可能な形式は、P / L / PZ / LZ / LP です。

なお、本指定が省略された場合、プリンタドライバでの設定値が有効となります。

[注意]

- デフォルトの印刷形式を指定する場合は、用紙サイズも指定する必要があります。 指定可能な用紙サイズと印刷形式の組合せについては、 COBOL 文法書の「付録 F 制御レコード」の“表 F.1 用紙サイズと印刷形式の組合せ”を参照してください。
- デフォルトの印刷形式を変更する場合は、指定した形式に合わせて、FCB 制御文を定義・指定する必要があります。 詳細は、FCB を参照してください。
- "PZ"、"LZ"指定は、80%縮刷をサポートしているプリンタおよびプリンタドライバに対してだけ有効となります。 80%縮刷をサポートしていないプリンタおよびプリンタドライバでは、"PZ"、"LZ"の縮小指定は無視され、 "P"、"L"が指定されたものとみなします。 また、"LP"指定は、連続用紙(ストックフォーム)サイズで設計された帳票を、 A4 用紙を横向きに使用したときのサイズに文字間および行間を縮小して印刷します。 この場合、文字サイズ自身は縮小されませんので、文字の大きさや形態(平体や倍角など)によっては、 隣接する文字同士が重なって印刷されることがあります。 なお、"PZ"、"LZ"、"LP"指定で、フォームオーバレイは縮小されません。

FOVLDIR(フォームオーバレイパターンファイルを格納したフォルダ名)[省略可能]

フォームオーバレイパターンファイルの格納先フォルダを絶対パス名で指定します。 省略された場合、フォームオーバレイパターンの焼付けは行われません。 また、複数のフォルダを指定することはできません。

なお、本指定は、実行環境変数 FOVLDIR の指定よりも優先されます。

FOVLTYPE(フォームオーバレイパターンファイルのプレフィックス(形式)名)[省略可能]

フォームオーバレイパターンのファイル名の先頭 4 文字が"KOL5"(省略値)以外の場合、形式に、ファイル名の先頭 4 文字を指定します。 ファイル名に形式を持たない場合は、文字列 "None" を指定してください。

例: フォームオーバレイパターンファイルが"C:¥FOVLDATA¥KOL2OVD1.OVD"の場合

```
FOVLDIR=C:¥FOVLDATA
FOVLTYPE=KOL2
FOVLNAME=OVD1 または I/S 制御の FOVL フィールドで "OVD1" を指定して WRITE
```

なお、本指定は、実行環境変数 FOVLTYPE の指定よりも優先されます。

FOVLNAME(フォームオーバレイパターンファイルのファイル名)[省略可能]

フォームオーバレイパターンのファイル名から先頭 4 文字の形式部分を除いた、後半のファイル名を 4 文字以内で指定します。 本指定は、I/S 制御レコードが出力されていない場合、および I/S 制御レコードでオーバレイ出力が指示されていない(FOVL フィールドが空白指定)の場合に意味を持ちます。

例: フォームオーバレイパターンファイルが"C:¥FOVLDATA¥KOL2TEST.OVL"の場合

```
FOVLDIR=C:¥FOVLDATA
FOVLTYPE=KOL2
FOVLNAME=TEST かつ (I/S 制御レコードの WRITE なし
            または I/S 制御の FOVL フィールドに空白を指定して WRITE)
OVD_SUFFIX=OVL
```

なお、本指定は、実行環境変数 FOVLNAME の指定よりも優先されます。

OVD_SUFFIX(フォームオーバレイパターンファイルの拡張子名)[省略可能]

フォームオーバレイパターンファイルの拡張子が "OVD" (省略値)以外の場合、拡張子として使用する文字列を指定します。 ファイル名に拡張子を持たない場合は、文字列 "None" を指定してください。

例: フォームオーバレイパターンファイルが"C:¥FOVLDATA¥KOL5ABCD"の場合

```
FOVLDIR=C:¥FOVLDATA
FOVLTYPE=KOL5
FOVLNAME=ABCD または I/S 制御の FOVL フィールドで "ABCD" を指定して WRITE
OVD_SUFFIX=None
```

なお、本指定は、実行環境変数 OVD_SUFFIX の指定よりも優先されます。

NCOBU00303

印刷ファイル/表示ファイルの決定方法

COBOL プログラムで帳票印刷を行う場合、FORMAT 句なし印刷ファイル、FORMAT 句付き印刷ファイルまたは表示ファイルを使用します。

これらのファイル種別は、翻訳時に NetCOBOL for .NET コンパイラがソースプログラム中の特定の記述の有無を解析することにより決定付けられます。

以下に、特定の記述によって決定付けられるファイル種別の組み合わせを示します。

特定の記述

[1]: FORMAT 句

[2]: PRINTER 指定の ASSIGN 句

[3]: PRINTER-n 指定の ASSIGN 句

[4]: LINAGE 句

[5]: ADVANCING 指定の WRITE 文

[6]: ファイル参照子"GS-ファイル識別名"

特定の記述によって決定付けられるファイル種別の組合せ

上記[1]～[6]の記述によって決定付けられるファイル種別の組合せを下表に示します。

ファイル種別	特定の記述					
	[1]	[2]	[3]	[4]	[5]	[6]
FORMAT 句なし印刷ファイル(注)	×	○	○	○	○	×
FORMAT 句付き印刷ファイル	○	×	×	×	△	×
表示ファイル	△	×	×	×	×	○

○: ファイル種別を決定付ける記述。

△: 記述可能。ただし、ファイル種別を決定付ける条件にはならない。

×: 記述不可能。

注: [2]、[3]、[4]、[5]のどれか 1 つでも記述されていれば、FORMAT 句なし印刷ファイルであると決定付けられます。

サービス配下の注意事項(印刷時)

- サービス配下で COBOL プログラムを動作させる場合、通常サービスはシステムアカウントでプロセスを立ち上げ、COBOL プログラムをバックグラウンドジョブとして動作させます。この場合、システムアカウントに対するプリンタ情報が設定されていないと、COBOL ランタイムシステムはプリンタ名などの情報を取得することができず、印刷できません。
- ASP.NET からサーバ側のプリンタへ印刷する場合は、System ユーザ(Default User)に対して、プリンタ情報を設定する必要があります。 System ユーザに対するプリンタの設定については、Microsoft ホームページのサポート技術情報:KB419321(MSDN ライブラリがインストールされている場合、文書番号:J046633)"[IIS]ASP によるサーバ側での印刷方法について"を参照してください。
- FORMAT 句付き/なし印刷ファイルおよび表示ファイル(宛て先 PRT)を使用した COBOL プログラムをサービス配下で実行する場合、そのファイルの出力先への割当ては明にプリンタ名を指定してください。出力先の指定にデフォルトのプリンタ、ローカルプリンタポート名(LPTn:)および通信ポート名(COMn:)を指定した場合、印刷されないことがあります(ファイルの割当てエラーが発生します)。 また、ネットワークプリンタに印刷することはできません。

サービス配下で印刷可能なファイルの割当て例

- FORMAT 句なし印刷ファイルの場合
  - § ASSIGN 句の指定

  SELECT 印刷ファイル ASSIGN TO S-PRTF.
  - § 実行環境情報の指定

  PRTF=PRTNAME:FUJITSU FMLBP227
- FORMAT 句付き印刷ファイルの場合
  - § プリンタ情報ファイルの指定

  PRTDRV FUJITSU FMLBP227

サービスがシステムアカウントまたは特定のユーザアカウントでシステムに乗り込むかは、各サービスの仕様を確認してください。 なお、サービスによっては、[コントロールパネル]-[管理ツール]-[サービス]のプロパティの設定により、特定のユーザアカウントでログインするように設定できる場合があります。この場合、使用するプリンタドライバをインストールしたときのユーザアカウントを設定することで、印刷できることがあります。

- サービス配下で印刷ファイルを使用する場合、プリンタ用紙サイズは、以下に設定した情報が有効になります。
  - サービス配下の場合は、プリンタのプロパティに設定された情報
  - 通常のアプリ動作の場合は、プリンタの印刷設定に設定された情報

NCOBU00305

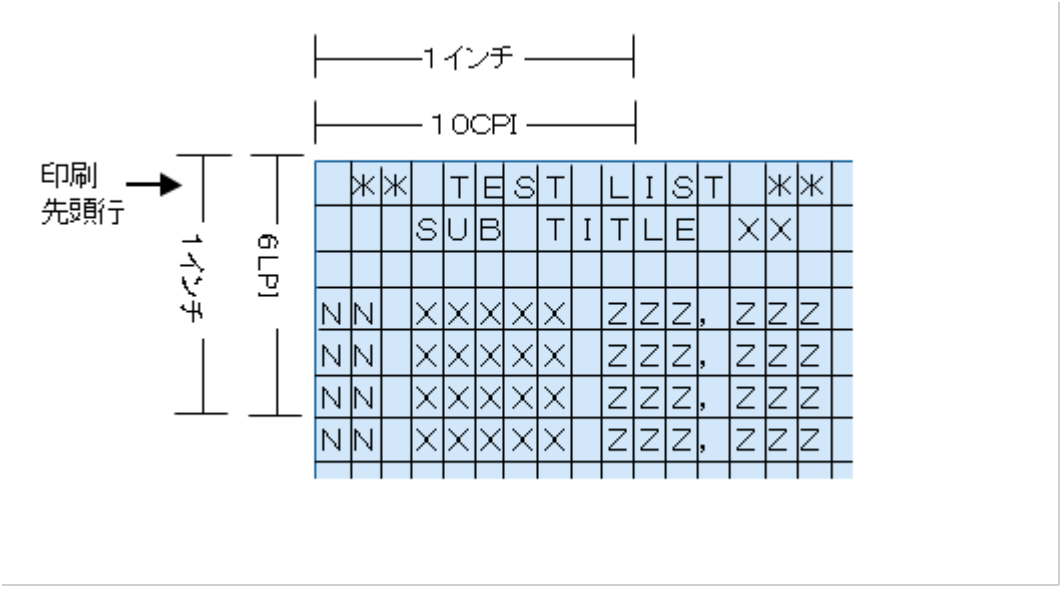
帳票設計

COBOL の帳票印刷には、行と桁の概念が不可欠です。特に FORMAT 句なし印刷ファイルのように行レコード主体で帳票印刷を行う場合、実際にプログラムを作成する前にきめ細かい帳票設計が必要です。

このため、実際にプログラミングに入る前に、スペーシングチャートのような設計用紙(または、FORM のようなグリッド表示可能なツール)を用いて、実際の印刷イメージで設計した後、この帳票設計をもとにプログラミングを行ってください。

スペーシングチャートの縦方向のマスは、COBOL の WRITE～ADVANCING の行制御および FCB 制御文の LPI オペランドや CH オペランドに対応(反映)させ、横方向のマスは PICTURE 句の桁数および CHARACTER TYPE 句の文字間隔(CPI)PRINTING POSITION 句の水平スキップの情報に対応(反映)させてください。また、帳票設計時は、縦方向は 1 インチ内に何行、横方向は 1 インチ内に何文字配置するかを把握してください。

例: [スペーシングチャートの例]



[FCB 制御文]

```
LPI ( ( 6 ) ) , CH1 ( 1 ) , FORM (…
```

[COBOL プログラム]

```
       SPECIAL-NAMES.
          PRINTING MODE PM1 IS FOR ALL
          AT PITCH 10.00 CPI.
      DATA DIVISION.
       FILE SECTION.
       FD  印刷ファイル.
       01 印刷レコード PIC X(80)
                    CHARACTER TYPE IS PM1.
      PROCEDURE DIVISION.
          OPEN OUTPUT 印刷ファイル.
          MOVE " ** TEST LIST **" TO 印刷データ.
          WRITE 印刷レコード FROM 印刷データ
                           AFTER ADVANCING PAGE.
          CLOSE 印刷ファイル.
```

印刷不可能な領域

用紙内で印刷可能な文字数

用紙内に印字可能な最大文字数は、用紙サイズ(横方向)および文字ピッチ(CPI)によって決まります。 たとえば、使用する用紙サイズが 15×11 インチの連続用紙で文字ピッチが 10CPI であると仮定した場合、15 インチ(用紙サイズ横方向)×10CPI で単純計算により 150 文字印刷可能であることがわかります。

しかし、後述の注意事項でも述べているように、連続用紙の場合は左右にトラクタに掛けるための穴が空いており、一般的には左右合わせて約 1.4 インチ(プリンタ機種により異なります)は物理的に印字が不可能な領域があります。 したがって、実際には横 15 インチすべてが印字可能な領域ではなく、印字不可能な領域を間引きした約 13.6 インチが印字可能な領域であり、この場合の印字可能文字数は 136 文字ということになります。

用紙内で印字可能な行数

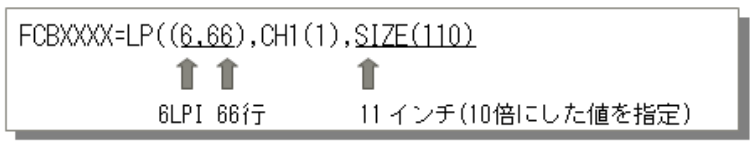
用紙内に印字可能な行数は、FCB 制御文の定義により決定します。

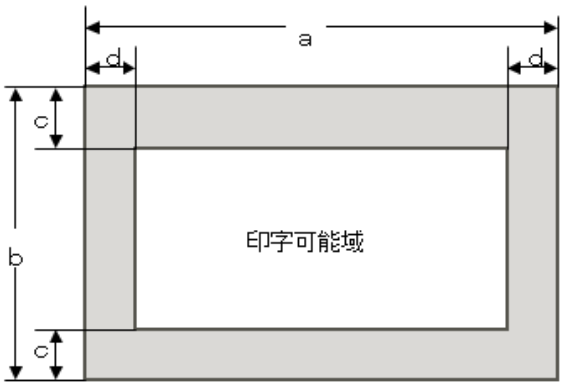
FCB 制御文は、プリンタ装置にセットされている物理的な用紙のサイズ(縦方向)、行間隔(LPI)およびチャネル位置を指定します。用紙内の印字可能行数は、FCB 制御文で定義された用紙サイズと行間隔から決まります。

以下に、15×11 インチの連続用紙を使用した場合の FCB 制御文の定義例を示します。

行間隔	6LPI
用紙内最大印字可能行数	66 行
用紙サイズ(縦方向)	11 インチ(10 倍にした値を指定)

FCB制御文の定義例





調整可能だが、出荷時の状態は以下の設定

a:用紙サイズ(横方向)=15インチ
b:用紙サイズ(縦方向)=11インチ
c:印字不可能域=22.0mm
d:印字不可能域(以下の表を参照)

用紙幅(インチ)	打ち出し位置(mm)
4.0～ 4.7	5.08 ～ 9.0
15.3～16.0	15.0 ～30.0

プリンタのハード仕様に依存する部分で、用紙の上下左右に物理的な印字不可能域が設けられているプリンタがあります。 これらのプリンタに対応したプリンタドライバはこの部分への印字を抑止しています(データが印字可能域までシフトされたり捨てられたりします)。物理的な印字不可能域の大きさはプリンタによりさまざまであり、利用者はプリンタの取扱い説明書などを参照し、印字不可能域を考慮した帳票設計を行う必要があります。 このため、上記の印字可能文字数および行数はあくまでも目安であり、すべてがこの限りではありませんので注意してください。

フォントテーブル

フォントテーブルとは、印刷ファイルを利用して帳票出力を行うときの書体情報を定義するテキスト形式のファイルのことです。

書体情報には、印字する文字のフォントフェース名および印字スタイルを書体番号と対応付けて指定します。

CHARACTER TYPE 句に印字モード名を指定して、その印字モード名を定義した PRINTING MODE 句に書体番号を指定した場合、書体番号の情報を含むフォントテーブルが必要になります。

フォントテーブルの書式を、以下に示します。

書式および設定情報

```
[書体番号]

FontName= フォントフェース名

Style={R|B|I|BI}
```

セクション名(書体番号ごとに指定)

セクション名[書体番号]は、書体情報がどの書体番号に対応付けられた情報かを識別するための情報です。書体番号ごとに書体情報の設定が必要です。

書体番号には、COBOL ソースプログラムに記述した書体番号("FONT-nnn")を記述します。文字列は、英数字の半角文字で記述してください。

例

```
[FONT-001]
```

FontName(文字書体指定)[省略可能]

フォントフェース名には、印字に使用する文字書体を指定します。省略した場合、明朝(通常の書体)で印字されます。

- 印字に使用するフォントフェース名は、32 バイト以内の英数字または日本語文字で指定してください。
- フォントフェース名に指定できるフォント名は、コントロールパネルのフォントを選択し、表示されるフォントの一覧から選択してください。

Style(印字スタイル指定)[省略可能]

印字に使用する文字書体のスタイルを指定します。省略した場合、標準(R)で印字されます。

R	標準
B	太字
I	斜体
BI	太字・斜体

記述例

```
[FONT-001]                    ←MS 明朝 を太字で印字指定
FontName=MS 明朝
Style=B

[FONT-002]                    ←MS 明朝 を斜体で印字指定
FontName=MS 明朝
Style=I

  :
```

印刷ファイルでフォントテーブルを使用する場合は、環境変数情報@CBR_PrintFontTable (印刷ファイルで使用するフォントテーブルの指定)またはファイル識別名にフォントテーブル名を指定します。

ファイル識別名にフォントテーブル名を指定する方法については、以下を参照してください。

- ファイル識別名 (プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定)
- ファイル識別名 (プログラムで使用するプリンタおよび各種パラメタの指定)

行単位のデータを印刷する方法

ここでは、FORMAT 句なし印刷ファイルを使って、行単位のデータを印刷する方法について説明します。なお、FORMAT 句なし印刷ファイルを使った例題プログラムがサンプルとして提供されていますので、NetCOBOL for .NET コマンドラインサンプルアプリケーションを参考にしてください。

このセクションの内容

NCOBU00309

概要

印刷ファイルは、レコード順ファイルと同様に定義し、レコード順ファイルの創成処理と同様の処理を行います。FORMAT 句なし印刷ファイルでは、以下を指示することができます。

- 論理的な 1 ページの大きさ(ファイル記述項の LINAGE 句)
- 文字の大きさ、書体、形態、方向および間隔(データ記述項の CHARACTER TYPE 句)
- 行送りやページ替え(WRITE 文の ADVANCING 指定)

プログラムの記述

ここでは、行単位のデータを使った印刷ファイルのプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     [機能名 IS 呼び名].
 INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT  ファイル名
      ASSIGN TO PRINTER
     [ORGANIZATION IS SEQUENTIAL]
     [FILE STATUS IS   入出力状態].
DATA DIVISION.
 FILE SECTION.
 FD  ファイル名
    [RECORD  レコードの大きさ]
    [LINAGE IS   論理ページ構成の指定].
 01  レコード名  [CHARACTER TYPE IS   {MODE-1 | MODE-2 | MODE-3 | 呼び名} ].
     レコード記述項
 WORKING-STORAGE SECTION.
[01  入出力状態    PIC  X(2).]
[01  データ名    [CHARACTER TYPE IS   {MODE-1 | MODE-2 | MODE-3 | 呼び名} ].]
PROCEDURE DIVISION.
    OPEN OUTPUT  ファイル名.
    WRITE  レコード名 [FROM  データ名]  [AFTER  ADVANCING  ～].
    CLOSE  ファイル名.
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

環境部には、機能名と呼び名の対応付け(プログラム中で印字文字を指示する場合)および印刷ファイルの定義を記述します。

機能名と呼び名の対応付け

印字文字の大きさ、書体、形態、方向および間隔の値を示す機能名を呼び名に対応付けます。 この呼び名は、レコード中のデータ項目および作業用のデータ項目を定義するときに、CHARACTER TYPE 句に指定します。機能名の種類については、COBOL 文法書の「4.2.3.1 機能名-1 句」を参照してください。

印刷ファイルの定義

ファイル管理記述項を記述するために必要な情報を以下の表に示します。

表: ファイル管理記述項に指定する情報

	指定する場所	情報の種類	指定する内容および用途
必須	SELECT 句	ファイル名	COBOL プログラム中で使用するファイル名を指定します。 このファイル名は、COBOL の利用者語の規則に従った名前にします。

	ASSIGN 句	ファイル参照子	PRINTER 、PRINTER-n 、アクセス名、ローカルプリンタポート名(LPTn:) 、シリアルポート名(COMn:) またはプリンタ名を指定します。 PRINTER を指定すると、通常使うプリンタに設定されているプリンタに出力されます。
任意	ORGANIZATION 句	ファイル編成を示す文字列	SEQUENTIAL を指定します。
	FILE STATUS 句	データ名	作業場所節または連絡節で、2 桁の英数字項目として定義したデータ名を指定します。このデータ名には、入出力処理の実行結果が設定されます。(注)

注: 設定される値については、ファイル入出力状態一覧を参照してください。

データ部(DATA DIVISION)

データ部には、レコードの定義およびファイル管理記述項に指定したデータ名の定義を記述します。レコードの定義は、ファイル記述項とレコード記述項で記述します。 ファイル記述項を記述するために必要な情報を以下の表に示します。

表: ファイル記述項に指定する情報

	指定する場所	情報の種類	指定する内容および用途
任意	RECORD 句	レコードの大きさ	印字可能領域の大きさを指定します。
	LINAGE 句	論理ページの構成	論理的な 1 ページを構成する行数、上端と下端の余白の大きさおよび脚書き領域が始まる位置を指定します。この句にデータ名を指定すると、これらの情報をプログラム中で変更することができます。

手続き部(PROCEDURE DIVISION)

以下の順序で入出力文を実行します。

1. OUTPUT 指定の OPEN 文: 印刷処理の開始
2. WRITE 文: データの出力
3. CLOSE 文: 印刷処理の終了

〔OPEN 文および CLOSE 文〕

OPEN 文は印刷処理の開始時に、CLOSE 文は印刷処理の終了時に、それぞれ 1 回だけ実行します。

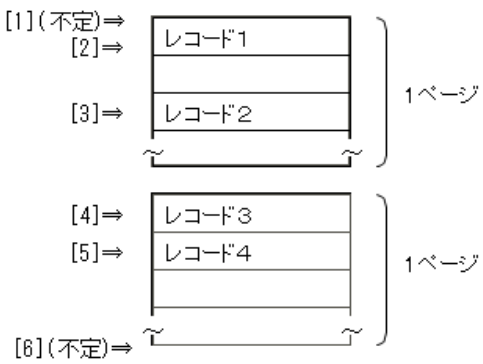
〔WRITE 文〕

1 回の WRITE 文は、1 行のデータを出力します。 WRITE 文の ADVANCING 指定に PAGE を記述すると、ページ替えが行われます。 また、行数を記述すると、指定した行数が行送りされます。ADVANCING 指定には、AFTER 指定と BEFORE 指定があり、データの出力をページ替えまたは行送りの後に行うか、前に行うかを指定します。 ADVANCING 指定を省略した場合、"AFTER ADVANCING 1"を指定したものとみなされます。

AFTER ADVANCING 指定による印刷行の制御を以下に示します。

```
OPEN OUTPUT ファイル1  .                  …[1]
WRITE レコード1   AFTER ADVANCING PAGE. …[2]
WRITE レコード2   AFTER ADVANCING 2.    …[3]
WRITE レコード3   AFTER ADVANCING PAGE. …[4]
WRITE レコード4  .                      …[5]
CLOSE ファイル1  .                      …[6]
```

[1]⇒～[6]⇒:[1]～[6]の入出力文を実行した後の
行位置(プリンタ装置のポインタ)



注意

- FROM 指定を記述した WRITE 文で、"WRITE A FROM B."と記述した場合、CHARACTER TYPE 句は、A ではなく B に定義します。 CHARACTER TYPE 句が両方に定義された場合には、B の指定が有効となります。
- OPEN 文の実行直後の AFTER ADVANCING PAGE 指定の WRITE 文の実行では、ページ替えは行われません。

入出力エラー処理

入出力エラーの検出方法および入出力エラーが発生したときの実行結果については、入出力エラー処理を参照してください。

NCOBU00311

プログラムのビルドおよび実行

プログラムのビルド

とくに必要なコンパイラオプションはありません。

プログラムの実行

はじめに、印刷装置の割当てを行います。 印刷装置の割当ては、ファイル管理記述項の ASSIGN 句の記述内容によって異なります。 ここでは、まず、プログラムおよびアプリケーション構成ファイルでの印刷装置の指定方法について説明し、次に ASSIGN 句の記述内容と出力先の関係を例を用いて説明します。なお、印刷ファイルの出力先を印刷装置以外にした場合、出力内容は保証されません。

印刷装置の指定方法

印刷装置の指定は、以下の 2 種類の方法で設定することができます。

- プリンタ名
- ローカルプリンタポート名("LPTn:")またはシリアルポート名("COMn:")

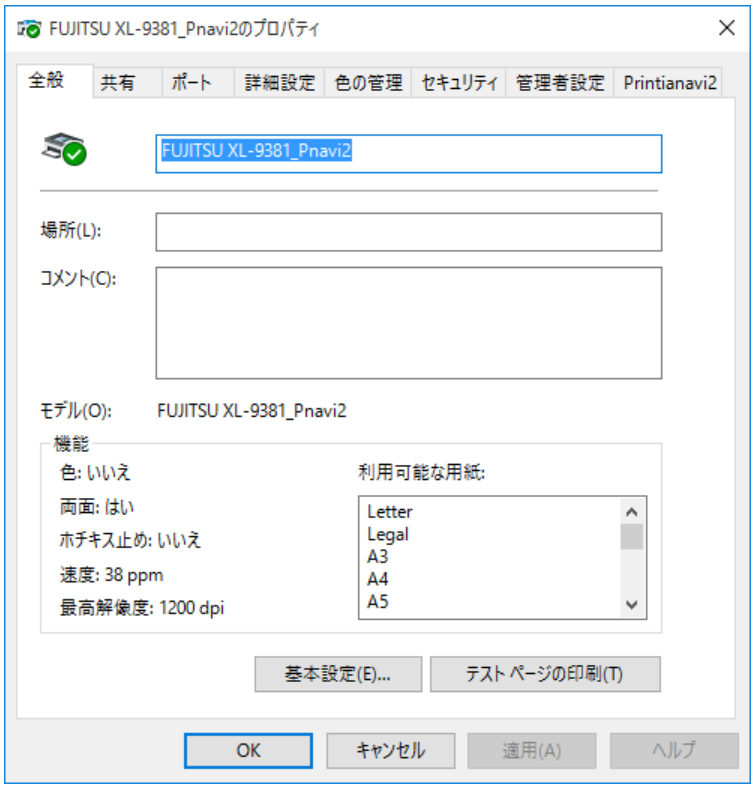
  n は、1～9 の範囲です。

プリンタ名の情報には、各プリンタの[プロパティ]ダイアログの情報を使用します。

各プリンタの[プロパティ]ダイアログは、以下の手順で表示することができます。(Windows 10 の場合)

1. コントロールパネルから"デバイスとプリンター"を起動します。
2. 表示されたプリンターの一覧からプリンタを選択します。
3. メニューから"プリンターのプロパティ"を選択します。

以下に[プロパティ]ダイアログの[全般]シートを示します。



◆プリンタ名

プリンタ名とは、[プロパティ]ダイアログの[全般]シートのプリンタマークの右横のエディットボックスに表示されているプリンタ名の先頭に、"PRTNAME:"を付けた名前のことです。

例：

```
PRTNAME:FUJITSU XL-9381_Pnave2
PRTNAME:Canon LBP-A404E
```

ネットワーク上のプリンタに接続している場合は、プリンタの名前に**¥¥サーバ名¥**の指定が必要です。

例：

```
PRTNAME:¥¥PRTSVR¥FUJITSU XL-9381_Pnave2
PRTNAME:¥¥PRTSVR¥Canon LBP-A404E
```

備考

プリンタ名の取得は、実行環境設定ツールの〔環境設定〕メニューから"プリンタ名"を選択することによって表示される〔プリンタ名の選択〕ダイアログを利用すると便利です。

プリンタのプロパティにネットワークプリンタの名前は表示されませんので、必ず[プリンタ名の選択]ダイアログで指定してください。

◆ローカルプリンタポート名/シリアルポート名

[プロパティ]ダイアログの[ポート]シートの印刷先のポートのリストボックスに表示されている名前を指定します。

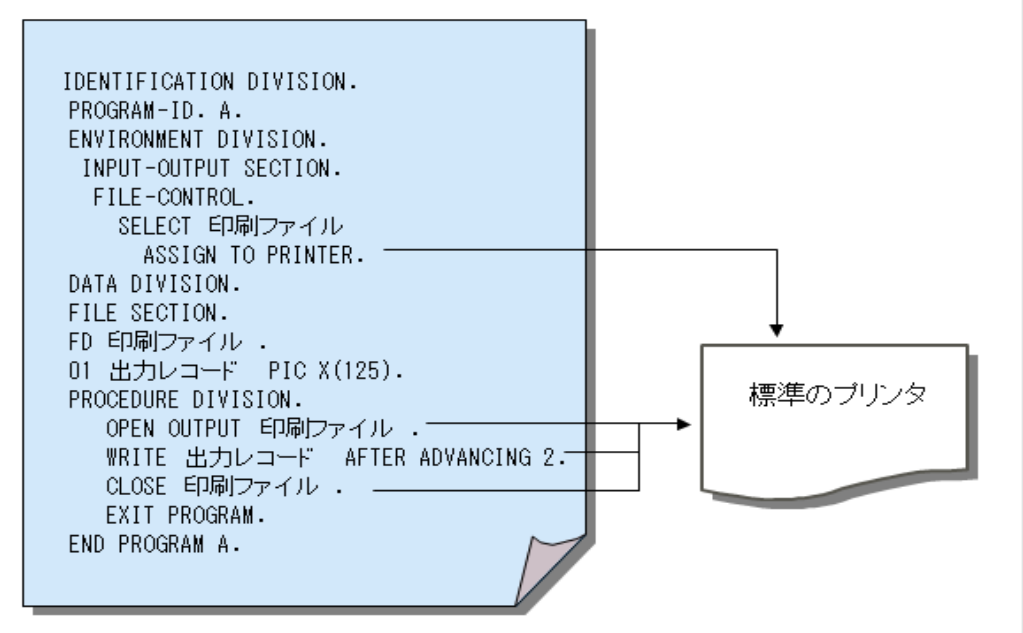
例：

```
"LPT1:"
```

プログラム例

ASSIGN 句に文字列 PRINTER を記述した場合

出力先は、標準のプリンタ（注）になります。



標準のプリンタとは

プリンタの設定で "通常使うプリンタに設定" がチェックされているプリンタのことです。

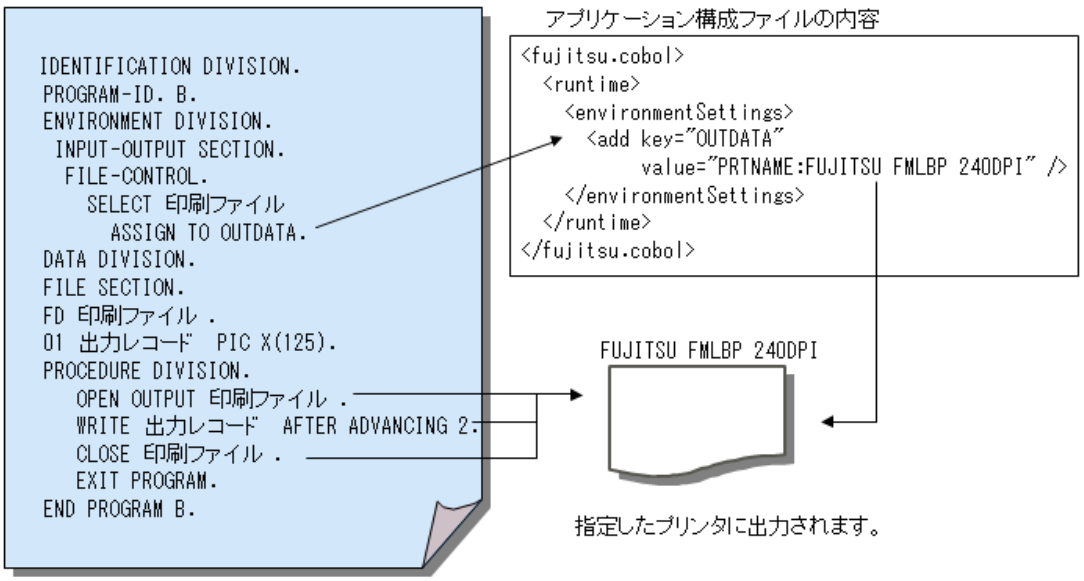
なお、特定のポートに接続されているプリンタ装置に出力する場合は、以降で説明する方法で行ってください。

ASSIGN 句にファイル識別名を記述した場合

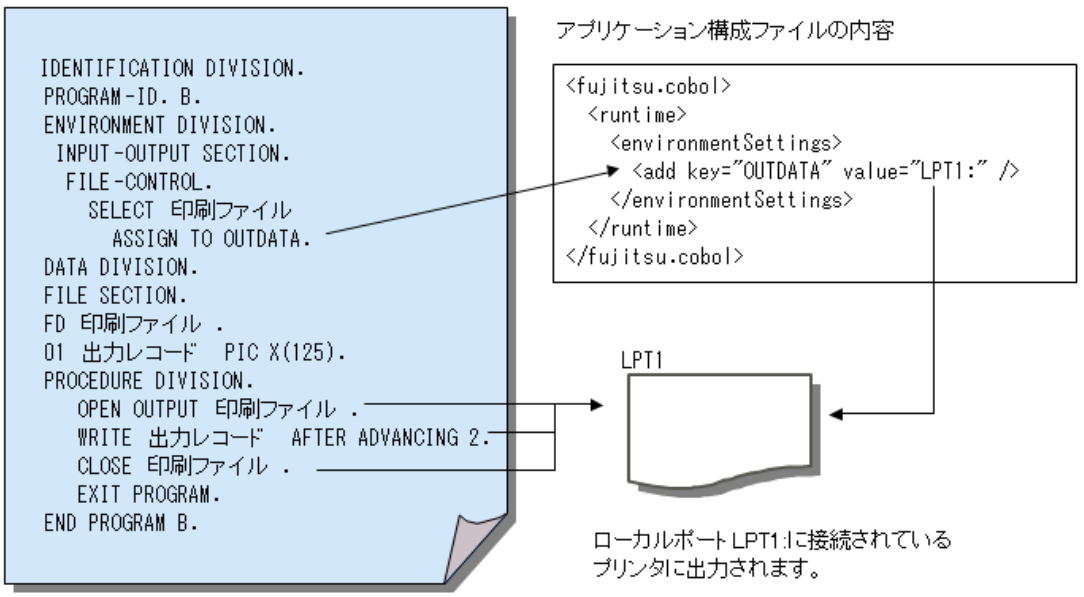
ファイル識別名を環境変数情報名として、出力先のプリンタ名またはローカルプリンタポート名/シリアルポート名を設定します。 環境変数情報の設定方法については、実行環境の設定を参照してください。

なお、ファイル識別名にプリンタが割り当てられていない場合、ファイルの割当てエラーとなります。以下に、実行用の初期化ファイルを使った場合の例を示します。

プリンタ名を指定した場合の例



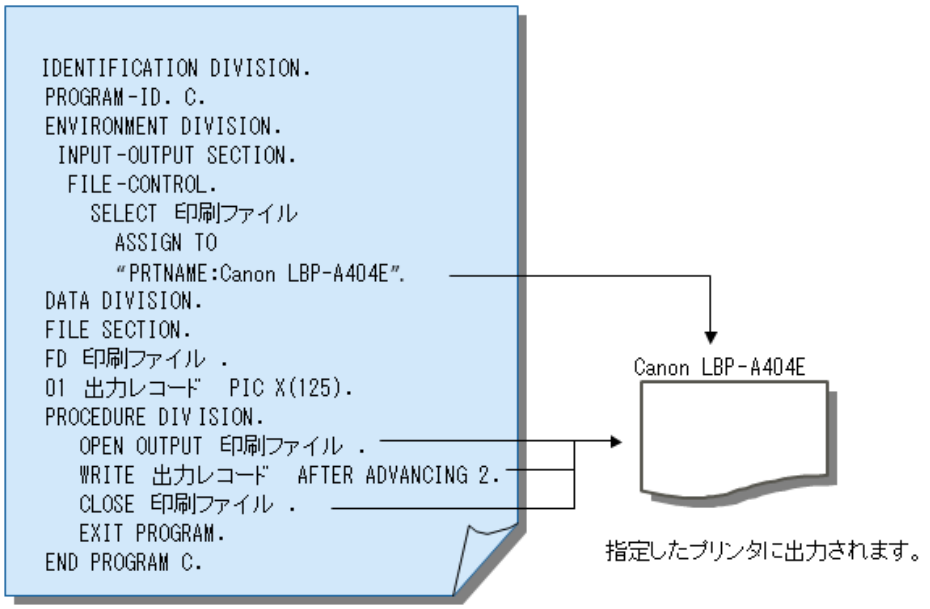
ローカルポート名を指定した場合の例



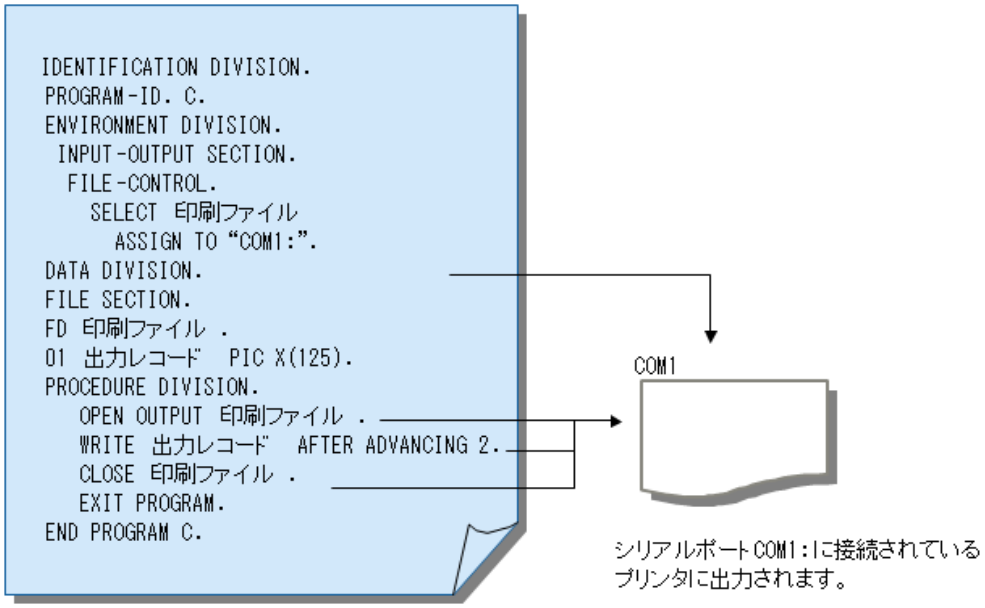
ASSIGN 句にファイル識別名定数を記述した場合

ファイル識別名定数に直接記述されたプリンタまたはファイル識別名定数に記述したローカルポート/シリアルポートに接続されているプリンタが出力先になります。

プリンタ名を指定した場合の例



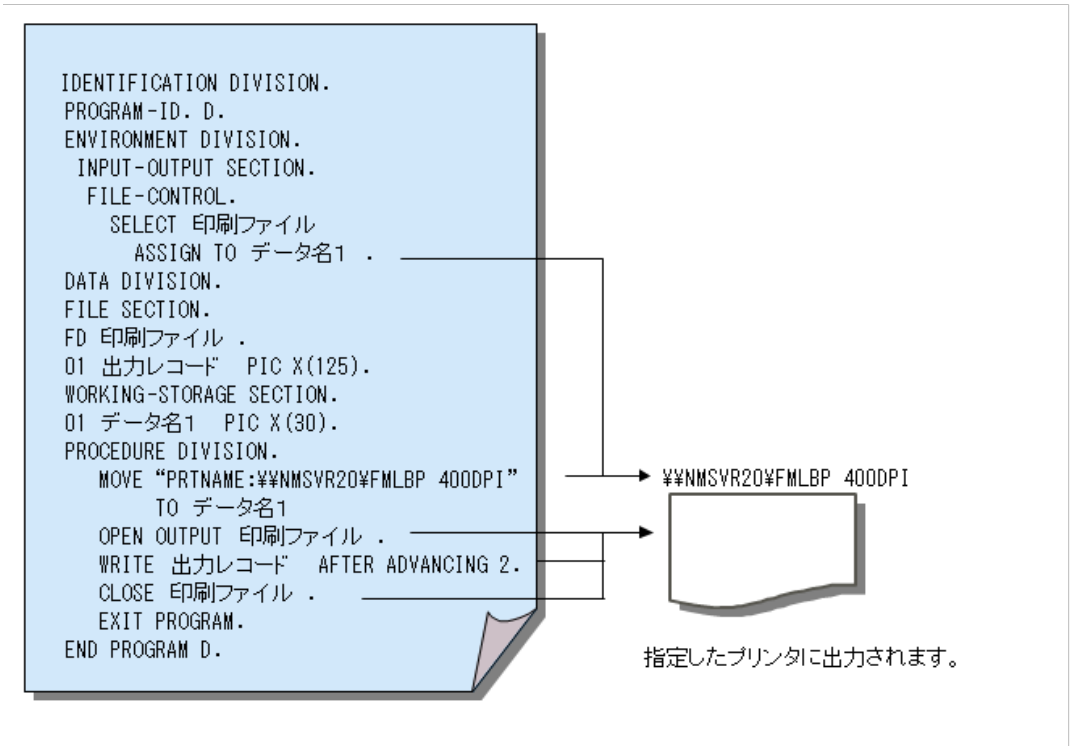
シリアルポート名を指定した場合の例



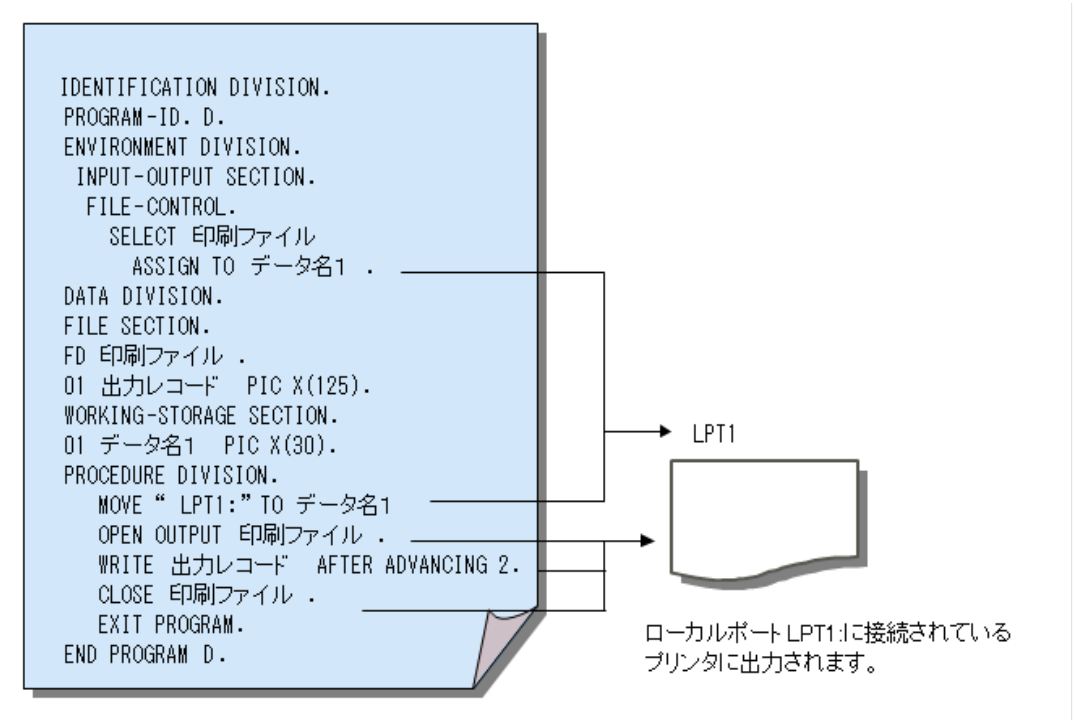
ASSIGN 句にデータ名を記述した場合

データ名に直接記述されたプリンタまたはデータ名に設定されたローカルポート/シリアルポートに接続されているプリンタが出力先になります。 データ名の内容が空白の場合、ファイルの割当てエラーとなります。

プリンタ名を指定した場合の例



ローカルポート名を指定した場合の例

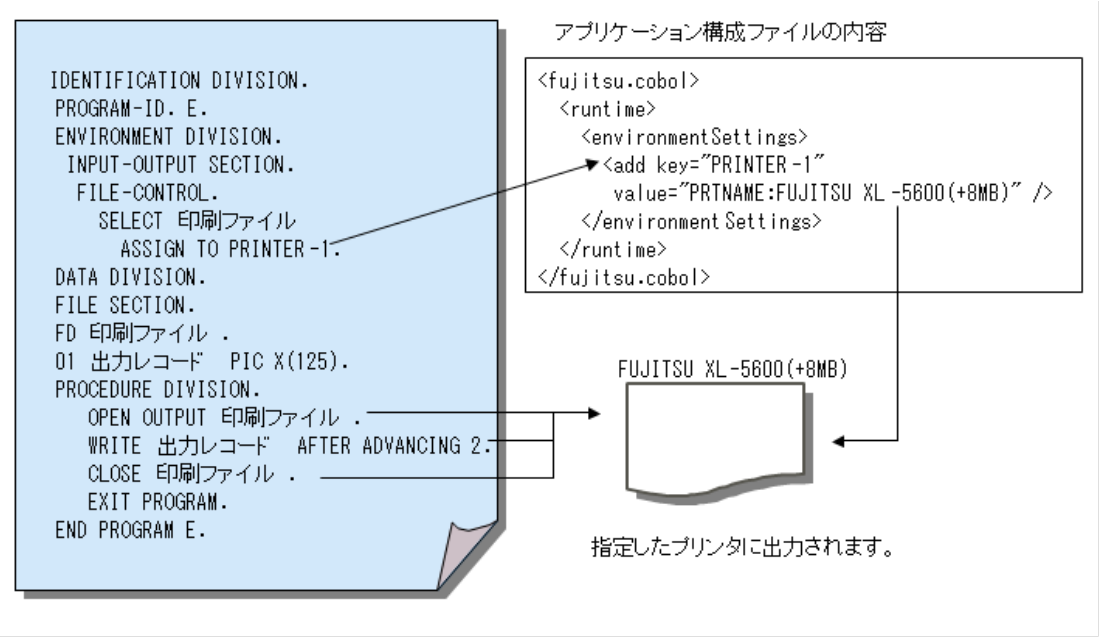


ASSIGN 句に文字列 PRINTER-n を記述した場合

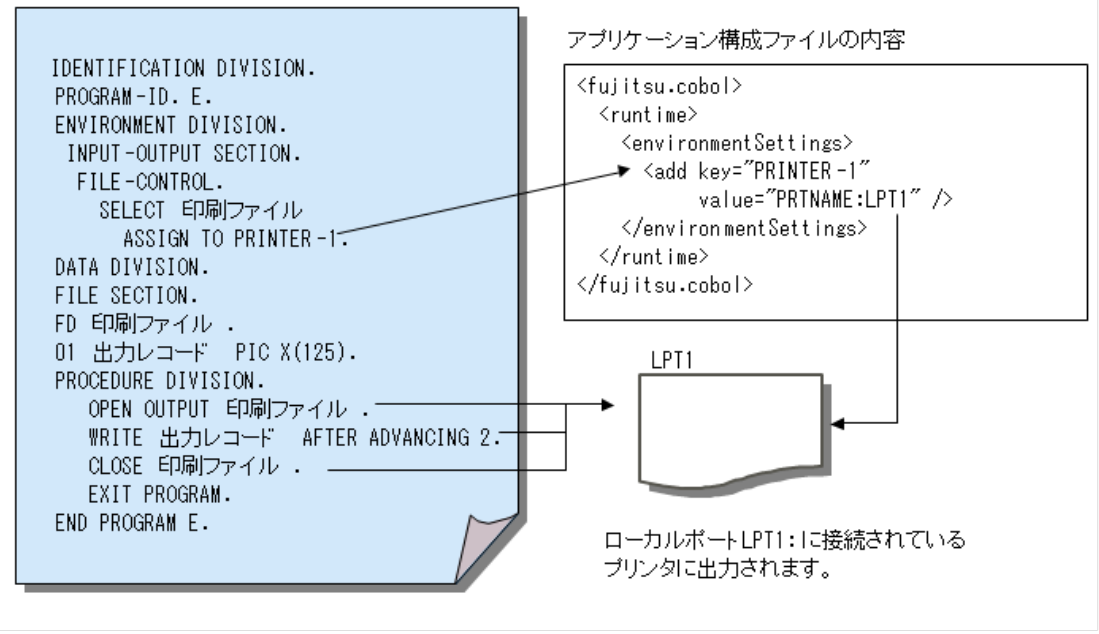
ファイル識別名を記述した場合と同様に、文字列 PRINTER-n(n=1～9)を環境変数情報名として、出力先のプリンタ名またはローカルプリンタポート名/シリアルポート名を設定します。環境変数情報の設定方法については、、実行環境の設定を参照してください。

なお、PRINTER-n にプリンタが割り当てられていない場合、ファイルの割当てエラーとなります。以下に、実行用の初期化ファイルを使った場合の例を示します。

プリンタ名を指定した場合の例



ローカルポート名を指定した場合の例



フォームオーバレイおよびFCBを使う方法

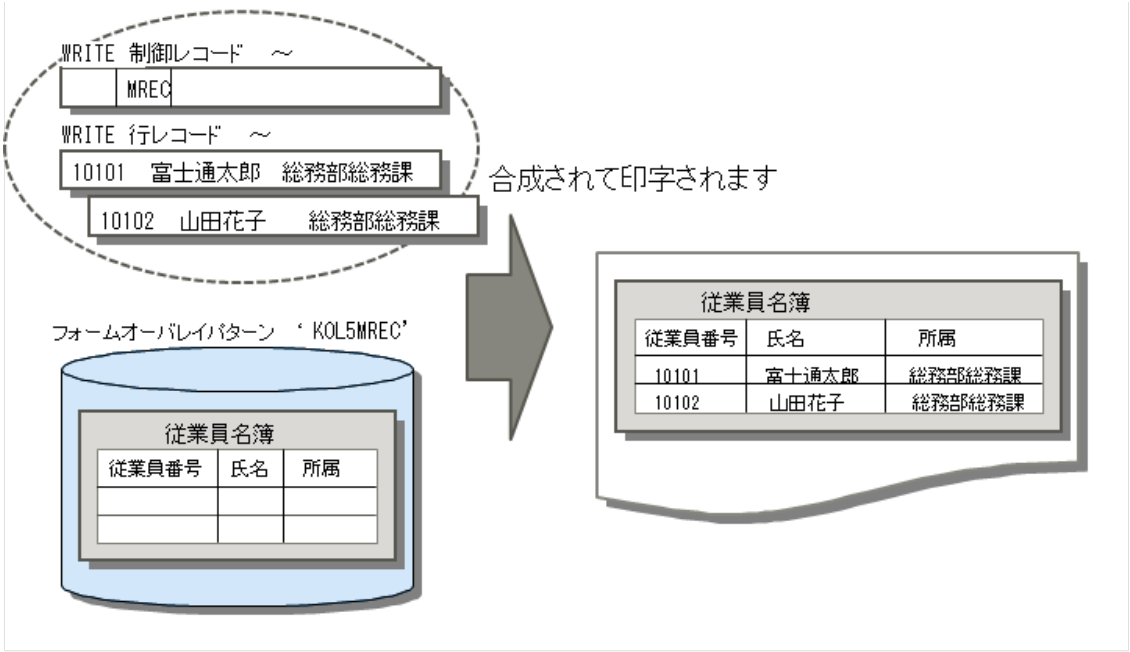
ここでは、FORMAT句なし印刷ファイルでフォームオーバレイパターンやFCBを使う方法について説明します。

このセクションの内容

NCOBU00313

概要

印刷ファイルでフォームオーバレイパターンを使う場合には、制御レコードを使います。制御レコードは、通常のデータと同様に、WRITE 文で出力します。フォームオーバレイパターン名を設定した制御レコードを出力すると、その次のページに書き出したデータと、フォームオーバレイパターンが合成されて印字されます。データを印刷するときの印字する文字の大きさ、書体、形態、方向および間隔は、COBOL プログラムおよびフォームオーバレイパターンで定義することができます。指定できる内容については、印字文字および"FORM ヘルプ"を参照してください。



プログラムの記述

ここでは、フォームオーバレイパターンを使って帳票を印刷するときのプログラムの記述内容について、COBOLの各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    [機能名  IS  呼び名１]
     CTL     IS  呼び名２.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT  ファイル名
      ASSIGN TO PRINTER
     [ORGANIZATION IS SEQUENTIAL]
     [FILE  STATUS IS 入出力状態].
DATA DIVISION.
 FILE SECTION.
 FD  ファイル名
    [RECORD  レコードの大きさ]
    [LINAGE  IS  論理ページ構成の指定].
 01  行レコード名  [CHARACTER  TYPE  IS  {MODE-1｜MODE-2｜MODE-3 ｜呼び名} ].
       レコード記述項
 01  制御レコード名.
       レコード記述項
 WORKING-STORAGE SECTION.
[01  入出力状態    PIC  X(2).]
[01  データ名      [CHARACTER  TYPE  IS  {MODE-1｜MODE-2｜MODE-3 ｜呼び名} ].]
PROCEDURE DIVISION.
    OPEN   OUTPUT  ファイル名.
    WRITE  制御レコード名   AFTER  ADVANCING  呼び名２.
    WRITE  行レコード名     AFTER  ADVANCING  PAGE.
   [WRITE  行レコード名    [FROM データ名] [AFTER ADVANCING  ～].]
    CLOSE  ファイル名.
END PROGRAM  プログラム名.
```

環境部(ENVIRONMENT DIVISION)

環境部には、機能名と呼び名の対応付けおよび印刷ファイルの定義を記述します。

機能名と呼び名の対応付け

制御レコードを指定するための呼び名を、機能名 CTL に対応付けます。この呼び名は、制御レコードを出力するときに WRITE 文に指定します。

CHARACTER TYPE 句を使って印字文字を指示する場合、印字文字の大きさ、形態、方向、書体および間隔の値を示す機能名を呼び名に対応付けます。機能名の種類については、COBOL 文法書の「4.2.3.1 機能名-1 句」を参照してください。

印刷ファイルの定義

印刷ファイルは、ファイル管理記述項で定義します。ファイル管理記述項に記述する内容については、プログラムの記述の"表:ファイル管理記述項に指定する情報"を参照してください。

データ部(DATA DIVISION)

データ部には、レコードの定義および環境部で使用したデータ名の定義を記述します。

レコードの定義

レコードは、ファイル記述項とレコード記述項で定義します。ファイル記述項に記述する内容については、プログラムの記述の"表:ファイル記述項に指定する情報"を参照してください。レコード記述項には、以下のレコードを定義します。

行レコード

プログラム中で編集したデータを印刷するためのレコードを定義します。行レコードは複数個記述することができます。行レコードの 1 つのレコードの内容は、印字可能領域の左端から順に印字されます。行レコードの大きさは、印字可能領域の 1 行の大きさを超えないように指定します。また、印字する文字の大きさを、データ記述項の CHARACTER TYPE 句に指定することができます。CHARACTER TYPE 句に指定できる内容については、印字文字を参照してください。

制御レコード

制御レコードには、I 制御レコードと S 制御レコードがあります。I 制御レコードについては、I 制御レコードを、S 制御レコードについては、S 制御レコードを参照してください。

手続き部(PROCEDURE DIVISION)

以下の順序で入出力文を実行します。

1. OUTPUT 指定の OPEN 文: 印刷処理の開始
2. WRITE 文: データの出力
3. CLOSE 文: 印刷処理の終了

〔OPEN 文および CLOSE 文〕

OPEN 文は印刷処理の開始時に、CLOSE 文は印刷処理の終了時にそれぞれ 1 回だけ実行します。

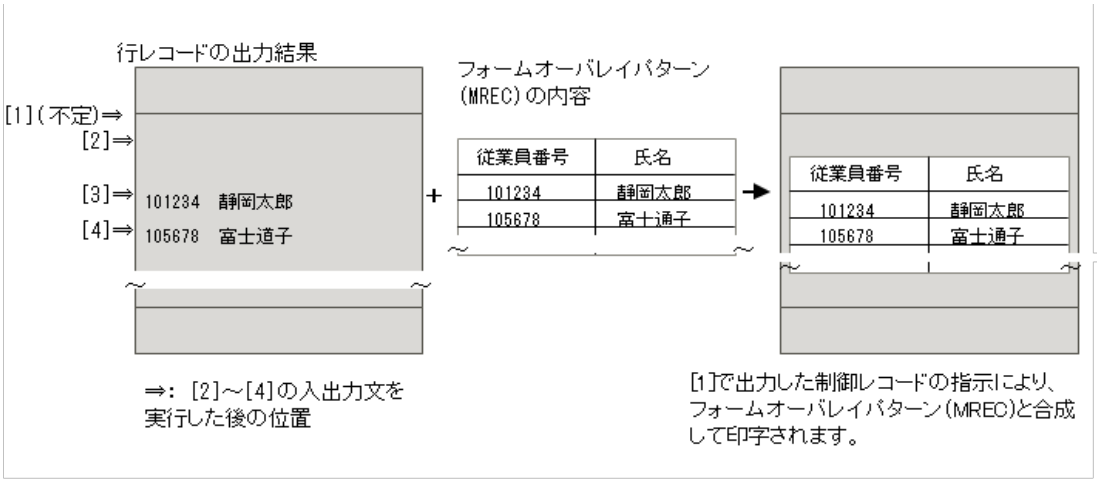
〔WRITE 文〕

WRITE 文は、制御レコードおよび行レコードを出力するときに実行します。行レコードの出力は、行単位のデータを出力するときの WRITE 文の使い方と同じです。詳細については、プログラムの記述を参照してください。

制御レコードを出力するには、ADVANCING 指定に、機能名 CTL に対応付けた呼び名を記述します。

行レコードを使って出力したデータをフォームオーバレイパターンと合成して印字するには、フォームオーバレイパターン名を設定した制御レコードを出力します。また、フォームオーバレイパターンと合成しないで印字するには、フォームオーバレイパターン名に空白を設定した制御レコードを出力します。制御レコードを出力すると、その後に出力するページの形式が制御レコードの内容のとおり設定されます。ただし、制御レコードを出力すると、現在のページには行レコードを出力できなくなるので、制御レコード出力直後の行レコードの出力には、AFTER ADVANCING PAGE を指定する必要があります。

```
      SPECIAL-NAMES.
         CTL     IS  呼び名.
      FILE SECTION.
      FD  ファイル１.
      01  行レコード.
        02 従業員番号 PIC 9(6).
        02 氏名       PIC N(20).
      01  制御レコード.
      *       :
          03  FOVL                 PIC X(4).
      *       :
          MOVE  "MREC"    TO FOVL.
          WRITE  制御レコード AFTER ADVANCING  呼び名.        *>[1]
          MOVE  SPACE TO   行レコード.
          WRITE  行レコード   AFTER ADVANCING  PAGE.          *>[2]
          MOVE  101234    TO   従業員番号.
          MOVE  NC"静岡太郎" TO  氏名.
          WRITE  行レコード   AFTER ADVANCING  2.             *>[3]
          MOVE  105678    TO   従業員番号.
          MOVE  NC"富士通子" TO  氏名.
          WRITE  行レコード   AFTER ADVANCING  1.             *>[4]
```



入出力エラー処理

入出力エラーの検出方法および入出力エラーが発生したときの実行結果については入出力エラー処理を参照してください。

NCOBU00315

プログラムのビルドおよび実行

プログラムのビルド

とくに必要なコンパイラオプションはありません。

プログラムの実行

以下に、フォームオーバレイパターンおよび FCB を使うプログラムの実行方法を示します。

フォームオーバレイパターンを使うプログラム

印刷ファイルでフォームオーバレイパターンを使用する場合、以下の設定を行っておく必要があります。

- 環境変数情報 FOVLDIR (フォームオーバレイパターンのフォルダの指定)または印刷情報ファイルの FOVLDIR キーにフォームオーバレイパターンの格納先フォルダを指定します。環境変数情報 FOVLDIR の設定が省略された場合、フォームオーバレイパターンの焼付けは行われません。

  なお、FORMAT 句付き印刷ファイルの場合、プリンタ情報ファイルに指定します。

- フォームオーバレイパターン格納ファイルの拡張子が"OVD"以外の場合、環境変数情報 OVD_SUFFIX (フォームオーバレイパターンのファイルの拡張子の指定)または印刷情報ファイルの OVD_SUFFIX キーに拡張子の文字列を指定します。なお、拡張子がない場合、文字列"None"を指定します。

  なお、FORMAT 句付き印刷ファイルの場合、プリンタ情報ファイルに指定します。

- フォームオーバレイパターン格納ファイル名の先頭 4 文字が"KOL5"以外の場合、環境変数情報 FOVLTYPE (フォームオーバレイパターンのファイル名の形式の指定)または印刷情報ファイルの FOVLTYPE キーにファイル名の先頭 4 文字を指定します。先頭 4 文字を省略する場合、文字列"None"を指定します

以下に、アプリケーション構成ファイルの記述例を示します。

例: フォームオーバレイパターンを使ったプログラムを実行するときの実行用の初期化ファイルの内容

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="FOVLDIR" value="C:¥FOVLDIR" />       …[1]
      <add key="OVD_SUFFIX" value="" />              …[2]
      <add key="FOVLTYPE" value="FOVL" />            …[3]
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

[1]: フォームオーバレイパターンが格納されているフォルダ(C:¥FOVLDIR)を設定します。

[2]: フォームオーバレイパターン格納ファイルの拡張子は、省略時の"OVD"となります。

[3]: フォームオーバレイパターン格納ファイル名の先頭 4 文字は、"FOVL"となります。

フォームオーバレイパターンの出力

◆フォームオーバレイパターンの出力が可能なプリンタ

Windows システムのグラフィックデバイスインタフェース(GDI)を利用したソフトオーバレイ機能に対応しています。したがって、Windows システムにプリンタドライバが用意されているプリンタの場合、フォームオーバレイパターンの出力が可能です。

FCB を使うプログラム

印刷ファイルで FCB を使用する場合、アプリケーション構成ファイルに FCB 制御文を記述します。

FCB 制御文の指定形式および内容については、FCB を参照してください。なお、I 制御レコードに FCB 名を指定し、アプリケーション構成ファイルに該当する FCB 制御文がない場合、エラーとなります。

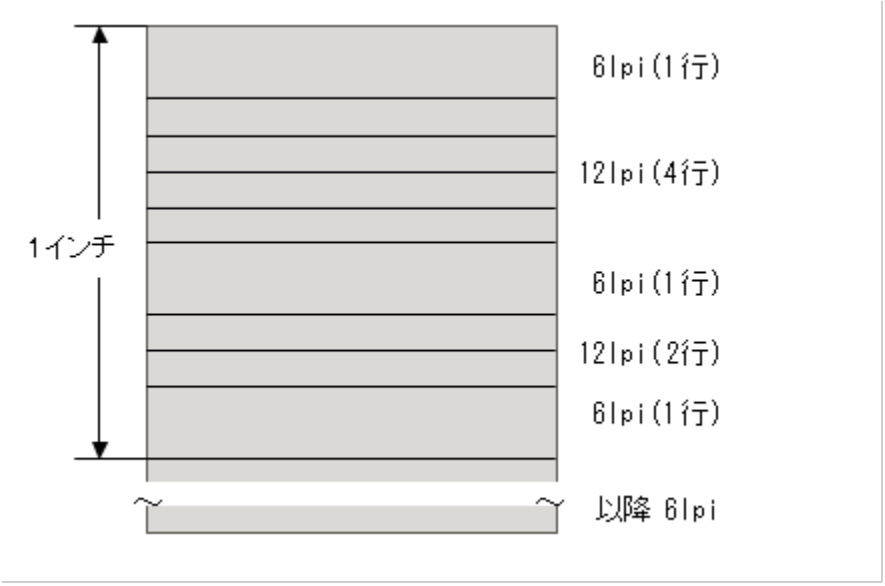
以下に FCB 制御文の例を示します。

例: FCB 名"FCB1"を使ったプログラムを実行するときのアプリケーション構成ファイルの内容

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="FCBFCB1" value="LPI((6,1),(12,4),(6,1),(12,2),(6,1))" />
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

〔出力形式〕

FCB 制御文で設定される 1 ページの形式は、以下のとおりです。



NCOBU00316

帳票定義体を使う印刷ファイルの使い方

ここでは、FORMAT 句付き印刷ファイルでパーティション形式の帳票定義体を使う方法について説明します。なお、FORMAT 句付き印刷ファイルを使って帳票を出力する例題プログラムがサンプルとして提供されていますので、NetCOBOL for .NET コマンドラインサンプルアプリケーションを参考にしてください。

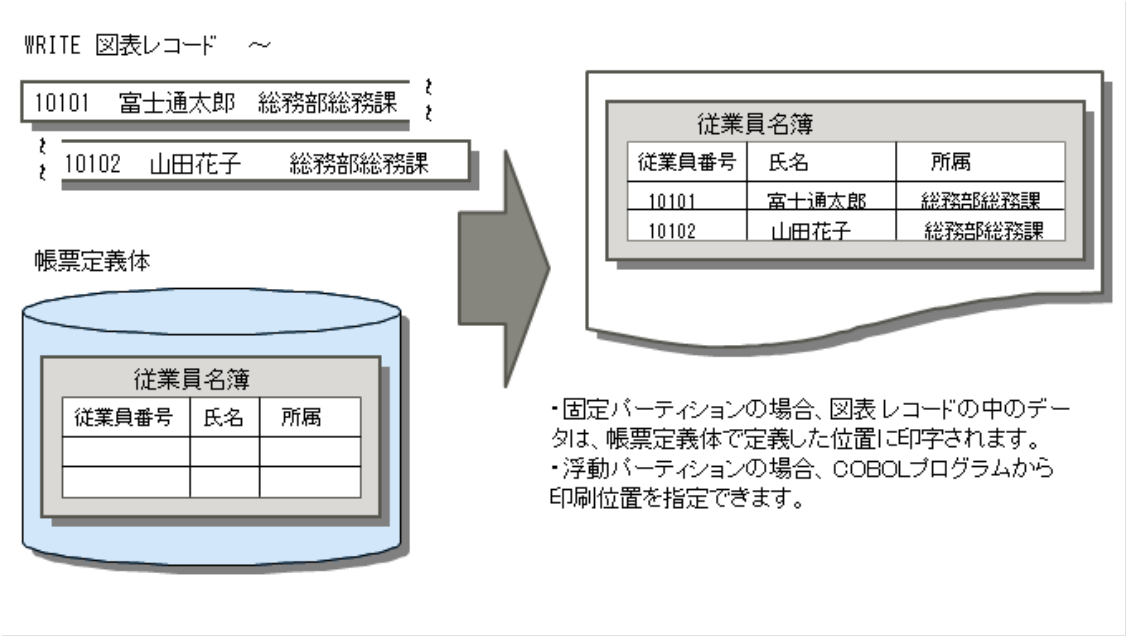
このセクションの内容

概要

パーティション形式の帳票定義体を使った帳票の印刷には、図表レコードを使います。図表レコードには、帳票定義体に定義したパーティション(または項目群)を指定します。図表レコードは、通常のデータを出力するときと同様に、WRITE 文を使って出力します。このとき、1 回の WRITE 文の実行でパーティションに含まれる複数行を印刷することができます。帳票の 1 ページは 1 つ以上のパーティションから構成されます。そのページがどのような構成で印刷されるのかは、図表レコードと行レコードの出力順序から決定されます。

図表レコードの定義項目は、COBOL の COPY 文を使って、帳票定義体から取り込むことができるため、利用者自身が記述する必要はありません。

印字する文字の大きさ、書体、形態、方向および間隔は、行レコードは COBOL プログラムで、図表レコードは帳票定義体の作成時に指示することができます。行レコードのデータに対して指定できる内容については、印字文字、図表レコードのデータに対して指定できる内容については"FORM ヘルプ"を参照してください。



注意

- 自由形式の帳票定義体は、FORMAT 句付き印刷ファイルでは使用できません。表示ファイルの帳票印刷機能を使用してください。
- 帳票定義体を作成する場合、COBOL プログラムで設定/参照される名前は、以下の注意が必要です。
  - 帳票定義体名は 8 文字以内の半角英数字で指定します。
  - 項目群名およびパーティション名は 6 文字以内の半角英数字で指定します。
  - 項目名は COBOL の利用者語の記述規則に従って指定します。

帳票のパーティション

パーティションには、固定パーティション、浮動パーティションと呼ばれる、2 種の属性があります。

固定パーティション

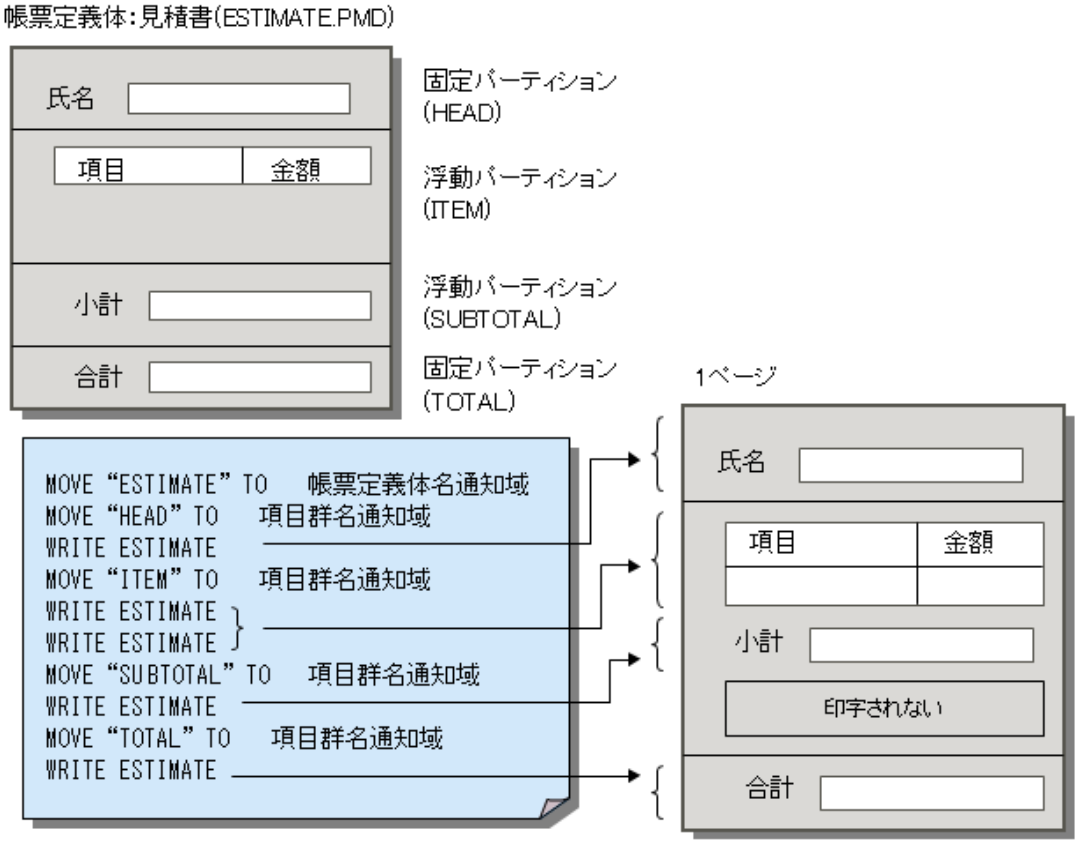
固定パーティションとはページ内での印刷位置が固定のパーティションで、帳票定義体で指定された位置に印刷されます。

浮動パーティション

浮動パーティションとはページ内での印刷位置が出力順序により決められるパーティションで、WRITE 文を実行したときの印刷位置にしたがって印刷されます。

◆出力例

図表レコードに対する WRITE 文と、それに対して出力される固定パーティションおよび浮動パーティションの例を以下に示します。



浮動パーティションは、WRITE 文の実行順にしたがって(ADVANCING 指定が記述されていればその行数分だけ行送りされた後に)印刷されます。点線内は、パーティション TOTAL が帳票定義体で定義された固定位置に印刷されるため、印字されません。

パーティションと行レコードの組合せ

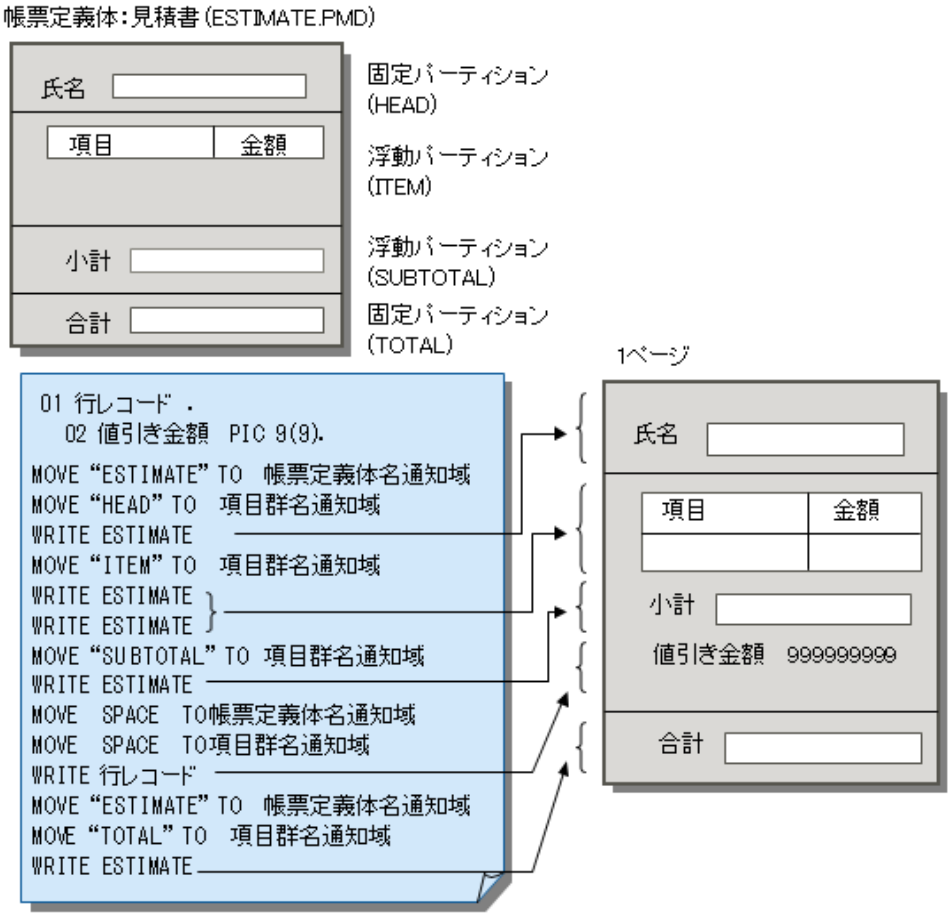
図表レコードを使うことで、帳票定義体に定義したパーティションを印刷することができます。また、そのページの任意の位置に、通常の行レコードを出力することもできます。

パーティションと行レコードは WRITE 文の実行順にしたがって(ADVANCING 指定が記述されていればその行数分だけ行送りされた後に)印刷されます。

◆出力例

図表レコードと行レコードを混在した場合の WRITE 文と、それに対して出力される固定パーティシ

ョン、浮動パーティションおよび行データの例を以下に示します。



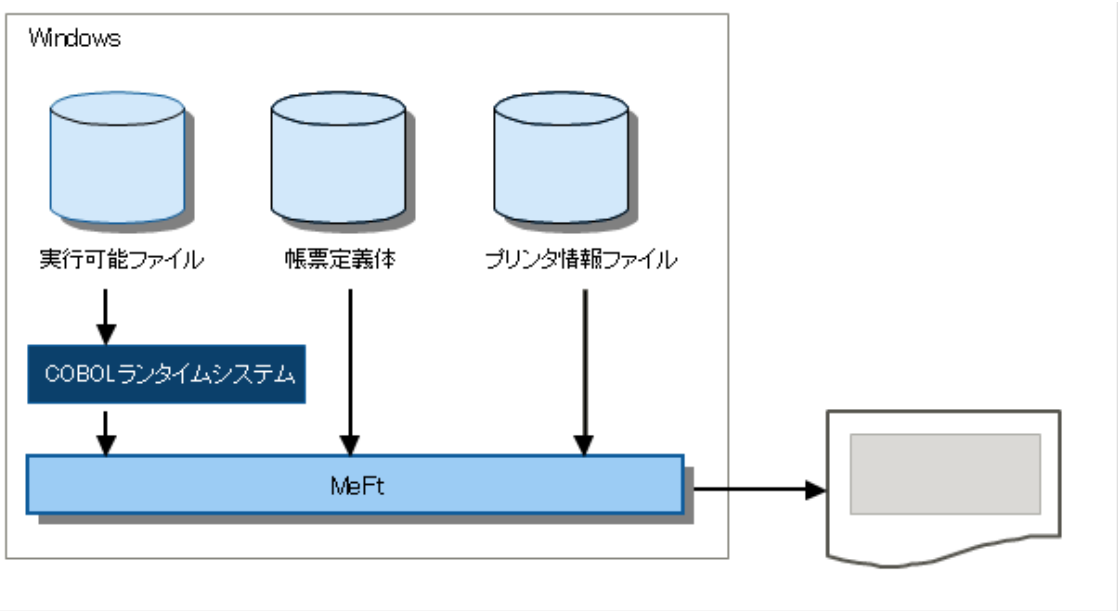
行レコードまたは浮動パーティションを固定パーティションの印刷位置に出力した場合には、その位置に印刷する固定パーティションは、そのページ中では印刷できません。そのページは改ページされて、固定パーティションは次のページの印刷位置に印刷されます。

帳票の電子化

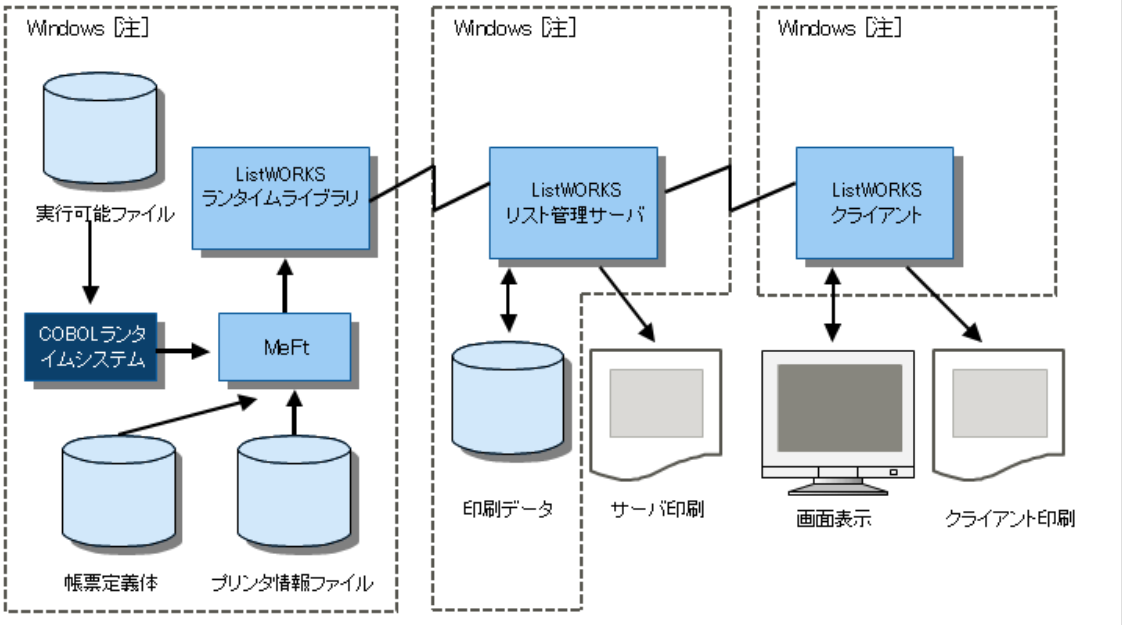
プリンタ情報ファイルに指定を追加することにより、MeFt 経由で出力する帳票を電子化することができます。帳票を電子化する方法には、 電子帳票保存と PDF ファイル出力があります。 それぞれの詳細については、"MeFt ユーザーズガイド"および各連携製品のマニュアルを参照してください。

以下にそれぞれの関連図を示します。

【プリンタ出力の場合】

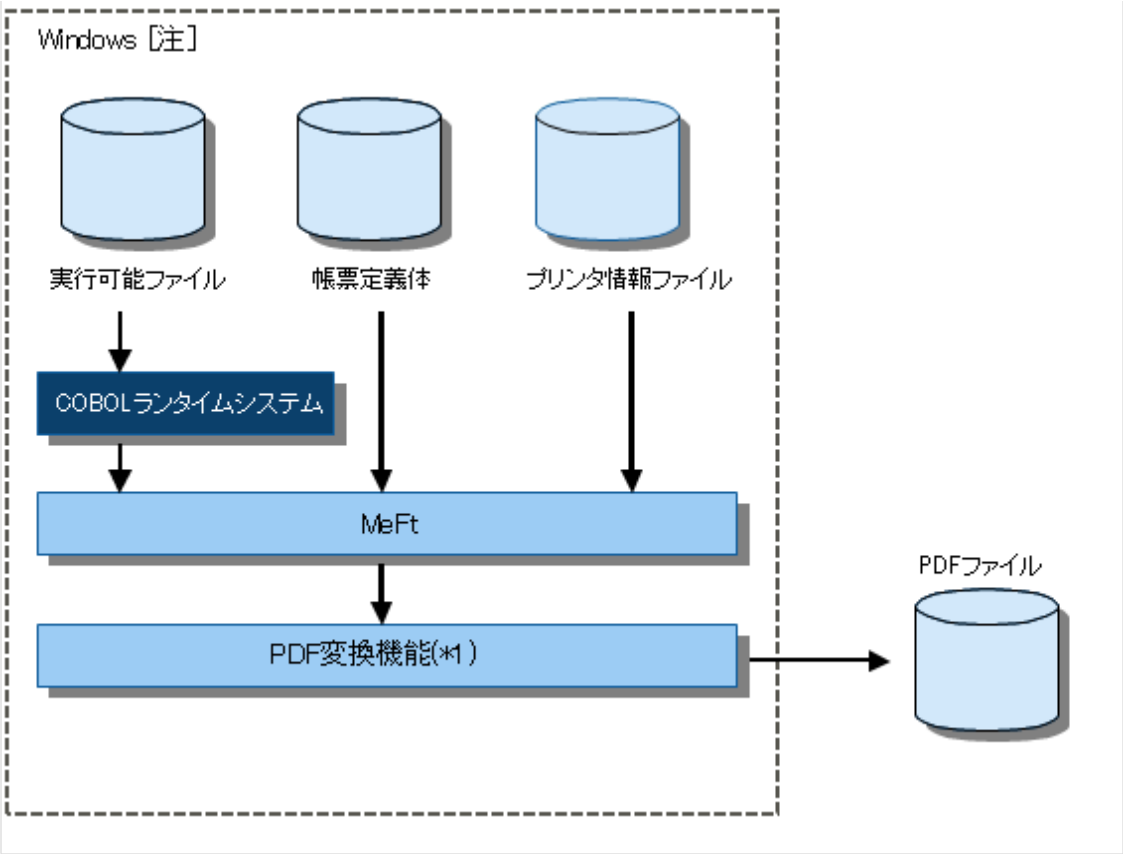


【 電子帳票保存 の場合】



◎MeFt の出力帳票を ListWORKS で扱うことのできる電子帳票の形式にします。

【 PDF ファイル出力の場合】



*1 : Interstage List Creator Enterprise Edition の PDF 変換機能

◎Interstage List Creator Enterprise Edition の PDF 変換機能を使用して、MeFt の出力帳票を PDF(Portable Document Format)ファイルにします。

注: ListWORKS および Interstage List Creator が動作可能なオペレーティングシステムについては、各製品のマニュアルおよびソフトウェア説明書を確認してください。

NCOBU00318

プログラムの記述

ここでは、FORMAT 句付き印刷ファイルで帳票定義体を使うプログラムの記述方法について説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    [ 機能名  IS  呼び名.]
 INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT  ファイル名
    ASSIGN  TO   ファイル参照子
   [ORGANIZATION   IS  SEQUENTIAL]
    FORMAT  IS   帳票定義体名通知域
    GROUP   IS   項目群名通知域
   [FILE STATUS IS   入出力状態１  入出力状態２].
DATA DIVISION.
 FILE SECTION.
 FD  ファイル名
    [RECORD  レコードの大きさ]
    [CONTROL RECORD IS   制御レコード名].
 01  行レコード名  [CHARACTER  TYPE  IS  {MODE-1｜MODE-2｜MODE-3｜呼び名} ].
       レコード記述項
 01  制御レコード名.
       レコード記述項
     COPY  帳票定義体名  OF  XMDLIB.
(01  図表レコード名.    ) （注）
(      レコード記述項   )
 WORKING-STORAGE SECTION.
 01  帳票定義体名通知域  PIC  X(8).
 01  項目群名通知域      PIC  X(8).
[01  入出力状態１        PIC  X(2).]
[01  入出力状態２        PIC  X(4).]
[01  データ名      [CHARACTER  TYPE  IS  {MODE-1｜MODE-2｜MODE-3｜呼び名} ].]
PROCEDURE DIVISION.
    OPEN OUTPUT  ファイル名.
    MOVE   帳票定義体名  TO  帳票定義体名通知域.
    MOVE   項目群名      TO  項目群名通知域.
    WRITE  図表レコード名  [AFTER  ADVANCING ～].
    WRITE  制御レコード名.
    MOVE   SPACE         TO  帳票定義体名通知域.
    MOVE   SPACE         TO  項目群名通知域.
    WRITE  行レコード名  [FROM データ名]  AFTER ADVANCING PAGE.
    CLOSE  ファイル名.
END PROGRAM  プログラム名.
```

注: ()内は、COPY 文の展開を表します。

環境部(ENVIRONMENT DIVISION)

環境部には、機能名と呼び名の対応付け(プログラム中で印字文字を指示する場合)および印刷ファイルの定義を記述します。

機能名と呼び名の対応付け

CHARACTER TYPE 句を使って印字文字を指示する場合、印字文字の大きさ、形態、書体、方向および間隔の値を示す機能名を呼び名に対応付けます。機能名の種類については、COBOL 文法書の「4.2.3.1 機能名-1 句」を参照してください。

FORMAT 句付き印刷ファイルの定義

印刷ファイルは、ファイル管理記述項で定義します。ファイル管理記述項を記述するために必要な情報を以下の表に示します。

表: ファイル管理記述項に指定する情報

	指定する場所	情報の種類	指定する内容および用途
必須	SELECT 句	ファイル名	COBOL プログラム中で使用するファイル名を指定します。このファイル名は、COBOL の利用者語の規則に従った名前にします。
	ASSIGN 句	ファイル参照子	ファイル識別名、ファイル識別名定数またはデータ名のどれかを記述します。ファイル参照子は、実行時に MeFt の使用するプリンタ情報ファイルを割り当てるために使用します。
	FORMAT 句	データ名	作業場所節または連絡節で、８桁の英数字項目として定義したデータ名を指定します。このデータ名は、帳票定義体名を設定するために使用します。
	GROUP 句	データ名	作業場所節または連絡節で、８桁の英数字項目として定義したデータ名を指定します。このデータ名は、帳票定義体で定義した項目群名を設定するために使用します。
任意	FILE STATUS 句	データ名	作業場所節または連絡節で、２桁の英数字項目として定義したデータ名を指定します。このデータ名には、入出力処理の実行結果が設定されます。(注)詳細情報は、４桁の英数字項目を指定します。

注: 設定される値については、ファイル入出力状態一覧を参照してください。

ファイル参照子に、ファイル識別名、ファイル識別名定数またはデータ名のどれを指定したかによって、実行時にプリンタ情報ファイルを割り当てる方法が異なります。ファイル参照子に何を指定するかは、プリンタ情報ファイルの名前がいつ決まるかで決定されます。COBOL プログラム作成時にプリンタ情報ファイルの名前が決定し、その後変更されない場合には、ファイル識別名定数を指定します。COBOL プログラム作成時に名前が決定しなかったり、毎回のプログラム実行時に名前を決定したい場合には、ファイル識別名を指定します。また、プログラムの中で名前を決定したい場合には、データ名を指定します。

データ部(DATA DIVISION)

データ部には、レコードの定義および環境部で指定したデータ名の定義を記述します。

レコードの定義

レコードは、ファイル記述項とレコード記述項で定義します。ファイル記述項を記述するために必要な情報を以下の表に示します。

表: ファイル記述項に指定する情報

	指定する場所	情報の種類	指定する内容および用途
任意	RECORD 句	レコードの大きさ	印字可能領域の大きさを指定します。
	CONTROL RECORD 句	制御レコード名	制御レコード名を指定します。

レコード記述項には、行レコード、制御レコードおよび図表レコードを定義することができます。行レコードと制御レコードの定義方法および使い方については、フォームオーバレイパターンを使うときと同様のため、フォームオーバレイおよび FCB を使う方法のプログラムの記述を参照してください。

図表レコードに定義するレコード記述文は、翻訳時に IN/OF XMDLIB を指定した COPY 文によって、帳票定義体から取り込むことができます。COPY 文で展開される内容については、表示ファイル(帳票印刷)の使い方のプログラムの記述を参照してください。

手続き部(PROCEDURE DIVISION)

以下の順序で入出力文を実行します。

1. OUTPUT 指定の OPEN 文: 印刷処理の開始
2. WRITE 文: データの出力
3. CLOSE 文: 印刷処理の終了

〔OPEN 文および CLOSE 文〕

OPEN 文は印刷処理の開始時に、CLOSE 文は印刷処理の終了時にそれぞれ 1 回だけ実行します。

〔WRITE 文〕

WRITE 文では、行レコード、制御レコードおよび図表レコードを出力することができます。これらのレコードの出力内容により、1 ページは固定形式ページまたは不定形式ページのどちらかとなります。固定形式ページとは、帳票定義体によってその構成が定義されているページであり、帳票定義体に定義された図表レコードまたは行レコードを印字できます。不定形式ページとは、帳票定義体によって定義されないページ、すなわち、FORMAT 句なし印刷ファイルの印字ページと同じ意味のページであり、行レコードだけを印字できます。

固定形式ページに図表レコードと行レコードを混在させる場合は、ファイル記述項の CONTROL RECORD 句に指定した制御レコードに、その図表レコードを定義した帳票定義体名を設定して出力しておかなければなりません。

OPEN 文の実行の直後および帳票定義体名として空白を設定した制御レコードを出力した場合、そのページは不定形式ページとなります。ファイル管理記述項の FORMAT 句に指定したデータ名に帳票定義体名を設定した図表レコード、または帳票定義体名を設定した制御レコードを出力すると、固定形式ページとなります。

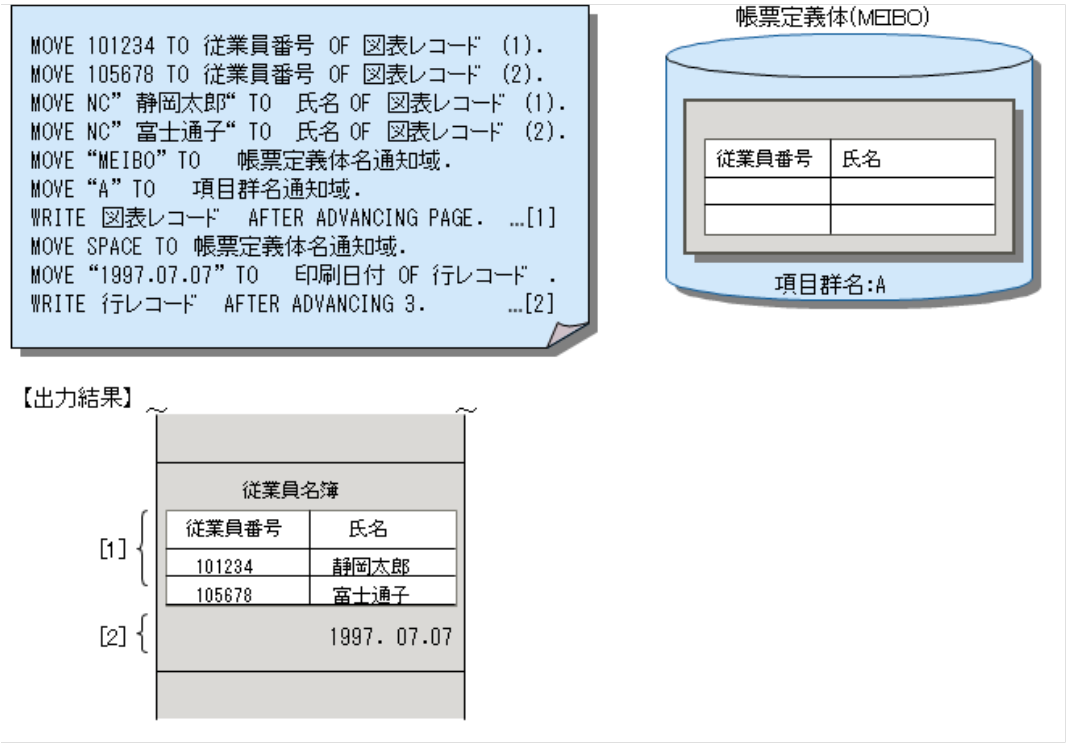
これらのページの形式は、ページの形式を変更する上記の WRITE 文を実行しないかぎり、次のページへも引き継がれます。

図表レコードの出力時に、そのページの形式を決定した帳票定義体名から別の帳票定義体名に変更して出力する場合は、改ページされます。

なお、制御レコードの出力の直後の WRITE 文には、AFTER ADVANCING PAGE を指定します。

WRITE 文の ADVANCING 指定については、行単位のデータを印刷する方法のプログラムの記述を

参照してください。



注意

改ページ指定(AFTER ADVANCING PAGE 指定)がない WRITE 文の実行では、印字開始位置は有効になりません。改ページ指定のない WRITE 文では、印刷装置の印字可能な先頭行から印字されます。

入出力エラー処理

入出力エラーの検出方法および入出力エラーが発生したときの実行結果については、入出力エラー処理を参照してください。

NCOBU00319

プログラムのビルドおよび実行

プログラムのビルド

コンパイラオプション/formpath で、帳票定義体を格納したファイルの格納先を指定します。

複数の帳票定義体を使用し、その拡張子がそれぞれ異なっている場合、コンパイラオプション/formext で、拡張子を指定します。

プログラムの実行

印刷ファイルで帳票定義体を使うプログラムを実行するときには、MeFt の使用するプリンタ情報ファイルが必要です。プリンタ情報ファイルの作成については、プリンタ情報ファイルの作成を参照してください。

印刷ファイルで帳票定義体を使うプログラムを実行するときには、以下の環境設定が必要です。

- 環境変数 PATH に MeFt の格納されているフォルダを追加しておく必要があります。 (注 1)
- MeFt の使用するプリンタ情報ファイルを作成し、ASSIGN 句の記述内容に従い、ファイルの割当てを行います。ファイルの割当て方法は、通常のファイルを割り当てるときと同様のため、ファイルの割当てを参照してください。なお、プリンタ情報ファイルの内容および作成方法については、"MeFt ユーザーズガイド"を参照してください。

  備考

  プリンタ情報ファイルを相対パス名で指定した場合、次の順序で検索されます。

  - 環境変数 MEFTDIR に設定したフォルダ。 (注 1)
  - カレントフォルダ。

  注 1

  MeFt に指定する環境変数は、アプリケーション構成ファイルに指定しても有効になりません。 以下の方法で設定する必要があります。

  - SET コマンドでの設定
  - OS のユーザ環境変数に設定

印刷ファイルで帳票定義体を使う場合の実行例を以下に示します。

例:

[COBOL プログラムの ASSIGN 句の記述内容]

```
ASSIGN TO PRTFILE
```

[アプリケーション構成ファイルの内容]

```
<fujitsu.cobol>
  <runtime>
    <environmentSettings>
      <add key="PRTFILE" value="C:¥DIR1¥MEFPRC" />
    </environmentSettings>
  </runtime>
</fujitsu.cobol>
```

COBOL プログラムの ASSIGN 句に指定したファイル識別名に、プリンタ情報ファイルを割当てます。

FORMAT 句付き印刷ファイルでフォームオーバレイパターンを使用する場合、フォームオーバレイパターンが格納されているフォルダのパス名およびファイル名の拡張子は、プリンタ情報ファイルに設定します。環境変数情報 FOVLDIR (フォームオーバレイパターンのフォルダの指定)または環境変数情報 OVD_SUFFIX (フォームオーバレイパターンのファイルの拡張子の指定)に設定しても有効となりません。

NCOBU00320

表示ファイル(帳票印刷)の使い方

ここでは、表示ファイルを使って、帳票を印刷する方法について説明します。

このセクションの内容

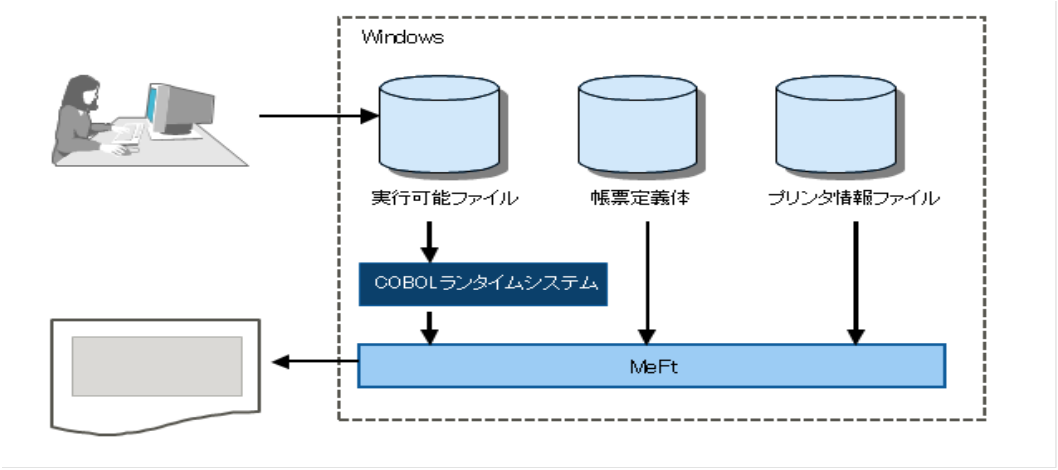
概要

表示ファイル機能では、FORM で定義した帳票形式(帳票定義体)を使って帳票印刷を行います。帳票定義体は、FORM を使って画面イメージで簡単に作成することができます。帳票定義体に定義したデータ項目は、COBOL の COPY 文を使って、翻訳時に COBOL プログラムに取り込むことができます。そのため、帳票印刷のためのデータ項目の定義を、利用者自身が COBOL プログラムに記述する必要はありません。

表示ファイル機能を実際に使用するときには、帳票定義体および MeFt が必要です。

これらの関連図を以下に示します。

【ローカル環境で使用する場合】



MeFt が動作可能なオペレーティングシステムについては、"MeFt ユーザーズガイド"および MeFt のソフトウェア説明書を確認してください。

なお、表示ファイル(帳票印刷)では、FORMAT 句付き印刷ファイルと同様に帳票の電子化機能を使用することができます。詳細については、帳票定義体を使う印刷ファイルの使い方の概要の"帳票の電子化"を参照してください。

NCOBU00322

作業手順

表示ファイル機能を使って帳票印刷を行うには、帳票定義体、COBOL ソースプログラムおよびプリンタ情報ファイルが必要です。帳票定義体および COBOL ソースプログラムは翻訳時までに、プリンタ情報ファイルは実行時までに作成します。以下に表示ファイル機能を使って帳票印刷を行うときの標準的な作業手順を示します。

1. FORM を使って帳票定義体を作成します。
2. テキストエディターを使って COBOL ソースプログラムを作成します。
3. COBOL ソースプログラムを翻訳し、実行可能プログラムを作成します。
4. テキストエディターを使ってプリンタ情報ファイルを作成します。
5. 実行可能プログラムを実行します。

帳票定義体の作成

ここでは、表示ファイル機能で使用するための帳票定義体を作成するときに設定する情報および注意事項について記述します。FORM の詳しい機能や使用方法については、"FORM ヘルプ"を参照してください。

帳票定義体を作成するときに設定する情報を、以下の表に示します。

表: 帳票定義体に設定する情報

情報の種類		指定する内容および用途
必須	ファイル名	帳票定義体を格納するファイルの名前を指定します。
	定義サイズ	帳票の大きさを行数と桁数で指定します。
	定義体形式	形式を指定します。
	データ項目	印刷するデータを設定するためのデータ項目を指定します。ここで指定した項目名は、COBOL プログラムを記述するときにデータ名として使用されます。
	項目群	1 回の印刷処理で印字する 1 つ以上の項目を 1 つの項目群としてまとめます。ここで指定した項目群名は、COBOL プログラムを記述するときに使用します。
任意	項目制御部(注)	COBOL プログラム中で帳票定義体の定義内容を特殊レジスタを使って変更したい場合、5 バイトの項目制御部を指定します。

注: 項目制御部は、帳票定義体に定義したデータ項目に付加される情報で、入力処理と出力処理で"共用する(3 バイト)"と"共用しない(5 バイト)"または"なし"の 3 種類があります。COBOL プログラムで特殊レジスタを使用する場合、"共用しない(5 バイト)"を指定する必要があります。 また、項目制御部の種類の違う帳票定義体を同じプログラムで混在して使用することはできません。

帳票定義体を作成する場合、COBOL プログラムで設定/参照される名前は、以下の注意が必要です。

- 帳票定義体名は 8 文字以内の半角英数字で指定します。
- 項目群名およびパーティション名は 6 文字以内の半角英数字で指定します。
- 項目名は COBOL の利用者語の記述規則に従って指定します。

NCOBU00324

プログラムの記述

ここでは、表示ファイル機能を使って帳票を印刷するときのプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT  ファイル名
      ASSIGN TO  GS-ファイル識別名
      SYMBOLIC DESTINATION IS "PRT"
      FORMAT IS  帳票定義体名通知域
      GROUP  IS  項目群名通知域
     [PROCESSING MODE IS   処理種別通知域]
     [UNIT  CONTROL   IS   特殊制御情報通知域]
     [FILE  STATUS  IS   入出力状態１  入出力状態２].
DATA DIVISION.
 FILE SECTION.
 FD  ファイル名.
     COPY  帳票定義体名 OF XMDLIB.
(01  表示レコード名.  ) （注）
(      レコード記述項 )
 WORKING-STORAGE SECTION.
 01  帳票定義体名通知域    PIC X(8).
 01  項目群名通知域        PIC X(8).
[01  処理種別通知域        PIC X(2).]
[01  特殊制御情報通知域    PIC X(6).]
[01  入出力状態１          PIC X(2).]
[01  入出力状態２          PIC X(4).]
PROCEDURE DIVISION.
    OPEN OUTPUT  ファイル名.
   [MOVE   出力の指定    TO  EDIT-MODE   OF  データ名.]
   [MOVE   強調の指定    TO  EDIT-OPTION OF  データ名.]
   [MOVE   色            TO  EDIT-COLOR  OF  データ名.]
    MOVE   帳票定義体名  TO  帳票定義体名通知域.
    MOVE   項目群名      TO  項目群名通知域.
   [MOVE   処理種別      TO  処理種別通知域.]
   [MOVE   制御情報      TO  特殊制御情報通知域.]
    WRITE  表示レコード名.
    CLOSE  ファイル名.
END PROGRAM  プログラム名.
```

注: ()内は、COPY 文の展開を表します。

環境部(ENVIRONMENT DIVISION)

表示ファイルを定義します。表示ファイルは、通常のファイルを定義するときと同様に、入出力節のファイル管理段落にファイル管理記述項を記述します。ファイル管理記述項に記述する内容を、以下の表に示します。なお、これらの情報は、FORM で作成した帳票定義体の定義内容とは関係なく値を決めることができます。

表: ファイル管理記述項に指定する情報

	指定する場所	情報の種類	指定する内容および用途
必須	SELECT 句	ファイル名	COBOL プログラム中で使用するファイル名を指定します。このファイル名は、COBOL の利用者語の規則に従った名前にします。

	ASSIGN 句	ファイル参照子	"GS-ファイル識別名"の形式で指定します。このファイル識別名は、実行時に接続製品が使用するプリンタ情報ファイルのパス名を設定する環境変数情報となります。
	FORMAT 句	データ名	作業場所節または連絡節で、8 桁の英数字項目として定義したデータ名を指定します。このデータ名には、帳票印刷を行うとき、帳票定義体名を設定します。
	GROUP 句	データ名	作業場所節または連絡節で、8 桁の英数字項目として定義したデータ名を指定します。このデータ名には、帳票印刷を行うとき、出力の対象となる項目群名を設定します。
	SYMBOLIC DESTINATION 句	出力先の指定	"PRT" を指定します。
任意	FILE STATUS 句	データ名	作業場所節または連絡節で、2 桁の英数字項目として定義したデータ名を指定します。このデータ名には、入出力処理の実行結果が設定されます。(注 1)詳細情報は、４桁の英数字項目を指定します。
	PROCESSING MODE 句	データ名	作業場所節または連絡節で、２桁の英数字項目として定義したデータ名を指定します。このデータ名には、帳票入出力を行うとき、入出力処理の処理種別を設定します。 (注 2)
	UNIT CONTROL 句	データ名	作業場所節または連絡節で、6 桁の英数字項目として定義したデータ名を指定します。このデータ名には、印刷処理を行うとき、入出力処理の制御情報を設定します。(注 2)

注 1: 設定される値については、ファイル入出力状態一覧を参照してください。

注 2: 詳細については、"表:入出力処理の種類と指定する値"を参照してください。

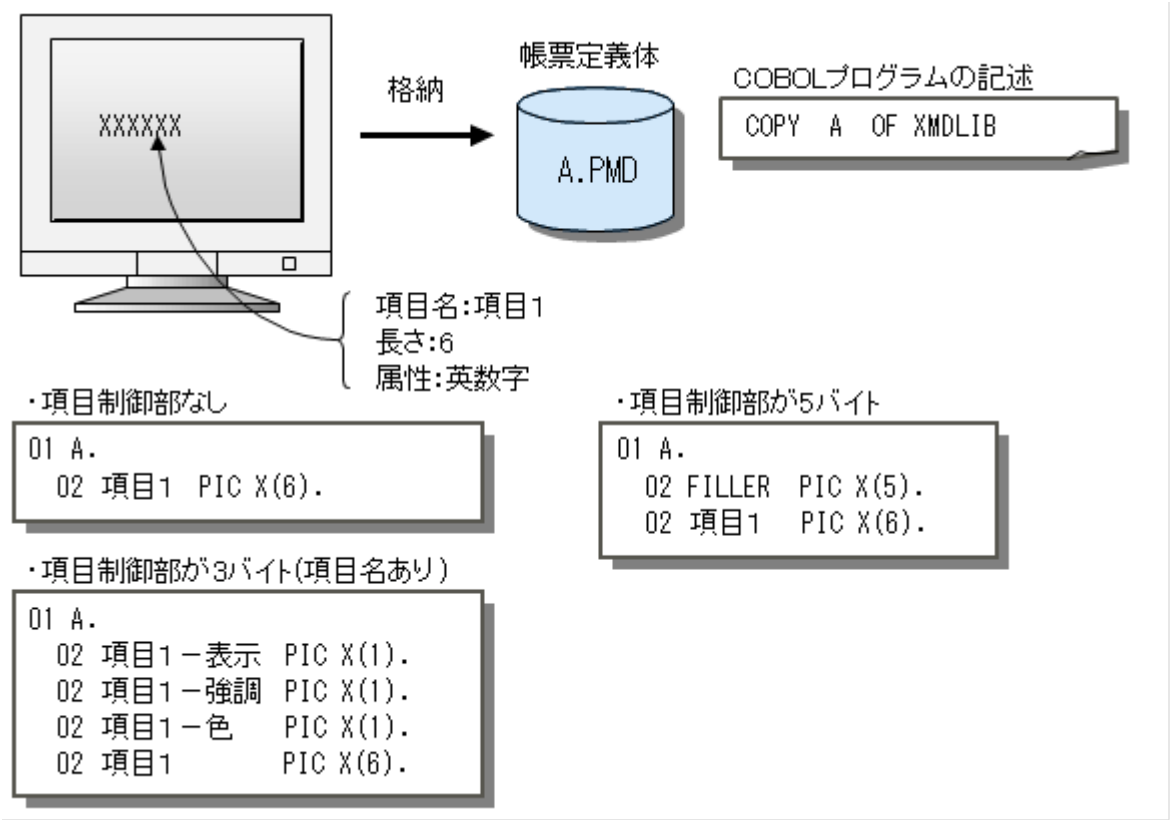
表: 入出力処理の種類と指定する値

処理種別	値	制御情報	
プリンタ装置の制御	"CT"	改ページ	"PAGE"
パーティション出力	"PW"	nnn 行改行後出力	"Annn"
		出力後 nnn 行改行	"Bnnn"
		nnn 行位置に出力	"Pnnn"
行移動出力	"FW"	後ろへ nnn 行移動	"Annn"
		前へ nnn 行移動	"Snnn"

データ部(DATA DIVISION)

データ部には、表示レコードの定義およびファイル管理記述項に指定したデータ名の定義を記述します。

表示レコードに定義するレコード記述文は、IN/OF XMDLIB を指定した COPY 文を使って帳票定義体から取り込むことができます。展開されるレコードの内容を以下に示します。



手続き部(PROCEDURE DIVISION)

帳票の印刷処理には、通常のファイル処理を行うときと同様に、入出力文を使います。

入出力は、以下に示す順序で実行します。

1. OUTPUT または I-O 指定の OPEN 文: 印刷処理の開始
2. WRITE 文: 帳票の出力
3. CLOSE 文: 印刷処理の終了

〔OPEN 文および CLOSE 文〕

OPEN 文は印刷処理の開始時に、CLOSE 文は印刷処理の終了時に、それぞれ 1 回だけ実行します。

〔WRITE 文〕

1 回の WRITE 文では、1 つの帳票が印刷されます。印刷に使用する帳票定義体の名前は、WRITE 文を実行する前に、FORMAT 句に指定したデータ名に設定しておく必要があります。

WRITE 文では、GROUP 句に指定したデータ名に設定されている項目群に属するデータ項目が印刷の対象となります。また、WRITE 文の実行前に、特殊レジスタに値を設定することにより、データ項目の属性を変更することもできます。特殊レジスタの使い方については、特殊レジスタを参照してください。

入出力エラー処理

入出力エラーの検出方法および入出力エラーが発生したときの実行結果については、入出力エラー処理を参照してください。

プログラムのビルド

コンパイラオプション/formpath で、帳票定義体を格納したファイルの格納先を指定します。

複数の帳票定義体を使用し、その拡張子がそれぞれ異なっている場合、コンパイラオプション/formext で、拡張子を指定します。

NCOBU00326

プリンタ情報ファイルの作成

ここでは、表示ファイル機能を使って帳票印刷を行うときのプリンタ情報ファイルに設定する情報および注意事項について記述します。プリンタ情報ファイルの詳しい内容や作成方法については、"MeFt ユーザーズガイド" を参照してください。

プリンタ情報ファイルに設定する代表的な情報を、以下の表に示します。

表: プリンタ情報ファイルの設定情報

情報の種類	指定する内容および用途
PRTDRV	出力するプリンタ装置のデバイス名を指定します。
MEDDIR	帳票定義体を格納したフォルダのパス名を設定します。
MEDSUF	帳票定義体を格納したファイルの拡張子を指定します。省略した場合の拡張子は MeFt の定めた省略値となります。

プログラムの実行

表示ファイル機能を使った帳票印刷を行うプログラムを実行するときには、以下の環境設定が必要です。

MeFt を使用する場合

- 環境変数 PATH に MeFt の格納されているフォルダを追加しておく必要があります。(注 1)
- ファイル識別名を環境変数情報名として、プリンタ情報ファイル名を設定します。

備考：プリンタ情報ファイルを相対パス名で指定した場合、次の順序で検索されます。

1. 環境変数 MEFTDIR に設定したフォルダ。(注 1)
2. カレントフォルダ。

注 1: MeFt に指定する環境変数は、アプリケーション構成ファイルに指定しても有効になりません。 以下の方法で設定する必要があります。

- SET コマンドでの設定
- OS のユーザ環境変数に設定

NCOBU00328

整列併合機能（SORT 文および MERGE 文の使い方）

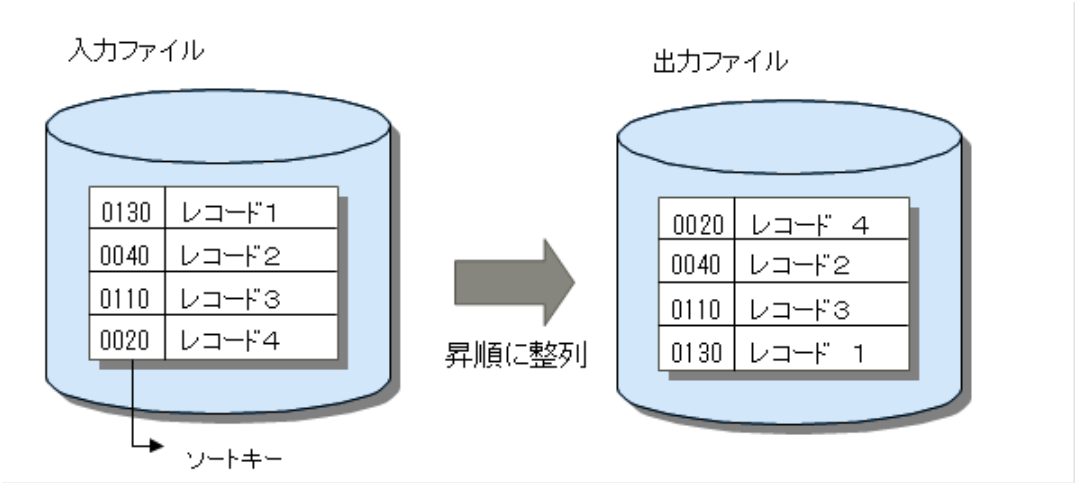
ファイルのレコードを一定の順序に並べ替えることをソート(整列)といい、複数のファイルを 1 つのファイルに並べ替えることをマージ(併合)といいます。ここでは、ソート・マージ(整列・併合)機能について説明します。

このセクションの内容

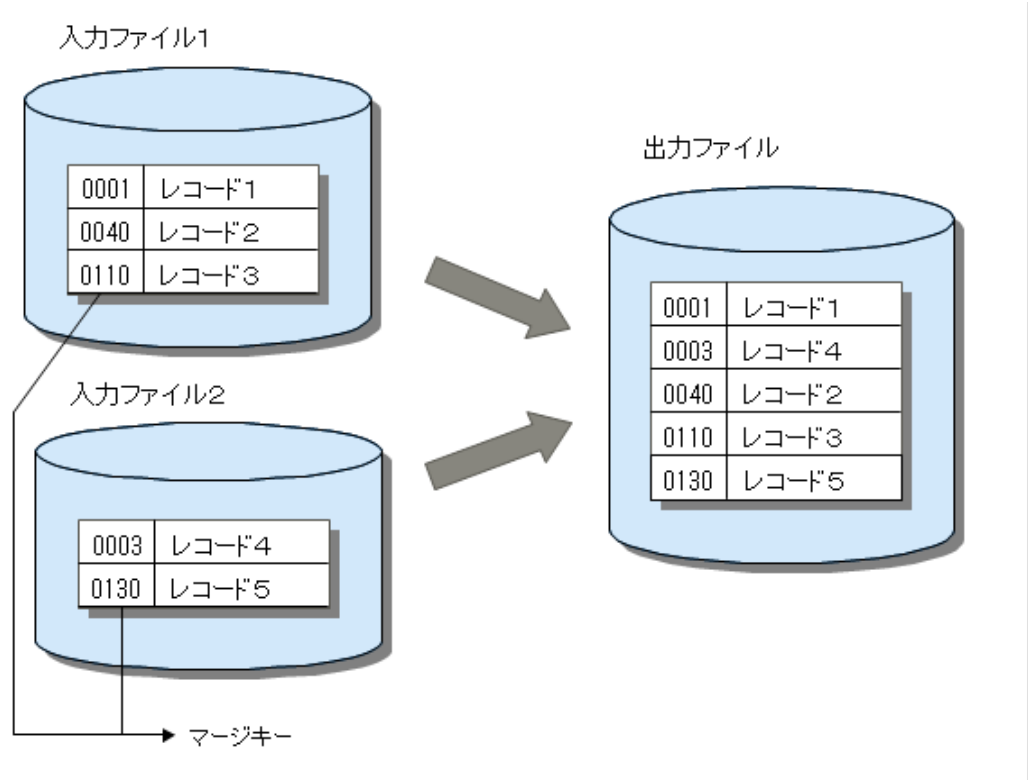
ソート・マージ処理の概要

ここでは、ソート(整列)処理と、マージ(併合)処理の概要を説明します。

ソートとは、ファイル中のレコードの情報をキーとして、レコードを昇順または降順に並べ替える処理です。レコードの並べ替えは、プログラムのキー項目の属性に従って行われます。



マージとは、昇順または降順に整列(ソート)された複数のファイルを 1 つのファイルにまとめることです。



NCOBU00330

ソートの使い方

ここでは、ソート処理の種類、プログラムの書き方、ビルドおよび実行方法について説明します。

このセクションの内容

ソート処理の種類

ソート処理には、次の 4 種類の方法があります。

ソートの種類		入力	出力
[1]	入力ファイルのレコードすべてを昇順または降順に出力ファイルに出力する方法	ファイル	ファイル
[2]	入力ファイルのレコードすべてを昇順または降順に出力データとして扱う方法	ファイル	レコード
[3]	特定のレコードまたはデータを昇順または降順に出力ファイルに出力する方法	レコード	ファイル
[4]	特定のレコードまたはデータを昇順または降順に出力データとして扱う方法	レコード	レコード

通常、入力ファイル(ソートするファイル)のレコードの内容を変更しないで、そのままソートする場合は、[1]または[2]を使います。入力ファイルを使用しない場合やレコードの内容の変更を行う場合は、[3]または[4]を使います。

また、ソートしたレコードの内容を変更しないで、そのまま出力ファイルに書き出す場合は、[1]または[3]を使います。出力ファイルを使用しない場合やレコードの内容を変更する場合は、[2]または[4]を使います。

NCOBU00332

プログラムの記述

ソートを使うプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
     SELECT  ソートファイル１ ASSIGN TO  SORTWK01.
    [SELECT  入力ファイル１   ASSIGN TO  ファイル参照子１  ～.]
    [SELECT  出力ファイル１   ASSIGN TO  ファイル参照子２  ～.]
DATA DIVISION.
 FILE SECTION.
 SD  ソートファイル１
    [RECORD  レコードの大きさ].
 01  ソートレコード１.
     02  ソートキー１  ～.
     02  データ１     ～.
 FD  入力ファイル１  ～.
 01  入力レコード１.
       レコード記述項
 FD  出力ファイル１  ～.
 01  出力レコード１.
       レコード記述項
PROCEDURE DIVISION.
    SORT   ソートファイル１ ON
        { ASCENDING  | DESCENDING }  KEY  ソートキー１
          { USING  入力ファイル１   | INPUT  PROCEDURE IS  入力手続き１}
          { GIVING 出力ファイル１   | OUTPUT PROCEDURE IS  出力手続き１} .
   入力手続き１ SECTION.
     OPEN INPUT   入力ファイル１.
   入力開始.
     READ   入力ファイル１ AT END GO TO 入力終了.
     MOVE   入力レコード１ TO ソートレコード１.
     RELEASE  ソートレコード１.
     GO TO  入力開始.
   入力終了.
     CLOSE  入力ファイル１.
   入力開始終了.
     EXIT.
   出力手続き１ SECTION.
     OPEN OUTPUT  出力ファイル１.
   出力開始.
     RETURN ソートファイル１ AT END GO TO 出力終了 END-RETURN.
     MOVE   ソートレコード１ TO 出力レコード１.
     WRITE  出力レコード１.
     GO TO  出力開始.
   出力終了.
     CLOSE  出力ファイル１.
   出力開始終了.
     EXIT.
END PROGRAM  プログラム名.
```

環境部 (ENVIRONMENT DIVISION)

以下のファイルを定義します。

- 整列併合用ファイル
- 入力ファイル
- 出力ファイル

整列併合用ファイル

ソート処理を行うための作業ファイルを定義します。 英字で始まる 8 文字以内の英数字を指定する必要があります。なお、ASSIGN 句は注釈とみなされます。

同じプログラムでマージ処理を行う場合、整列併合用ファイルの定義は 1 つだけ行います。

入力ファイル

ソート処理で入力ファイルを使用するときには、通常のファイル処理を行うときと同様にファイルを定義します。

出力ファイル

ソート処理で出力ファイルを使用するときには、通常のファイル処理を行うときと同様にファイルを定義します。

データ部 (DATA DIVISION)

環境部に定義したファイルのレコードを定義します。

手続き部 (PROCEDURE DIVISION)

ソート処理には、SORT 文を使います。 ソート処理の入力と出力がファイルかレコードかによって、SORT 文の記述内容が異なります。

PowerSORT をインストールしている際に、特殊レジスタ SORT-CORE-SIZE を用いると、PowerSORT が使用するメモリ空間の容量を限定することができます。 この特殊レジスタは、暗に PIC S9(8) COMP-5 で定義された数字項目です。 設定する値は、バイト単位の数値です。

入力がファイルの場合	"USING 入力ファイル名"を記述します。
入力がレコードの場合	"INPUT PROCEDURE 入力手続き名"を記述します。
出力がファイルの場合	"GIVING 出力ファイル名"を記述します。
出力がレコードの場合	"OUTPUT PROCEDURE 出力手続き名"を記述します。

INPUT PROCEDURE で指定した入力手続きでは、RELEASE 文を使って、ソートするレコードを 1 件ずつ受け渡します。

OUTPUT PROCEDURE で指定した出力手続きでは、RETURN 文を使って、ソート済みのレコードを 1 件ずつ受け取ります。

ソートキーは複数指定することができます。

ソート処理が終了すると、ソート処理の結果が特殊レジスタ SORT-STATUS に設定されます。特殊レジスタ SORT-STATUS は、COBOL コンパイラによって自動的に生成されるので、COBOL プログラムの中で定義する必要はありません。SORT 文の実行後に特殊レジスタ SORT-STATUS の値を検査することにより、ソート処理が異常終了しても、COBOL プログラムの実行を続行させることができます。また、SORT 文で指定した入力手続きまたは出力手続きの中で、SORT-STATUS に 16 を設定することにより、ソート処理を終了させることもできます。以下の表に、特殊レジスタ SORT-STATUS に設定される値と意味を示します。

表: SORT-STATUS に設定される値と意味

値	意味
0	正常終了
16	異常終了

USING 指定の入力ファイルおよび GIVING 指定の出力ファイルを使用する場合、SORT 文実行時にそれらのファイルが開かれた状態であってはなりません。

特殊レジスタ SORT-CORE-SIZE は、翻訳オプション SMSIZE および実行時オプション smsize に指定する値の意味と等価ですが、同時に指定された場合の優先順位は、特殊レジスタ SORT-CORE-SIZE が一番強く、以降、実行時オプション smsize、翻訳オプション SMSIZE の順で弱くなります。

例：

```
特殊レジスタ     MOVE 102400 TO SORT-CORE-SIZE
(102400=100 キロです)
翻訳オプション   SMSIZE(500000)
実行時オプション smsize300k
```

この場合、一番強い特殊レジスタ SORT-CORE-SIZE の値 100 キロバイトを優先します。

使用するメモリサイズを限定する機能は、別製品の PowerSORT をインストールしている場合だけ有効です。

プログラムのビルドと実行

プログラムのビルド

必要に応じて翻訳オプション EQUALS を指定します。

EQUALS は、ソート処理でソートキーの値が同じレコードが複数存在する場合、出力するレコードの順序を、レコードを入力した順序と同じにすることを保証することを指定します。 ただし、この翻訳オプションを指定すると実行性能が低下します。

プログラムの実行

ソートを使ったプログラムは、以下の手順で実行します。

1. 環境変数 BSORT_TMPDIR の設定

   ソート処理では、整列併合用ファイルという作業用ファイルが必要です。整列併合用ファイルは、環境変数 BSORT_TMPDIR に指定したフォルダに一時的に作成されます。環境変数の指定がない場合には、環境変数 TEMP に指定したフォルダに一時的に作成されます。なお、これらの環境変数は、あらかじめ設定しておく必要があります。

2. 入力ファイルおよび出力ファイルの割当て

   入力ファイルおよび出力ファイルの定義にファイル識別名を使用した場合、ファイル識別名を環境変数情報名として、入力ファイルおよび出力ファイルの名前を設定します。

3. プログラムの実行

   プログラムを実行します。

ソートマージ処理は、通常、COBOL ランタイムシステムを使用して処理を行いますが、PowerSORT をインストールした場合、PowerSORT を使用します。

COBOL ランタイムを利用した場合、ソート処理で利用できるデータの総量は 1G バイトまでです。 より大きいデータをソートする場合は PowerSORT をご利用ください。

PowerSORT を使用する場合の作業用ファイルは、以下の優先順位に従います。

1. 環境変数 BSORT_TMPDIR で指定されたフォルダ
2. 環境変数 TEMP で指定されたフォルダ
3. 環境変数 TMP で指定されたフォルダ
4. Windows システムのフォルダ

BSORT_TMPDIR などのソートで指定する環境変数は、アプリケーション構成ファイルに指定しても有効になりません。 以下の方法で設定する必要があります。

- SET コマンドでの設定
- OS のユーザ環境変数に設定

NCOBU00334

マージの使い方

ここでは、マージ処理の種類、プログラムの書き方、ビルドおよび実行方法について説明します。

このセクションの内容

マージ処理の種類

マージ処理には、次の 2 種類の方法があります。

マージの種類		入力	出力
[1]	すでに昇順または降順にソートされた複数のファイルのレコードすべてを、昇順または降順に出力ファイルに出力する方法	ファイル	ファイル
[2]	すでに昇順または降順にソートされた複数のファイルのレコードすべてを、昇順または降順に出力データとして扱う方法	ファイル	レコード

通常、マージしたレコードの内容を変更しないで、そのまま出力ファイルに書き出す場合は、[1]を使います。出力ファイルを使用しない場合やレコードの内容を変更する場合は、[2]を使います。

NCOBU00336

プログラムの記述

マージを使うプログラムの記述内容について、COBOL の各部ごとに説明します。

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  プログラム名.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
     SELECT  マージファイル１  ASSIGN TO SORTWK01.
     SELECT  入力ファイル１    ASSIGN TO ファイル参照子１  ～.
     SELECT  入力ファイル２    ASSIGN TO ファイル参照子２  ～.
    [SELECT  出力ファイル１    ASSIGN TO ファイル参照子３  ～.]
DATA DIVISION.
 FILE SECTION.
 SD  マージファイル１
    [RECORD  レコードの大きさ].
 01  マージレコード１.
     02  マージキー１  ～.
     02  データ１      ～.
 FD  入力ファイル１    ～.
 01  入力レコード１.
       レコード記述項
 FD  入力ファイル２    ～.
 01  入力レコード２.
       レコード記述項
 FD  出力ファイル１    ～.
 01  出力レコード１.
       レコード記述項
PROCEDURE DIVISION.
    MERGE  マージファイル１ ON
        { ASCENDING  | DESCENDING } KEY  マージキー１
          USING   入力ファイル１  入力ファイル２  ～
         { GIVING 出力ファイル１  | OUTPUT PROCEDURE IS  出力手続き１  } .
   出力手続き１ SECTION.
     OPEN OUTPUT  出力ファイル１.
   出力開始.
     RETURN マージファイル１ AT END GO TO 出力終了 END-RETURN.
     MOVE   マージレコード１ TO 出力レコード１.
     WRITE  出力レコード１.
     GO TO  出力開始.
   出力終了.
     CLOSE  出力ファイル１.
   出力開始終了.
     EXIT.
END PROGRAM  プログラム名.
```

環境部 (ENVIRONMENT DIVISION)

以下のファイルを定義します。

- 整列併合用ファイル
- 入力ファイル
- 出力ファイル

整列併合用ファイル

マージ処理を行うための作業ファイルを定義します。 英字で始まる 8 文字以内の英数字を指定する必要があります。なお、ASSIGN 句は注釈とみなされます。

同じプログラムでソート処理を行う場合、整列併合用ファイルの定義は 1 つだけ行います。

入力ファイル

マージ処理の入力ファイルは、通常のファイル処理を行うときと同様にファイルを定義します。

出力ファイル

マージ処理で出力ファイルを使用するときには、通常のファイル処理を行うときと同様にファイルを定義します。

データ部 (DATA DIVISION)

環境部に定義したファイルのレコードを定義します。

手続き部 (PROCEDURE DIVISION)

マージ処理には、MERGE 文を使います。 マージ処理の出力がファイルかレコードかによって、MERGE 文の記述内容が異なります。

PowerSORT をインストールしている際に、特殊レジスタ SORT-CORE-SIZE を用いると、PowerSORT が使用するメモリ空間の容量を限定することができます。この特殊レジスタは、暗に PIC S9(8) COMP-5 で定義された数字項目です。設定する値は、バイト単位の数値です。

出力がファイルの場合	"GIVING 出力ファイル名" を記述します。
出力がレコードの場合	"OUTPUT PROCEDURE 出力手続き名" を記述します。

OUTPUT PROCEDURE で指定した出力手続きでは、RETURN 文を使って、マージ済みのレコードを 1 件ずつ受け取ります。

マージキーは複数指定することができます。

マージ処理が終了すると、マージ処理の結果が特殊レジスタ SORT-STATUS に設定されます。 特殊レジスタ SORT-STATUS は、COBOL コンパイラによって自動的に生成されるので、一般のデータとは異なり、COBOL プログラムの中で定義する必要はありません。MERGE 文の実行後に特殊レジスタ SORT-STATUS の値を検査することにより、マージ処理が異常終了しても、COBOL プログラムの実行を続行させることができます。また、MERGE 文で指定した出力手続きの中で、特殊レジスタ SORT-STATUS に 16 を設定することにより、マージ処理を終了させることができます。以下の表に、特殊レジスタ SORT-STATUS に設定される値と意味を示します。

表: SORT-STATUS に設定される値と意味

値	意味
0	正常終了
16	異常終了

入力ファイルおよび GIVING 指定の出力ファイルは、MERGE 文実行時にそれらのファイルが開かれた状態であってはなりません。

特殊レジスタ SORT-CORE-SIZE は、翻訳オプション SMSIZE および実行時オプション smsize に指定する値の意味と等価ですが、同時に指定された場合の優先順位は、特殊レジスタ SORT-CORE-SIZE が一番強く、以降、実行時オプション smsize、翻訳オプション SMSIZE の順で弱くなります。

例:

```
特殊レジスタ     MOVE 102400 TO SORT-CORE-SIZE
(102400=100 キロです)
翻訳オプション   SMSIZE(500000)
実行時オプション smsize300k
```

この場合、一番強い特殊レジスタ SORT-CORE-SIZE の値 100 キロバイトを優先します。

使用するメモリサイズを限定する機能は、別製品の PowerSORT をインストールしている場合だけ有効です。

プログラムのビルドと実行

プログラムのビルド

とくに必要な翻訳オプションおよび追加ライブラリはありません。

プログラムの実行

マージを使ったプログラムは、以下の手順で実行します。

1. 環境変数 BSORT_TMPDIR の設定

   マージ処理では、整列併合用ファイルという作業用ファイルが必要です。整列併合用ファイルは、環境変数 BSORT_TMPDIR に指定したフォルダに一時的に作成されます。環境変数の指定がない場合には、環境変数 TEMP に指定したフォルダに一時的に作成されます。なお、これらの環境変数は、あらかじめ設定しておく必要があります。

2. 入力ファイルおよび出力ファイルの割当て

   入力ファイルおよび出力ファイルの定義にファイル識別名を使用した場合、ファイル識別名を環境変数情報名として、入力ファイルおよび出力ファイルの名前を設定します。

3. プログラムの実行

   プログラムを実行します。

ソートマージ処理は、通常、COBOL ランタイムシステムを使用して処理を行いますが、PowerSORT をインストールした場合、PowerSORT を使用します。

PowerSORT を使用する場合の作業用ファイルは、以下の優先順位に従います。

1. 環境変数 BSORT_TMPDIR で指定されたフォルダ
2. 環境変数 TEMP で指定されたフォルダ
3. 環境変数 TMP で指定されたフォルダ
4. Windows システムのフォルダ

BSORT_TMPDIR などのソートで指定する環境変数は、アプリケーション構成ファイルに指定しても有効になりません。 以下の方法で設定する必要があります。

- SET コマンドでの設定
- OS のユーザ環境変数に設定

NCOBU01162

CSV 形式データの操作

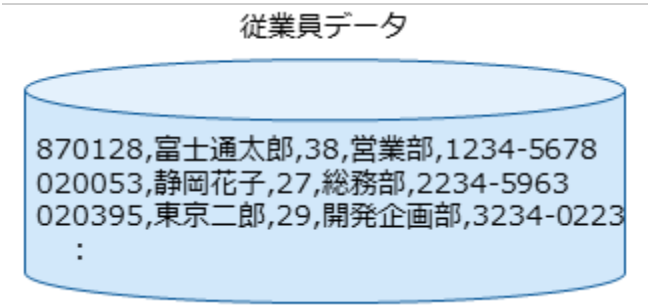
STRING 文および UNSTRING 文を使用した CSV 形式データの操作について説明します。NetCOBOL for .NET では、容易に CSV 形式データを操作できるよう、STRING 文および UNSTRING 文に拡張機能を用意しました。

このセクションの内容

CSV 形式データとは

CSV(Comma Separated Values)形式データは、カンマで区切られた複数の文字列データの並びで、表計算ソフトやデータベースソフトで多く用いられてきました。最近では、これらソフトウェアに限らず、各種ツール類やミドルウェアとのデータ連携でも用いられるようになっています。

例えば、以下のようにテキストファイルで流通します。



これは、従業員番号、氏名、年齢、所属、内線をカンマで区切った CSV データです。

このデータを COBOL プログラムで入力および編集する場合、行順ファイルとしてレコード単位に読み込み、 カンマで区切られた文字列データを集団項目に従属する基本項目に分解して操作するのが一般的です。 しかし、従来の言語仕様で実現するのは容易ではありませんでした。

上記の従業員データを以下の集団項目へ転記する場合を考えます。

```
01 従業員.
  02 従業員番号  PIC 9(6).
  02 氏名        PIC N(10).
  02 年齢        PIC 9(2).
  02 所属        PIC N(10).
  02 内線        PIC X(10).
```

うち PERFORM 文を用いて CSV データを先頭から 1 文字ずつ検査しながら転記することもできますが、ここでは、UNSTRING 文(書き方 1)を使用した場合で考えます。

```
FILE-SECTION.
FD CSV-FILE
01 CSV-REC PIC X(80).
   :
WORKING-STORAGE SECTION.
01 従業員.
  02 従業員番号  PIC 9(6).
  02 氏名        PIC N(10).
  02 氏名－Ｒ    REDEFINES 氏名 PIC X(20).    *> 字類を合わせるために追加
  02 年齢        PIC 9(2).
  02 所属        PIC N(10).
  02 所属－Ｒ    REDEFINES 所属 PIC X(20).    *> 字類を合わせるために追加
  02 内線        PIC X(10).
   :
READ CSV-FILE.
UNSTRING CSV-REC
        DELIMITED BY ","                    *> カンマで分解
        INTO 従業員番号 氏名－Ｒ 年齢 所属－Ｒ 内線
END UNSTRING.
   :
```

簡単に記述できるように見えますが、上の例には以下の問題があります。

- UNSTRING 文は、転記の規則に従って、受取り側の字類を合わせなければなりません。 上の例では、REDEFINE 句を使って解決していますが、 実行時コード系が Unicode の場合は、エンコードが異なる

ため、解決できません。

- CSV 形式データでは､データ全体を引用符で囲めば､カンマをデータとして使用することができます。しかし、上の例では、そのようなデータを処理することはできません。
- 引用符をデータとして使用する場合、連続する 2 つの引用符で 1 つの引用符を表現します。 しかし、上の例では、そのようなデータを処理することはできません。

上記の通り、UNSTRING 文(書き方 1)を使って CSV 形式データを分解することは、困難でした。 また、逆に、STRING 文(書き方 1)を使って CSV 形式データを生成する場合も、 同様の問題(後置空白の処理などを考慮すると、更に大きな問題)を抱えていました。

NetCOBOL for .NET では、STRING 文および UNSTRING 文に、 CSV 形式のデータ操作に特化した新たな構文を追加して、容易に操作できるようにしました。

CSV 形式データの作成（STRING 文）

ここでは、CSV 形式データを作成する方法を説明します。

このセクションの内容

NCOBU01165

基本操作

集団項目に格納されている文字列データを、従属する項目単位で CSV 形式へ編集する場合、 STRING 文(書き方 2)を使用します。

```
WORKING-STORAGE SECTION.
77 従業員編集 PIC X(80).
01 従業員.
  02 従業員番号 PIC 9(6). *> 870128 が格納されている状態
  02 氏名 PIC N(10).      *> 富士通太郎 (以下同)
  02 年齢 PIC 9(2).       *> 38
  02 所属 PIC N(10).      *> 営業部
  02 内線 PIC X(10).      *> 1234-5678
        :
    MOVE SPACE TO 従業員編集.
    STRING 従業員 INTO 従業員編集 BY CSV-FORMAT.
```

上の例の STRING 文を実行すると、データ項目“従業員編集”には、以下の CSV 形式データが格納されます。

```
870128,富士通太郎,38,営業部,1234-5678
```

なお、CSV 形式を生成する際、格納されているデータを以下のとおり編集します。

- 送出し側項目と受取り側項目の字類が異なる場合、受取り側項目の字類に合わせ、以下のように変換します。
  - 送出し側項目が日本語の場合、Unicode 動作時にはエンコードの変換を行います。
  - 送出し側項目が符号つき数字項目の場合、SIGN 句の指定に関わらず、符号を左端に付加します。また、小数部を含む場合、小数点文字を付加します。
- 送出し側データ中に区切り文字が含まれていた場合、データ全体を二重引用符で囲みます。
- 送出し側データ中に二重引用符が含まれていた場合、連続する 2 つの二重引用符に置き換え、 データ全体を二重引用符で囲みます。
- 送出し側項目の字類が英数字または日本語の場合、後置空白は削除します。
- 送出し側項目が数字項目の場合、先行するゼロ列は削除します。 ただし、値がゼロだった場合は、1 けたのゼロを転記します。 また、小数部を含む場合、後置ゼロは削除します。
- TYPE 指定に従い、データ全体を二重引用符で囲みます。 詳細は、“CSV 形式のバリエーション”を参照してください。

STRING 文の文法や仕様の詳細は、“COBOL 文法書”を参照してください。

- 受取り側への転記処理は、STRING 文が作用した部分しか実行されません。 したがって、文字転記のような後続領域への空白づめは期待できません。 受取り側項目は、STRING 文を実行する前に必ず初期化してください。
- List Creator では、CSV 形式データ中に 2 つの二重引用符が含まれていた場合でも、1 つの二重引用符に置き換えません。

処理異常の検出

CSV 形式データ作成時における異常とは、以下の状態を指します。

表：CSV 形式データ作成時における異常

(1)	送出し側項目に不正なデータが格納されている
(2)	受取り側項目が小さく、全てのデータが入り切らない
(3)	POINTER 指定のデータ項目の値が 1 より小さい

STRING 文では、ON OVERFLOW 指定を記述することによって異常発生時の動作を記述することができます。このとき、正常に処理できたところまで、受取り側に格納されます。

例えば、前述の例で受取り側領域の大きさが 20 けたしか用意されてなかった場合、 以下のように ON OVERFLOW 指定を記述すると、異常を検知することができます。

```
WORKING-STORAGE SECTION.
77 従業員編集 PIC X(20).
01 従業員.
  02 従業員番号 PIC 9(6). *> 870128 が格納されている状態
  02 氏名 PIC N(10).      *> 富士通太郎 (以下同)
  02 年齢 PIC 9(2).       *> 38
  02 所属 PIC N(10).      *> 営業部
  02 内線 PIC X(10).      *> 1234-5678
     :
    MOVE SPACE TO 従業員編集.
    STRING 従業員 INTO 従業員編集 BY CSV-FORMAT
      ON OVERFLOW DISPLAY "編集に失敗しました。データ=" 従業員編集
    END-STRING.
```

なお、ON OVERFLOW 指定が記述されてない場合に、“表：CSV 形式データ作成時における異常”が発生した時、 以下のように動作します。

異常(1),(3)の場合

実行時エラーを出力後、異常終了します。

異常(2)の場合

実行時エラーを出力後、STRING 文の次の文へ制御を移します。

NCOBU01167

CSV 形式データの分解（UNSTRING 文）

ここでは、CSV 形式データを分解する方法を説明します。

このセクションの内容

基本操作

UNSTRING 文(書き方 2)を使用して、CSV 形式データを分解する方法を説明します。

処理異常の検出

CSV 形式データ分解時における異常終了時の動作を説明します。

基本操作

CSV 形式データを分解して、集団項目に従属する項目へ転記する場合、UNSTRING 文(書き方 2)を使用します。

```
WORKING-STORAGE SECTION.
77 従業員データ PIC X(80)
            VALUE "870128,富士通太郎,38,営業部,1234-5678".
01 従業員.
02 従業員番号 PIC 9(6).
02 氏名 PIC N(10).
02 年齢 PIC 9(2).
02 所属 PIC N(10).
02 内線 PIC X(10).
     :
   UNSTRING 従業員データ(1:FUNCTION STORED-CHAR-LENGTH(従業員データ))
     INTO 従業員 BY CSV-FORMAT.
```

上の例の UNSTRING 文を実行すると、集団項目“従業員”に従属する各基本項目には、 先頭から順番にカンマで分割されたデータが格納されます。 なお、CSV 形式データの分解では、データを以下のとおり編集します。

- 送出し側項目と受取り側項目の字類が異なる場合、受取り側項目の字類に合わせ、次のような変換を行います。
  - 受取り側項目が日本語の場合、Unicode 動作時には文字コードの変換を行います。
  - 受取り側項目が数字の場合、受取り側の SIGN 句の指定に応じ、符号処理を行います。 また、小数部を含む数値の場合、桁合わせを行います。
- 分割したデータが二重引用符で囲まれていた場合、二重引用符を除いてから転記します。
- 二重引用符で囲まれた分割データ中に、連続する二重引用符が含まれていた場合、 1 つの二重引用符に置き換えます。
- 分解したデータを受取り側項目へ転記する際は、転記の規則に従います。

UNSTRING 文の文法や仕様の詳細は、“COBOL 文法書”を参照してください。

TSV 形式データが格納されているテキストファイルを、行順ファイルを用いて読み込み、 UNSTRING 文を使用して分割すると、意図したとおりに動作しません。 これは、READ 文が実行されるタイミングで、タブが空白に置き換えられるためです。 行順ファイルに高速処理(",BSAM")を指定すれば、タブが空白に置き換えられず、正しく処理することができます。

[参照]行順ファイルの処理、ファイルの高速処理

NCOBU01169

処理異常の検出

CSV 形式のデータ分解時における異常とは、以下の状態を指します。

表：CSV 形式データ分解時における異常

(1)	送出し側項目に不正なデータが格納されている。
(2)	受取り側項目への転記の際、けたあふれが発生する。
(3)	分解した文字列データの数が、受取り側項目の数より多い。
(4)	POINTER 指定のデータ項目の値が 1 より小さい、もしくは受取り側項目の桁数より大きい。

UNSTRING 文では、ON OVERFLOW 指定を記述することで、異常発生時の動作を記述することができます。このとき、正常に処理できたところまで、受取り側に格納されます。

例えば、前述の例で、受取り側項目に"内線"の定義を忘れていた場合、 以下のように ON OVERFLOW 指定を記述すると、異常を検知することができます。

```
WORKING-STORAGE SECTION.
77 従業員データ PIC X(80)
             VALUE "870128,富士通太郎,38,営業部,1234-5678".
01 従業員.
  02 従業員番号 PIC 9(6).
  02 氏名       PIC N(10).
  02 年齢       PIC 9(2).
  02 所属       PIC N(10).
:
    UNSTRING 従業員データ(1:FUNCTION STORED-CHAR-LENGTH(従業員データ))
      INTO 従業員 BY CSV-FORMAT
      ON OVERFLOW DISPLAY "データの分解に失敗しました。"
    END-UNSTRING.
```

このとき、POINTER 指定を記述しておくと、異常発生の原因となった送出し側データの文字位置を取得できるので、 より詳細な情報を得ることができます。

また、TALLYING 指定を記述した場合には、転記に成功した項目数を得ることもできます。

```
WORKING-STORAGE SECTION.
77 CNT PIC 9(2).
    :
    MOVE 1 TO CNT.
    UNSTRING 従業員データ(1:FUNCTION STORED-CHAR-LENGTH(従業員データ))
      INTO 従業員 BY CSV-FORMAT POINTER CNT
      ON OVERFLOW DISPLAY "データの分解に失敗しました。"
      DISPLAY " 失敗データ= " 従業員データ(CNT:)
    END-UNSTRING.
```

なお、ON OVERFLOW 指定が記述されてない場合に、“表：CSV 形式データ分解時における異常”が発生した時、 以下のように動作します。

異常(1),(4)の場合

実行時エラーを出力後、異常終了します。

異常(2)の場合

実行時エラーを出力後、UNSTRING 文の処理を継続します。

異常(3)の場合

実行時エラーを出力後、UNSTRING 文の次の文へ制御を移します。

NCOBU01170

CSV 形式のバリエーション

ここでは、CSV 形式のバリエーションについて説明します。

CSV 形式には、ISO が制定した国際規格のように正式に成立した仕様はありません。 そのため、Microsoft 社の表計算ソフトである Excel の仕様をデファクトスタンダードとして、 いくつかの派生形式が流通している状況にあります。

NetCOBOL for .NET では、STRING 文(書き方 2)によって生成する CSV 形式について、 以下の 4 つのバリエーションを選択することができます。 データ連携する相手に合わせて指定してください。

バリエーション	内容
MODE-1	・ 送出し側データ中に区切り文字または二重引用符が存在する場合、 データ全体を二重引用符で囲みます。 ・ 送出し側データ項目の字類が英数字または日本語の場合で、 全て空白が格納されていた場合、区切り文字のみを転記します。
MODE-2	・ 送出し側データを二重引用符で囲みます。 ・ 送出し側データ項目の字類が英数字または日本語の場合で、 全て空白が格納されていた場合、区切り文字のみを転記します。
MODE-3	・ 送出し側データ項目の字類が数字以外の場合、データ全体を二重引用符で囲みます。 ・ 送出し側データの字類が英数字または日本語の場合で、 全て空白が格納されていた場合、区切り文字のみを転記します。
MODE-4	・ 送出し側データ項目の字類が数字以外の場合、データ全体を二重引用符で囲みます。 ・ 送出し側データの字類が英数字または日本語の場合で、 全て空白が格納されていた場合、連続する 2 つの二重引用符を転記します。

これらバリエーションは、STRING 文(書き方 2)の TYPE 指定、 または実行環境変数@CBR_CSV_TYPE（生成する CSV 形式のバリエーション）で指定します。 なお、省略時は、MODE-1 が選択されたものとみなします。以下に例を示します。

```
01 従業員.
  02 従業員番号 PIC 9(6) VALUE 870128.
  02 氏名       PIC N(10) VALUE NC"富士通太郎".
  02 所属       PIC N(10) VALUE NC"営業部".
  02 役職       PIC X(10) VALUE SPACE.
  02 内線       PIC X(20) VALUE "1234-5678,4536".
```

上の例のデータ項目“従業員”を送出し側項目に指定した場合、 バリエーションの指定によって、それぞれ以下の結果となります。

バリエーション	結果
MODE-1	870128,富士通太郎,営業部,, "1234-5678,4536"
MODE-2	"870128","富士通太郎","営業部",,"1234-5678,4536"
MODE-3	870128,"富士通太郎","営業部",,"1234-5678,4536"
MODE-4	870128,"富士通太郎","営業部","","1234-5678,4536"

なお、いずれの CSV 形式データも、 UNSTRING 文(書き方 2)を用いて集団項目に従属する項目へ分解および転記することができます。

NCOBU00338

データベースアクセス

データベース(SQL)機能は、埋込み SQL を使用して PC クライアントからサーバ上のデータベースをアクセスするための機能です。埋込み SQL（Structured Query Language）とは、COBOL ソースプログラム中に記述されたデータベース操作言語 SQL です。

NetCOBOL for .NET では、埋込み SQL 文を使用してデータベースをアクセスする接続方法が 2 つあります。

- ADO.NET データプロバイダーを使用するデータベース接続
- ODBC ドライバを使用するデータベース接続

接続方法の指定は、アプリケーション構成ファイルに設定します。そのため、COBOL ソースプログラムを変更せずに、接続方法を切り替えることができます。

ここでは、COBOL プログラムに埋込み SQL を記述し、ADO.NET データプロバイダー経由または、ODBC ドライバを経由してデータベースをアクセスする方法について説明します。

ADO.NET は、サーバカーソルをサポートしていません。 ADO.NET 接続によるデータベースアクセス機能では、COBOL プログラムに記述されたカーソルは、 COBOL ランタイムがクライアントカーソルとしてエミュレートしています。 サーバカーソルを利用したい場合は、ODBC 接続によるデータベースアクセス機能を使用してください。 ADO.NET データプロバイダー利用時の注意事項については、ADO.NET データプロバイダー使用時の注意事項を参照してください。

このセクションの内容

データベースアクセス概要

COBOL プログラムから、データベースにアクセスする場合の概要について説明します。

サンプルデータベース

このセクションで使用する、サンプルデータベースについて説明します。

コネクション操作

コネクションとは、データベースをアクセスするためにクライントとサーバの間を結んだ接続関係のことをいいます。コネクションの接続、切断、変更などについて説明します。

データの操作

さまざまなデータ操作を行なう、埋込み SQL の記述方法について説明します。

ストアドプロシージャの呼出し

ストアドプロシージャの概要とストアドプロシージャの呼出し方法について説明します。

高度なデータ操作

一度に複数の行にアクセスする技術について説明します。

[オブジェクト指向プログラミング機能を使用したデータベースアクセス]

オブジェクト指向プログラミング機能を使用して、データベースにアクセスする方法を説明します。

[プログラムのビルド]

プログラムのビルドについて説明します。

[プログラムの実行]

プログラムを実行するために必要な環境設定について説明します。

[ADO.NET データプロバイダー使用時の注意事項]

ADO.NET 経由でデータベースにアクセスする場合の注意事項について説明します。

[ODBC ドライバ使用時の注意事項]

ODBC ドライバを使用する場合および特定の ODBC ドライバを使用する場合の注意事項について説明します。

[Azure SQL データベース使用時の注意事項]

Azure SQL データベースを使用する場合の注意事項について説明します。

関連トピックス

[SQL 情報]

SQL 情報を示しています。

[埋込み SQL 文のキーワード一覧]

埋込み SQL 文のキーワード一覧を示しています。

[埋込み SQL 文で使用可能なホスト変数]

埋込み SQL 文に指定できるホスト変数の形式を示しています。

[SQLSTATE/SQLCODE/SQLMSG]

埋め込み SQL 文を実行した時に通知される情報について示します。

[データ型の対応]

ADO.NET 接続または、ODBC 接続で扱うデータと COBOL で扱うデータの対応を示しています。

NCOBU00840

データベースアクセス概要

このセクションの内容

ADO.NET 概要

ADO.NET 接続によるデータベースアクセスの構成図を示します。

ODBC 概要

ODBC 接続によるデータベースアクセスの構成図を示します。

COBOL プログラムの構成

COBOL プログラムの全体構成について説明します。

埋込み SQL 文による操作

埋込み SQL 文で行うことのできる操作について説明します。

ADO.NET 概要

ここでは、COBOL アプリケーションから ADO.NET 接続でデータベースにアクセスする場合の構成について説明します。

ADO.NET データプロバイダー経由でのデータベースアクセスでは、 埋込み SQL は、指定された接続文字列情報により、ローカルまたはリモートのデータベースにアクセスできる適切な ADO.NET データプロバイダーのクラスライブラリ呼出しに変換されます。

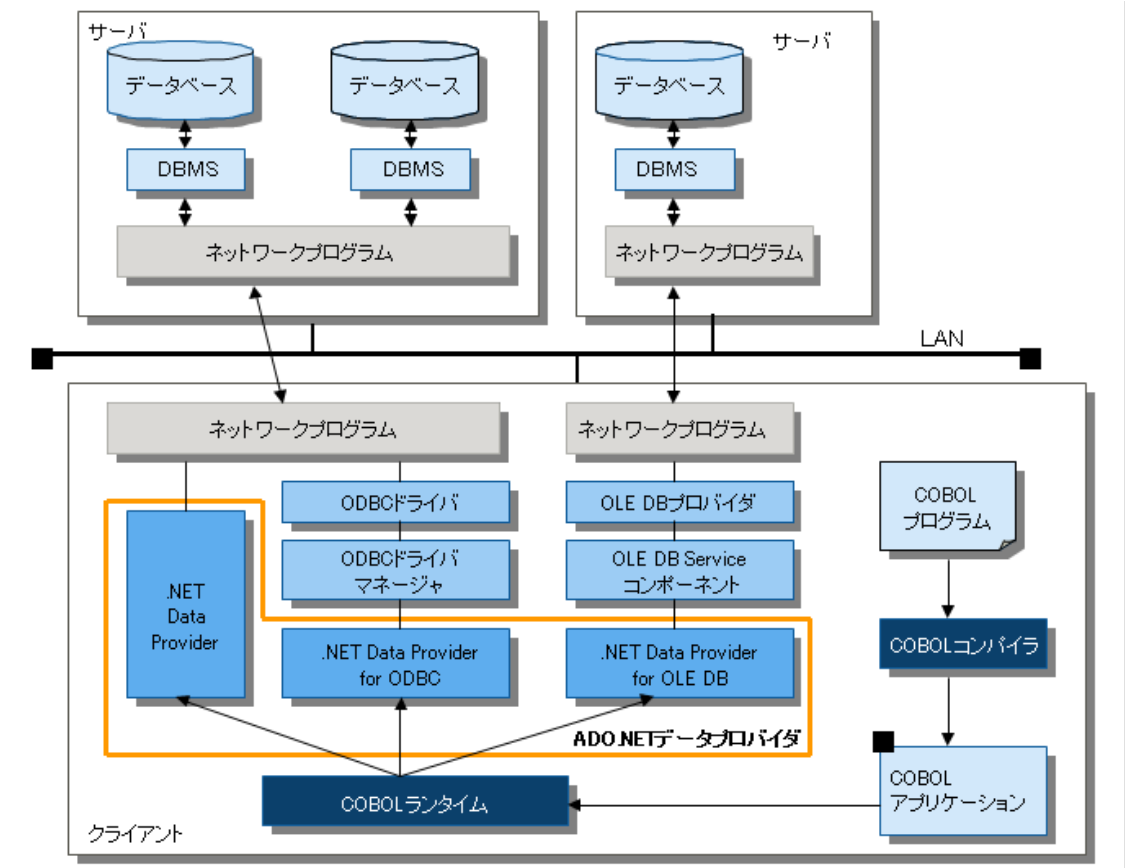


図: ADO.NET 接続でのデータベースアクセスの概要図

参考

ADO.NET についての詳細は、.NET Framework のドキュメントの ADO.NET を参照して下さい。

NCOBU00339

ODBC 概要

ここでは、COBOL アプリケーションから ODBC 接続でデータベースをアクセスする場合の概要について説明します。

ODBC ドライバ経由のアクセスでは、埋込み SQL は、設定された接続情報により、ローカルまたはリモートのデータベースにアクセスできる適切な ODBC 呼出しに変換されます。

> ODBC(Open DataBase Connectivity)は、Microsoft が提唱している、データベースをアクセスするためのアプリケーションプログラムインタフェースです。

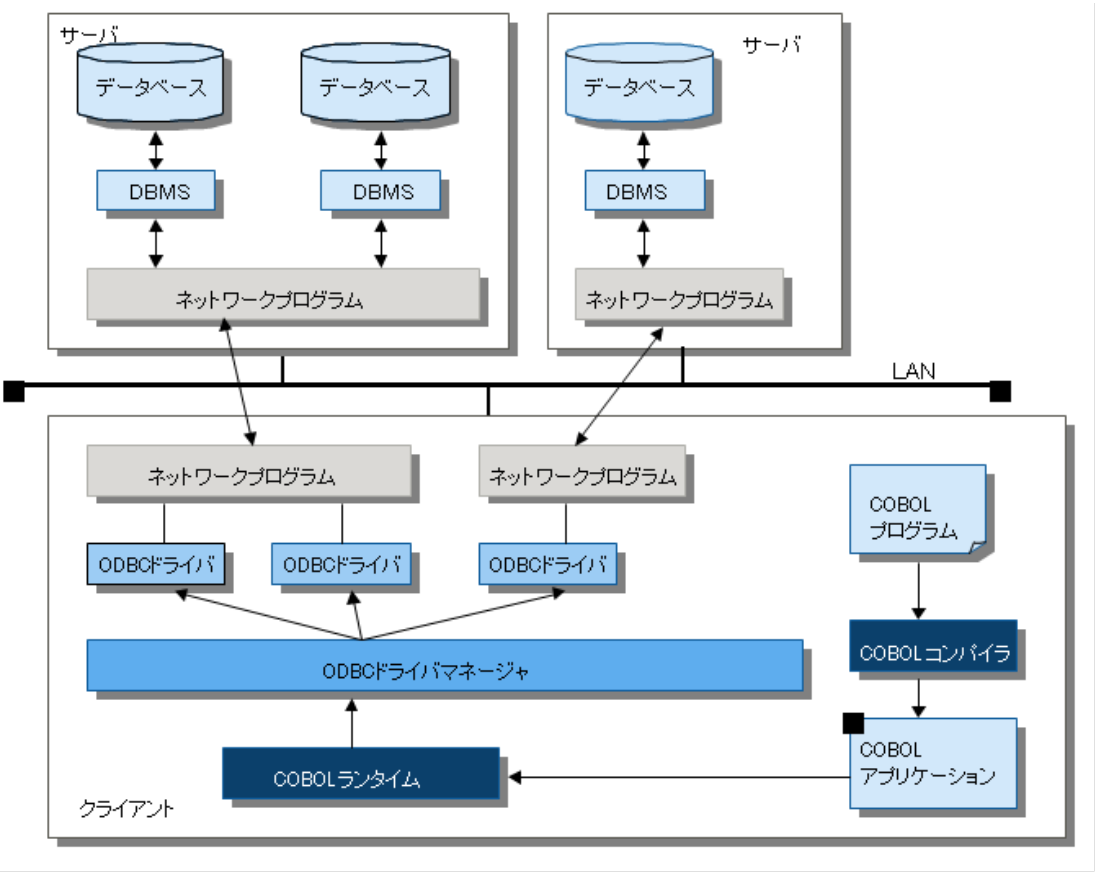


図: ODBC ドライバを使用したデータベースアクセスの概要図

COBOL プログラムの構成

COBOL プログラムの全体構成を以下に示します。

```
      IDENTIFICATION DIVISION.
        :
      ENVIRONMENT DIVISION.                       *>[1]
        :
      DATA DIVISION.
        :
      WORKING-STORAGE SECTION.
        :
          EXEC SQL BEGIN DECLARE SECTION END-EXEC.    *>[2]
      01 SQLSTATE      PIC X(5).
        :
          EXEC SQL END DECLARE SECTION END-EXEC.
        :
      PROCEDURE DIVISION.
        :
          EXEC SQL CONNECT ... END-EXEC             *>[3]
        :
          EXEC SQL DECLARE CUR1 ... END-EXEC         *>[4]
        :
          EXEC SQL OPEN CUR1 END-EXEC              *>[5]
        :
          EXEC SQL FETCH CUR1 ... END-EXEC
        :
          EXEC SQL CLOSE CUR1 END-EXEC             *>[6]
        :
          EXEC SQL ROLLBACK WORK END-EXEC
        :
          EXEC SQL DISCONNECT ... END-EXEC          *>[7]
        :
          STOP RUN.
```

図: COBOL プログラムの全体構成

[図の説明]

[1]: 環境部に SQL 固有の記述はありません。

[2]: 宣言節内で SQLSTATE を宣言します。また、必要ならホスト変数を定義します。

[3]: サーバに接続します。

[4]: カーソルを使用する場合、カーソルを宣言します。

[5]: カーソルを開きます。

[6]: カーソルを閉じます。

[7]: サーバを切断します。

上の図に示すように、COBOL プログラム中に SQL を記述する場合は、"EXEC SQL"(SQL 先頭子)と"END-EXEC"(SQL 終了子)で囲んで記述します。手続き部に記述された SQL 文によって、実際のデータベースの処理が行われます。

NCOBU00341

埋込み SQL 文による操作

埋込み SQL 文を記述することにより、以下の操作ができます。

コネクションの接続

コネクションとは、データベースをアクセスするためにクライアントとサーバの間を結んだ接続関係のことです。この接続関係を結ぶことにより、クライアントからサーバのデータベースをアクセスする SQL 文を実行することができます。

コネクションを接続するには、CONNECT 文を使います。CONNECT 文の使い方については、コネクションの接続を参照してください。

コネクションの変更

コネクションを変更するためには、SET CONNECTION 文を使います。SET CONNECTION 文の使い方については、コネクションの変更を参照してください。

データ操作

データ操作とは、データベースにデータを設定したり、データベースに格納されているデータを参照したりすることです。データ操作は、データベースの 1 つの行だけを対象に行うことも、複数の行を対象に行うこともできます。

データを操作するためには、SELECT 文、INSERT 文、UPDATE 文および DELETE 文を使います。カーソルを定義し、FETCH 文によってデータを取り出すこともできます。また、これらの文を動的に実行することもできます。データの操作については、データの操作」および「高度なデータ操作を参照してください。

> ADO.NET または、ODBC ドライバで扱うデータとの対応は、「データ型の対応」に示しています。

ストアドプロシージャの呼出し

ストアドプロシージャを呼び出すには、CALL 文を使います。

呼び出すストアドプロシージャは、データベース上に登録されていなければなりません。ストアドプロシージャについては、ストアドプロシージャの呼出しを参照してください。

> ADO.NET または、ODBC ドライバで扱うデータとの対応は、「データ型の対応」に示しています。

トランザクション処理

トランザクションは、データベースに対するデータ操作の一貫性を保証する単位です。トランザクションは、最初の SQL 文を実行したときに開始され、COMMIT 文または ROLLBACK 文を実行したときに終了します。

コネクションの切断

プログラムとサーバとのコネクションの切断は、DISCONNECT 文で行います。DISCONNECT 文の使い方については、コネクションの切断を参照してください。

コネクションを切断する前には、トランザクションを終了させておかなくてはなりません。

サンプルデータベース

このセクションのプログラム例で使用するサンプルデータベースの 3 つの表について説明します。

- STOCK 表(在庫表)

  製品番号(GNO)、製品名(GOODS)、在庫数量(QOH)、倉庫番号(WHNO)を格納しています。

- ORDERS 表(注文表)

  取引番号(ORDERID)、取引先番号(COMPANYNO)、取引製品番号(GOODSNO)、仕入れ価格(PRICE)、発注数量(OOH)を格納しています。

- COMPANY 表(会社表)

  会社番号(CNO)、会社名(NAME)、電話番号(PHONE)、住所(ADDRESS)を格納しています。

STOCK 表(在庫表)

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2

351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

ORDERS 表(注文表)

ORDERID	COMPANYNO	GOODSNO	PRICE	OOH
1	61	123	48000	60
2	61	124	64000	40
3	61	138	6400	180
4	61	140	8000	80
5	61	215	240000	10
6	61	240	80000	20
7	62	110	37500	120
8	62	226	112500	20
9	62	351	375	800
10	63	111	57400	80
11	63	200	123000	60
12	63	201	164000	50
13	63	212	205000	30
14	63	215	246000	10
15	71	140	7800	50
16	71	351	390	600
17	72	137	3500	120
18	72	140	7000	70
19	72	215	210000	10
20	72	226	105000	20
21	72	243	84000	10

22	72	351	350	1000
23	73	141	16000	60
24	73	380	2400	250
25	73	390	2400	150
26	74	110	39000	120
27	74	111	54000	120
28	74	226	117000	20
29	74	227	140400	10
30	74	351	390	700

COMPANY 表(会社表)

CNO	NAME	PHONE	ADDRESS
61	ADAM LTD.	731-1111	SANTA CLARA C.A USA
62	IDEA INC.	433-2222	LONDON W.C.2 ENGLAND
63	MOON CO.	143-3333	FIFTH AVENUE N.Y USA
71	RIVER CO.	344-1212	SAKAI OOSAKA JAPAN
72	DRAGON CO.	373-7777	YAO OOSAKA JAPAN
73	BIG INC.	391-0808	HARAJUKU TOKYO JAPAN
74	FIRST CO.	255-9955	SYDONEY AUSTRALIA

NCOBU00343

コネクション操作

ここでは、コネクションの接続方法、切断方法および変更方法について説明します。

このセクションの内容

コネクションの接続

ここでは、CONNECT 文でサーバとコネクションを接続する方法について説明します。

以下の手順で、サーバとコネクションを接続します。

1. サーバの情報を登録します。
2. 以下のどちらかの方法で、サーバとコネクションを接続します。
    - サーバ名を指定した接続
    - DEFAULT を指定した接続

このセクションの内容

NCOBU00842

サーバ名を指定した接続

プログラムを実行する前に、サーバの情報を SQL 情報(ODBC 情報)に設定します。定義するサーバ情報の詳細と定義方法については、プログラムの実行を参照してください。

サーバ名を指定して CONNECT 文を実行すると、指定したサーバ名の情報を SQL 情報(ODBC 情報)内で検索します。同名のセクションが見つかると、そこからサーバの情報を参照してサーバとコネクションを接続します。

サーバ名の SQL 情報は、アプリケーション構成ファイル、または、ODBC 情報ファイルに指定します。NetCOBOL for .NET では、アプリケーション構成ファイルに設定することを推奨しています。アプリケーション構成ファイルの設定は、実行環境設定ユーティリティを使用して設定してください。

アプリケーション構成ファイルに SQL 情報を設定する場合

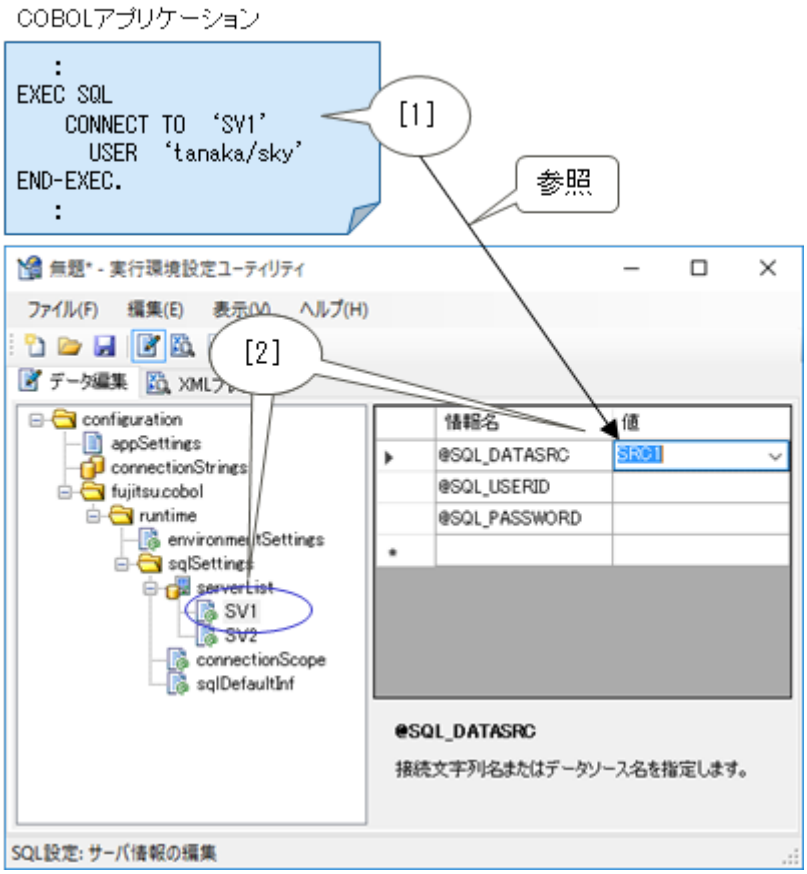


図:アプリケーション構成ファイルを使用しサーバ名を指定して実行する方法

[図の説明]

[1]: CONNECT 文にサーバ名を指定して実行します。

[2]: CONNECT 文実行時に、SQL 情報よりサーバとコネクションを接続します。

ODBC 情報ファイルに SQL 情報を設定する場合

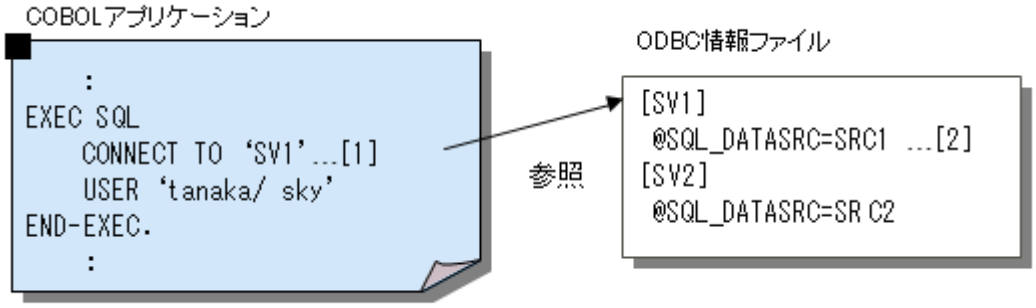


図:ODBC 情報ファイルにサーバ名を指定した接続

[図の説明]

[1]: CONNECT 文にサーバ名を指定して実行します。

[2]: CONNECT 文の実行時に、ODBC 情報ファイルの情報を参照して、サーバとコネクションを接続します。

NCOBU00843

DEFAULT を指定した接続

プログラムを実行する前に、デフォルトコネクションの情報を SQL 情報(ODBC 情報)に設定しておきます。定義するサーバ情報の詳細と定義方法については、プログラムの実行を参照してください。

DEFAULT を指定して CONNECT 文を実行すると、SQL 情報(ODBC 情報)を設定するファイルのデフォルトコネクション情報を参照します。デフォルトコネクション情報にはサーバ名の情報が設定されているので、さらにそのサーバ名で SQL 情報(ODBC 情報)が設定されたファイル内を検索します。サーバ名の情報が見つかると、その情報を参照してサーバとコネクションを接続します。

DEFAULT の SQL 情報は、アプリケーション構成ファイル、または、ODBC 情報ファイルに設定します。NetCOBOL for .NET では、アプリケーション構成ファイルに設定することを推奨しています。アプリケーション構成ファイルは、実行環境設定ユーティリティを使用して設定してください。

アプリケーション構成ファイルに SQL 情報を設定する場合

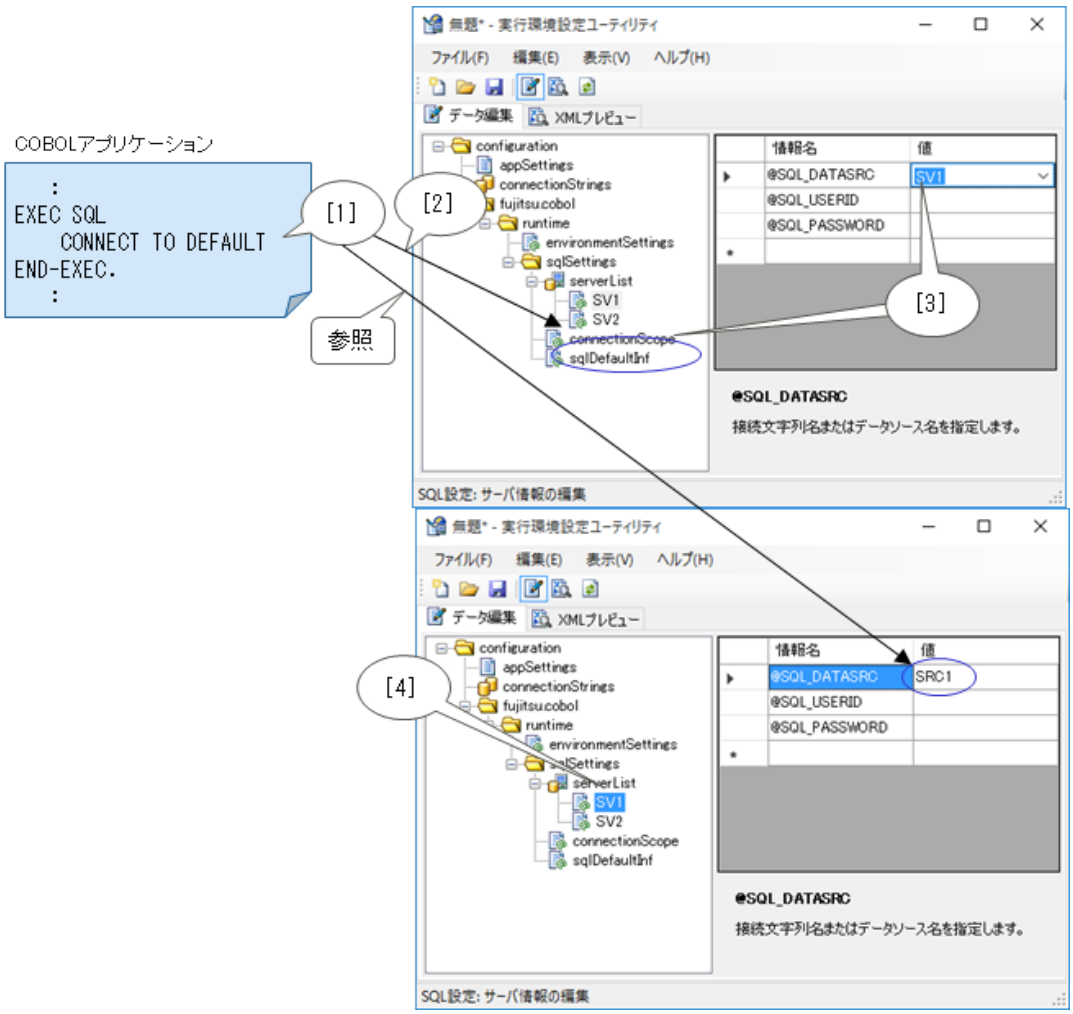


図:アプリケーション構成ファイルを使用して DEFAULT を指定して実行する方法

[図の説明]

[1]:CONNECT 文に DEFAULT を指定して実行します。

[2]:CONNECT 文実行時に、デフォルトコネクション情報を参照します。

[3]:さらに、デフォルトコネクション情報のサーバ名を取得します。

[4]:取得した SV1 の情報を参照して、サーバとコネクションを接続します。

ODBC 情報ファイルに設定する場合

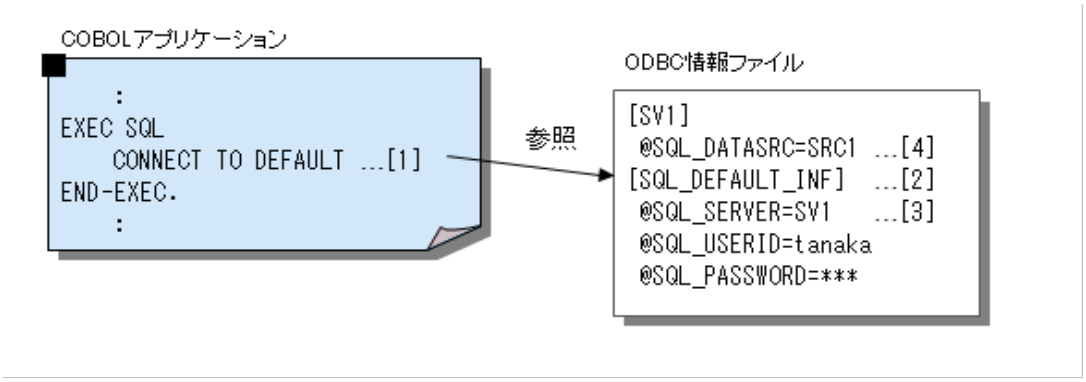


図:ODBC 情報ファイルに DEFAULT を指定した接続方法

[図の説明]

[1]: CONNECT 文に DEFAULT を指定して実行します。

[2]:CONNECT 文の実行時に、ODBC 情報ファイルのデフォルトコネクション情報を参照します。

[3]: さらに、デフォルトコネクション情報のサーバ名を取得します。

[4]: 取得した SV1 の情報を参照して、サーバとコネクションを接続します。

NCOBU00345

コネクションの切断

コネクションを切断するには、DISCONNECT 文を使用します。

DISCONNECT 文には、切断の対象となるコネクションを指定します。切断対象となるコネクションには、コネクション名、DEFAULT、CURRENT、および、ALL が指定できます。それぞれの使い方は、複数コネクションを接続/変更/切断する例を参照してください。

コネクションの変更

コネクションを変更するには、SET CONNECTION 文を使用します。

複数の CONNECT 文が実行されると、複数のコネクションが接続されます。アプリケーションが複数のコネクションを持つ場合、SQL 文の実行対象となるコネクションを決めておく必要があります。複数の CONNECT 文が実行された場合、最後の CONNECT 文で接続されたコネクションが現在のコネクションとなり、SQL 文の実行対象となります。現在のコネクションを変更するには、SET CONNECTION 文を実行します。SET CONNECTION 文を実行することによって、SQL 文の実行対象となるコネクションを変更することができます。

NCOBU00347

複数コネクションを接続/変更/切断する例

ここでは、複数コネクションの接続、変更、切断を行う COBOL プログラムの例を示します。

この例では、デフォルトコネクションのほかに、"SV1"、"SV2"、"SV3"および"SV4"という名前の SQL サーバが、アプリケーション構成ファイル、または、ODBC 情報ファイルに登録されています。

```
       EXEC SQL CONNECT TO DEFAULT END-EXEC              *>[1]
       EXEC SQL
         CONNECT TO 'SV1' AS 'CNN1' USER 'tanaka/sky'
       END-EXEC                                         *>[2]
       EXEC SQL
         CONNECT TO 'SV2' AS 'CNN2' USER 'tanaka/sky'
       END-EXEC                                         *>[3]
       EXEC SQL
         CONNECT TO 'SV3' AS 'CNN3' USER 'tanaka/sky'
       END-EXEC                                         *>[4]
       EXEC SQL
         CONNECT TO 'SV4' AS 'CNN4' USER 'tanaka/sky'
       END-EXEC                                         *>[5]
       EXEC SQL DISCONNECT 'CNN4' END-EXEC              *>[6]
       EXEC SQL SET CONNECTION 'CNN1' END-EXEC          *>[7]
      *      :
       EXEC SQL ROLLBACK WORK END-EXEC                  *>[8]
       EXEC SQL DISCONNECT CURRENT END-EXEC             *>[9]
       EXEC SQL SET CONNECTION DEFAULT END-EXEC         *>[10]
      *      :
       EXEC SQL COMMIT WORK END-EXEC                    *>[11]
       EXEC SQL DISCONNECT DEFAULT END-EXEC             *>[12]
       EXEC SQL SET CONNECTION 'CNN2' END-EXEC
      *      :
       EXEC SQL ROLLBACK WORK END-EXEC
       EXEC SQL DISCONNECT ALL END-EXEC                 *>[13]
       STOP RUN.
```

[プログラムの説明]

[1]: デフォルトコネクションに接続します。

[2]: SQL サーバ"SV1"に接続します。この接続の名前を"CNN1"とします。

[3]: SQL サーバ"SV2"に接続します。この接続の名前を"CNN2"とします。

[4]: SQL サーバ"SV3"に接続します。この接続の名前を"CNN3"とします。

[5]: SQL サーバ"SV4"に接続します。この接続の名前を"CNN4"とします。最後に接続した"CNN4"が現在のコネクションになります。

[6]: コネクション名"CNN4"を指定して、コネクションを切断します。

[7]: コネクション名"CNN1"を指定し、コネクションを変更します。"CNN1"が現在のコネクションになります。

[8]: コネクション"CNN1"中のデータの変更をすべて取り消します。

[9]: 現在のコネクションを切断します。現在のコネクションは"CNN1"なので、"CNN1"が切断されます。

[10]: DEFAULT を指定してコネクションを変更します。デフォルトコネクションが現在のコネクションになります。

[11]: デフォルトコネクション中のデータの変更をすべて保存します。

[12]: デフォルトコネクションを切断します。

[13]: ALL を指定した DISCONNECT 文を実行することによって、接続されているすべてのコネクションを切断します。

CONNECT 文により同時に接続できるコネクション数に制限はありませんが、ドライバやデータプロバイダーなどにより制限を受けることがあります。

NCOBU00348

データの操作

ここでは、さまざまなデータ操作を行う、埋込み SQL の使用方法を説明します。

このセクションの内容

データの参照

データベースのデータを参照する方法について説明します。

データの更新

データベースのデータを更新する方法について説明します。

データの削除

データベースのデータを削除する方法について説明します。

データの挿入

データベースのデータを挿入する方法について説明します。

動的 SQL

動的 SQL の使用方法について説明します。

可変長文字列の使用

可変長文字列データを使用する方法について説明します。

複数コネクションでのカーソル操作

複数コネクションでのカーソル操作について説明します。

関連トピックス

ADO.NET で扱うデータ型との対応

ADO.NET で扱うデータと COBOL で扱うデータの対応を示しています。

ODBC で扱うデータとの対応

ODBC で扱うデータと COBOL で扱うデータの対応を示しています。

データの参照

データベースのデータを参照する方法について説明します。

表の全行を参照する

データベースのすべての行を参照する方法について説明します。

以下は、サンプルデータベースの STOCK 表のすべての行を参照する COBOL プログラムの例です。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
      01 在庫表.
        02 製品番号             PIC S9(4) COMP-5.             *>|
        02 製品名               PIC X(20).                   *>|
        02 在庫数量             PIC S9(9) COMP-5.             *>|[1]
        02 倉庫番号             PIC S9(4) COMP-5.             *>|
      01 SQLSTATE               PIC X(5).                     *>|
       EXEC SQL END DECLARE SECTION END-EXEC.
      PROCEDURE DIVISION.
       EXEC SQL WHENEVER NOT FOUND GO TO :P-END END-EXEC.    *>[2]
       EXEC SQL
         DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
       END-EXEC.                                             *>[3]
      P-START.
       EXEC SQL CONNECT TO DEFAULT END-EXEC.                 *>[4]
       EXEC SQL OPEN CUR1 END-EXEC.                          *>[5]
      P-LOOP.
       EXEC SQL
         FETCH CUR1 INTO :在庫表
       END-EXEC.                                            *>[6]
      *   :
       GO TO P-LOOP.
      P-END.
       EXEC SQL CLOSE CUR1 END-EXEC.                         *>[7]
       EXEC SQL ROLLBACK WORK END-EXEC.                      *>[8]
       EXEC SQL DISCONNECT DEFAULT END-EXEC.                 *>[9]
       STOP RUN.
```

[プログラムの説明]

[1]: 作業場所節に埋込み SQL 宣言節を記述し、STOCK 表の全列に対応するデータの入力用領域をホスト変数として定義します。ホスト変数宣言の規則については、COBOL 文法書の「8.2.2 ホスト変数定義」を参照してください。

[2]: 埋込み例外宣言を記述すると、SQL 文で例外が発生したときの動作を指定することができます。ここでは、条件として"NOT FOUND"を指定しているので、SQLSTATE の値が"データなし"を示すときに有効になります。すなわち、[6]の FETCH 文によって次々と行を取り出し、取り出す行がなくなったときに、手続き名"P-END"の位置に分岐することを指定しています。

> [補足]: COBOL の IF 文で SQLSTATE を判定して、例外が発生した場合の動作を指定することもできます。

[3]: カーソル宣言によって、STOCK 表を参照するカーソル名を定義します。この例では STOCK 表に対してとくに列を選択したり、探索条件を指定していないので、問合せ式から導出される表は、元の STOCK 表と同じものになります。

[4]: CONNECT 文を実行し、サーバとのコネクションを接続します。

[5]: OPEN 文を実行し、指定したカーソルを使用可能な状態にします。

[6]: FETCH 文によって表からデータを 1 行ずつ取り出し、各列の値を対応するホスト変数の領域に設定します。

[7]: CLOSE 文を実行し、指定したカーソルを無効な状態にします。

[8]: ROLLBACK 文を実行し、トランザクションを終了します。

[9]: DISCONNECT 文を実行し、サーバとのコネクションを切断します。

問合せ式の選択リストに列名の並びを書かずにアスタリスク(*)を指定した場合、選択される列の順序は、表を定義したときに指定した順序と同じになります。

上の例では、複数列指定ホスト変数を使用しています。複数列指定ホスト変数は、データベースの各列に対応するホスト変数を 1 つの集団項目の従属項目として定義します。これを埋込み SQL 文で参照する場合には、集団項目の名前で参照します。これは一種の省略記法であり、次のようにホスト変数を定義または参照する場合と同じです。

```
     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
*    :
 01 製品番号         PIC S9(4) COMP-5.
 01 製品名           PIC X(20).
 01 在庫数量         PIC S9(9) COMP-5.
 01 倉庫番号         PIC S9(4) COMP-5.
 01 SQLSTATE         PIC X(5).
*    :
    EXEC SQL END DECLARE SECTION END-EXEC.
 PROCEDURE DIVISION.
    EXEC SQL
      FETCH  CUR1
        INTO :製品番号, :製品名, :在庫数量, :倉庫番号
    END-EXEC.
*    :
```

条件を指定して参照する

表のすべての行を参照するのではなく、条件を指定してその条件を満たす行だけを参照する方法について説明します。

以下は、サンプルデータベースの STOCK 表から、在庫数量が 50 以下の製品の製品番号、製品名および在庫数量を参照する COBOL プログラムの例です。

```
*    :
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 在庫表.
   02 製品番号          PIC S9(4) COMP-5.         *>|
   02 製品名            PIC X(20).               *>|[1]
   02 在庫数量          PIC S9(9) COMP-5.         *>|
 01 SQLSTATE            PIC X(5).                 *>|
    EXEC SQL END DECLARE SECTION END-EXEC.
 PROCEDURE DIVISION.
    EXEC SQL WHENEVER NOT FOUND GO TO :P-END END-EXEC.
    EXEC SQL
      DECLARE CUR2 CURSOR FOR
        SELECT GNO, GOODS, QOH FROM STOCK
          WHERE QOH < 51
    END-EXEC.                                    *>[2]
 P-START.
    EXEC SQL CONNECT TO DEFAULT END-EXEC.       *>[3]
    EXEC SQL OPEN CUR2 END-EXEC.                *>[4]
 P-LOOP.
    EXEC SQL
      FETCH CUR2 INTO :在庫表
    END-EXEC.                                    *>[5]
*    :
    GO TO P-LOOP.
 P-END.
    EXEC SQL CLOSE CUR2 END-EXEC.               *>[6]
    EXEC SQL ROLLBACK WORK END-EXEC.            *>[7]
    EXEC SQL DISCONNECT DEFAULT END-EXEC.       *>[8]
    STOP RUN.
```

[プログラムの説明]

[1]: 埋込み SQL 文中に指定するホスト変数は、すべて埋込み SQL 宣言節で定義します。

[2]: カーソル宣言によって表が探索されることはありません。表の探索は、[4]の OPEN 文の実行時に行われます。

[3]: CONNECT 文を実行し、サーバとのコネクションを接続します。

[4]: OPEN 文を実行すると、[2]のカーソル宣言で指定した条件を満たす行からなる仮想的な表が作られます。ここで作られる仮想表は、STOCK 表から QOH 列の値が 50 以下である行を抽出し、さらにそこから GNO 列、GOODS 列および QOH 列を抜き出した形態をしています。一般に、このような仮想表の行の順番は不定です。

[5]: FETCH 文は、[4]で導出された仮想表の先頭から、データを 1 行ずつ取り出し、各列の値を対応するホスト変数の領域に設定します。特定の列について、値の昇順または降順で取り出したい場合は、[2]の問合せ式の後に ORDER BY 句を指定します。

[6]: CLOSE 文は、[4]で導出された仮想表を無効にします。再び OPEN 文を実行しないかぎり、以降の SQL 文でこの仮想表を参照することはできません。

[7]: ROLLBACK 文を実行し、トランザクションを終了します。

[8]: DISCONNECT 文を実行し、サーバとのコネクションを切断します。

1 つの行を参照する

"表の全行を参照する"および"条件を指定して参照する"のプログラム例では、複数の行を参照する場合について説明しました。

これに対し、探索結果が 1 行しかないことがわかっている場合は、SELECT 文でデータを参照することができます。カーソルを使用しないため、カーソル宣言、OPEN 文、FETCH 文および CLOSE 文による一連の処理は不要です。

たとえば､STOCK 表から製品番号 200 の製品名と在庫数量を参照する場合は､以下の SELECT 文を実行します。

```
EXEC SQL
  SELECT GOODS, QOH INTO :製品名,:在庫数量 FROM STOCK
  WHERE GNO = 200
END-EXEC.
```

SELECT 文では、選択リストの値式に集合関数指定を記述することにより、指定された値式の最大値、最小値、平均値、総和(合計)、および表の基数(行数)を求めることができます。

ORDERS 表について仕入れ価格の最大値、最小値、および平均値を求める例では、次の SQL 文を実行します。

```
EXEC SQL
  SELECT MAX(PRICE), MIN(PRICE), AVG(PRICE)
  INTO :最大値, :最小値, :平均値 FROM ORDERS
END-EXEC.
```

結果は、ホスト変数"最大値"、"最小値"および"平均値"に設定されます。

表を関連付けてデータを参照する

ここでは、表を関連付けてデータを参照する方法について説明します。

複数表を関連付けて参照する

複数の表を、それぞれの表に含まれる列の値によって関連付けて探索し、データを参照することができます。たとえば、サンプルデータベースを使用して、特定の製品名(TELEVISION)を取り扱っている会社名と発注数量を参照するとします。この場合、サンプルデータベースの 3 つの表を以下のように関連付け、求めるデータを参照します。

- STOCK 表と ORDERS 表を、製品番号(GNO 列)と取引製品番号(GOODSNO 列)の値に基づいて関連付ける。
- ORDERS 表と COMPANY 表を、取引番号(COMPANYNO 列)と会社番号(CNO 列)の値に基づいて関連付ける。

問合せ式は、以下のようになります。

```
SELECT NAME, OOH
FROM STOCK, ORDERS, COMPANY
WHERE GOODS = 'TELEVISION' AND
     GNO = GOODSNO       AND
     COMPANYNO = CNO
```

3 つの表を関連付けた表から、探索条件を満たす行が導出されます。

GNO	…	WHNO	COMPANYNO	…	OOH	CNO	…	ADDRES
110	…	2	6	…	60	61	…	SANTA CLARA C.A USA
110	…	2	61	…	60	62	…	LONDON W.C.2 ENGLAND
110	…	2	61	…	60	63	…	FIFTH AVENUE N.Y USA
～	～	～	～	～	～	～	～	～
110	…	2	61	…	40	61	…	SANTA CLARA C.A USA
110	…	2	61	…	40	62	…	LONDON W.C.2 ENGLAND
110	…	2	61	…	40	63	…	FIFTH AVENUE N.Y USA
～	～	～	～	～	～	～	～	～
111	…	2	61	…	60	61	…	SANTA CLARA C.A USA
111	…	2	61	…	60	62	…	LONDON W.C.2 ENGLAND
111	…	2	61	…	60	63	…	FIFTH AVENUE N.Y USA
～	～	～	～	～	～	～	～	～
390	…	3	74	…	700	72	…	YAO OOSAKA JAPAN
390	…	3	74	…	700	73	…	HARAJUKU TOKYO JAPAN
390	…	3	74	…	700	74	…	SYDNEY AUSTRALIA

↓ 問合わせ式の結果

NAME	OOH
IDEA INC.	120
MOON CO.	80
FIRST CO.	120
FIRST CO.	120

同一表を関連付けて参照する

複数の表の関連付けと同様の考え方で、同一の表に対し、行と行で関連付けて探索することもできます。たとえば STOCK 表で、製品名 TELEVISION と同じ倉庫に在庫のある製品名を参照するとします。このような場合、STOCK 表に対して 2 つの異なる別名(相関名)を与え、それらをあたかも別の表であるかのように扱います。

以下は、同一の表を関連付け、製品名 TELEVISION と同じ倉庫に在庫のある製品名を参照する COBOL プログラムの例です。

```
*     :
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 製品名        PIC X(20).
 01 SQLSTATE      PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
 PROCEDURE DIVISION.
    EXEC SQL WHENEVER NOT FOUND GO TO :P-END END-EXEC.
    EXEC SQL
       DECLARE CUR4 CURSOR FOR
         SELECT DISTINCT X2.GOODS
         FROM STOCK X1,STOCK X2
         WHERE X1.GOODS = 'TELEVISION' AND
             X1.WHNO = X2.WHNO
    END-EXEC.                                    *>[1]
 P-START.
    EXEC SQL CONNECT TO DEFAULT END-EXEC.
    EXEC SQL OPEN CUR4 END-EXEC.
 P-LOOP.
    EXEC SQL
      FETCH CUR4 INTO :製品名
    END-EXEC.
*    :
    GO TO P-LOOP.
 P-END.
    EXEC SQL CLOSE CUR4 END-EXEC.
    EXEC SQL ROLLBACK WORK END-EXEC.
    EXEC SQL DISCONNECT DEFAULT END-EXEC.
    STOP RUN.
```

[プログラムの説明]

[1]: カーソル宣言文では、FROM 句で STOCK 表に対して相関名(例では"X1"および"X2")を指定し、あたかもそれが別々の表であるかのように扱います。WHERE 句の探索条件では、列名を相関名で修飾し、製品名 TELEVISION と同じ倉庫番号の倉庫に存在する製品の行を探索するように指定します。

NCOBU00350

データの更新

ここでは、データベースのデータを更新する方法について説明します。

データを更新するには、UPDATE 文を使用します。たとえば、STOCK 表の在庫数量(QOH 列)の値を一律に 10%減らす場合は、CONNECT 文によってサーバと接続した後、次の文を実行します。

```
EXEC SQL
   UPDATE STOCK SET QOH = QOH * 0.9
END-EXEC.
```

この UPDATE 文では、STOCK 表のすべての行の QOH 列の値を、それに 0.9 を乗じた値で置き換えます。

特定の条件を満たす行に対してだけ更新したいときには、その条件を UPDATE 文の WHERE 句の探索条件に指定します。たとえば、製品名 TELEVISION の在庫数量だけを 10%減らすのであれば、前述の UPDATE 文を次のように変更します。

```
EXEC SQL
   UPDATE STOCK SET QOH = QOH * 0.9
   WHERE GOODS = 'TELEVISION'
END-EXEC.
```

複数の表を関連付けて得られる表に対して、データの更新を行うことはできません。

データの削除

ここでは、データベースのデータを削除する方法について説明します。

データを削除するには、DELETE 文を使用します。たとえば、STOCK 表から製品名が"CASSETTE DECK"である行をすべて削除する場合は、CONNECT 文によってサーバと接続した後、次の文を実行します。

```
        EXEC SQL
          DELETE FROM STOCK WHERE GOODS = 'CASSETTE DECK'
        END-EXEC.
```

複数の表を関連付けて得られる表に対して、データの削除を行うことはできません。

NCOBU00352

データの挿入

ここでは、データベースにデータを挿入する方法について説明します。 データを挿入するには、INSERT 文を使用します。INSERT 文により、以下のどちらかの方法でデータを挿入できます。

- 1 行だけのデータを挿入する
- ある探索条件に従って別の表から抽出した行の集合を挿入する

単一行の挿入

"STOCK 表" に、製品番号 301 の行を追加する場合について説明します。 この行の製品名は "WASHER" 、在庫数量は 50、倉庫番号は 1 とします。このデータを挿入するには、CONNECT 文によってサーバと接続した後、次の文を実行します。

```
EXEC SQL
  INSERT INTO STOCK (GNO,GOODS, QOH, WHNO)
    VALUES (301, 'WASHER', 50, 1)
END-EXEC.
```

別表からの複数行の挿入

"STOCK 表"と同じデータベース上に別の在庫表(表名は"SUBSTOCK"で列名およびその属性は STOCK 表と同じ)があり、その表から製品名が"RADIO"であるデータを STOCK 表に挿入する場合について説明します。挿入するすべての行の倉庫番号を 2 にします。この処理を行うには、CONNECT 文によってサーバに接続した後、次の文を実行します。

```
EXEC SQL
  INSERT INTO STOCK (GNO, GOODS, QOH, WHNO)
    SELECT GNO, GOODS, QOH, 2 FROM SUBSTOCK
    WHERE GOODS = 'RADIO'
END-EXEC.
```

カーソルで開かれた状態の表に対して、非カーソルのデータ操作を実行すると、エラーが発生する場合があります。表示されるエラーメッセージは、使用するデータベースや動作設定に依存します。エラーメッセージが表示された場合は、使用するデータベースの仕様を確認してください。

動的 SQL

動的 SQL を使用することによって、プログラムの実行時に SQL 文を生成し、実行することができます。

探索条件を動的に決定する

探索条件で用いる値は、ホスト変数を使用して実行時に設定することができました。ここでは、探索条件そのものを実行時に決定する方法について説明します。

以下は、動的カーソル宣言によって実行時に探索条件を決定する COBOL プログラムの例です。この例で、カーソル定義のための問合せ式は以下のとおりです。

```
SELECT GNO, GOODS, QOH FROM STOCK
  WHERE GOODS = 'REFRIGERATOR' AND QOH < 10
```

問合せ式は、実行時に ACCEPT 文により読み込まれます。

```
     *     :
           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
      01  在庫表.
       02  製品番号          PIC S9(4) COMP-5.
       02  製品名            PIC X(20).
       02  在庫数量          PIC S9(9) COMP-5.
      01  STMVAR             PIC X(254).                  *>[1]
      01  SQLSTATE PIC X(5).
           EXEC SQL END DECLARE SECTION END-EXEC.
      PROCEDURE DIVISION.
           EXEC SQL WHENEVER NOT FOUND GO TO :P-END END-EXEC.
           EXEC SQL
             DECLARE CUR8 CURSOR FOR STMIDT
           END-EXEC.                                      *>[2]
           ACCEPT STMVAR FROM CONSOLE.                   *>[3]
      P-START.
           EXEC SQL CONNECT TO DEFAULT END-EXEC.
           EXEC SQL PREPARE STMIDT FROM :STMVAR END-EXEC.  *>[4]
           EXEC SQL OPEN CUR8 END-EXEC.                    *>[5]
      P-LOOP.
           EXEC SQL
             FETCH CUR8 INTO :在庫表                       *>[6]
           END-EXEC.
     *     :
           GO TO P-LOOP.
      P-END.
           EXEC SQL CLOSE CUR8 END-EXEC.                  *>[7]
           EXEC SQL ROLLBACK WORK END-EXEC.
           EXEC SQL DISCONNECT DEFAULT END-EXEC.
           STOP RUN.
```

[プログラムの説明]

[1]: SQL 文変数 STMVAR は、[4]の PREPARE 文で参照されます。

[2]: SQL 文識別子 STMIDT は、[4]の PREPARE 文で SQL 文変数 STMVAR と対応付けられます。

[3]: ACCEPT 文によって、問合せ式を読込み、SQL 文変数 STMVAR に設定します。

[4]: PREPARE 文によって、この時点で SQL 文変数に設定されていた文(この例では動的 SELECT 文)と SQL 文識別子 STMIDT が対応付けられます。

[5]: 動的 OPEN 文によって、STOCK 表から在庫が 10 未満の REFRIGERATOR に関する行が抽出され、それらの中から、"GNO(製品番号)"、"GOODS(製品名)"および"QOH(在庫数量)"の 3 つの列からなる表が導出されます。

[6]: 動的 FETCH 文によって、表からデータを 1 行ずつ取り出し、各列の値を対応するホスト変数に設定します。

[7]: 動的 CLOSE 文によって、指定したカーソルおよびそのカーソルに対応する表を無効状態にします。

SQL 文を動的に決定する

動的 SQL では、カーソル宣言での問合せ式以降の探索条件だけでなく、そこで実行する SQL 文を動的に変更することができます。ここでは、EXECUTE 文を用いた方法について説明します。

以下は、ACCEPT 文によって入力された SQL 文を動的に実行する COBOL プログラムの例です。実行時には、以下の UPDATE 文を入力します。

```
UPDATE STOCK SET QOH = 0 WHERE GOODS = 'TELEVISION'
```

```
     *   :
         EXEC SQL BEGIN DECLARE SECTION END-EXEC.
      01 STMVAR    PIC X(254).
      01 SQLSTATE   PIC X(5).
         EXEC SQL END DECLARE SECTION END-EXEC.
      PROCEDURE DIVISION.
         ACCEPT STMVAR FROM CONSOLE.                        *>[1]
         EXEC SQL CONNECT TO DEFAULT END-EXEC.
         EXEC SQL PREPARE STMIDT FROM :STMVAR END-EXEC.     *>[2]
         EXEC SQL EXECUTE STMIDT END-EXEC.                  *>[3]
     *    :
         EXEC SQL ROLLBACK WORK END-EXEC.
         EXEC SQL DISCONNECT DEFAULT END-EXEC.
         STOP RUN.
```

[プログラムの説明]

[1]: ACCEPT 文によって、UPDATE 文を読込み、SQL 文変数 STMVAR に設定します。

[2]: PREPARE 文によって、この時点で SQL 文変数に設定されていた文(この例では UPDATE 文)と SQL 文識別子 STMIDT が対応付けられます。

[3]: EXECUTE 文は、指定された SQL 文識別子に対応付けられた被準備文を実行します。

SQL 文を動的に決定する場合、特にパラメタを指定する必要がないときには、EXECUTE IMMEDIATE 文を用いて同様の処理を行うことができます。

以下は、上の例と同様の処理を EXECUTE IMMEDIATE 文によって行う COBOL プログラムの例です。実行時に入力されるのは、上の例と同じ UPDATE 文です。

```
     *    :
         EXEC SQL BEGIN DECLARE SECTION END-EXEC.
      01  STMVAR  PIC X(254).
      01  SQLSTATE  PIC X(5).
         EXEC SQL END DECLARE SECTION END-EXEC.
      PROCEDURE DIVISION.
         ACCEPT STMVAR FROM CONSOLE.                    *>[1]
         EXEC SQL CONNECT TO DEFAULT END-EXEC.
         EXEC SQL EXECUTE IMMEDIATE :STMVAR END-EXEC.   *>[2]
     *    :
         EXEC SQL ROLLBACK WORK END-EXEC.
         EXEC SQL DISCONNECT DEFAULT END-EXEC.
         STOP RUN.
```

[プログラムの説明]

[1]: ACCEPT 文によって、UPDATE 文を読込み、SQL 文変数 STMVAR に設定します。

[2]: EXECUTE IMMEDIATE 文は、SQL 文変数に設定されている SQL 文を直接実行します。

動的パラメタ指定を記述する

以下は、動的パラメタ指定を記述し、STOCK 表のデータを参照する COBOL プログラムの例です。実行時には、以下の SELECT 文を入力します。

```
SELECT GNO, GOODS FROM STOCK WHERE WHNO = ?
```

```
*    :
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 製品番号          PIC S9(4) COMP-5.
 01 製品名            PIC X(20).
 01 在庫数量          PIC S9(9) COMP-5.
 01 倉庫番号          PIC S9(4) COMP-5.
 01 STMVAR            PIC X(254).
 01 SQLSTATE          PIC X(5).
 01 SQLMSG            PIC X(254).
    EXEC SQL END DECLARE SECTION END-EXEC.
 PROCEDURE DIVISION.
    EXEC SQL WHENEVER NOT FOUND GO TO :P-END END-EXEC.
    EXEC SQL
      DECLARE CUR11 CURSOR FOR STMIDT
    END-EXEC.                                        *>[1]
    ACCEPT STMVAR FROM CONSOLE.                        *>[2]
 P-START.
    EXEC SQL CONNECT TO DEFAULT END-EXEC.
    EXEC SQL PREPARE STMIDT FROM :STMVAR END-EXEC.       *>[3]
    ACCEPT 倉庫番号 FROM CONSOLE.                       *>[4]
    EXEC SQL OPEN CUR11 USING :倉庫番号
    END-EXEC.                                        *>[5]
  P-LOOP.
    EXEC SQL
      FETCH CUR11 INTO :製品番号,
                       :製品名                       *>[6]
    END-EXEC.
*   :
    GO TO P-LOOP.
 P-END.
    EXEC SQL CLOSE CUR11 END-EXEC.                     *>[7]
    EXEC SQL ROLLBACK WORK END-EXEC.
    EXEC SQL DISCONNECT DEFAULT END-EXEC.
    STOP RUN.
```

[プログラムの説明]

[1]: 動的カーソル宣言によって、CUR11 を定義します。

[2]: ACCEPT 文によって、動的 SELECT 文を読み込みます。

[3]: PREPARE 文によって、この時点で SQL 文変数 STMVAR に設定されていた文と、SQL 文識別子 STMIDT が対応付けられます。

[4]: 動的パラメタに対応付ける探索条件の値を読み込みます。

[5]: 動的 OPEN 文により、探索条件を満たす行が表から導出されます。このとき、USING 句で指定した値が、被準備文中の動的パラメタの値として参照されます。USING 句で指定した値と、被準備文中の動的パラメタは、その出現順序によって対応付けられます。

[6]: 動的 FETCH 文によって、表からデータを 1 行ずつ取り出し、各列の値を INTO 句で指定されたホスト変数に設定します。

[7]: 動的 CLOSE 文によって、指定されたカーソル名に対応する表を、カーソルとともに無効状態にします。

動的パラメタに複数列指定ホスト変数を使用するときは、その変数が使用できる SQL 文を被準備文として用意してください。使用できない SQL 文に指定した場合、動作は保証されません。

NCOBU00354

可変長文字列の使用

ここでは、可変長文字列データを COBOL プログラムのホスト変数で使用する方法について説明します。

可変長文字列データを操作する場合には、文字列の長さの情報が必要です。このため、可変長文字列型のホスト変数は、文字列の長さの情報を格納する符号付き 2 進項目と、文字列そのものを格納する英数字または日本語項目からなる集団項目として定義されています。

以下は、COMPANY 表から住所を参照する COBOL プログラムの例です。探索条件に用いるホスト変数および取り出したデータを格納するホスト変数は、どちらも可変長文字列データです。

```
*   :
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 会社名           PIC X(20).                          *>|
01 電話番号.                                            *>|
  49 電話番号の長さ         PIC S9(4) COMP-5.           *>|
  49 電話番号列             PIC X(20).                  *>|[1]
01 住所.                                                *>|
  49 住所の長さ             PIC S9(9) COMP-5.           *>|
  49 住所の文字列           PIC X(30).                  *>|
01 SQLSTATE                  PIC X(5).                  *>|
    EXEC SQL END DECLARE SECTION END-EXEC.
PROCEDURE DIVISION.
    DISPLAY "電話番号から会社を検索します."
    DISPLAY "電話番号を入力してください→ " WITH NO ADVANCING.
    ACCEPT 電話番号列  FROM CONSOLE.                    *>[2]
    INSPECT 電話番号列 TALLYING 電話番号の長さ          *>[3]
      FOR CHARACTERS BEFORE SPACE
    EXEC SQL CONNECT TO DEFAULT END-EXEC.
    EXEC SQL
      SELECT NAME,ADDRESS INTO :会社名,
                          :住所                     *>[4]
            FROM COMPANY
            WHERE PHONE = :電話番号
    END-EXEC.
*    :
    EXEC SQL ROLLBACK WORK END-EXEC.
    EXEC SQL DISCONNECT DEFAULT END-EXEC.
    STOP RUN.
```

[プログラムの説明]

[1]: 可変長文字列のホスト変数を宣言します。

[2]: ACCEPT 文によって、探索条件に用いるホスト変数の値を"電話番号列"に読み込みます。

[3]: 読み込んだ値の長さを、"電話番号の長さ"に設定します。

[4]: 単一行 SELECT 文によって、探索条件を満たす行の ADDRESS 列の値を取り出します。ホスト変数"住所"には、取り出した列の長さと値が設定されます。

探索条件に用いるホスト変数のデータ型が文字列型または各国語文字列型の場合は、そのホスト変数の長さを列の長さ以下に定義してください。文字列型および各国語文字列型については、COBOL 文法書の「8.3.1 文字」を参照してください。

複数コネクションでのカーソル操作

ここでは、複数のコネクションを接続してカーソルを操作する方法について説明します。

以下の例では、サーバとして SV1 および SV2 があり、さらにそれぞれのサーバ上に、同じ表名と形式を持つ表が存在しています。

```
*   :
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 製品番号          PIC S9(4) COMP-5.
01 製品名            PIC X(20).
01 在庫数量          PIC S9(9) COMP-5.
01 倉庫番号          PIC S9(4) COMP-5.
01 SQLSTATE          PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
01 NEXTFLAG          PIC X(4) VALUE SPACE.
PROCEDURE DIVISION.
    EXEC SQL DECLARE CUR9 CURSOR FOR
      SELECT * FROM STOCK
    END-EXEC.                                          *>[1]
    EXEC SQL  WHENEVER NOT FOUND
      GO TO :P-NEXT
    END-EXEC.                                          *>[2]
P-START.
    EXEC SQL
      CONNECT TO 'SV1' AS 'CNN1' USER 'summer/w43'
    END-EXEC.                                          *>[3]
    EXEC SQL
      CONNECT TO 'SV2' AS 'CNN2' USER 'tanaka/sky'
    END-EXEC.                                          *>[4]
P-CNN2-1.
    EXEC SQL OPEN CUR9 END-EXEC.                       *>[5]
    GO TO P-LOOP.
P-NEXT.
    IF NEXTFLAG = "NEXT" THEN
      GO TO P-CNN1-2
    END-IF.
P-CNN1-1.
    EXEC SQL SET CONNECTION 'CNN1' END-EXEC.             *>[7]
    EXEC SQL
      INSERT INTO STOCK
      VALUES(:製品番号, :製品名,
             :在庫数量, :倉庫番号)
    END-EXEC.                                          *>[8]
    EXEC SQL OPEN CUR9 END-EXEC.                       *>[9]
    MOVE "NEXT" TO NEXTFLAG
    GO TO P-LOOP.
P-CNN1-2.
    EXEC SQL CLOSE CUR9 END-EXEC.                      *>[10]
    EXEC SQL ROLLBACK WORK END-EXEC.
    EXEC SQL DISCONNECT 'CNN1' END-EXEC.
P-CNN2-2.
    EXEC SQL SET CONNECTION 'CNN2' END-EXEC.            *>[11]
    EXEC SQL CLOSE CUR9 END-EXEC.                      *>[12]
    EXEC SQL ROLLBACK WORK END-EXEC.
    EXEC SQL DISCONNECT CURRENT END-EXEC.
P-END.
    STOP RUN.
P-LOOP.
    EXEC SQL
      FETCH CUR9
      INTO :製品番号, :製品名, :在庫数量, :倉庫番号
    END-EXEC.                                        *>[6]
*   :
    GO TO P-LOOP.
```

[プログラムの説明]

[1]: STOCK 表のデータを参照するカーソルを定義します。

[2]: 埋込み例外宣言によって、取り出す行がない場合は、手続き名"P-NEXT"の位置へ分岐することを指定します。

[3]: サーバ"SV1"とコネクションを接続し、この接続を"CNN1"とします。

[4]: サーバ"SV2"とコネクションを接続し、この接続を"CNN2"とします。これ以降、コネクションが変更されるまで、"CNN2"が現在のコネクションになります。

[5]: コネクション"CNN2"上で OPEN 文を実行し、カーソル"CUR9"を使用可能な状態にします。

[6]: FETCH 文によって、表からデータを 1 行ずつ取り出し、各列の値を対応するホスト変数の領域に設定します。

[7]: 現在のコネクションを"CNN1"に変更します。

[8]: コネクション"CNN1"上の STOCK 表に、INSERT 文によってデータを 1 行挿入します。

[9]: コネクション"CNN1"上で OPEN 文を実行し、カーソル"CUR9"を使用可能な状態にします。このときのカーソル"CUR9"は、コネクション"CNN2"上で OPEN したカーソルとは別のカーソルとして扱われます。

[10]: コネクション"CNN1"上で CLOSE 文を実行し、カーソル"CUR9"を無効な状態にします。

[11]: 現在のコネクションを"CNN2"に変更します。コネクション"CNN2"上では、カーソル"CUR9"はまだ有効な状態であることに注意してください。

[12]: コネクション"CNN2"上で CLOSE 文を実行し、カーソル"CUR9"を無効な状態にします。

ストアドプロシージャの呼出し

ここでは、ストアドプロシージャの呼出しについて説明します。

このセクションの内容

NCOBU00364

ストアドプロシージャとは

ストアドプロシージャとは、サーバに登録された処理手続きのことです。ストアドプロシージャを、クライアント側から呼び出し、サーバ側で処理手続きを実行します。ストアドプロシージャを使用することによる利点は、各データベースによって異なりますが、共通して以下のような利点があります。

- 処理手続きの実行速度の向上
- クライアントーサーバ間の通信負荷の減少
- 開発/保守の生産性の向上
- セキュリティの向上

ストアドプロシージャの詳細、作成方法については、各データベース管理システムのマニュアルを参照してください。

ストアドプロシージャの呼び出しには時間がかかる場合があります。そのため、ストアドプロシージャ内の処理が軽い場合、パフォーマンスを発揮できない場合があります。その場合はストアドプロシージャを使用せずにプログラミングしてください。

ストアドプロシージャの呼出し例

ストアドプロシージャを呼び出す COBOL プログラムの例を以下に示します。ここでは、ストアドプロシージャ PROC を呼び出しています。SQLERRD(1)により、ストアドプロシージャの戻り値を取得することができます。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01  入力変数  PIC S9(4) COMP-5.                  *>[1]
    01  出力変数  PIC S9(4) COMP-5.
    01  SQLSTATE  PIC X(5).
    01  SQLINFOA.
      02 SQLERRD  PIC S9(9) COMP-5 OCCURS 6 TIMES.
       EXEC SQL END DECLARE SECTION END-EXEC.
    PROCEDURE DIVISION.
       EXEC SQL CONNECT TO DEFAULT END-EXEC        *>[2]
       MOVE 100 TO 入力変数                        *>[3]
       EXEC SQL
         CALL PROC (:入力変数, :出力変数)
       END-EXEC                                    *>[4]
       DISPLAY SQLERRD(1).                         *>[5]
       EXEC SQL ROLLBACK WORK END-EXEC             *>[6]
       EXEC SQL DISCONNECT DEFAULT END-EXEC.       *>[7]
       STOP RUN.
```

ストアドプロシージャを呼び出すプログラム例

[プログラムの説明]

[1]: 作業場所節に埋込み SQL 宣言節を記述し、呼び出すストアドプロシージャの引数をすべてホスト変数として定義します。ホスト変数宣言の規則については、COBOL 文法書の「8.2.2 ホスト変数定義」を参照してください。

[2]: CONNECT 文を実行し、サーバとのコネクションを接続します。

[3]: ストアドプロシージャの入力パラメタをホスト変数に設定します。

[4]: CALL 文を実行し、サーバに登録された PROC ストアドプロシージャを呼び出します。呼び出し後、ストアドプロシージャからの出力パラメタがホスト変数に設定されます。

[5]: ストアドプロシージャの戻り値を返します。

[6]: ROLLBACK 文を実行し、トランザクションを終了します。

[7]: DISCONNECT 文を実行し、サーバとのコネクションを切断します。

- ストアドプロシージャが戻り値を返すことが可能かどうかは、データベースによって異なります。使用するデータベースのマニュアルを確認してください。

NCOBU00356

高度なデータ操作

ここでは、高度なデータ操作を可能にするホスト変数について説明します。

説明中の COBOL プログラムの例は、サンプルデータベースの STOCK 表(在庫表)を使用しています。

このセクションの内容

高度なデータ操作を可能とするホスト変数

データベース表の複数の行を一度に操作できる、複数行指定ホスト変数および表指定ホスト変数について説明します。

動的 SQL 文で使用する方法

動的 SQL 文で複数行指定ホスト変数および表指定ホスト変数を使用する場合の注意事項について説明します。

SQLERRD による処理行数の確認方法

SQLERRD を参照して、取り出した行の件数や処理された件数を確認する方法について説明します。

カーソルのタイプによるオプション指定

カーソルのタイプによる SQL オプション情報の指定について説明します。

FOR 句による処理行数制御

FOR 句を指定して、処理行数または処理回数を制御する方法について説明します。

スクロール可能なカーソルを使用したデータの取得

FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文を使用してデータを取得する方法について説明します。

高度なデータ操作を可能とするホスト変数

ここまでに説明してきた埋込み SQL 文によるデータ操作では、基本的に、埋込み SQL 文の 1 回の実行でデータベース表の 1 行を操作しました。しかし、以下のホスト変数を指定した埋込み SQL 文を使用すると、データベース表の複数の行を一度に操作できます。

- 複数行指定ホスト変数
- 表指定ホスト変数

これらを使用することによって、応用プログラムの性能を向上させることができます。

NCOBU00358

複数行指定ホスト変数

機能

表の 1 つの列に対し、複数の行データを操作することができます。

定義

OCCURS 句を持つ繰り返し項目として定義した基本項目です。複数行指定ホスト変数の詳細については、COBOL 文法書の「8.2.2 ホスト変数定義」を参照してください。

使用方法

データの参照

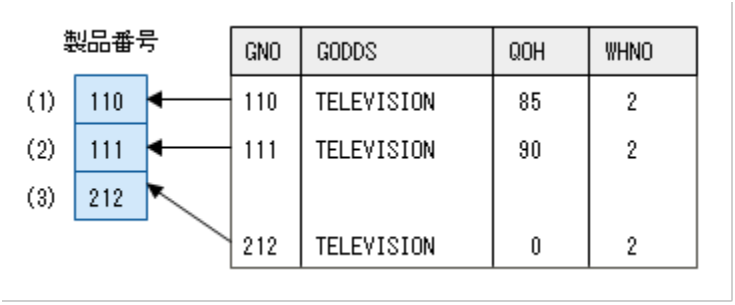
複数行指定ホスト変数を相手指定に使用すると、複数行指定ホスト変数の繰り返し回数と同じ件数のデータを参照することができます。

"STOCK 表" から、製品名が "TELEVISION" である行の製品番号(GNO 列)を 3 件取り出す COBOL プログラムの例を以下に示します。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号   PIC S9(4) COMP-5 OCCURS 3 TIMES.     *>[1]
    01 SQLSTATE    PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *     :
    PROCEDURE DIVISION.
    *     :
       EXEC SQL
         SELECT GNO FROM STOCK
         INTO :製品番号
         WHERE GOODS = 'TELEVISION'
       END-EXEC                                          *>[2]
    *    :
```

[1]: OCCURS 句の繰り返し回数を宣言することにより、複数行指定ホスト変数"製品番号"を定義します。

[2]: SELECT 文で"製品番号"にデータを取り出すと、製品番号(1)には 110、製品番号(2)には 111、製品番号(3)には 212 が格納されます。

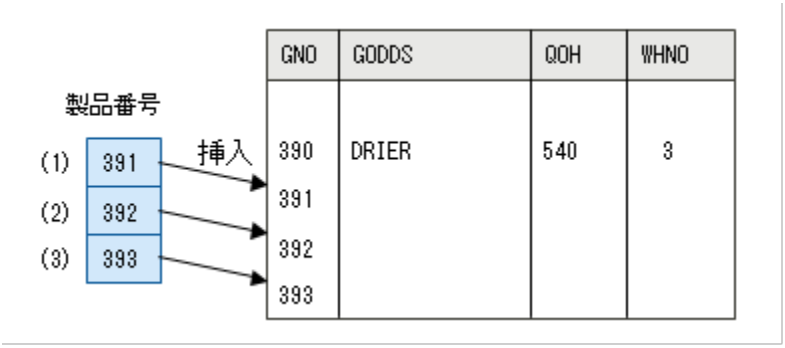


データの挿入

複数行指定ホスト変数を値指定に使用すると、複数行指定ホスト変数の繰り返し回数と同じ件数のデータを挿入することができます。

"STOCK 表" の製品番号(GNO 列)に 3 件のデータを挿入する COBOL プログラムの例を以下に示します。

```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 在庫表.
   02 製品番号 PIC S9(4) COMP-5 OCCURS 3 TIMES.
 01 SQLSTATE   PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
*    :
 PROCEDURE DIVISION.
*    :
    MOVE 391 TO 製品番号 (1)
    MOVE 392 TO 製品番号 (2)
    MOVE 393 TO 製品番号 (3)
    EXEC SQL
      INSERT INTO STOCK(GNO)
      VALUES (:製品番号)
    END-EXEC
*    :
```



複数行指定ホスト変数の繰り返し回数と同じ数だけ INSERT 文が実行され、複数行指定ホスト変数に格納されている先頭の値から順番に、表に挿入されます。つまり、以下のように記述した場合と同じ結果になります。

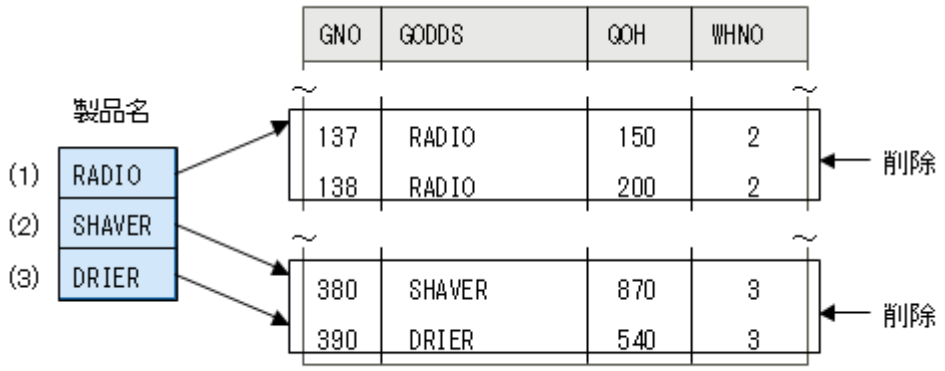
```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 製品番号  PIC S9(4) COMP-5.
 01 SQLSTATE  PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
*    :
 PROCEDURE DIVISION.
*    :
    MOVE 391 TO 製品番号
    EXEC SQL
      INSERT INTO STOCK(GNO) VALUES(:製品番号)
    END-EXEC
    MOVE 392 TO 製品番号
    EXEC SQL
      INSERT INTO STOCK(GNO) VALUES(:製品番号)
    END-EXEC
    MOVE 393 TO 製品番号
    EXEC SQL
      INSERT INTO STOCK(GNO) VALUES(:製品番号)
    END-EXEC
*    :
```

データの削除

複数行指定ホスト変数を問合わせ指定に使用すると、複数の条件に合致するデータを一度に削除できます。

"STOCK 表" から、3 つの条件に従ってデータの削除を行う COBOL プログラムの例を以下に示します。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品名  PIC X(20) OCCURS 3 TIMES.
    01 SQLSTATE  PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
   *   :
    PROCEDURE DIVISION.
   *   :
       MOVE "RADIO"  TO 製品名 (1)
       MOVE "SHAVER" TO 製品名 (2)
       MOVE "DRIER"  TO 製品名 (3)
       EXEC SQL
         DELETE FROM STOCK WHERE GOODS = :製品名
       END-EXEC
   *   :
```



これは次のように記述した場合と同じ結果になります。

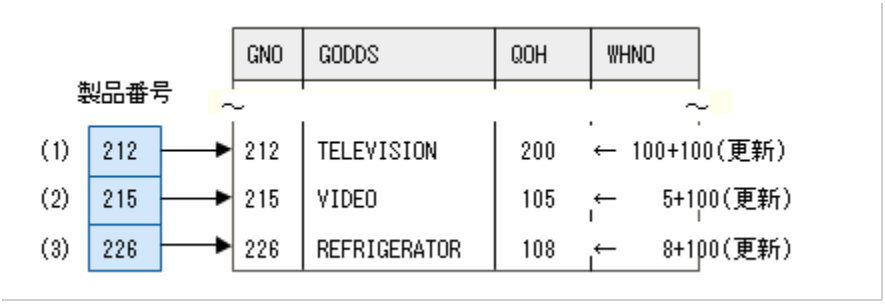
```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 製品名    PIC X(20).
    01 SQLSTATE  PIC X(5).
       EXEC SQL END  DECLARE SECTION END-EXEC.
   *   :
    PROCEDURE DIVISION.
   *   :
       MOVE "RADIO"  TO 製品名
       EXEC SQL
         DELETE FROM STOCK WHERE GOODS = :製品名
       END-EXEC
       MOVE "SHAVER" TO 製品名
       EXEC SQL
         DELETE FROM STOCK WHERE GOODS = :製品名
       END-EXEC
       MOVE "DRIER"  TO 製品名
       EXEC SQL
         DELETE FROM STOCK WHERE GOODS = :製品名
       END-EXEC
   *   :
```

データの更新

複数行指定ホスト変数を問合わせ指定に使用すると、複数の条件に合致するデータを一度に更新することができます。

条件に合致する列のデータに、一定の処理を加えながら更新する COBOL プログラムの例を以下に示します。

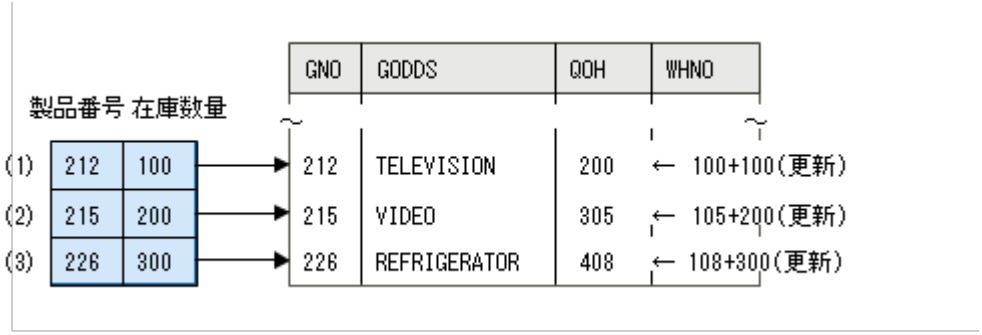
```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号  PIC S9(4) COMP-5 OCCURS 3 TIMES.
    01 SQLSTATE    PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *    :
    PROCEDURE DIVISION.
    *    :
       MOVE 212 TO 製品番号 (1)
       MOVE 215 TO 製品番号 (2)
       MOVE 226 TO 製品番号 (3)
       EXEC SQL
         UPDATE STOCK SET QOH = QOH + 100
           WHERE GNO = :製品番号
       END-EXEC
    *    :
```



また、複数行指定ホスト変数を問合わせ指定と設定句(SET)の値指定に同時に使用して、複数の条件に合致するデータを順番に更新することができます。

以下の例では、設定句および問合わせ指定に複数行指定ホスト変数を使用しています。 これらの変数に格納された値は、それぞれの対応する列(GNO 列および QOH 列)に対して、配列の先頭から順に使用されます。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号  PIC S9(4) COMP-5 OCCURS 3 TIMES.
      02 在庫数量  PIC S9(9) COMP-5 OCCURS 3 TIMES.
    01 SQLSTATE    PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *    :
    PROCEDURE DIVISION.
    *    :
       MOVE 212 TO 製品番号 (1)
       MOVE 215 TO 製品番号 (2)
       MOVE 226 TO 製品番号 (3)
       MOVE 100 TO 在庫数量 (1)
       MOVE 200 TO 在庫数量 (2)
       MOVE 300 TO 在庫数量 (3)
       EXEC SQL
         UPDATE STOCK SET QOH = QOH + :在庫数量
           WHERE GNO = :製品番号
       END-EXEC
    *    :
```



注意事項

- 複数行指定できないドライバもあります。各ドライバの仕様を確認してください。
- 1 つの SQL 文に対して複数行指定ホスト変数を複数使用する場合、複数行指定ホスト変数の繰り返し回数は同じ数にしてください。 同じ数でない場合、FOR 句を使用してください。FOR 句については、FOR 句による処理行数制御を参照してください。
- 複数行指定ホスト変数の繰り返し回数が異なり、かつ FOR 句の指定もない場合は、複数行指定ホスト変数に指定された最小の繰り返し回数が有効になります。
- 1 つの SQL 文で使用される複数行指定ホスト変数の繰り返し回数が、それぞれ異なる場合の COBOL プログラムの例を以下に示します。 以下の例では、繰り返し回数 8 と繰り返し回数 5 を持つ複数行指定ホスト変数が定義されていますが、この場合、最小値である 5 が UPDATE 文に対して有効になります。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号     PIC S9(4) COMP-5 OCCURS 8 TIMES.
      02 在庫数量     PIC S9(9) COMP-5 OCCURS 5 TIMES.
    01 SQLSTATE       PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *    :
     PROCEDURE DIVISION.
    *   :
       EXEC SQL
         UPDATE STOCK SET QOH = QOH + :在庫数量
           WHERE GNO = :製品番号
       END-EXEC
    *    :
```

表指定ホスト変数

機能

複数行および複数列にまたがるデータを一度に操作できます。

定義

データベース表の各列に対応する複数行ホスト変数を従属する集団項目として定義します。詳細については、COBOL 文法書の「8.2.2 ホスト変数定義」を参照してください。

使用方法

データの参照

表指定ホスト変数を使用すると、表指定ホスト変数の従属項目である複数行指定ホスト変数の繰り返し回数と同じ件数のデータを参照することができます。

"STOCK 表" から、製品名が "TELEVISION" である行のデータ(製品番号、製品名、在庫数量、倉庫番号)を 3 件取り出す COBOL プログラムの例を以下に示します。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.                                        *>|
      02 製品番号    PIC S9(4) COMP-5 OCCURS 3 TIMES.     *>|
      02 製品名      PIC X(20) OCCURS 3 TIMES.           *>|[1]
      02 在庫数量    PIC S9(9) COMP-5 OCCURS 3 TIMES.     *>|
      02 倉庫番号    PIC S9(4) COMP-5 OCCURS 3 TIMES.     *>|
    01 SQLSTATE             PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *    :
     PROCEDURE DIVISION.
    *    :
         EXEC SQL
           SELECT  *
           INTO :在庫表 FROM STOCK
           WHERE GOODS = 'TELEVISION'
         END-EXEC.                                     *>[2]
    *    :
```

[1]: 表指定ホスト変数"在庫表"を定義します。

[2]: SELECT 文でデータを取り出すと、ホスト変数"在庫表"に以下の値が格納されます。

在庫表

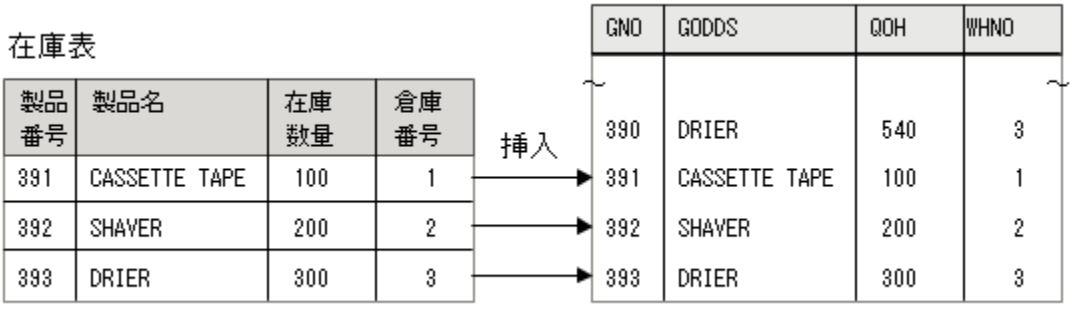
製品番号	製品名	在庫数量	倉庫番号
110	TELEVISION	85	2
111	TELEVISION	90	2
212	TELEVISION	0	2

データの挿入

表指定ホスト変数を使用すると、表指定ホスト変数の従属項目である複数行指定ホスト変数の繰り返し回数と同じ件数のデータを挿入することができます。

"STOCK 表" に、3 件のデータを挿入する COBOL のプログラムの例を以下に示します。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号    PIC S9(4) COMP-5 OCCURS 3 TIMES.
      02 製品名      PIC X(20) OCCURS 3 TIMES.
      02 在庫数量    PIC S9(9) COMP-5 OCCURS 3 TIMES.
      02 倉庫番号    PIC S9(4) COMP-5 OCCURS 3 TIMES.
    01 SQLSTATE           PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *   :
    PROCEDURE DIVISION.
    *   :
       MOVE 391 TO 製品番号(1)
       MOVE 392 TO 製品番号(2)
       MOVE 393 TO 製品番号(3)
       MOVE "CASSETTE TAPE" TO 製品名(1)
       MOVE "SHAVER"        TO 製品名(2)
       MOVE "DRIER"         TO 製品名(3)
       MOVE 100 TO 在庫数量(1)
       MOVE 200 TO 在庫数量(2)
       MOVE 300 TO 在庫数量(3)
       MOVE 1   TO 倉庫番号(1)
       MOVE 2   TO 倉庫番号(2)
       MOVE 3   TO 倉庫番号(3)
       EXEC SQL
         INSERT INTO STOCK(GNO, GOODS, QOH, WHNO)
           VALUES(:在庫表)
       END-EXEC
    *   :
```



注意事項

- 複数行指定できないドライバもあります。各ドライバの仕様を確認してください。

動的 SQL 文で使用する方法

複数行指定ホスト変数および表指定ホスト変数は、従来のホスト変数と同じように動的 SQL 文で使用できます。ただし、これらのホスト変数が使用できる SQL 文は限られています。 使用できない SQL 文に指定した場合、動作は保証されません。使用できる SQL 文については、埋込み SQL 文で使用可能なホスト変数を参照してください。

NCOBU00361

SQLERRD による処理行数の確認方法

複数行指定ホスト変数または表指定ホスト変数を使用したデータ操作では、SQLERRD(3)を参照することによって、取り出した行の件数や処理された行の件数を知ることができます。

使用例

- SELECT 文または FETCH 文で複数行データを取り出す場合、SQLERRD(3)には取り出したデータの件数が格納されます。たとえば、繰り返し回数 100 を持つ複数行指定ホスト変数を使用し、取り出す対象のデータが 50 件である場合、SQLERRD(3)には 50 が格納されます。
- INSERT 文、UPDATE 文(検索)または DELETE 文(検索)で複数行データを操作する場合、SQL 文のデータ操作によって処理された行の件数が SQLERRD(3)に格納されます。

"STOCK 表"を、WHNO の条件によって QOH を更新する動作を 2 回行う COBOL プログラムの例を以下に示します。WHNO=1 に対する QOH の更新が 6 件、WHNO=2 に対する QOH の更新が 11 件、合わせて 17 件であるという値が SQLERRD(3)に格納されます。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 在庫数量   PIC S9(9) COMP-5 OCCURS 2 TIMES.
      02 倉庫番号   PIC S9(4) COMP-5 OCCURS 2 TIMES.
    01 SQLINFOA.
      02 SQLERRD  PIC S9(9) COMP-5 OCCURS 6 TIMES.
    01 SQLSTATE   PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
   *    :
    PROCEDURE DIVISION.
   *    :
       MOVE 100 TO 在庫数量(1)
       MOVE 200 TO 在庫数量(2)
       MOVE 1   TO 倉庫番号(1)
       MOVE 2   TO 倉庫番号(2)
       EXEC SQL
         UPDATE STOCK
         SET QOH = QOH + :在庫数量
         WHERE WHNO = :倉庫番号
       END-EXEC
       DISPLAY SQLERRD(3)
   *    :
```

- 取り出し対象のデータ件数が繰り返し回数より多くても少なくても、SQLSTATE の値は"データなし"にはなりません。SQLSTATE の値が"データなし"になるのは、1 つも行が取り出せなかった場合です。
- CALL 文で呼び出されたストアドプロシージャでの変更行は、SQLERRD(3)では保証されません。

カーソルのタイプによるオプション指定

ODBC ドライバ接続では、FOR UPDATE 句の指定がないカーソルと、FOR UPDATE 句の指定があるカーソル毎に、以下の SQL オプション情報を指定できます。FOR UPDATE 句の指定がないカーソルの SQL オプション情報は、@SQL_ODBC_CURSOR_DEFAULT に指定したオプション名に関連付けられ、FOR UPDATE 句の指定があるカーソルの SQL オプション情報は、@SQL_ODBC_CURSOR_UPDATE に指定したオプション名に関連付けられます。

- @SQL_CURSOR_TYPE (カーソル種別の指定)
- @SQL_ROW_ARRAY_SIZE(メモリにキャッシュする行数の指定)
- @SQL_CONCURRENCY (カーソルの同時実行の指定)

これらの SQL オプション情報を指定することで、アプリケーション内でのカーソルの使用目的に応じた、適切なカーソル属性にすることができます。

利用サンプルを以下に示します。

ソースファイル

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号   PIC S9(4) COMP-5 OCCURS 10 TIMES.
      02 製品名     PIC X(20) OCCURS 10 TIMES.
      02 在庫数量   PIC S9(9) COMP-5 OCCURS 10 TIMES.
      02 倉庫番号   PIC S9(4) COMP-5 OCCURS 10 TIMES.
    01 SQLSTATE    PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
   *    :
    PROCEDURE DIVISION.
   *    :
       EXEC SQL
         DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK FOR UPDATE   ···[1]
       END-EXEC.
   *    :
   *    :
   *    :
       EXEC SQL
         DECLARE CUR2 CURSOR FOR SELECT * FROM STOCK       ·········[2]
       END-EXEC.
   *    :
```

アプリケーション構成ファイル

```
          :
<serverList>
  <server name="DEFAULT" description="···">
          :
    <add key = "@SQL_ODBC_CURSOR_UPDATE" value="update_cursor" />   ··[A]
    <add key = "@SQL_ODBC_CURSOR_DEFAULT" value="default_cursor"/>  ··[B]
  </server>
</serverList>
<sqlOptionInf>
   <option name="update_cursor">                 ·····················[C]
     <add key = "@SQL_ROW_ARRAY_SIZE" value="10" />
     <add key = "@SQL_CURSOR_TYPE"  value="KEYSET_DRIVEN" />
     <add key = "@SQL_CONCURRENCY"  value="READ_WRITE" />
   </option>
   <option name="default_cursor">                ·····················[D]
     <add key = "@SQL_ROW_ARRAY_SIZE" value="5" />
     <add key = "@SQL_CURSOR_TYPE"  value="FORWARD_ONLY" />
     <add key = "@SQL_CONCURRENCY"  value="READ_ONLY" />
   </option>
</sqlOptionInf>
          :
```

- FOR UPDATE 句の指定があるカーソル[1]は、@SQL_ODBC_CURSOR_UPDATE 指定[A]によって指定された SQL オプション情報[C]を指定できます。
- FOR UPDATE 句の指定がないカーソル[2]は、@SQL_ODBC_CURSOR_DEFAULT 指定[B]によって SQL オプション情報[D]を指定できます。

- 指定できるカーソルオプションは、SQL オプション情報を参照してください。
- データベースによっては、指定できるカーソルタイプをサポートしていない場合があります。各データベースまたは、使用するデータベースドライバを確認してください。

FOR 句による処理行数制御

複数行指定ホスト変数または表指定ホスト変数を使用したデータ操作では、FOR 句を指定することで、処理行数または処理回数を制御することができます。

使用方法

データ参照

"STOCK 表" の先頭から 5 件分のデータを参照する COBOL プログラムの例を以下に示します。

繰り返し回数が 10 である複数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、FETCH 文でデータを取り出します。 この場合、複数行指定ホスト変数の繰り返し回数は 10 であっても、FETCH 文の FOR 句に 5 が指定されているため、5 件分のデータが表指定ホスト変数に格納されます。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.
      02 製品番号   PIC S9(4) COMP-5 OCCURS 10 TIMES.
      02 製品名     PIC X(20) OCCURS 10 TIMES.
      02 在庫数量   PIC S9(9) COMP-5 OCCURS 10 TIMES.
      02 倉庫番号   PIC S9(4) COMP-5 OCCURS 10 TIMES.
    01 SQLSTATE     PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *    :
    PROCEDURE DIVISION.
    *    :
       EXEC SQL
         DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
       END-EXEC.
    *    :
       EXEC SQL
         OPEN CUR1
       END-EXEC.
    *    :
       EXEC SQL
         FOR 5
         FETCH CUR1 INTO :在庫表
       END-EXEC.
    *    :
```

また、データの参照の"表の全行を参照する"で示した表の全体を参照するプログラムの例は、次のように書くことができます。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.                                          *>|
      02 製品番号   PIC S9(4) COMP-5 OCCURS 20 TIMES.    *>|
      02 製品名     PIC X(20) OCCURS 20 TIMES.           *>|[1]
      02 在庫数量   PIC S9(9) COMP-5 OCCURS 20 TIMES.    *>|
      02 倉庫番号   PIC S9(4) COMP-5 OCCURS 20 TIMES.    *>|
    01 行数        PIC S9(9) COMP-5.
    01 SQLSTATE     PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *    :
    PROCEDURE DIVISION.
    *    :
       EXEC SQL
         DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
       END-EXEC
    *    :
       EXEC SQL
         SELECT COUNT(*) INTO :行数 FROM STOCK          *>[2]
       END-EXEC
    *    :
       EXEC SQL OPEN CUR1 END-EXEC
    *    :
```

```
        EXEC SQL
          FOR :行数
          FETCH CUR1 INTO :在庫表
        END-EXEC                                        *>[3]
    *   :
```

[1]: 表指定ホスト変数"在庫数"を定義します。

[2]: 集合関数 COUNT(*)で表の行数を求めます。

[3]: 表のすべての行のデータを"在庫表"に取り出します。

データ挿入

"STOCK 表"に 3 件の行データを挿入する COBOL プログラムの例を以下に示します。

繰り返し回数が 10 の複数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、INSERT 文でデータを挿入します。 この場合、複数行指定ホスト変数の繰り返し回数が 10 であっても、INSERT 文の FOR 句に 3 が指定されているため、STOCK 表には 3 件のデータが挿入されます。

```
        EXEC SQL BEGIN DECLARE SECTION END-EXEC.
     01 在庫表.
       02 製品番号   PIC S9(4) COMP-5 OCCURS 10 TIMES.
       02 製品名     PIC X(20) OCCURS 10 TIMES.
       02 在庫数量   PIC S9(9) COMP-5 OCCURS 10 TIMES.
       02 倉庫番号   PIC S9(4) COMP-5 OCCURS 10 TIMES.
     01 SQLSTATE     PIC X(5).
        EXEC SQL END DECLARE SECTION END-EXEC.
    *   :
     PROCEDURE DIVISION.
    *   :
        MOVE 391 TO 製品番号(1)
        MOVE 392 TO 製品番号(2)
        MOVE 393 TO 製品番号(3)
        MOVE "CASSETTE TAPE" TO 製品名(1)
        MOVE "SHAVER"        TO 製品名(2)
        MOVE "DRIER"         TO 製品名(3)
        MOVE 100 TO 在庫数量(1)
        MOVE 200 TO 在庫数量(2)
        MOVE 300 TO 在庫数量(3)
        MOVE 1   TO 倉庫番号(1)
        MOVE 2   TO 倉庫番号(2)
        MOVE 3   TO 倉庫番号(3)
        EXEC SQL
          FOR 3
          INSERT INTO STOCK(GNO, GOODS, QOH, WHNO)
          VALUES(:在庫表)
        END-EXEC
    *   :
```

FOR 句に指定する値は、複数行指定ホスト変数の繰り返し回数と同じか、それより小さい値でなければなりません。繰り返し回数より大きい値を指定した場合、エラーが出力されます。

スクロール可能なカーソルを使用したデータの取得

スクロール可能なカーソルとは、カーソルの OPEN 文で作成された結果を、順不同にアクセスすることができるカーソルのことを言います。 NetCOBOL for .NET では、以下の文が該当します。

スクロール可能なカーソルを使用する FETCH 文

FETCH 文	動作説明
FETCH PRIOR 文	最後に FETCH された行の前のデータを取り出します。
FETCH FIRST 文	カーソルを先頭行に位置付け、その行から後ろのデータを取り出します。
FETCH LAST 文	カーソルを最終行に位置付け、その行から前のデータを取り出します。

ここでは、スクロール可能なカーソルを使用してデータを取得する方法について説明します。

使用方法

スクロール可能なカーソルを使用するために必要なオプションの指定

デフォルトのカーソルの設定ではカーソルは順方向専用で作成されるため、 スクロール可能なカーソルを使用することができません。 スクロール可能なカーソルを使用するためには以下のオプションを設定してください。 また、各オプションの詳細についてはオプションの説明を参照してください。

- ODBC 経由の接続の場合

スクロール可能なカーソルを使用するため必要なオプションの設定値

オプション	設定値
@SQL_CURSOR_TYPE	KEYSET_DRIVEN または STATIC または DYNAMIC
@SQL_ROW_ARRAY_SIZE	1 または オプションを指定しない

@SQL_CURSOR_TYPE オプションには FORWARD_ONLY 以外を指定してください。FORWARD_ONLY を指定した場合または@SQL_CURSOR_TYPE オプションを指定しない場合は、順方向専用カーソルとなり、スクロール可能なカーソルを実行することができません。

@SQL_ROW_ARRAY_SIZE オプションに 2 以上の値が設定されている場合、スクロール可能なカーソルを実行することはできません。@SQL_ROW_ARRAY_SIZE オプションに 1 を設定するかオプションを指定しないでください。

- ADO.NET 経由の接続の場合

スクロール可能なカーソルを使用するため必要なオプションの設定値

オプション	設定値
@SQL_ADONET_CURSOR_DEFAULT	READ_WRITE

カーソル宣言に FOR UPDATE 句が指定されていないカーソルを使用する場合、@SQL_ADONET_CURSOR_DEFAULT オプションに READ_WRITE を指定してください。READ_ONLY 指定で作成されたカーソルは順方向専用カーソルとなり、スクロール可能なカーソルを実行することはできません。

FETCH PRIOR 文によるデータ取得

"STOCK 表" の先頭から 3 件のデータをそれぞれ FETCH NEXT 文で取り出し、FETCH PRIOR 文で"STOCK 表"の先頭まで戻る COBOL プログラムの例を以下に示します。

単数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、FETCH NEXT 文で 3 件のデータを取り出し、次に FETCH PRIOR 文で 2 件のデータを取り出します。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
    01 在庫表.                              *>|
      02 製品番号   PIC S9(4) COMP-5.       *>|
      02 製品名     PIC X(20).              *>|[1]
      02 在庫数量   PIC S9(9).              *>|
      02 倉庫番号   PIC S9(4).              *>|
    01 SQLSTATE    PIC X(5).
       EXEC SQL END DECLARE SECTION END-EXEC.
    *   :
    PROCEDURE DIVISION.
    *   :
       EXEC SQL
         DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
       END-EXEC.
    *   :
       EXEC SQL
         OPEN CUR1                       *>[2]
       END-EXEC.
    *   :
       EXEC SQL
         FETCH NEXT CUR1 INTO :在庫表      *>[3]
       END-EXEC.
    *   :
       EXEC SQL
         FETCH NEXT CUR1 INTO :在庫表      *>[4]
       END-EXEC.
    *   :
       EXEC SQL
         FETCH NEXT CUR1 INTO :在庫表      *>[5]
       END-EXEC.
    *   :
       EXEC SQL
         FETCH PRIOR CUR1 INTO :在庫表     *>[6]
       END-EXEC.
    *   :
       EXEC SQL
         FETCH PRIOR CUR1 INTO :在庫表     *>[7]
       END-EXEC.
    *   :
```

[1]: 表指定ホスト変数"在庫表"を定義します。

[2]: カーソル"CUR1"をオープンします。

[3]: 在庫表の GNO が 110 のデータを"在庫表"に取り出します。

在庫表

製品番号	製品名	在庫数量	倉庫番号
110	TELEVISION	85	2

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
	:		

[4]: 在庫表の GNO が 111 のデータを"在庫表"に取り出します。

在庫表

製品番号	製品名	在庫数量	倉庫番号
111	TELEVISION	90	2

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
	:		

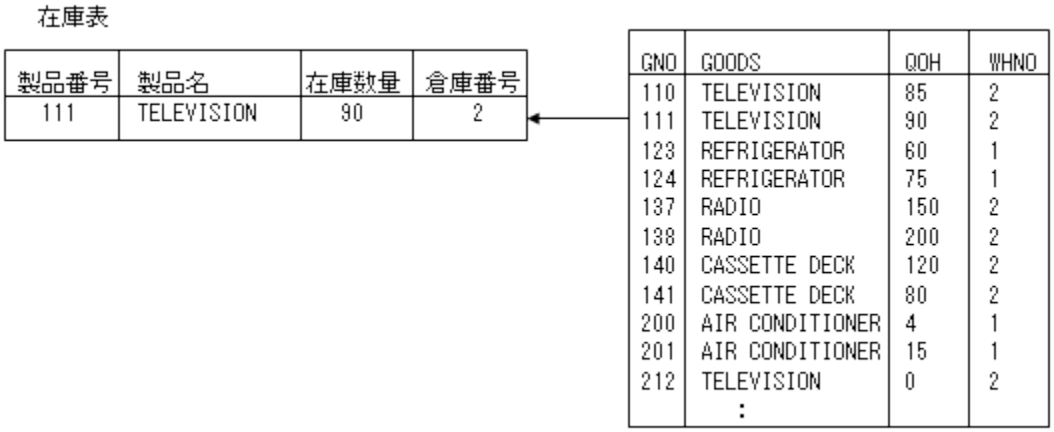
[5]: 在庫表の GNO が 123 のデータを"在庫表"に取り出します。

GNO	GOODS	QOH	WHNO
	:		
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

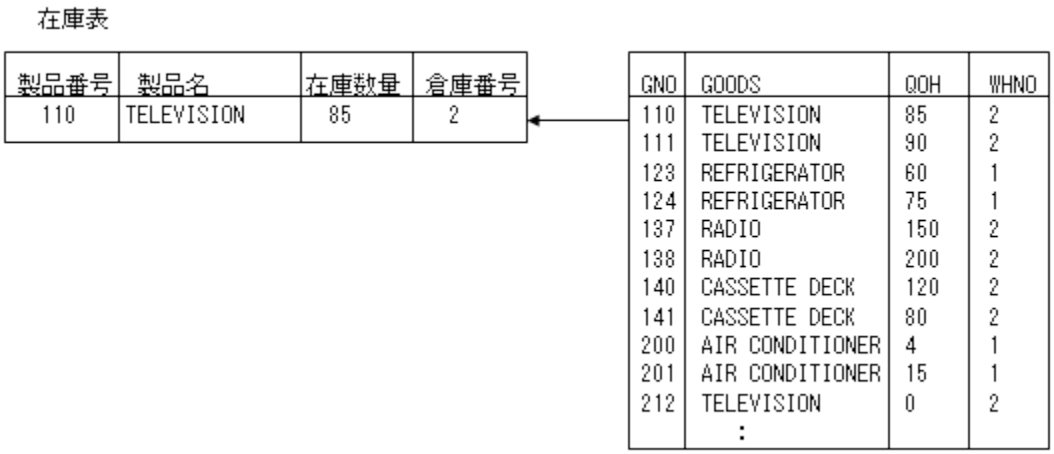
在庫表

製品番号	製品名	在庫数量	倉庫番号
390	DRIER	540	3

[6]: 在庫表の GNO が 111 のデータを"在庫表"に取り出します。



[7]: 在庫表の GNO が 110 のデータを"在庫表"に取り出します。



複数行指定ホスト変数を使用した FETCH PRIOR 文によるデータ取得

"STOCK 表" の先頭から 9 件のデータを 3 件ずつ 3 回の FETCH NEXT 文で取り出し、"STOCK 表" の先頭まで 3 件ずつ、2 回の FETCH PRIOR 文で戻る COBOL プログラムの例を以下に示します。

複数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、FETCH NEXT 文で 3 件のデータ取り出しを 3 回行い、次に FETCH PRIOR 文で 3 件のデータ取り出しを 2 回行います。

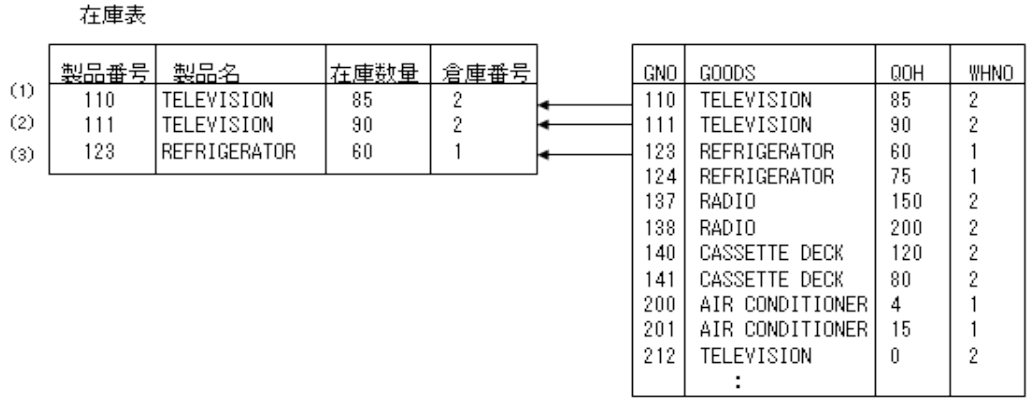
```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 在庫表.                                    *>|
   02 製品番号   PIC S9(4) COMP-5 OCCURS 3.      *>|
   02 製品名     PIC X(20) OCCURS 3.            *>|[1]
   02 在庫数量   PIC S9(9) OCCURS 3.            *>|
   02 倉庫番号   PIC S9(4) OCCURS 3.            *>|
 01 SQLSTATE    PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
*   :
 PROCEDURE DIVISION.
*   :
    EXEC SQL
      DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK ORDER BY GNO ASC
    END-EXEC.
*   :
    EXEC SQL
      OPEN CUR1                                 *>[2]
    END-EXEC.
*   :
```

```
           EXEC SQL
             FOR 3 FETCH NEXT CUR1 INTO :在庫表         *>[3]
           END-EXEC.
       *   :
           EXEC SQL
             FOR 3 FETCH NEXT CUR1 INTO :在庫表         *>[4]
           END-EXEC.
       *   :
           EXEC SQL
             FOR 3 FETCH NEXT CUR1 INTO :在庫表         *>[5]
           END-EXEC.
       *   :
           EXEC SQL
             FOR 3 FETCH PRIOR CUR1 INTO :在庫表        *>[6]
           END-EXEC.
       *   :
           EXEC SQL
             FOR 3 FETCH PRIOR CUR1 INTO :在庫表        *>[7]
           END-EXEC.
       *   :
```

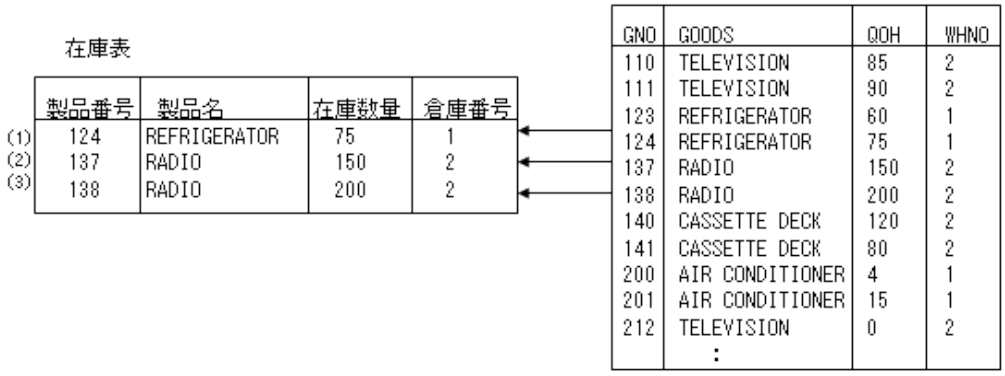
[1]: 表指定ホスト変数"在庫表"を定義します。

[2]: カーソル"CUR1"をオープンします。

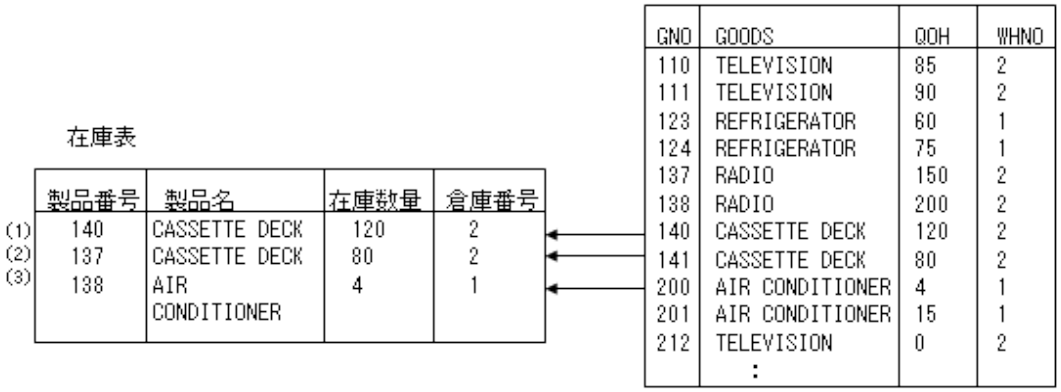
[3]: 在庫表の GNO が 110、111、123 のデータを"在庫表"に取り出します。



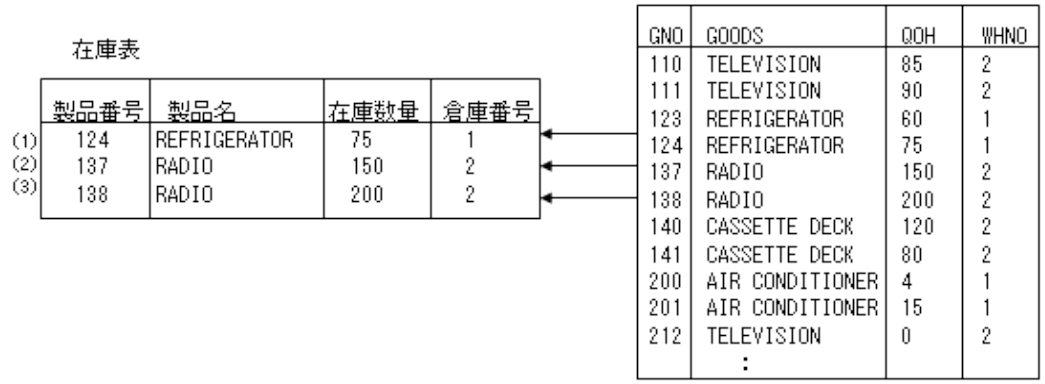
[4]: 在庫表の GNO が 124、137、138 のデータを"在庫表"に取り出します。



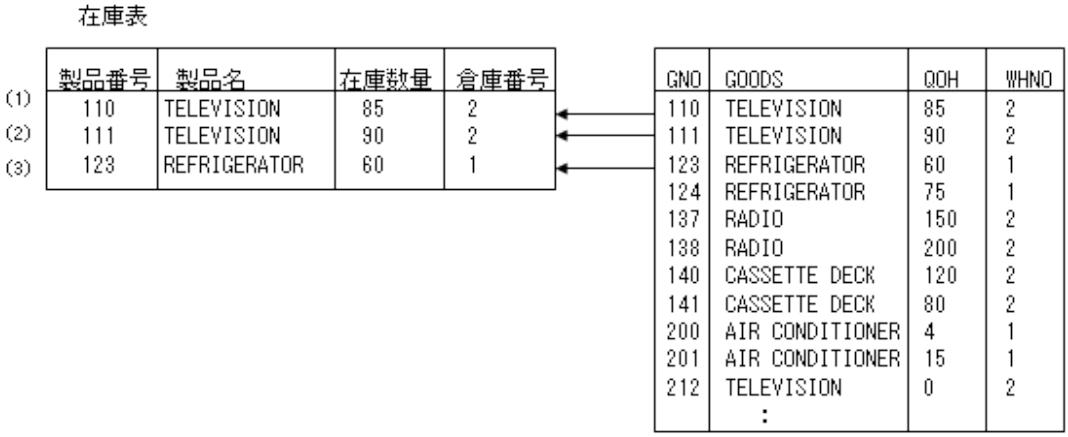
[5]: 在庫表の GNO が 140、141、200 のデータを"在庫表"に取り出します。



[6]: 在庫表の GNO が 124、137、138 のデータを"在庫表"に取り出します。添え字(1)のデータ項目から順にデータが設定されます。



[7]: 在庫表の GNO が 110、111、123 のデータを"在庫表"に取り出します。添え字(1)のデータ項目から順にデータが設定されます。



FETCH FIRST 文によるデータ取得

"STOCK 表" の先頭から 2 件のデータをそれぞれ FETCH NEXT 文で取り出し、FETCH FIRST 文で"STOCK 表"の先頭まで戻る COBOL プログラムの例を以下に示します。

単数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、FETCH NEXT 文で 2 件のデータを取り出し、次に FETCH FIRST 文で 1 件のデータを取り出します。

```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 在庫表.                              *>|
   02 製品番号   PIC S9(4) COMP-5.        *>|
   02 製品名     PIC X(20).               *>|[1]
   02 在庫数量   PIC S9(9).               *>|
   02 倉庫番号   PIC S9(4).               *>|
 01 SQLSTATE     PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
*    :
 PROCEDURE DIVISION.
*    :
     EXEC SQL
       DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
     END-EXEC.
*    :
     EXEC SQL
       OPEN CUR1                          *>[2]
     END-EXEC.
*    :
     EXEC SQL
       FETCH NEXT CUR1 INTO :在庫表       *>[3]
     END-EXEC.
*    :
     EXEC SQL
       FETCH NEXT CUR1 INTO :在庫表       *>[4]
     END-EXEC.
*    :
     EXEC SQL
       FETCH FIRST CUR1 INTO :在庫表      *>[5]
     END-EXEC.
*    :
```

[1]: 表指定ホスト変数"在庫表"を定義します。

[2]: カーソル"CUR1"をオープンします。

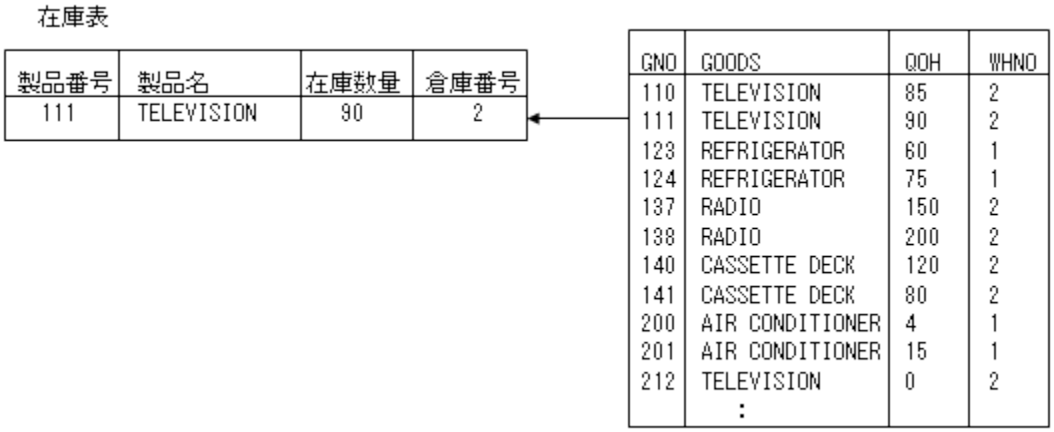
[3]: 在庫表の GNO が 110 のデータを"在庫表"に取り出します。

在庫表

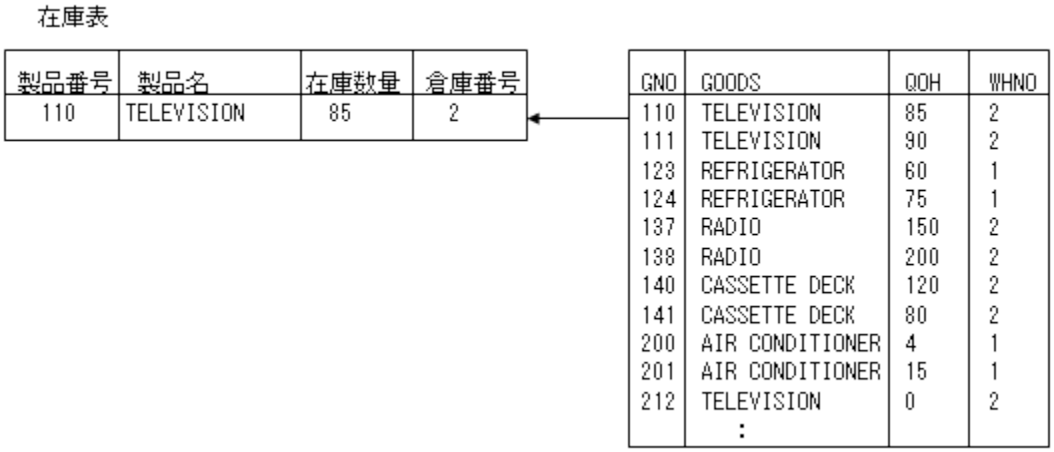
製品番号	製品名	在庫数量	倉庫番号
110	TELEVISION	85	2

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
111	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
	:		

[4]: 在庫表の GNO が 111 のデータを"在庫表"に取り出します。



[5]: 在庫表の GNO が 110 のデータを"在庫表"に取り出します。



FETCH LAST 文によるデータ取得

"STOCK 表" の最終行を FETCH LAST 文で取り出し、最終行から前の行を FETCH PRIOR 文で取り出す COBOL プログラムの例を以下に示します。

単数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、FETCH LAST 文で 1 件のデータを取り出し、次に FETCH PRIOR 文で 2 件のデータを取り出します。

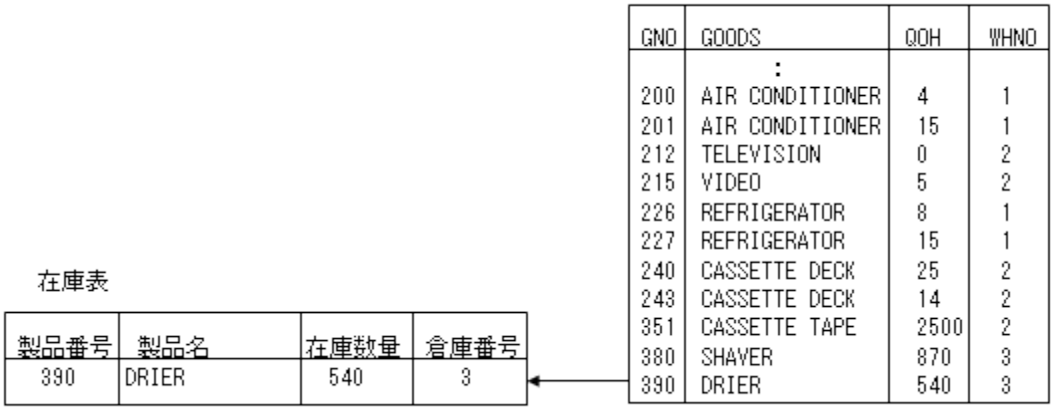
```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 在庫表.                                *>|
   02 製品番号   PIC S9(4) COMP-5.          *>|
   02 製品名     PIC X(20).                 *>|[1]
   02 在庫数量   PIC S9(9).                 *>|
   02 倉庫番号   PIC S9(4).                 *>|
 01 SQLSTATE    PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
*   :
 PROCEDURE DIVISION.
*   :
    EXEC SQL
      DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
    END-EXEC.
*   :
    EXEC SQL
      OPEN CUR1                            *>[2]
```

```
           END-EXEC.
       *   :
           EXEC SQL
             FETCH LAST CUR1 INTO :在庫表      *>[3]
           END-EXEC.
       *   :
           EXEC SQL
             FETCH PRIOR CUR1 INTO :在庫表     *>[4]
           END-EXEC.
       *   :
           EXEC SQL
             FETCH PRIOR CUR1 INTO :在庫表     *>[5]
           END-EXEC.
       *   :
```

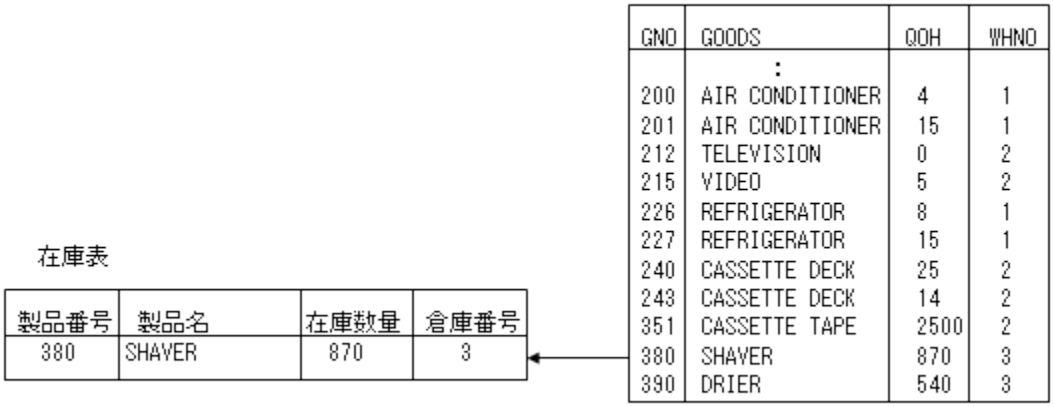
[1]: 表指定ホスト変数"在庫表"を定義します。

[2]: カーソル"CUR1"をオープンします。

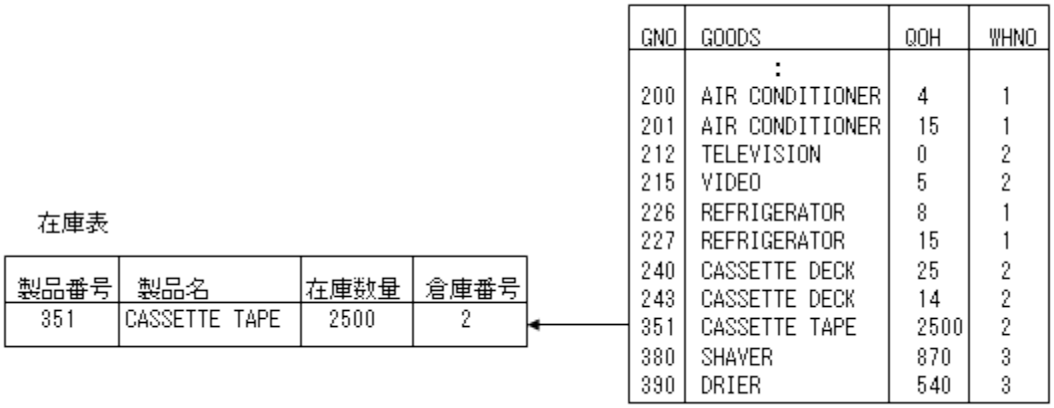
[3]: 在庫表の GNO が 390 のデータを"在庫表"に取り出します。



[4]: 在庫表の GNO が 380 のデータを"在庫表"に取り出します。



[5]: 在庫表の GNO が 351 ののデータを"在庫表"に取り出します。



複数行指定ホスト変数を使用した FETCH LAST 文によるデータ取得

"STOCK 表" の最終行から 3 件のデータを FETCH LAST 文で取り出し、そこから 3 件ずつ前の行のデータを FETCH PRIOR 文で取り出す COBOL プログラムの例を以下に示します。

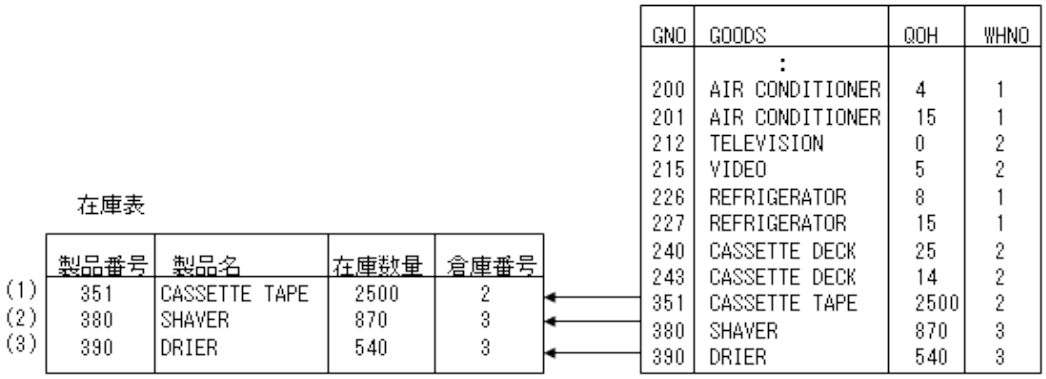
複数行指定ホスト変数を従属する表指定ホスト変数"在庫表"を使用し、FETCH LAST 文で 3 件のデータ取り出しを行い、次に FETCH PRIOR 文で 3 件のデータ取り出しを 2 回行います。

```
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 在庫表.                                          *>|
   02 製品番号   PIC S9(4) COMP-5 OCCURS 3.         *>|
   02 製品名     PIC X(20) OCCURS 3.                *>|[1]
   02 在庫数量   PIC S9(9) OCCURS 3.                *>|
   02 倉庫番号   PIC S9(4) OCCURS 3.                *>|
 01 SQLSTATE    PIC X(5).
    EXEC SQL END DECLARE SECTION END-EXEC.
*   :
 PROCEDURE DIVISION.
*   :
    EXEC SQL
      DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK ORDER BY GNO ASC
    END-EXEC.
*   :
    EXEC SQL
      OPEN CUR1                                     *>[2]
    END-EXEC.
*   :
    EXEC SQL
      FOR 3 FETCH LAST CUR1 INTO :在庫表            *>[3]
    END-EXEC.
*   :
    EXEC SQL
      FOR 3 FETCH PRIOR CUR1 INTO :在庫表           *>[4]
    END-EXEC.
*   :
    EXEC SQL
      FOR 3 FETCH PRIOR CUR1 INTO :在庫表           *>[5]
    END-EXEC.
*   :
```

[1]: 表指定ホスト変数"在庫表"を定義します。

[2]: カーソル"CUR1"をオープンします。

[3]: 在庫表の GNO が 351、380、390 の 3 件のデータを"在庫表"に取り出します。添え字(1)のデータ項目から順にデータが設定されます。

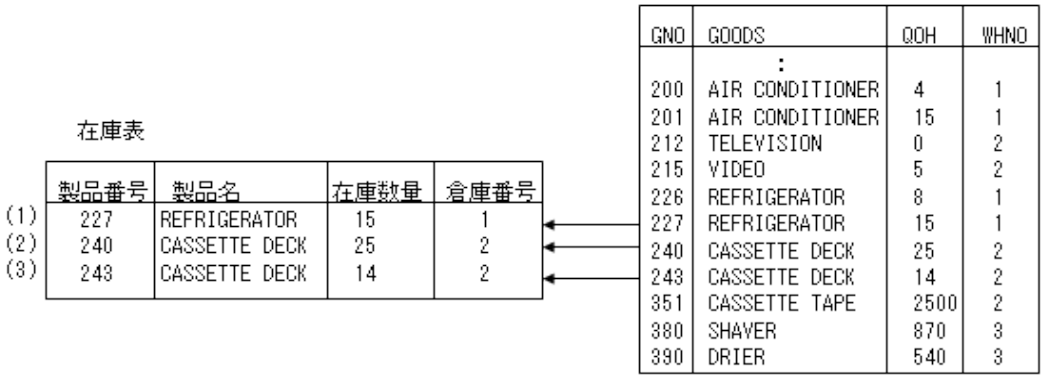


在庫表

	製品番号	製品名	在庫数量	倉庫番号
(1)	351	CASSETTE TAPE	2500	2
(2)	380	SHAVER	870	3
(3)	390	DRIER	540	3

GNO	GOODS	QOH	WHNO
	:		
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

[4]: 在庫表の GNO が 227、240、243 の 3 件のデータを"在庫表"に取り出します。添え字(1)のデータ項目から順にデータが設定されます。

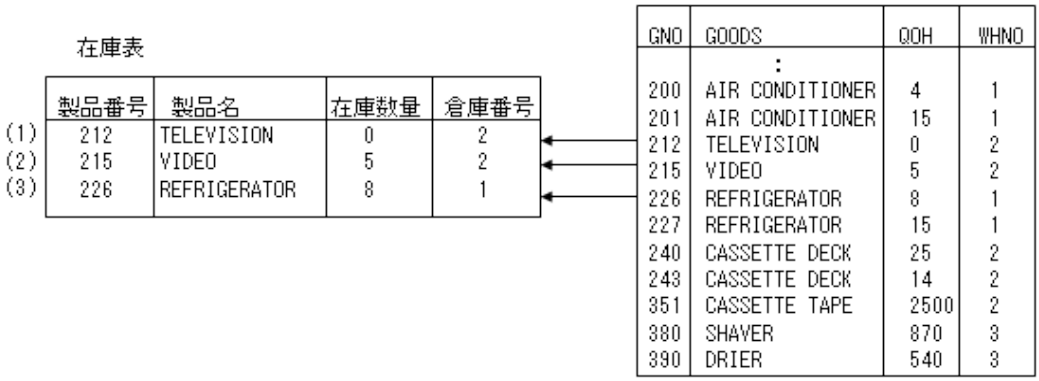


在庫表

	製品番号	製品名	在庫数量	倉庫番号
(1)	227	REFRIGERATOR	15	1
(2)	240	CASSETTE DECK	25	2
(3)	243	CASSETTE DECK	14	2

GNO	GOODS	QOH	WHNO
	:		
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

[5]: 在庫表の GNO が 212、215、226 の 3 件のデータを"在庫表"に取り出します。添え字(1)のデータ項目から順にデータが設定されます。



在庫表

	製品番号	製品名	在庫数量	倉庫番号
(1)	212	TELEVISION	0	2
(2)	215	VIDEO	5	2
(3)	226	REFRIGERATOR	8	1

GNO	GOODS	QOH	WHNO
	:		
200	AIR CONDITIONER	4	1
201	AIR CONDITIONER	15	1
212	TELEVISION	0	2
215	VIDEO	5	2
226	REFRIGERATOR	8	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

NCOBU00366

オブジェクト指向プログラミング機能を使用したデータベースアクセス

ここでは、オブジェクト指向プログラミング機能を使用したデータベースアクセス方法について説明します。

なお、説明中のクラス定義およびプログラム定義の例では、サンプルデータベースの STOCK 表(在庫表)を使用しています。

このセクションの内容

クラス定義からデータベースをアクセスする

クラス定義の中でデータベース表のデータを取り出し、データを更新する方法について説明します。

以下の例は、在庫管理(MANAGE-STOCK)メソッドの定義を含む COBOL のクラス定義です。在庫管理(MANAGE-STOCK)メソッドは、在庫管理対象のデータを取り出し、入庫または出庫を切り分けて在庫数量を計算し、再び表に格納するメソッドです。

```
      CLASS-ID.  DBACCESS-CLASS.
      ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
      OBJECT.
      DATA DIVISION.
      PROCEDURE DIVISION.
      METHOD-ID.  MANAGE-STOCK.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
          EXEC SQL BEGIN DECLARE SECTION END-EXEC.                *>|
      01 SQLSTATE            PIC X(5).                            *>|[1]
      01 INVENTORY-QUANTITY    PIC S9(9) COMP-5.                  *>|
      01 PRODUCT-NUMBER       PIC S9(4) COMP-5.                   *>|
          EXEC SQL END DECLARE SECTION END-EXEC.                  *>|
      LINKAGE SECTION.
      01 INPUT-PRODUCT-NUMBER  PIC S9(4) COMP-5.                   *>|
      01 INPUT-OUTPUT-ID      PIC S9(4) COMP-5.                    *>|[2]
      01 INPUT-OUTPUT-QUANTITY PIC S9(9) COMP-5.                    *>|
     *
      PROCEDURE DIVISION USING INPUT-PRODUCT-NUMBER                *> ─
                           INPUT-OUTPUT-ID                         *> ↑
                           INPUT-OUTPUT-QUANTITY.                  *> [3]
          MOVE INPUT-PRODUCT-NUMBER TO PRODUCT-NUMBER.
          EXEC SQL
            SELECT QOH INTO  :INVENTORY-QUANTITY FROM STOCK
            WHERE GNO = :PRODUCT-NUMBER
          END-EXEC
          IF INPUT-OUTPUT-ID = 1 THEN
            COMPUTE INVENTORY-QUANTITY
                 = INVENTORY-QUANTITY + INPUT-OUTPUT-QUANTITY
          ELSE
            COMPUTE INVENTORY-QUANTITY
                 = INVENTORY-QUANTITY - INPUT-OUTPUT-QUANTITY
          END-IF
          EXEC SQL
            UPDATE STOCK SET QOH = :INVENTORY-QUANTITY
                       WHERE GNO = :PRODUCT-NUMBER
          END-EXEC.                                               *> ↓
      END METHOD MANAGE-STOCK.                                    *> ─
      END OBJECT.
      END CLASS DBACCESS-CLASS.
```

データベースアクセスのクラス定義例

[プログラムの説明]

[1]: メソッドデータを定義します。

[2]: 在庫管理(MANAGE-STOCK)メソッドのインタフェースを定義します。

[3]: 在庫管理(MANAGE-STOCK)メソッドの手続きを記述します。

このメソッドでは、該当する製品番号の在庫数量を STOCK 表(在庫表)より取り出し、入庫または出庫を切り分けて在庫数量を計算し、再び表に格納します。

また、以下の例は、在庫管理(MANAGE-STOCK)メソッドを呼び出す COBOL プログラムの例です。

```
CONFIGURATION SECTION.
REPOSITORY.
   CLASS DBACCESS-CLASS.
DATA DIVISION.
WORKING-STORAGE SECTION.
   EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE            PIC  X(5).
   EXEC SQL END  DECLARE SECTION END-EXEC.
01 DBACCESS-OBJECT  OBJECT REFERENCE DBACCESS-CLASS.
01 製品番号   PIC S9(4) COMP-5.
01 入出区別   PIC S9(4) COMP-5.
01 入出数量   PIC S9(9) COMP-5.
01 終了要求   PIC X(1).
PROCEDURE DIVISION.
   EXEC SQL CONNECT TO DEFAULT END-EXEC.
   INVOKE  DBACCESS-CLASS "NEW" RETURNING DBACCESS-OBJECT.        *> [1]
   PERFORM TEST AFTER UNTIL 終了要求  = "Y"                       *>|
     DISPLAY "製品番号、入出区別(1or2)、入出数量を入力してください"     *>|
     ACCEPT  製品番号                                            *>|
     ACCEPT  入出区別                                            *>|
     ACCEPT  入出数量                                            *>|[2]
     INVOKE  DBACCESS-OBJECT "MANAGE-STOCK"                      *>|
                          USING 製品番号  入出区別  入出数量         *>|
     DISPLAY "在庫管理を終了しますか?(Y/N)"                         *>|
     ACCEPT  終了要求                                            *>|
   END-PERFORM                                                   *>|
   EXEC SQL COMMIT WORK END-EXEC
   EXEC SQL DISCONNECT DEFAULT END-EXEC
   STOP RUN.
```

データベースアクセスクラスのメソッドを呼び出すプログラムの例

[プログラムの説明]

[1]: "NEW"メソッドでオブジェクトインスタンスを生成します。

[2]: 製品番号、入出区別、入出数量の入力を要求し、在庫管理(MANAGE-STOCK)メソッドを呼び出します。 終了要求に"Y"が入力されるまで処理を繰り返し行います。

コネクションをオブジェクトインスタンス単位で利用する

コネクションをオブジェクトインスタンス単位で利用することによって、分散オブジェクト環境下で、オブジェクトインスタンス単位でトランザクション管理を行うことができます。

ここでは、コネクションをオブジェクトインスタンス単位で利用する方法について説明します。

コネクションをオブジェクトインスタンス単位で利用するプログラムを作成する

以下に、コネクションをオブジェクトインスタンス単位で利用するクラス定義の例を示します。

```
CLASS-ID. DBACCESS-CLASS.
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
OBJECT.
DATA DIVISION.
PROCEDURE DIVISION.
METHOD-ID.  MANAGE-STOCK.
DATA DIVISION.
 WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.                 *>─
 01 SQLSTATE  PIC X(5).                                      *>↑
 01 在庫数量  PIC S9(9) COMP-5.                              *>
 01 製品番号  PIC S9(4) COMP-5.                              *>[1]
 01 入出区別  PIC S9(4) COMP-5.                              *>
 01 入出数量  PIC S9(9) COMP-5.                              *>↓
    EXEC SQL END DECLARE SECTION END-EXEC.                  *>─
 01 終了要求  PIC X(1).
 01 反映要求  PIC X(1).
PROCEDURE DIVISION.
   EXEC SQL CONNECT TO DEFAULT END-EXEC                      *>─
   PERFORM TEST AFTER UNTIL 終了要求 = "Y"                   *>↑[2]
     DISPLAY "製品番号、入出区別(1or2)、入出数量を入力してください"
     ACCEPT 製品番号
     ACCEPT 入出区別
     ACCEPT 入出数量
     IF 入出区別 = 1 THEN
       EXEC SQL
         UPDATE STOCK SET QOH = QOH + :入出数量
           WHERE GNO = :製品番号
       END-EXEC
     ELSE
       EXEC SQL
         UPDATE STOCK SET QOH = QOH - :入出数量
           WHERE GNO = :製品番号
       END-EXEC
     END-IF
     EXEC SQL
       SELECT QOH INTO :在庫数量 FROM STOCK
         WHERE GNO = :製品番号
     END-EXEC
     DISPLAY "現在の在庫数量＝" 在庫数量
     DISPLAY "変更結果を反映してもいいですか？(Y/N)"
     ACCEPT 反映要求
     IF 反映要求 = "Y" THEN
       EXEC SQL COMMIT WORK END-EXEC
     ELSE
       EXEC SQL ROLLBACK WORK END-EXEC
     END-IF
     DISPLAY "在庫管理を終了しますか？(Y/N)"
     ACCEPT 終了要求
   END-PERFORM
   EXEC SQL DISCONNECT DEFAULT END-EXEC
   EXIT METHOD.                                             *>↓
END METHOD MANAGE-STOCK.                                    *>─
END OBJECT.
END CLASS DBACCESS-CLASS.
```

[プログラムの説明]

[1]: メソッドデータを定義します。

[2]: 在庫管理(MANAGE-STOCK)メソッドの手続きを記述します。このメソッドでは、コネクションの接続からデータ操作、コネクションの切断まで一連のデータベースアクセスを行います。データ操作では、製品番号、入出区別、入出数量の入力を要求し、入庫または出庫を切り分けて在庫数量を再計算します。次に再計算後の在庫数量を表示し、変更結果の反映有無を入力させます。データ操作は、終了要求に"Y"が入力されるまで繰り返し行います。

以下にデータベースアクセスクラスの在庫管理(MANAGE-STOCK)メソッドを呼び出すプログラムの例を示します。

```
CONFIGURATION SECTION.
REPOSITORY.
   CLASS DBACCESS-CLASS.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  OBJECT-COUNT   PIC  S9(1).
01  DBACCESS-OBJECT-1 OBJECT REFERENCE DBACCESS-CLASS.
01  DBACCESS-OBJECT-2 OBJECT REFERENCE DBACCESS-CLASS.
01  DBACCESS-OBJECT-3 OBJECT REFERENCE DBACCESS-CLASS.
PROCEDURE DIVISION.
     INVOKE DBACCESS-CLASS "NEW"                          *>─
                       RETURNING DBACCESS-OBJECT-1        *>↑[1]
     INVOKE DBACCESS-OBJECT-1 "MANAGE-STOCK"
     INVOKE DBACCESS-CLASS "NEW"
                       RETURNING DBACCESS-OBJECT-2
     INVOKE DBACCESS-OBJECT-2 "MANAGE-STOCK"
     INVOKE DBACCESS-CLASS "NEW"
                       RETURNING DBACCESS-OBJECT-3        *>↓
     INVOKE  DBACCESS-OBJECT-3 "MANAGE-STOCK"             *>─
     STOP RUN.
```

[プログラムの説明]

[1]: "NEW"メソッドによって 3 つのオブジェクトインスタンスを生成し、それぞれの在庫管理(MANAGE-STOCK)メソッドを呼び出します。

複数クライアントからの要求に対して、それぞれの在庫管理(MANAGE-STOCK)メソッドが実行されます。

コネクションをオブジェクトインスタンス単位で利用するプログラムを実行する

コネクションをオブジェクトインスタンス単位で利用するには、SQL 情報(ODBC 情報ファイル)のコネクション有効範囲に、オブジェクトインスタンスを指定して実行します。コネクション有効範囲にオブジェクトインスタンスを指定する方法については、コネクション有効範囲を参照してください

SQL 情報は、実行環境設定ユーティリティを使用してください。

プログラムのビルド

埋込み SQL を使用してデータベースにアクセスする COBOL プログラムのビルドは、データベース固有の翻訳オプションを指定せずにビルドできます。

- ホスト変数名に使用している英字の大小を区別する場合は、翻訳オプション NOALPHAL を指定して翻訳します。
- 利用者が定義する名前に埋込み SQL 文のキーワードは使用できません。埋込み SQL 文のキーワードについては、埋込み SQL 文のキーワード一覧を参照してください。
- 実行時データのコード系は、アクセスするデータベースのデータ型のコード系に合わせて翻訳オプション RCS (実行時データのコード系)を指定してください。実行時データのコード系と、データ型のコード系があっていない場合は、レスポンスが遅くなる可能性があります。データベースで扱うデータ型の対応は、データ型の対応を参照してください。

NCOBU00370

プログラムの実行

ここでは、プログラムを実行するために必要な実行環境の構築と、SQL 情報(ODBC 情報)の設定方法について説明します。

このセクションの内容

SQL 情報

SQL 情報の内容について説明します。

SQL 情報の設定

SQL 情報を設定する方法について説明します。

SQL 情報

アプリケーション構成ファイルおよび ODBC 情報ファイルに設定する SQL 情報は、主に、クライアントとサーバ間のコネクション(CONNECT 文などで指定する)を確立するために必要な情報が設定されています。

SQL 情報は、実行環境設定ユーティリティ、または、ODBC 情報設定ユーティリティを使用して設定してください。NetCOBOL for .NET では、実行環境設定ユーティリティを使用して設定することを推奨しています。

アプリケーション構成ファイル、または、ODBC 情報ファイルの内容は、以下の 4 つの SQL 情報に分類されます。

- サーバ情報
- デフォルトコネクション情報
- コネクション有効範囲
- SQL オプション情報

SQL オプション情報は、アプリケーション構成ファイルのみ設定することができます。

NCOBU00845

SQL 情報の設定

SQL 情報を設定する方法には、以下の 2 つがあります。

- 実行環境設定ユーティリティを使用して、アプリケーション構成ファイルに設定する方法 （推奨）
- ODBC 情報設定ユーティリティを使用して、ODBC 情報ファイルに設定する方法

このセクションの内容

連携ソフトウェアおよびハードウェア環境の整備

連携するソフトウェアおよびハードウェアの環境を整備するための手順について説明します。

アプリケーション構成ファイルを使用した SQL 情報の設定

アプリケーション構成ファイルに SQL 情報を設定するための実行環境設定ユーティリティの使い方と実行環境の構築について説明します。

ODBC 情報ファイルを使用した SQL 情報の設定

ODBC 情報ファイルに SQL 情報を設定するための ODBC 情報設定ユーティリティの使い方と実行環境の構築について説明します。

[参考]: ODBC 情報ファイルは、実行環境変数@ODBC_Inf (ODBC 情報ファイルの指定)に指定したファイルです。SQL 情報(主に、クライアントとサーバ間のコネクション(CONNECT 文などで指定する)を確立するために必要な情報)が設定されています。NetCOBOL for .NET では、ODBC 情報ファイルを使用した SQL 情報の設定と実行は、互換のために残されています。

連携ソフトウェアおよびハードウェア環境の整備

COBOL アプリケーションから、ODBC を使用してサーバのデータベースへアクセスするためには、連携するソフトウェアおよびハードウェアの環境を整備する必要があります。以下に、ODBC 環境のセットアップの手順について簡単に説明します。

ODBC 環境のセットアップ

- Windows システムへの ODBC の初期導入

  Windows システムへ ODBC の環境をはじめて導入する場合は、ODBC ドライバと同時に提供される ODBC セットアップを使用してください。

  この導入作業により、Windows のコントロールパネルに ODBC が追加されます。

- ODBC ドライバの導入とデータソースの作成

  ODBC データソースアドミニストレータ(通常、Windows のコントロールパネルのグループに存在しています)を起動して、セットアップを行ってください。このセットアップでは、ODBC ドライバの導入操作およびデータソースの定義を行います。

リモートデータベースアクセス(ODBC)機能を利用した COBOL プログラムが、サービスから呼び出される場合は、データソースをシステムデータソースとして定義する必要があります。システムデータソースは、ユーザではなくコンピュータに対して定義するデータソースです。システムデータソースは、サービスを含むコンピュータ上のすべてのユーザが認識することができます。システムデータソースについては、ODBC データソースアドミニストレータのオンラインヘルプを参照してください。

- ODBC ドライバに関係する環境の整備

  ODBC ドライバは、そのドライバが動作するために必要な環境をあらかじめ想定しています。ODBC ドライバを動作させるために必要な環境については、各 ODBC ドライバのヘルプおよびマニュアルを参照してください。ODBC ドライバのヘルプは、ODBC データソースアドミニストレータを起動することにより参照できます。

データソースとの接続の確認

通常、データベースは、クライアントからサーバのデータベースを操作するためのプログラムを提供しています。ODBC を使用した COBOL アプリケーションを実行する前に、これらのプログラムを利用して、クライアントとサーバの間で接続が成功しているかどうかを確認しておくと安全です。

ODBC データソースアドミニストレータ、アプリケーション構成ファイルなどの設定が完了した後に COBOL アプリケーションを実行してください。

NCOBU00371

アプリケーション構成ファイルを使用した SQL 情報の設定

ここでは、プログラムを実行するために必要な実行環境の構築と、アプリケーション構成ファイルに SQL 情報を設定するための実行環境設定ユーティリティの使い方について説明します。

このセクションの内容

実行環境の構築

プログラムを実行するには、SQL 情報を設定したアプリケーション構成ファイル(config ファイル)が必要です。アプリケーション構成ファイルに SQL 情報を設定するには、実行環境設定ユーティリティを使用します。

以下の図では、ファイルのそれぞれの設定情報がどのように関連づけられるかを示します。

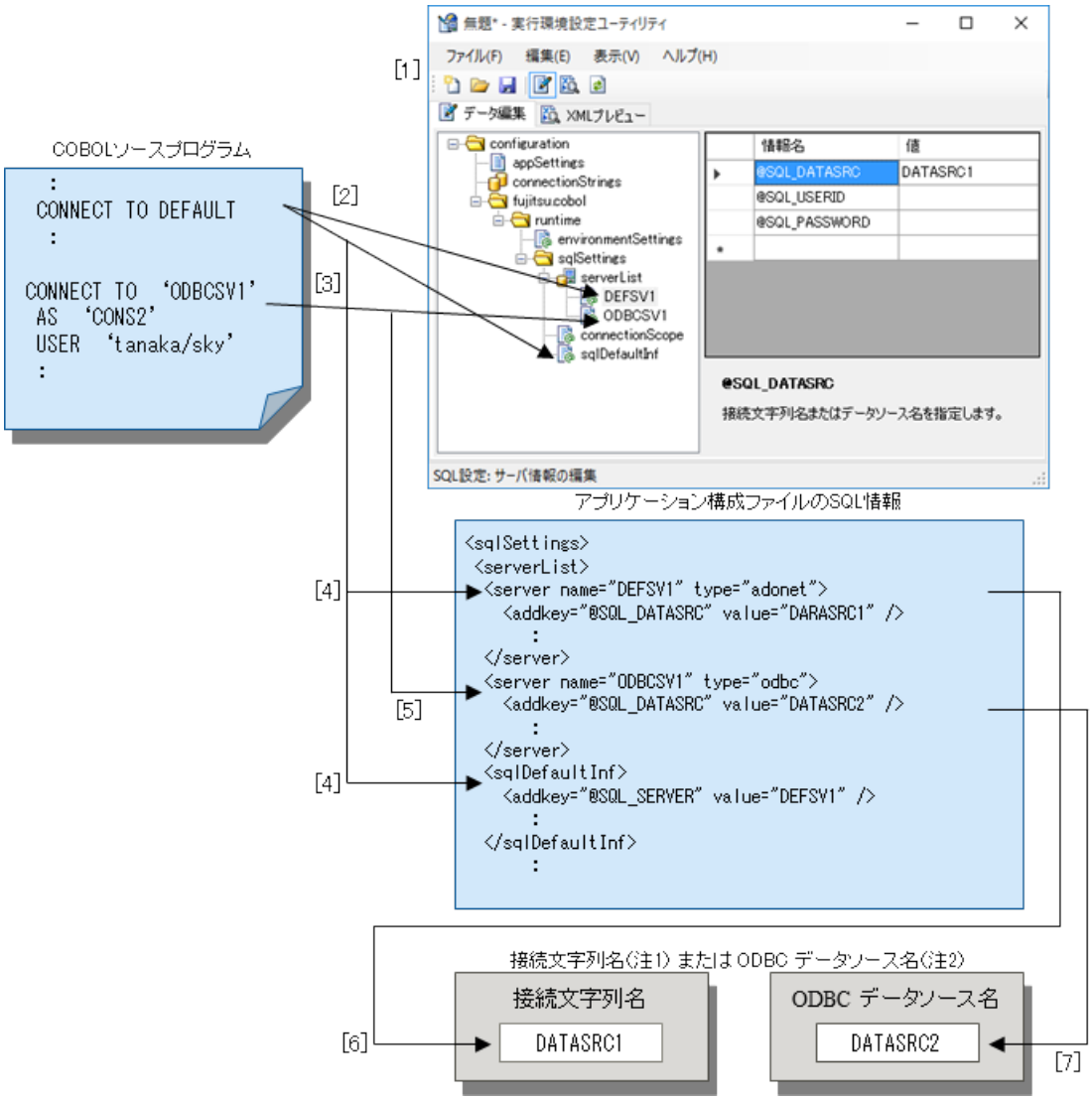


図:アプリケーション構成ファイルと SQL 情報の関連

注 1: 接続文字列名とは、アプリケーション構成情報に設定された接続文字列設定の名前です。接続文字列については、接続文字列の設定方法(ADO.NET)を参照して下さい。

注 2: ODBC データソースとは、ODBC ドライバ、ネットワークシステムおよびデータベースなどの環境全般を総称したものです。

[図の説明]

[1]: 実行環境設定ユーティリティを使用して、SQL 情報を設定します。

[2]:CONNECT 文に DEFAULT を指定した場合、SQL 情報にデフォルトコネクションとするサーバ名の定義を、<sqlDefaultInf>要素にそのサーバ名を関連づけます。

[3]:CONNECT 文にサーバ名を指定した場合、SQL 情報にそのサーバ名を定義します。

[4]:[2]によって、<server>要素および<sqlDefaultInf>要素がアプリケーション構成ファイルに設定されます。

[5]:[3]によって、<server>要素がアプリケーション構成ファイルに設定されます。

[6]:SQL 情報の各サーバの ADO.NET 接続文字列名は、アプリケーション構成ファイルの<connectionStrings>要素に定義されている接続文字列の名前です。

[7]:SQL 情報の各サーバの ODBC データソース名の定義は、Windows システムの ODBC データソースアドミニストレータで定義したデータソースの名前を指定します。

実行環境設定ユーティリティの使い方

実行環境設定ユーティリティは、データベースアクセスに必要な SQL 情報を、アプリケーション構成ファイルの決められた形式に設定する機能を持ちます。また、SQL 情報を変更する場合も、実行環境設定ユーティリティを使用します。

ここでは、実行環境設定ユーティリティを使用した各種 SQL 情報の設定方法を示します。実行環境設定ユーティリティの操作方法については、実行環境設定ユーティリティの操作を参照してください。

このセクションの内容

サーバ情報の設定方法(ADO.NET/ODBC)

サーバ情報の設定方法について説明します。

デフォルトコネクション情報の設定方法(ADO.NET/ODBC)

デフォルトコネクション情報の設定方法について説明します。

コネクション有効範囲の設定方法(ADO.NET/ODBC)

コネクション有効範囲の設定方法について説明します。

SQL オプション情報の設定方法(ODBC)

SQL オプション情報の設定方法について説明します。

接続文字列の設定方法(ADO.NET)

ADO.NET 接続で使用する接続文字列の設定方法について説明します。

NCOBU00846

サーバ情報の設定方法(ADO.NET/ODBC)

CONNECT 文およびデフォルトコネクション情報で指定するサーバ名に対応する、サーバ情報の設定方法について説明します。

1. 実行環境設定ユーティリティの起動

   実行環境設定ユーティリティを起動する方法は 2 つあります。

   - 開発環境から起動 Visual Studio でプロジェクトを開いている場合は、プロジェクトから起動することができます。詳細については、プロジェクトの実行環境設定を参照してください。
   - 運用環境（開発環境がない場合）の起動

     [スタート] - アプリ一覧(*) - お使いの NetCOBOL for .NET 製品名 - [実行環境設定ユーティリティ]を選択します。

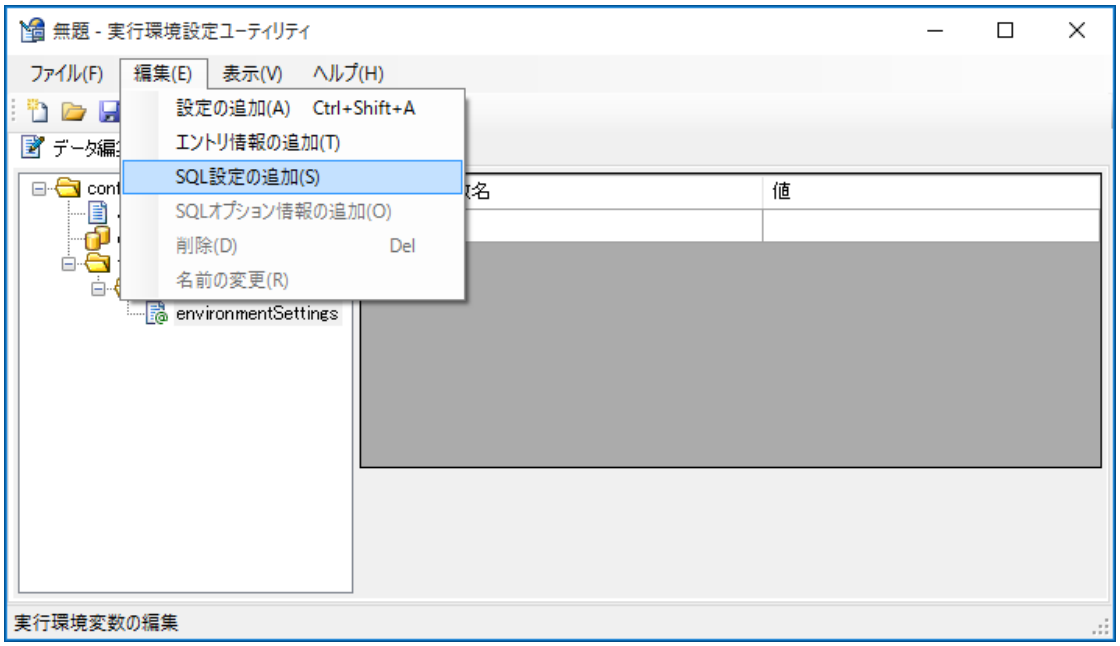
     * : ご使用の Windows によって、以下の操作をすることで同様の画面になります。

       § Windows 7 : [スタート]メニュー - [すべてのプログラム]

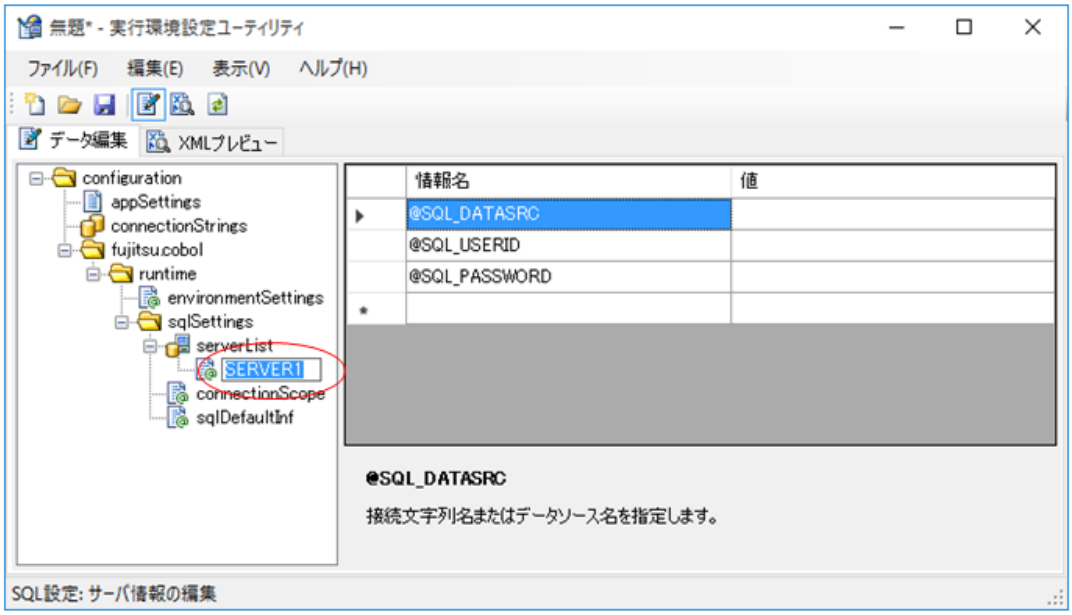
       § Windows 8.1 および Windows Server 2012 R2 : [スタート]画面 - [↓] - [アプリ]

2. SQL 情報の追加

   実行環境設定ユーティリティのメニューから [編集]−[SQL 設定の追加]を選択し、アプリケーション構成ファイルに SQL 情報のテンプレートを追加します。



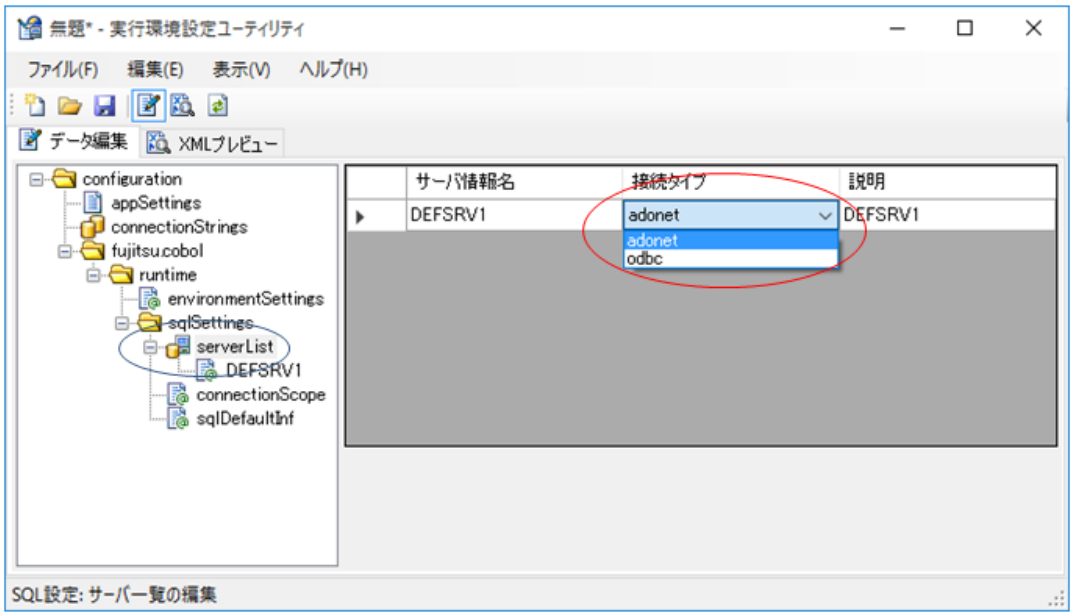
3. 以下は、CONNECT 文で指定されたサーバ名の設定を行っています。



"SERVER1"にサーバ名を設定します。ここでは、"DEFSV1"というサーバ名に変更します。

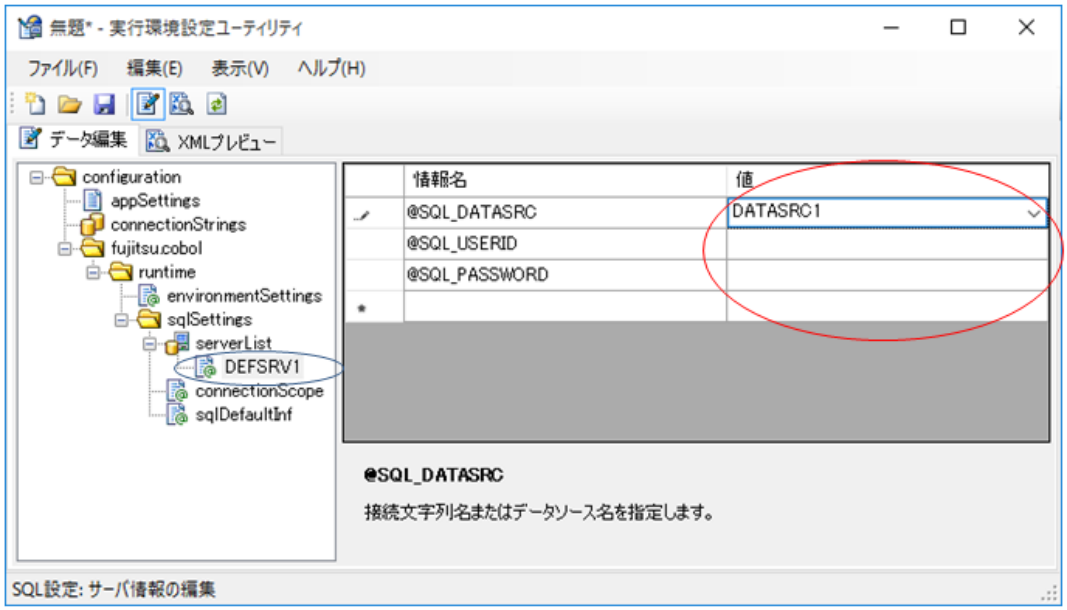
4. 接続タイプの設定

ADO.NET 接続を行う場合には"adonet"を、ODBC 接続を行う場合には"odbc"を選択します。



サーバ情報一覧(青枠の部分)の"serverList"を選択し、赤枠の部分で接続タイプを設定します。

5. サーバ名に対応する接続文字列名またはデータソース名など、必要な情報の設定



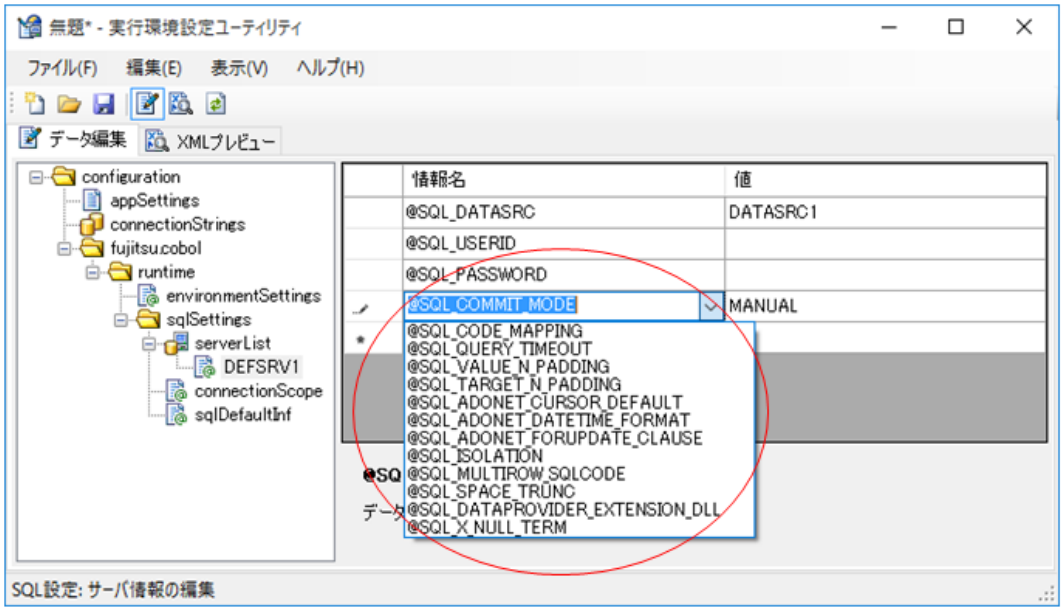
サーバ名 "DEFSV1"(青枠の部分)を選択し、赤枠の部分に必要な情報を入力します。

@SQL_DATASRC には、接続文字列名または ODBC データソース名を指定します。接続タイプとして ADO.NET が指定されている場合は接続文字列の設定方法(ADO.NET)で設定した接続文字列名の一覧がドロップダウンリストに表示されます。接続タイプとして ODBC が指定されている場合はデータソース(注)の一覧がドロップダウンリストに表示されます。

注: マシンデータソース(ユーザデータソースおよびシステムデータソース)のみ表示されます。ファイルデータソースの一覧は表示されません。

データベースの接続に認証が必要な場合は、@SQL_USERID、@SQL_PASSWORD にそれぞれユーザ ID とパスワードを指定します。

6. 情報の追加



赤枠の部分で情報名を選択し、必要な情報を追加・設定します。

設定可能な情報については、サーバ情報を参照してください。

デフォルトコネクション情報の設定方法(ADO.NET/ODBC)

DEFAULT 指定の CONNECT 文が記述された場合に使用する、デフォルトコネクション情報(サーバ名、ユーザ ID およびパスワード)を設定する方法について説明します。

デフォルトコネクション情報(青枠の部分)の"sqlDefaultInf"を選択し、赤枠の部分を入力します。

@SQL_SERVER には、サーバ情報の設定方法(ADO.NET/ODBC)で追加したサーバ名を指定します。@SQL_USERID、@SQL_PASSWORD にはそれぞれデフォルト接続を行うデータベースサーバへのログインに必要なユーザ ID とパスワードを指定します。

NCOBU00848

コネクション有効範囲の設定方法(ADO.NET/ODBC)

コネクション有効範囲の設定方法について説明します。

コネクション有効範囲(青枠の部分)の"connectionScope"を選択し、赤枠の部分に適切な値を設定します。

@SQL_CONNECTION_SCOPE には接続の有効な利用可能な範囲を指定します。

SQL オプション情報の設定方法(ODBC)

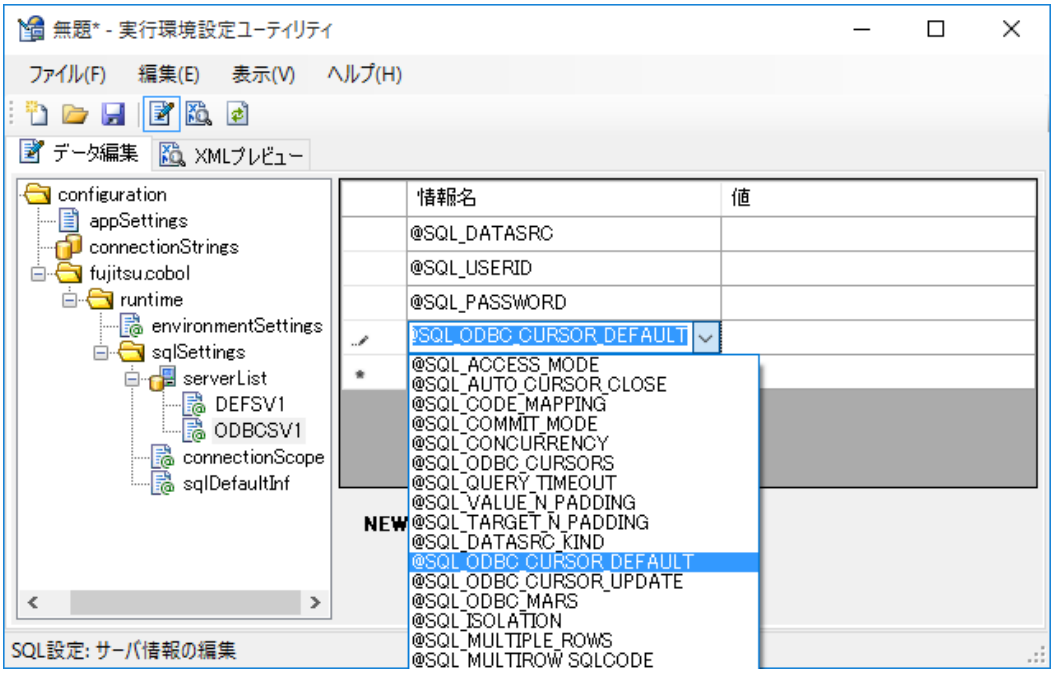
SQL オプション情報の設定方法について説明します。

SQL オプション情報は、ODBC 接続でカーソルタイプなどを指定する場合に設定します。 詳細については、カーソルのタイプによるオプション指定を参照してください。

ODBC 接続の場合に有効であるため、サーバ情報の設定方法(ADO.NET/ODBC)の操作では、接続タイプを "odbc" に設定しておく必要があります。

ここでは、FOR UPDATE が省略されているカーソル宣言で、そのカーソルのオプションを指定する方法を説明します。

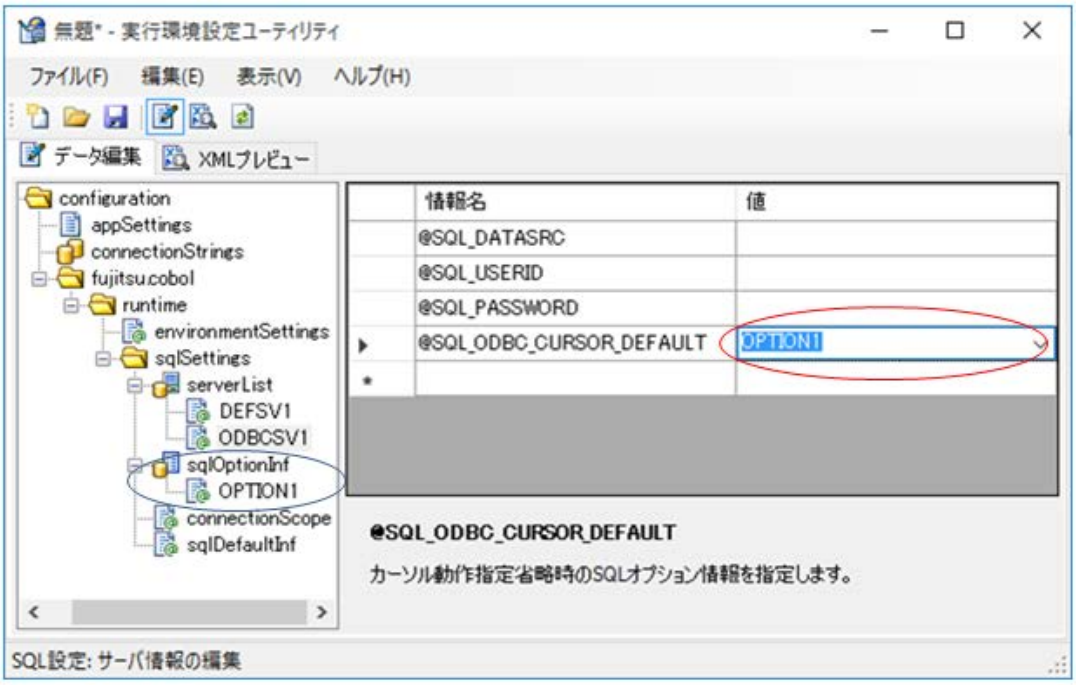
1. サーバ情報に@SQL_ODBC_CURSOR_DEFAULT を追加



"@SQL_ODBC_CURSOR_DEFAULT"を選択します。

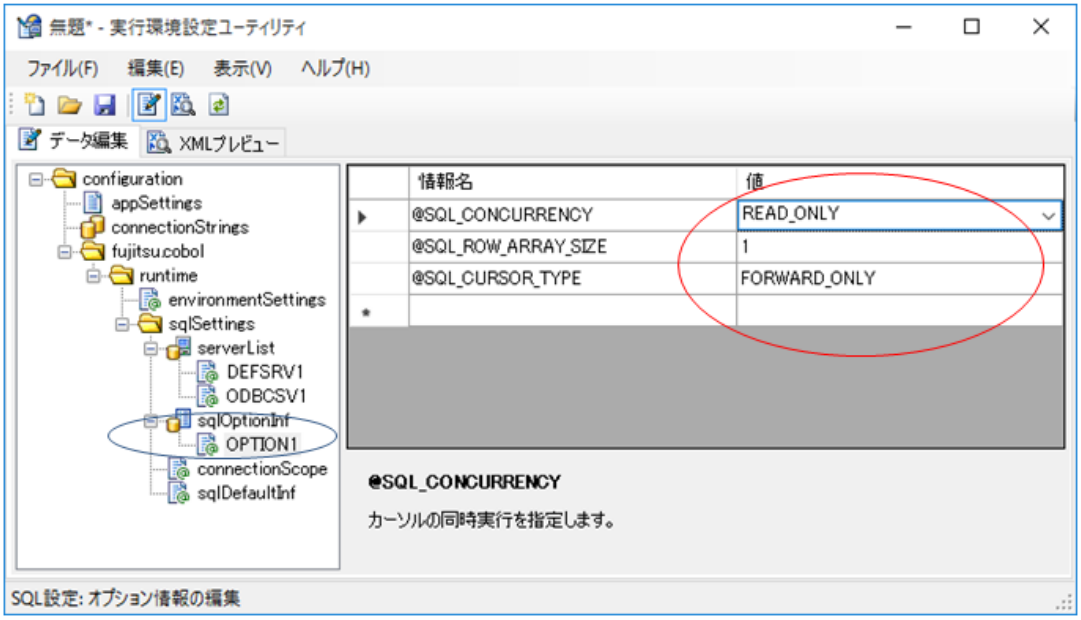
2. SQL オプション情報の設定

   サーバ情報に、@SQL_ODBC_CURSOR_DFFAULT および@SQL_ODBC_CURSOR_UPDATE が追加されると、自動的に SQL オプション情報が作成されます。



   青枠の部分には SQL オプション情報が追加され、赤枠の部分にはオプション名を設定します(通常は自動的に設定されます)。

3. オプション名に対応する必要な情報を設定

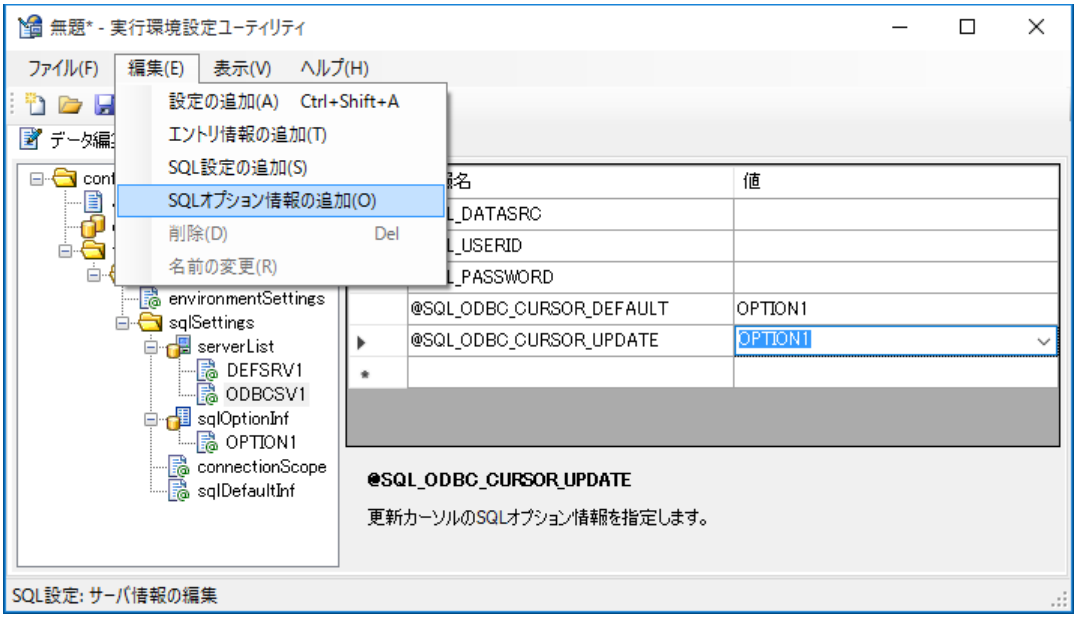


   SQL オプション情報(青枠の部分)のオプション名"OPTION1"を選択し、赤枠の部分を入力します。

   @SQL_CONCURRENCY には、カーソルの CONCURRENCY の種別を指定します。
   @SQL_ROW_ARRAY_SIZE には、カーソルの読み取りブロックサイズを指定します。
   @SQL_CURSOR_TYPE には、カーソルタイプを指定します。

4. 別の SQL オプション情報の追加

さらに@SQL_ODBC_CURSOR_DFFAULT や@SQL_ODBC_CURSOR_UPDATE に別の SQL オプション情報を設定したい場合は、実行環境設定ユーティリティのメニューから [編集]−[SQL オプション情報の追加]を選択し、アプリケーション構成ファイルに SQL オプション情報のテンプレートを追加します。



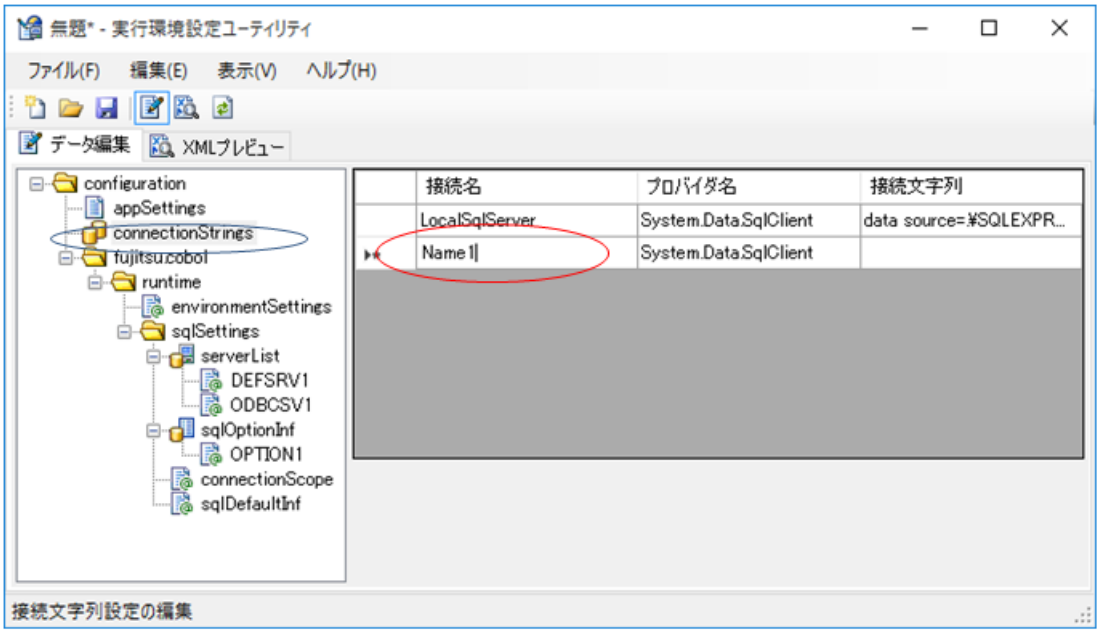
NCOBU00850

接続文字列の設定方法(ADO.NET)

接続文字列とは、サーバ名、データベース名、認証方法など、データベースへの接続に必要な環境情報が文字列として保持され、接続に使用するデータプロバイダーを含めた接続文字列情報としてアプリケーション構成ファイルに設定されます。 接続文字列情報は、接続文字列名(接続名)で識別されるため、ADO.NET 接続では、サーバ情報の@SQL_DATASRC に ODBC データソース名の代わりに、接続文字列名を指定します。

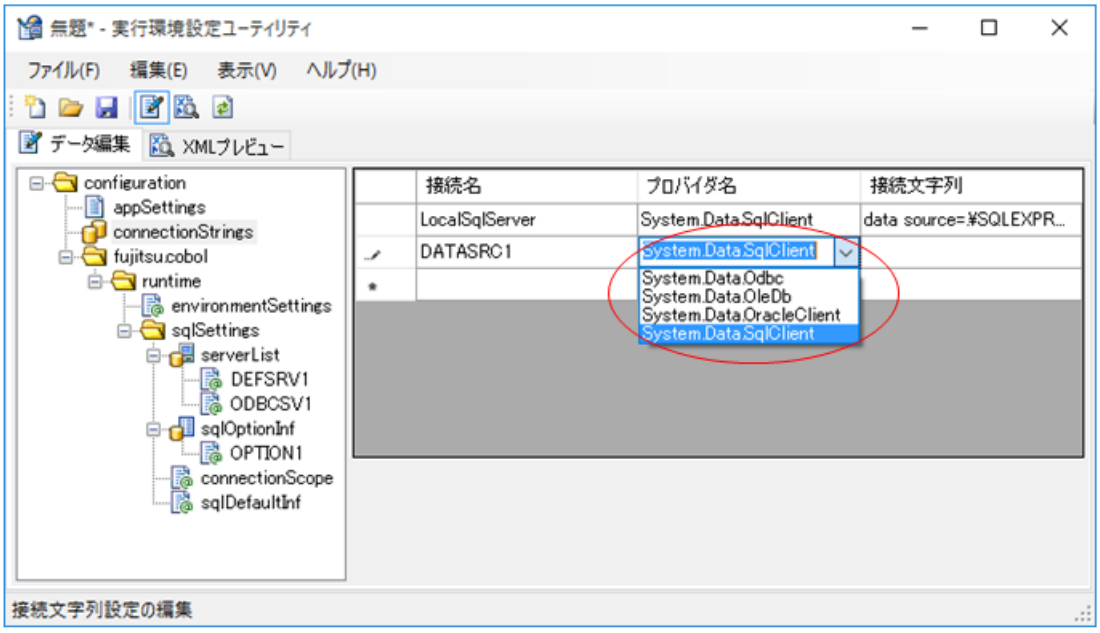
以下に、接続文字列の設定方法について説明します。

1. 接続文字列名の追加



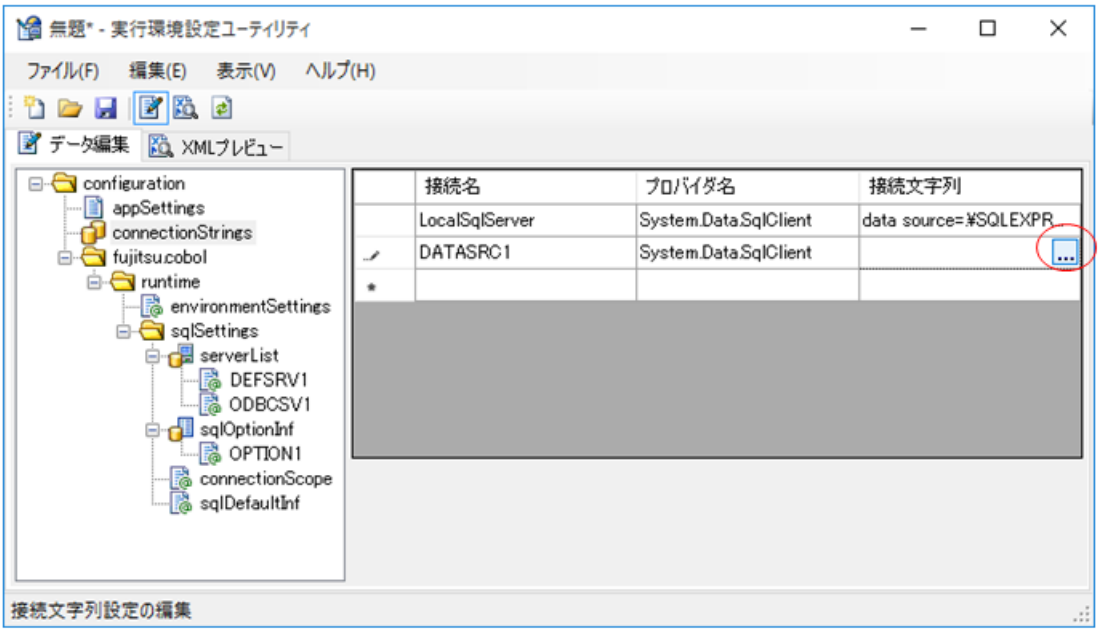
接続文字列設定(青枠の部分)の"connectionStrings"を選択し、赤枠の部分に接続文字列名を入力します。ここでは、接続文字列名を"DATASRC1"とします。

2. データプロバイダーの選択

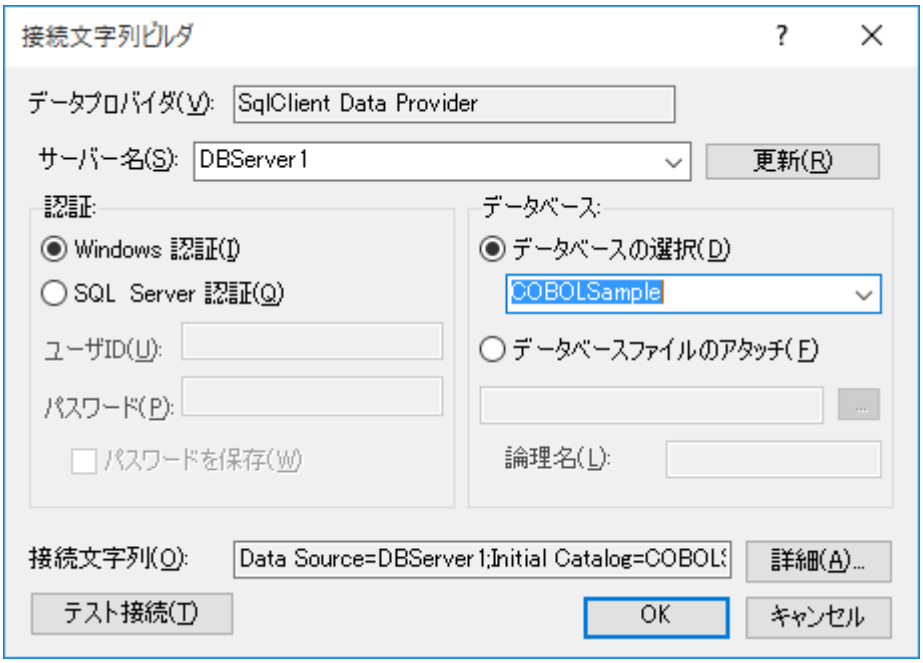


赤枠の部分でデータベースの接続に使用するデータプロバイダーを選択します。 SQL Server に接続する場合は、"System.Data.SqlClient"を選択します。

3. 接続文字列の編集



接続文字列を編集する場合は、赤枠の部分のボタンをクリックし、[接続文字列ビルダ]ダイアログボックスを開きます。このダイアログボックスで設定できる内容は、データプロバイダーによって異なります。以下は、プロバイダ名として、"System.Data.SqlClient"(SQLClient Data Provider)を選択した場合に表示される[接続文字列ビルダ]ダイアログボックスです。



ここでは、接続文字列情報に必要なサーバ名として、"DBServer1"、認証方法は Windows 認証、接続するデータベース名は、"COBOLSample"を設定しています。

この場合、接続文字列は以下のように設定されます。

```
Data Source=DBServer1;Initial Catalog=COBOLSample;
           Integrated Security=True;Persist Security Info=False
```

実行環境設定ユーティリティでは、.NET Framework 標準のデータプロバイダーに対応した[接続文字列ビルダ]ダイアログボックスを用意しています。それぞれの[接続文字列ビルダ]ダイアログボックスについては以下を参照してください。

- [接続文字列ビルダ] ダイアログ (SqlClient データプロバイダ)
- [接続文字列ビルダ] ダイアログ (OLE DB データプロバイダー)
- [接続文字列ビルダ] ダイアログ (ODBC データプロバイダ)
- [接続文字列ビルダ] ダイアログ (OracleClient データプロバイダ)

- 接続文字列については、Visual Studio のドキュメントの接続文字列を参照してください。
- アプリケーション構成ファイルの connectionStrings 要素については、.NET Framework のドキュメントの connectionStrings 要素 (ASP.NET 設定スキーマ)を参照してください。

ODBC 情報ファイルを使用した SQL 情報の設定

この機能は互換のために残されています。NetCOBOL for .NET では、ODBC 情報ファイルおよび ODBC 情報設定ユーティリティを使用することは推奨していません。アプリケーション構成ファイルを使用した SQL 情報の設定をお勧めします。

ここでは、プログラムを実行するために必要な実行環境の構築と、ODBC 情報設定ユーティリティの使い方について説明します。

ODBC 情報ファイルを使用した場合、ADO.NET 接続でのデータベースアクセスはサポートしていません。

このセクションの内容

実行環境の構築

プログラムを実行するための実行環境の構築について説明します。

実行環境情報の設定

ODBC 環境に必要な実行環境情報について説明します。

ODBC 情報ファイルの形式

ODBC 情報ファイルの形式について説明します。

ODBC 情報設定ユーティリティの使い方

ODBC 情報設定ユーティリティの使い方について説明します。

ODBC 情報ファイルの設定内容の最大長

ODBC 情報ファイルの内容の最大長について説明します。

NCOBU00375

実行環境の構築

この機能は互換のために残されています。NetCOBOL for .NET では、ODBC 情報ファイルおよび ODBC 情報設定ユーティリティを使用することは推奨していません。アプリケーション構成ファイルを使用した SQL 情報の設定をお勧めします。

プログラムを実行するには、実行環境情報を設定した実行用の初期化ファイルおよび ODBC 情報ファイルが必要になります。各ファイルに設定する情報は、以下に示すように対応付けて指定します。

以下の図では、それぞれのファイルの設定情報がどのように関連付けられるかを示します。

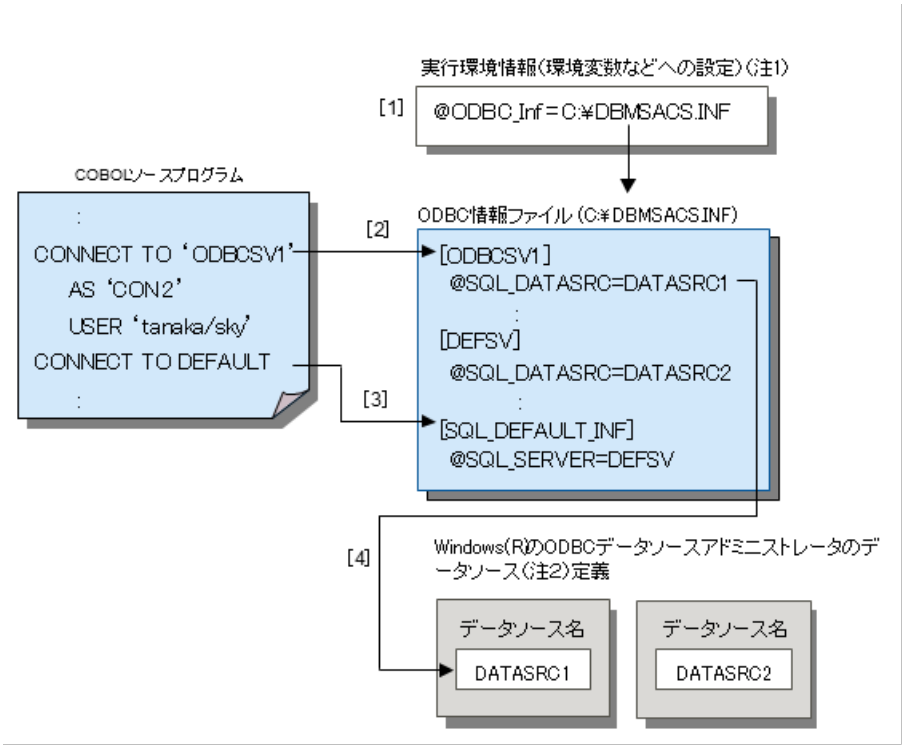


図:各情報ファイルの関連

注 1: 詳細については、実行環境の設定を参照してください。

注 2 : データソースとは、ODBC ドライバ、ネットワークシステムおよびデータベースなどの環境全般を総称したものです。

[図の説明]

[1]: ODBC 情報ファイル名(`C:¥DBMSACS.INF`)を指定します。

[2]: CONNECT 文にサーバ名を記述した場合、ODBC 情報ファイルにサーバ名を指定します。

[3]: CONNECT 文に DEFAULT を記述した場合、ODBC 情報ファイルにデフォルトコネクション情報の定義を示す固定文字列を指定します。

[4]: ODBC 情報ファイルの各サーバのデータソース名の定義には、Windows システムの ODBC データソースアドミニストレータで定義したデータソース名を指定します。

実行環境情報の設定

この機能は互換のために残されています。NetCOBOL for .NET では、ODBC 情報ファイルおよび ODBC 情報設定ユーティリティを使用することは推奨していません。アプリケーション構成ファイルを使用した SQL 情報の設定をお勧めします。

クライアントとサーバ間の連携ソフトウェアとして ODBC 環境を選択する場合は、実行環境変数@ODBC_Inf を設定します。実行環境変数@ODBC_Inf の詳細については、@ODBC_Inf (ODBC 情報ファイルの指定)を参照してください。

NCOBU00377

ODBC 情報ファイルの形式

この機能は互換のために残されています。NetCOBOL for .NET では、ODBC 情報ファイルおよび ODBC 情報設定ユーティリティを使用することは推奨していません。アプリケーション構成ファイルを使用した SQL 情報の設定をお勧めします。

ODBC 情報ファイルに設定する SQL 情報(ODBC 情報)は、主に、クライアントとサーバ間のコネクション(CONNECT 文などで指定する)を確立するために必要な情報が設定されています。

ODBC 情報ファイルは、ODBC 情報設定ユーティリティを使用して設定してください。

ODBC 情報ファイルの内容は、以下の 3 つの SQL 情報に分類されます。

- サーバ情報
- デフォルトコネクション情報
- コネクション有効範囲

ODBC 情報ファイルでは、SQL オプション情報はサポートしていません。

各情報は、以下に示すセクション内に設定します。

セクション名	説明
[サーバ名]	サーバ情報の定義開始を示すセクション名です。CONNECT 文またはデフォルトコネクション情報に指定したサーバ名と対応付けます。次のセクション名またはファイルの終端までが、指定されたサーバ名に対応するサーバ情報として扱われます。
[SQL_DEFAULT_INF]	デフォルトコネクション情報の定義開始を示すセクション名です。[SQL_DEFAULT_INF]は、固定文字です。
[CONNECTION_SCOPE]	コネクション有効範囲の定義開始を示すセクション名です。[CONNECTION_SCOPE]は、固定文字です。

パスワードを ODBC 情報ファイルに設定する場合、ファイルのセキュリティには十分注意してください。パスワードを ODBC 情報ファイルに設定しないで、アプリケーションの実行時にパスワードを入力するなど、アプリケーションで解決する方法もあります。

ODBC 情報設定ユーティリティの使い方

この機能は互換のために残されています。NetCOBOL for .NET では、ODBC 情報ファイルおよび ODBC 情報設定ユーティリティを使用することは推奨していません。アプリケーション構成ファイルを使用した SQL 情報の設定をお勧めします。

ODBC 情報設定ユーティリティは、ODBC 接続のデータベースアクセスに必要となる情報を、ODBC 情報ファイルに設定するためのユーティリティです。

ODBC 情報設定ユーティリティには、以下の機能があります。

- ODBC 情報ファイルの選択
- サーバ情報の設定
- デフォルトコネクション情報の設定
- コネクション有効範囲の設定

ODBC 情報設定ユーティリティでは、SQL オプション情報を設定することはできません。

以下に ODBC 情報設定ユーティリティの使い方を示します。詳細については、ODBC 情報設定ユーティリティを参照してください。

1. ODBC 情報設定ユーティリティの起動

   ODBC 情報設定ユーティリティ(F5FHSQL.EXE)を起動します。

2. ODBC 情報ファイルの選択

   情報を設定する ODBC 情報ファイルを指定します。 指定されたファイルが存在しない場合、新規に作成されます。

3. サーバ情報の設定

   CONNECT 文またはデフォルトコネクション情報で、指定されたサーバ名に対する情報の設定を行います。

4. デフォルトコネクション情報の設定

   DEFAULT 指定の CONNECT 文が記述された場合に、使用するデフォルトコネクション情報(サーバ名、ユーザ ID およびパスワード)を設定します。

5. コネクション有効範囲の設定

   コネクションの有効範囲を設定します。

NCOBU00379

ODBC 情報ファイルの設定内容の最大長

この機能は互換のために残されています。NetCOBOL for .NET では、ODBC 情報ファイルおよび ODBC 情報設定ユーティリティを使用することは推奨していません。アプリケーション構成ファイルを使用した SQL 情報の設定をお勧めします。

ODBC 情報ファイルに設定する内容の最大長を以下に示します。

表: ODBC 情報ファイルに設定する内容の最大長

情報の種別	最大長	備考
ユーザ ID	32 バイト	(注 1)
パスワード	32 バイト	
サーバ名	32 バイト	−
データソース名	32 バイト	−

注 1: コネクションを確立するデータソースの仕様により異なります。したがって、ODBC ドライバと関係する環境のマニュアルを参照してください。

パスワードは、ODBC 情報設定ユーティリティを使用して、暗号化して設定してください。エディターでは、直接編集しないでください。

ADO.NET データプロバイダー使用時の注意事項

ここでは、ADO.NET データプロバイダー使用時の注意事項について記述しています。すでに説明していることも含まれますが、重要な情報ですので必ずお読みください。

このセクションの内容

SQL 文の文法上の制限事項

SQL 文の文法上の制限事項について説明します。

埋込み SQL 文の実行時の注意事項

埋込み SQL 文の実行時の注意事項について説明します。

ADO.NET データプロバイダーの留意事項

使用できるデータプロバイダーについて説明します。

埋込み SQL 文の実行時の定量制限

埋込み SQL 文の実行時の定量制限について説明します。

分散トランザクションを併用する場合の注意事項

分散トランザクションを併用する場合の注意事項について説明します。

NCOBU00852

SQL 文の文法上の制限事項

データ部

- COBOL では、ADO.NET のデータ型とホスト変数のデータ型の対応付けを規定しています。 ホスト変数を使用する場合は、ADO.NET データプロバイダーの規定するデータ型に対応するホスト変数を使用してください。 規定外の対応付けを行った場合は、データの内容は保証されません。 [参照]ADO.NET で扱うデータ型との対応
- ADO.NET データプロバイダーの仕様によっては、あるデータ型が、COBOL のホスト変数で扱えない場合もあります。

手続き部

- ADO.NET データプロバイダーでは、カーソルの位置付け更新を行う場合、DECLARE CURSOR 宣言の SELECT 文に、PRIMARY KEY または ALLOW NULL の存在しない UNIQUE KEY の指定が必要になります。
- SQL CLR ストアドプロシージャを CALL 文で使用する場合、 処理された行の件数は SQLERRD(3)に格納されません。
- 複数テーブルで作成したカーソルは、更新できません。
- ADO.NET データプロバイダーによって、使用できる埋め込み SQL 文やその指定方法が異なります。COBOL ソースプログラム中に埋込み SQL 文を記述する場合は、COBOL の仕様だけでなく、 各データプロバイダーとデータベースドライバを確認してください。また、データベース管理システムの関連マニュアルを参照してください。
- @SQL COMMIT MODE (COMMIT モードの指定)に MANUAL を指定し、COMMIT 文または ROLLBACK 文を使用してトランザクションを終了しない場合、動作を保証できないことがあります。また、DISCONNECT 文でコネクションを切断する前には、必ず COMMIT 文または ROLLBACK 文を記述してトランザクションを終了してください。
- SQL 記述子域は使用できません。したがって、以下の埋込み SQL 文または指定は記述できません。
  - 記述子名を指定した INTO 句および USING 句
  - ALLOCATE DESCRIPTOR 文、DEALLOCATE DESCRIPTOR 文、SET DESCRIPTOR 文、DESCRIBE 文などの記述子域に関係する SQL 文
- DEALLOCATE PREPARE 文は記述できません。
- 定義系(DDL 系)SQL 文は記述できません。
- 動的 SQL 文である PREPARE/EXECUTE 文、EXECUTE IMMEDIATE 文を使用する場合、埋込み例外宣言の NOT FOUND 指定は以下の SQL 文に対してだけ有効です。
  - SELECT 文 (PREPARE/EXECUTE 文の場合だけ)
  - UPDATE 文 (探索)
  - UPDATE 文 (位置付け)
  - DELETE 文 (探索)
  - DELETE 文 (位置付け)
  - INSERT 文
  - 動的 UPDATE 文(探索)
  - 動的 UPDATE 文(位置付け)

- 動的 DELETE 文(探索)
- 動的 DELETE 文(位置付け)

- 動的カーソル宣言で使用する文識別子と同名の文識別子を EXECUTE 文で指定することはできません。または、EXECUTE 文に指定する文識別子と同名の文識別子を動的カーソル宣言で使用することはできません。
- 埋込み SQL 文の文法の詳細は、各データベースおよびデータベース関連製品の仕様に従います。
- スキーマが付与されているテーブルに対するカーソルの位置付け更新はサポートしていません。

NCOBU00853

埋込み SQL 文の実行時の注意事項

- NetCOBOL for .NET から ADO.NET データプロバイダーの環境を使用してデータベースにアクセスする場合、アクセス対象のデータベースのための ADO.NET データプロバイダーおよび ADO.NET データプロバイダーが必要とするドライバなどの環境を構築する必要があります。
- ADO.NET データプロバイダーによって、使用できる埋込み SQL 文やその指定方法が異なります。COBOL ソースプログラム中に埋込み SQL 文を記述する場合は、COBOL の仕様だけではなく、各 ADO.NET データプロバイダーとデータベースドライバを確認してください。また、データベース管理システムの関連マニュアルを参照してください。
- データ操作系以外の埋込み SQL 文を動的に実行する場合、ADO.NET データプロバイダーによっては以下の現象が発生することがあります。データ操作文以外の埋込み SQL 文を動的に実行しないようにしてください。

```
現象:
  SQLSTATE : 02000
  SQLCODE : +100
  SQLMSG : データがありません。
```

- @SQL_COMMIT_MODE (COMMIT モードの指定)に MANUAL を指定し、COMMIT 文または ROLLBACK 文を使用してトランザクションを終了しない場合、動作の保証はできません。 ADO.NET データプロバイダーによっては、更新したデータがデータベースに反映されないことがあります。DISCONNECT 文でコネクションを切断する前には、必ず COMMIT 文または ROLLBACK 文を記述してトランザクションを終了してください。
- @SQL_COMMIT_MODE (COMMIT モードの指定)に MANUAL を指定し、トランザクションが COMMIT 文または ROLLBACK 文で終了した場合、該当するコネクションでオープンしているカーソルは暗黙的に全てクローズされます。
- 文字型のデータ中に値として X'00'が格納されている場合、その格納結果および取出し結果に対する保証はできません。
- 浮動小数点型のデータを使用したデータ操作を行う場合、サーバ側とクライアント側でデータの表現方法が異なるため、変換誤差が発生し、期待した結果が得られない場合があります。
- 埋込み SQL 文の動作の詳細は、各データベースおよびデータベース関連製品の仕様に従います。
- SQLSTATE/SQLCODE/SQLMSG の値は、使用するデータプロバイダー、データベースによって異なります。
- @SQL_QUERY_TIMEOUT (タイムアウト時間の指定)を設定しない場合、タイムアウト値は、使用する ADO.NET データプロバイダーごとのデフォルト値に従います。
- ADO.NET は､サーバカーソルをサポートしていません。COBOL プログラムに記述されたカーソルは、クライアントカーソルとしてエミュレートされます。 サーバカーソルを利用したい場合は、ODBC 接続によるデータベースアクセス機能を使用してください。
- FETCH 文、FETCH NEXT 文、FETCH PRIOR 文、FETCH FIRST 文および FIRST LAST 文は、カーソルのオープン時点でのデータを返します。
- カーソル系データ操作文の UPDATE 文(位置付け)、DELETE 文(位置付け)の対象となるカーソルには、一意なインデックスが１つ必要です。また、データベースによっては、複合キー中に NULL 値が存在している場合や、複数キーを使用している場合は、カーソルの更新ができないことがあります。
- カーソル宣言で FOR UPDATE 句が指定されていないカーソルを更新可能カーソルとして使用する場合には、@SQL_ADONET_CURSOR_DEFAULT(カーソル動作省略時のカーソル動作の指定)に READ_WRITE を設定してください。

- カーソル宣言で FOR UPDATE 句が指定されるか、@SQL_ADONET_CURSOR_DEFAULT(カーソル動作省略時のカーソル動作の指定)で READ_WRITE が設定された場合、カーソルはクライアント側に結果セットを持ちます。クライアント側では結果セットのサイズ分のメモリを消費するため、ADO.NET でカーソルを使用する場合は、SELECT 文で結果セットのサイズを絞り込むようにしてください。

- SQL Server ではカーソル宣言で FOR UPDATE 句が指定されたカーソルを使用した場合、以下のメッセージを表示することがあります。

```
現象:
  SQLSTATE : 9999F
  SQLCODE  : -000001003
  SQLMSG   : SQL 実行時に例外が発生しました. '行 1: FOR UPDATE 句は
  DECLARE CURSOR だけに許可されます。'
```

  このメッセージが表示された場合、カーソル宣言で FOR UPDATE 句を削除するか、@SQL_ADONET_FORUPDATE_CLAUSE (カーソル宣言の FOR UPDATE 句の無効化の指定)に DISABLE を設定してください。

- カーソルはクライアント側に結果セットを持ちます。このため、オプティミスティック同時実行制御を行います。FETCH 文で位置付けた行の更新時に、該当行の値がカーソルオープン時の状態のままである場合にだけ行が更新されます。行が変更されていた場合には、以下のメッセージを表示します。

```
現象:
  SQLSTATE : 999SK
  SQLCODE  : -999999006
  SQLMSG   : 同時実行違反：処理予定の 1 レコードのうち 0 件が処理されました
```

オプティミスティック同時実行制御については、.NET Framework のドキュメントのオプティミスティック同時実行制御を参照してください。

- データ操作の実行中に以下のメッセージが発生した場合は、開いているカーソルをクローズするか、MARS 機能を使用してください。

```
現象:
  SQLSTATE : 9999F
  SQLCODE  : -999999993
  SQLMSG   : SQL 実行時に例外が発生しました. 'このコマンドに関連付けられている
  DataReader が既に開かれています。このコマンドを最初に閉じる必要があります。'
```

MARS 機能については、SQL Server オンラインブックの複数のアクティブな結果セット (MARS)を参照してください。

- 動的 SQL 文による UPDATE 文(位置付け)、DELETE 文(位置付け)は、@SQL_CONNECTION_SCOPE (コネクションの有効範囲の指定)に OBJECT_INSTANCE または、THREAD が指定された場合に使用することができます。

翻訳オプション SQLSCOPE(METHOD)が指定されている場合は使用できません。

- UPDATE 文(位置付け)、DELETE 文(位置付け)を同じレコードに対して連続して実行できません。
- FETCH 文、FETCH NEXT 文、FETCH PRIOR 文、FETCH FIRST 文および FIRST LAST 文を、FOR 句、複数行指定ホスト変数または表指定ホスト変数を使用してデータを取得した後、そのカーソルを使用して UPDATE 文(位置付け)または DELETE 文(位置付け)を実行する場合、取得したデータのうち最終行が更新対象になります。
- 読み取り専用カーソルを使用して、FETCH PRIOR 文、FETCH FIRST 文および FIRST LAST 文を実行することはできません。
- SQLServer で datetime 型のデータを ODBC 経由の結果と同一の形式で取得する場合には、@SQL_ADONET_DATETIME_FORMAT に"yyyy-MM-dd hh:mm:ss.fff"を指定してください。

ADO.NET データプロバイダーの留意事項

ここでは、使用するデータプロバイダーの留意事項について示します。

- データプロバイダーは、アプリケーション構成ファイルの接続文字列情報のプロバイダ名に指定します。プロバイダ名の指定方法は、接続文字列の設定方法(ADO.NET)を参照してください。
- コネクションごとに異なるデータプロバイダーの指定ができます。
- MARS 機能が使用できるかどうかは、データベースとデータプロバイダーによります。MARS 機能を使用するには、使用するデータベースのマニュアルを確認し、接続文字列に指定してください。
- SQLSTATE/SQLCODE/SQLMSG の値は、使用するデータプロバイダー、データベースによって異なります。
- .NET Data Provider for ODBC を使用し、かつ接続文字列に DSN(データソース名)を指定する場合は、別途 ODBC データソースの設定が必要になります。ODBC データソースの設定については、連携ソフトウェアおよびハードウェア環境の整備を参照してください。

- ADO.NET データプロバイダーについては、.NET Framework のドキュメントの.NET Framework データ プロバイダーを参照してください。
- MARS 機能については、.NET Framework のドキュメントの複数のアクティブな結果セット(MARS)を参照してください。

NCOBU00855

埋込み SQL 文の実行時の定量制限

ここでは、埋込み SQL 文の実行時の定量制限について説明します。

- SQLMSG に設定されるデータソースのメッセージ文字列の最大長は、1024 バイトです。最大長を超えたメッセージ文字列は、切り捨てられます。

分散トランザクションを併用する場合の注意事項

分散トランザクションを併用する場合の注意点を以下に示します。

SystemTransactions.TransactionScope を使用した例を以下に示します。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. MAIN AS "COBOLTransactionTest.Main".
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
          CLASS CLASS-TRANSACTIONSCOPE AS "System.Transactions.TransactionScope".
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 TRANSCOPE OBJECT REFERENCE CLASS-TRANSACTIONSCOPE.
      01 ERRFLG      PIC X(1).
          EXEC SQL BEGIN DECLARE SECTION END-EXEC.
      01 SQLSTATE    PIC X(5).
      01 SQLMSG      PIC X(128).
          EXEC SQL END DECLARE SECTION END-EXEC.
      PROCEDURE DIVISION.
          MOVE "N"          TO ERRFLG.
      *TransactionScope（トランザクション）の開始
          TRY                                                                  *>[1]
             SET TRANSCOPE  TO CLASS-TRANSACTIONSCOPE::"NEW"()                 *>[2]
      *サンプルデータベースへ接続する
             EXEC SQL CONNECT TO DEFAULT END-EXEC                              *>[3]
             IF SQLSTATE NOT = "00000" THEN
                DISPLAY "データベースへ接続時にエラーが発生:", SQLMSG
                MOVE "Y"      TO ERRFLG
             END-IF
      *STOCK 表から製品番号 110 の倉庫番号を 3 に変更する
             EXEC SQL UPDATE STOCK SET WHNO = 3 WHERE GNO = 110 END-EXEC        *>[4]
             IF SQLSTATE NOT = "00000" THEN
                DISPLAY "データベースを更新時にエラーが発生:", SQLMSG
                MOVE "Y"      TO ERRFLG
             END-IF
             EXEC SQL DISCONNECT DEFAULT END-EXEC                              *>[5]
             IF ERRFLG = "N" THEN
                INVOKE TRANSCOPE "Complete"                                    *>[6]
             END-IF
          FINALLY
             INVOKE TRANSCOPE "Dispose"                                        *>[7]
          END-TRY.
          STOP RUN.
      END PROGRAM MAIN.
```

[1] : トランザクションを有効化したい範囲を TRY～FINALLY 句で区切ります。

[2] : SystemTransactions.TransactionScope クラスを New メソッドでインスタンス化することで、トランザクションを開始します。

[3]: データベースへ接続します。@SQL_COMMIT_MODE は必ず AUTO または ENLIST を指定してください。

[4] : UPDATE 文でデータベースを更新します。

[5] : データベースから切断します。

[6] : TransactionScope インスタンスの Complete メソッドを実行することで、トランザクションを確定します。ROLLBACK が必要な場合には、Complete メソッドを実行しないようにします。

[7] : TransactionScope インスタンスを Dispose メソッドで解放することでトランザクションが確定します。

System.Transactions 名前空間の使用については、.NET Framework のドキュメントの SQL Server と System.Transactions の統合を参照してください。

[注意]

分散トランザクションへの参加はデータプロバイダーに依存します。(各データベース・データプロバイダーの仕様を確認してください。)

ODBC ドライバ使用時の注意事項

ここでは、ODBC ドライバ使用時の注意事項について記述しています。すでに説明していることも含まれていますが、重要な情報ですので必ずお読みください。

このセクションの内容

NCOBU00383

SQL 文の文法上の制限事項

データ部

- COBOL では、ODBC のデータ型とホスト変数のデータ型の対応付けを規定しています。 ホスト変数を使用する場合は、ODBC ドライバの規定するデータ型に対応するホスト変数を使用してください。 規定外の対応付けを行った場合は、データの内容は保証されません。 [参照]ODBC で扱うデータとの対応
- ODBC のドライバの仕様によっては、あるデータ型が、COBOL のホスト変数で扱えない場合もあります。

手続き部

- ODBC ドライバによって、使用できる埋込み SQL 文やその指定方法が異なります。 COBOL ソースプログラム中に埋込み SQL 文を記述する場合は、COBOL の仕様だけでなく、ODBC ドライバの規定する SQL 文を確認し、ODBC ドライバに関係するソフトウェア/ハードウェア、およびデータベース管理システムの関連マニュアルも参照してください。
- @SQL_COMMIT_MODE (COMMIT モードの指定)に MANUAL を指定し、COMMIT 文または ROLLBACK 文を使用してトランザクションを終了しない場合、動作を保証できないことがあります。また、DISCONNECT 文でコネクションを切断する前には、必ず COMMIT 文または ROLLBACK 文を記述してトランザクションを終了してください。
- SQL 記述子域は使用できません。したがって、以下の埋込み SQL 文または指定は記述できません。
  - 記述子名を指定した INTO 句および USING 句
  - ALLOCATE DESCRIPTOR 文、DEALLOCATE DESCRIPTOR 文、GET DESCRIPTOR 文、SET DESCRIPTOR 文、DESCRIBE 文などの記述子域に関係する SQL 文
- DEALLOCATE PREPARE 文は記述できません。
- 定義系(DDL 系)SQL 文は記述できません。
- 動的 SQL 文である PREPARE/EXECUTE 文、EXECUTE IMMEDIATE 文を使用する場合、埋込み例外宣言の NOT FOUND 指定は以下の SQL 文に対してだけ有効です。
  - SELECT 文 (PREPARE/EXECUTE 文の場合だけ)
  - UPDATE 文 (探索)
  - UPDATE 文 (位置付け)
  - DELETE 文 (探索)
  - DELETE 文 (位置付け)
  - INSERT 文
  - 動的 UPDATE 文(探索)
  - 動的 UPDATE 文(位置付け)
  - 動的 DELETE 文(探索)
  - 動的 DELETE 文(位置付け)
- 動的カーソル宣言で使用する文識別子と同名の文識別子を EXECUTE 文で指定することはできません。または、EXECUTE 文に指定する文識別子と同名の文識別子を動的カーソル宣言で使用することはできません。
- 埋込み SQL 文の文法の詳細は、各データベースおよびデータベース関連製品の仕様に従います。

埋込み SQL 文の実行時の注意事項

- NetCOBOL for .NET から ODBC の環境を使用してデータベースにアクセスする場合、アクセス対象のデータベースのための ODBC ドライバおよび ODBC ドライバが必要とする環境を構築する必要があります。

- ODBC ドライバによって、使用できる埋込み SQL 文やその指定方法が異なります。

  COBOL ソースプログラム中に埋込み SQL 文を記述する場合は、COBOL の仕様だけではなく、ODBC ドライバの規定する SQL 文を確認し、ODBC ドライバに関係するソフトウェア/ハードウェア、およびデータベース管理システムのマニュアルも参照してください。

- 埋込み SQL 文の記述、ODBC 情報ファイルの指定、データソースの設定によって、ODBC 環境のオプション値が変更されることがあります。

  この場合、埋込み SQL 文が実行される時に各 ODBC ドライバから通知メッセージが出力されます。埋込み例外宣言を使用する場合は、エラーが発生したものとして処理されますので、必要に応じてプログラムを変更してください。

- FOR 句、複数行指定ホスト変数または、表指定ホスト変数を使用する場合は、使用する ODBC ドライバのマニュアルを参照し、複数行操作をサポートするか確認してください。複数行操作をサポートしていない場合は、@SQL_MULTIPLE_ROWS (複数行操作のサポートの指定)を OFF に設定します。

  @SQL_MULTIPLE_ROWS を OFF に設定した場合、FOR 句で 1 より大きい値を指定しているときや、複数行指定ホスト変数または、表指定ホスト変数で OCCURS 句に 1 より大きい値を指定しているときは、以下の実行時エラーが発生します。

```
現象:
  SQLSTATE : 9999G
  SQLCODE : -999999900
  SQLMSG : FOR 句、複数行指定ホスト変数または、表指定ホスト変数は使用できません
```

- データ操作文以外の埋込み SQL 文を動的に実行する場合､ODBC ドライバによっては以下の現象が発生することがあります。データ操作文以外の埋込み SQL 文を動的に実行しないようにしてください。

```
現象:
  SQLSTATE : 02000
  SQLCODE : +100
  SQLMSG : データがありません。
```

- @SQL_COMMIT_MODE (COMMIT モードの指定)に MANUAL を指定し、COMMIT 文または ROLLBACK 文を使用してトランザクションを終了しない場合、動作を保証できないことがあります。また、DISCONNECT 文でコネクションを切断する前には、必ず COMMIT 文または ROLLBACK 文を記述してトランザクションを終了してください。

- 文字型のデータ中に値として X'00'が格納されている場合、その格納結果および取出し結果に対する保証はできません。

- 浮動小数型のデータを使用したデータ操作を行う場合、サーバ側とクライアント側でデータの表現方法が異なるため、変換誤差が発生し、期待した結果が得られない場合があります。

- 埋込み SQL 文の動作の詳細は、各データベースおよびデータベース関連製品の仕様に従います。

- 動的 SQL 文による UPDATE 文(位置付け)、DELETE 文(位置付け)は、@SQL_CONNECTION_SCOPE (コネクションの有効範囲の指定)に OBJECT_INSTANCE または、THREAD が指定された場合に使用することができます。

翻訳オプション SQLSCOPE(METHOD)が指定されている場合は使用できません。

- @SQL_ROW_ARRAY_SIZE に 1 より大きい値を設定し、カーソルのデータをメモリにキャッシュしている状態で FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文を実行した場合、以下のようなエラーが発生します。@SQL_ROW_ARRAY_SIZE に 1 を指定するか、オプションを指定しないでください。

```
現象:
  SQLSTATE : 999SL
  SQLCODE : -999999007
  SQLMSG : カーソルに処理できないオプションが指定されました
```

- @SQL_CURSOR_TYPE(カーソルタイプ種別の指定)に FORWARD_ONLY を指定している、または省略している場合、FETCH PRIOR 文、FETCH FIRST 文、または FETCH LAST 文を実行すると以下のようなエラーが発生しますが、エラーの内容はデータベースにより異なります。FETCH PRIOR 文、FETCH FIRST 文、または FETCH LAST 文を実行するには、@SQL_CURSOR_TYPE に STATIC、KEYSET_DRIVEN、または DYNAMIC のいずれかを指定してください。カーソル種別は、データソース（ODBC ドライバ、データベース、データベース関連製品）に依存します。使用するデータベースに該当するカーソルタイプがあるか確認してください。

```
現象:
  SQLSTATE : S1106
  SQLCODE : +0
  SQLMSG : フェッチの型が範囲を超えています。
```

各 ODBC ドライバ固有の留意事項

ここでは、各 ODBC ドライバ固有の留意事項について説明します。

Symfoware Server (Postgres)、FUJISTU Enterprise Postgres 使用時の留意事項

- カーソルの位置づけ更新はサポートされていません。
- 更新処理時の SQLERRD(3)による処理行数はサポートされていません。常に 1 となります。
- FETCH PRIOR でカーソルの先頭に移動後に、FETCH NEXT を実行する動作は保証されません。

Oracle ODBC ドライバ使用時の留意事項

CALL 文で、以下を呼び出すことはできません。

- ファンクション
- パッケージ化されたストアドプロシージャ
- パッケージ化されたファンクション

Microsoft SQL Server Native Client ODBC ドライバ使用時の留意事項

- CONNECT 文の実行時に、通知メッセージが出力されますが、正常に接続されています。以下は、SQL Server ODBC ドライバでの通知メッセージの例です。

```
現象:
  SQLSTATE: 01000
  SQLCODE: +5701
  SQLMSG: (SQL server)データベースのコンテキストを 'データベース名' に変更しました。
```

  埋込み例外宣言が CONNECT 文より前に記述されている場合、エラーが発生したものとして処理されますので、埋込み例外宣言は CONNECT 文より後ろに記述してください。

- カーソルが位置付け UPDATE 文か位置付け DELETE 文を実行できるかどうかは、@SQL_CONCURRENCY (カーソルの同時実行の指定)と@SQL_CURSOR_TYPE (カーソル種別の指定)の組み合わせにより決まります。プログラムに適したカーソルの属性にするために SQL オプション情報を正しく設定してください。
- カーソル系データ操作文の位置付け UPDATE 文、位置付け DELETE 文の対象となる表では、1 つ以上の一意なインデックスが存在しなければならない場合があります。
- カーソル宣言に FOR UPDATE 句を指定した場合、@SQL_CONCURRENCY (カーソルの同時実行の指定)と@SQL_CURSOR_TYPE (カーソル種別の指定)を正しく設定してください。
- データを更新しないカーソルで以下のオプションを指定すると、性能の向上が期待できます。
  - @SQLCONCURRENCY = READ_ONLY
  - @SQL_CURSOR_TYPE = FORWARD_ONLY
- 1 つのコネクション接続で、カーソルの既定の結果セットを利用中に SQL 文を実行した場合、接続エラーが発生します。このエラーは、MARS 機能を有効にするか、カーソルタイプをサーバカーソルに変更することで回避できます。サーバカーソルへの変更は、@SQL_CONCURRENCY (カーソルの同時実行の指定)と@SQL_CURSOR_TYPE (カーソル種別の指定)によって変更できます。デフォルトでは、この値は、@SQL_CONCURRENCY が READ_ONLY で、@SQL_CURSOR_TYPE が FORWARD_ONLY になっているため、既定の結果セットになっています。

- MARS 機能については、SQL Server オンラインブックの複数のアクティブな結果セット (MARS)を参照してください。
- 既定の結果セットについては、SQL Server オンラインブックの SQL Server の既定の結果セットの使用を参照してください。

埋込み SQL 文の実行時の定量制限

ここでは、埋込み SQL 文の実行時の定量制限について説明します。

- 埋込み SQL 文の最大長は、16384 バイトです。ただし、これは、ODBC ドライバと関係する環境によって制限を受けることがあります。たとえば、ネットワーク部分を受け持つソフトウェアのデータ転送の最大長によって制限を受ける場合があります。

  また、ODBC ドライバが COBOL から受け渡された埋込み SQL 文を加工している場合は、COBOL ソースプログラムの記述よって実際の埋込み SQL 文が長くなる場合もあります。

- クライアントとサーバ間で入出力できる領域長に制限はありませんが、データソース(ODBC ドライバ、データベース、データベース関連製品)の環境によって制限を受けることがあります。

- SQLMSG に設定されるデータソースのメッセージ文字列の最大長は、1024 バイトです。最大長を超えたメッセージ文字列は、切り捨てられます。

NCOBU01121

Azure SQL データベース使用時の注意事項

Azure SQL データベースは、Microsoft Azure Platform で動作するリレーショナルデータベースサービスです。Azure SQL データベースの大きな特長は、SQL Server との高い互換性です。 しかし、スペックや機能面全てが等価というわけではありません。 既存の SQL Server アプリケーションを Azure SQL データベースアプリケーションに移行する場合や SQL Azure アプリケーションを開発する場合には、事前の調査と機能検証を行ってください。

Azure SQL データベースの詳細や SQL Server との機能差については、以下を確認することをお勧めします。

- MSDN ライブラリの Azure SQL データベース
- MSDN ライブラリの Azure SQL データベースの一般的なガイドラインと制限事項

このセクションの内容

Azure SQL データベースアプリケーションの注意事項

SQL Azure アプリケーションの注意事項について説明します。

Azure SQL データベースへアクセスする際の制限事項

SQL Azure へアクセスする時の制限事項について説明します。

Azure SQL データベースアプリケーションの注意事項

ここでは、Azure SQL データベースアプリケーションの注意点について説明します。

- Azure SQL データベースを使用するためには、Azure SQL データベース ファイアウォールの設定が必要です。 開始および終了アドレスが 0.0.0.0 のファイアウォール設定は、 Windows Azure 接続が許可されています。 コンピューティング エミュレーターによるデバッグまたはオンプレミスのアプリケーションから Azure SQL データベースを使用するには、要求の発生元の IP アドレスを許可する必要があります。 ファイアウォールを構成するには、Azure SQL データベース ポータルで許容される IP アドレスの範囲を指定します。 詳細については、MSDN ライブラリの Azure SQL データベース ファイアウォールを参照してください。

- 日本語文字列定数には N プレフィックスが必須です。 埋め込み SQL 文の中で日本語文字列定数または Unicode 文字列定数を利用する場合、N プレフィックスを付与してください。正しく文字認識がされず、文字化けの原因になります。

```
EXEC SQL
INSERT INTO STOCK (GNO,GOODS, QOH, WHNO)
VALUES (301, N'洗濯機', 50, 1)
END-EXEC.
```

- Azure SQL データベースの既定の照合順序は"SQL_Latin1_General_CP1_CI_AS"となっています。 char 型または varchar 型のテーブルに、以下に示す文字以外を設定する場合は、COLLATE 句により照合順序を指定してください。

  - a、b、C などの英大文字と英小文字
  - 1、2、3 などの数字
  - アットマーク、アンパサンド、感嘆符などの特殊記号

  詳細については、MSDN ライブラリの COLLATE(Transact-SQL)を参照してください。

NCOBU01123

Azure SQL データベースへアクセスする際の制限事項

- ストアドプロシージャ定義で指定された char 型および varchar 型のパラメタに、ASCII 範囲以外の文字を設定することはできません。ASCII 範囲以外の文字を設定する場合は、nchar 型または nvarchar 型でパラメタを定義してください。

マルチスレッドアプリケーション

ここでは、NetCOBOL for .NET によるマルチスレッドアプリケーションの作成方法とマルチスレッドの使用方法について説明します。

このセクションの内容

マルチスレッドプログラムの基本動作

マルチスレッドプログラムでのデータ領域の管理のされ方ついて説明します。

スレッド間の共有資源

スレッド間の資源を共有する方法について説明します。

マルチスレッドの使用方法(基本編)

マルチスレッド機能の基本的な使用方法について説明します。

マルチスレッドの使用方法(応用編)

マルチスレッド機能のより高度な使用方法について説明します。

スレッド同期制御サブルーチン

スレッドの同期制御を行うためのサブルーチンについて説明します。

NCOBU00388

マルチスレッドプログラムの基本動作

マルチスレッドプログラムのデータの扱い

ここでは、マルチスレッドプログラムでのデータ領域の管理のされ方について説明します。

マルチスレッドプログラムには、アプリケーションドメイン、スレッドおよび呼出し(呼び出されてから復帰まで)単位で確保/管理されるデータがあります。

アプリケーションドメイン単位で確保/管理されるデータ

- スレッド間共有外部データと外部ファイル(注 1)
- スタティック定義(スタティック定義とオブジェクトインスタンス)
- オブジェクトインスタンス(スタティック定義とオブジェクトインスタンス)

スレッド単位で確保/管理されるデータ

- プログラム定義に宣言されたデータ(注 2)

呼出し単位で確保/管理されるデータ

- メソッド定義に宣言されたデータ

注 1: 翻訳オプション SHREXT を指定して翻訳された COBOL ソースプログラム中の EXTERNAL 句が指定されたデータまたはファイルを指します。

注 2: スレッド間共有外部データとスレッド間共有外部ファイルを除きます。

プログラム定義に宣言されたデータ

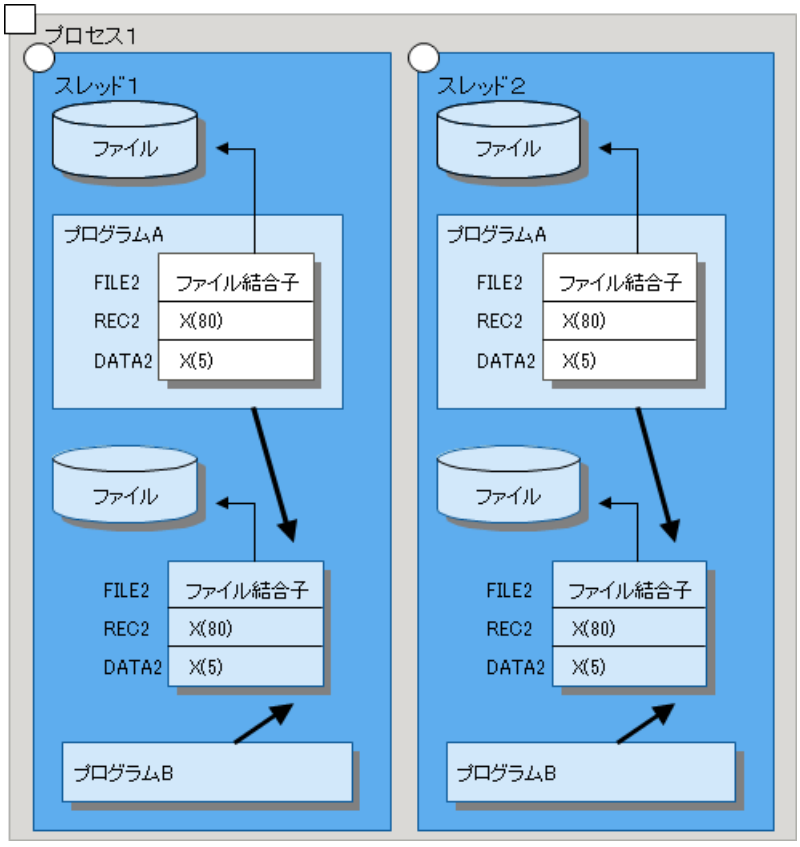
プログラム定義に宣言されたデータはスレッドごとに確保されます。

プログラム定義に宣言されたデータとファイル

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. A.
*   :
 DATA DIVISION.
 FILE SECTION.
 FD FILE1 EXTERNAL.
 01 REC1  PIC X(80).
 FD FILE2.
 01 REC2  PIC X(80).
*   :
 WORKING-STORAGE SECTION.
 01 DATA1 PIC X(5) EXTERNAL.
 01 DATA2 PIC X(5).
*   :
 PROCEDURE DIVISION.
*   :
      CALL "B".
*   :
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. B.
*   :
 DATA DIVISION.
 FILE SECTION.
 FD FILE1 EXTERNAL.
 01 REC1  PIC X(80).
*   :
 WORKING-STORAGE SECTION.
 01 DATA1 PIC X(5) EXTERNAL.
*   :
 PROCEDURE DIVISION.
*   :
```

以下の図は、上記のプログラムが 2 つ起動された場合を表しています。マルチスレッドプログラムでは 2 つのスレッドが起動されています。図から分かるように、プログラムで定義したデータやファイルは、スレッドごとに確保されます。



NCOBU00390

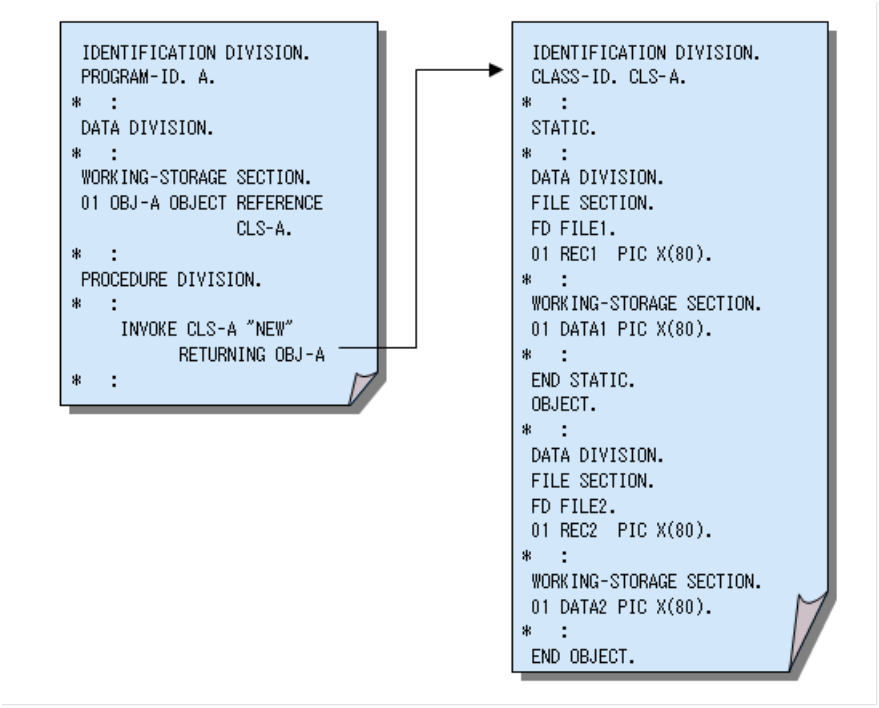
スタティック定義とオブジェクトインスタンス

スタティック定義とオブジェクトインスタンスはアプリケーションドメインで管理されます。

各クラスのスタティック定義はプロセスに 1 つだけ存在するため、マルチスレッドプログラムでは、つねにスレッド間で共有されます。

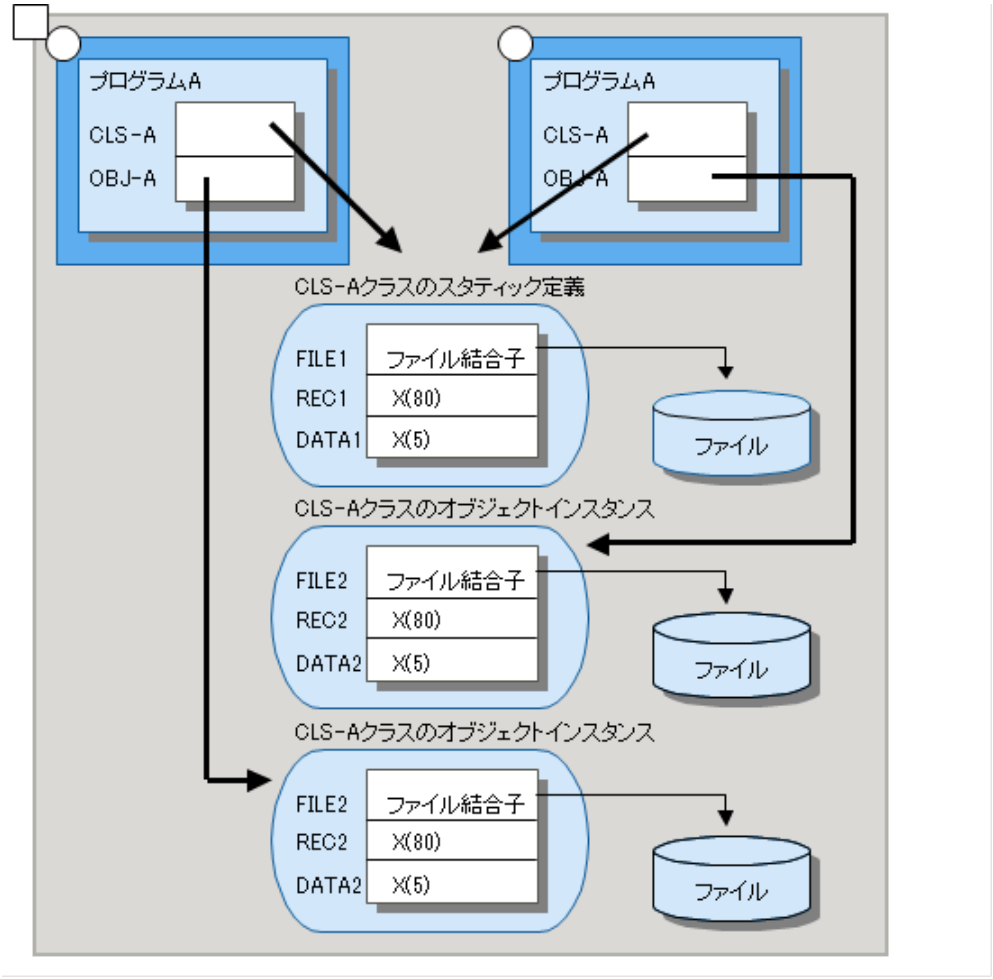
オブジェクトインスタンスもスタティック定義を介することによってスレッド間で共有することができます。[参照]オブジェクトインスタンスの操作

スタティック定義とオブジェクト定義に宣言されたデータとファイル



以下の図は、上記のプログラムが 2 つ起動された場合を表しています。 マルチスレッドプログラムでは 2 つのスレッドが起動されています。

図から分かるように、マルチスレッドプログラムでは、オブジェクトはプロセスで管理されます。このため、マルチスレッドプログラムでは、スタティック定義がつねにスレッド間で共有されることになります。


プログラムA
CLS-A
OBJ-A
プログラムA
CLS-A
OBJ-A
CLS-Aクラスのスタティック定義
FILE1
ファイル結合子
REC1
X(80)
DATA1
X(5)
ファイル
CLS-Aクラスのオブジェクトインスタンス
FILE2
ファイル結合子
REC2
X(80)
DATA2
X(5)
ファイル
CLS-Aクラスのオブジェクトインスタンス
FILE2
ファイル結合子
REC2
X(80)
DATA2
X(5)
ファイル

NCOBU00391

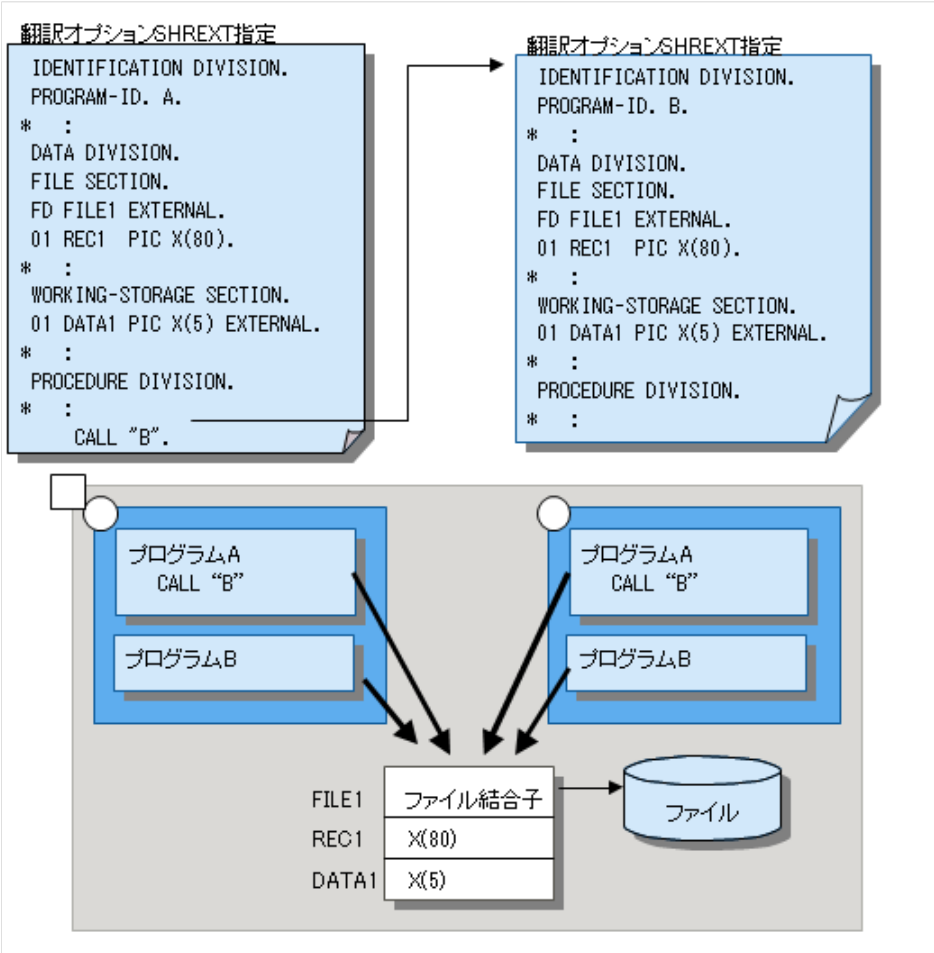
メソッド定義に宣言されたデータ

メソッド定義に宣言されたデータの割り付けは、メソッドの呼出し時に確保され、そのメソッドの呼出し元に戻るときに解放されます。メソッドの呼出し元に戻るとき、クローズされていないファイルは強制的にクローズされます。

スレッド間共有外部データと外部ファイル

データ記述項またはファイル記述項に EXTERNAL 句を指定し、翻訳オプション SHREXT を指定して翻訳することにより、スレッド間で共通のデータ領域を使用することができます。

以下に、データとファイルをスレッド間で共有した場合のデータ領域の持ち方を示します。この図は、マルチスレッドプログラムが 2 つのスレッドで実行されているところを表しています。



NCOBU00393

スレッド間の共有資源

COBOL のマルチスレッドプログラムでは、スレッド間でデータやファイルなどの資源を共有するマルチスレッドの特性を活かしたプログラムを簡単に作成できます。

このセクションの内容

競合状態

スレッド間で資源を共有する場合に、一般的に発生する競合状態について説明します。

資源の共有

スレッド間で共有することができる資源について説明します。

競合状態

ここでは、スレッド間で資源を共有する場合に、一般的に発生する競合状態について説明します。

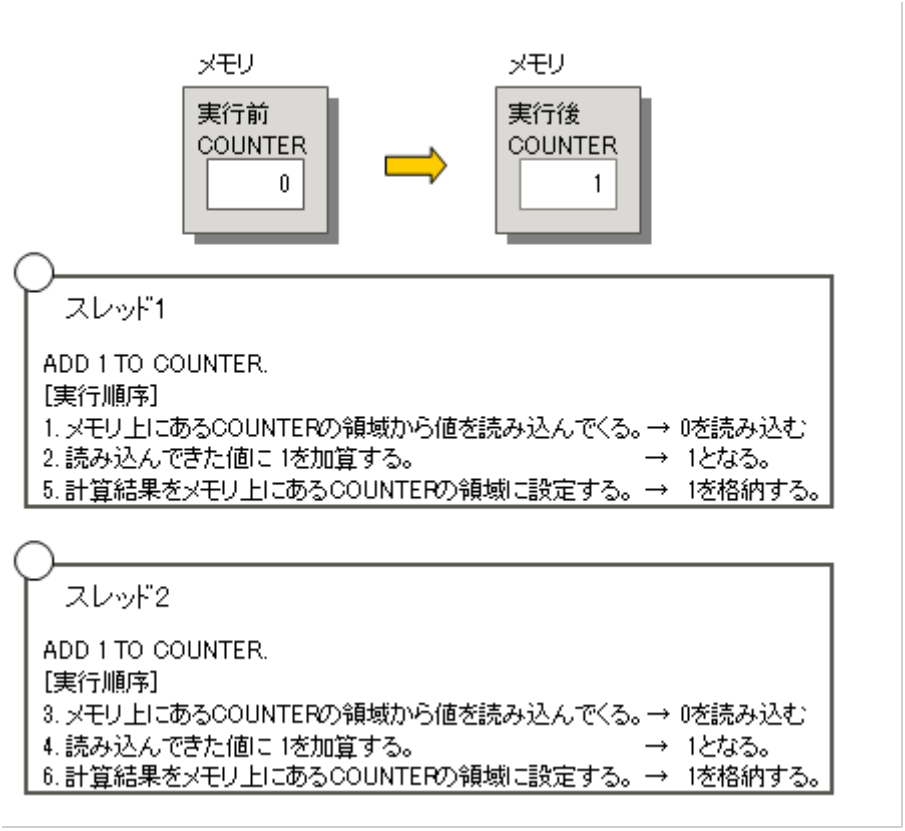
システムは、実行するスレッドを強制的に切り換えるため、スレッドの実行順序は予測できません。このため、スレッド間で資源を共有する場合、スレッドの実行順序が、プログラムの結果に影響を与える場合があります。これを「競合状態」と呼びます。

競合状態を例で説明します。

スレッド間で共有する COUNTER というデータに対して、"ADD 1 TO COUNTER"の文を実行する 2 つのスレッドがあったとします。"ADD 1 TO COUNTER"は 1 文ですが、コンパイラによっていくつかの機械語に展開され、システムは次のような順番で実行します。

1. メモリ上にある COUNTER の領域から値を読み込んでくる。
2. 読み込んできた値に 1 を加算する。
3. 計算結果をメモリ上にある COUNTER の領域に格納する。

このため､スレッド 1 とスレッド 2 の実行順序が以下のように切り替わった場合､COUNTER の値は 2 とならず、1 となってしまいます。



このような競合状態が発生するのは、次の条件のときです。もちろん、同一データをすべてのスレッドが参照するだけなら、同時にデータをアクセスしても問題ありません。

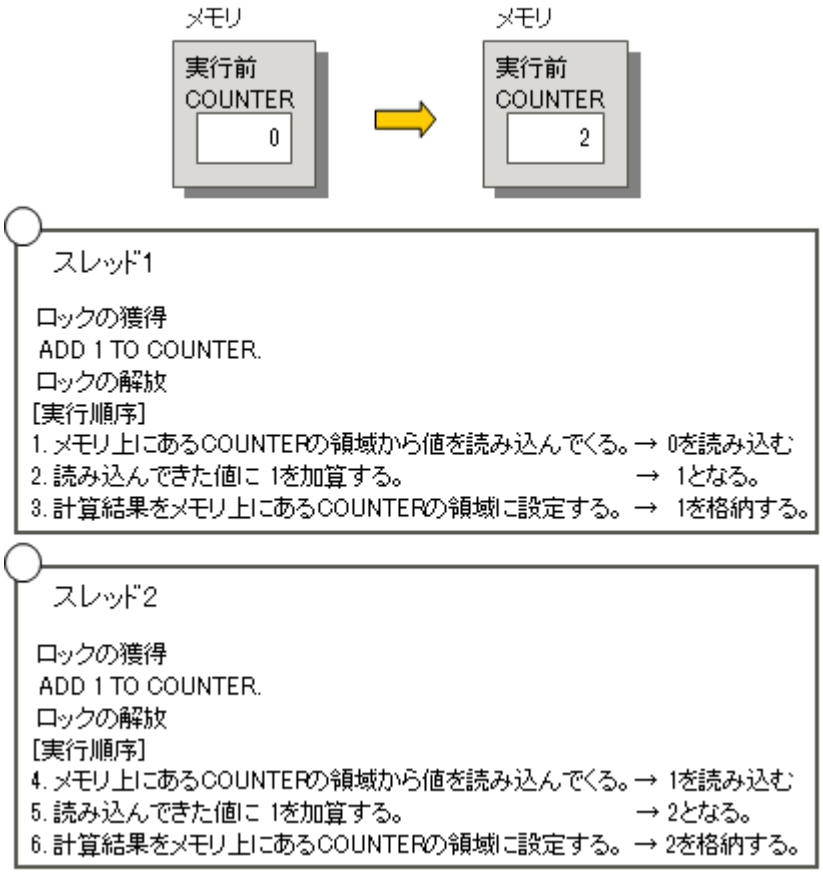
- 同一データを更新するスレッドと参照するスレッドが同時にデータをアクセスした場合。
- 同一データを更新するスレッドと更新するスレッドが同時にデータをアクセスした場合。

同一データに複数のスレッドが同時にアクセスしないようにするため、スレッド間で同期制御が必要となります。

これを行うための機構として、ロックがあります。プログラムを実行するために獲得しなければならない権利が

あり、これを「ロック」と呼びます。ロックを獲得できるスレッドは 1 つだけであり、ロックを獲得したスレッドだけが実行されます。ロックを獲得しようとしたほかのスレッドは、ロックを獲得しているスレッドがロックを解放するまで待つことになります。

上記の例でロックを使用すると、スレッド 1 の実行後にスレッド 2 が実行されるため、COUNTER の値は 2 となります (この例では、スレッド 1 が先にロックを獲得したとします)。



資源の共有

COBOL では、スレッド間で共有することができる資源として次のものがあります。

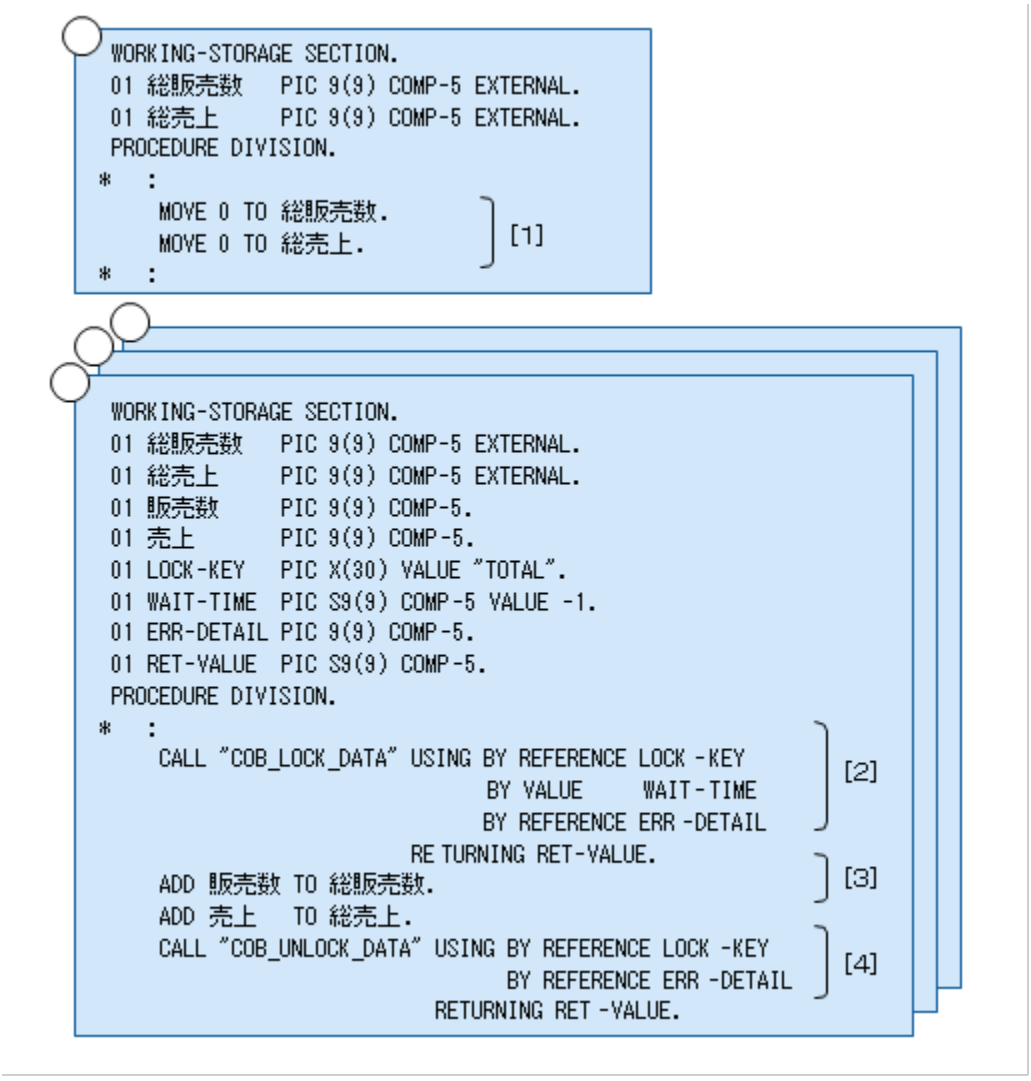
- スレッド間共有外部データとスレッド間共有外部ファイル
- スタティック定義
- オブジェクトインスタンス

共有できる資源に複数のスレッドが同時にアクセスしないようにするため、スレッドの同期制御が必要となります。同期制御の方法については、COBOL が提供するデータロックサブルーチン、または、.NET Framework から提供されるクラスライブラリを使用してください。

スレッド間共有外部データと外部ファイル

データ記述項またはファイル記述項に EXTERNAL 句を指定し、翻訳オプション SHREXT を指定して翻訳することにより、スレッド間で共通のデータ領域を使用することができます。

外部データをスレッド間で共有する例を以下に示します。外部ファイルをスレッド間で共有する場合は、スレッド間共有外部ファイルを参照してください。



[図の説明]

[1]: 初期化スレッドは、実行環境で最初に 1 回だけ起動されます。初期化スレッドで、共有データを初期化しています。

[2]: 共有データを複数のスレッドで同時に更新しないようにするため、データロックサブルーチンを使用して、"TOTAL"というデータ名に対応するロックキーに対してロックを獲得しています。データロックサブルーチンについては、 "データロックサブルーチン" を参照してください。

[3]: 共有データに値を加算しています。この処理は、[2]でロックを獲得したスレッドにより実行されます。

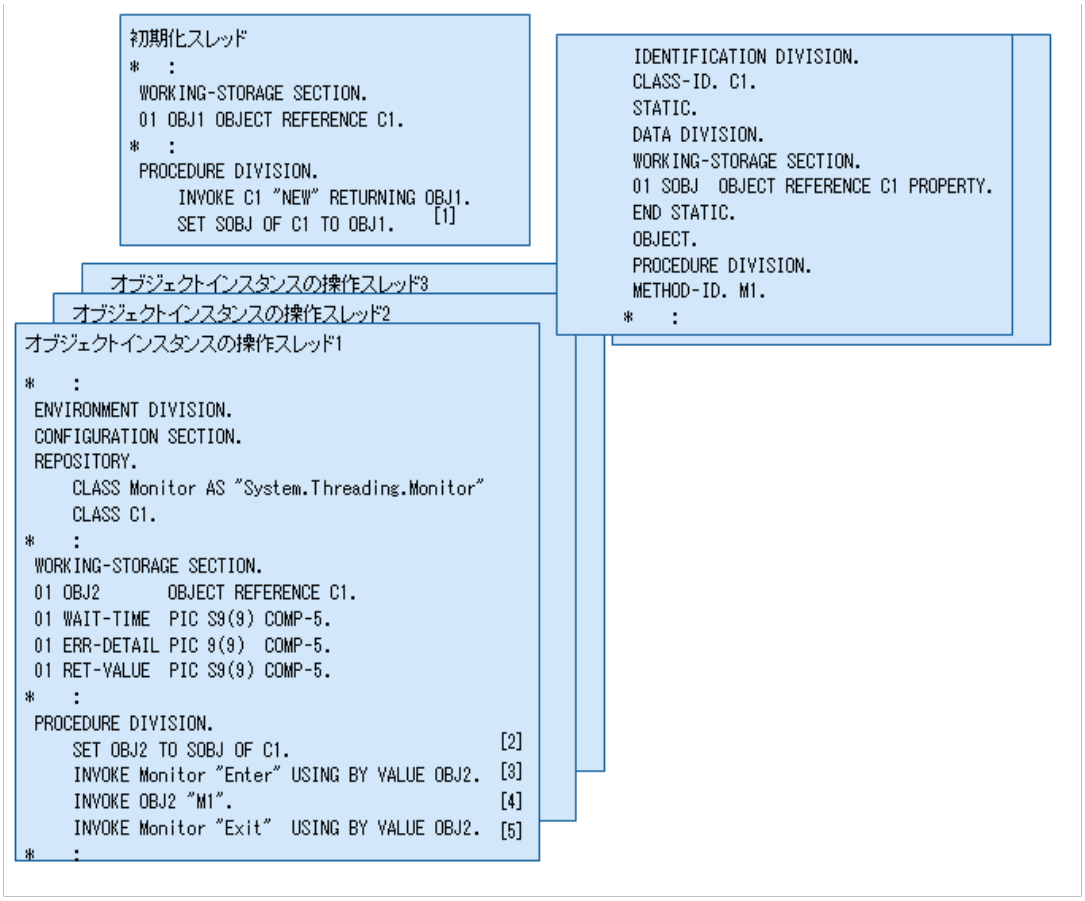
[4]: "TOTAL"というデータ名に対応するロックキーに対して獲得したロックを解放しています。ロックの解放により、別のスレッドで、[2]と同様の処理でロックを獲得できます。

スタティック定義

スタティック定義はスレッド間で共有されます。このため、スタティックデータを利用して、データやファイルをスレッド間で共有することができます。

オブジェクトインスタンス

スタティックデータを介して、スレッドからスレッドへオブジェクトインスタンスのオブジェクト参照を受け渡すことにより、オブジェクトデータをスレッド間で共有することができます。



[図の説明]

[1]: 初期化スレッドは、実行環境で最初に 1 回だけ起動されます。 初期化スレッドで、C1 クラスのスタティック定義のプロパティメソッドを呼び出し、C1 クラスのオブジェクトインスタンスをスタティックデータに設定しています。

[2]: C1 クラスのスタティック定義のプロパティメソッドを呼び出し、スタティックデータから[1]で設定された C1 クラスのオブジェクトインスタンスを取得します。

[3]: C1 クラスのオブジェクトインスタンスが複数のスレッドで同時に使用されないようにするため、.NET Framework の System.Threading.Monitor クラスを使用して､オブジェクトインスタンスのロックを獲得します。

もちろん、オブジェクトデータを持たない、またはオブジェクトデータを参照するだけであるなら、ロックする必要はありません。

[4]: C1 クラスのオブジェクトインスタンスの M1 メソッドを呼び出し、処理を行っています。この処理は、[3]でロックを獲得したスレッドにより実行されます。

[5]: オブジェクトインスタンスのロックを解放しています。ロックの解放により、別のスレッドが[3]でロックを獲得できます。

NCOBU00396

マルチスレッドの使用方法(基本編)

ここでは、マルチスレッドの基本的な使い方について説明します。

このセクションの内容

入出力機能の利用

ここでは、入出力機能を利用してファイル操作を行うマルチスレッドプログラムを開発する方法について説明します。

同一ファイルを共有する

外部媒体上のファイルとプログラムとの関係付けは、ファイル結合子を通して行われます。 ファイル結合子には、内部属性と外部属性を持つ 2 つのファイル結合子があります。 内部属性/外部属性に関係なく、スレッド間で異なるファイル結合子を操作して、ファイルを共有することができます。

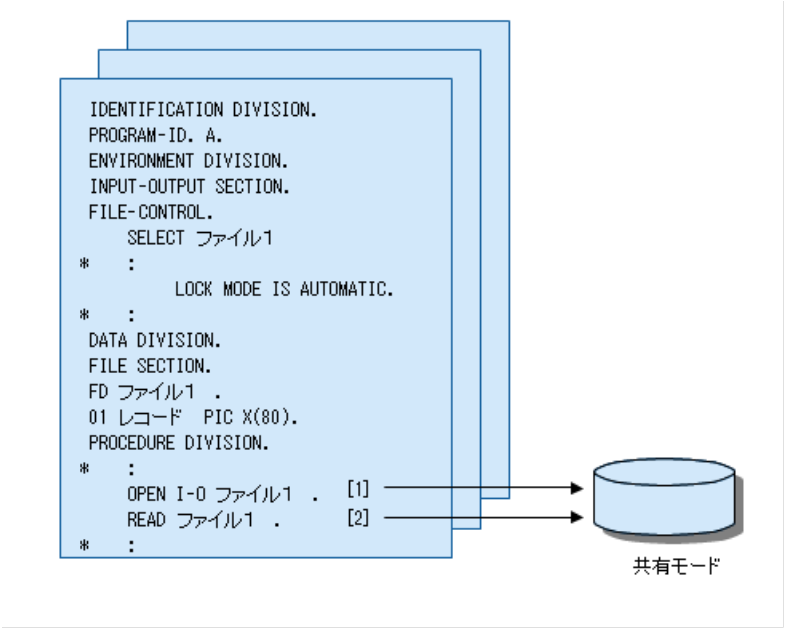
ここでは、内部属性を持つファイル結合子を操作して同一ファイルを共有する方法について説明します。

プログラム内に定義したファイル/メソッド内に定義したファイル/オブジェクト内に定義したファイル

プログラム内/メソッド内/オブジェクト内のファイル記述項に定義したファイルが内部属性を持つファイル結合子の場合、同一のファイルを割り当てることにより、スレッド間でファイルを共有することができます。

オブジェクト内のファイル記述項に定義したファイルの場合、別々のオブジェクトインスタンスのオブジェクト参照子を操作することで、プログラム内に定義したファイルと同様にファイルを共有することができます。

以下に、プログラム内に定義したファイルの共有処理を示します。



[図の説明]

[1]: 共用モードでファイルを開きます。

[2]: レコードを読み込みます。同一ファイルを操作する場合は、ファイルの排他制御に従って処理してください。ファイルの排他制御については、ファイルの排他制御を参照してください。

注意

スタティック定義内のファイル記述項に定義したファイルは、スレッド間で共有されます。 スタティック定義内に定義したファイルについては、マルチスレッドの使用方法(応用編)を参照してください。

NCOBU00398

リモートデータベースアクセスの利用

ここでは、リモートデータベースアクセスを行う既存のプログラムを、マルチスレッドプログラムに移行する方法について説明します。

マルチスレッドプログラムへの移行

プログラムの実行時に使用する SQL 情報のコネクション有効範囲(@SQL_CONNECTION_SCOPE)には、THREAD を指定します。 SQL 情報は、実行環境設定ユーティリティ(推奨)または ODBC 情報設定ユーティリティを使用して設定してください。

コネクション有効範囲に THREAD を指定すると、スレッドごとのコネクションが保証されます。 ただし、クラス定義に埋込み SQL 文が記述されているアプリケーション(オブジェクト指向機能使用時)は、この環境では動作しません。 以下の「マルチスレッドプログラムへの移行(オブジェクト指向機能使用時)」を参照してください。

マルチスレッドプログラムへの移行(オブジェクト指向機能使用時)

プログラムの実行時に使用する SQL 情報のコネクション有効範囲(@SQL_CONNECTION_SCOPE)には、OBJECT_INSTANCE を指定します。 SQL 情報は、実行環境設定ユーティリティ(推奨)または ODBC 情報設定ユーティリティを使用して設定してください。

コネクション有効範囲に OBJECT_INSTANCE を指定すると、 オブジェクトインスタンスごとのコネクションが保証されます。 実行時は、プログラム定義に埋込み SQL 文が記述されているアプリケーションは、 この環境では動作しません。 オブジェクトインスタンス単位でデータベースの接続から切断までの処理が完結していなければなりません。

注意事項

- 複数のスレッドが同一データベース表にアクセスする場合、 データベースがトランザクションの一貫性を保証するために、スレッドを待ち状態にしたり、 またはエラーを返すことがあります。 トランザクションの管理については、データベースのマニュアルを参照してください。 なお、トランザクションの管理は、 SQL 情報の@SQL_CONCURRENCY (カーソルの同時実行の指定)の指定値により、動作が異なる場合があります。
- 一部のデータベースでは、接続文字列や ODBC データソースの設定で、 マルチスレッド(スレッドセーフティ)の動作を指定できる場合があります。 この場合、必ずマルチスレッド(スレッドセーフティ)を有効にしてください。
- 一部のデータベースでは、マルチスレッドサポートされていない場合があります。 この場合、マルチスレッドプログラムは正常に動作しません。

環境変数の操作機能

環境変数の操作機能

環境変数の操作機能で使用する環境変数名は、スレッドごとに持ちます。このため、特殊名段落のENVIRONMENT-NAMEに割り当てる環境変数名については､複数のスレッドで別々のものを使用しても問題ありません。ただし、環境変数値はプロセス内で共通のため、マルチスレッドモードで環境変数操作を行うと、別スレッドに影響があります。

```
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
         ENVIRONMENT-NAME IS ENVNAME    *>[1]
         ENVIRONMENT-VALUE IS ENVVAL.   *>[2]
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 DATA01 PIC X(10).
      PROCEDURE DIVISION.
         DISPLAY "ABC" UPON ENVNAME.    *>[1]
         ACCEPT DATA01 FROM ENVVAL.     *>[2]
```

[プログラムの説明]

[1]: 環境変数名は、スレッドごとに持ちます。

[2]: 環境変数値は、プロセスで共通です。

NCOBU00400

マルチスレッドの使用方法(応用編)

ここでは、マルチスレッドの少し進んだ使い方について説明します。

このセクションの内容

入出力機能の利用

基本的な使い方では、スレッド間で異なるファイル結合子を操作したファイルの共有について説明しました。

ここでは、スレッド間で同じファイル結合子を操作してファイルを共有する方法について説明します。

スレッド間で同じファイル結合子を操作するには、以下の方法があります。

- スレッド間共有外部ファイル
- スタティック定義内に定義したファイル
- オブジェクト内に定義したファイル

PowerRDBconnector は、スレッド間で同じファイル結合子を操作することはできません。

各ファイルの利用方法については、それぞれのトピックで説明します。

なお、同じファイル結合子を操作してファイルを共有する場合、スレッドの競合状態が発生します。スレッドの競合状態を防ぐために、スレッドの同期制御が必要になります。これについては、各ファイルの利用方法で説明します。

同一ファイル結合子を使用して 1 つのファイルをアクセスする場合、入出力文の実行によりファイル位置指示子の状態が変化します。そのため、マルチスレッドで動作する場合、ファイル位置指示子の状態を意識したプログラムを設計する必要があります。

NCOBU00402

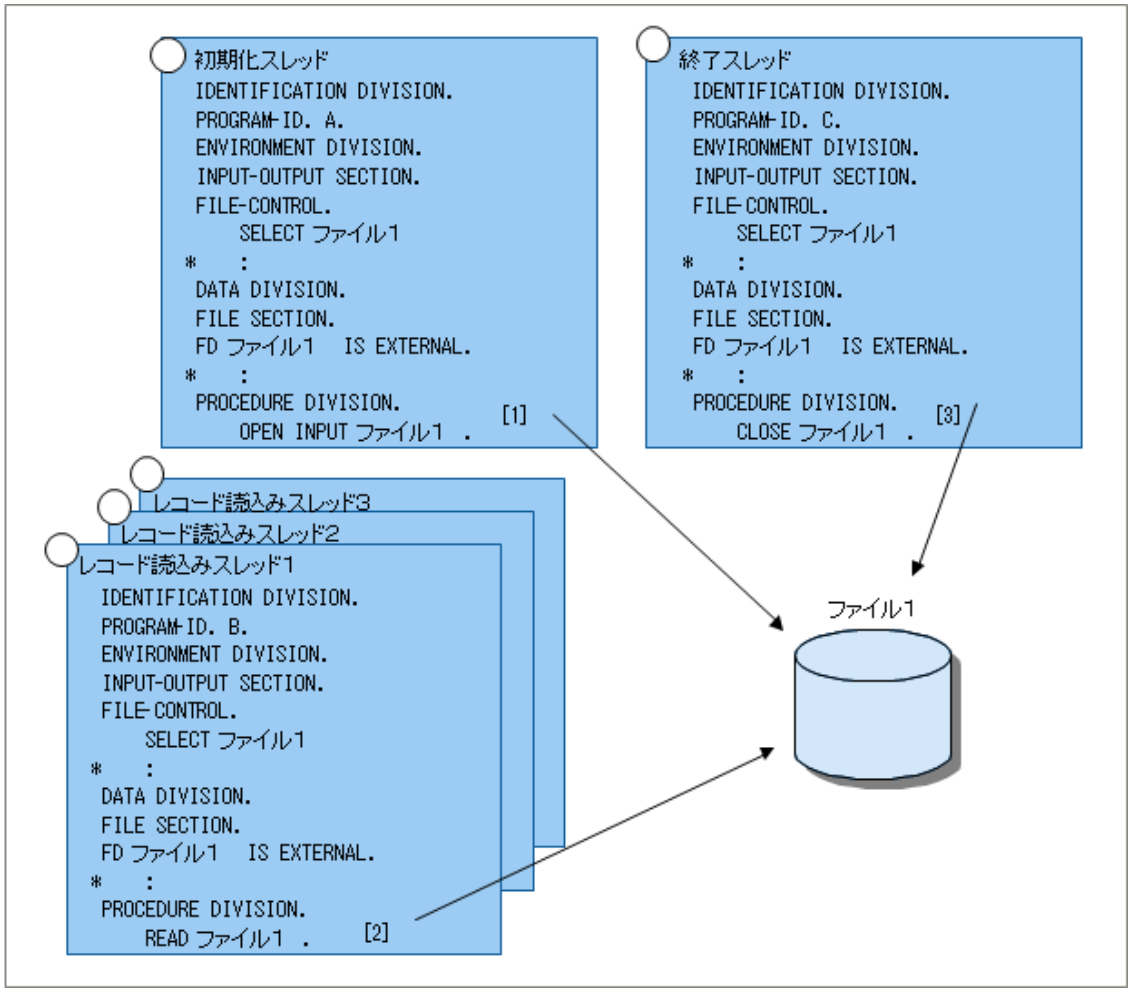
スレッド間共有外部ファイル

ファイル記述項に EXTERNAL 句を指定した場合、ファイル結合子に外部属性が与えられます。外部属性を持つファイル結合子は、プログラム間で同じファイル結合子を共有することができます。

複数のスレッドの間で同じファイル結合子を共有する場合は、翻訳オプション SHREXT を指定します。

なお、EXTERNAL 句は、メソッド内のファイル記述項にも指定することができます。

以下に、スレッド間共有外部ファイルを使用したプログラム例を示します。



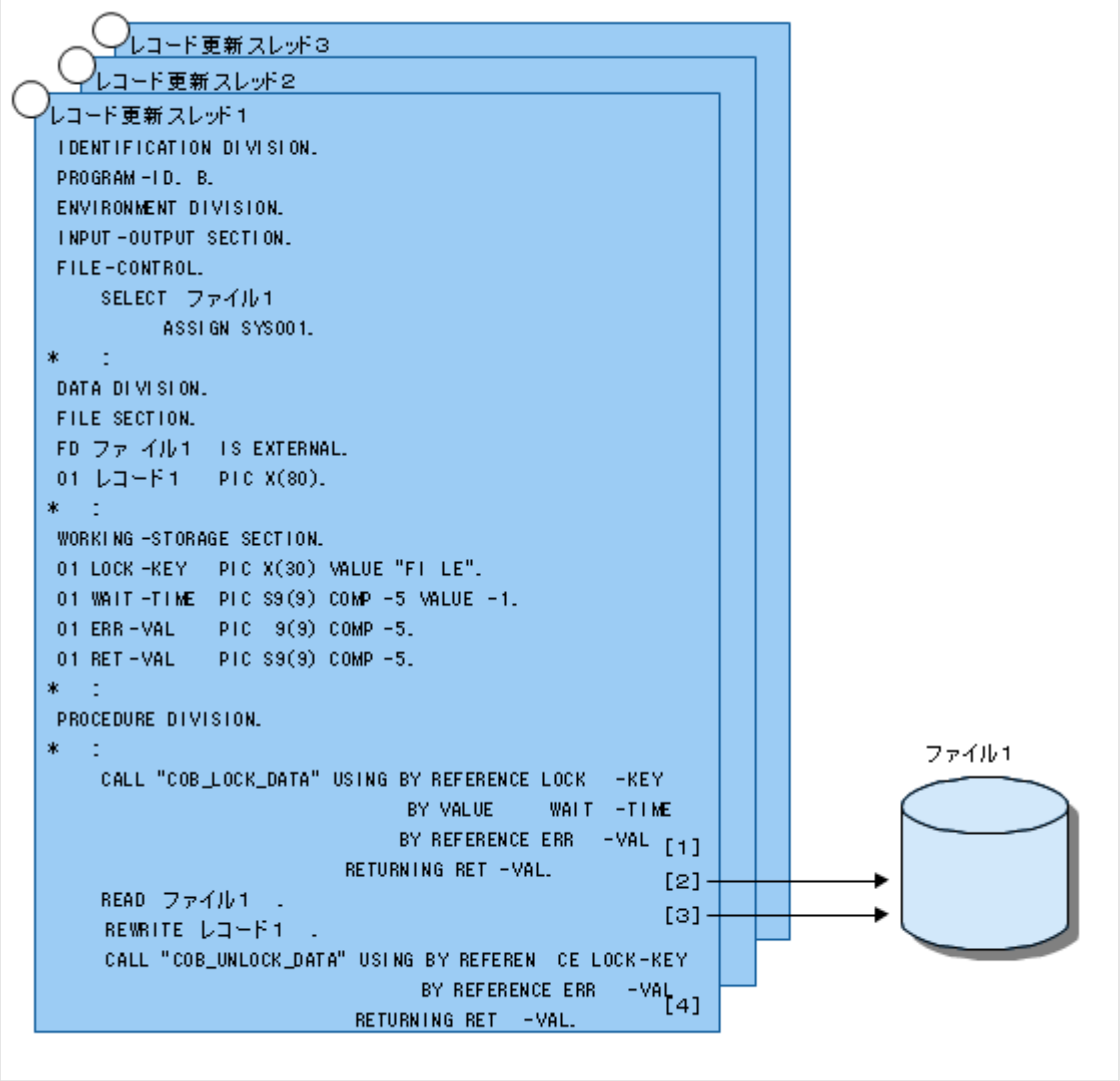
[図の説明]

[1]: 初期化スレッドは、実行環境で最初に 1 回だけ起動されます。初期化スレッドでは、外部属性のファイル結合子を持つファイルを開きます。

[2]: レコード読込みスレッドは、複数(プログラム例では 3 つ)同時に起動されます。レコード読込みスレッドでは、初期化スレッドによって開かれているファイルに対し、レコードを読み込みます。

[3]: 終了スレッドは、実行環境で最後に 1 回だけ起動されます。終了スレッドでは、初期化スレッドによって開かれているファイルを閉じます。

外部ファイルの場合、単一の入出力文については、COBOL ランタイムシステムで自動的にスレッドの同期制御を行います。しかし、複数の入出力文の実行で、意図した処理を行う場合、スレッドの同期制御が必要になります。外部ファイルへの同期制御を行うには、データロックサブルーチンを使用します。このサブルーチンを使用することにより、[2]の READ 文と[3]の REWRITE 文の間に、ほかのスレッドで別の READ 文が実行されることを防止することができます。



[図の説明]

[1]: データロックサブルーチンにより、"FILE"というデータ名に対応するロックキーに対してロックを獲得します。

[2]: ファイル１のレコードを読み込みます。

[3]: [2]で読み込んだレコードを更新します。

[4]: データロックサブルーチンにより、"FILE"というデータ名に対応するロックキーに対して獲得したロックを解放します。

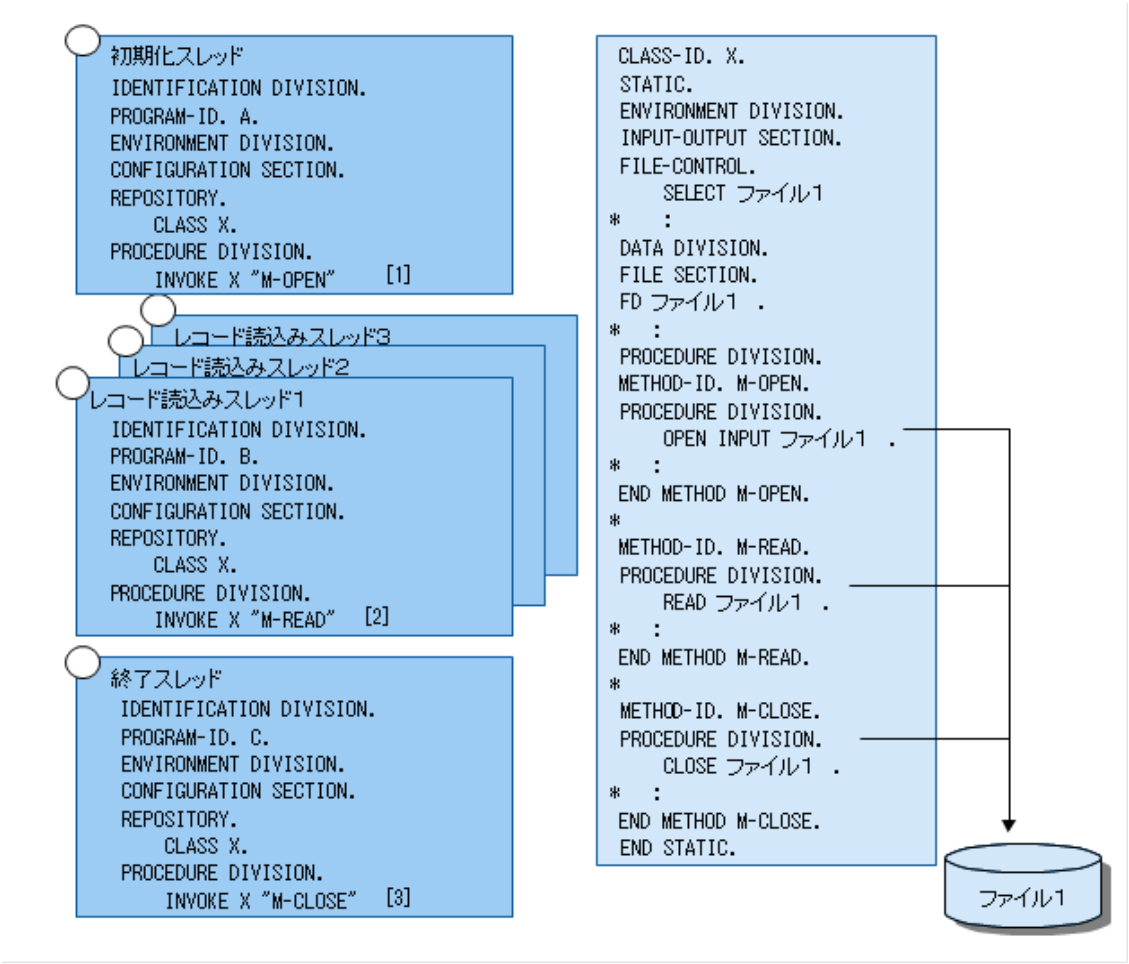
データロックサブルーチンについては、データロックサブルーチンを参照してください。

NCOBU00403

スタティック定義内に定義したファイル

スタティック定義内のファイル記述項に定義したファイルは、ファイル結合子を共有することができます。

以下に、スタティック定義内のファイルを使用したプログラム例を示します。



[図の説明]

[1]: 初期化スレッドは、実行環境で最初に 1 回だけ起動されます。初期化スレッドで、スタティックメソッド"M-OPEN"を呼び出し、ファイルを開きます。

[2]: レコード読込みスレッドは、複数(プログラム例では 3 つ)同時に起動されます。スタティックメソッド"M-READ"を呼び出し、初期化スレッドによって開かれているファイルに対し、レコードを読み込みます。

[3]: 終了スレッドは、実行環境で最後に 1 回だけ起動されます。終了スレッドで、スタティックメソッド"M-CLOSE"を呼び出し、初期化スレッドによって開かれているファイルを閉じます。

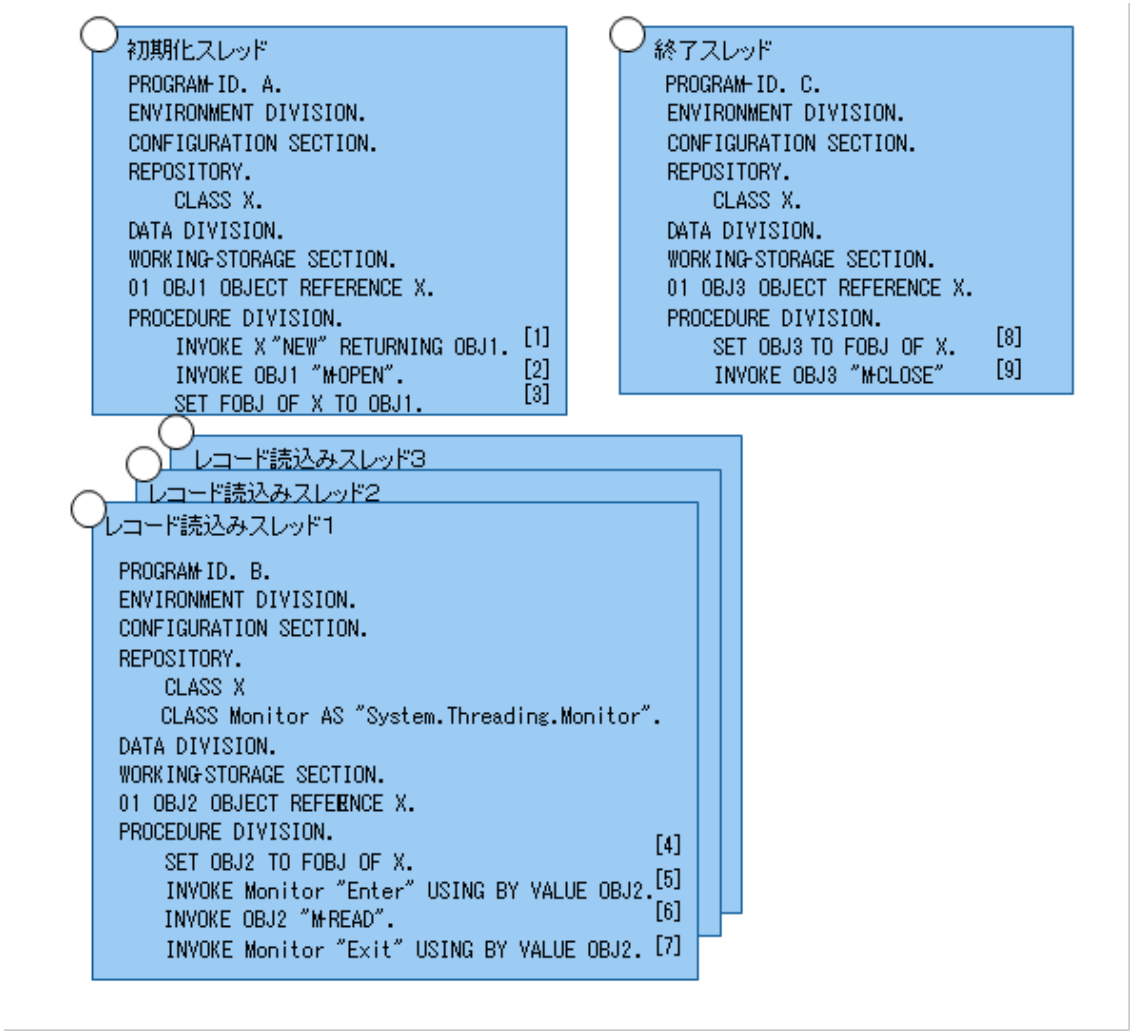
オブジェクト内に定義したファイル

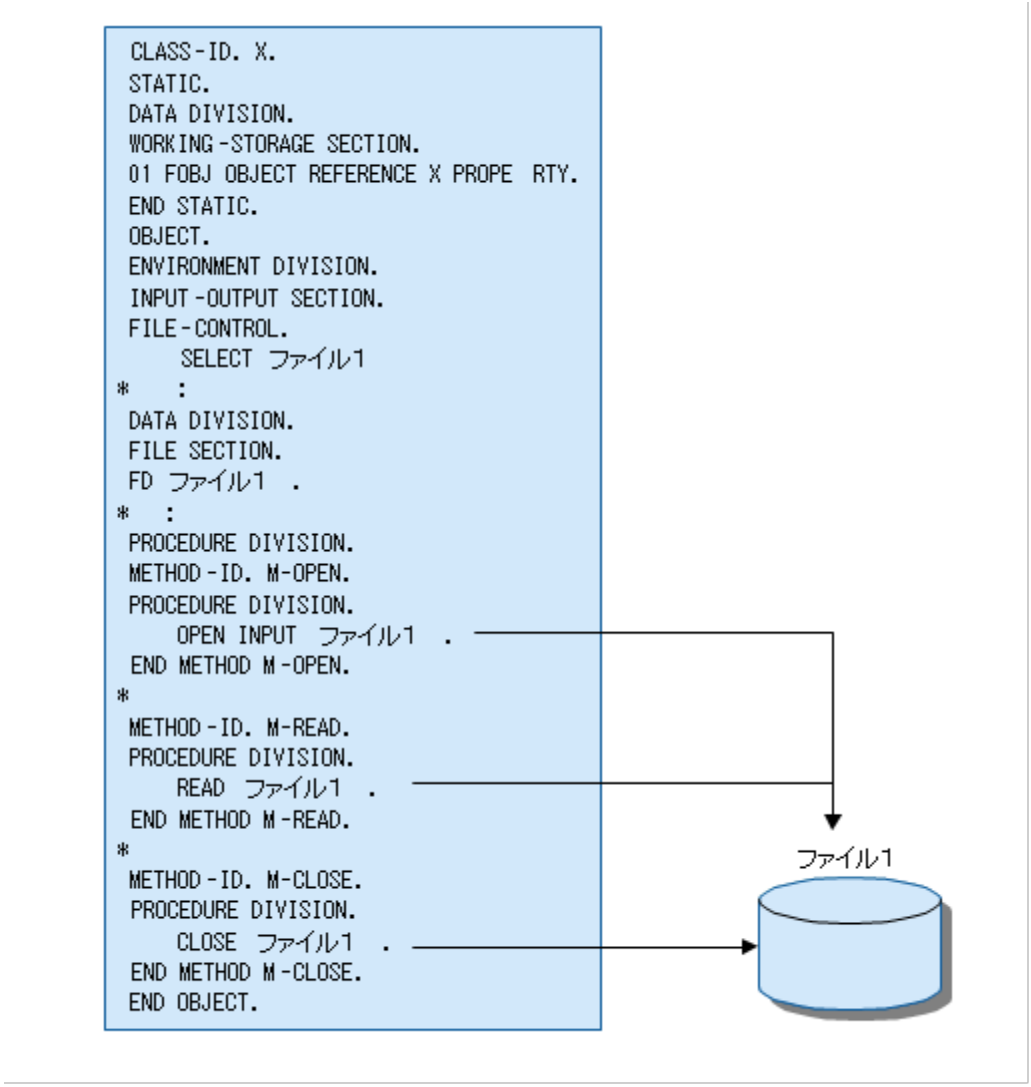
オブジェクト内のファイル記述項に定義したファイルは、1 つのオブジェクトインスタンスを共有することで、同じファイル結合子を共有することができます。

COBOL ランタイムシステムは、オブジェクトインスタンスに対して、スレッドの同期制御は行いません。このため、1 つのオブジェクトインスタンスを共有してオブジェクト内のファイルを操作する場合、スレッドの同期制御を行う必要があります。

オブジェクト内のファイルに対するスレッド間の同期制御は、.NET Framework の System.Threading.Monitor クラスを使用して行います。

以下に、オブジェクト内のファイルを使用したプログラム例を示します。





[図の説明]

初期化スレッド([1]～[3])は、実行環境で最初に 1 回だけ起動されます。

[1]: クラス'X'のオブジェクトインスタンスを獲得します。

[2]: メソッド'M-OPEN'を呼び出し、ファイルを開きます。

[3]: PROPERTY 句を指定したスタティックデータ'FOBJ'に、オブジェクトインスタンスを代入します。

レコード読込みスレッド([4]～[7])は、複数(プログラム例では 3 つ)同時に起動されます。

[4]: オブジェクトインスタンス'OBJ2'にスタティックデータ'FOBJ'を代入します。これにより、プログラム A で使用したオブジェクトインスタンスを共有することができます。

[5]:System.Threading.Monitor クラスにより、オブジェクトインスタンスのロックを獲得します。

[6]: メソッド"M-READ"を呼び出し、プログラム'A'で開いたファイルのレコードを読み込みます。

[7]: System.Threading.Monitor クラスにより、オブジェクトインスタンスのロックを解放します。

終了スレッド([8]～[9])は、実行環境で最後に 1 回だけ起動されます。

[8]: オブジェクトインスタンス'OBJ3'にスタティックデータ'FOBJ'を代入します。 これにより、プログラム A で使用したオブジェクトインスタンスを共有することができます。

[9]: メソッド"M-CLOSE"を呼び出し、初期化スレッドで開いたファイルを閉じます。

リモートデータベースアクセスの利用

ここでは、マルチスレッドの利点を活かしたマルチスレッドプログラムを新規に構築する方法など、少し進んだ使い方について説明します。

コネクションをスレッド間で共有する

コネクションをスレッド間で共有することにより、負荷の大きい接続および切断処理をそれぞれ初期化スレッドおよび終了スレッドに分離したり、データ操作処理を複数のスレッドで行うことが可能になります。これにより、アプリケーション性能が向上します。ただし、コネクションをスレッド間で共有して利用する場合、以下の点に注意する必要があります。

複数のスレッドで 1 つのコネクションを利用する場合、トランザクションも共有することになります。これは、あるスレッドでトランザクション処理を行った場合、コネクションを共有するすべてのスレッドのデータ操作に影響することを意味します。データベース表を変更しないマルチスレッドプログラムを動作させる場合は問題ありませんが、データベース表を変更するマルチスレッドプログラムを動作させる場合は、同時に実行されるスレッドがそれぞれ別々のコネクションを利用してトランザクション処理を行うようにプログラミングする必要があります。

以降では、データベース表を変更しない(参照だけ行う)例、およびデータベース表を更新する例を説明します。

例では、サンプルデータベースの STOCK 表を使用しています。

コネクション共有時のデータ参照

コネクションをスレッド間で共有して利用し、かつ、データベース表を変更しない(参照だけ行う)例を以下に示します。

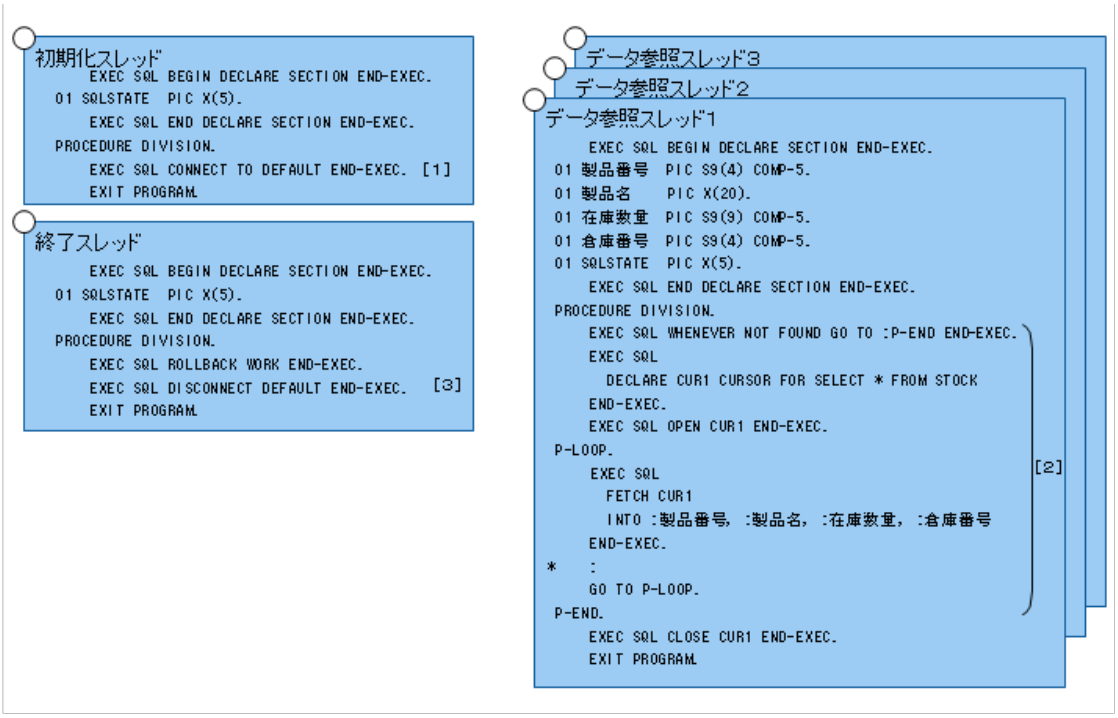


図:コネクション共有時のデータ参照例

[図の説明]

[1]: 初期化スレッドは、実行環境で最初に 1 回だけ起動されます。初期化スレッドで、CONNECT 文を実行し、サーバとのコネクションを接続します。

[2]: データ参照スレッドは、クライアントからの要求ごとに起動されます。データ参照スレッドで、カーソルを使用して STOCK 表からデータを 1 行ずつ取り出し、各列の値を対応するホスト変数の領域に設定します。 取り

出す行がなくなったとき、WHENEVER 文で指定した手続き名"P-END"の位置に分岐し、データ参照スレッドを終了します。

[3]: 終了スレッドは、実行環境で最後に 1 回だけ起動されます。終了スレッドで、ROLLBACK 文を実行し、トランザクションを終了させ、DISCONNECT 文を実行し、サーバとのコネクションを切断します。

コネクション共有時のデータ更新

下図は、コネクションをスレッド間で共有して利用し、かつ、データベース表を更新する COBOL プログラム例です。

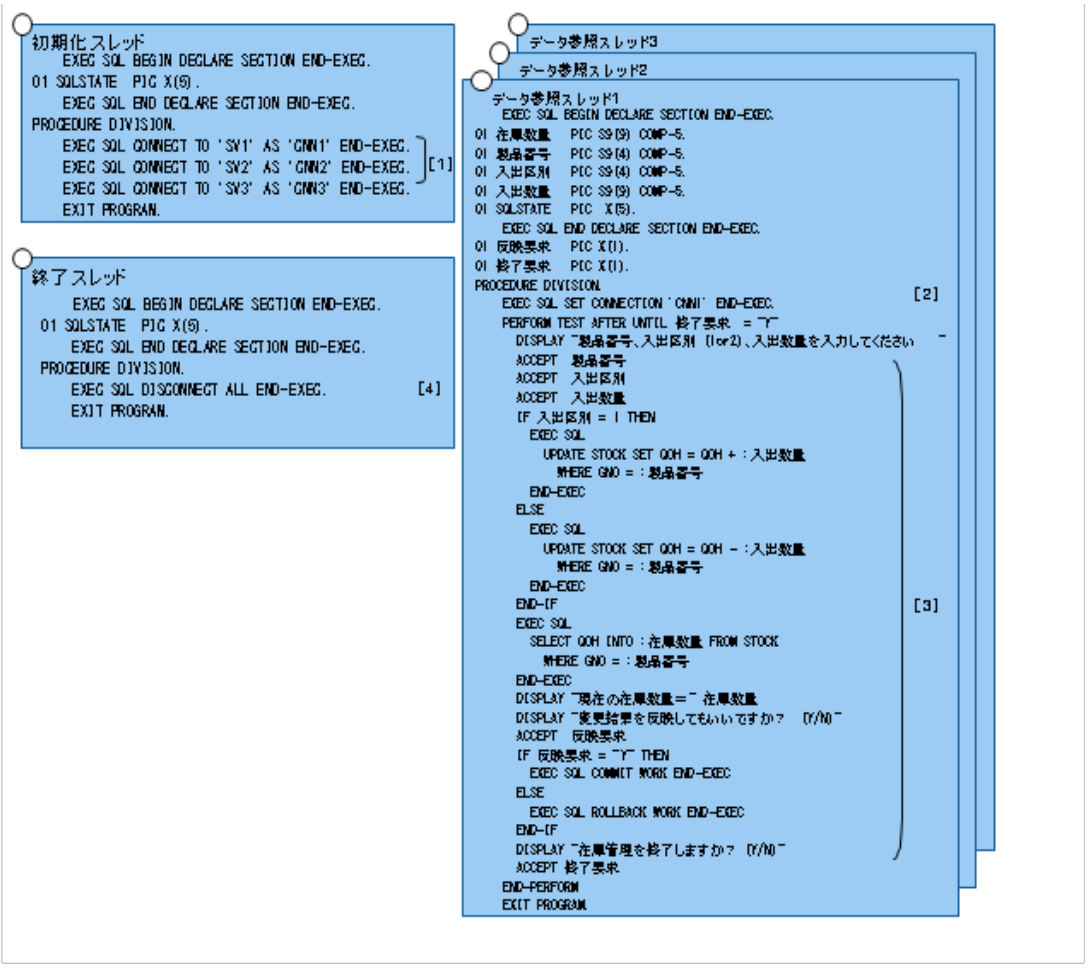


図:コネクション共有時のデータ更新例

[図の説明]

[1]: 初期化スレッドは、実行環境で最初に 1 回だけ起動されます。初期化スレッドで、データ変更スレッドを同時に起動する数(プログラム例では 3 つ)だけ CONNECT 文を実行し、サーバとのコネクションを接続します。

[2]: データ更新スレッドは、複数(プログラム例では 3 つ)同時に起動されます。データ更新スレッドで、SET CONNECTION 文を実行し、利用する現コネクションを決定します。 指定するコネクション名は、データ変更スレッドごとに異なるコネクション名(CNN1～CNN3)を指定します。

[3]: 製品番号、入出区別、入出数量の入力を要求し、入庫または出庫を切り分けて在庫数量を再計算します。次に再計算後の在庫数量を表示し、変更結果の反映有無を入力させます。 一連の処理は、終了要求に"Y"が入力されるまで繰り返し行います。

[4]: 終了スレッドは、実行環境で最後に 1 回だけ起動されます。終了スレッドで、DISCONNECT 文を実行し、すべてのサーバとのコネクションを切断します。

実行時の環境設定

コネクションをスレッド間で共有するには、 SQL 情報の@SQL_CONNECTION_SCOPE (コネクションの有効範囲の指定)に APPLICATION_DOMAIN を指定します。 SQL 情報は、実行環境設定ユーティリティ(推奨)または ODBC 情報設定ユーティリティを使用して設定してください。

@SQL_CONNECTION_SCOPE (コネクションの有効範囲の指定)に APPLICATION_DOMAIN を指定することにより、スレッド間でコネクションを共有して利用することができます。

NCOBU00406

スレッド同期制御サブルーチン

ここでは、スレッドの同期制御を行うためのサブルーチンについて説明します。スレッド同期制御サブルーチンには、データロックサブルーチンがあります。

このセクションの内容

データロックサブルーチン

データロックサブルーチンには以下があります。

サブルーチン名	機能
COB_LOCK_DATA	ロックキーに対するロックの獲得
COB_UNLOCK_DATA	ロックキーに対するロックの解放

同一プロセス内のスレッド間で同期制御が必要となる場合に、当サブルーチンを使用して、互いに同じデータ名のロックキーに対してロックの獲得と解放を行うことで相互排他ロックが可能となります。このときに指定するデータ名は、プロセス内で一意にする必要があります。

COB_LOCK_DATA の呼び出し時に、パラメタで指定されたデータ名に対応するロックキーが作成され、ロックを獲得します。指定されたデータ名に対応するロックキーが既に存在する場合は、そのロックキーに対してロックを獲得します。

ロックを獲得できるスレッドは 1 つだけであり、ロックを獲得したスレッドだけが実行されます。 同じロックキーに対してロックを獲得しようとしたほかのスレッドは、ロックを獲得しているスレッドがロックを解放するまで待つことになります。

ロックは、COB_UNLOCK_DATA を呼び出すことによって解放されます。

NCOBU00408

COB_LOCK_DATA

指定されたデータ名に対応するロックキーに対してロックを獲得します。

呼び出し形式

```
DATA DIVISION.
WORKING-STORAGE SECTION.
 01  LOCK-KEY       PIC X(30).
 01  WAIT-TIME      PIC S9(9) COMP-5.
 01  ERR-DETAIL     PIC 9(9)  COMP-5.
 01  RET-VALUE      PIC S9(9) COMP-5.
PROCEDURE DIVISION.
   CALL   "COB_LOCK_DATA"  USING  BY REFERENCE LOCK-KEY
                                  BY VALUE WAIT-TIME
                                  BY REFERENCE ERR-DETAIL
                          RETURNING  RET-VALUE.
```

パラメタ

LOCK-KEY

ロックを獲得するロックキーの名前を英数字項目 30 バイト以内で指定します。 ロックキーの名前が 30 バイトに満たない場合は、末尾に空白が必要となります。 ロックキーの名前の中に空白がある場合、空白以降の文字は切り捨てられます。

翻訳オプション RCS で指定した文字コードセットに含まれない文字をロックキーの名前に使用した場合、期待した動作にならないことがあります。

WAIT-TIME

ロックを獲得するまでの待ち時間(秒)を指定します。待ち時間は、-1 から最大 2147483(秒)の数字で指定します(注 1)。-1 を指定すると無限待ちとなります。

無限待ちを指定した場合、環境変数情報@CBR_THREAD_TIMEOUT に待ち時間(秒)を指定することにより、待ち時間を変更できます。 これは、デッドロックが発生した場合の箇所を特定したりする場合に利用します。

例: 無限待ち指定(-1)の待ち時間を 5 秒に変更する環境変数の指定

```
@CBR_THREAD_TIMEOUT=5
```

注 1: 範囲を越える値を指定した場合、動作は保障されません。

ERR-DETAIL

他システムとの互換のために存在します。このシステムでは使用されません。

戻り値

RET-VALUE

成功時は、0 が返ります。

失敗時は、負の値が返ります。 詳細については、スレッド同期制御サブルーチンエラーコードを参照してください。

NCOBU00409

COB_UNLOCK_DATA

指定されたデータ名に対応するロックキーに対してロックを解放します。

呼び出し形式

```
DATA DIVISION.
WORKING-STORAGE SECTION.
 01  LOCK-KEY       PIC X(30).
 01  ERR-DETAIL     PIC 9(9)  COMP-5.
 01  RET-VALUE      PIC S9(9) COMP-5.
PROCEDURE DIVISION.
   CALL   "COB_UNLOCK_DATA"  USING  BY REFERENCE LOCK-KEY
                                    BY REFERENCE ERR-DETAIL
                            RETURNING  RET-VALUE.
```

パラメタ

LOCK-KEY

ロックを解放するロックキーの名前を英数字項目 30 バイト以内で指定します。 ロックキーの名前が 30 バイトに満たない場合は、末尾に空白が必要となります。 ロックキーの名前の中に空白がある場合、空白以降の文字は切り捨てられます。

翻訳オプション RCS で指定した文字コードセットに含まれない文字をロックキーの名前に使用した場合、期待した動作にならないことがあります。

ERR-DETAIL

他システムとの互換のために存在します。このシステムでは使用されません。

戻り値

RET-VALUE

成功時は、0 が返ります。

失敗時は、負の値が返ります。 詳細については、スレッド同期制御サブルーチンエラーコードを参照してください。

スレッド同期制御サブルーチンエラーコード

ここでは、スレッド同期制御サブルーチンのエラー発生時の戻り値について説明します。

エラーコード	意味と処置	対象サブルーチン	
		COB_LOCK_DATA	COB_UNLOCK_DATA
-1	COBOL の実行環境が開設されていません。COBOL の実行環境を開設してから使用してください。	○	○
-2	パラメタの指定が誤っています。ロックキーの名前の指定誤り、待ち時間の指定誤り(COB_LOCK_DATA のみ)の可能性があります。	○	○
-3	あるスレッドのプログラムが異常終了したため、ロックが正しく解放されませんでした。 異常終了したプログラムのエラーの原因を取り除き、再度実行してください。	○	
-4	ロックの獲得の待ち時間を経過しました。	○	
-5	ロックを獲得していない、または別のスレッドが獲得したロックを解放しようとしました。		○
-6(注 2)	Hashtable が読み取り専用です。または、Hashtable が固定サイズです。		○
-7(注 2)	Mutex の作成に失敗しました。	○	
-8(注 2)	データベースに登録されたプログラムでは利用できない機能です。	○	○
-255(注 1)	システムエラーが発生しました。 この場合は、パラメタの ERR-DETAIL にシステムエラーコードが設定されます。	○	○

注 1: このシステムでは発生しないエラーコードです。他システムとの互換性を保つために予約されています。

注 2: このシステム固有のエラーコードです。

NCOBU00411

Unicode

ここでは、ソースプログラムや登録集などのリソースのコード系、および NetCOBOL for .NET アプリケーションの実行時データのコード系について説明します。

このセクションの内容

概要

NetCOBOL for .NET では、リソースを２つのコード体系(シフト JIS、UTF-8)から、実行時データの内部表現を３つのコード体系(シフト JIS、UCS-2、UTF-8)から目的に合わせて選択できます。リソースのコード系は SCS 翻訳オプション、実行時データのコード系は RCS 翻訳オプションで指定します。

リソースのコード系とアプリケーションの実行時データのコード系の組み合わせによって、処理モードが決定します。

以下の表は、リソースのコード系と実行時データのコード系の組み合わせによって決定する処理モードを示しています。

処理モード	リソースのコード系	実行時データのコード系		翻訳オプションの指定	プログラム定義	クラス定義
		英数字字類	日本語字類			
処理モード１	シフト JIS	シフト JIS	シフト JIS	SCS(ACP), RCS(SJIS)	○	×
処理モード２		シフト JIS	UCS-2	SCS(ACP), RCS(SJIS-UCS2)	○	○
処理モード３		UTF-8	UCS-2	SCS(ACP), RCS(UTF8-UCS2)	○	○
処理モード４	UTF-8	UTF-8	UCS-2	SCS(UTF8), RCS(UTF8-UCS2)	○	○

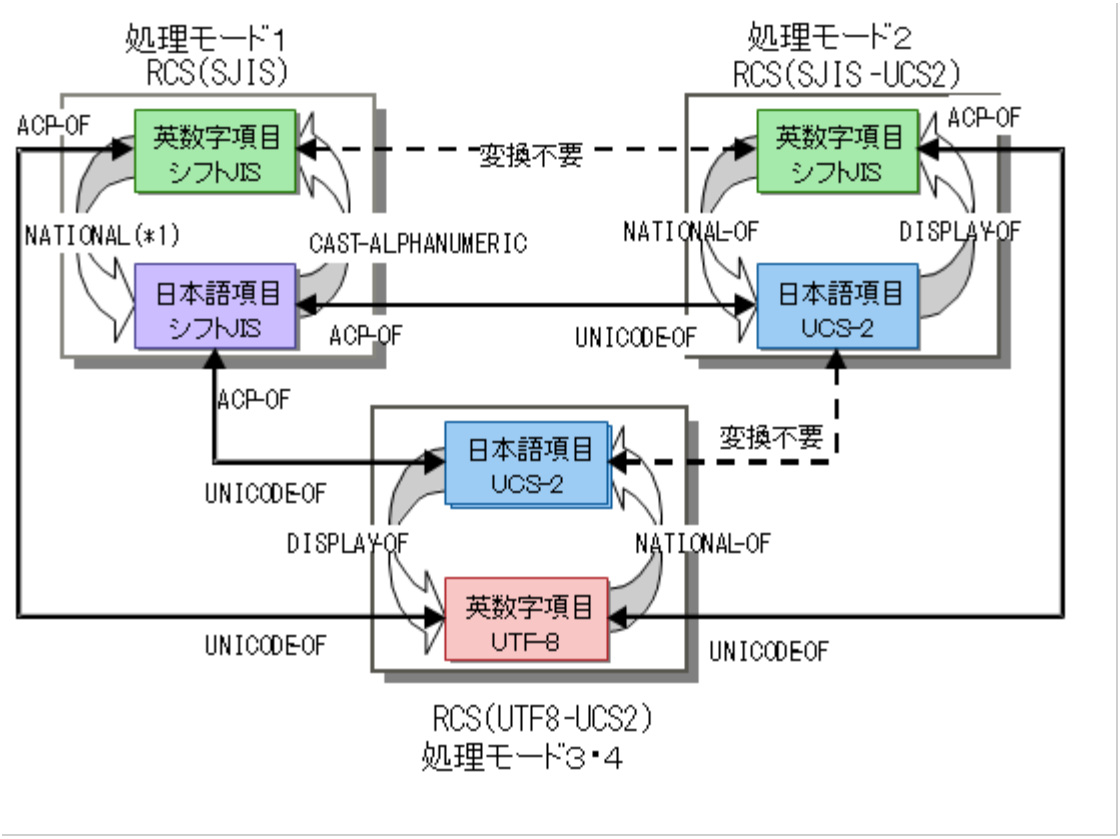
UCS-2 の表現はリトルエンディアンです。

なお、翻訳オプションの省略値は SCS(ACP)および RCS(UTF8-UCS2)です。このため、SCS 翻訳オプションおよび RCS 翻訳オプションのどちらも指定しない場合は、上の表から「処理モード３」が指定されたものとして動作します。

翻訳オプションについては、SCS (ソースプログラムのコード系)、および、RCS (実行時データのコード系)を参照してください。

各処理モード間の実行時データのコード系の関係

処理モード間の実行時データのコード系の関係は以下のようになります。



*1: NATIONAL 関数は、半角の文字を全角に変換します。

実行時データのコード系の間で行う変換は、それぞれのコード系がサポートしている範囲内での変換は相互に行うことができます。

- ACP-OF 関数は、文字列を ACP(Ansi Code Page)に変換します。
- UNICODE-OF 関数は、文字列を UNICODE に変換します。
- DISPLAY-OF 関数は、日本語文字を対応する英数字文字に置き換えます。
- NATIONAL-OF 関数は、英数字文字を対応する日本語文字に置き換えます。

それぞれの関数の詳細については、COBOL 文法書の「11.10.2.1 ACP-OF 関数」、「11.10.2.2 DISPLAY-OF 関数」、「11.10.2.6 NATIONAL-OF 関数」および「11.10.2.7 UNICODE-OF 関数」を参照してください。

注意

変換できない文字は、代替文字に変換されます。 この場合、元の文字コード値に戻すことはできません。

コード系の特徴

ここではコード系の特徴について簡単に説明します。

シフト JIS

Windows システムで広く利用されているコード体系で、Solaris でも使用できます。

ASCII や半角カナなどの半角文字(JIS8 とも呼ばれます)は 1 バイト固定で、日本語などの全角文字は 2 バイト固定で表現されます。収録されている文字数が少ないという弱点がある反面、半角カナが 1 バイトで表現できるなど、コード値が表示長と同じことから扱いやすい側面を持っています。

Windows で動作させていたアプリケーションを単純に.NET Framework へ移植したい場合や、COBOL ファイルなどの既存資産をそのまま利用したい場合などに適しています。

Unicode(UCS-2)

Unicode の表現形式のひとつで、Windows で使用できます。

シフト JIS と比較して収録されている文字数は多くなりますが、ASCII 文字を含めて 1 文字は 2 バイト固定で表現するため、やはりシフト JIS とは性質が異なります。

.NET Framework の実行時流通コードに採用されているため、他言語や.NET Framework と連携する場合には、この UCS-2 表現を使用します。

Unicode(UTF-8)

Unicode の表現形式のひとつで、Windows、Solaris、Linux などで使用できます。

ASCII 文字は 1 バイト、欧州などの国語文字は 2 バイト、日本語や半角カナ類は 3 バイトで表現する可変長形式です。このため、コード値だけでなくデータ長の点でシフト JIS と互換がありません。

UCS-2 で表現できる文字はすべて UTF-8 でも表現できるため、UCS-2 との親和性は高いといえます。

- Windows では、 JIS X 0213:2004 に対応した文字セット(以降、JIS2004 と記述します)が使用できます。この JIS2004 には、UCS-2 で表現できない文字が約 300 文字収納されています。本 COBOL コンパイラおよびランタイムシステムでは、UCS-2 で表現できない文字は使用できません。
- Windows 8.1 および Windows Server 2012 R2 以降では、IVS(Ideographic Variation Sequence)を利用した文字を使用できますが、本 COBOL コンパイラおよびランタイムシステムでは、IVS を利用した文字は使用できません。

NCOBU00414

処理モードの選択のポイント

処理モード 1～4 は、リソースのコード系と実行時データのコード系の組み合わせで決定されます。逆をいえば、処理モードを選択してから、コード系を決定することもできます。

ここでは、それぞれの処理モードが、他のシステムで動作させていたアプリケーションを移植する場合に、どの程度適しているのかについて説明しています。

コード系の特徴を理解した上で、処理モードの選択の参考にしてください。

処理モード 1

Windows で動作させていたアプリケーションを、そのまま.NET へ移植する場合や、COBOL ファイルなどの既存資産をそのまま継続利用する場合などに適しています。

ただし、.NET Framework との親和性は低いため、将来に渡ってエンハンスを続けるようなアプリケーションで全面的に採用することはお勧めしません。エンハンス予定のないアプリケーションや、既存資産を利用するための部分的な組込みに向いています。

処理モード 1 は、プログラム定義の場合だけ選択できます。プログラム定義以外のクラス定義などでは、このモードを選択できないため注意してください。

処理モード 2

Windows で動作させていたアプリケーションのエンハンスを伴う移植や、移植後もエンハンスを続ける場合などに適しています。

日本語項目を使用して、.NET Framework や他言語で記述されたモジュールと文字データを流通できます。

英数字字類はシフト JIS、日本語字類は UCS-2 となるため、データ長の観点で移植性が高いと言えます。COBOL ファイルなどの既存資産は、英数字字類のみを使用している場合にはそのまま使用できますが、日本語字類は表現形式が変わるため、日本語項目を使用している場合は、そのままでは利用できません。

処理モード 3

.NET 向けに新しくアプリケーションを作成する場合、および、データ宣言の見直しを許容できる移植に適しています。また、Windows で動作させていた Unicode アプリケーションを、そのまま.NET へ移植する場合や、Unicode アプリケーションで作成した COBOL ファイルなどの既存資産をそのまま継続利用する場合などに適しています。

Unicode で動作するため、.NET Framework や他言語で記述されたモジュールとの親和性が高く、英数字字類と日本語字類との相性も良いといえます。逆に、データ長の観点から既存アプリケーションの単純移植や既存資産の利用は困難となります。

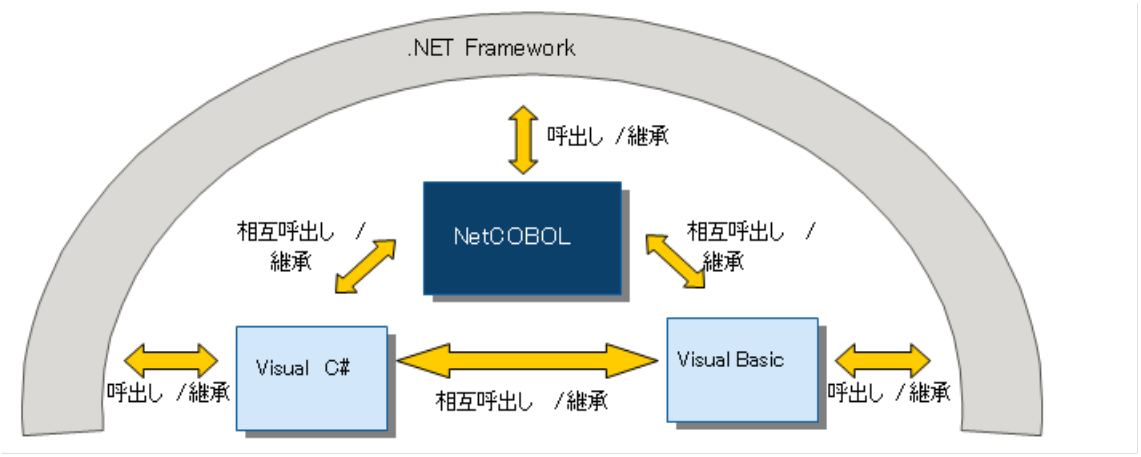
なお、プログラム定義だけでなく、すべてのソース定義でこのモードを選択できます。

処理モード 4

リソースが UTF-8 に変わるだけで、特長は処理モード 3 と同じです。

.NET 向けに新しくアプリケーションを作成する場合、および、データ宣言の見直しを許容できる移植に適しており、リソースを UTF-8 で作成する場合に使用します。

リソースを UTF-8 で作成すると使用できる文字が増えるメリットがある反面、文字の表示長とデータ長が異なることから、正書法の規則に従ったソース編集が難しくなるといったデメリットもあります。



処理モード 2,3,4 の解説図

NCOBU00415

処理モードに合わせたコード系の処理

ここでは、アプリケーションを作成したり動作させたりする場合に、処理モードの意識が必要なコード系の処理について説明します。

このセクションの内容

処理モード 1～4 共通

処理モード 1～4 に共通な内容について説明します。

処理モード 1

処理モード 1 を意識したコード系の処理について説明します。

処理モード 2

処理モード 2 を意識したコード系の処理について説明します。

処理モード 3

処理モード 3 を意識したコード系の処理について説明します。

処理モード 4

処理モード 4 を意識したコード系の処理について説明します。

処理モード 1～4 共通

処理モードの混在利用

ひとつの実行単位内で複数の処理モードを混在させることができます。

このとき、流通させるデータのコード系については、利用者責任で相手先に合わせてください。NetCOBOL for .NET ランタイムシステムがコード変換に介在することはありません。また、継承の局面では、親クラスと同じ処理モードで作成することを推奨します。

外部属性を持つデータおよびファイルを利用する場合は、処理モードを一致させてください。 なお、異なる処理モードで利用する場合は、実行環境変数@CBR_EXTERNAL_RCS_OPTION(外部属性の RCS オプションチェックの指定)に文字列"NOCHECK"を指定します。

暗黙の型変換

.NET Framework では、文字列を System.String オブジェクトとして扱いますが、COBOL では英数字項目や日本語項目などのデータ項目で操作します。このため、両者間でスムーズなやりとりができるように、SET 文を使用した暗黙の型変換と呼ばれる機能を提供しています。

この暗黙の型変換機能は処理モード 1～4 で使用できます。

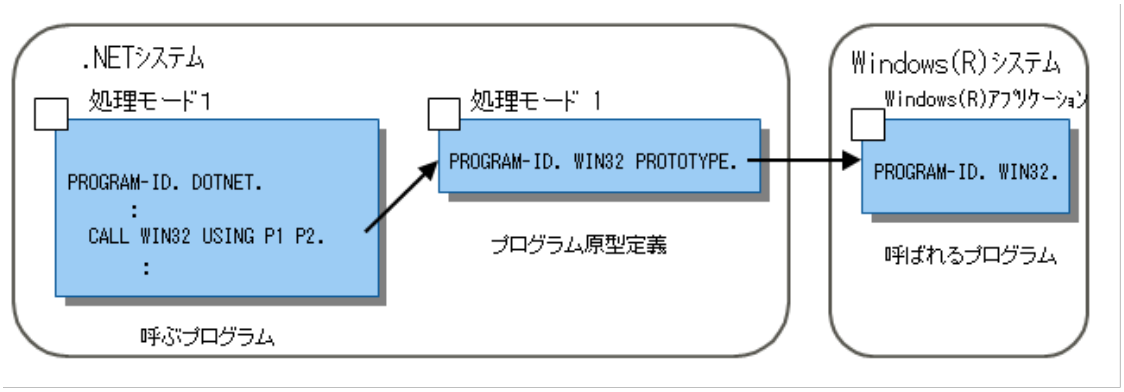
```
REPOSITORY.
   CLASS STR-CLASS AS "System.String".
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 PICX    PIC X(10).
 01 PICN    PIC N(10).
 01 OBJSTR  OBJECT REFERENCE STR-CLASS.
PROCEDURE DIVISION.
   SET OBJSTR TO PICX.  *> 英数字項目に String オブジェクトを代入
   SET OBJSTR TO PICN.  *> 日本語項目に String オブジェクトを代入
*    :
   SET PICX TO OBJSTR.  *> String オブジェクトに英数字項目を代入
   SET PICN TO OBJSTR.  *> String オブジェクトに日本語項目を代入
```

System.String オブジェクトと英数字項目または日本語項目との変換については、英数字項目・日本語項目との変換も参照してください。

プログラム原型定義の呼出し

NetCOBOL for .NET では、プログラム原型定義を使用して、アンマネージコードのモジュールを呼び出すことができます。

この機能は処理モード 1～4 で利用できます。このとき、呼ばれるプログラムの情報を定義するプログラム原型定義の処理モードは、呼ぶプログラムのモードに合わせることをお勧めします。



プログラム原型定義の詳細については、プログラム原型定義を参照してください。

注意

呼ぶプログラムとプログラム原型定義の処理モードが異なる場合、パラメタで受渡しする文字データは、呼ぶプログラムでプログラム原型定義の処理モードに合わせてコード変換しておく必要があります。 コード変換は、概要の「各処理モード間の実行時データのコード系の関係」に示したように、ACP-OF 関数または UNICODE-OF 関数を使用して行います。ACP-OF 関数または UNICODE-OF 関数の詳細については、COBOL 文法書の「11.10.2.1 ACP-OF 関数」および「11.10.2.7 UNICODE-OF 関数」を参照してください。

処理モード 1

処理モード 1 は、リソースのコード系をシフト JIS、実行時データのコード系をシフト JIS とした場合の処理モードです。翻訳オプション SCS(ACP)および RCS(SJIS)を指定すると、処理モード 1 として認識されます。

資源

リソースなどの各種資源を以下の表に示します。

	資源	コード系
翻訳時	ソースプログラム	シフト JIS
	登録集	
	翻訳リスト	シフト JIS
	各種定義体	無依存
実行時	COBOL ファイル（順/行順/相対/索引）	シフト JIS
	印刷ファイル	
	表示ファイル	
	小入出力ファイル	Unicode(UCS-2)
	実行時メッセージファイル	

表現形式

データの表現形式は以下の表のとおりです。

項目のレベル	字類	表現形式
基本項目	英字	ASCII
	英数字	ASCII(シフト JIS)
	日本語	シフト JIS
集団項目	英数字	ASCII(シフト JIS)

コーディング上の注意点

言語仕様

以下の言語要素は使用できません。

- プログラム定義以外のソース定義（クラス定義など）
- 組込み関数の UCS2-OF 関数および UTF8-OF 関数

文字定数

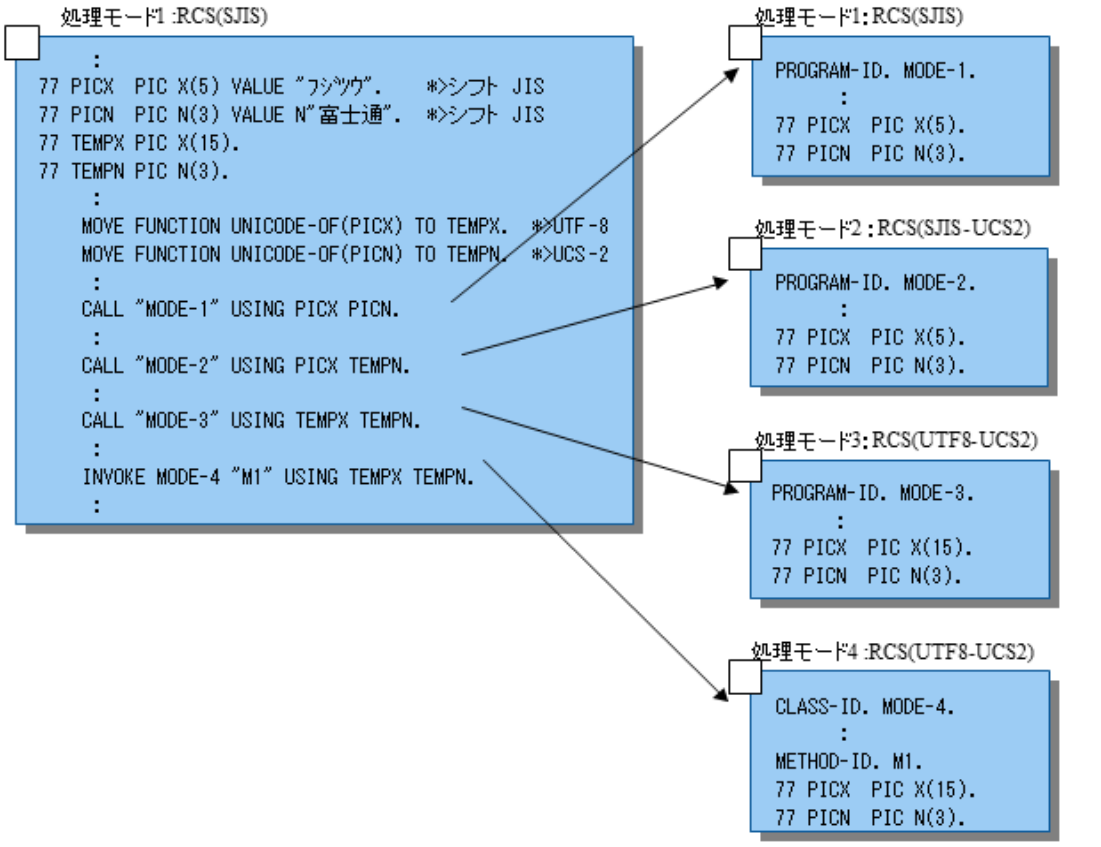
16 進文字定数はシフト JIS のコード値で記述してください。

```
      WORKING-STORAGE SECTION.
      01 NAMEN  PIC N(3) VALUE NX"95788E6D92CA".  *> NC"富士通".
     *    :
          IF NAMEN = NX"95788E6D92CA" THEN DISPLAY "OK!!".
```

なお、表意定数 SPACE は、作用対象が英数字項目の場合は半角空白、日本語項目の場合は全角空白になります。

プログラムやクラスの呼出し

特に制限なくプログラム定義やクラス定義を呼び出すことができますが、文字データを 受渡しする場合、相手先のコードへ合わせる必要があります。



処理モード 2

処理モード 2 は、リソースのコード系をシフト JIS、実行時データのコード系をシフト JIS/Unicode とした場合の処理モードです。翻訳オプション SCS(ACP)および RCS(SJIS-UCS2)を指定すると、処理モード 2 として認識されます。

資源

リソースなどの各種資源を以下の表に示します。

	資源	コード系
翻訳時	ソースプログラム	シフト JIS
	登録集	
	翻訳リスト	シフト JIS
	各種定義体	無依存
実行時	COBOL ファイル（順/行順/相対/索引）	シフト JIS/Unicode
	印刷ファイル(*1)(*2)	Unicode(UCS-2)
	表示ファイル(*2)	
	小入出力ファイル	
	実行時メッセージファイル	

*1: 印刷出力時

*2: 表示ファイルおよび FORMAT 句付き印刷ファイルは、翻訳エラーになります。

表現形式

データの表現形式は以下の表のとおりです。

項目のレベル	字類	表現形式
基本項目	英字	ASCII
	英数字	ASCII(シフト JIS)
	日本語	UCS-2
集団項目	英数字	ASCII(シフト JIS)

コーディング上の注意点

言語仕様

以下の言語要素は使用できません。

- プログラム定義以外のソース定義（クラス定義など）
- 組込み関数の UCS2-OF 関数および UTF8-OF 関数

文字定数

16 進文字定数はシフト JIS 表現で、日本語 16 進文字定数は UCS-2 のビッグエンディアンで記述してください。

```
     01 NAMEN  PIC N(3) VALUE NX"5BCC58EB901A". *> NC"富士通".
```

なお、表意定数 SPACE は、作用対象が英数字項目、日本語項目ともに半角空白となります。

転記・比較

NetCOBOL for .NET では、集団項目を使用することによって、異なる字類間の転記や比較ができます。

処理モード 2(処理モード 3 および処理モード 4 も同様)では英数字字類と日本語字類の表現形式が異なるため、日本語項目を含む集団項目の扱いに注意が必要です。

```
     WORKING-STORAGE SECTION.
     01 GRP-ITEM.
       02 NAT-ITEM   PIC N(10) VALUE NC"富士通". *> X"5BCC58EB901A" (UCS-2)
     01 ALP-ITEM    PIC X(10) VALUE "富士通".   *> X"95788E6D92CA" (シフト JIS)
     PROCEDURE DIVISION.
         IF NAT-ITEM = ALP-ITEM THEN    *>[1]
     *     :
         IF GRP-ITEM = ALP-ITEM THEN    *>[2]
     *     :
         MOVE ALP-ITEM TO GRP-ITEM.     *>[3]
```

[1]: 字類が異なるため、翻訳エラーになります。

[2]: 表現形式が異なるため、比較結果が偽(ELSE)になります。

[3]: [2]と同じく表現形式が異なるため、NAT-ITEM には不当なデータが格納されます。

集団項目を利用して字類の異なる転記や比較を行う場合は、表現形式を意識して作用対象のデータ構造を合わせるなどの注意が必要です。

文字の大小比較

日本語項目は UCS-2 のリトルエンディアンで管理されます。このため、日本語項目を含む集団項目を大小比較に用いる場合には注意が必要です。

```
     WORKING-STORAGE SECTION.
     01 GRP.
       02 SMALL  PIC N(1) VALUE NC"あ".  *> X"3042"
     01 LARGE    PIC N(1) VALUE NC"A".  *> X"FF21"
     PROCEDURE DIVISION.
     *     :
         IF GRP < LARGE THEN DISPLAY "OK??".    *>[1]
         IF SMALL < LARGE THEN DISPLAY "OK!!".   *>[2]
```

[1]: 左辺は集団項目のためリトルエンディアンのまま、右辺は日本語項目のためビッグエ

ンディアンに変換された状態で大小比較が行われます。その結果、左辺の方が大きくなり、偽になります。

[2]: 左辺、右辺ともに日本語項目のため、両方ともビッグエンディアンに変換された状態で大小比較が行われます。このため、正しい大小比較が行えます。

大小比較では、作用対象の字類およびデータ構造を合わせてください。

実際のコーディングでは、上記の例のような使い方はまれですが、同様なことが以下の場合にも当てはまります。

- 索引ファイルのキー
- 整列併合用ファイルのキー
- SEARCH ALL のキー指定 など

これらの局面で日本語を含む集団項目を使用する場合には注意してください。

ACCEPT 文・DISPLAY 文

作用対象が日本語を含む集団項目の場合に注意が必要です。

```
      WORKING-STORAGE SECTION.
      01 PERSONAL-DATA.
        02 NAME   PIC N(8).
        02 TEL    PIC 9(10).
      PROCEDURE DIVISION.
     *      :
          DISPLAY PERSONAL-DATA.   *>[1]
          DISPLAY NAME TEL.        *>[2]
```

[1]: 処理モード 2（処理モード 3 および処理モード 4 も同様）は字類によって表現形式が異なるため、集団項目の中に日本語が含まれる場合、不当なデータとみなされて文字化けが発生します。

[2]: [1]のような場合は基本項目ごとに指定します。

ACCEPT 文の入力先をファイルにした場合、NetCOBOL for .NET ランタイムが入力ファイルの表現形式を確認し、処理モードに合わせた表現形式に変換します。

DISPLAY 文の出力先をファイルにした場合、出力先のファイルの表現形式は UCS-2 のリトルエンディアンになりますが、実行環境変数@CBR_DISPLAY_SYSOUT_CODE (DISPLAY 文の出力ファイルのコード系の指定)を指定することでシフト JIS にすることができます。

COBOL ファイル

レコード順ファイル、相対ファイル、索引ファイル、および行順ファイルは、データの表現形式を変換しないで入出力しますが、印刷ファイルおよび表示ファイルは、各項目の字類に合わせて出力時にコード変換処理を行います。

以下に、注意点をまとめます。

◆ファイル識別名

ファイル識別名にデータ名を指定している場合、そのデータ名に日本語項目を含む集団項目は指定できません。

```
      FILE-CONTROL.
          SELECT OUTFILE ASSIGN TO FILE-NAME.
      WORKING-STORAGE SECTION.
      01 FILE-NAME.                          *>翻訳エラー
        02 F-PATH  PIC X(3).
        02 F-NAME  PIC N(4) VALUE NC"ファイル".
      PROCEDURE DIVISION.
```

```
        MOVE "C:¥" TO F-PATH.
        OPEN OUTPUT OUTFILE.
```

これは、異なる表現形式の混在によって発生するファイル名の文字化けを防ぐためです。ファイル識別名に集団項目を指定する場合は字類を英数字で統一してください。

◆行順ファイル

行順ファイルは、各種テキストエディターで表示・編集可能な形式であることが前提となるため、ひとつのファイルはひとつの表現形式で統一する必要があります。つまり、レコードを構成する各項目の字類を統一しなければなりません。処理モード 2 では、レコード定義の字類が英数字で統一されている場合はシフト JIS で、字類が日本語で統一されている場合は UCS-2 のリトルエンディアンで入出力します。以下の例のように字類が混在する場合、翻訳時エラーが出力されるため、用途に合わせて字類を統一してください。

```
     FILE-CONTROL.
         SELECT OUTFILE ASSIGN TO "data.txt"
                        ORGANIZATION IS LINE SEQUENTIAL.
    *    :
     DATA DIVISION.
     FILE SECTION.
     FD OUTFILE.                    *>翻訳エラー
     01 OUT-REC.
       02 REC-ID    PIC X(4).
       02 REC-DATA  PIC N(20).
```

Unicode のテキストファイルには、ファイルの先頭に signature と呼ばれる識別コードを 付加する決まりがあります。この signature は、NetCOBOL for .NET ランタイムシステムが処理するため、利用者が意識する必要はありません。

◆索引ファイル

索引キーに集団項目を指定し、かつ、その集団項目が日本語項目を含む場合、注意が必要です。

```
     FILE-CONTROL.
         SELECT IX-FILE ASSIGN TO F-NAME
                        ORGANIZATION IS INDEXED
                        RECORD KEY IS IX-KEY.
    *       :
     DATA DIVISION.
     FILE SECTION.
     FD IX-FILE.
     01 IX-REC.
       02 IX-KEY.
         03 KEY1     PIC X(4).
         03 KEY2     PIC N(8).
       02 IX-DATA   PIC X(80).
```

日本語項目には、データが UCS-2 のリトルエンディアン、つまり上位と下位バイトが逆転した形式で格納されます。これを集団項目で参照した場合、逆転した形式のまま処理されるため、START 文などで意図したレコードに位置づけられない可能性があります。

このような場合、翻訳時に警告の意味で W レベルエラーが出力されるので、基本項目で参照するようにプログラムを修正してください。

◆印刷ファイル

印刷ファイルの場合、レコード内の字類の統一は不要です。 各項目の字類に合わせて NetCOBOL for .NET ランタイムシステムがコード変換するため、表現形式が混在しても問題なく動作します。

```
      FILE-CONTROL.
          SELECT OUT-FILE ASSIGN TO PRTFILE.
     *       :
      DATA DIVISION.
      FILE SECTION.
      FD OUT-FILE.
      01 OUT-REC   PIC X(80).
      WORKING-STORAGE SECTION.
      01 PRT-DATA  CHARACTER TYPE IS MODE-1.
        02 PRT-NO    PIC 9(4).
        02 PRT-ID    PIC X(4).
        02 PRT-NAME  PIC N(20).
      PROCEDURE DIVISION.
     *       :
          WRITE OUT-REC FROM PRT-DATA AFTER ADVANCING 1 LINE.
```

ただし、日本語項目を含む集団項目を転記の受取り側項目に使用した場合、集団項目に字類と合わないコードの空白づめが行われます。これによって、文字化けなどの意図しない印刷結果になることがあります。

```
      FILE-CONTROL.
          SELECT OUT-FILE ASSIGN TO PRTFILE.
     *       :
      DATA DIVISION.
      FILE SECTION.
      FD OUT-FILE.
      01 OUT-REC   CHARACTER TYPE IS MODE-1.
        02 OUT-DATA  PIC N(40).
      WORKING-STORAGE SECTION.
      01 PRT-DATA  CHARACTER TYPE IS MODE-1.
        02 PRT-NO    PIC N(4).
        02 PRT-ID    PIC N(4).
        02 PRT-NAME  PIC N(20).
      PROCEDURE DIVISION.
     *       :
          MOVE  PRT-DATA TO OUT-REC.                      *>[1]
          WRITE OUT-REC AFTER ADVANCING 1 LINE.
     *
          MOVE NC"あいうえお" TO OUT-REC.                 *>[2]
          WRITE OUT-REC AFTER ADVANCING 1 LINE.
```

[1]: 受取り側項目が送出し側項目よりも大きい場合、集団項目転記の規則によって、半角空白が詰められます。このため、OUT-DATA には、UCS-2 と ASCII のデータが混在して格納されます。WRITE 文を実行すると、OUT-DATA は UCS-2 データとして扱われるため、ASCII データが格納された部分の印刷結果は文字化けします。

[2]: [1]の場合と同様です。

このような場合は、以下の例のように明示的に日本語項目を転記の対象に指定することで回避できます。

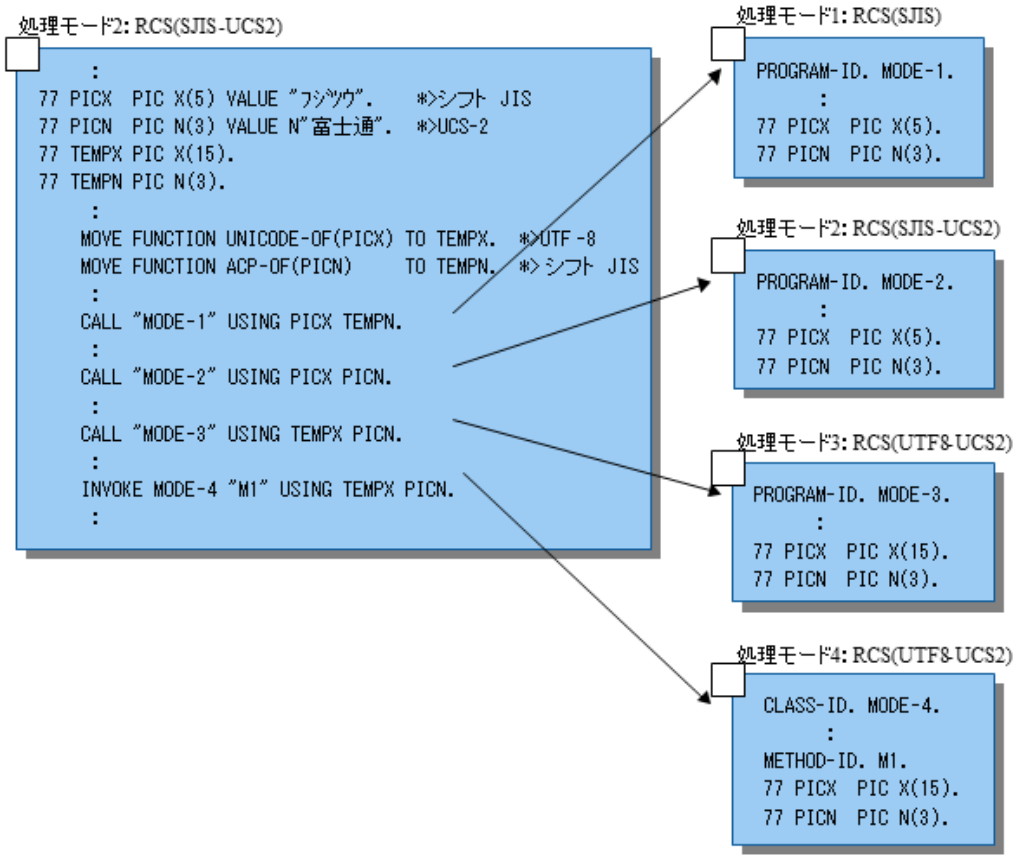
```
          WRITE OUT-REC FROM PRT-DATA AFTER ADVANCING 1 LINE.  *>[3]
     *
          MOVE NC"あいうえお" TO OUT-DATA.                *>[4]
          WRITE OUT-REC AFTER ADVANCING 1 LINE.
```

[3]: FROM 句指定の WRITE 文を使用した場合は、FROM 句に指定したデータ項目の字類に従って印字されます。

[4]: 転記の受取り側を日本語項目にした場合は、空白詰めには UCS-2 の半角空白が使用されるため、日本語項目に ASCII のデータが混在することはありません。

プログラムやクラスの呼出し

特に制限なくプログラム定義やクラス定義を呼び出すことができますが、文字データを受渡しする場合、相手先のコードへ合わせる必要があります。



データベースアクセス(ODBC)

- アクセス対象となるデータソース(データベース、ODBC ドライバ、および、その他のネットワークコンポーネント)が Unicode データの入出力をサポートしている必要があります。
- 埋込み SQL 文の値指定によって、データベースの Unicode データ型列に Unicode データを入出力する場合は、文字列定数または日本語文字列定数をシフト JIS データで記述してください。この場合、データソースがシフト JIS データを Unicode データに変換します。

処理モード 3

処理モード 3 は、リソースのコード系をシフト JIS、実行時データのコード系を Unicode とした場合の処理モードです。翻訳オプション SCS(ACP)および RCS(UTF8-UCS2)を指定すると、処理モード 3 として認識されます。

資源

リソースなどの各種資源を以下の表に示します。

	資源	コード系
翻訳時	ソースプログラム	シフト JIS
	登録集	
	翻訳リスト	シフト JIS
	各種定義体	無依存
実行時	COBOL ファイル（順/行順/相対/索引）	Unicode
	印刷ファイル	
	表示ファイル	
	小入出力ファイル	
	実行時メッセージ	

表現形式

データの表現形式は以下の表のとおりです。

項目のレベル	字類	表現形式
基本項目	英字	ASCII
	英数字	ASCII(UTF-8)
	日本語	UCS-2
集団項目	英数字	ASCII(UTF-8)

コーディング上の注意点

文字定数

16 進文字定数は UTF-8 表現で、日本語 16 進文字定数は UCS-2 のビッグエンディアンで記述してください。

```
01 NAMEX  PIC X(9) VALUE X"E5AF8CE5A3ABE9809A". *> "富士通"
01 NAMEN  PIC N(3) VALUE NX"5BCC58EB901A".     *> NC"富士通".
```

なお、表意定数 SPACE は、作用対象が英数字項目、日本語項目ともに半角空白となります。

転記・比較

英数字字類と日本語字類の表現形式が異なるため、日本語項目を含む集団項目の扱いに注意が必要です。詳細については、処理モード 2 の"コーディング上の注意"の"転記 ・比較"の説明を参照してください。

文字の大小比較

日本語項目は UCS-2 のリトルエンディアンで管理されます。このため、日本語項目を含む集団項目を大小比較に用いる場合は注意が必要です。詳細については、処理モード 2 の"コーディング上の注意"の"文字の大小比較"を参照してください。

ACCEPT 文・DISPLAY 文

作用対象が日本語を含む集団項目の場合に注意が必要です。詳細については、処理モード 2 の"コーディング上の注意"の"ACCEPT 文・DISPLAY 文"を参照してください。

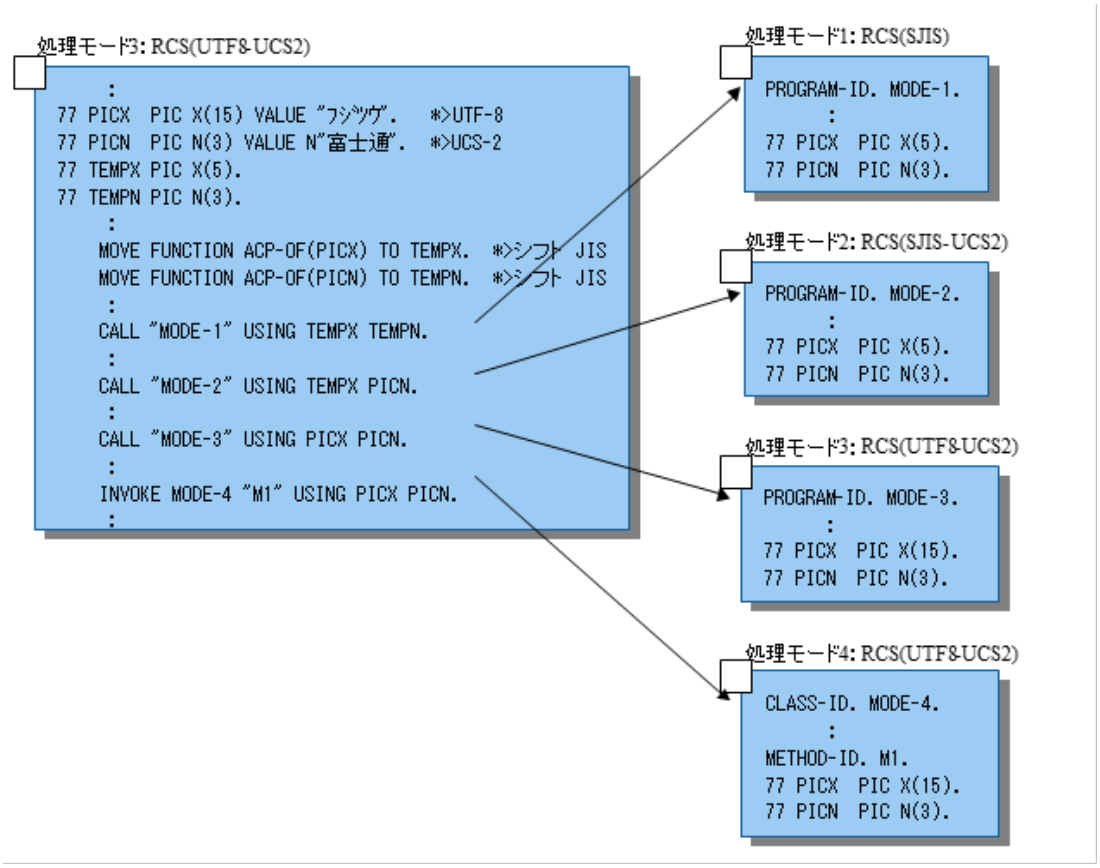
COBOL ファイル

レコード順ファイル、相対ファイル、索引ファイル、および行順ファイルは、データの表現形式を変換しないで入出力しますが、印刷ファイルおよび表示ファイルは、各項目の字類に合わせて出力時にコード変換処理を行います。なお、行順ファイルのレコードを構成する字類をすべて英数字項目にした場合、UTF-8 のファイルで入出力します。

注意点については、処理モード 2 の"コーディング上の注意"の"COBOL ファイル"を参照してください。

プログラムやクラスの呼出し

特に制限なくプログラム定義やクラス定義などを呼び出すことができます。また、クラス定義は、他の言語から呼び出せるだけでなく、.NET Framework や他言語で定義されたクラスを継承することもできます。



データベースアクセス(ODBC)

- アクセス対象となるデータソース(データベース、ODBC ドライバ、および、その他のネットワークコンポーネント)が Unicode データの入出力をサポートしている必要があります。
- 埋込み SQL 文の値指定によって、データベースの Unicode データ型列に Unicode データを入出力する場合は、文字列定数または日本語文字列定数をシフト JIS データで記述してください。この場合、データソースがシフト JIS データを Unicode データに変換します。

帳票

帳票定義体に定義した英数字項目には、1 バイトコードで表現される文字しか格納できません。 帳票定義体に定義した英数字項目に、 日本語文字(一部記号類、半角カナを含む)を格納して入出力を行うと、 意図した結果が得られません。

```
    FILE-CONTROL.
        SELECT IO-FILE ASSIGN TO GS-PRTF
                       SYMBOLIC DESTINATION IS "PRT".
*       :
    DATA DIVISION.
    FILE SECTION.
    FD IO-FILE.
       COPY 帳票定義体 OF XMDLIB.
*(01 表示レコード名.      )[注]
*(   02 DATA-1 PIC X(10). )
*       :
    PROCEDURE DIVISION.
        MOVE "ABC あいうエオ" TO DATA-1.      *>[1]
```

注: ()内は、COPY 文の展開を表します。

[1]: 日本語文字が混在するデータを格納したい場合、 帳票定義体には、英数字項目ではなく混在項目を定義します。 定義したい項目が英数字項目、混在項目のどちらでも、COPY 文で展開されるデータの属性は英数字項目になりますが、 実行時の扱いが異なるため、混在項目で定義されている場合だけ問題なく動作します。

処理モード 4

処理モード 4 は、リソースのコード系を Unicode、実行時データのコード系を Unicode とした場合の処理モードです。翻訳オプション SCS(UTF8)および RCS(UTF8-UCS2)を指定すると、処理モード 4 として認識されます。

資源

リソースなどの各種資源を以下の表に示します。

	資源	コード系
翻訳時	ソースプログラム	UTF-8
	登録集	
	翻訳リスト	Unicode(UTF-8)
	各種定義体	無依存
実行時	COBOL ファイル（順/行順/相対/索引）	Unicode
	印刷ファイル	
	表示ファイル	
	小入出力ファイル	
	実行時メッセージ	

表現形式

処理モード 3 と同じです。

コーディング上の注意点

処理モード 3 と同じです。

NCOBU00421

組込み関数の使用

ここでは、組込み関数の用例やコーディング時の注意点について説明します。

このセクションの内容

関数の型と記述の関係

関数はそれぞれに型を持っています。そして、その型の違いによってプログラム中に記述できる場所も異なります。 関数と型の対応については、"表:組込み関数一覧"を参照してください。

それぞれの型の関数の記述について、以下に説明します。なお、関数の呼出し形式に沿って書かれた記述のことを正しくは「関数一意名」と呼びますが、ここでは単に「関数」と呼んでいます。

整数関数

整数関数は、算術式中にだけ記述できます。 整数関数を算術式以外の場所、たとえば MOVE 文の送出し側に記述することはできません。

例では COMPUTE 文で使用しています。

例: 通日計算

```
      DATA          DIVISION.
      WORKING-STORAGE SECTION.
       01 INT        PIC S9(9) COMP-5.
       01 IN-YMD     PIC 9(8).
       01 OUT-YMD    PIC 9(8).
       01 OUT-YMD-ED  PIC XXXX/XX/XX.
      PROCEDURE      DIVISION.
          MOVE 20021225 TO IN-YMD.
      *  年月日→通日計算
          COMPUTE INT = FUNCTION INTEGER-OF-DATE(IN-YMD).
          DISPLAY "2002 年 12 月 25 日は，基準日から" INT "日目です．".
      *  通日→年月日計算
          COMPUTE OUT-YMD = FUNCTION DATE-OF-INTEGER (INT).
          MOVE OUT-YMD TO OUT-YMD-ED.
          DISPLAY "基準日から" INT "日目は，" OUT-YMD-ED "です．".
```

実行結果

```
2002 年 12 月 25 日は，基準日から+000146821 日目です.
基準日から+000146821 日目は，2002/12/25 です.
```

数字関数

数字関数は、整数関数と同様、算術式中にだけ記述できます。 数字関数を算術式以外の場所、たとえば MOVE 文の送出し側に記述することはできません。

英数字関数

英数字関数は、英数字項目が記述できるところに書くことができます。

例: UPPER-CASE 関数

```
      DATA          DIVISION.
      WORKING-STORAGE SECTION.
      01 LOWCASE     PIC X(13) VALUE "fujitsu cobol".
      PROCEDURE DIVISION.
         DISPLAY "変換前：" LOWCASE.
         DISPLAY "変換後：" FUNCTION UPPER-CASE(LOWCASE).
```

実行結果

```
変換前：fujitsu cobol
変換後：FUJITSU COBOL
```

例では、大文字に変換した文字を直接 DISPLAY 文で表示していますが、MOVE 文の送出し側に記述して作業域

に転記することもできます。

日本語関数

日本語関数は、日本語項目が記述できるところに書くことができます。 例では、変換した文字を一度転記してから表示します。

例: NATIONAL 関数

```
      DATA          DIVISION.
      WORKING-STORAGE SECTION.
      01 NCHAR  PIC N(7).
      01 CHAR   PIC X(7) VALUE "FUJITSU".
      PROCEDURE DIVISION.
     *  英数字を日本語文字に変換して表示
         MOVE FUNCTION NATIONAL(CHAR) TO NCHAR.
         DISPLAY "英数字 : " CHAR.
         DISPLAY "日本語 : " NCHAR.
```

実行結果

```
英数字 : FUJITSU
日本語 : ＦＵＪＩＴＳＵ
```

引数の型によって決定される関数の型

関数の中には、引数の型によって関数の型が決まるものがあります。 最大値を求める関数を例に挙げて説明します。

例: MAX 関数

```
      DATA          DIVISION.
      WORKING-STORAGE SECTION.
      01 C1   PIC X(10).
      01 C2   PIC X(5).
      01 C3   PIC X(5).
      01 V1   PIC S9(3).
      01 V2   PIC S9(3)V9(2).
      01 V3   PIC S9(3).
      01 MAXCHAR  PIC X(10).
      01 MAXVALUE PIC S9(3)V9(2).
      PROCEDURE DIVISION.
          MOVE FUNCTION MAX(C1 C2 C3) TO MAXCHAR.          *>[1]
     *       :
          COMPUTE MAXVALUE =  FUNCTION MAX(V1 V2 V3).       *>[2]
```

MAX 関数は、最大値を求める関数で、関数の型は引数の型によって決まります。

[1]: 引数の型が英数字であるため、関数の型は英数字となります。

[2]: 引数の型が数字であるため、関数の型は数字となります。

NCOBU00424

CURRENT-DATE 関数を利用した西暦の取得

小入出力機能を使った日付入力では、西暦の下 2 桁しか取得できませんが、CURRENT-DATE 関数を利用すれば、4 桁の西暦を得ることができます。

例

```
     DATA          DIVISION.
     WORKING-STORAGE SECTION.
     01 TODAY.
      02 YEAR   PIC X(4).
      02       PIC X(17).
     PROCEDURE DIVISION.
        MOVE FUNCTION CURRENT-DATE TO TODAY.
```

転記後の変数 YEAR の内容が、4 桁の西暦を示します。

環境変数情報@CBR_JOBDATE に任意の日付を指定すると、CURRENT-DATE 関数により指定された日付を受け取ることができます。

例

```
DATA          DIVISION.
WORKING-STORAGE SECTION.
  01 TODAY.
     02 YEAR   PIC X(4).
     02 MONTH  PIC X(2).
     02 DAY    PIC X(2).
     02       PIC X(13).
  PROCEDURE DIVISION.
      MOVE FUNCTION CURRENT-DATE TO TODAY.
```

プログラム実行時に環境変数情報@CBR_JOBDATE に 1999.09.01 を設定します。 転記後の変数 YEAR に 1999, 変数 MONTH に 09,変数 DAY に 01 が格納されます。 環境変数情報の指定形式については、任意の日付の入力を参照してください。

例では、MOVE 文の受取り側が集団項目であるため、集団項目転記が行われていますが、受取り側が数字項目であった場合は、数字転記が行われます。数字転記と集団項目転記では、桁よせの規則が異なるため、以下のように 4 桁の領域を準備しても、西暦は取得できません。

```
     DATA          DIVISION.
     WORKING-STORAGE SECTION.
     77 LAG    PIC 9(4).
     PROCEDURE DIVISION.
        MOVE FUNCTION CURRENT-DATE TO LAG.
```

転記後の変数 LAG の内容は、西暦ではなく、グリニッジ標準時からの進みまたは遅れ(CURRENT-DATE 関数の関数値の、18～21 桁)を示します。

任意の基準日からの通日計算

通日計算の結果得られた値の差を取り、任意の基準日からの通日を知ることができます。 例では、任意の基準日から現在の日付までの通日を計算し、その期間内での利息計算を行っています。

例

```
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 TODAY    PIC X(8).
      01 TODAY-R   REDEFINES TODAY  PIC 9(8).
      01 FROM-DAY  PIC 9(8).
      01 INT      PIC 9(5).
      01 PAY      PIC 9(6).
      01 EDT-PAY   PIC ZZZ,ZZ9.
      01 EDT-INT   PIC ZZZZ9.
      01 EDT1     PIC XXXX/XX/XX.
      01 EDT2     PIC XXXX/XX/XX.
      PROCEDURE DIVISION.
     * 基準日を設定する
         ACCEPT FROM-DAY
         MOVE FROM-DAY TO EDT1
     * 今日の日付を取得する
         MOVE FUNCTION CURRENT-DATE TO TODAY EDT2
     * ２つの日付間の日数を計算する
         COMPUTE INT =   (FUNCTION INTEGER-OF-DATE(FROM-DAY))
                     - (FUNCTION INTEGER-OF-DATE(TODAY-R))
     * 利息計算（例：20 日あたり 133.3 円固定）
         COMPUTE PAY = FUNCTION INTEGER-PART((INT / 20) * 133.3)
         MOVE PAY TO EDT-PAY.
         MOVE INT TO EDT-INT.
     * 結果の表示
         DISPLAY "預入日(" EDT1 ")から " EDT-INT "日 経過しています。" .
         DISPLAY EDT2 " 現在の利息合計は " EDT-PAY "円です。" .
```

実行結果 (基準日として"19920521"を入力した場合)

```
預入日(1992/05/21)から 2272 日 経過しています。
1998/08/10 現在の利息合計は  15,062 円です。
```

NCOBU01157

RANDOM 関数を利用した疑似乱数の取得

RANDOM 関数の関数値として、疑似乱数を取得できます。 このとき、関数値の範囲は 0≦関数値＜1 で、作用対象の小数部桁数に合わせて、桁よせされます。

例

```
 DATA          DIVISION.
 WORKING-STORAGE SECTION.
 01 A PIC  9(08).
 01 B PIC V9(07).
 PROCEDURE DIVISION.
*
* 引数は省略できます。
    PERFORM 5 TIMES
      COMPUTE B = FUNCTION RANDOM
      DISPLAY B
    END-PERFORM.
*
* 同じ種子の場合、同じ疑似乱数の値が返却されます。
    MOVE 12345678 TO A.
    COMPUTE B = FUNCTION RANDOM(A).
    DISPLAY B.
    COMPUTE B = FUNCTION RANDOM(A).
    DISPLAY B.
*
    STOP RUN.
```

- 引数の値(種子)が同じ場合は、同じ疑似乱数が返却されます。 ただし、種子が異なる場合および、種子を指定しない場合でも、同じ値が返却されることもあります。
- 関数値の範囲内であっても、返却されない疑似乱数はあります。

ポインタデータ項目の使用方法

ここでは、ポインタデータ項目の使用方法について説明します。

ポインタデータ項目

.NET Framework 環境には、ポインタに相当する機能がありません。そのため、NetCOBOL for .NET では擬似的にポインタの機能を再現した、擬似ポインタを提供しています。

擬似ポインタのため、ポインタデータ項目が指すことができる領域のサイズと個数には次の制限があります。

領域サイズ	個数
0byte < サイズ <= 64Kbyte	16,383
64Kbyte < サイズ <= 1Mbyte	1,023
1Mbyte < サイズ <= 16Mbyte	127

構文の規則の詳細については、COBOL 文法書を参照してください。

- .NET データ型は COBOL ポインタデータ項目でポイントすることができません。.NET データ型については、.NET データ型の COBOL 表現を参照してください。
- ポインタデータ項目はプログラム原型定義では使用できません。
- 16Mbyte を超える COBOL 独自データ型に割り付けられたデータに対して、そのデータへのポインタをポインタデータ項目に設定することはできません。

このセクションの内容

メモリの獲得

動的にメモリを獲得する方法について説明します。

獲得したメモリの使用方法

獲得したメモリの使用方法について説明します。

メモリの解放

獲得したメモリを解放する方法について説明します。

NCOBU00427

メモリの獲得

ここでは、NetCOBOL for .NET で動的にメモリを獲得する場合のプログラムの記述について説明します。

メモリを獲得するには、Fujitsu.COBOL.Subroutines.MemoryAllocator クラスの Alloc メソッドを呼び出します。

環境部の定義

```
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
  REPOSITORY.
     CLASS MEMORYALLOCATER AS "Fujitsu.COBOL.Subroutines.MemoryAllocator"
     .
```

データ部の定義

```
 01 MemoryPointer USAGE POINTER.
 01 MemorySize   PIC S9(9) USAGE COMP-5.
 01 ReturnCode   PIC S9(9) USAGE COMP-5.
```

INVOKE 文

```
   INVOKE MEMORYALLOCATER "Alloc"
        USING BY REFERENCE  MemoryPointer
              BY VALUE      MemorySize
        RETURNING           ReturnCode.
```

◆パラメタの説明

MemoryPointer	獲得したメモリへのハンドルが設定されます。
MemorySize	獲得するメモリのサイズを指定します。指定できる値の範囲は 1 ～ 16,777,216 です。

◆復帰値

ReturnCode には、復帰値として以下の値が返却されます

0	領域の獲得に成功
1	領域の獲得に失敗

メモリ獲得するメソッドの呼出し例

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MEMALLOC.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS MEMORYALLOCATER AS "Fujitsu.COBOL.Subroutines.MemoryAllocator"
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 RESULT    PIC S9(9) USAGE COMP-5.
LINKAGE SECTION.
01 PTR              USAGE POINTER.
01 ALLOCSIZE PIC S9(9) USAGE COMP-5.
PROCEDURE DIVISION USING PTR ALLOCSIZE.
    INVOKE MEMORYALLOCATER "Alloc" USING BY REFERENCE PTR
                                         BY VALUE ALLOCSIZE
                                RETURNING RESULT.
END PROGRAM MEMALLOC.
```

獲得できる領域の制限

獲得した領域はポインタデータ項目に設定されて返されます。 このため、獲得できる領域は、ポインタデータ項目に設定できる領域のサイズと個数の制限の影響を受けます。

領域サイズ	最大個数
0 byte < サイズ <= 64 Kbytes	16,383
64 Kbyte < サイズ <= 1 Mbytes	1,023
1 Mbyte < サイズ <= 16 Mbytes	127

システムの状態により、実際に獲得できる領域の個数は、最大個数に満たない可能性があります。

NCOBU00428

獲得したメモリの使用方法

獲得したメモリは、連絡節に宣言したデータのメモリとして、また、基底場所節に宣言したデータのポインタ修飾用として使用できます。

```
        SET ADDRESS OF linkage-section-item TO pointer-item
        MOVE pointer-item->based-storage-item TO data-item
        MOVE data-item TO pointer-item->based-storage-item
```

また、ポインタデータ項目には、宣言されている COBOL データ型に対するポインタを設定することができます。

```
        SET pointer-item TO ADDRESS OF linkage-section-item
        MOVE FUNCTION ADDR(data-item) TO pointer-item
```

以下のように.NET データ型をポインタ項目に設定することはできません。エラーとなります。

```
    01 A   PIC S9(9) COMP-5.  *> System.Int32
       :
       MOVE FUNCTION ADDR(A) TO pointer-item  *> Error
```

しかし、COBOL データ型となるようにすることでポインタ項目に設定することができます。

```
    01 B.
      02 PIC S9(9) COMP-5.
       :
       MOVE FUNCTION ADDR(B) TO pointer-item *> OK
```

メモリの解放

メモリの獲得で獲得したメモリを解放する場合のプログラムの記述について説明します。

メモリを解放するには、Fujitsu.COBOL.Subroutines.MemoryAllocator クラスの Free メソッドを呼び出します。

環境部の定義

```
ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
  REPOSITORY.
     CLASS MEMORYALLOCATER AS "Fujitsu.COBOL.Subroutines.MemoryAllocator"
     .
```

データ部の定義

```
01 MemoryPointer USAGE POINTER.
01 ReturnCode    PIC S9(9) USAGE COMP-5.
```

INVOKE 文

```
    INVOKE MEMORYALLOCATER "Free"
                      USING BY REFERENCE  MemoryPointer
                      RETURNING         ReturnCode.
```

◆パラメタの説明

MemoryPointer	メモリ獲得メソッドで獲得した領域を設定します。

◆復帰値

ReturnCode には、復帰値として以下の値が返却されます

0	領域の解放に成功
1	領域の解放に失敗

メモリ解放するメソッドの呼出し例

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MEMFREE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS MEMORYALLOCATER AS "Fujitsu.COBOL.Subroutines.MemoryAllocator"
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 RESULT    PIC S9(9) USAGE COMP-5.
LINKAGE SECTION.
01 PTR               USAGE POINTER.
PROCEDURE DIVISION USING PTR.
    INVOKE MEMORYALLOCATER "Free" USING BY REFERENCE PTR
                                  RETURNING RESULT.
END PROGRAM MEMFREE.
```

NCOBU00430

特殊な定数の書き方

ここでは、プログラム名やファイル名などのシステムで定められた名前を指定する、各種定数の書き方を説明します。

このセクションの内容

プログラム名定数

プログラム名段落(PROGRAM-ID)、CALL 文および CANCEL 文に定数指定でプログラム名を指定する場合、使用できる文字に制限はありません。ただし、定数の長さは 60 バイト以内でなければなりません。

NCOBU00432

原文名定数

COPY 文に記述する原文名定数には、登録集原文を格納したファイルの名前を以下の形式で記述します。

```
"[ドライブ名:] [パス名] ファイル名 [. 拡張子]"
```

ドライブ名

ドライブ名の指定を A から Z までの英字 1 文字で指定します。

ドライブ名が省略された場合、カレントドライブとみなします。

パス名

ファイルの格納先を以下の形式で指定します。

```
[¥][フォルダ名 [¥ フォルダ名] … ]¥
```

パス名が省略された場合、コンパイラオプション/copypath で指定されたフォルダまたはカレントフォルダ中のファイルとみなします。

ファイル名

ファイルの名前を指定します。

拡張子

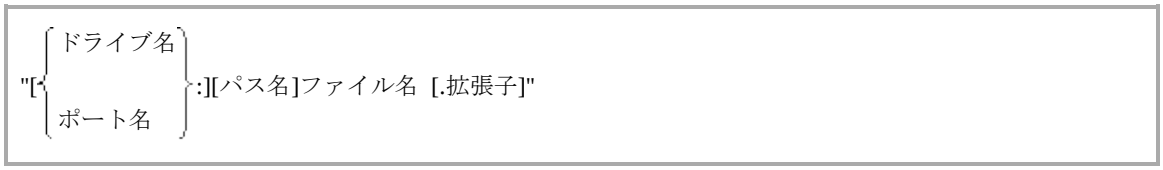
ファイルが拡張子を持つ場合、指定します。

拡張子は、"CBL"や"COB"以外でも可能です。

- "C:¥COPY¥A.CBL"
- "A.CPY"
- "C:¥COPY¥A"

ファイル識別名定数

ファイル管理記述項の ASSIGN 句に指定するファイル識別名定数は、実行時に処理するファイルを以下の形式で指定します。



ドライブ名

ドライブ名の指定を A から Z までの英字 1 文字で指定します。A から Z までの英字 1 文字でない場合は、ポート名として扱われます。

ドライブ名が省略された場合、カレントドライブとみなします。

ポート名

ポート名は順ファイルにだけ指定でき、ポート名を指定した場合、パス名およびファイル参照名の指定は無効となります。

ポート名でプリンタ装置を指定する場合、ポート名 LPT1、LPT2 または LPT3 を使用してください。

パス名

ファイルの格納先を以下の形式で指定します。

```
[¥][フォルダ名 [¥ フォルダ名] … ]¥
```

パス名が省略された場合、ファイル名で示されるファイルはカレントフォルダ中のファイルとみなされます。

ファイル名

ファイルの名前を指定します。

拡張子

ファイルの種類を区別するための、任意の文字列を指定します。

- "A:¥COBOL¥A.DAT"
- "LPT1:"
- "B.TXT"

NCOBU00434

外部名を指定するための定数

見出し部で定義する以下の名前には、AS 指定に定数を指定して外部名を付けることができます。AS 指定を省略すると内部名と外部名は同じになります。以下にいくつか例を記します。

プログラム名

```
PROGRAM-ID.  CODE-GET  AS  "XY1234".
```

クラス名

```
CLASS-ID.  特殊処理  AS  "SP-CLASS-001".
```

メソッド名

```
METHOD-ID.  値  AS  "VALUE".
```

この AS 指定に指定する定数は、最初の文字がアンダースコアで始まってはならないことを除けば、使用できる文字およびその構成規則に制限はありません。

オブジェクト指向と従来機能の組合せ

COBOL のオブジェクト指向では、従来の COBOL で使用していた機能であっても、クラス定義、メソッド定義などで使用できないものがあります。

ここでは、クラス定義やメソッド定義などで使用できない機能について説明します。

機能	説明
ANSI'85 規格の廃要素	以下の機能は、ANSI'85 規格の廃要素です。これらの機能をプログラム定義以外で使用することはできません。 ・ ALL 定数と数字項目または数字編集項目との関連付け ・ AUTHOR 段落、INSTALLATION 段落、DATE-WRITTEN 段落、DATE-COMPILED 段落および SECURITY 段落 ・ RERUN 句 ・ MULTIPLE FILE TYPE 句 ・ LABEL RECORD 句 ・ VALUE OF 句 ・ DATA RECORDS 句 ・ ALTER 文 ・ 手続き名-1 を省略した GO TO 文 ・ OPEN 文の REVERSED 指定 ・ 定数指定の STOP 文
翻訳用計算機段落および実行用計算機段落	プログラム定義以外の環境部構成節には、翻訳用計算機段落および実行用計算機段落を指定できません。
CHARACTER TYPE 句	プログラム定義およびメソッド定義以外のデータ部では、CHARACTER TYPE 句を指定できません。
EXTERNAL 句	プログラム定義およびメソッド定義以外のデータ部では、EXTERNAL 句を指定できません。
GLOBAL 句	プログラム定義以外のデータ部には、GLOBAL 句を指定できません。なお、スタティック定義およびオブジェクト定義のデータ部で宣言された名前は、すべて大域名として扱われます。
LINAGE 句	プログラム定義およびメソッド定義以外で定義したファイルには、LINAGE 句を指定できません。また、EXTERNAL 句を指定したファイルには指定できません。
特殊レジスタ PROGRAM-STATUS	プログラム定義以外では、特殊レジスタ PROGRAM-STATUS を使用できません。
翻訳オプション PGMNAME	COBOL G 資産の移行を支援する目的で提供しているため、 COBOL G の資産として存在しないクラス定義やメソッド定義などでは指定できません。

NCOBU00436

広域最適化

ここでは、コンパイラが行う広域最適化の内容および使用上の注意事項について説明します。

広域最適化では、1 入口・1 出口の手続き部を、記述順に実行されるような文の列(これを基本ブロックといいます)に分割します。そして、制御の移行およびデータの使用状態を解析し、主にループ(繰り返し実行される部分)に着目した最適化を行います。

このセクションの内容

広域最適化

最適化の内容について説明します。

広域最適化での注意事項

広域最適化での注意事項について説明します。

広域最適化

共通式の除去

以前に行われた演算や変換の結果が利用できる場合に、演算や変換を実行しないで、前の結果を保存しておいて、それを使用します。

例 1

```
77  添字１    PIC S99 COMP-5.
77  添字２    PIC S99 COMP-5.
77  添字３    PIC S99 COMP-5.
01  集団項目.
    02  項目１ OCCURS 25 TIMES.
        03  項目１１   PIC XX OCCURS 10 TIMES.
    02  項目２ OCCURS 35 TIMES.
        03  項目２１   PIC XX OCCURS 10 TIMES.
PROCEDURE DIVISION.
*   :
    MOVE SPACE TO  項目１１ (添字１, 添字２).      *>[1]
*   :
    MOVE SPACE TO  項目２１ (添字１, 添字３).      *>[2]
```

例 1 では、[1]と[2]の間で"添字１"の値が不変であれば、"項目１(添字１,添字２)"のデータ位置計算式(注 1)と、"項目２(添字１,添字３)"のデータ位置計算式(注 2)で、(添字１＊20)の部分が共通となるので、[2]は[1]の結果を使用するように最適化されます。

注 1 :

```
項目１ ＋ (添字１ － １) ＊ ２０ ＋ (添字２ － １) ＊ ２ ＝ 項目１ － ２２ ＋ 添字１ ＊ ２０ ＋ 添字２ ＊ ２
```

注 2:

```
項目２ ＋ (添字１ － １) ＊ ２０ ＋ (添字３ － １) ＊ ２ ＝ 項目２ － ２２ ＋ 添字１ ＊ ２０ ＋ 添字３ ＊ ２
```

例 2

```
77  計算結果１ PIC S9(9)  DISPLAY.
77  計算結果２ PIC S9(9)  DISPLAY.
77  数字１    PIC S9(4)  COMP-5.
77  数字２    PIC S9(4)  COMP-5.
PROCEDURE DIVISION.
*   :
    COMPUTE 計算結果１ =  数字１ *  数字２.      *>[1]
*   :
    COMPUTE 計算結果２ =  数字１ *  数字２.      *>[2]
```

例 2 では、[1]と[2]の間で"数字１"と"数字２"の値が不変であれば、(数字１ * 数字２)が共通となるので、[2]は[1]の結果を使用するように最適化されます。

不変式の移動

演算や変換がループ内にあり、ループ内外両方で行っても結果が変わらない場合、これをループ外に移動します。

例 3

```
77  添字           PIC S9(4)  COMP-5.
77  外部１０進項目   PIC S9(7)  DISPLAY.
01  集団項目.
  02  ２進項目      PIC S9(7)  COMP-5  OCCURS 20 TIMES.
*      :
PROCEDURE DIVISION.
   MOVE 1  TO  添字.
ループ開始.
   IF  ２進項目 (添字) = 外部１０進項目  GO TO ループ終了.  *>|
*      :                                                  *>|ループ
   ADD 1 TO  添字.                                        *>|
   IF  添字 <= 20 GO TO ループ開始.                       *>|
ループ終了.
```

例 3 では、ループ内で"外部１０進項目"の値が不変であれば、"２進項目(添字)"と比較するときの外部 10 進数を 2 進数に変換する処理は、ループ外に移動されます。

誘導変数の最適化

ループ内で、定数または値が不変な項目によってだけ再帰的に定義される項目を誘導変数といいます。誘導変数を使用している部分式がある場合、新しい誘導変数を導入することにより、添字計算のための乗算を加算に変更します。

例 4

```
77  添字           PIC S9(4) COMP-5.
01  集団項目.
   02  繰返し項目   PIC X(10) OCCURS 20 TIMES.
PROCEDURE DIVISION.
*     :
ループ開始.
   IF  繰返し項目(添字) =                *>[1]    |
*     :                                            |ループ
   ADD 1 TO 添字.                        *>[2]    |
   IF  添字 <= 20 GO TO ループ開始.      *>[3]    |
```

例 4 で、"添字"は誘導変数です(注 1)。 ここでは、新しい誘導変数(これを t とします)を導入し、"繰返し項目(添字)"のアドレス計算式(注 2)の中の乗算(添字*10)が t で置き換えられ、[2]の後に"ADD 10 TO t"が生成されます。さらに、ループ中で"添字"が他に使用されず、かつ、ループ中で計算した"添字"の値をループが出た後に使用していない場合、[3]は"IF t <= 200 GO TO ループ開始."で置き換えられ、[2]は削除されます。

注 1: ループ内で定数により再帰的にだけ定義されています。

注 2:

```
繰返し項目 ＋ （添字 － １） ＊ １０＝ 繰返し項目 － １０ ＋ 添字 ＊ １０
```

PERFORM 文の最適化

PERFORM 文は、その復帰機構として、戻り先のアドレスを退避、設定および復元するために、いくつかの機械命令に展開されます。そこで、PERFORM 文の出口に、その PERFORM 文以外で制御が渡る場合、復帰機構の機械命令のうちのいくつかが冗長となることがあります。これらの冗長な機械命令は削除されます。

隣接転記の統合

複数の英数字転記文があり、領域の連続した項目が同じように領域の連続した項目に転記される場合、これらを 1 つの文にまとめます。

例 5

```
      01 集団項目Ａ.
        02 基本項目Ａ１ PIC X(8).
        02 基本項目Ａ２ PIC X(8).
       PROCEDURE DIVISION.
      *     :
          MOVE "ABCDEFGH" TO 基本項目Ａ１.     *>[1]
      *     :
          MOVE "IJKLMLOP" TO 基本項目Ａ２.     *>[2]
```

例 5 において、[1]と[2]の間で"基本項目Ａ２"が参照されていなければ、 内部的には以下のように転記処理をまとめて行います。

- [1]の箇所では，以下に相当する転記処理を行う.

```
MOVE "ABCDEFGHIJKLMLOP" TO 集団項目Ａ
```

- [2]の箇所の，転記処理は削除される.

無駄な代入の除去

それ以降一度も明または暗に参照されないデータ項目への代入は、削除されます。

NCOBU00438

広域最適化での注意事項

翻訳オプション OPTIMIZE が有効な場合、コンパイラは広域最適化を行った目的プログラムを生成します。このときの注意事項を以下に示します。なお、詳細については、OPTIMIZE (広域最適化の扱い)を参照してください。

連絡機能を使用する場合

呼ばれるプログラムに対し、CALL "SUB" USING A,A.や CALL "SUB" USING A,B.（注）のように、同時に指定された複数のパラメタにおいて、領域の一部または全部を共有しているものがあります。呼ばれるプログラムでその内容を書き換えていると、呼ばれるプログラムの最適化により、意図したとおりの結果が得られない場合があります。

注: ただし、A と B は領域の一部を共有します。

広域最適化が行われない場合

以下のプログラムでは、広域最適化は行われません。

- 広域最適化の対象となる属性を持った項目および指標名が1つも定義されていないプログラム
- 区分化機能を使用しているプログラム

以下のプログラムでは、広域最適化の効果は少なくなります。

- 入出力操作を主とし、もともと CPU をあまり使わないプログラム
- 数字項目を使わず、英数字項目ばかりを使うプログラム
- 宣言部分から非宣言部分を参照しているプログラム
- 非宣言部分から宣言部分を参照しているプログラム
- 翻訳オプション TRUNC (桁落とし処理の可否)を指定しているプログラム
- コンパイラオプション/debug を指定しているプログラム

デバッグを行う場合

以下の注意が必要です。

- 広域最適化によって文の削除、移動および変更が行われるので、データ例外などのプログラム割込みの起こる回数や場所が変わることがあります。
- プログラム割込みなどで中断したとき、プログラム上の記述でデータ項目に値を設定していても、実際には設定されていないことがあります。
- 再帰的に定義される項目が内部 10 進項目または外部 10 進項目の場合、翻訳オプション NOTRUNC が指定されていると、意図したとおりにプログラムが動作しないことがあります。

例外処理

ここでは、NetCOBOL for .NET における例外処理について説明します。 なお、このトピックでは例外オブジェクトによる例外処理を中心に説明しています。 入出力誤り処理手続きの詳細については、入出力エラー処理を参照してください。

NetCOBOL for .NET では、COBOL 標準の例外処理(USE 文)と構造化例外処理(TRY 文)の二種類の例外処理をサポートしています。

このセクションの内容

COBOL 標準の例外処理(USE 文)

NetCOBOL for .NET における COBOL 標準の例外処理と従来の COBOL の例外処理の違いについて説明します。

構造化例外処理(TRY 文)

.NET の構造化例外処理について説明します。

ひとつの手続きにおいて、COBOL 標準の例外処理と構造化例外処理を併用することはできません。併用した場合、翻訳時にエラーとなります。

NCOBU00439

COBOL 標準の例外処理(USE 文)

ここでは、NetCOBOL for .NET における COBOL 標準の例外処理について説明します。

このセクションの内容

COBOL 標準の例外処理の概要

COBOL 標準の例外処理の概要について説明します。

例外の通知(throw)

COBOL 標準の例外処理において、独自に作成した例外を、例外として通知(throw)する方法について説明します。

例外の再通知(rethrow)

COBOL 標準の例外処理において、例外の再通知(rethrow)する方法について説明します。

なお、以上の説明は、COBOL の規格で定められている USE 文を使用した例外処理を対象としています。NetCOBOL for .NET では、 これとは別に TRY 文を使用した構造化例外処理(try-catch-finally ブロックを定義する形での例外処理)もサポートしています。詳しくは、COBOL 文法書を参照してください。

COBOL 標準の例外処理の概要

NetCOBOL for .NET における COBOL 標準の例外処理(例外オブジェクト処理)は、Windows 版 NetCOBOL とは動作が異なります。

Windows 版 NetCOBOL では例外オブジェクトの USE 手続きが実行された後は、同じ手続き内の、その USE が呼び出されるきっかけとなった文の直後に制御が戻りますが、NetCOBOL for .NET では、途中で RESUME 文が実行される場合を除いて、USE 手続きの最後に暗の EXIT METHOD(または EXIT PROGRAM)が実行され、そのメソッドまたはプログラムの処理は終了します。

また、発生した例外に対する USE が存在しない場合、Windows 版 NetCOBOL ではそこで異常終了しますが、NetCOBOL for .NET では、その例外が呼び出し元に通知されます。

USE が呼び出されるきっかけとなった文の直後に制御を戻す場合、翻訳オプション USEEXTRET を指定します。

なお例外オブジェクトに対する処理の動作は異なりますが、入出力誤り処理については、Windows 版 NetCOBOL と同じ動作となります。つまり NetCOBOL for .NET では、例外オブジェクト(CLR 例外)処理と入出力誤り処理の動作は異なります。

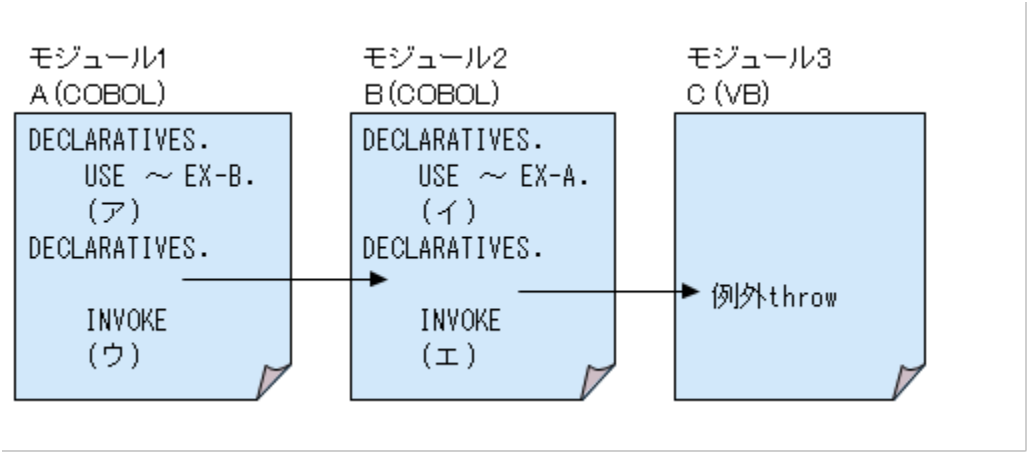
Windows 版 NetCOBOL の動作

条件	入出力誤り	例外オブジェクト
USE 文が存在しない場合	異常終了	異常終了
USE 手続き終了後の動作	USE への移行原因の文の直後に復帰し処理継続	USE への移行原因の文の直後に復帰し処理継続

NetCOBOL for .NET の動作

条件	入出力誤り	例外オブジェクト
USE 文が存在しない場合	COBOL ランタイム例外を通知(throw)	呼び出し元に例外を通知
USE 手続き終了後の動作	USE への移行原因の文の直後に復帰し処理継続	翻訳オプション USEEXTRET が無効な場合、暗の EXIT METHOD(または、EXIT PROGRAM)を実行。翻訳オプション USEEXTRET が有効な場合、USE への移行原因の文の直後に復帰し処理継続。

次に NetCOBOL for .NET における例外オブジェクト処理(CLR 例外処理)の動作を、具体的な例を用いて説明します。



上記のような呼び出し関係において、Cの手続きで例外が発生した(あるいは明に例外を投げた)場合の、条件ごとの制御の移行の仕方を示します。

Cの例外が EX-A である場合

- Cに EX-A の例外手続き(catch ブロック)が存在する場合
  1. Cの例外手続きを処理(注 1)
  2. Bの(エ)に復帰し、以降の手続き継続

  注 1: その try-catch-finally ブロック以降に手続きが存在するならば、例外手続き後はその手続きを継続
- Cに EX-A の例外手続き(catch ブロック)が存在しない場合
  1. 例外はCからBに通知(throw)
  2. Bに EX-A の USE が存在するので、そこに制御が移行
  3. (イ)において、暗黙の EXIT METHOD を実行(注 2)
  4. Aの(ウ)に復帰し、以降の手続きを継続

  注 2: 翻訳オプション USEEXTRET が有効な場合、USE への移行原因の文の直後に復帰して処理を継続

Cの例外が EX-B である場合

- Cに EX-B の例外手続き(catch ブロック)が存在する場合

  上の EX-A の場合と同様
- Cに EX-B の例外手続き(catch ブロック)が存在しない場合
  1. 例外はCからBに通知(throw)
  2. Bに EX-B の USE は存在しないので、例外はさらにAに通知(throw)
  3. Aには EX-B の USE が存在するので、そこに制御が移行
  4. (ア)において、暗黙の EXIT METHOD を実行(注 3)

  注 3: 翻訳オプション USEEXTRET が有効な場合、USE への移行原因の文の直後に復帰して処理を継続

Cの例外が EX-C の場合

- Cに EX-C の例外手続き(catch ブロック)が存在する場合

  上の EX-A, EX-B の場合と同様

- Cに EX-C の例外手続き(catch ブロック)が存在しない場合

  1. 例外はCからBに通知(throw)
  2. Bに EX-C の USE は存在しないので、例外はさらにAに通知(throw)
  3. Aにも EX-C の USE は存在しないので、例外はさらにその上に通知(throw)される。 呼び出し元の手続きが無ければ異常終了する

例外の発生による USE 手続きへの制御の移行後、その USE 手続きが最後まで終了する前に、間接的または直接的な再度の例外発生で、同じ USE 手続きへの制御の移行が生じた場合、以降の手続きは継続されずに COBOL ランタイム例外が通知されます。

ただし、翻訳オプション USEEXTRET が有効な場合、 以降の手続きは継続して最初の USE 手続きの最後まで終了した後に COBOL ランタイム例外が通知されます。

NCOBU00441

例外の通知(throw)

CLR が発生させる例外の他に、独自の例外を作成してそれを例外として通知(throw)することができます。呼び出し元の例外処理でその例外に対する処理を記述しておけば、その例外に対する回復処理を記述しておくことが可能です。

例外を通知(throw)するには、次の 2 つの方法があります。

- RAISE 文により、自手続きに例外を通知(throw)する。
- RAISING 指定の EXIT 文により、呼び出し元の手続きに例外を通知(throw)する。

「RAISE 文」および、「RAISING 指定の EXIT 文」の詳細については、COBOL 文法書の「11.8.3.6 EXIT 文」、「11.8.3.13 RAISE 文」を参照してください。

- 通知(throw)する例外オブジェクトは "System.Exception" を継承している必要があります。
- 通知(throw)する例外オブジェクトは RAISE 文、または、RAISING 指定の EXIT 文実行時にインスタンスが生成(NEW)されている必要があります。

例外処理の例:

```
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
          CLASS CLASS-ARGUMENTNULLEXCEPTION AS "System.ArgumentNullException"
          CLASS CLASS-EXCEPTION AS "System.Exception"
          CLASS CLASS-CONSOLE AS "System.Console"
          CLASS CLASS-STRING AS "System.String"
          CLASS CLASS-THROW AS "Exception1.Throw1"
          PROPERTY PROP-MESSAGE AS "Message"
          .
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 WK-MESSAGE OBJECT REFERENCE CLASS-STRING.
      01 WK-OBJ OBJECT REFERENCE CLASS-THROW.
      01 WK-STR OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
      DECLARATIVES.
      ERR SECTION.
          USE AFTER EXCEPTION CLASS-ARGUMENTNULLEXCEPTION.        *>[5]
              SET WK-MESSAGE TO PROP-MESSAGE OF EXCEPTION-OBJECT.  *>[6]
              INVOKE CLASS-CONSOLE "WriteLine" USING WK-MESSAGE.   *>[7]
              MOVE 1 TO PROGRAM-STATUS.                            *>[8]
              EXIT PROGRAM.                                        *>[9]
      END DECLARATIVES.
          SET WK-STR TO NULL.
          INVOKE CLASS-THROW "NEW" RETURNING WK-OBJ.
      *   SET WK-STR TO N"データ１".
          INVOKE WK-OBJ "Method1" USING WK-STR.                  *>[1]
          DISPLAY "正常終了".                                     *>[10]
```

```
       CLASS-ID. Throw1 AS "Exception1.Throw1".
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
          CLASS CLASS-ARGUMENTNULLEXCEPTION AS "System.ArgumentNullException"
          CLASS CLASS-STRING AS "System.String"
          CLASS CLASS-CONSOLE AS "System.Console"
```

```
       .
STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

END STATIC.

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. Method1 AS "Method1".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-EXCEPTION OBJECT REFERENCE CLASS-ARGUMENTNULLEXCEPTION.
LINKAGE SECTION.
01 ARG1 OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION USING BY VALUE ARG1 RAISING CLASS-ARGUMENTNULLEXCEPTION. *>[2]
   IF ARG1 = NULL THEN
      INVOKE CLASS-ARGUMENTNULLEXCEPTION "NEW" RETURNING WK-EXCEPTION     *>[3]
      EXIT METHOD RAISING WK-EXCEPTION                                    *>[4]
   END-IF.
   INVOKE CLASS-CONSOLE "WriteLine" USING ARG1.
END METHOD Method1.

END OBJECT.
END CLASS Throw1.
```

[プログラムの説明]

[1]: INVOKE 文で Method1 メソッドを呼出します。このとき、パラメタとして WK-STR を持ちます。

[2]: 手続き部見出しの RAISING 指定には、このメソッドの EXIT 文で発生させる可能性のある例外オブジェクトのクラスまたはそのクラスを継承するクラスの名前を指定します。

[3]: 渡されたパラメタが NULL オブジェクトの場合、System.ArgumentNullException クラスの例外を発生させます。

[4]: RAISING 指定の EXIT 文によって、呼出し元の手続きに例外を通知します。

[5]～[9]: 宣言部分(DECLARATIVES)に、例外処理を記述します。

[5]: 発生した例外オブジェクトのクラスが、System.ArgumentNullException または System.ArgumentNullException を継承する場合に実行される処理を記述します。

[6]: 発生した例外のエラーメッセージを取得します。

[7]: エラーメッセージ文字列を標準出力ストリームに書き込みます。

[8]: プログラムの復帰コードとして、PROGRAM-STATUS に 1 を設定しています。ここではこのように、復帰値を設定したり、例外を通知する処理を入れたりします。

[9]: 例外処理を終了します。NetCOBOL for .NET の USE 文では、EXIT 文を記述しなくても暗黙の EXIT 文が実行されます。(*1)

*1:翻訳オプション USEEXTRET が有効な場合、暗黙の EXIT 文は実行されず、 USE への移行原因の文の直後([10])に復帰し処理を継続します。

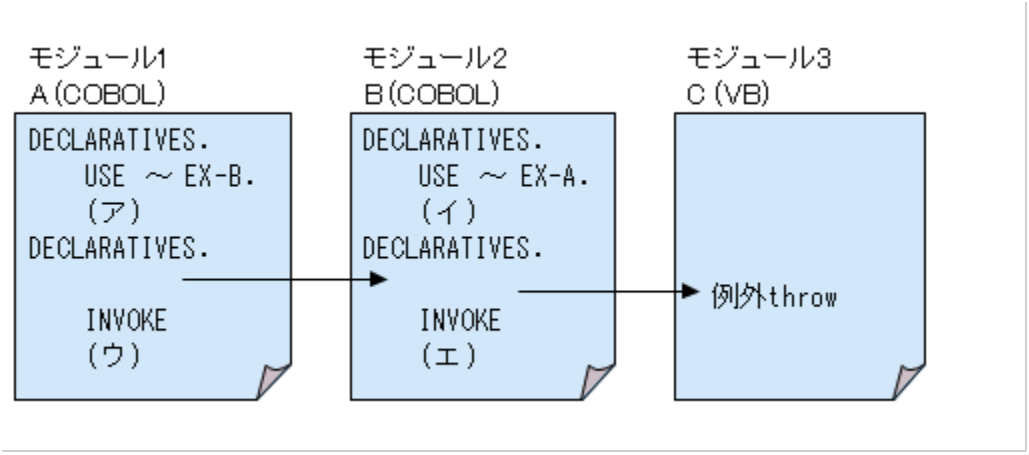
NCOBU00442

例外の再通知(rethrow)

例外はどこかで例外処理(catch)を行えば、それ以降は呼び出し元へ通知しません。しかし、例外処理(catch)したUSE 手続きにおいて、次のような RAISING 指定の EXIT 文が実行された場合は、その例外はさらに呼び出し元の手続きに通知されます。

- EXIT METHOD RAISING EXCEPTION-OBJECT
- EXIT PROGRAM RAISING EXCEPTION-OBJECT

モジュール 2 の USE 手続きにおいて、上記のような RAISING 指定の EXIT 文を記述し、それが実行された場合は、モジュール 1 に例外が通知され、モジュール 1 の例外処理が実行されます。



注意

例外処理で例外を処理しない場合と、処理して再通知(rethrow)する場合では、通知される例外の発生場所が異なります。処理して再通知(rethrow)した場合は、再通知(rethrow)した場所が例外の発生場所として通知されます。

構造化例外処理(TRY 文)

ここでは、NetCOBOL for .NET における構造化例外処理について説明します。

CLR は標準の例外処理機構として構造化例外処理をサポートしています。構造化例外処理は、手続きを 3 つのブロックに分けて定義し、例外／終了処理を制御する手法です。

- try ブロック

  例外が監視される手続きコード。

- catch ブロック

  try ブロックで例外が発生した場合に実行される例外手続きコード。

- finally ブロック

  上記 try-finally ブロックから制御が外に出る場合に必ず実行される終了処理手続きコード。

構造化例外処理には、次の特徴があります。

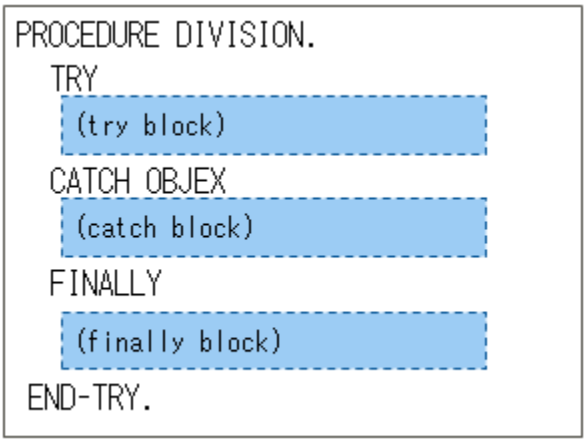
- 例外が監視される手続きと例外手続きとを明確に分離して定義できる。
- 例外処理のネスト(入れ子)構造を構築できる。
- 例外の発生の有無に関わらず、try-catch から脱出する際には必ず実行される終了処理を定義できる。

このセクションの内容

NCOBU00982

TRY 文

TRY 文は COBOL の手続きで、try、catch、finally の各ブロックを定義します。



TRY 文内での発生事象とその後の動作の対応を以下に示します。

発生事象	その後の動作
try ブロックが正常終了	1) finally ブロックが存在するならば制御を移行し、手続きを実行する。 2) END-TRY の直後に制御を移行する。
try ブロック内で例外発生	【発生例外に対応する catch ブロックが存在する場合】 1) 対応する catch ブロックに制御を移行し、手続きを実行する。 2) finally ブロックが存在するならば制御を移行し手続きを実行する。 3) END-TRY の直後に制御を移行する。 【発生例外に対応する catch ブロックが存在しない場合】 1) finally ブロックが存在するならば制御を移行し、手続きを実行する。 2)TRY 文の外に発生例外を通知する。
catch ブロック内で例外発生	1) finally ブロックが存在するならば制御を移行し、手続きを実行する。 2) TRY 文の外に発生例外を通知する。
finally ブロック内で例外発生	TRY 文の外に発生例外を通知する。

CATCH 指定および FINALLY 指定は、必要に応じて一方を省略することができます。つまり CATCH 指定のみ、あるいは FINALLY 指定のみの TRY 文も記述できます。

CATCH 指定

CATCH 指定は、try ブロックで例外が発生した場合の例外処理手続き(catch ブロック)を定義します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-DIVIDEBYZEROEXCEPTION    AS "System.DivideByZeroException"
   CLASS CLASS-INDEXOUTOFRANGEEXCEPTION AS "System.IndexOutOfRangeException"
   CLASS CLASS-MEMBER                   AS "MemberClass"
   PROPERTY P-MESSAGE                   AS "Message"
  .
```

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-DIVIDEBYZERO   OBJECT REFERENCE CLASS-DIVIDEBYZEROEXCEPTION.
01 WK-INDEXOUTOFRANGE OBJECT REFERENCE CLASS-INDEXOUTOFRANGEEXCEPTION.
01 WK-DATA1           PIC S9(9) COMP-5.
01 WK-INDEX           PIC S9(9) COMP-5.
01 WK-EXCEPTION-MESSAGE PIC X(150).
PROCEDURE DIVISION.
   TRY
     COMPUTE WK-DATA1 = WK-DATA1 / CLASS-MEMBER::"GetNumber"(WK-INDEX)
   CATCH WK-DIVIDEBYZERO                                          *> [1]
     SET  WK-EXCEPTION-MESSAGE  TO  P-MESSAGE OF WK-DIVIDEBYZERO
     DISPLAY "Exception message : " WK-EXCEPTION-MESSAGE
   CATCH WK-INDEXOUTOFRANGE                                       *> [1]
     SET  WK-EXCEPTION-MESSAGE  TO  P-MESSAGE OF WK-INDEXOUTOFRANGE
     DISPLAY "Exception message : " WK-EXCEPTION-MESSAGE
   CATCH                                                          *> [2]
     DISPLAY "--  Exception occurred !! --"
   FINALLY
     INVOKE  CLASS-MEMBER "Initial"
   END-TRY.
```

［1］：対象を特定の例外に限定する catch ブロックを定義する場合、その例外の型のオブジェクト参照データ項目を指定した CATCH 指定を記述します。try ブロックで例外が発生したとき、その例外の型が上記オブジェクト参照データ項目の型あるいはその派生型であれば、この catch ブロックに制御が移行します。 このとき CATCH 指定のオブジェクト参照データ項目には、発生した例外のオブジェクトが設定されますので、手続きで例外オブジェクトを参照することができます。

［2］：オブジェクト参照データ項目の指定を省略した CATCH 指定も記述できます。この場合の catch ブロックは、“System.Exception”の派生型の全ての例外が対象となります。なお、この catch ブロックにおいては、手続きで例外オブジェクトを参照することはできません。

ひとつの try ブロックに対し、複数の例外手続きを定義する場合、上記例のように CATCH 指定を複数記述します。この場合、発生した例外に対し、複数の catch ブロックが該当する可能性がありますが、その場合は記述順で検索したときに最初に見つかる catch ブロックだけが実行されます。

FINALLY 指定

FINALLY 指定は、終了処理手続き(finally ブロック)を定義します。

finally ブロックは、その FINALLY 指定が書かれた TRY 文の範囲から、その外に制御が移行する際に実行されます。TRY 文の内から外への制御の移行が発生する契機は、手続きの正常な終了、EXIT TRY 文の実行、例外の発生など、いくつかのケースがありますが、そのいずれかに関わらず、finally ブロックは必ず実行されます。

FINALLY 指定の記述例：

```
PROCEDURE DIVISION.
   OPEN  INPUT MEMBER-SALES-FILE.
   TRY
     PERFORM
       READ  MEMBER-SALES-FILE
            AT END EXIT TRY
       END-READ
      CALL "Monthly-Report" USING MEMBER-SALES-REC
      COMPUTE  WK-TOTAL-SALES = WK-TOTAL-SALES + MEMBER-SALES
     END-PERFORM
   CATCH
     DISPLAY "--  Exception occurred !!  --"
   FINALLY
     CLOSE MEMBER-SALES-FILE
   END-TRY.
```

TRY 文のネスト

TRY 文は、TRY 文の中にも書くことができます。

```
PROCEDURE DIVISION.
    TRY
      CALL "GET-INFORMATION" RETURNING WK-INFORMATION
      TRY
        CALL "ADD-DATA"  USING WK-INFORMATION
        CALL "MAIN-PROC"
      CATCH OBJEX
        CALL "RECORVER" USING OBJEX
      END-TRY
    FINALLY
      CALL "FINALIZE"
    END-TRY.
```

TRY 文における例外処理の制御は、CLR の例外機構により行われますが、CLR はブロック間の制御の移行について、いくつか制限を設けています。その関係上 TRY 文内部では、COBOL の一部機能が使用できません。具体的な機能については、COBOL 文法書の「11.8.3.18 TRY 文」を参照してください。

Visual C#や Visual Basic の“using”ステートメントを COBOL に書き換える場合

IDisposable インタフェースを実装したオブジェクト(スコープの出口でリソースを解放する必要があるオブジェクト)を使用する場合、Visual C#や Visual Basic では通常 using ステートメントを使用しますが、NetCOBOL for .NET では using ステートメントに相当する構文がありません。COBOL に書き換える場合、以下のように TRY 文を使用してください。

Font クラスを使用した例

Visual C#

```
using (Font font1 = new Font("Arial", 10.0f))
{
  // font1 を使う処理
}
```

COBOL (NetCOBOL for .NET)

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS CLASS-FONT AS "System.Drawing.Font"
    INTERFACE INTERFACE-IDISPOSABLE AS "System.IDisposable"
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FONT1     OBJECT REFERENCE CLASS-FONT.
01 DISPOSER   OBJECT REFERENCE INTERFACE-IDISPOSABLE.
01 FAMILYNAME PIC X(20) VALUE "Arial".
01 EMSIZE    COMP-1   VALUE 10.0.
PROCEDURE DIVISION.
    TRY
        SET FONT1 TO CLASS-FONT::"new" (FAMILYNAME, EMSIZE)
        *> FONT1 を使う処理
    FINALLY
        SET DISPOSER TO FONT1 AS INTERFACE-IDISPOSABLE
        INVOKE DISPOSER "Dispose"
    END-TRY.
END PROGRAM MAIN.
```

例外の通知(throw)

TRY 文の内部で、手続き上で任意に生成した例外オブジェクトを例外として通知する(throw)ことができます。この通知には、RAISE 文を使用します。

RAISE 文は、COBOL 標準の例外処理にも存在する機能ですが、TRY 文の内部では、多少扱いが異なります。COBOL 標準の例外処理の場合、同じ手続き内に RAISE 文が通知する例外に対応する USE 文が存在することが前提になりますが(存在しない場合は CONTINUE 文として扱われます)、TRY 文内部における RAISE 文では、常に指定されたオブジェクトを例外として通知します。

TRY 文内部での RAISE 文の使用例：

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
  SPECIAL-NAMES.
    SYMBOLIC CONSTANT
    MAX-VALUE IS 500000000
    .
REPOSITORY.
    CLASS CLASS-EXCEPTION-A  AS "Exception-A"
    CLASS CLASS-EXCEPTION-B  AS "Exception-B"
  .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-EXCEPTION-A    OBJECT REFERENCE CLASS-EXCEPTION-A.
01 WK-EXCEPTION-B    OBJECT REFERENCE CLASS-EXCEPTION-B.
01 WK-EXCEPTION      OBJECT REFERENCE CLASS-EXCEPTION-A.
LINKAGE SECTION.
01 PARM-1    USAGE BINARY-LONG.
PROCEDURE DIVISION  USING PARM-1.
    TRY
      IF  PARM-1 > MAX-VALUE   THEN
        INVOKE CLASS-EXCEPTION-A "NEW" USING PARM-1 RETURNING WK-EXCEPTION   *> [1]
        RAISE  WK-EXCEPTION                                                  *> [1]
      ELSE
        CALL "DATA-INPUT" USING PARM-1
      END-IF
    CATCH WK-EXCEPTION-A                                  *> [2]
      DISPLAY "--  Exception-A occurred !! --"
      INVOKE  CLASS-EXCEPTION-B "NEW" RETURNING WK-EXCEPTION-B
      SET  EX-OBJ OF WK-EXCEPTION-B  TO  WK-EXCEPTION-A
      RAISE  WK-EXCEPTION-B                               *> [3]
    FINALLY
      DISPLAY "--  Process end  --"
    END-TRY.
```

[プログラムの説明]

［1］: 生成した任意の例外クラスのオブジェクトを RAISE 文に指定します。

［2］: try ブロック内で RAISE 文を実行したときには、通知する例外に対応する catch ブロックが存在するならば catch され、その catch ブロックの手続きが実行されます。この手続きにおいて例外の再通知(rethrow)が行われない限り、ここで例外処理は完結します。対応する catch ブロックが存在しない場合は、例外は TRY 文外部に通知(throw)されます。

［3］: catch ブロックまたは finally ブロックで RAISE 文を実行した場合、例外はそれが所属する TRY 文の外部に通知(throw)されます。

- 通知(throw)する例外オブジェクトは“System.Exception”クラスを継承している必要があります。
- 通知(throw)する例外オブジェクトが NULL オブジェクトの場合、実行時に“System.NullReferenceException”クラスの例外が発生します。

例外の再通知(rethrow)

例外処理は catch ブロックの手続きが完了した時点で完結し、そこで例外の伝播は止まります。しかし処理によっては、一旦 catch した例外をさらにその外側に通知したい、つまり例外を再通知(rethrow)したい場合もあります。例えば次のような場合です。

- TRY 文のネストにおいて、内側の TRY 文で catch した例外を、外側の TRY 文にも通知したい。
- プログラム／メソッドの呼出しにおいて、子の TRY 文で catch した例外を、親にも通知したい。

この例外の再通知(rethrow)には、例外の通知(throw)と同様に RAISE 文を使用します。ただし再通知(rethrow)の場合、オブジェクト参照データ項目の指定を省略します。再通知(rethrow)の RAISE 文は、catch ブロックでのみ使用することができます。

再通知(rethrow)の RAISE 文の使用例：

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-EXCEPTION-A  AS "Exception-A"
  .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-EXCEPTION-A    OBJECT REFERENCE CLASS-EXCEPTION-A.
LINKAGE SECTION.
01 PARM-1     USAGE BINARY-LONG.
PROCEDURE DIVISION  USING PARM-1.
   TRY
     CALL "DATA-INPUT" USING PARM-1           *> [1]
   CATCH WK-EXCEPTION-A                       *> [2]
     DISPLAY "--  Exception-A occurred !! --"
     RAISE                                    *> [3]
   FINALLY
     DISPLAY "--  Process end  --"
   END-TRY.
```

[プログラムの説明]

［1］: try ブロックにおいて、プログラム“DATA-INPUT”を呼び出しています。 “DATA-INPUT”は、“Exception-A”クラスの例外を通知する可能性があります。

［2］:“Exception-A”クラスの例外を処理する catch ブロックを定義するための CATCH 指定です。

［3］: 再通知(rethrow)の RAISE 文によって、一旦 catch した例外が TRY 文の外側に再通知(rethrow)されます。

なお、単に catch した例外オブジェクトを TRY 文の外側に通知するだけであれば、次のように書くことでも実現できます。

```
CATCH WK-EXCEPTION-A
  DISPLAY "--  Exception-A occurred !! --"
  RAISE WK-EXCEPTION-A
```

ただしこの場合は、あくまでも通知(throw)の RAISE 文の扱いとなりますので、元々は“DATA-INPUT”から通知された例外であることが、通知先に情報として伝わりません。

CATCH 指定のオブジェクト参照データ項目を、その catch ブロックにおいて更新しても、再通知(rethrow)の RAISE 文の実行には影響を与えません。再通知(rethrow)では、必ずその catch ブロックに移行する契機となった例外が対象となります。

NCOBU00985

TRY 文からの脱出

try ブロックまたは catch ブロックの手続きでは、任意の箇所で手続きを完了して TRY 文の外に脱出することができます。この脱出には EXIT TRY 文を使用します。ただし、EXIT TRY 文は finally ブロックでは使用できません。

EXIT TRY 文の使用例：

```
       PROCEDURE DIVISION.
          OPEN  INPUT MEMBER-SALES-FILE.
          TRY
            PERFORM                              *> [1]
              READ  MEMBER-SALES-FILE
                 AT END EXIT TRY                 *> [2]
              END-READ
             CALL "Monthly-Report" USING MEMBER-SALES-REC
             COMPUTE  WK-TOTAL-SALES = WK-TOTAL-SALES + MEMBER-SALES
            END-PERFORM
          CATCH
            DISPLAY "--  Exception occurred !!  --"
          FINALLY                                *> [3]
            CLOSE MEMBER-SALES-FILE
          END-TRY.
```

[プログラムの説明]

［1］: try ブロックで、"MEMBER-SALES-FILE"を読込み、1 件 1 件のレコードデータを基に特定の処理を行います。これを最後のレコードに対する処理が終了するまで繰り返します。

［2］: READ 文の実行が AT END 条件になったならば、EXIT TRY 文を実行して、TRY 文から脱出します。

［3］: EXIT TRY 文の実行による TRY 文からの脱出であっても、FINALLY 指定が存在するならば finally ブロックが実行されます。

ネストした TRY 文で､その内側の TRY 文中の EXIT TRY 文が実行された場合､脱出するのは内側の TRY 文だけです。

動的構造の例外処理

プログラムの呼出しにおいて以下の条件を満たす場合、動的構造になります。

- CALL 文の一意名呼出しで呼び出す場合
- 翻訳オプション DLOAD を有効にして翻訳した場合

動的構造で例外が発生するかもしれない手続きの呼び出しを、構造化例外処理を使って処理する場合、例外処理(catch)する対象に以下の例外を追加する必要があります。

- System.Reflection.TargetInvocationException

また、例外処理(catch)した例外が目的の例外かどうかを検査する構文を追加して、該当する処理を行わせます。

例： "System.DivideByZeroException"が発生する処理を呼び出す場合

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS EXCEPTION-DIVIDEBYZERO
        AS "System.DivideByZeroException"
   CLASS EXCEPTION-TARGETINVOCATION
         AS "System.Reflection.TargetInvocationException"
    PROPERTY INNEREXCEPTION AS "InnerException"
   .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 EXC-DBZ OBJECT REFERENCE EXCEPTION-DIVIDEBYZERO.
01 EXC-TI  OBJECT REFERENCE EXCEPTION-TARGETINVOCATION.
01 SUBPGM  PIC X(4) VALUE "SUB".
PROCEDURE DIVISION.
   TRY
      CALL SUBPGM
   CATCH EXC-DBZ
      DISPLAY "DIVIDE BY ZERO"
   CATCH EXC-TI
       IF INNEREXCEPTION OF EXC-TI IS EXCEPTION-DIVIDEBYZERO
          DISPLAY "INNER EXCEPTION DIVIDE BY ZERO"
       END-IF
   END-TRY
   STOP RUN.
END PROGRAM MAIN.
```

NCOBU00443

動的構造

動的構造のサポート

NetCOBOL for .NET では、以下の方法で呼び出すプログラムと呼び出されるプログラム間の動的構造をサポートします。

- 翻訳オプション DLOAD
- CALL 文の一意名呼出し

動的構造では、呼び出されるプログラムは、実行時に動的に決定されます。このため、グループで開発しているアプリケーションの一部が未完成であったり、またはプロトタイプしかなかったりする場合でも、できている部分だけを選んでアプリケーションを実行することができます。

プログラムの配置

呼び出されるプログラムが、どの DLL またはアセンブリに存在するかは、エントリ情報ファイルを使って指定します。

DLL 名を指定する場合、以下のように指定します。

- COBOL 実行環境を確立したプログラムが存在するフォルダからの相対パス。COBOL 実行環境を確立するプログラムとは、一般的には、最初に実行される COBOL の EXE、または DLL です。
- 絶対パス

アセンブリ名を指定する場合、厳密アセンブリ名、または厳密アセンブリ名の部分名を指定します。厳密な名前は、単純テキスト名、バージョン番号、カルチャ情報 (設定されている場合) から成るアセンブリの識別子と、公開キーおよびデジタル署名から構成されます。つまり、厳密な名前を持つアセンブリを作成した場合、単純テキスト名とは別に、バージョン、公開キー、およびその性質を指定することができます。 結果として、文字列によるバージョンの指定もできます。

動的構造では、呼び出しの延長で例外が発生した場合、発生した例外は以下の例外でラッピングされます。

- System.Reflection.TargetInvocationException

呼び出しの延長で発生した例外を例外処理(catch)する場合、例外処理(catch)の対象に上記の例外を加えてください。発生した例外の確認方法については、構造化例外処理(TRY 文)の動的構造の例外処理を参照してください。

関連トピックス

エントリ情報

エントリ情報の概要について説明しています。

例外処理

NetCOBOL for .NET での例外処理について説明しています。

他形式の外部 10 進互換モード

SEPARATE 指定のない符号付外部 10 進項目の符号部の表現形式は標準化されていない ため、各ベンダによって表現形式が異なります。NetCOBOL システムでは、他形式の表 現形式を扱うための翻訳オプションと変換関数を用意しています。

- NetCOBOL システムでサポートしている符号形式

  NetCOBOL システムでサポートしている符号形式の内部表現を以下に示します。

[正の値]	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
富士通形式	40	41	42	43	44	45	46	47	48	49
MicroFocus COBOL 形式	30	31	32	33	34	35	36	37	38	39
88 コンソーシアム形式	7b	41	42	43	44	45	46	47	48	49

[負の値]	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9
富士通形式	50	51	52	53	54	55	56	57	58	59
MicroFocus COBOL 形式	70	71	72	73	74	75	76	77	78	79
88 コンソーシアム形式	7d	4a	4b	4c	4d	4e	4f	50	51	52

  88 コンソーシアム形式：Btrieve ファイルなどで採用されている形式です。

- 翻訳オプション

  翻訳オプションにより、翻訳単位ごとに使用する形式を指定することができます。

  翻訳オプションの指定形式

```
          ì FJ  ü
DECIMAL（í MF  ý  )
          î 88  þ
```

  SEPARATE 指定無しの外部 10 進項目の内部表現を、富士通形式(DECIMAL(FJ)) とするか、MicroFocus COBOL 互換形式(DECIMAL(MF))とするか、88 コンソーシアム 互換形式(DECIMAL(88))とするかを指定します。省略時は富士通形式となります。

  DECIMAL オプションの指定が異なるプログラムの呼び出しは、実行時にチェックされます。[参照]@CBR_DECIMAL_OPTION（DECIMAL オプションチェックの指定）

- 変換関数

  以下の関数を使用することで、データ項目単位に形式を変換することができます。

関数名	意味
#DECFJTOMF	富士通形式のデータを MicroFocus COBOL 互換形式に変換します。
#DECFJTO88	富士通形式のデータを 88 コンソーシアム互換形式に変換します。
#DECMFTOFJ	MicroFocus COBOL 互換形式のデータを富士通形式に変換します。
#DECMFTO88	MicroFocus COBOL 互換形式のデータを 88 コンソーシアム互換形式に変換します。
#DEC88TOFJ	88 コンソーシアム互換形式のデータを富士通形式に変換します。
#DEC88TOMF	88 コンソーシアム互換形式のデータを MicroFocus COBOL 互換形式に変換します。

記述形式

```
CALL  "関数名"  USING [BY REFERENCE] 一意名
```

機能概要

- 一意名または一意名中に含まれる、すべての SEPARATE 指定のない符号付き外部 10 進項目の内部形式を変換します。
- 一意名が基本項目の場合、その項目が SEPARATE 指定のない符号付き外部 10 進項目でなければ、変換処理は行われません。
- 一意名が集団項目で、かつ従属する項目に以下の項目が含まれている場合、その項目は変換の対象とはなりません。
  - SEPARATE 指定のない符号付き外部 10 進項目ではない項目
  - REDEFINES 句が指定された項目、およびその項目に従属する項目
  - RENAMES 句が指定された項目
- 以下の場合、実行結果は保証されません。
  - CALL 文で呼ぶプログラム名が一意名で指定されており、その一意名により関数名が指定されている
  - 上記記述形式とは異なる形式で、関数を呼び出している(USING 句に複数の一意名が指定されている、BY CONTENT 句が指定されているなど)
  - 一意名が集団項目であり、従属する項目に DEPENDING 指定付きの OCCURS 句を持つ項目が存在する
  - 一意名が定数節に定義されたデータ項目である
  - 一意名が部分参照付けされたデータ項目である
  - 変換関数の実行で翻訳オプションに指定された形式と異なる形式のデータが格納されているデータ項目に対して、字類条件による字類検査を行っている
  - 変換関数の実行で翻訳オプションに指定された形式と異なる形式のデータが格納されているデータ項目が存在するプログラムに対して、翻訳オプション CHECK(NUMERIC)が指定されている

使用上の注意

本関数呼出しを使用して、ビルド時に関数名の外部参照エラーとなった場合は、呼び出す関数名が正しくないか、本関数の使い方を機能概要で示した規則に違反して使用されています。

セキュリティ

ネットワーク環境では、不正なアクセスによるシステムおよび資源の改ざんや破壊、情報の漏えいなどの危険があります。 このため、システムの構築にあたっては、Web サーバのユーザ認証機能と暗号化通信機能を使用し、さらに、アプリケーションでユーザ制限を行うなど、自己防衛手段を講じる必要があります。

このセクションの内容

NCOBU01155

資源の保護

プログラム、データに関する資源(データベースファイル、入出力ファイル等)およびプログラムの動作に必要な各種の定義・情報ファイルは、OS の機能やプログラムによるアクセス制限を行い、不正なアクセスや改ざんから保護してください。 特に重要な資源は、信頼できる安全な業務セグメント（イントラネット環境）内に保持してください。

なお、Web サーバ上に配置した、プログラムおよびプログラムの動作に必要な各種の情報ファイルについても、OS の機能によるアクセス制限を行い、不正なアクセスや改ざんから保護してください。

アプリケーション作成のための指針

セキュリティを考慮したアプリケーションを作成するための参考にしてください。

事前確認と処理結果の通知

対話・応答を行う処理の場合、重要なデータへのアクセスや処理については、事前の確認および処理結果を通知して、誤った処理を検知できる設計を行ってください。 また、ログを記録すると処理の解析に役立ちます。

匿名性

ユーザの実名、実物を識別できるデータについては、特に漏えいの危険性を考慮してください。

インタフェースの検査

外部インタフェースについては、バッファオーバフロー(バッファオーバラン)やクロスサイトスクリプティングなどを考慮して、セキュリティホールの作り込みを防止してください。 バッファオーバフローを防止するためには、外部インタフェースの入力データの長さ、型や属性などの検査が有効です。 クロスサイトスクリプティングは、動的に生成されたページ中に意図しないタグが含まれないようにする事で防止できます。 例えば、出力時にメタキャラクタをエスケープする方法があります。

マネージコードでは、通常バッファオーバーフローは発生しません。 ネイティブコードとの連携部分等に注意してください。 ASP.NET などの Web アプリケーションのためのフレームワークには、クロスサイトスクリプティングを防ぐ機構が組み込まれています。 なるべくこれらのフレームワークを利用することを検討してください。

クロスサイトスクリプティング

クロスサイトスクリプティングとは、入力データをプログラムでチェックせず出力データとして HTML に埋め込んでいる場合、入力データに JavaScript などのスクリプトコードが含まれると、その HTML を表示したクライアントでスクリプトが実行されるというものです。 悪意のあるスクリプトコードが入力されることにより、Cookie データの盗聴や改ざんが行われ、Cookie による認証がパスされたり、セションの乗っ取りが行われたりする危険があります。また、スクリプトコード以外にも HTML タグを使って、意図していたものとは異なる HTML を表示させられる危険もあります。

繰り返し実行

同じ接続先からの一定時間内でのリクエスト数を制限するなどの考慮をしてください。

監査ログの記録

Web サーバや OS の監査ログ機能、およびアプリケーションによるログ出力処理の作成などにより、セキュリティに関するイベントを記録して、セキュリティ侵害が発生した場合の分析や追跡方法を考慮してください。

セキュリティのためのルールの制定

セキュリティに関する脆弱な処理がない堅牢なアプリケーションを作成するためには、セキュリティ侵害の脅威から保護すべき重要な資源を特定し、資源のアクセスやインタフェース設計のために特定のルールを制定することが有効です。

NCOBU00444

.NET プラットフォームでの COBOL プログラミング

ここでは、.NET プラットフォームで動作するアプリケーションを作成するために必要な、.NET Framework のデータ型と.NET 特有の COBOL のプログラミング技法について説明します。

このセクションの内容

[.NET 言語とトランザクション連携]

NetCOBOL for .NET と.NET 言語において、トランザクションを共有する方法について説明します。

[XML ドキュメントコメント]

XML ドキュメントコメントの記述方法について説明します。

NCOBU00445

.NET Framework のデータ型

.NET Framework ではすべてのデータをオブジェクトとして扱います。単なるメモリ領域という扱いはしません。各データに対して厳密に型付けを行い、 さらに IL を通してその用途を追跡し、 コードが安全かどうか確認できるようにしています。

NetCOBOL for .NET は.NET Framework 上で動作するプログラムを生成します。 NetCOBOL for .NET で作成したプログラムは.NET Framework の型システムの上で動くプログラムになります。 このため、.NET Framework の型システムを理解することは重要です。

また、.NET Framework 上では多くの言語が連携して動作します。 言語間の連携のために各言語での表現の違いを乗り越えたり、各言語から共通に利用する.NET Framework クラスライブラリのドキュメントを読んで理解する上でも、 データ型を.NET Framework の型として理解することは重要になります。

ここでは、.NET Framework の型システムと、 NetCOBOL for .NET のデータ型との対応について説明します。

このセクションの内容

.NET Framework のデータ型と COBOL のデータ型の対応

ここでは、 NetCOBOL for .NET で扱うデータ型と .NET Framework のデータ型の関係について説明します。

NetCOBOL for .NET で扱うデータ型は、 .NET Framework の型システムとの関係から、 以下のように分類されます。

.NET 基本データ型、その他の.NET データ型および COBOL 独自データ型については、それぞれの説明を参照してください。

NCOBU00447

.NET 基本データ型

.NET 基本データ型とは

多くの言語には組み込みデータ型という概念があります。.NET Framework において組み込みデータ型に相当するものが.NET 基本データ型です。

組み込みデータ型は、各言語であらかじめ定義され、その他のデータ型（*1）を組み立てる際の最小要素になります。 各言語は、組み込みデータ型に対する操作(代入や演算、定数からのデータの生成など)を、基本機能として提供します。 組み込みデータ型には、整数、浮動小数点、文字などがあります。文字列を組み込みデータ型に含む言語もあります。 多くの言語では、組み込みデータ型をオブジェクトとして扱いません。

言語における組み込みデータ型と同様に、 .NET 基本データ型は.NET Framework の他の型を組み立てる際の最小要素になります。 .NET Framework はすべてのデータをオブジェクトとして扱うので、 .NET 基本データ型も他のクラスと同様に型名を持ちます。

*1: 組み込みデータ型以外のデータ型。言語によっては複合型とも呼びます。

COBOL データ型との対応

NetCOBOL for .NET での、 .NET 基本データ型と COBOL 言語としてのデータ型との対応を以下に示します。

.NET プラットフォーム対応の多くの言語では、 .NET 基本データ型をその言語の組み込みデータ型に対応付けています。 NetCOBOL for .NET では、 いくつかの.NET 基本データ型は非オブジェクトデータ型として扱いますが、 オブジェクトとして扱うデータ型もあります。

.NET 基本データ型と COBOL データ型の対応

.NET Framework での型名	COBOL 言語上の扱い	説明	CLS 準拠	値型
System.Byte	USAGE BINARY-CHAR UNSIGNED	8 ビット符号なし整数	Yes	Yes
System.SByte	"System.SByte"型オブジェクト参照	8 ビット符号付き整数	No	Yes
System.Int16	USAGE BINARY-SHORT SIGNED	16 ビット符号付き整数	Yes	Yes
System.Int32	USAGE BINARY-LONG SIGNED	32 ビット符号付き整数	Yes	Yes
System.Int64	USAGE BINARY-DOUBLE SIGNED	64 ビット符号付き整数	Yes	Yes
System.UInt16	"System.UInt16"型オブジェクト参照	16 ビット符号なし整数	No	Yes
System.UInt32	"System.UInt32"型オブジェクト参照	32 ビット符号なし整数	No	Yes
System.UInt64	"System.UInt64"型オブジェクト参照	64 ビット符号なし整数	No	Yes
System.Single	USAGE COMP-1	単精度浮動小数点	Yes	Yes
System.Double	USAGE COMP-2	倍精度浮動小数点	Yes	Yes
System.Boolean	"System.Boolean"型オブジェクト参照	ブール値（真または偽）	Yes	Yes
System.Char	PIC N	Unicode 文字（16 ビット）	Yes	Yes

System.Decimal	"System.Decimal"型オブジェクト参照	96 ビット 10 進値	Yes	Yes
System.IntPtr	"System.IntPtr"型オブジェクト参照	基になるプラットフォームによってサイズが決まる符号付き整数（32 ビットのプラットフォームでは 32 ビット値、64 ビットのプラットフォームでは 64 ビット値）	Yes	Yes
System.UIntPtr	"System.UIntPtr"型オブジェクト参照	基になるプラットフォームによってサイズが決まる符号なし整数（32 ビットのプラットフォームでは 32 ビット値、64 ビットのプラットフォームでは 64 ビット値）	No	Yes
System.Object	"System.Object"型オブジェクト参照	.NET Framework のすべての型のルート	Yes	No
System.String	"System.String"型オブジェクト参照	Unicode 文字の文字列	Yes	No

表中にある CLS 準拠についての説明は CLS 準拠データ型を参照してください。 また、表中にある値型についての説明は参照型と値型を参照してください。

用途の BINARY-SHORT SIGNED, BINARY-LONG SIGNED, BINARY-DOUBLE SIGNED において、SIGNED は省略可能です。

たとえば、 NetCOBOL for .NET でメソッドに USAGE BINARY-LONG 型の引数を定義した場合、.NET Framework の表現を用いれば、 メソッドに System.Int32 型の引数が定義されていることになります。

.NET Framework のドキュメントを読んだり、 .NET Framework 上で他言語と連携するアプリケーションを開発したりする場合には、 インタフェース部分での言語間の表現の違いを乗り越えるために、 データ型を.NET Framework の型として理解することが大切になります。 Visual C#や Visual Basic の組み込みデータ型と.NET 基本データ型の対応については、 .NET Framework のドキュメントの「.NET Framework クラス ライブラリの概要」の「名前空間 System」の項を参照してください。 その他の言語については、その言語のドキュメントを参照してください。

上の表に示した NetCOBOL for .NET のデータ型は、 原則として対応する.NET 基本データ型に対応付けられますが、 以下の場合には例外的に COBOL 独自データ型として取り扱われます。 COBOL 独自データ型については「COBOL 独自データ型」を参照してください。

この例外は、.NET 基本データ型に対応付けられた COBOL データ型のうち、 オブジェクトとして扱われないものに対して発生します。 これを非オブジェクト.NET 基本データ型と呼ぶことにします。 一方、オブジェクトとして扱われるデータ型は常に.NET データ型として取り扱われます。 例えば、用途が BINARY-LONG であるデータ（System.Int32）は非オブジェクト.NET 基本データ型ですが、 用途が System.String 型 OBJECT REFERENCE であるデータ（System.String）はそうではありません。

集団項目中に埋め込まれた非オブジェクト.NET 基本データ型

集団項目に属する項目はすべて COBOL 独自データ型として取り扱われます。 例えば、02 レベルに定義された用途が COMP-1 のデータ項目は System.Single 型の.NET 基本デ

ータ型ではなく、 COBOL 独自データ型として取り扱われます。

REDEFINE された非オブジェクト.NET 基本データ型

REDEFINES 句が記述された項目、または REDEFINES 句の対象になった項目は常に COBOL 独自データ型として取り扱われます。 例えば、REDEFINES 句が記述された用途が BINARY-LONG のデータ項目は、それが 01 レベルや 77 レベルであっても COBOL 独自データ型として取り扱われます。

プログラム定義の引数として記述された非オブジェクト.NET 基本データ型

プログラム定義の引数として記述された非オブジェクト.NET 基本データ型は、 COBOL 独自データ型として取り扱われます。

NetCOBOL for .NET V2.0 以前のバージョンでは、 以下の整数型の対応付けを行っていました。

.NET Framework での型名	COBOL 言語上の扱い
System.Int16	PIC S9(4) USAGE COMP-5
System.Int32	PIC S9(9) USAGE COMP-5
System.Int64	PIC S9(18) USAGE COMP-5

この対応は現行のバージョンでも有効ですが、 新しくコードを記述する場合には、上の一覧表で挙げた型（BINARY-SHORT, BINARY-LONG, BINARY-DOUBLE）を使うことをお勧めします。

たとえば、 System.Int32 型データを扱うために PIC S9(9) USAGE COMP-5 型のデータを定義すると、 .NET Framework 上は System.Int32 型データが確保されます。 他言語とのインタフェースという観点からは特に問題ありません。 しかし、COBOL 言語で記述された手続きの内部では、 手続きの記述によってはこのデータに対して 9 桁への丸めが発生する場合があります。 これは COBOL 言語としては正しい動作ですが、 .NET Framework との親和性を考えると、 桁数を意識しないバイナリ型として扱われる BINARY-LONG 型を使用したほうがよいでしょう。

関連トピックス

.NET 基本データ型のデータをオブジェクトとして扱う

非オブジェクト.NET 基本データ型（BINARY-LONG や COMP-1 など）をオブジェクトとして扱う方法について説明しています。

その他の.NET データ型

.NET Framework のデータ型のうち、 .NET 基本データ型以外のデータ型を「その他の.NET データ型」と呼びます。 その他の.NET データ型には、以下があります。

- .NET Framework が提供するクラスライブラリのほとんどの型（.NET 基本データ型以外の型）
- NetCOBOL for .NET で利用者が開発したクラスやインタフェースなど

NetCOBOL for .NET では、 その他の.NET データ型を COBOL のオブジェクト参照で扱います。

逆に、NetCOBOL for .NET でのオブジェクト参照は、すべて.NET データ型（.NET 基本データ型またはその他の.NET データ型）です。

NetCOBOL for .NET で記述したクラス定義、インタフェース定義、Enum 定義、デリゲート定義は、.NET データ型（その他の.NET データ型）になります。 これに対し、プログラム定義は.NET データ型に直接対応付けられません。

NCOBU00449

COBOL 独自データ型

NetCOBOL for .NET で使用するデータ型のうち、 .NET データ型（.NET 基本データ型およびその他の.NET データ型）でないものを COBOL 独自データ型と呼びます。 COBOL 独自データ型は COBOL 言語に特有なデータ型であり、 .NET Framework の型に直接対応付けられません。 また、他の言語から直接利用することもできません。

以下に.NET データ型と COBOL 独自データ型の例を示します。

```
...
      REPOSITORY.
         CLASS CLASS-ARRAY-LIST AS "System.Collections.ArrayList"
...
      WORKING-STORAGE SECTION.
      *> 以下は.NET データ型の例です。
      01 WK-ARRAYLIST OBJECT REFERENCE CLASS-ARRAY-LIST.  *> オブジェクト参照
      01 WK-INT      BINARY-LONG.                         *> .NET 基本データ型
      01 WK-FLOAT    COMP-1.                              *> .NET 基本データ型

      *> 以下は COBOL 独自データ型の例です。
      01 WK-ALPHANUMERIC    PIC X(32).
      01 WK-NATIONAL       PIC N(8).
      01 WK-NUMERIC        PIC 99.
      01 WK-GROUP.
       02 WK-FLOAT-IN-GROUP COMP-1.                       *> 集団項目は常に COBOL 独自データ型
```

.NET データ型の分類

.NET Framework では.NET データ型をいくつかの種類に分類しています。 特に、参照型と値型という大きな分類は重要です。 参照型と値型では、オブジェクトのためのメモリの利用方法に違いがあるため、 プログラミングに影響があります。 ここでは、これら.NET Framework での型の分類について説明します。

このセクションの内容

参照型と値型

参照型と値型について説明します。

ボックス化とボックス化解除

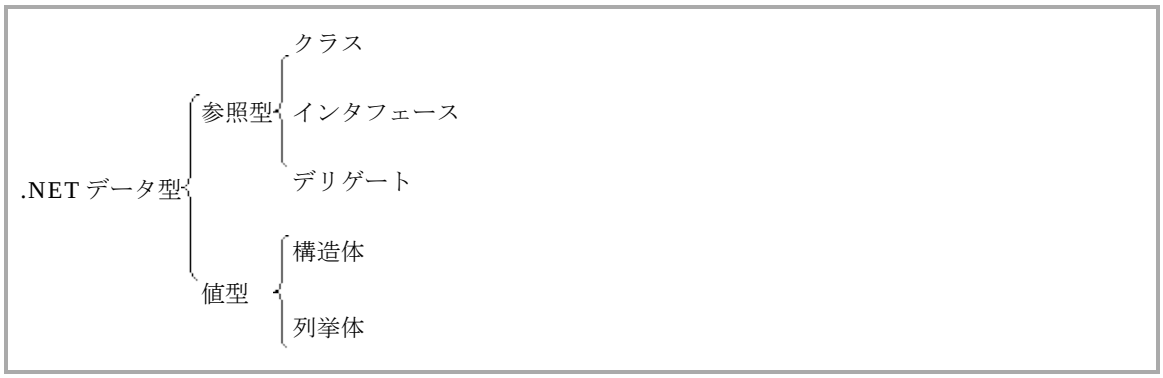
ボックス化とボックス化解除について説明します。

NCOBU00451

参照型と値型

.NET データ型は、大きく参照型と値型に分類されます。

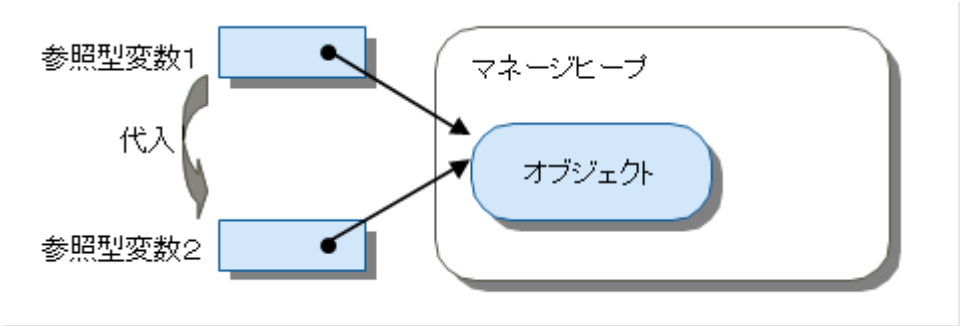
参照型は、さらにクラス、インタフェース、およびデリゲートに分類されます。 また、値型は、構造体および列挙体に分類されます。 これらの各分類の詳細については、.NET Framework のドキュメントの共通型システムを参照してください。



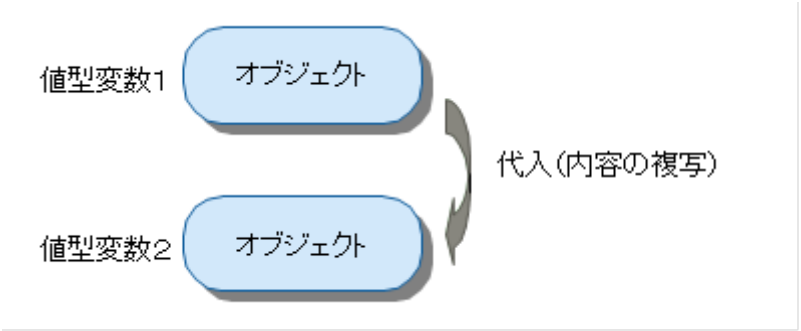
.NET Framework のクラスライブラリリファレンスを見ると、 .NET Framework の各型がどれに該当するかがわかります。 たとえば、 System.Version 型は「Version クラス」と記載されていることから、 クラスであり、つまり参照型であることがわかります。 また、System.DateTime 型は「DateTime 構造体」と記載されていることから、 構造体であり、つまり値型であることがわかります。 また、.NET Framework の配列は参照型になります。

参照型と値型は、オブジェクトのためのメモリの割り当て方法において大きな違いがあります。

参照型のオブジェクトはマネージヒープ上に作成されます。 参照型のオブジェクト変数には、 マネージヒープ上のオブジェクト本体への参照が格納されます。 参照型のオブジェクト変数を別の変数に代入すると、 オブジェクトへの参照が複写されます。 この結果、二つのオブジェクト変数が同一オブジェクトを参照することになります。



これに対し、 値型のオブジェクトはその場に埋め込まれます。 メソッドのローカル変数として値型のオブジェクト変数を定義すると、 ローカル変数領域にオブジェクトが埋め込まれます。 ある型の中に値型のオブジェクトをフィールドとして定義すると、その型のオブジェクト領域の内部に値型オブジェクトが埋め込まれます。 値型のオブジェクト変数を別の変数に代入すると、 オブジェクトの内容が複写されます。 この結果、同一内容をもつ二つのオブジェクトが存在することになります。



以下は、参照型と値型のメモリの使い方の違いを表すサンプルです。

```
...
      REPOSITORY.
          CLASS CLASS-ARRAYLIST AS "System.Collections.ArrayList" *> 参照型の例
          CLASS CLASS-SIZE AS "System.Drawing.Size"               *> 値型の例
          PROPERTY PROP-CAPACITY AS "Capacity"
          PROPERTY PROP-WIDTH AS "Width"
...
      WORKING-STORAGE SECTION.
      01 WK-ARRAYLIST-1 OBJECT REFERENCE CLASS-ARRAYLIST.
      01 WK-ARRAYLIST-2 OBJECT REFERENCE CLASS-ARRAYLIST.
      01 WK-SIZE-1 OBJECT REFERENCE CLASS-SIZE.
      01 WK-SIZE-2 OBJECT REFERENCE CLASS-SIZE.
...
          *> 参照型の代入
          *> 参照型はマネージヒープ上に作成され
          *> 参照型の変数にはオブジェクトへの参照が格納されます。
          INVOKE CLASS-ARRAYLIST "NEW" USING 0 RETURNING WK-ARRAYLIST-1.

          *> 参照型の代入では、オブジェクトへの参照が複写されます。
          *> したがって、WK-ARRAYLIST-1 と WK-ARRAYLIST-2 は同一オブジェクトを参照します。
          SET WK-ARRAYLIST-2 TO WK-ARRAYLIST-1.

          DISPLAY PROP-CAPACITY OF WK-ARRAYLIST-1.   *> 0 が出力される。
          DISPLAY PROP-CAPACITY OF WK-ARRAYLIST-2.   *> 0 が出力される。

          MOVE 16 TO PROP-CAPACITY OF WK-ARRAYLIST-1.
          DISPLAY PROP-CAPACITY OF WK-ARRAYLIST-1.   *> 16 が出力される。
          DISPLAY PROP-CAPACITY OF WK-ARRAYLIST-2.   *> 16 が出力される。


          *> 値型の代入
          *> 値型の変数はオブジェクト領域そのものです。
          INVOKE CLASS-SIZE "NEW" USING 0 0 RETURNING WK-SIZE-1.

          *> 値型の代入では、オブジェクトの内容が複写されます。
          *> したがって、WK-SIZE-1 と WK-SIZE-2 は同じ内容をもつ２つのオブジェクトになります。
          SET WK-SIZE-2 TO WK-SIZE-1.

          DISPLAY PROP-WIDTH OF WK-SIZE-1.     *> 0 が出力される。
          DISPLAY PROP-WIDTH OF WK-SIZE-2.     *> 0 が出力される。

          MOVE 3 TO PROP-WIDTH OF WK-SIZE-1.
          DISPLAY PROP-WIDTH OF WK-SIZE-1.     *> 3 が出力される。
          DISPLAY PROP-WIDTH OF WK-SIZE-2.     *> 0 が出力される。
...
```

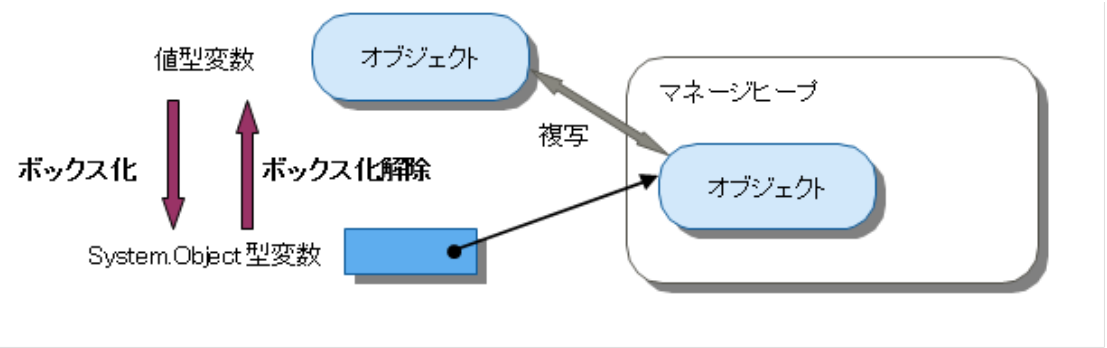
NCOBU00452

ボックス化とボックス化解除

.NET Framework 上のすべての型は System.Object クラスから派生しています。 そのため、.NET Framework 上で「任意のデータ」を取り扱う場合、 System.Object オブジェクトの形でデータを扱います。 たとえば、動的に拡張可能な配列としての機能をもつ System.Collections.ArrayList オブジェクトは、 配列に格納する要素を System.Object オブジェクトの形で保持することによって、任意の.NET Framework データを保持できるようになっています。

ところが、値型と参照型はメモリの使い方が異なるため、 値型のオブジェクトをそのまま参照型である System.Object オブジェクトとして取り扱うことができません。そのため、値型のオブジェクトを System.Object オブジェクトとして取り扱う場合には、 ボックス化という操作を行います。 逆に、ボックス化されて System.Object オブジェクトとして取り扱われている値型オブジェクトを元の値型の形式に戻す操作をボックス化解除と呼びます。

値型オブジェクトに対してボックス化を行うと、 その値型オブジェクトのコピーをマネージヒープ上に作成し、そのコピーへの参照をあたかも参照型であるかのように扱います。 逆に、ボックス化解除の際には、 マネージヒープ上の値型オブジェクトのコピーの内容を値型オブジェクトへ転送します。



ボックス化やボックス化解除はコンパイラによって自動的に行われます。 以下にボックス化やボックス化解除が行われる場面の例を示します。

```
      REPOSITORY.
          CLASS CLASS-OBJECT AS "System.Object"
          CLASS CLASS-ARRAYLIST AS "System.Collections.ArrayList"
          CLASS CLASS-DATETIME AS "System.DateTime" *> 値型の例
          PROPERTY PROP-NOW AS "Now"
...
      WORKING-STORAGE SECTION.
      01 WK-ARRAYLIST OBJECT REFERENCE CLASS-ARRAYLIST.
      01 WK-DATETIME  OBJECT REFERENCE CLASS-DATETIME.
      01 WK-OBJECT    OBJECT REFERENCE CLASS-OBJECT.
...
          *> ArrayList オブジェクトを作成します。
          INVOKE CLASS-ARRAYLIST "NEW" RETURNING WK-ARRAYLIST.

          *> 現在時刻を求めます
          SET WK-DATETIME TO PROP-NOW OF CLASS-DATETIME.

          *> 値型を System.Object 型変数へ代入すると、
          *> 自動的にボックス化が実行されます。
          SET WK-OBJECT TO WK-DATETIME.

          *> System.Object 型のメソッド引数に値型を指定した場合にも、
          *> 自動的にボックス化が実行されます。
          INVOKE WK-ARRAYLIST "Add" USING WK-DATETIME.

          *> 実際にはボックス化れた値型である System.Object オブジェクトを
          *> オブジェクト指定子によって値型に変換すると、
          *> 自動的にボックス化解除が実行されます。
          SET WK-OBJECT TO WK-ARRAYLIST::"get_Item" (0).
          SET WK-DATETIME TO WK-OBJECT AS CLASS-DATETIME.
```

値型を System.Object 型の変数に設定したり、 System.Object 型のメソッド引数に値型を設定したりすると、自動的にボックス化処理が行われます。

逆に、 ボックス化された値型である System.Object オブジェクトをオブジェクト指定子によって値型に変換すると、 ボックス化解除処理が行われます。 オブジェクト指定子によってオブジェクト参照の型を変更する処理の詳細については、オブジェクト参照の型を変更するを参照してください。

NCOBU00453

CLS 準拠データ型

.NET Framework では複数の言語で作成したプログラムを連携して動作させることができます。 しかし、ある言語で開発したプログラムのインタフェース部分にその言語の独自機能が含まれていれば、 そのプログラムを別の言語から利用することは困難になります。 そのため、.NET プラットフォームは.NET プラットフォーム対応言語が共通にサポートすべき機能範囲を規定しています。 これを共通言語仕様（CLS: Common Language Specification）といいます。 CLS の範囲内で記述された.NET Framework プログラムは、 CLS に対応した他の言語から利用することができます。

CLS は CLS で用いるデータ型について規定しています。メソッドやプロパティなどを CLS 準拠データ型だけを用いて記述すると、 そのメソッドやプロパティは CLS 準拠になります。 CLS 準拠のメソッドやプロパティは CLS 対応の他言語から利用することができます。

CLS 準拠のデータ型は以下のとおりです。

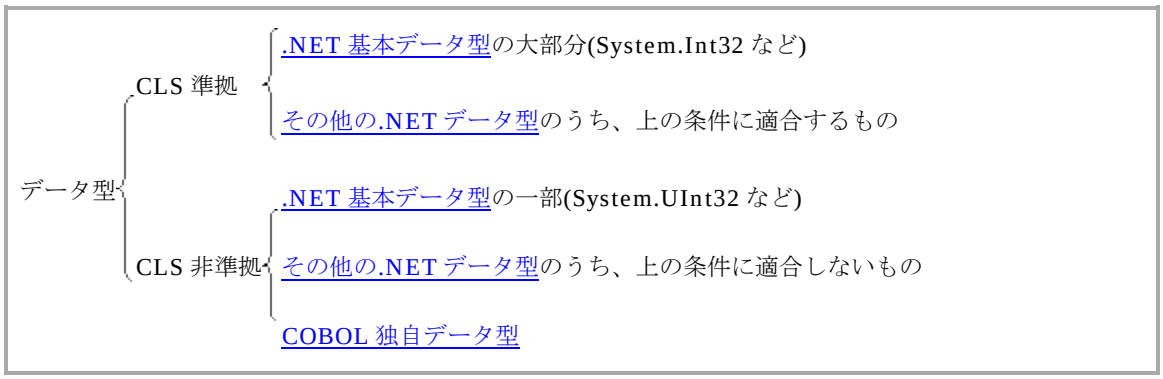
.NET 基本データ型のうち、CLS 準拠であると定義されているもの

.NET 基本データ型で説明している.NET 基本データ型のうち、CLS 準拠であると記述されているもの。

.NET 基本データ型以外の型のうち、公開インタフェース部分に CLS 非準拠のデータ型を使用していないもの

.NET 基本データ型以外の.NET Framework の型で、 公開メソッドや公開プロパティなど、他の型へ公開するインタフェース部分に CLS 非準拠のデータ型を使用していないもの。 非公開インタフェースに CLS 非準拠のデータ型を使用してもかまいません。

NetCOBOL for .NET で扱うデータ型は、CLS 準拠/非準拠の観点から、以下のように分類されます。



CLS についての詳細は、 .NET Framework のドキュメントの「言語間の相互運用性」を参照してください。

NetCOBOL for .NET の COBOL 独自データ型は CLS 非準拠です。

.NET Framework のオブジェクトを使用する

.NET Framework の機能はクラスライブラリで提供されます。 つまり、.NET Framework が提供する豊富な機能は、 .NET Framework のオブジェクトを操作して利用することになります。

ここでは、 .NET Framework のオブジェクトを NetCOBOL for .NET から利用する方法について説明します。

このセクションの内容

.NET Framework のオブジェクトの基本的な使い方

.NET Framework のオブジェクトを NetCOBOL for .NET で利用するための基本的な使い方について説明します。

特別な.NET Framework 型を使う

.NET Framework が提供している型のうち、 注意が必要な型について説明します。

NCOBU00455

.NET Framework のオブジェクトの基本的な使い方

NetCOBOL for .NET では、 COBOL 言語のオブジェクト指向プログラミング機能をそのまま適用することで.NET Framework のオブジェクトを利用することができます。 しかし、.NET Framework のオブジェクトには、 通常の COBOL 言語には存在しない機能（フィールドやイベントなど）があるため、 NetCOBOL for .NET では 言語仕様を拡張して、.NET Framework のオブジェクトの機能を十分に利用できるようにしています。

ここでは、 .NET Framework のオブジェクトを NetCOBOL for .NET で利用するための基本的な使い方について説明します。 なお、特に注意が必要な.NET Framework 型については、「特別な.NET Framework 型を使う」でそれぞれの型について説明しています。

このセクションの内容

オブジェクトを作成する

NetCOBOL for .NET では、 コンストラクターによってオブジェクトを作成します。

ここでは、オブジェクトを生成する方法について説明します。 以下の例では、System.Collections.ArrayList(*1) オブジェクトを作成します。

*1:ArrayList クラスは、 必要に応じてサイズが動的に増加する配列としての機能をもち、IList インタフェースを実装しています。

```
...
      REPOSITORY.
          CLASS CLASS-ARRAY-LIST AS "System.Collections.ArrayList"   *> [1]
          INTERFACE IF-LIST AS "System.Collections.IList"            *> [1]
          .
...
      WORKING-STORAGE SECTION.
      01 WK-OBJ OBJECT REFERENCE CLASS-ARRAY-LIST.                   *> [2]
      01 WK-IF  OBJECT REFERENCE IF-LIST.                           *> [2]
      PROCEDURE DIVISION.
          *> System.Collections.ArrayList オブジェクトを作成します。
          *> 作成したオブジェクトへの参照は、WK-OBJ に格納されます。
          INVOKE CLASS-ARRAY-LIST "NEW" RETURNING WK-OBJ.            *> [3]

          *> System.Collections.ArrayList オブジェクトを作成します。
          *> 初期化パラメタを渡してオブジェクトを作成します。
          *> 作成したオブジェクトへの参照は、WK-IF に格納されます。
          INVOKE CLASS-ARRAY-LIST "NEW" USING 16 RETURNING WK-IF.    *> [4]
...
```

型名を宣言する

リポジトリ段落で型の外部名を内部名に対応付けます。

型の外部名には、 .NET Framework 上での型名を名前空間も含めた形式で記述します（[1]）。 上の例では、System.Collections 名前空間の ArrayList クラスに対して CLASS-ARRAY-LIST という内部名を、System.Collections 名前空間の IList インタフェースに対して IF-LIST という内部名を対応付けています。

既定では型の外部名の大文字小文字は区別されませんが、 ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

型名の宣言は、以下の指定子を用いて記述します。

利用したい型の種類	使用する指定子
クラスまたは構造体	クラス指定子
インタフェース	インタフェース指定子
列挙体	ENUM 指定子
デリゲート	デリゲート指定子

上の例では、 System.Collections.ArrayList 型はクラスなのでクラス指定子を、 System.Collections.IList 型はインタフェースなのでインタフェース指定子を使って記述します。

列挙体やデリゲートの宣言については、 「列挙値を使う」、 「デリゲートオブジェクトを作成する」を参照して

ください。

オブジェクト参照データ項目を定義する

作成したオブジェクトへの参照を保持するデータは、 用途が OBJECT REFERENCE であるデータ項目(オブジェクト参照データ項目)として定義します。 オブジェクト参照データ項目は、OBJECT REFERENCE 句の後ろにオブジェクトの型を表す内部名を記述します（[2]）。レベル番号は 01 または 77 でなければなりません。

オブジェクトを作成する

オブジェクトを作成するには、 コンストラクターを呼び出します。 コンストラクターは、実際にはメソッドではありませんが、 形式上は型の"NEW"静的メソッドのように記述します（[3]）。 NetCOBOL for .NET コンパイラはこの構文を特別扱いして、 コンストラクター呼出しに翻訳します。 RETURNING 指定に記述されたデータ項目には、作成されたオブジェクトへの参照が格納されます。

型によっては、 コンストラクターに初期化パラメタを指定しなければならない場合があります。 コンストラクターの引数については、 それぞれの型のドキュメントを参照してください。 たとえば、System.Collections.ArrayList クラスのドキュメントの「ArrayList コンストラクター」を見ると、System.Collections.ArrayList クラスには 3 種類のコンストラクターが定義されていることがわかります。 その中から、 [3]では引数なしのコンストラクターを呼び出して、 [4]では System.Int32 型の引数を取るコンストラクターを呼び出してオブジェクトを生成しています。

インタフェースは実装をもたないため、 インタフェース型に対してコンストラクターを呼び出すことはできません。 インタフェース型のオブジェクト参照を利用するには、 そのインタフェースを実装するオブジェクトを別に作成する必要があります。 この例では、 System.Collections.IList インタフェースを実装する System.Collections.ArrayList クラスのオブジェクトを作成して、 それをインタフェース型のオブジェクト参照データ項目に格納しています（[4]）。

- COBOL2002 規格では、 オブジェクトを作成する方法について決まった方法を用意せず、 ファクトリオブジェクトの実装にまかせています。 一方、.NET Framework にはファクトリに相当する実体がないため、 NetCOBOL for .NET ではファクトリをサポートしていません。 このため、NetCOBOL for .NET では、 .NET Framework の仕様にあわせて、 コンストラクターによってオブジェクトを作成するようにしています。
- 値型のオブジェクトは変数領域に埋め込まれるため、 値型の場合、正確には、 コンストラクターによってオブジェクトが生成されるのではなく、 コンストラクターによってオブジェクトが初期化されます。

オブジェクトを破棄する

オブジェクトを明に破棄する方法はありません。 .NET Framework のランタイムは GC(ガベージコレクション)を行い、 どこからも参照されなくなったオブジェクトを自動的に破棄します。

オブジェクト参照データ項目に定義済みオブジェクト一意名 NULL を設定すると、 そのデータ項目はどのオブジェクトも参照していない状態になります。 しかし、その操作によって参照していたオブジェクトが破棄されるわけではありません。

オブジェクトによっては、 後始末が必要な資源を保持している場合があります。 この場合、オブジェクトを使用した後に後始末のためのメソッドを呼び出す必要があります。 詳細については、各クラスのドキュメントを参照してください。 .NET Framework の慣習として、 多くの場合、後始末のためのメソッドの名前は"Dispose"です。 たとえば、ファイルの入出力を行う System.IO.FileStream クラスのオブジェクトは、OS のファイルハンドルを保持しています。 そのため、System.IO.FileStream オブジェクトを使用した後には、Dispose メソッドまたは Close メソッドを呼び出してファイルハンドルを解放する必要があります。

値型のオブジェクトはデータ領域に埋め込まれるため、 値型オブジェクトの寿命はデータの寿命と一致します。 メソッドのローカル変数として定義された値型オブジェクトの寿命はメソッド内であり、 オブジェクトのフィールドとして定義された値型オブジェクトの寿命はオブジェクトの寿命と一致します。

また、値型のオブジェクト参照データ項目は実際には参照ではなくオブジェクト本体であるため、 値型オブジェクト参照データ項目に NULL を設定することはできません。

NCOBU00458

メンバーを利用する

オブジェクトの機能はオブジェクトのメンバーを利用して使用します。 .NET Framework の オブジェクトのメンバーには、以下があります。

メンバーの種類	利用方法の説明
メソッド	メソッドを利用する
プロパティ	プロパティを利用する
フィールド	フィールドを利用する
イベント	イベントを利用する
インデクサ	インデクサを利用する

ここでは、 それぞれのメンバーの利用方法について説明します。

また、静的メンバーを利用するでは、操作対象となるオブジェクトを示すオブジェクト参照一意名のかわりに、型名(クラス名、Enum 名、デリゲート名)を記述して利用する方法について説明します。

メソッドを利用する

オブジェクトに何かの作業をさせるには、 オブジェクトのメソッドを呼び出します。

.NET Framework のオブジェクトのメソッドを呼び出すには、COBOL 言語のオブジェクト指向プログラミング機能のメソッド呼出し構文をそのまま用いることができます。

ここでは、メソッドを呼び出す方法を説明します。 以下の例では、乱数を生成する機能をもつ System.Random クラスのオブジェクトを作成し、 その Next メソッドを使って乱数を発生させます。

```
...
      REPOSITORY.
          CLASS CLASS-RANDOM AS "System.Random"
          CLASS CLASS-CONSOLE AS "System.Console"
          .
...
      WORKING-STORAGE SECTION.
      01 WK-RANDOM-GENERATOR OBJECT REFERENCE CLASS-RANDOM.
      01 WK-MAX            BINARY-LONG VALUE 10.
      01 WK-RANDOM          BINARY-LONG.
      PROCEDURE DIVISION.
          *> System.Random オブジェクトを作成します。
          INVOKE CLASS-RANDOM "NEW" RETURNING WK-RANDOM-GENERATOR.

          *> INVOKE 文によって Next メソッドを呼び出します。
          *> この操作によって、WK-MAX より小さい乱数が生成され、
          *> WK-RANDOM に格納されます。
          INVOKE WK-RANDOM-GENERATOR "Next"
            USING WK-MAX RETURNING WK-RANDOM.                    *> [1]

          *> 行内呼出しによって Next メソッドを呼び出します。
          *> この操作によって、WK-MAX より小さい乱数が生成され、
          *> WK-RANDOM に格納されます。
          MOVE WK-RANDOM-GENERATOR::"Next" (WK-MAX) TO WK-RANDOM.    *> [2]

          *> 引数渡し方法を明示して Next メソッドを呼び出します。
          INVOKE WK-RANDOM-GENERATOR "Next"
            USING BY VALUE WK-MAX RETURNING WK-RANDOM.             *> [3]

          *> 最後に生成した乱数を画面に表示します。
          INVOKE CLASS-CONSOLE "WriteLine" USING WK-RANDOM.         *> [4]
...
```

メソッドを呼び出す

INVOKE 文を記述することによってオブジェクトのメソッドを呼び出すことができます（[1]）。INVOKE 文は、予約語 INVOKE、 オブジェクトを識別するオブジェクト参照一意名、 呼び出すメソッドの名前、の順で記述します。 さらに、引数がある場合は USING 指定を、 復帰項目がある場合は RETURNING 指定を記述します。

メソッドが復帰値を持つ場合は、 メソッドの行内呼出しを用いることができます（[2]）。 この形式では、 オブジェクトを識別するオブジェクト参照一意名、特殊文字語"::"、呼び出すメソッドの名前、の順で記述します。 さらに、引数がある場合は括弧の中に引数を記述します。 メソッドの行内呼出しの値はメソッドの復帰値になります。 [2]では、メソッドの行内呼出しで呼び出されたメソッドの復帰値を MOVE 文によって WK-RANDOM に転記しています。

メソッド名は英数字定数または日本語定数で指定します。 既定ではメソッド名の大文字小文字は区別されません。しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

引数渡し方法を指定する

COBOL のメソッド呼出しにおける既定の引数渡し方法は、参照渡しです。 しかし、.NET Framework では、多くのメソッドの引数は値渡しです。 ほとんどの場合、NetCOBOL for .NET コンパイラが引数渡し方法を適切に選択しますが、 まれにオーバロードされたメソッドの解決候補が複数みつかり、翻訳エラーが発生する場合があります。 その場合は、USING 指定に記述する引数に対して BY VALUE または BY REFERENCE を記述し、 引

数渡し方法を明示してください（[3]）。 ただし、メソッドの行内呼出しでは引数渡し方法を明示することはできません。 引数渡し方法を明示する必要がある場合は INVOKE 文を使用してください。

関連トピックス

メソッド呼出し

INVOKE 文および行内呼出しの使い方について説明しています。

COBOL 文法書の「11.3.3.2 メソッドの行内呼出し」

メソッドの行内呼出しの書き方について、説明しています。

COBOL 文法書の「11.8.3.9 INVOKE 文」

INVOKE 文の書き方について、説明しています。

プロパティを利用する

オブジェクトが保持しているデータを参照・設定するには、 オブジェクトのプロパティを利用します。 プロパティは内部的にはメソッドとして実装されています。 このため、プロパティへのアクセスは、プロパティ参照メソッドまたはプロパティ設定メソッド呼出しに翻訳されます。 プロパティの実装では値のチェックなどの細かい作業を行うことができる一方、 ソース上ではオブジェクトのデータを単純に参照・設定しているように見えます。

.NET Framework のオブジェクトのプロパティを参照するには、COBOL のオブジェクト指向プログラミング機能のオブジェクトプロパティ構文をそのまま用います。

ここでは、プロパティを参照・設定する方法を説明します。 以下の例では、 WebForms アプリケーションの GUI 部品である System.Web.UI.WebControls.TextBox オブジェクトの MaxLength プロパティを利用しています。 MaxLength プロパティはテキストボックスに入力できる最大文字数を表します。

```
...
      REPOSITORY.
         CLASS CLASS-TEXT-BOX AS "System.Web.UI.WebControls.TextBox"
         PROPERTY PROP-MAX-LENGTH AS "MaxLength"                  *> [1]
         .
...
      WORKING-STORAGE SECTION.
      LINKAGE SECTION.
      01 PARAM-TEXT-BOX OBJECT REFERENCE CLASS-TEXT-BOX.
      PROCEDURE DIVISION USING PARAM-TEXT-BOX.
         *> System.Collections.ArrayList オブジェクトを作成します。
         IF PROP-MAX-LENGTH OF PARAM-TEXT-BOX < 32 THEN        *> [2]
           MOVE 32 TO PROP-MAX-LENGTH OF PARAM-TEXT-BOX         *> [3]
         END-IF.
...
```

プロパティ名を宣言する

リポジトリ段落でプロパティの外部名を内部名に対応付けます。 プロパティ名を宣言するには、プロパティ指定子を利用します（[1]）。

既定ではプロパティの外部名の大文字小文字は区別されません。 しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

プロパティを利用する

プロパティを利用するには、オブジェクトプロパティを記述します。 オブジェクトプロパティは、 プロパティの内部名、予約語 OF、オブジェクト参照一意名、の順で記述します。 [2]ではプロパティを参照し、[3]ではプロパティを設定しています。

オブジェクトプロパティの値はプロパティの値となります。 [3]では MOVE 文によって 32 をプロパティの値として設定しています。 もし、プロパティの値がオブジェクト参照の場合は、 MOVE 文の代わりに SET 文を用いる必要があります。

関連トピックス

プロパティ

プロパティの使い方について、説明しています。

COBOL 文法書の「11.3.3.5 オブジェクトプロパティ」

オブジェクトプロパティの書き方について、説明しています。

NCOBU00461

フィールドを利用する

フィールドはオブジェクトが保持しているデータです。 COBOL 言語のオブジェクト指向プログラミング機能では、 オブジェクトが保持しているデータを直接外部に公開することはできません。 データを外部に公開する場合はプロパティを通して公開します。 しかし、.NET Framework ではデータを直接外部に公開することができます。 そのため、NetCOBOL for .NET では、 プロパティアクセスと同じ構文を用いてオブジェクトのフィールドを利用できるようにしています。

ここでは、フィールドを参照・設定する方法を説明します。以下の例では、System.Web.UI.ImageClickEventArgs オブジェクトのフィールドを参照しています。System.Web.UI.ImageClickEventArgs オブジェクトは、 Web アプリケーションでページ上の画像がクリックされた際にその情報を格納します。 この例では、クリックされた位置が原点かどうかを調べています。

```
      REPOSITORY.
         CLASS CLASS-IMAGE-CLICK-EVENT-ARGS
           AS "System.Web.UI.ImageClickEventArgs"
         PROPERTY PROP-X AS "X"                                  *> [1]
         PROPERTY PROP-Y AS "Y"                                  *> [1]
         .
...
      LINKAGE SECTION.
      01 PARAM-POINT OBJECT REFERENCE CLASS-IMAGE-CLICK-EVENT-ARGS.
      01 RET-VAL    PIC 1.
      PROCEDURE DIVISION USING PARAM-POINT RETURNING RET-VAL.
         *> クリックされた点が原点かどうかチェックします。
         *> System.Web.UI.ImageClickEventArgs オブジェクトの
         *> X フィールドと Y フィールドにアクセスしています。
         IF PROP-X OF PARAM-POINT = 0 AND
            PROP-Y OF PARAM-POINT = 0 THEN                       *> [2]
           MOVE B"1" TO RET-VAL
         ELSE
           MOVE B"0" TO RET-VAL
         END-IF.
```

フィールド名を宣言する

リポジトリ段落でフィールドの外部名を内部名に対応付けます（[1]）。 NetCOBOL for .NET では、 フィールドへのアクセスはプロパティへのアクセスと同じ構文を使用しているため、プロパティ指定子を使用します。 対応付けた内部名は、COBOL 構文上はプロパティ名になります。

既定ではフィールドの外部名の大文字小文字は区別されません。 しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

フィールドを利用する

フィールドの参照・設定方法は、 プロパティの参照・設定と同様に、 オブジェクトプロパティの構文を用います（[2]）。

関連トピックス

プロパティ

プロパティの使い方について説明しています。

COBOL 文法書の「11.3.3.5 オブジェクトプロパティ」

オブジェクトプロパティの書き方について説明しています。

COBOL 文法書の「11.3.3.7 フィールド参照」

フィールドを参照する機能を持つ、フィールド参照について説明しています。

イベントを利用する

オブジェクトがイベントを提供している場合、 その「イベント」に対して、イベントが発生した時に呼び出されるメソッドを登録することができます。 この仕組みによって、 ボタンがクリックされた時に何かの処理を行う、といった処理を簡単に記述できるようになります。

イベントは、COBOL 言語のオブジェクト指向プログラミング機能には含まれていない概念です。 Visual C#や Visual Basic ではイベントへのアクセスのために特別な構文を用意していますが、 NetCOBOL for .NET ではメソッド呼出し構文を用いてイベントを利用します。

ここでは、イベントの使用方法を説明します。 以下の例では、System.Timers.Timer オブジェクトの Elapsed イベントを利用しています。

```
...
      REPOSITORY.
         CLASS CLASS-TIMER AS "System.Timers.Timer"
         DELEGATE DEL-ELAPSED-EVENT-HANDLER
           AS "System.Timers.ElapsedEventHandler"
         .
...
      WORKING-STORAGE SECTION.
      01 WK-TIMER         OBJECT REFERENCE CLASS-TIMER.
      01 WK-EVENT-HANDLER OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      PROCEDURE DIVISION.
         *> すでに WK-EVENT-HANDLER にデリゲートオブジェクトが
         *> 設定されているとします。                          *> [1]

         *> System.Timers.Timer オブジェクトを作成します。
         INVOKE CLASS-TIMER "NEW" RETURNING WK-TIMER.

         *> System.Timers.Timer オブジェクトの
         *> Elapsed イベントにデリゲートオブジェクト(イベントハンドラ)を
         *> 登録します。
         INVOKE WK-TIMER "add_Elapsed" USING WK-EVENT-HANDLER.     *> [2]

         *> System.Timers.Timer オブジェクトの
         *> Elapsed イベントからデリゲートオブジェクト(イベントハンドラ)の
         *> 登録を解除します。
         INVOKE WK-TIMER "remove_Elapsed" USING WK-EVENT-HANDLER.   *> [3]
...
```

イベントハンドラを用意する

イベントには、イベントが発生した時に呼び出されるメソッドを登録できます。 イベントが発生した時に呼び出されるメソッドを「イベントハンドラ」と呼びます。

イベントハンドラはデリゲートオブジェクトとして登録します。 デリゲートオブジェクトはメソッドを指すオブジェクトで、 .NET Framework では関数ポインタのような役割を持っています。

イベントに登録できるデリゲートオブジェクトは、そのイベントの型のオブジェクトだけです。 上の例では、System.Timers.Timer クラスの Elapsed イベントのドキュメントを見ると、 Elapsed イベントの型は System.Timers.ElapsedEventHandler デリゲート型であることがわかります。 そのため、Elapsed イベントに登録できるデリゲートオブジェクトは System.Timers.ElapsedEventHandler 型のオブジェクトになります。

ここでは、System.Timers.ElapsedEventHandler オブジェクトがすでに作成されていて、WK-EVENT-HANDLER に設定されているものとします ([1])。 メソッドを指すデリゲートオブジェクトの作成する方法については、 「デリゲートを使う」を参照してください。

イベントにイベントハンドラを登録する

イベントにイベントハンドラを登録するには、 そのイベントの登録メソッドを呼び出します ([2])。 登録メソッドは、イベント名の先頭に"add_"という文字列を付加した名前になります。 上の例では、イベント名が"Elapsed"なので、登録メソッドの名前は"add_Elapsed"になります。

登録メソッドは登録するイベントハンドラを引数に取ります。 復帰値はありません。

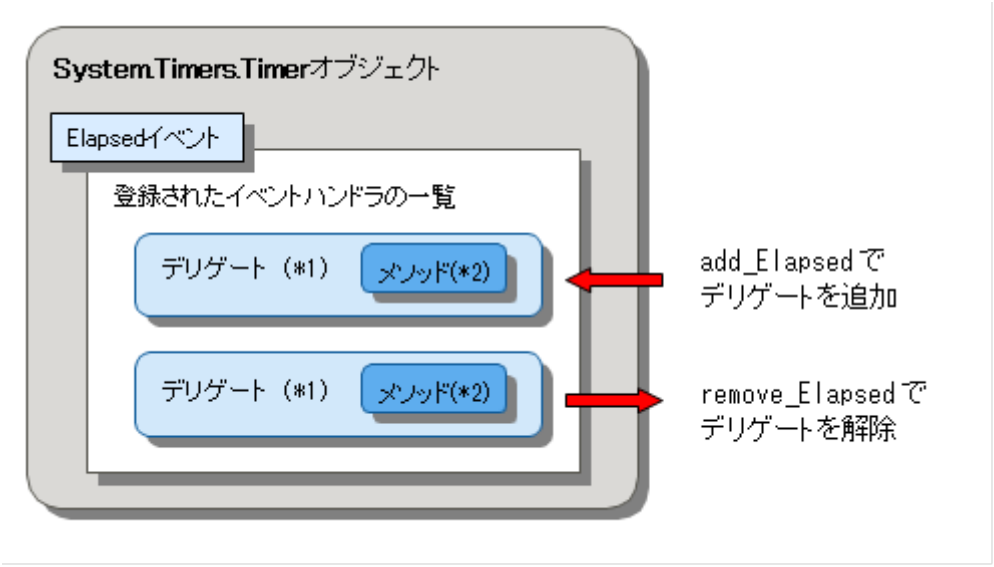
既定では登録メソッド名の大文字小文字は区別されません。 しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

イベントからイベントハンドラの登録を解除する

イベントからイベントハンドラの登録を解除するには、そのイベントの登録解除メソッドを呼び出します([3])。登録解除メソッドは、イベント名の先頭に"remove_"という文字列を付加した名前になります。 上の例では、イベント名が"Elapsed"なので、登録解除メソッドの名前は"remove_Elapsed"になります。

登録解除メソッドは登録を解除するイベントハンドラを引数に取ります。 復帰値はありません。

既定では登録解除メソッド名の大文字小文字は区別されません。 しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。



*1: System.Timers.ElapsedEventHandler 型のデリゲート

*2: デリゲートオブジェクトとメソッドの関連付けについては、デリゲートを使うを参照してください。

参考

イベントに対して登録解除メソッドを呼び出すと、 現在イベントに登録されているイベントハンドラ群から引数で指定されたイベントハンドラを検索し、 その登録を解除します。 この検索で行うイベントハンドラの比較は、 同一オブジェクト（同一インスタンス）かどうかの比較ではなく、 オブジェクトの内容が同じかどうかで比較されます。 そのため、 登録したイベントハンドラを、登録を解除するときまで保持し続ける必要はありません。 登録を解除する時に同じデリゲートオブジェクトを再び作成して、 それを登録解除メソッドに渡せば登録を解除できます。

インデクサを利用する

インデクサは「添字付け」のための特殊なメンバーです。 多くの言語では、 配列のための特別な構文を用意しています。 配列に対して添字付けを行うことによって、 配列内の特定の要素を参照することができます。 同様に、 あるオブジェクトが添字付けによって特定の要素を参照する機能を持つとき、 その機能をインデクサで公開することがあります。 言語によっては、 配列に対する添字付けと同じ構文でインデクサを呼び出します。 そのため、配列と同じ考え方で、添字付けをソース上で簡単に記述できるようになります。

インデクサは、COBOL 言語のオブジェクト指向プログラミング機能には含まれない概念です。 NetCOBOL for .NET ではインデクサはメソッド呼出しで利用します。 COBOL 言語にも配列に似た表という機能がありますが、 COBOL の表は.NET Framework の配列とは仕様が大きく異なります。 このため、NetCOBOL for .NET ではインデクサに対して特別な構文を用意していません。

ここでは、インデクサの使用方法を説明します。 以下の例では、System.Collections.Specialized.NameValueCollection オブジェクトのインデクサを利用しています。 System.Collections.Specialized.NameValueCollection オブジェクトは名前と値という文字列のペアを複数個格納することができます。 このオブジェクトには、名前を添字として対応する値を参照するインデクサが定義されています。

```
...
      REPOSITORY.
         CLASS CLASS-NAME-VALUE-COLLECTION
           AS "System.Collections.Specialized.NameValueCollection"
         .
...
      WORKING-STORAGE SECTION.
      01 WK-COLLECTION OBJECT REFERENCE CLASS-NAME-VALUE-COLLECTION.
      01 WK-VALUE     PIC N(32).
      PROCEDURE DIVISION.
         *> System.Collections.Specialized.NameValueCollection オブジェクトを
         *> 作成します。
         INVOKE CLASS-NAME-VALUE-COLLECTION "NEW" RETURNING WK-COLLECTION.

         *> 名前と値のペアを追加します。
         INVOKE WK-COLLECTION "Add" USING N"東京" N"成田".
         INVOKE WK-COLLECTION "Add" USING N"大阪" N"伊丹".

         *> N"大阪"を添字として値を求めます。
         *> （インデクサを使用します）
         *> WK-VALUE に、N"伊丹"が設定されます。
         INVOKE WK-COLLECTION "get_Item" USING N"大阪"
                                   RETURNING WK-VALUE.    *> [1]

         *> N"大阪"を添字とする値を変更します。
         *> （インデクサを使用します）
         INVOKE WK-COLLECTION "set_Item" USING N"大阪" N"関西".  *> [2]
...
```

インデクサの名前

インデクサの名前を調べるには、その型のドキュメントを調べる必要があります。 インデクサは.NET Framework のリファレンスの中では、 引数つきプロパティとして扱われており、 型のプロパティ一覧の中に記述されています。 多くの場合、インデクサの名前は"Item"ですが、それ以外の名前であることもあります。 たとえば、System.String クラスのインデクサの名前は"Chars"です。 上の例の場合、System.Collections.Specialized.NameValueCollection クラスのドキュメントのプロパティ一覧を見ると、Item プロパティがインデクサであることが分かります。

添字付けによって指定した要素の値を取得する

添字付けによって指定した要素の値を取得するには、インデクサの取得メソッドを呼び出します（[1]）。 取得メソッドは、インデクサの名前の先頭に"get_"という文字列を付加した名前になります。 上の例では、インデクサ名が"Item"なので、取得メソッドの名前は"get_Item"になります。

取得メソッドの引数は添字です。 添字は複数個の引数からなる場合もあります。 復帰値は添字によって指定し

た要素の値です。

既定では取得メソッド名の大文字小文字は区別されません。 しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

添字付けによって指定した要素に値を設定する

添字付けによって指定した要素に値を設定するには、インデクサの設定メソッドを呼び出します（[2]）。 設定メソッドは、インデクサの名前の先頭に"set_"という文字列を付加した名前になります。 上の例では、インデクサ名が"Item"なので、設定メソッドの名前は"set_Item"になります。

設定メソッドの引数は添字と設定する値です。 添字は複数個の引数からなる場合もあります。 末尾の引数が設定する値になります。 復帰値はありません。

既定では設定メソッド名の大文字小文字は区別されません。 しかし、ALPHAL 翻訳オプションの設定によっては大文字小文字の違いを厳密に記述する必要があります。 詳細については、「型名やメンバー名と大文字小文字の区別」を参照してください。

静的メンバーを利用する

静的メンバーを利用するには、 操作対象となるオブジェクトを示すオブジェクト参照一意名のかわりに型名（クラス名、Enum 名、デリゲート）を記述します。これ以外は、 オブジェクトメンバーの利用方法と同じです。

```
...
      REPOSITORY.
          CLASS CLASS-DATETIME AS "System.DateTime"
          CLASS CLASS-CONSOLE AS "System.Console"
          PROPERTY PROP-NOW AS "Now"
          .
...
      WORKING-STORAGE SECTION.
      01 WK-NOW OBJECT REFERENCE CLASS-DATETIME.
      PROCEDURE DIVISION.
          *> 現在の時刻を求めます。
          *> （System.DateTime 型の静的プロパティを参照します）
          SET WK-NOW TO PROP-NOW OF CLASS-DATETIME.

          *> 時刻を画面に出力します。
          *> （System.Console 型の静的メソッドを呼び出します）
          INVOKE CLASS-CONSOLE "WriteLine" USING WK-NOW.
...
```

NCOBU00465

オブジェクトを比較する

2 つのオブジェクト参照データ項目を比較すると、 両者が同一オブジェクト（同一インスタンス）を参照しているかどうかが比較されます。 オブジェクトの内容が同じかどうかは比較されません。 これは、COBOL 言語での比較条件式の仕様です。

```
...
      REPOSITORY.
         CLASS CLASS-VERSION AS "System.Version"
...
      WORKING-STORAGE SECTION.
      01 WK-VERSION-1 OBJECT REFERENCE CLASS-VERSION.
      01 WK-VERSION-2 OBJECT REFERENCE CLASS-VERSION.
      PROCEDURE DIVISION.
         *> System.Version オブジェクトを作成します。
         *>   WK-VERSION-1 は、バージョン値 1.1.0.0 を表します。
         *>   WK-VERSION-2 も、バージョン値 1.1.0.0 を表します。
         INVOKE CLASS-VERSION "NEW" USING 1 1 0 0 RETURNING WK-VERSION-1.
         INVOKE CLASS-VERSION "NEW" USING 1 1 0 0 RETURNING WK-VERSION-2.

         *> 同一オブジェクトかどうかを比較します。
         *> （内容の比較ではありません）
         IF WK-VERSION-1 = WK-VERSION-2 THEN
           *> 条件は偽になるので、以下の行は実行されません。
           DISPLAY N"オブジェクト参照が同じ"
         END-IF.
...
```

値型のオブジェクト参照は比較することはできません。 値型のオブジェクト参照はオブジェクトそのものであり、 参照の比較にならないためです。 ただし、例外的に System.Boolean 構造体と列挙体だけは比較することができます。 System.Boolean 構造体の比較については「System.Boolean オブジェクトを比較する」を参照してください。 列挙体の比較については「列挙値を使う」を参照してください。

一方、.NET Framework の型の中には、 オブジェクトの内容による比較など、 独自の比較方法を提供しているものもあります。 独自の比較方法をサポートしている型は、 等値演算子と非等値演算子を定義しています。 このような型のオブジェクトは、 等値演算子または非等値演算子をメソッドとして呼び出すことでオブジェクト独自の比較を行うことができます。

ここでは､等値演算子と非等値演算子の使い方を説明します。以下の例では､バージョン値を表す System.Version オブジェクトを比較しています。 System.Version 型はオブジェクトの内容（バージョン値）で比較を行う等値演算子と非等値演算子を提供しています。

```
...
      REPOSITORY.
         CLASS CLASS-VERSION AS "System.Version"
...
      WORKING-STORAGE SECTION.
      01 WK-VERSION-1 OBJECT REFERENCE CLASS-VERSION.
      01 WK-VERSION-2 OBJECT REFERENCE CLASS-VERSION.
      01 WK-VERSION-3 OBJECT REFERENCE CLASS-VERSION.
      01 WK-RESULT    PIC 1.
      PROCEDURE DIVISION.
         *> System.Version オブジェクトを作成します。
         *>   WK-VERSION-1 は、バージョン値 1.1.0.0 を表します。
         *>   WK-VERSION-2 は、バージョン値 1.1.0.0 を表します。
         *>   WK-VERSION-3 は、バージョン値 2.0.0.0 を表します。
         INVOKE CLASS-VERSION "NEW" USING 1 1 0 0 RETURNING WK-VERSION-1.
         INVOKE CLASS-VERSION "NEW" USING 1 1 0 0 RETURNING WK-VERSION-2.
```

```
          INVOKE CLASS-VERSION "NEW" USING 2 0 0 0 RETURNING WK-VERSION-3.

          *> WK-RESULTにはB"1"（真）が設定されます。
          INVOKE CLASS-VERSION "op_Equality"
            USING WK-VERSION-1 WK-VERSION-2
            RETURNING WK-RESULT.                              *> [1]

          *> WK-RESULTにはB"0"（偽）が設定されます。
          INVOKE CLASS-VERSION "op_Inequality"
            USING WK-VERSION-1 WK-VERSION-2
            RETURNING WK-RESULT.                              *> [2]


          *> WK-RESULTにはB"0"（偽）が設定されます。
          INVOKE CLASS-VERSION "op_Equality"
            USING WK-VERSION-1 WK-VERSION-3
            RETURNING WK-RESULT.

          *> WK-RESULTにはB"1"（真）が設定されます。
          INVOKE CLASS-VERSION "op_Inequality"
            USING WK-VERSION-1 WK-VERSION-3
            RETURNING WK-RESULT.
...
```

等値演算子

[1]では、等値演算子を利用しています。 等値演算子は型の静的メソッドとして定義されます。 等値演算子のメソッド名は"op_Equality"です。 引数として比較したい 2 つのオブジェクトを指定します。 復帰値は System.Boolean 型の値で、 2 つのオブジェクトが同じであれば真を、異なれば偽を返します。

どのようなオブジェクトを同じとみなすかは型によって異なります。 詳細については、その型のドキュメントを参照してください。

非等値演算子

[2]では、非等値演算子を利用しています。 非等値演算子は型の静的メソッドとして定義されます。 非等値演算子のメソッド名は"op_Inequality"です。 引数として比較したい 2 つのオブジェクトを指定します。 復帰値は System.Boolean 型の値で、 2 つのオブジェクトが同じであれば偽を、異なれば真を返します。

どのようなオブジェクトを同じとみなすかは型によって異なります。 詳細については、その型のドキュメントを参照してください。

Visual C#や Visual Basic などの言語では、 演算子のオーバーライドがサポートされています。 もしある型が等値演算子（または非等値演算子）を持っていれば、 これらの言語でその型のオブジェクトを比較するコードを記述すると、 そのコードはオブジェクトの型の等値演算子（または非等値演算子）呼出しへ翻訳されます。

したがって、 Visual C#や Visual Basic などのサンプルを参考に NetCOBOL for .NET のコードを記述する場合、 サンプル中に記述されたオブジェクトの比較は、 NetCOBOL for .NET ではオブジェクトの型の等値演算子（または非等値演算子）呼出しに置き換えた方が適切な場合があります。

NCOBU00466

オブジェクトの型を確認する

実行時にオブジェクトの型を確認したい場合、型条件を使用します。オブジェクトの型を確認したい場合には、以下のような場合があります。

- ある型を継承しているかどうか
- あるインタフェースを実装しているかどうか

ここでは、オブジェクトの型の確認方法について説明します。

```
...
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-VERSION AS "System.Version"
...
      LINKAGE SECTION.
      01 PARAM-OBJECT OBJECT REFERENCE CLASS-OBJECT.
      01 RET-VAL     PIC 1.
      PROCEDURE DIVISION USING PARAM-OBJECT RETURNING RET-VAL.
         *> PARAM-OBJECT が参照するオブジェクトが、System.Version クラスを
         *> 継承しているかどうかを調べます。
         IF PARAM-OBJECT IS CLASS-VERSION THEN
           MOVE B"1" TO RET-VAL
         ELSE
           MOVE B"0" TO RET-VAL
         END-IF.
...
```

上の例では、型条件を用いて PARAM-OBJECT が参照しているオブジェクトが System.Version クラスを継承しているかどうかを調べています。 型条件は、以下の場合に真になります。

- 左辺が参照しているオブジェクトの型が右辺の型である場合
- 左辺が参照しているオブジェクトの型が右辺の型を継承した型である場合
- 右辺の型がインタフェースで、左辺が参照しているオブジェクトが右辺のインタフェースを実装している場合

関連トピックス

COBOL 文法書の「11.8.2.4.2 型条件」

型条件の書き方について説明しています。

オブジェクト参照の型を変更する

あるオブジェクト参照データから、 別のオブジェクト参照データへオブジェクト参照を代入する場合、 送出し側の型が受取り側の型を継承（あるいは実装）しているときは、 SET 文で代入できます。 それ以外の場合は、オブジェクト指定子が必要になります。

ここでは、オブジェクト参照の型を変更する方法について説明します。

```
...
      REPOSITORY.
          CLASS CLASS-OBJECT AS "System.Object"
          CLASS CLASS-VERSION AS "System.Version"
          PROPERTY PROP-MAJOR AS "Major"
...
      WORKING-STORAGE SECTION.
      01 WK-VERSION   OBJECT REFERENCE CLASS-VERSION.
      LINKAGE SECTION.
      01 PARAM-OBJECT OBJECT REFERENCE CLASS-OBJECT.
      01 RET-VAL      BINARY-LONG.
      PROCEDURE DIVISION USING PARAM-OBJECT RETURNING RET-VAL.
          *> PARAM-OBJECT が参照するオブジェクトが、System.Version クラスを
          *> 継承しているなら、System.Version オブジェクトの
          *> メジャーバージョン値を求めます。
          IF PARAM-OBJECT IS CLASS-VERSION THEN
            *> WK-OBJECT が参照するオブジェクトを WK-VERSION へ設定します。
            SET WK-VERSION TO PARAM-OBJECT AS CLASS-VERSION
            MOVE PROP-MAJOR OF WK-VERSION TO RET-VAL
          ELSE
            MOVE ZERO TO RET-VAL
          END-IF.
...
```

上の例では、 System.Object 型のオブジェクト参照データを System.Version 型のオブジェクト参照データに代入しようとしています。 これがもし逆（System.Version 型のオブジェクト参照データを System.Object 型のオブジェクト参照データに代入）ならば何も問題ありません。 System.Version 型は System.Object 型を継承しているため、 単純な SET 文によってオブジェクト参照を代入できます。 しかし、上の例の場合、PARAM-OBJECT が参照しているオブジェクトは System.Version 型以外のオブジェクトである可能性があるため、 単純に SET 文を記述すると翻訳エラーになります。 このような場合、オブジェクト指定子を記述しなければなりません。

オブジェクト指定子を記述すると、 実行時にオブジェクトの型がチェックされます。 参照されているオブジェクトが AS の後に記述された型名として取り扱うことができれば、 その型に変換したオブジェクト参照を一意名の値とします。 もし、参照されているオブジェクトが AS の後に記述された型名として取り扱うことができなければ、 例外(System.InvalidCastException)が発生します。 上の例では、型条件によって PARAM-OBJECT が System.Version クラス（もしくはそれを継承したクラス）であることがわかっているので、 安全にオブジェクト参照の型を変換することができます。

オブジェクト指定子は、 参照しているオブジェクト自身には何の変更も加えません。 オブジェクトへの参照を格納しているデータ項目の型を変更するためのものです。 受取り側のデータ項目の型にあわせてオブジェクトを変換する機能はありません。 C/C++言語のキャストに慣れている方は注意してください。

関連トピックス

COBOL 文法書の「11.3.3.3 オブジェクト指定子」

オブジェクト指定子の書き方について説明しています。

NCOBU00468

型オブジェクトを取得する

.NET Framework は、 各型に対して、 その型の情報を管理する System.Type オブジェクトを実行時に作成しています。 .NET Framework 上でプログラミングする場合、 型に対する System.Type オブジェクトを取得するには、TYPE OF 特殊レジスタを用います。

ここでは、System.Type オブジェクトを取得する方法について説明します。 以下の例では、System.Collections.ArrayList(*1)オブジェクトに対して、 その内容を配列オブジェクトに変換させる ToArray メソッドを呼び出しています。 ToArray メソッドの引数には、どの型の配列に変換させるかを示す System.Type オブジェクトを指定します。

*1: ArrayList クラス は、必要に応じてサイズが動的に増加する配列を使用します。

```
...
      REPOSITORY.
         CLASS CLASS-ARRAY-LIST AS "System.Collections.ArrayList"
         CLASS CLASS-TYPE AS "System.Type"
         CLASS CLASS-ARRAY AS "System.Array"
         CLASS CLASS-STRING AS "System.String"
         CLASS ARRAY-STRING AS "System.String[]"
...
      WORKING-STORAGE SECTION.
      01 WK-TYPE    OBJECT REFERENCE CLASS-TYPE.
      01 WK-ARRAY   OBJECT REFERENCE CLASS-ARRAY.
      LINKAGE SECTION.
      01 PARAM-LIST OBJECT REFERENCE CLASS-ARRAY-LIST.
      01 RET-VAL    OBJECT REFERENCE ARRAY-STRING.
      PROCEDURE DIVISION USING PARAM-LIST RETURNING RET-VAL.
         *> System.Collections.ArrayList オブジェクトが保持している要素を、
         *> System.String オブジェクトの配列にコピーします。
         *> （ここでは、ArrayList オブジェクトの内容はすべて文字列であると
         *>   仮定しています。）
         *> ToArray メソッドの引数には、どの型の配列を作成したいのかを
         *> 示す System.Type オブジェクトを指定します。
         SET WK-TYPE TO TYPE OF CLASS-STRING.
         INVOKE PARAM-LIST "ToArray" USING WK-TYPE RETURNING WK-ARRAY.

         *> WK-ARRAY は、System.String オブジェクトの配列です。
         *> オブジェクト指定子を用いてオブジェクト参照の型を変換します。
         SET RET-VAL TO WK-ARRAY AS ARRAY-STRING.
...
```

関連トピックス

COBOL 文法書の「11.3.3.6 TYPE OF 特殊レジスタ」

TYPE OF 特殊レジスタの書き方について説明しています。

型名やメンバー名と大文字小文字の区別

NetCOBOL for .NET コンパイラは、 ソース上に記述された以下の外部名に対応する型情報をメタデータから検索します。

- リポジトリ段落に記述された型（クラス、インタフェース、Enum、デリゲート）の外部名
- リポジトリ段落に記述されたプロパティの外部名
- INVOKE 文に記述されたメソッドの外部名
- メソッドの行内呼出しに記述されたメソッドの外部名

このとき、ALPHAL 翻訳オプションの値が外部名の検索方法に影響します。

ALPHAL 翻訳オプションの既定の値は ALPHAL(AUTO)です。 この場合、コンパイラは記述された外部名の大文字小文字を区別せずにメタデータを検索します。

ALPHAL 翻訳オプションの値が、ALPHAL(WORD)または NOALPHAL の場合、 コンパイラは記述された外部名の大文字小文字を区別してメタデータを検索します。 このため、外部名は.NET Framework 上の型名を大文字小文字の違いも含めて正確に記述しなければなりません。

.NET Framework 上でのプログラミングでは、ALPHAL(ALL)は通常用いられません。

ここでは、.NET Framework の型を利用するという観点からのみ ALPHAL 翻訳オプションの影響について説明しています。 ALPHAL 翻訳オプションの本来の機能は、 ソースに記述されたテキストの大文字小文字の扱いを指定するものです。 ALPHAL 翻訳オプションの詳細については、ALPHAL (プログラム中の英小文字の扱い)を参照してください。

NCOBU00470

特別な.NET Framework 型を使う

ここでは、.NET Framework が提供している型のうち、 注意が必要な型の使い方について説明します。

このセクションの内容

.NET Framework の文字列を使う

通常、COBOL 言語で文字列を編集するには、英数字項目や日本語項目を利用します。 一方、.NET Framework で文字列を扱う場合には、System.String クラスのオブジェクトを使用します。 NetCOBOL for .NET から、.NET Framework の機能を利用したり、 .NET プラットフォーム対応の他言語と連携するには、 文字列を System.String クラスのオブジェクトとして利用する必要があります。

.NET Framework での文字列はオブジェクトなので、 文字列を操作するには System.String クラスのメソッドやプロパティを利用します。 System.String クラスの機能については、.NET Framework のドキュメントの「String クラス」を参照してください。

ここでは、 NetCOBOL for .NET での System.String オブジェクトの扱い方について説明します。

このセクションの内容

NCOBU00472

System.String オブジェクトを作成する

一般のオブジェクトと同様に、 コンストラクタを呼び出すことで System.String オブジェクトを作成することができます。 しかし、実際にコンストラクタによって System.String オブジェクトを作成することはあまりありません。 .NET プラットフォームでは、文字列は基本型のひとつです。 このため、.NET プラットフォームに対応する多くの言語では、 各言語の文字列定数を直接 System.String オブジェクトへ変換する機能をもっており、その機能を用いて System.String オブジェクトを作成します。

NetCOBOL for .NET でも、 SET 文によって英数字定数や日本語定数を System.String オブジェクトに変換することができます。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> 日本語定数から
         *> System.String オブジェクトを作成します。
         SET WK-STRING TO N"あいうえお".

         *> 英数字定数から
         *> System.String オブジェクトを作成します。
         SET WK-STRING TO "ABC".
...
```

また、SET 文によって英数字データ項目や日本語データ項目を System.String オブジェクトに変換することもできます。

これらの変換規則の詳細については、「COBOL 独自データ型との間で変換する」を参照してください。

英数字定数を System.String オブジェクトに変換するより、 日本語定数を System.String オブジェクトに変換した方がオーバーヘッドが少なくなります。 詳細については、「COBOL 独自データ型との変換と文字コード」を参照してください。

COBOL 独自データ型との間で変換する

NetCOBOL for .NET には、 COBOL の英数字項目や日本語項目と System.String オブジェクトとを変換する機能が用意されています。

このセクションの内容

NCOBU00474

英数字定数・日本語定数からの変換

SET 文を使って、 英数字定数または日本語定数を System.String オブジェクトに変換することができます。

SET 文の送出し側に英数字定数または日本語定数を記述し、受取り側に System.String クラス型のオブジェクト参照項目を記述します。 この SET 文によって、記述された定数と同じ内容をもつ System.String オブジェクトが作成され、 そのオブジェクトへの参照が受取り側のオブジェクト参照項目に格納されます。 このとき、末尾の空白を含む定数全体の内容が System.String オブジェクトに変換されます。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> WK-STRINGには、"ABCDE"という内容を持つ
         *> System.Stringオブジェクトへの参照が設定されます。
         SET WK-STRING TO N"ABCDE".

         *> WK-STRINGには、"FGHIJ"という内容を持つ
         *> System.Stringオブジェクトへの参照が設定されます。
         SET WK-STRING TO "FGHIJ".

         *> WK-STRINGには、"XYZ   "という内容を持つ
         *> System.Stringオブジェクトへの参照が設定されます。
         *> （末尾の空白も変換されます）
         SET WK-STRING TO N"XYZ   ".
...
```

データ項目を System.String オブジェクトに変換する場合は、末尾の空白は変換されませんが、 定数を System.String オブジェクトに変換する場合は、末尾の空白も含めて変換されます。

英数字項目・日本語項目との変換

SET 文を使って、 英数字項目または日本語項目を System.String オブジェクトに変換することができます。 逆に System.String オブジェクトを英数字項目または日本語項目に変換することもできます。

英数字項目および日本語項目を System.String オブジェクトに変換する

SET 文の送出し側に英数字項目または日本語項目を記述し、受取り側に System.String クラス型のオブジェクト参照項目を記述します。 この SET 文によって、COBOL のデータ項目に格納されている文字列と同じ文字列内容をもつ System.String オブジェクトが作成され、 そのオブジェクトへの参照が受取り側のオブジェクト参照項目に格納されます。 このとき、データ項目の文字列の末尾の空白は System.String オブジェクトへ変換されません。

System.String オブジェクトを英数字項目または日本語項目に変換する

SET 文の送出し側に System.String クラス型のオブジェクト参照項目を記述し、受取り側に英数字項目または日本語項目を記述します。 この SET 文によって、送出し側の System.String オブジェクトの文字列内容が受取り側のデータ項目に複写されます。 このとき、 データ項目の長さが System.String オブジェクトの内容よりも長ければ、領域の末尾には空白が詰められます。 データ項目の長さが System.String オブジェクトの内容よりも短ければ、データ領域に格納できない部分は切り捨てられます。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-ALPHANUMERIC   PIC X(8).
      01 WK-NATIONAL      PIC N(8).
      01 WK-NATIONAL-SHORT PIC N(2).
      01 WK-STRING        OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> WK-STRING には、"ABCDE"という内容を持つ
         *> System.String オブジェクトが設定されます。
         MOVE N"ABCDE" TO WK-NATIONAL.
         SET WK-STRING TO WK-NATIONAL.

         *> WK-STRING には、"FGHIJ"という内容を持つ
         *> System.String オブジェクトが設定されます。
         MOVE "FGHIJ" TO WK-ALPHANUMERIC.
         SET WK-STRING TO WK-ALPHANUMERIC.

         *> WK-NATIONAL の内容は、N"FGHIJ   "となります。
         SET WK-NATIONAL TO WK-STRING.

         *> WK-NATIONAL-SHORT の内容は、N"FG"となります。
         SET WK-NATIONAL-SHORT TO WK-STRING.

         *> WK-ALPHANUMERIC の内容は、"FGHIJ   "となります。
         SET WK-ALPHANUMERIC TO WK-STRING.
...
```

文字定数を System.String オブジェクトに変換する場合は、末尾の空白も含めて変換されますが、 データ項目を System.String オブジェクトに変換する場合は、末尾の空白は変換されません。

NCOBU00476

メソッド呼出し時の自動変換

メソッドやプログラム呼出しでは、 System.String 型の引数に対して COBOL の英数字項目や日本語項目を指定することができます。

INVOKE 文、CALL 文、およびメソッドの行内呼出しでは、 メソッド（またはプログラム）の System.String 型の値渡し引数に、以下を記述することができます。

- 英数字項目
- 日本語項目
- 英数字定数
- 日本語定数

また、System.String 型の参照渡し引数および復帰項目には、以下を記述することができます。

- 英数字項目
- 日本語項目

これらの引数は、NetCOBOL for .NET が System.String オブジェクトに変換してから 引数として引き渡します。さらに、参照渡し引数および復帰項目については、メソッド呼出し後に System.String オブジェクトから英数字項目または日本語項目への変換が自動的に行われます。

このとき行われる変換の規則は、 「英数字定数・日本語定数からの変換」や「英数字項目・日本語項目との変換」で説明されているものと同じです。

```
...
      REPOSITORY.
         CLASS CLASS-ENVIRONMENT AS "System.Environment"
...
      WORKING-STORAGE SECTION.
      01 WK PIC N(260).
      PROCEDURE DIVISION.
         *> WK に TEMP 環境変数の値を取得します。
         *> GetEnvironmentVariable メソッドは System.String 型の引数を取り、
         *> System.String オブジェクトを返します。
         *> NetCOBOL for .NET は引数に指定された日本語定数（N"TEMP"）を
         *> System.String オブジェクトに変換してメソッドを呼び出し、
         *> 呼出し後に復帰値である System.String オブジェクトの内容を
         *> WK へ複写しています。
         SET WK TO CLASS-ENVIRONMENT::"GetEnvironmentVariable" (N"TEMP").
...
```

COBOL 独自データ型との変換と文字コード

英数字データまたは日本語データと System.String オブジェクトとの間で変換を行う場合、 NetCOBOL for .NET は、必要があれば文字コードを自動的に変換します。

.NET プラットフォームでは文字コードとして UCS-2 を採用しています。 一方、NetCOBOL for .NET では、英数字や日本語の文字コードを、翻訳時に翻訳オプション RCS で指定された実行時文字コードで格納しています。NetCOBOL for .NET は、 英数字データまたは日本語データと System.String オブジェクトとの間で変換を行う場合、 COBOL の実行時文字コードと UCS-2 の変換を行います。

RCS 翻訳オプションの値が RCS(SJIS)の場合を除いて、 NetCOBOL for .NET の日本語データの実行時文字コードは、.NET Framework と同じ UCS-2 です。 そのため、日本語データと System.String オブジェクトとの間で変換を行う方が、 英数字データと System.String オブジェクトとの間で変換を行うよりもオーバーヘッドが少なくなります。

NetCOBOL for .NET が自動的に文字コード変換を行うため、 変換対象の英数字データや日本語データの中に実行時文字コードにおいて不正なデータ（バイナリデータや他の文字コードのデータ）が含まれていると、 予期しない結果になることがあります。

NCOBU00478

System.String オブジェクトを比較する

System.String クラス型の 2 つのオブジェクト参照項目を比較すると、 両者が同一オブジェクト（同一インスタンス）かどうかが比較されます。 文字列の内容が同じかどうかは比較されません。 これは、COBOL 言語での比較条件式の仕様です。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING-1 OBJECT REFERENCE CLASS-STRING.
      01 WK-STRING-2 OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         SET WK-STRING-1 TO N"ABC".
         SET WK-STRING-2 TO N"ABC".

         *> 同一オブジェクトかどうかが比較されます。
         *> （文字列の内容の比較はされません）
         IF WK-STRING-1 = WK-STRING-2 THEN
           *> 条件が真となるかどうかは
           *> 状況に依存します。
           DISPLAY N"オブジェクト参照が同じ"
         END-IF.
...
```

System.String オブジェクトの文字列内容を比較するには、 以下の 2 つの方法があります。

- System.String クラスの等値演算子を呼び出す
- System.String クラスの Compare 静的メソッドを呼び出す

System.String クラスの等値演算子を呼び出す

文字列を比較する最も単純な方法は、 System.String クラスの等値演算子（op_Equality メソッド）を呼び出すことです。 等値演算子は、2 つの System.String オブジェクトの文字列内容が同じかどうかを判定します。 非等値演算子（op_Inequality メソッド）は、 2 つの System.String オブジェクトの文字列内容が異なっているかどうかを判定します。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING-1 OBJECT REFERENCE CLASS-STRING.
      01 WK-STRING-2 OBJECT REFERENCE CLASS-STRING.
      01 WK-STRING-3 OBJECT REFERENCE CLASS-STRING.
      01 WK-RESULT   PIC 1.
      PROCEDURE DIVISION.
         SET WK-STRING-1 TO N"ABC".
         SET WK-STRING-2 TO N"ABC".
         SET WK-STRING-3 TO N"DEF".

         *> WK-RESULT には B"1"（真）が設定されます。
         INVOKE CLASS-STRING "op_Equality"
           USING WK-STRING-1 WK-STRING-2
           RETURNING WK-RESULT.

         *> WK-RESULT には B"0"（偽）が設定されます。
         INVOKE CLASS-STRING "op_Inequality"
           USING WK-STRING-1 WK-STRING-2
           RETURNING WK-RESULT.


         *> WK-RESULT には B"0"（偽）が設定されます。
         INVOKE CLASS-STRING "op_Equality"
           USING WK-STRING-1 WK-STRING-3
           RETURNING WK-RESULT.
```

```
         *> WK-RESULT には B"1"（真）が設定されます。
         INVOKE CLASS-STRING "op_Inequality"
           USING WK-STRING-1 WK-STRING-3
           RETURNING WK-RESULT.
...
```

Visual C#や Visual Basic のコード中に記述された文字列に対する（各言語での）等値演算子と非等値演算子は、それぞれ System.String クラスの等値演算子と非等値演算子を呼び出す IL に翻訳されます。 したがって、Visual C#や Visual Basic のサンプルを参考に COBOL コードを記述する場合、 サンプル中の文字列比較は System.String クラスの等値演算子呼出しまたは非等値演算子呼出しに置き換えるとよいでしょう。

System.String クラスの Compare 静的メソッドを呼び出す

System.String クラスの Compare メソッドを利用して、文字列の内容を比較することもできます。 Compare メソッドにはいくつかのオーバロードが存在し、 大文字小文字を無視した比較などのさまざまな方法で比較することができます。 Compare メソッドは、文字列が一致する場合に 0 を返します。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING-1  OBJECT REFERENCE CLASS-STRING.
      01 WK-STRING-2  OBJECT REFERENCE CLASS-STRING.
      01 WK-STRING-3  OBJECT REFERENCE CLASS-STRING.
      01 WK-RESULT    BINARY-LONG.
      PROCEDURE DIVISION.
         SET WK-STRING-1 TO N"ABC".
         SET WK-STRING-2 TO N"ABC".
         SET WK-STRING-3 TO N"abc".

         *> Compare メソッドは、文字列の内容を比較します。
         *> WK-RESULT には 0 が設定されます。
         INVOKE CLASS-STRING "Compare"
           USING WK-STRING-1 WK-STRING-2
           RETURNING WK-RESULT.

         *> 大文字小文字を区別しないで、文字列の内容を比較します。
         *> WK-RESULT には 0 が設定されます。
         INVOKE CLASS-STRING "Compare"
           USING WK-STRING-1 WK-STRING-2 B"1"
           RETURNING WK-RESULT.

         *> 大文字小文字を区別して、文字列の内容を比較します。
         *> WK-RESULT には 0 以外が設定されます。
         INVOKE CLASS-STRING "Compare"
           USING WK-STRING-1 WK-STRING-2 B"0"
           RETURNING WK-RESULT.
...
```

大文字小文字の対応など、比較ついての規則はさまざまな国の言語によって異なります。 System.String クラスの Compare メソッドは、実行時のカルチャを用いて比較を行います。 カルチャとは、各国語向けの設定を実行時に利用するために.NET Framework が標準で提供している枠組みです。 Compare メソッドの各オーバロードが行う比較方法の詳細については、.NET Framework のドキュメントの「String.Compare メソッド」を参照してください。

NCOBU00479

空文字列を使う

空文字列（長さ 0 の文字列）は使用頻度が高いため、 System.String クラスは Empty 静的フィールドに空文字列を保持しています。 このフィールドを参照することによって、空文字列を簡単に利用できます。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
         PROPERTY PROP-EMPTY AS "Empty"
         .
...
      WORKING-STORAGE SECTION.
      01 WK-STRING OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> WK-STRING には空文字列が設定されます。
         SET WK-STRING TO PROP-EMPTY OF CLASS-STRING.
...
```

COBOL 言語では、長さ 0 の英数字定数および日本語定数は認められていません。 そのため、以下のソースは翻訳エラーとなりますので上記の方法で設定してください。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> 以下はエラー
         *> 日本語定数の長さが 0 であってはなりません。
         SET WK-STRING TO N"".
...
```

複雑な文字列を組み立てる

System.String オブジェクトは、一度作成されるとその文字列内容は変更できません。 文字列内容の変更を含む操作は、新しい System.String オブジェクトを作成することで行います。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING-1 OBJECT REFERENCE CLASS-STRING.
      01 WK-STRING-2 OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         SET WK-STRING-1 TO N"abc".

         *> 文字列を大文字化します。
         *> 大文字化された文字列内容をもつ System.String オブジェクトへの参照が
         *> WK-STRING-2 に設定されます。
         *> つまり、WK-STRING-1 の他に新しい System.String オブジェクトが
         *> 作成されことになります。
         SET WK-STRING-2 TO WK-STRING-1::"ToUpper"().
...
```

このため、繰り返しなどを用いて複雑な文字列を組み立てる場合、 慎重に処理を記述しないと、大量の System.String オブジェクトを消費して、 結果としてメモリ消費量や性能に影響することがあります。

ここでは、不必要な System.String オブジェクトを作成しないためによく用いられる方法について説明します。

文字列の連結

文字列を連結するには、System.String クラスの Concat 静的メソッドを使用します。 Concat メソッドは可変個の引数を取ることができ、複数の文字列を一度に連結することができます。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> WK-STRING の文字列内容は"ABCDEF"となります。
         SET WK-STRING TO CLASS-STRING::"Concat"(N"ABC" N"DEF").

         *> WK-STRING の文字列内容は"0123456789"となります。
         SET WK-STRING TO CLASS-STRING::"Concat"(N"01" N"23" N"45" N"67" N"89").
...
```

Visual C#や Visual Basic のサンプルを参考に COBOL コードを記述する場合、 サンプル中の文字列連結演算子は Concat メソッドに置き換えるとよいでしょう。

データの整形

System.String クラスの Format 静的メソッドを使ってデータを整形すると、 文字列の組み立て処理が簡単になる場合があります。 Format メソッドは、引数で指定されたオブジェクトを決められた書式で文字列化し、基本となる文字列の中に挿入します。 Format メソッドの詳細については、 .NET Framework のドキュメントの「String.Format メソッド」を参照してください。

以下の例では、名前と点数の組を整形した文字列を作成しています。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      LINKAGE SECTION.
      01 PARAM-NAME  OBJECT REFERENCE CLASS-STRING.
      01 PARAM-POINT BINARY-LONG.
      01 RET-VAL    OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION USING PARAM-NAME PARAM-POINT RETURNING RET-VAL.
         *> RET-VAL には"名前: 点数"のように整形された文字列が設定されます。
         SET RET-VAL TO CLASS-STRING::"Format"(N"{0}: {1}" PARAM-NAME PARAM-POINT).
...
```

複雑な文字列の編集

複雑な文字列を編集するには、System.Text.StringBuilder オブジェクトを使用します。
System.Text.StringBuilder オブジェクトは内部的にバッファを保持しており、 このバッファ上で文字列を編集した後に、バッファの内容を文字列化します。 そのため、編集の過程で余分な System.String オブジェクトを消費しません。 System.Text.StringBuilder クラスの詳細については、 .NET Framework のドキュメントの「StringBuilder クラス」を参照してください。

以下の例では、指定された回数"../"を繰り返した文字列を作成しています。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
         CLASS CLASS-STRINGBUILDER AS "System.Text.StringBuilder"
         PROPERTY PROP-EMPTY AS "Empty"
...
      WORKING-STORAGE SECTION.
      01 WK-FRAGMENT OBJECT REFERENCE CLASS-STRING.
      01 WK-BUFFER   OBJECT REFERENCE CLASS-STRINGBUILDER.
      LINKAGE SECTION.
      01 PARAM-REPETITION BINARY-LONG.
      01 RET-VAL         OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION USING PARAM-REPETITION RETURNING RET-VAL.
         SET WK-FRAGMENT TO N"../".

         SET WK-BUFFER TO CLASS-STRINGBUILDER::"NEW".
         PERFORM PARAM-REPETITION TIMES
           INVOKE WK-BUFFER "Append" USING WK-FRAGMENT
         END-PERFORM.
         SET RET-VAL TO WK-BUFFER::"ToString"().

         *> 以下のコードでは、
         *> 繰り返しごとに System.String オブジェクトを消費します。
      *>  SET RET-VAL TO PROP-EMPTY OF CLASS-STRING.
      *>  PERFORM PARAM-REPETITION TIMES
      *>    SET RET-VAL TO CLASS-STRING::"Concat"(RET-VAL WK-FRAGMENT)
      *>  END PERFORM.
...
```

.NET Framework のブール型を使う

System.Boolean 構造体は.NET 基本データ型のひとつです。 .NET Framework 上では、真偽値を表すのに用いられます。 NetCOBOL for .NET では、他の多くの.NET 基本データ型とは異なり、 System.Boolean 構造体をオブジェクト参照として扱います。

ここでは、 NetCOBOL for .NET での System.Boolean オブジェクトの扱い方について説明します。

このセクションの内容

System.Boolean オブジェクトを作成する

ブール定数から System.Boolean ブジェクトを作成する方法について説明します。

COBOL 独自データ型との間で変換を行う

COBOL のブール項目と System.Boolean オブジェクトとを変換する方法について説明します。

System.Boolean オブジェクトを比較する

System.Boolean オブジェクトの内容を比較する方法について説明します。

NCOBU00482

System.Boolean オブジェクトを作成する

System.Boolean オブジェクトは.NET 基本データ型のひとつです。 そのため、 多くの.NET プラットフォーム対応言語では、 各言語のブール定数から System.Boolean オブジェクトを作成するようにしています。

NetCOBOL for .NET でも、 ブール定数を用いて System.Boolean オブジェクトを作成することができます。

```
...
      REPOSITORY.
         CLASS CLASS-BOOLEAN AS "System.Boolean"
...
      WORKING-STORAGE SECTION.
      01 WK-BOOLEAN-TRUE  OBJECT REFERENCE CLASS-BOOLEAN.
      01 WK-BOOLEAN-FALSE OBJECT REFERENCE CLASS-BOOLEAN.
      PROCEDURE DIVISION.
         *> ブール定数から
         *> System.Boolean オブジェクトを作成します。
         SET WK-BOOLEAN-TRUE  TO B"1".
         SET WK-BOOLEAN-FALSE TO B"0".
...
```

また、SET 文を使って長さ 1 のブール項目を System.Boolean オブジェクトに変換することもできます。COBOL のブール項目と System.Boolean オブジェクトとの変換方法や、 変換規則の詳細については、「COBOL 独自データ型との間で変換を行う」を参照してください。

COBOL 独自データ型との間で変換を行う

NetCOBOL for .NET には、 長さ 1 のブールデータと System.Boolean オブジェクトとを変換する機能が用意されています。ブールデータは、ブール項目およびブール定数を指します。

ブールデータと System.Boolean オブジェクトの値の対応を以下に示します。

System.Boolean オブジェクトの値	COBOL のブール値
真（true）	B"1"
偽（false）	B"0"

ブール定数を System.Boolean オブジェクトに変換

SET 文を使って、 長さ 1 のブール定数を System.Boolean オブジェクトに変換することができます。

SET 文の送出し側に長さ 1 のブール定数を記述し、受取り側に System.Boolean 構造体型のオブジェクト参照項目を記述します。 これにより、ブール定数の値に対応する System.Boolean オブジェクトが作成され、 そのオブジェクトへの参照が、受取り側のオブジェクト参照項目に格納されます。

```
...
      REPOSITORY.
         CLASS CLASS-BOOLEAN AS "System.Boolean"
...
      WORKING-STORAGE SECTION.
      01 WK-BOOLEAN-TRUE  OBJECT REFERENCE CLASS-BOOLEAN.
      01 WK-BOOLEAN-FALSE OBJECT REFERENCE CLASS-BOOLEAN.
      PROCEDURE DIVISION.
         *> WK-BOOLEAN-TRUEには真（true）が設定されます。
         SET WK-BOOLEAN-TRUE TO B"1".

         *> WK-BOOLEAN-FALSEには偽（false）が設定されます。
         SET WK-BOOLEAN-TRUE TO B"0".
...
```

ブール項目と System.Boolean オブジェクトとを変換

SET 文を使って、 長さ 1 のブール項目を System.Boolean オブジェクトに変換することができます。 また、System.Boolean オブジェクトを長さ 1 のブール項目に変換することもできます。

SET 文の送出し側に長さ 1 のブール項目を記述し、受取り側に System.Boolean 構造体型のオブジェクト参照項目を記述します。 これにより、ブール項目の値に対応する System.Boolean オブジェクトが作成され、 そのオブジェクトへの参照が、受取り側のオブジェクト参照項目に格納されます。

SET 文の送出し側に System.Boolean 構造体型のオブジェクト参照項目を記述し、受取り側に長さ 1 のブール項目を記述します。 これにより、System.Boolean オブジェクトの値に対応するブール値が、ブール項目に格納されます。

```
...
      REPOSITORY.
         CLASS CLASS-BOOLEAN AS "System.Boolean"
...
      WORKING-STORAGE SECTION.
      01 WK-BOOLEAN-DISPLAY   PIC 1.
      01 WK-BOOLEAN-BIT      PIC 1 BIT.
      01 WK-BOOLEAN-OBJECT    OBJECT REFERENCE CLASS-BOOLEAN.
      PROCEDURE DIVISION.
         *> WK-BOOLEAN-OBJECTには、真（true）が設定されます。
         MOVE B"1" TO WK-BOOLEAN-DISPLAY.
         SET WK-BOOLEAN-OBJECT TO WK-BOOLEAN-DISPLAY.

         *> WK-BOOLEAN-OBJECTには、偽（false）が設定されます。
```

```
         MOVE B"0" TO WK-BOOLEAN-BIT.
         SET WK-BOOLEAN-OBJECT TO WK-BOOLEAN-BIT.


         *> WK-BOOLEAN-DISPLAY には、B"0"が設定されます。
         SET WK-BOOLEAN-OBJECT TO B"0".
         SET WK-BOOLEAN-DISPLAY TO WK-BOOLEAN-OBJECT.

         *> WK-BOOLEAN-BIT には、B"1"が設定されます。
         SET WK-BOOLEAN-OBJECT TO B"1".
         SET WK-BOOLEAN-BIT TO WK-BOOLEAN-OBJECT.
...
```

メソッド呼出し時の自動変換

メソッドやプログラム呼出しでは、 System.Boolean 型の引数に、長さ 1 のブールデータを指定することができます。

INVOKE 文、CALL 文およびメソッドの行内呼出しでは、 メソッド（またはプログラム）の System.Boolean 型の値渡し引数に、以下を記述することができます。

- 長さ 1 のブール項目
- 長さ 1 のブール定数

また、System.Boolean 型の参照渡し引数および復帰項目には、以下を記述することができます。

- 長さ 1 のブール項目

これらの引数は NetCOBOL for .NET が、System.Boolean オブジェクトに変換してから 引数として引き渡します。 さらに、参照渡し引数および復帰項目については、 メソッド呼出し後に System.Boolean オブジェクトからブール項目への変換が自動的に行われます。

このとき行われる変換の規則は、 上で説明されているものと同じです。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK BINARY-LONG.
      LINKAGE SECTION.
      01 PARAM-STRING-1 OBJECT REFERENCE CLASS-STRING.
      01 PARAM-STRING-2 OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION USING PARAM-STRING-1 PARAM-STRING-2.
         *> 引数で与えられた文字列オブジェクトを比較します。
         *> ３番目の引数は System.Boolean 型で、
         *> 大文字小文字を区別して比較するかどうかを指定します。
         *> NetCOBOL for .NET は引数に指定されたブール定数を
         *> System.Boolean オブジェクトに変換してメソッドを呼び出します。
         INVOKE CLASS-STRING "Compare"
           USING PARAM-STRING-1 PARAM-STRING-2 B"0"
           RETURNING WK.
...
```

System.Boolean オブジェクトを比較する

NetCOBOL for .NET では、 比較条件式で値型を比較することができません。 ただし、例外として System.Boolean オブジェクトは比較条件式で比較することができます。これは、System.Boolean 構造体が.NET 基本データ型のひとつであるためです。

2 つの System.Boolean オブジェクトを比較すると、 両者ともに真、または両者ともに偽の場合に、比較の結果が真になります。

```
...
      REPOSITORY.
         CLASS CLASS-BOOLEAN AS "System.Boolean"
...
      LINKAGE SECTION.
      01 PARAM-BOOLEAN-1 OBJECT REFERENCE CLASS-BOOLEAN.
      01 PARAM-BOOLEAN-2 OBJECT REFERENCE CLASS-BOOLEAN.
      PROCEDURE DIVISION USING PARAM-BOOLEAN-1 PARAM-BOOLEAN-2.
         *> System.Boolean オブジェクトは値型ですが、
         *> 例外的に比較条件式で値を比較することができます。
         IF PARAM-BOOLEAN-1 = PARAM-BOOLEAN-2 THEN
           DISPLAY N"同じ値"
         END-IF.
...
```

また、System.Boolean オブジェクトと長さ 1 のブール定数やブール項目を直接比較することもできます。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
         CLASS CLASS-FILE AS "System.IO.File"
...
      LINKAGE SECTION.
      01 PARAM-PATH OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION USING PARAM-PATH.
         *> System.Boolean オブジェクトとブール定数を直接比較しています。
         *> （System.IO.File クラスの Exists メソッドは、ファイルが存在するか
         *> 調べて、結果を System.Boolean 型で返します。）
         IF CLASS-FILE::"Exists" (PARAM-PATH) = B"0" THEN
           DISPLAY N"ファイルが存在しません。"
         END-IF.
...
```

NCOBU00485

.NET Framework の配列を使う

.NET Framework の配列は、COBOL の「表」とは異なります。

- COBOL の表は COBOL 独自データ型に埋め込まれますが、.NET Framework の配列は配列自体がオブジェクトです。
- COBOL の表は最大要素数が翻訳時に決まりますが、.NET Framework の配列の要素数は、配列オブジェクトを作成する際に動的に決めることができます。
- COBOL の表の添字は 1 から始まりますが、.NET Framework の配列の添字（インデックス）は 0 から始まります。
- COBOL の表では次元ごとに異なるデータ型を指定できますが、.NET Framework の多次元配列はすべて同じ型のデータの集まりです。

.NET Framework の配列はオブジェクトであるため、 配列を操作するにはメソッドやプロパティを使用します。ここでは、.NET Framework の配列を NetCOBOL for .NET で扱う方法について説明します。

このセクションの内容

配列オブジェクトを作成する

一般のオブジェクトと同様に、 コンストラクタを呼び出すことによって配列オブジェクトを作成します。

以下の例では、配列オブジェクトの作成方法を説明します。

```
...
      REPOSITORY.
          CLASS CLASS-STRING  AS "System.String"
          CLASS ARRAY-STRING  AS "System.String[]"                  *> [1]
          CLASS ARRAY2-STRING AS "System.String[,]"                  *> [2]
          PROPERTY PROP-EMPTY AS "Empty"
...
      WORKING-STORAGE SECTION.
      01 WK-ARRAY       OBJECT REFERENCE ARRAY-STRING.
      01 WK-ARRAY2      OBJECT REFERENCE ARRAY2-STRING.
      01 WK-EMPTY-STRING OBJECT REFERENCE CLASS-STRING.
      01 WK-COUNTER-1    BINARY-LONG.
      01 WK-COUNTER-2    BINARY-LONG.
      PROCEDURE DIVISION.
          *> 8 個の要素をもつ 1 次元文字列配列オブジェクトを作成します。
          INVOKE ARRAY-STRING "NEW" USING 8 RETURNING WK-ARRAY.      *> [3]

          *> 4 x 8 個の要素をもつ 2 次元文字列配列オブジェクトを作成します。
          INVOKE ARRAY2-STRING "NEW" USING 4 8 RETURNING WK-ARRAY2.  *> [3]

          *> 配列を初期化します。                                    *> [4]
          SET WK-EMPTY-STRING TO PROP-EMPTY OF CLASS-STRING.

          MOVE ZERO TO WK-COUNTER-1.
          PERFORM UNTIL 8 <= WK-COUNTER-1
            INVOKE WK-ARRAY "Set" USING WK-COUNTER-1 WK-EMPTY-STRING
            ADD 1 TO WK-COUNTER-1
          END-PERFORM.

          MOVE ZERO TO WK-COUNTER-1.
          PERFORM UNTIL 4 <= WK-COUNTER-1
            MOVE ZERO TO WK-COUNTER-2
            PERFORM UNTIL 8 <= WK-COUNTER-2
              INVOKE WK-ARRAY2 "Set" USING WK-COUNTER-1 WK-COUNTER-2 WK-EMPTY-STRING
              ADD 1 TO WK-COUNTER-2
            END-PERFORM
            ADD 1 TO WK-COUNTER-1
          END-PERFORM.
...
```

型名の宣言

配列オブジェクトを利用するには、 リポジトリ段落で配列の型名を宣言する必要があります。 配列の型名を宣言するには、クラス指定子を記述します。

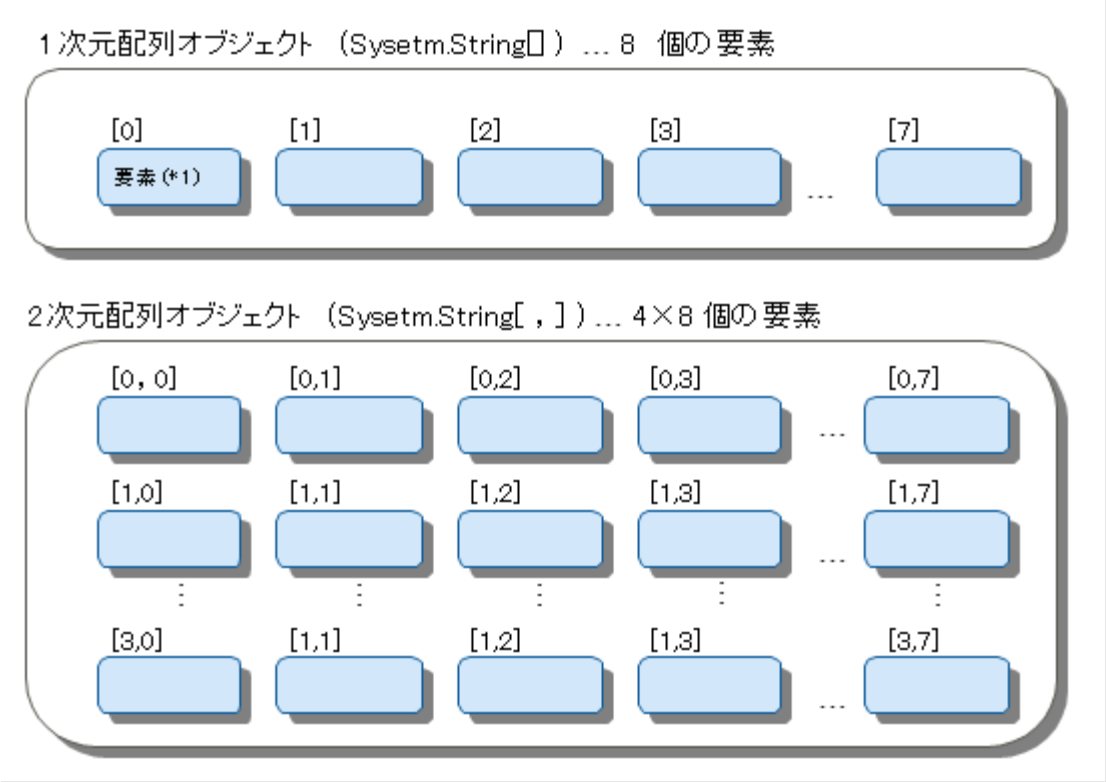
配列オブジェクトの名前は、配列要素の型に角括弧（[]）を追加したものになります。 たとえば、System.String オブジェクトの配列を作成する場合は、 配列の型名は"System.String[]"となります ([1])。 2 次元以上の配列を作成する場合は、 次元数に応じて角括弧の中にコンマを追加します。 たとえば、System.String オブジェクトの 2 次元配列の型名は"System.String[,]"、 3 次元配列の型名は"System.String[,,]"となります ([2])。

COBOL 言語上でオブジェクトとして扱われない.NET 基本データ型（BINARY-LONG 型など）の配列を作成する場合は、 .NET Framework での型名に対して角括弧を追加します。 たとえば、BINARY-LONG 型の 1 次元配列を表す型名は"System.Int32[]"となります。 .NET 基本データ型と.NET Framework 上での型名の関係については、 「.NET 基本データ型」を参照してください。

.NET Framework では配列型に配列要素の個数が含まれません。 つまり、要素の型と次元が同じで、配列要素の個数だけが異なる配列オブジェクトは同じ配列型のオブジェクトとみなされます。

オブジェクトの作成

配列オブジェクトを作成するには、 コンストラクタを呼び出します。 配列のコンストラクタは次元数と同じ数の BINARY-LONG 型の引数を取ります。 各引数はそれぞれの次元で確保する要素数を表します （[3]）。



*1: 要素の型は、リポジトリ段落で定義した型になります。上の例の場合、それぞれの要素は、System.String 型のオブジェクトです。

配列要素の初期化

NetCOBOL for .NET は配列要素を一度に初期化する特別な構文を用意していません。 配列要素を特定の値で初期化するには、 配列要素に値を設定する処理を記述します ([4])。 配列要素にアクセスする方法については、「配列要素にアクセスする」を参照してください。

配列要素にアクセスする

配列要素を取得する

配列オブジェクトの要素を取得するには Get メソッドを用います。

Get メソッドは次元数と同じ数の引数を取ります。 引数は、すべて BINARY-LONG 型です。 引数には、取得する要素を指す各次元の添字(インデックス)を指定します。 インデックスは 0 から始まります。 インデックスが配列の要素数の範囲を超えている場合は、例外(System.ArgumentOutOfRangeException)が発生します。 Get メソッドの復帰項目は、配列要素の値です。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
         CLASS ARRAY-STRING AS "System.String[]"
         CLASS ARRAY2-STRING AS "System.String[,]"
...
      WORKING-STORAGE SECTION.
      01 WK-STRING   OBJECT REFERENCE CLASS-STRING.
      LINKAGE SECTION.
      01 PARAM-ARRAY  OBJECT REFERENCE ARRAY-STRING.
      01 PARAM-ARRAY2 OBJECT REFERENCE ARRAY2-STRING.
      PROCEDURE DIVISION USING PARAM-ARRAY PARAM-ARRAY2.
         *> １次元配列の最初の要素を取得します。
         INVOKE PARAM-ARRAY "Get" USING 0 RETURNING WK-STRING.

         *> ２次元配列の位置(0, 3)の要素を取得します。
         INVOKE PARAM-ARRAY2 "Get" USING 0 3 RETURNING WK-STRING.
...
```

配列要素を設定する

配列オブジェクトの要素を設定するには Set メソッドを用います。

Set メソッドは(次元数 + 1)個の引数を取ります。 最初の次元数個の引数は BINARY-LONG 型です。 これらの引数には、設定する要素を指す各次元のインデックスを指定します。 インデックスは 0 から始まります。 インデックスが配列の要素数の範囲を超えている場合は、例外(System.ArgumentOutOfRangeException)が発生します。 末尾の引数は配列要素の型であり、設定したい値を指定します。

```
...
      REPOSITORY.
         CLASS ARRAY-STRING AS "System.String[]"
         CLASS ARRAY2-STRING AS "System.String[,]"
...
      LINKAGE SECTION.
      01 PARAM-ARRAY  OBJECT REFERENCE ARRAY-STRING.
      01 PARAM-ARRAY2 OBJECT REFERENCE ARRAY2-STRING.
      PROCEDURE DIVISION USING PARAM-ARRAY PARAM-ARRAY2.
         *> １次元配列の先頭から２番目の要素を設定します。
         *> （インデックスは０から始まります）
         INVOKE PARAM-ARRAY "Set" USING 1 N"ABC".

         *> ２次元配列の位置(1, 2)の要素を設定します。
         INVOKE PARAM-ARRAY2 "Set" USING 1 2 N"DEF".
...
```

NCOBU00488

配列オブジェクトのその他のメンバー

配列型は System.Array クラスの派生クラスになります。 そのため、配列オブジェクトに対して System.Array クラスのメソッドやプロパティを呼び出すことができます。 System.Array クラスの詳細については、.NET Framework のドキュメントの「Array クラス」を参照してください。

System.Array クラスのメソッドやプロパティの中で最もよく利用されるものに Length プロパティがあります。 Length プロパティは配列オブジェクトの要素数を表します。 以下の例では、Length プロパティを利用して配列の中に値が NULL の要素がないかどうかを調べています。 .NET Framework では、配列の要素数は実行時にプロパティから取得するものであることに注意してください。

```
...
      REPOSITORY.
          CLASS CLASS-STRING AS "System.String"
          CLASS ARRAY-STRING AS "System.String[]"
          PROPERTY PROP-LENGTH AS "Length"
...
      WORKING-STORAGE SECTION.
      01 WK-LENGTH   BINARY-LONG.
      01 WK-COUNTER  BINARY-LONG.
      01 WK-STRING   OBJECT REFERENCE CLASS-STRING.
      LINKAGE SECTION.
      01 PARAM-ARRAY OBJECT REFERENCE ARRAY-STRING.
      01 RET-VAL     PIC 1.
      PROCEDURE DIVISION USING PARAM-ARRAY RETURNING RET-VAL.
          *> 復帰値を初期化します。
          MOVE B"0" TO RET-VAL.

          *> 配列の長さを求めます。
          MOVE PROP-LENGTH OF PARAM-ARRAY TO WK-LENGTH.

          *> 配列中に値が NULL の要素がないか調べます。
          MOVE ZERO TO WK-COUNTER.
          PERFORM UNTIL WK-LENGTH <= WK-COUNTER
            SET WK-STRING TO PARAM-ARRAY::"Get" (WK-COUNTER)
            IF WK-STRING = NULL THEN
              *> 値が NULL である要素がありました。
              MOVE B"1" TO RET-VAL
              EXIT PERFORM
            END-IF
          END-PERFORM.
...
```

2 次元以上の配列の場合、Length プロパティの値は要素の総数になります。 各次元のそれぞれの要素数を求めるには、GetLength メソッドを用います。 詳細については、.NET Framework のドキュメントの「Array.GetLength メソッド」を参照してください。

列挙体を使う

ここでは、NetCOBOL for .NET で列挙体を利用する方法について説明します。

このセクションの内容

列挙値を使う

列挙値の一般的な使い方について説明します。

列挙値をビットフラグとして扱う

列挙値に対してビット演算を行なう方法について説明します。

列挙値を整数へ変換する

列挙値を整数に変換する方法について説明します。

関連トピックス

列挙型定義

NetCOBOL for .NET で列挙型を定義する方法について説明しています。

NCOBU00490

列挙値を使う

.NET プラットフォームでは、 列挙体の列挙値は単なる整数ではなく、 型付けられた値として扱われます。 列挙値と整数は異なる型のデータとして扱われ、 異なる列挙体の列挙値も異なる型のデータとして扱われます。 列挙値が型付けられたデータであることから、 NetCOBOL for .NET では列挙値をオブジェクトとして扱います。

ここでは、列挙値の使い方を説明します。 以下の例は、引数で与えられた曜日が日曜日かどうかを判定するプログラムです。 このプログラムでは、曜日を表す System.DayOfWeek 列挙体を用いています。

```
...
      REPOSITORY.
          ENUM ENUM-DAY-OF-WEEK AS "System.DayOfWeek"                    *> [1]
          PROPERTY PROP-SUNDAY AS "Sunday"
...
      LINKAGE SECTION.
      01 PARAM-DAY-OF-WEEK OBJECT REFERENCE ENUM-DAY-OF-WEEK.             *> [2]
      01 RET-VAL PIC 1.
      PROCEDURE DIVISION USING PARAM-DAY-OF-WEEK RETURNING RET-VAL.
          *> 引数で与えられた曜日が日曜日かどうかを判定します。
          IF PARAM-DAY-OF-WEEK = PROP-SUNDAY OF ENUM-DAY-OF-WEEK THEN  *> [3]
            MOVE B"1" TO RET-VAL
          ELSE
            MOVE B"0" TO RET-VAL
          END-IF.
...
```

列挙体の宣言

リポジトリ段落で列挙体の外部名を内部名に対応付けます。 Enum 名を定義するには、ENUM 指定子を使用します（[1]）。

列挙値変数の定義

列挙値を格納する変数を定義するには、 変数を列挙体型のオブジェクト参照として定義します（[2]）。

列挙値の参照

列挙値は列挙体の静的定数フィールドです。 したがって、 NetCOBOL for .NET ではフィールド参照の形式で列挙値を参照します（[3]）。フィールド参照については、COBOL 文法書の「11.3.3.7 フィールド参照」を参照してください。

列挙値の比較

列挙値は比較条件式の=, NOT =, EQUAL, NOT EQUAL で比較することができます（[3]）。 ただし、比較条件式の両辺は同じ列挙体のオブジェクトでなければなりません。

列挙値をビットフラグとして扱う

列挙体の中には、 FlagsAttribute カスタム属性が設定されているものがあります。 たとえば、System.IO.FileAttributes 列挙体は FlagsAttribute カスタム属性付きの列挙体です。このような列挙体の列挙値はビットフラグとして扱います。 このため、これらの列挙値に対して AND, OR, NOT などのビット演算を行う必要がしばしば起こります。 ここでは、このような列挙値に対してビット演算を行う方法について説明します。

列挙値に対する AND 演算

列挙値をビットフラグとみなして AND 演算を行うには、 組込み関数の ENUM-AND 関数を使用します。

以下の例では、ENUM-AND 関数を利用して、 指定されたファイルに ReadOnly 属性が設定されているかどうかを調べています。

```
...
      REPOSITORY.
          CLASS CLASS-STRING AS "System.String"
          CLASS CLASS-FILE AS "System.IO.File"
          ENUM ENUM-FILE-ATTRIBUTES AS "System.IO.FileAttributes"
          PROPERTY PROP-READONLY AS "ReadOnly"
...
      WORKING-STORAGE SECTION.
      01 WK-ATTRIBUTES       OBJECT REFERENCE ENUM-FILE-ATTRIBUTES.
      01 WK-MASKED-ATTRIBUTES OBJECT REFERENCE ENUM-FILE-ATTRIBUTES.
      LINKAGE SECTION.
      01 PARAM-PATH OBJECT REFERENCE CLASS-STRING.
      01 RET-VAL    PIC 1.
      PROCEDURE DIVISION USING PARAM-PATH RETURNING RET-VAL.
          *> ファイルの属性を取得します。
          INVOKE CLASS-FILE "GetAttributes" USING PARAM-PATH
                                      RETURNING WK-ATTRIBUTES.

          *> ファイルの属性に ReadOnly フラグが設定されているかどうかを調べます。
          SET WK-MASKED-ATTRIBUTES TO
              FUNCTION ENUM-AND (WK-ATTRIBUTES,
                                 PROP-READONLY OF ENUM-FILE-ATTRIBUTES).
          IF WK-MASKED-ATTRIBUTES = PROP-READONLY OF ENUM-FILE-ATTRIBUTES THEN
            *> ReadOnly フラグが設定されています。
            MOVE B"1" TO RET-VAL
          ELSE
            *> ReadOnly フラグが設定されていません。
            MOVE B"0" TO RET-VAL
          END-IF.
...
```

列挙値に対する OR 演算

列挙値をビットフラグとみなして OR 演算を行うには、 組込み関数の ENUM-OR 関数を使用します。

以下の例では、ENUM-OR 関数を利用して、 指定されたファイルに ReadOnly 属性を追加しています。

```
...
      REPOSITORY.
          CLASS CLASS-STRING AS "System.String"
          CLASS CLASS-FILE AS "System.IO.File"
          ENUM ENUM-FILE-ATTRIBUTES AS "System.IO.FileAttributes"
          PROPERTY PROP-READONLY AS "ReadOnly"
...
      WORKING-STORAGE SECTION.
      01 WK-OLD-ATTRIBUTES OBJECT REFERENCE ENUM-FILE-ATTRIBUTES.
      01 WK-NEW-ATTRIBUTES OBJECT REFERENCE ENUM-FILE-ATTRIBUTES.
      LINKAGE SECTION.
      01 PARAM-PATH OBJECT REFERENCE CLASS-STRING.
      01 RET-VAL    PIC 1.
      PROCEDURE DIVISION USING PARAM-PATH RETURNING RET-VAL.
          *> ファイルの属性を取得します。
          INVOKE CLASS-FILE "GetAttributes" USING PARAM-PATH
                                      RETURNING WK-OLD-ATTRIBUTES.
```

```
         *> ファイルの属性に ReadOnly フラグを追加します。
         SET WK-NEW-ATTRIBUTES TO
             FUNCTION ENUM-OR (WK-OLD-ATTRIBUTES,
                               PROP-READONLY OF ENUM-FILE-ATTRIBUTES).

         *> 必要ならば、ファイルの属性を更新します。
         IF NOT WK-NEW-ATTRIBUTES = WK-OLD-ATTRIBUTES THEN
           INVOKE CLASS-FILE "SetAttributes" USING PARAM-PATH
                                                   WK-NEW-ATTRIBUTES
         END-IF.
...
```

列挙値に対する NOT 演算

列挙値をビットフラグとみなして NOT 演算を行うには、 組込み関数の ENUM-NOT 関数を使用します。

以下の例では、ENUM-NOT 関数と ENUM-AND 関数を利用して、 指定されたファイルから ReadOnly 属性を取り除いています。

```
...
      REPOSITORY.
         CLASS CLASS-STRING AS "System.String"
         CLASS CLASS-FILE AS "System.IO.File"
         ENUM ENUM-FILE-ATTRIBUTES AS "System.IO.FileAttributes"
         PROPERTY PROP-READONLY AS "ReadOnly"
...
      WORKING-STORAGE SECTION.
      01 WK-OLD-ATTRIBUTES OBJECT REFERENCE ENUM-FILE-ATTRIBUTES.
      01 WK-NEW-ATTRIBUTES OBJECT REFERENCE ENUM-FILE-ATTRIBUTES.
      LINKAGE SECTION.
      01 PARAM-PATH OBJECT REFERENCE CLASS-STRING.
      01 RET-VAL    PIC 1.
      PROCEDURE DIVISION USING PARAM-PATH RETURNING RET-VAL.
         *> ファイルの属性を取得します。
         INVOKE CLASS-FILE "GetAttributes" USING PARAM-PATH
                                     RETURNING WK-OLD-ATTRIBUTES.

         *> ファイルの属性から ReadOnly フラグを取り除きます。
         SET WK-NEW-ATTRIBUTES TO
             FUNCTION ENUM-AND (WK-OLD-ATTRIBUTES,
                                FUNCTION ENUM-NOT (PROP-READONLY OF ENUM-FILE-ATTRIBUTES)).

         *> 必要ならば、ファイルの属性を更新します。
         IF NOT WK-NEW-ATTRIBUTES = WK-OLD-ATTRIBUTES THEN
           INVOKE CLASS-FILE "SetAttributes" USING PARAM-PATH
                                                   WK-NEW-ATTRIBUTES
         END-IF.
...
```

列挙値を整数へ変換する

.NET プラットフォームでは、 列挙値を整数とは異なる型付けられたオブジェクトとして扱います。 したがって、列挙値を整数のように扱うべきではありません。 しかし、まれに列挙値を整数としてを参照したい場合もあります。 このような場合、 SET 文を使って列挙値を整数に変換することができます。

逆に、 整数を列挙値へ変換する場合は、System.Enum クラスの ToObject メソッドを使用します。 詳細については、ToObject メソッドのドキュメントを参照してください。

```
...
      REPOSITORY.
         ENUM ENUM-DAY-OF-WEEK AS "System.DayOfWeek"
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-ENUM AS "System.Enum"
         CLASS CLASS-TYPE AS "System.Type"
         PROPERTY PROP-MONDAY AS "Monday"
...
      WORKING-STORAGE SECTION.
      01 WK-DAY-OF-WEEK OBJECT REFERENCE ENUM-DAY-OF-WEEK.
      01 WK-INT        BINARY-LONG.
      01 WK-TYPE       OBJECT REFERENCE CLASS-TYPE.
      01 WK-OBJECT     OBJECT REFERENCE CLASS-OBJECT.
      PROCEDURE DIVISION.
         *> System.DayOfWeek 列挙体の Monday 列挙子の整数値を求めます。
         SET WK-INT TO PROP-MONDAY OF ENUM-DAY-OF-WEEK.

         *> 整数 5 を System.DayOfWeek 列挙体の列挙値に変換します。
         *> 結果として、WK-DAY-OF-WEEK は System.DayOfWeek 列挙体の Friday
         *> 列挙値になります。
         MOVE 5 TO WK-INT.
         SET WK-TYPE TO TYPE OF ENUM-DAY-OF-WEEK.
         INVOKE CLASS-ENUM "ToObject" USING WK-TYPE WK-INT RETURNING WK-OBJECT.
         SET WK-DAY-OF-WEEK TO WK-OBJECT AS ENUM-DAY-OF-WEEK.
...
```

NCOBU00493

デリゲートを使う

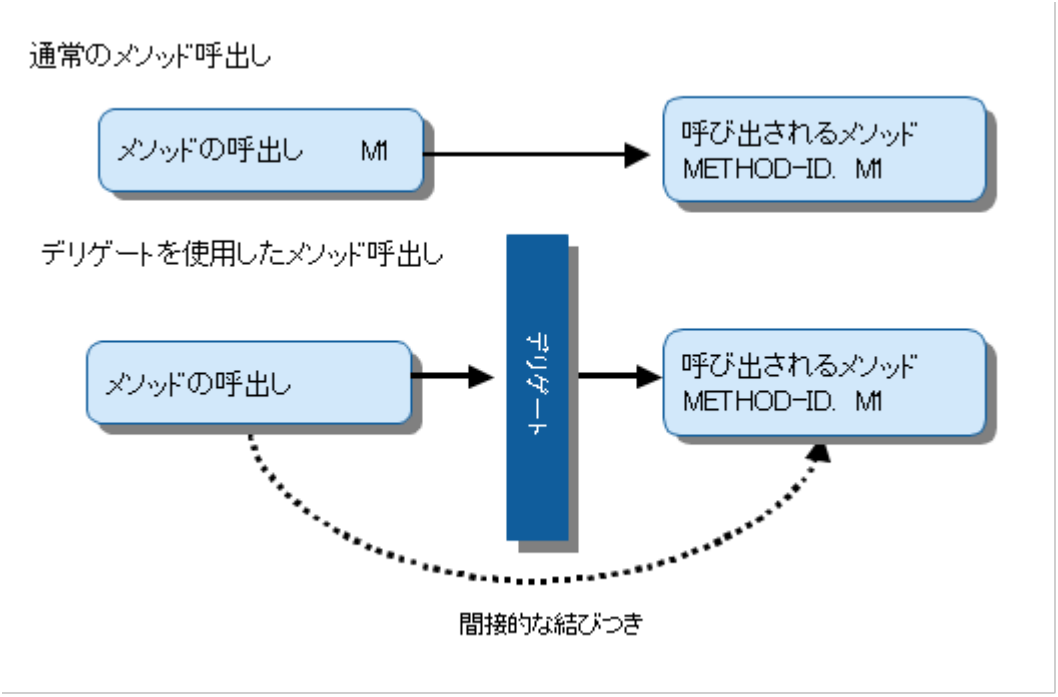
ここでは、NetCOBOL for .NET でデリゲートを利用する方法について説明します。

このセクションの内容

デリゲートとは

デリゲートは、 メソッドへのポインタと同様の機能をもつ.NET Framework のオブジェクトです。

デリゲートは、メソッドへの参照を内部的に持っており、 デリゲートの"Invoke"メソッドが呼び出されると、 保持しているメソッドを呼び出します。 このため、 デリゲートオブジェクトを指定することによって、 実際に呼び出すメソッドを実行時に指定することが可能になります。



.NET Framework は厳密な型付けを行うプラットフォームであるため、 デリゲートのメソッド型（シグネチャ）と参照しているメソッドの型（シグネチャ）は厳密にチェックされます。

関連トピックス

.NET Framework のドキュメントのデリゲート

デリゲートの一般的な情報について説明しています。

NCOBU00495

デリゲートオブジェクトを作成する

デリゲートはクラスの一種です。 したがって、コンストラクタを呼び出すことによってデリゲートオブジェクトを作成します。 ただし、デリゲートのコンストラクタには、 引数の指定に決まりがあるので注意が必要です。

ここでは、デリゲートオブジェクトの作成方法について説明します。 以下の例では、System.Timers.ElapsedEventHandler デリゲートオブジェクトを作成します。

次に示す条件を満たす SAMPLE-CLASS クラスがすでに存在していることを前提に説明を進めます。

- System.Timers.ElapsedEventHandler デリゲートと同じインタフェースをもつ、2 つのメソッド (SAMPLE-STATIC-METHOD と SAMPLE-OBJECT-METHOD)が定義されている。
- SAMPLE-STATIC-METHOD は静的メソッド。
- SAMPLE-OBJECT-METHOD はオブジェクトメソッド。

以下は SAMPLE-CLASS クラスのソースです。

```
      CLASS-ID. SAMPLE-CLASS.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-ELAPSED-EVENT-ARGS AS "System.Timers.ElapsedEventArgs"
         .
      STATIC.
      PROCEDURE DIVISION.

      *> System.Timers.ElapsedEventHandler デリゲートと同じインタフェースを持つ
      *> 静的メソッドを定義します。
      METHOD-ID. SAMPLE-STATIC-METHOD.
      DATA DIVISION.
      LINKAGE SECTION.
      01 PARAM-SENDER   OBJECT REFERENCE CLASS-OBJECT.
      01 PARAM-EVENTARG OBJECT REFERENCE CLASS-ELAPSED-EVENT-ARGS.
      PROCEDURE DIVISION USING BY VALUE PARAM-SENDER PARAM-EVENTARG.
         DISPLAY N"Elapsed イベントが発生しました".
      END METHOD SAMPLE-STATIC-METHOD.

      END STATIC.

      OBJECT.
      PROCEDURE DIVISION.

      *> System.Timers.ElapsedEventHandler デリゲートと同じインタフェースを持つ
      *> オブジェクトメソッドを定義します。
      METHOD-ID. SAMPLE-OBJECT-METHOD.
      DATA DIVISION.
      LINKAGE SECTION.
      01 PARAM-SENDER   OBJECT REFERENCE CLASS-OBJECT.
      01 PARAM-EVENTARG OBJECT REFERENCE CLASS-ELAPSED-EVENT-ARGS.
      PROCEDURE DIVISION USING BY VALUE PARAM-SENDER PARAM-EVENTARG.
         DISPLAY N"Elapsed イベントが発生しました".
      END METHOD SAMPLE-OBJECT-METHOD.

      END OBJECT.
      END CLASS SAMPLE-CLASS.
```

このとき、 SAMPLE-STATIC-METHOD および SAMPLE-OBJECT-METHOD を呼び出すデリゲートオブジェクトを作成するには、 以下のように記述します。

```
...
      REPOSITORY.
         CLASS SAMPLE-CLASS
         DELEGATE DEL-ELAPSED-EVENT-HANDLER AS "System.Timers.ElapsedEventHandler" *> [1]
...
      WORKING-STORAGE SECTION.
```

```
      01 WK-SAMPLE-OBJECT        OBJECT REFERENCE SAMPLE-CLASS.
      01 WK-STATIC-METHOD-DELEGATE OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.     *> [2]
      01 WK-OBJECT-METHOD-DELEGATE OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.     *> [2]
      PROCEDURE DIVISION.
          *> 静的メソッドに対するデリゲートオブジェクトを作成します。
          INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
            USING SAMPLE-CLASS N"SAMPLE-STATIC-METHOD"
            RETURNING WK-STATIC-METHOD-DELEGATE.                              *> [3]

          *> オブジェクトメソッドに対するデリゲートオブジェクトを作成します。
          INVOKE SAMPLE-CLASS "NEW" RETURNING WK-SAMPLE-OBJECT.
          INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
            USING WK-SAMPLE-OBJECT N"SAMPLE-OBJECT-METHOD"
            RETURNING WK-OBJECT-METHOD-DELEGATE.                              *> [3]
...
```

デリゲート型の宣言

リポジトリ段落でデリゲートの外部名を内部名に対応付けます。 デリゲート名を定義するには、デリゲート指定子を使用します（[1]）。

デリゲートオブジェクトのための変数の定義

デリゲートオブジェクトへの参照を格納する変数を定義するには、 変数をデリゲート型のオブジェクト参照として定義します（[2]）。

デリゲートオブジェクトの作成

デリゲートオブジェクトを作成するには、 そのデリゲート型のコンストラクタを呼び出します（[3]）。 すべてのデリゲート型のコンストラクタは 2 つの引数を取ります。

第 1 引数

静的メソッドを呼び出すデリゲートを作成する場合は、 メソッドが定義されているクラス名を記述します。 オブジェクトメソッドを呼び出すデリゲートを作成する場合は、 オブジェクトを識別するオブジェクト参照一意名を指定します。

第 2 引数

メソッド名を英数字定数または日本語定数で記述します。 データ名を記述することはできません。

デリゲートが呼び出すメソッドを記述する場合は、 メソッドの引数の定義をそのデリゲート型の引数と完全に一致させなければなりません。 .NET Framework では値渡し引数を多く用いますが、 COBOL 言語の引数のデフォルトは参照渡しです。 上の例で、SAMPLE-STATIC-METHOD および SAMPLE-OBJECT-METHOD メソッドの手続き部見出しの USING 指定に BY VALUE 句が記述されていることに注意してください。

デリゲートを作成することができるのは、クラスのメソッドに対してだけです。 プログラムに対するデリゲートは作成できません。

NCOBU00496

デリゲートを呼び出す

デリゲートオブジェクトに関連付けられたメソッドを呼び出すには、 デリゲートオブジェクトの「Invoke メソッド」を呼び出します。

デリゲートオブジェクトの Invoke メソッドは、 そのデリゲート型と同じ引数を取ります。

```
...
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-ELAPSED-EVENT-ARGS AS "System.Timers.ElapsedEventArgs"
         DELEGATE DEL-ELAPSED-EVENT-HANDLER AS "System.Timers.ElapsedEventHandler"

...
      LINKAGE SECTION.
      01 PARAM-DELEGATE OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      01 PARAM-SENDER   OBJECT REFERENCE CLASS-OBJECT.
      01 PARAM-EVENTARG OBJECT REFERENCE CLASS-ELAPSED-EVENT-ARGS.
      PROCEDURE DIVISION USING PARAM-DELEGATE PARAM-SENDER PARAM-EVENTARG.
         *> デリゲートオブジェクトに関連付けられたメソッドを呼び出します。
         IF NOT PARAM-DELEGATE = NULL THEN
           INVOKE PARAM-DELEGATE "Invoke" USING PARAM-SENDER PARAM-EVENTARG
         END-IF.
...
```

デリゲートを比較する

デリゲートオブジェクトである 2 つのオブジェクト参照項目を比較すると、 両者が同じオブジェクト（同一インスタンス）かどうかが比較されます。 デリゲートオブジェクトの内容（関連付けているメソッド、対象とするオブジェクト）が同じかどうかは比較されません。 これは、COBOL 言語での比較条件式の仕様です。

```
...
      REPOSITORY.
         CLASS SAMPLE-CLASS  *> デリゲートオブジェクトを作成するの
                             *> サンプルで用いた SAMPLE-CLASS クラスを利用します。
         DELEGATE DEL-ELAPSED-EVENT-HANDLER AS "System.Timers.ElapsedEventHandler"
...
      WORKING-STORAGE SECTION.
      01 WK-DELEGATE-1 OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      01 WK-DELEGATE-2 OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      PROCEDURE DIVISION.
         *> 静的メソッドに対するデリゲートオブジェクトを 2 つ作成します。
         INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
           USING SAMPLE-CLASS N"SAMPLE-STATIC-METHOD"
           RETURNING WK-DELEGATE-1.
         INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
           USING SAMPLE-CLASS N"SAMPLE-STATIC-METHOD"
           RETURNING WK-DELEGATE-2.

         *> 同一オブジェクトかどうか比較します。
         *> （内容の比較ではありません）
         IF WK-DELEGATE-1 = WK-DELEGATE-2 THEN
           *> 条件が偽になるため、このコードは実行されません。
           DISPLAY N"オブジェクト参照が同じ"
         END-IF.
...
```

デリゲートオブジェクトの内容が同じかどうかを比較するには、 そのデリゲート型の等値演算子（op_Equality）または非等値演算子（op_Inequality）を呼び出します。

```
...
      REPOSITORY.
         CLASS SAMPLE-CLASS  *> デリゲートオブジェクトを作成するの
                             *> サンプルで用いた SAMPLE-CLASS クラスを利用します。
         DELEGATE DEL-ELAPSED-EVENT-HANDLER AS "System.Timers.ElapsedEventHandler"
...
      WORKING-STORAGE SECTION.
      01 WK-SAMPLE-OBJECT OBJECT REFERENCE SAMPLE-CLASS.
      01 WK-DELEGATE-1    OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      01 WK-DELEGATE-2    OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      01 WK-DELEGATE-3    OBJECT REFERENCE DEL-ELAPSED-EVENT-HANDLER.
      01 WK-RESULT       PIC 1.
      PROCEDURE DIVISION.
         *> 静的メソッドに対するデリゲートオブジェクトを 2 つ作成します。
         INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
           USING SAMPLE-CLASS N"SAMPLE-STATIC-METHOD"
           RETURNING WK-DELEGATE-1.
         INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
           USING SAMPLE-CLASS N"SAMPLE-STATIC-METHOD"
           RETURNING WK-DELEGATE-2.

         *> オブジェクトメソッドに対するデリゲートオブジェクトを作成します。
         INVOKE SAMPLE-CLASS "NEW" RETURNING WK-SAMPLE-OBJECT.
         INVOKE DEL-ELAPSED-EVENT-HANDLER "NEW"
           USING WK-SAMPLE-OBJECT N"SAMPLE-OBJECT-METHOD"
           RETURNING WK-DELEGATE-3.


         *> WK-DELEGATE-1 と WK-DELEGATE-2 の内容は同じなので、
         *> WK-RESULT には B"1"（真）が設定されます。
         INVOKE DEL-ELAPSED-EVENT-HANDLER "op_Equality"
           USING WK-DELEGATE-1 WK-DELEGATE-2 RETURNING WK-RESULT.

         *> WK-DELEGATE-1 と WK-DELEGATE-2 の内容は同じなので、
         *> WK-RESULT には B"0"（偽）が設定されます。
```

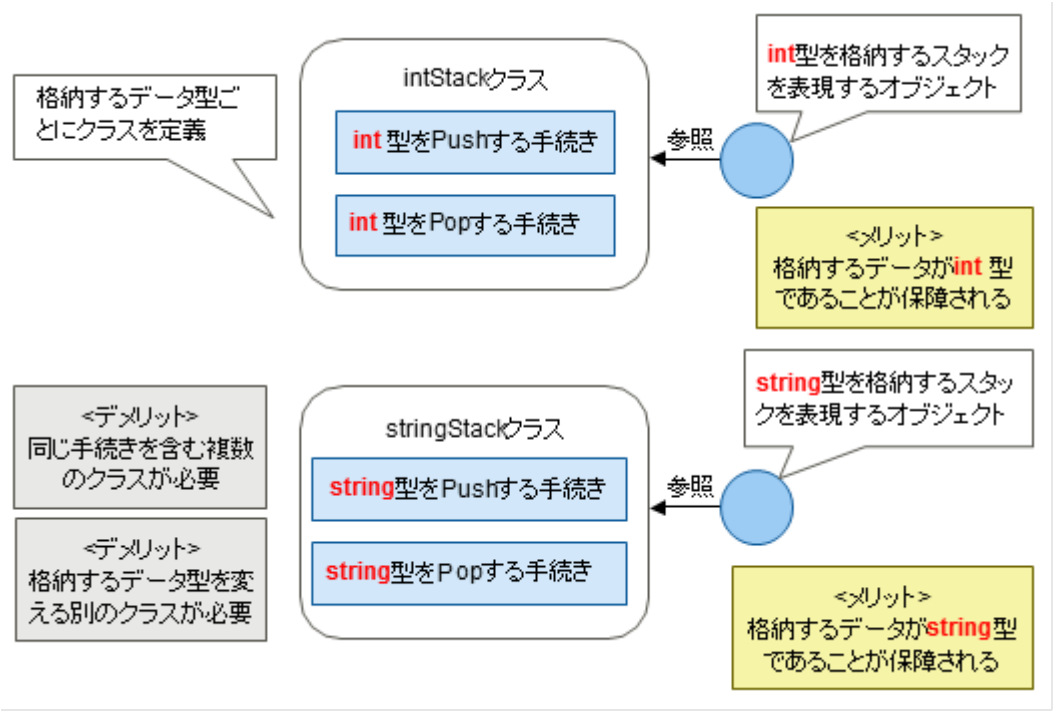
```
           INVOKE DEL-ELAPSED-EVENT-HANDLER "op_Inequality"
             USING WK-DELEGATE-1 WK-DELEGATE-2 RETURNING WK-RESULT.


           *> WK-DELEGATE-1 と WK-DELEGATE-3 の内容は異なるので、
           *> WK-RESULT には B"0"（偽）が設定されます。
           INVOKE DEL-ELAPSED-EVENT-HANDLER "op_Equality"
             USING WK-DELEGATE-1 WK-DELEGATE-3 RETURNING WK-RESULT.

           *> WK-DELEGATE-1 と WK-DELEGATE-3 の内容は異なるので、
           *> WK-RESULT には B"1"（真）が設定されます。
           INVOKE DEL-ELAPSED-EVENT-HANDLER "op_Inequality"
             USING WK-DELEGATE-1 WK-DELEGATE-3 RETURNING WK-RESULT.
...
```

Visual C#のコード中に記述されたデリゲートオブジェクトに対する（各言語としての）等値演算子と非等値演算子は、 それぞれデリゲートの等値演算子と非等値演算子を呼び出す IL に翻訳されます。 したがって、Visual C# や Visual Basic のサンプルを参考に COBOL コードを記述する場合、サンプル中のデリゲートオブジェクトの比較はデリゲートの等値演算子か非等値演算子呼び出しに置き換えるとよいでしょう。

ジェネリックの機能を使う

従来のプログラミングでは、手続きとその手続きで参照する型を分離することはできません。たとえば、int 型データのスタックと string 型データのスタックが必要な場合、従来のプログラミングでは、次のどちらかの方法をとる必要がありました。

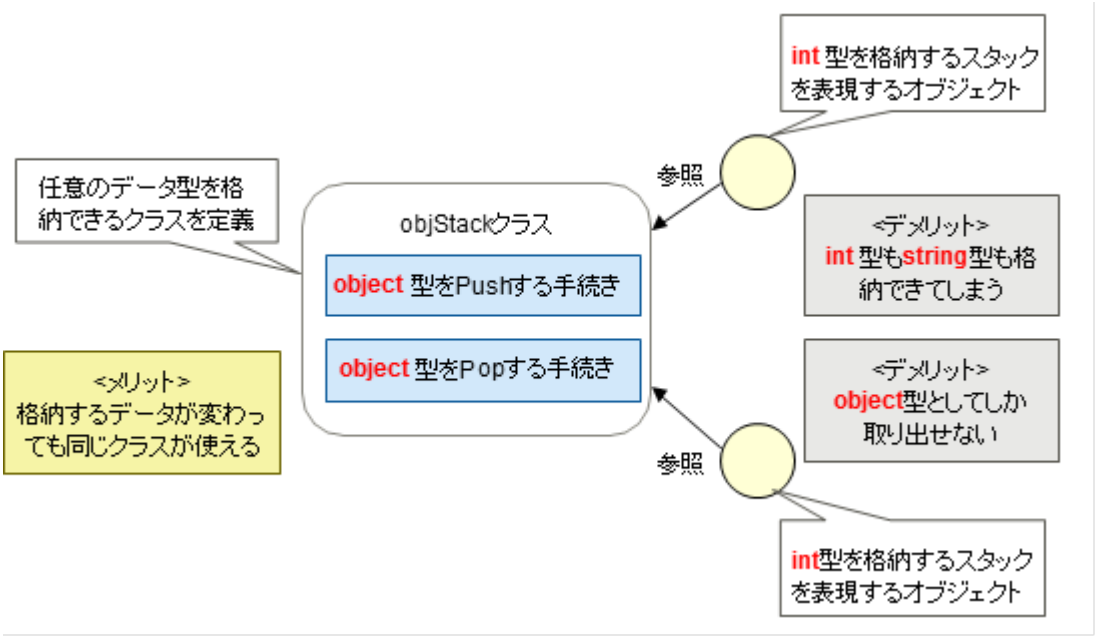
- 「int 型を格納するスタック」と「string 型を格納するスタック」を、それぞれ別のクラスとして定義する。



この場合、各クラスは格納するデータ型に特化して作成されています。そのため、翻訳時には、誤ったデータ型を使用したスタックの操作がチェックされます。しかし、次のようなデメリットがあります。

- 操作対象の型だけが異なる、ほとんど同じクラスを作成する必要がある。
- 別の型を操作するスタックが必要となった場合、さらに新しいクラスを定義する必要がある。

- 「object 型スタックを操作するクラス」を定義する。



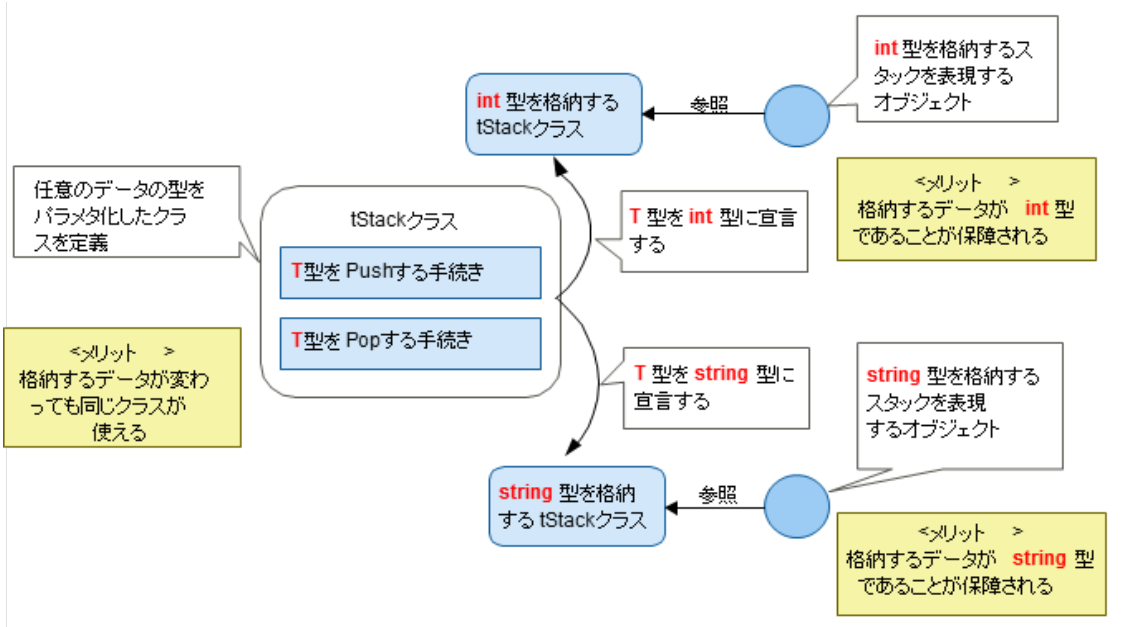
この場合、同じ手続きを含むクラスは１つです。たとえば、別のデータ型を格納するスタックが必要になっても、新しいクラスを作成する必要はありません。しかし、次のようなデメリットがあります。

- データは実際は object 型に格納されているので、取り出す時に適切なキャストが必要となる。
- データは実際は object 型に格納されているので、不適切な型のデータが格納できてしまう。

このような場合に有効な機能として、.NET Framework には、ジェネリックの機能が取り入れられました。

ジェネリックの機能を使用すると、クラスに含まれるメンバーの型や手続きに含まれるデータの型をパラメタ化することができます。そして、ジェネリックを使用する際は、パラメタ化した型に実際の型を当てはめて使用できます。

前述の int 型データのスタックと string 型データのスタックが必要な場合の例で、その概要を説明します。



この例では、tStack クラスに格納するデータの型をパラメタ化しています。この tStack クラスを使用するときに、パラメタ化した型を int 型であると宣言すると、作成されたオブジェクトは「int 型を格納するスタック」のオブジェクトとしてふるまいます。また、パラメタ化した型を string 型であると宣言す

ると、作成されたオブジェクトは「string 型を格納するスタック」のオブジェクトとしてふるまいます。

.NET Framework のジェネリックの機能では、パラメタ化した型を表現する識別子を型パラメタと呼びます。型パラメタ化の方法は、その有効範囲から次の 2 つに分けられます。

- 型のメンバーが参照する型を型パラメタとする。
- メソッドの手続きが参照する型を型パラメタとする。

NetCOBOL for .NET でのそれぞれの利用法について説明します。

総称型を使用したプログラミング

型のメンバーが参照する型が、型パラメタ化された型の場合の使用法について説明します。

型パラメタ付きメソッドを使用したプログラミング

メソッドの手続きが参照する型が、型パラメタ化されたメソッドの場合の使用法について説明します。

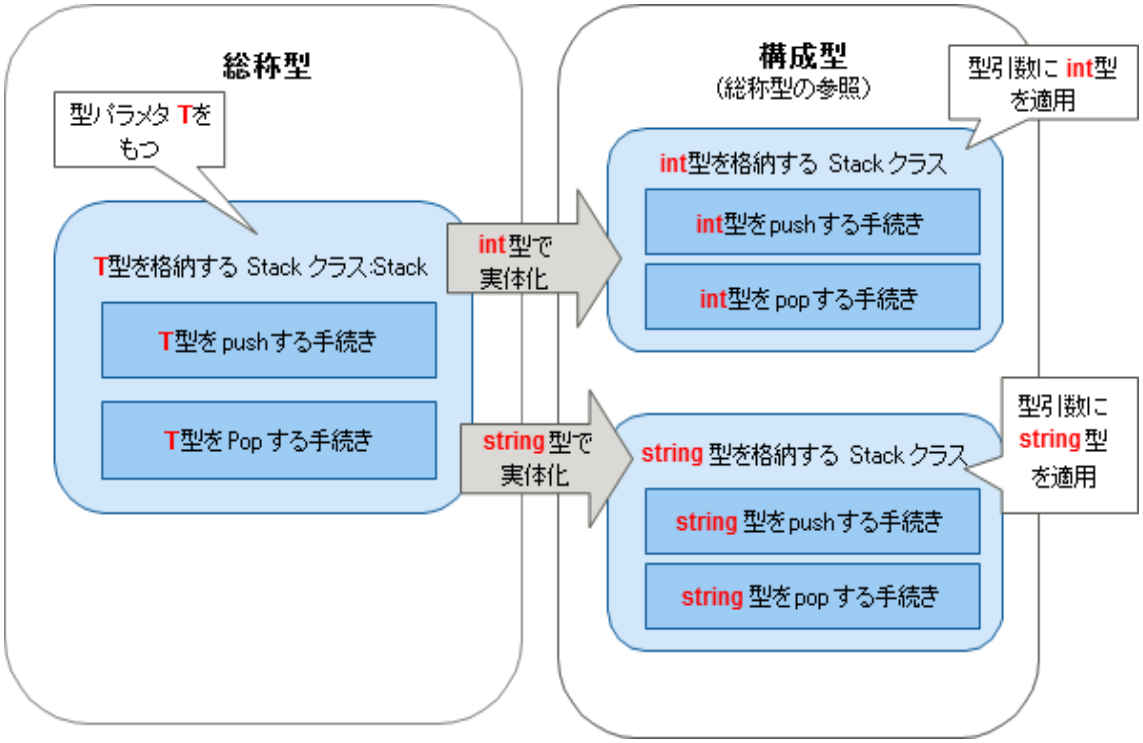
NCOBU00858

総称型を使用したプログラミング

型のメンバーが参照する型が、型パラメタ化された型の場合の使用法について説明します。

概要と用語の説明

格納するデータの型を型パラメタ化した Stack クラスを例にして、ジェネリックの機能の概要と用語を説明します。



- 型自身が型パラメタを持っている型のことを“総称型”と呼びます(この場合、クラス定義であるので特に“総称クラス”と呼びます)。総称型である Stack クラスの手続きでは、格納されるデータの型は、パラメタ化された型 T であるとして操作を記述しています。
- 総称型 Stack を使用する場合、パラメタ化された T 型に適用する具体的な型を指定した宣言が必要となります。これを“(総称型を) 実体化する”と呼びます。
- 実体化するために指定した具体的な型を“型引数”と呼びます。また、実体化した型のことを“構成型”と呼びます。
- int 型の型引数で実体化された構成型のオブジェクトは、「int 型スタックを操作するクラス」のオブジェクトとしてふるまいます。また、string 型の型引数で実体化された構成型のオブジェクトは、「string 型スタックを操作するクラス」のオブジェクトとしてふるまいます。

総称型と型パラメタ

.NET Framework で利用できる各種の総称型と、その中で型パラメタを使用する場合の制限について説明します。

.NET Framework では以下の型に、任意の個数の型パラメタを持たせることができます。

- クラス(構造体を含む)
- インタフェース
- デリゲート

それぞれ、型パラメタを持つものを総称クラス、総称インタフェース、総称デリゲートと呼びます。 これらの型

の定義の中で、型パラメタは次のような場合を除くと、通常の型と同等の操作ができます。

- 基底クラスや基底インタフェースリストに指定する型
- スタティックメンバーや入れ子の型にアクセスするために指定する型
- パラメタを必要とする new 式に指定する型

総称型を利用したプログラミングの手順

総称型を利用したプログラミングは、次の手順で行います。

1. 総称型を定義する
2. 総称型に型引数を与えて構成型を宣言する
3. 構成型のメンバーを参照する

NetCOBOL for .NET では、構成型を宣言すること、およびその構成型のメンバーを参照することができますが、総称型そのものは定義できません。このため、NetCOBOL for .NET で総称型を利用したプログラミングを行う場合、次のような総称型の定義を参照してプログラミングを行います。

- .NET Framework のクラスライブラリに含まれる総称型(たとえば、System.Collections.Generic 名前空間の型)
- C#/VB.NET などでユーザが定義した総称型

このため、以降の説明では総称型については C#の記述形式を用いて説明します。 たとえば、C#では、型パラメタを 1 つ持つ Stack クラスは次のように表現されます。

```
Class Stack<T> { … }
```

T が型パラメタの名前で、複数の型パラメタがある場合は、記号“<”と“>”の間をカンマ(“,”)で区切って型パラメタの名前を列記します。

以下では、このような総称型を参照して、COBOL で構成型を宣言し、利用する方法について説明します。

構成型を宣言する

構成型の宣言の方法について説明します。

構成型を使用する

リポジトリ段落で宣言した構成型およびそのメンバーにアクセスする方法について説明します。

構成型を継承/実装する

構成型を継承あるいは実装する方法について説明します。

NCOBU00859

構成型を宣言する

NetCOBOL for .NET での構成型の宣言は、通常の型の宣言と同じようにリポジトリ段落で行います。ただし、総称型と型引数を対応付けるため、クラス指定子、インタフェース指定子、デリゲート指定子の書き方が拡張されています。

構成型をリポジトリ段落で宣言する

総称クラスの場合を例にして、構成型を宣言する方法を説明します。 総称クラスから構成型を宣言する場合、次の 1 組のクラス指定子を使用する必要があります。

1. 総称クラスの使用を宣言するクラス指定子

   構成型のテンプレートにあたる総称型を宣言します。総称型の宣言は、外部名の書き方に特徴がある点を除いて、通常のクラス指定子によるクラス宣言と同じです。

   ```
   CLASS クラス名-1 AS “クラス外部名”
   ```

   総称型のクラスを宣言する場合、クラス指定子の AS 指定に定数で指定するクラスの外部名は、次の形式になります。

   ```
   名前<[{, }…]>
   ```

   名前に続く記号“<”と“>”は、クラス名が総称クラスであることを表します。また、記号“<”と“>”の間のカンマ(“,”)の数は、型パラメタの個数を表します(型パラメタの個数＝カンマの個数＋1)。 このクラス指定子で宣言したクラス名-1 が総称型のクラス名となります。

2. 構成型を宣言するクラス指定子

   テンプレートとなる総称型と型引数の組合わせに名前を付けて構成型を宣言します。このため、次に示す拡張されたクラス指定子を使用します。

   ```
   CLASS クラス名-2 EXPANDS クラス名-1
                USING {クラス名-3} …
   ```

   クラス指定子の EXPANDS 指定に、テンプレートとなる総称型のクラス名を指定し、USING 指定に型引数とするクラスの名前を指定します。 このクラス指定子で宣言したクラス名-2 が、構成型のクラス名となります。

   型パラメタを 1 つ持つ Stack クラスに対して、int 型を型引数とする構成型と String 型を型引数とする構成型を宣言する場合の例を次に示します。

   ```
   ...
         REPOSITORY.
            *>型引数に指定するクラスを宣言します。
             CLASS CLASS-INT  AS “System.Int32”
             CLASS CLASS-STR  AS “System.String”
            *>総称型 Stack<T>クラスを宣言します。
            *>外部名に<>をつけることにより、この型が総称型であることと、
            *>その型パラメタの個数が 1 個であることを示します。
             CLASS CLASS-STACK AS "System.Collections.Generic.Stack<>”
            *>構成型 Stack<int>クラスを宣言します。
             CLASS CLASS-INT-STACK EXPANDS CLASS-STACK
                                 USING CLASS-INT
            *>構成型 Stack<String>クラスを宣言します。
             CLASS CLASS-STR-STACK EXPANDS CLASS-STACK
                                 USING CLASS-STR
   ...
   ```

   総称インタフェースから構成型を宣言する場合および総称デリゲートから構成型を宣言する場合は、ク

ラス指定子の代わりに、それぞれインタフェース指定子、デリゲート指定子を使用することを除けば、同じ方法で構成型を宣言することができます。

総称型の名前は、リポジトリ段落の各指定子の EXPANDS 指定でだけ参照できます。 それ以外の場所では、総称型の名前を参照できません。

型引数と制約

型パラメタには適用可能な型引数を制限するための条件付けができます。この条件付けを“制約”と呼びます。

型引数を指定するときは、型パラメタに付けられた制約を満たす型を指定する必要があります。 制約の種類と概要は以下のとおりです。

- 参照型制約 型引数は参照型でなければなりません。
- 値型制約 型引数は値型でなければなりません。
- 基底クラス制約 型引数は、基底クラスとして規定されたクラスそのもの、またはそのクラスの派生クラスでなければなりません。
- 基底インタフェース制約 型引数は、以下のどれかでなければなりません。
  - 基底インタフェースとして規定されたインタフェースそのもの
  - 基底インタフェースとして規定されたインタフェースの派生インタフェース
  - 基底インタフェースを実装したクラス
- コンストラクター制約 型引数は、引数無しのインスタンスコンストラクターを持つか、または値型でなければなりません。

構成型の配列の宣言

通常の配列オブジェクトの利用と同じように、リポジトリ段落で配列の型名を宣言します。このとき、総称型の宣言の外部名に角括弧([])を追加して、配列型であることを示します。

型パラメタを 1 つ持つ Stack<T>型の配列に対して、int 型を型引数として構成型を宣言して、Stack<int>型の配列を宣言する例を次に示します。

```
...
      REPOSITORY.
         CLASS CLASS-INT  AS  “System.Int32”
         CLASS STACK-ARRAY AS "System.Collections.Generic.Stack<>[]”
         CLASS INT-STACK-ARRAY EXPANDS STACK-ARRAY
                           USING CLASS-INT
...
```

関連トピックス

[.NET Framework の配列を使う]

入れ子の型の構成型

.NET Framework では、型の内部に別の型(入れ子の型)を宣言できます。この入れ子には、次の 2 つの方法で型パラメタを指定できます。

1. 入れ子の型を含む型が総称型 入れ子の型の内部で、この型を含む型の型パラメタは有効です。
2. 入れ子の型自身が総称型 入れ子の型の型パラメタは、この入れ子の型の内部でだけ有効です。

入れ子の型の外部名は、外側の型の名前と内側の型の名前を、区切り文字"+"で連結した名前です。入れ子の型またはそれを含む型が総称型である場合、区切り文字"+"で連結する名前それぞれについて型パラメタの有無を指定します。

また、入れ子の型の構成型を宣言する場合、入れ子の型の型パラメタの分だけでなく、それを含む型の持つ型パラメタの分も合わせて型引数を指定する必要があります。 たとえば、Outer<T1>クラス内に定義された入れ子のクラス Inner<T2>の構成型を宣言する場合は、次のようにします。

```
...
      REPOSITORY.
         CLASS CLASS-INNER AS "Outer<>+Inner<>"
         CLASS CLASS-INNER-INT EXPANDS CLASS-INNER
                             USING CLASS-INT32 CLASS-INT32
         *>型引数は外側の型の分もすべて指定します。
...
```

関連トピックス

[入れ子になった型を使う]

構成型を使用する

NetCOBOL for .NET では、リポジトリ段落で宣言した構成型を、通常の型と同じように使用できます。

ここでは、Stack<T>に int 型を型引数として宣言した構成型 Stack<int>を使用する場合を例に説明します。

```
...
      REPOSITORY.
         *> 型引数に指定するクラスを宣言します。
          CLASS CLASS-INT  AS "System.Int32"
         *> 総称型 Stack<T>クラスを宣言します。
          CLASS CLASS-STACK AS "System.Collections.Generic.Stack<>"
         *> 構成型 Stack<int>クラスを宣言します。
          CLASS CLASS-INT-STACK EXPANDS CLASS-STACK
                            USING CLASS-INT
...
```

構成型のデータを定義し、インスタンスを生成する

リポジトリ段落で宣言した構成型のクラス名 CLASS-INT-STACK を次のように使用して、この構成型 Stack<int>型を参照するデータ項目を宣言します。また、そのデータ項目には、生成したインスタンスを格納できます。

```
...
      01 INTSTACK-OBJ     OBJECT REFERENCE CLASS-INT-STACK.
      01 WDATA           BINARY-LONG.
...
      PROCEDURE     DIVISION.
         *> インスタンスを生成
         INVOKE CLASS-INT-STACK "NEW" RETURNING INTSTACK-OBJ
...
```

構成型のメンバーを使用する

データ部で定義した構成型 Stack<int>型を参照するデータ項目、またはクラス名 CLASS-INT-STACK を使用して、この構成型のメンバーを使用できます。 たとえば、次のように Stack<int>型を参照するデータ項目を使用して、メソッドを呼び出したり、プロパティを参照することができます。

```
...
         *> 100 より小さい奇数をスタックに格納する。
         PERFORM TEST BEFORE
              VARYING WDATA FROM 1 BY 2 UNTIL 100 < WDATA
           INVOKE INTSTACK-OBJ  "Push" USING BY VALUE WDATA
         END-PERFORM
         *> スタック内の要素の個数を取得して表示
         MOVE P-COUNT OF INTSTACK-OBJ TO WMAX
         DISPLAY WMAX
         *> スタック内の要素を逆順に表示
         PERFORM WMAX TIMES
           INVOKE INTSTACK-OBJ  "Pop" RETURNING WDATA
           DISPLAY WDATA
         END-PERFORM
...
```

この例で使用した Stack<T>では、メソッド Push および Pop は次のように定義されています。

```
public class Stack<T>{
    public void Push(T p){ … }
    public T Pop(){ … }
}
```

しかし、int 型の型引数で実体化した構成型のインスタンスから参照する場合、次のように定義されていると見なされます。

```
   public void Push(int p){ … }
   public int Pop(){ … }
```

このため、WDATA の型が int 型に適合しない型(たとえば String 型)の場合、翻訳時にエラーとなります。

型名を使用したその他の操作

型名を使用する次のような操作に、構成型のクラス名 CLASS-INT-STACK を使用できます。

- 型のキャスト(オブジェクト指定子)
- 型の type オブジェクトを得る(TYPE-OF 特殊レジスタ)
- オブジェクトが型に適合するかチェックする(型条件)

総称インタフェースの構成型

インタフェースは、スタティックなメンバーとフィールドを持てない抽象的な型であるので、インタフェースオブジェクトに対する操作は次のような場合に限定されます。

- インタフェースを参照するデータ項目の定義
- インタフェースを参照するデータ項目を使用してのオブジェクトメンバーの参照

この点は、総称インタフェースの構成型の場合も同じです。

以下に、総称インタフェースの構成型を参照するデータ項目を定義する例を示します。

```
...
      REPOSITORY.
         *> 型引数に指定するクラスを宣言します。
          CLASS CLASS-INT  AS "System.Int32"
         *> 総称型 IStack<T>クラスを宣言します。
          INTERFACE INTERFACE-STACK AS "IStack<> "
         *> 構成型 IStack<int>クラスを宣言します。
          INTERFACE INTERFACE-INT-STACK EXPANDS INTERFACE-STACK
                            USING CLASS-INT
...
      *> リポジトリ段落で宣言した構成型の名前を参照してデータ項目を定義
      01 INTSTACK-OBJ     OBJECT REFERENCE INTERFACE-INT-STACK.
      01 WDATA           PIC S9(9) COMP-5.
...
```

また、総称インタフェースの構成型を参照するデータ項目を使用して、オブジェクトメンバーを参照する例を示します。

```
...
      PROCEDURE     DIVISION USING WKIND.
         *>インスタンスを生成・設定
         IF WKIND = “A” THEN
           INVOKE CLASS-INT-STACK1 "NEW" RETURNING INTSTACK-OBJ
         ELSE
           INVOKE CLASS-INT-STACK2 "NEW" RETURNING INTSTACK-OBJ
         END-IF
         *> 100 より小さい奇数をスタックに格納する。
         PERFORM TEST BEFORE
              VARYING WDATA FROM 1 BY 2 UNTIL 100 < WDATA
           INVOKE INTSTACK-OBJ  "Push" USING BY VALUE WDATA
         END-PERFORM
...
```

この例では、CLASS-INT-STACK1 クラスまたは CLASS-INT-STACK1 のインスタンスを総称インタフェースの構成型を参照する データ項目に設定していますが、このクラスは構成型 IStack<int>を実装するクラスならば、どのようなクラスでも構いません。

総称デリゲートの構成型

デリゲートは、特定の型またはオブジェクトのメソッド呼び出すための情報を保持するために特化された特殊な

型であり、その操作は次の場合に限定されます。

- デリゲートを参照するデータ項目の定義
- 特定の型・オブジェクトのメソッドとデリゲートオブジェクトの対応付け
- デリゲートオブジェクトからのメソッド呼び出し

この点は、総称デリゲートの構成型の場合も同じです。

以下に、総称デリゲートの構成型を参照するデータ項目を定義する例を示します。

```
...
      REPOSITORY.
         *> 型引数に指定するクラスを宣言します。
          CLASS CLASS-INT  AS "System.Int32"
         *> 総称型 IStack<T>クラスを宣言します。
          DELEGATE DELEGATE-ADD AS "ADD<> "
         *> 構成型 ADD<int>クラスを宣言します。
          DELEGATE DELEGATE-INT-ADD EXPANDS DELEGATE-ADD
                            USING CLASS-INT
...
      *> リポジトリ段落で宣言した構成型の名前を参照してデータ項目を定義
      01 ADD-DELEGATER    OBJECT REFERENCE DELEGATE-INT-ADD.
      01 INT-LIST-OBJ     OBJECT REFERENCE CLASS-INT-LIST.
      01 INT-STACK-OBJ       OBJECT REFERENCE CLASS-INT-ARRAY.
...
```

また、総称デリゲートの構成型を参照するデータ項目に特定の型・オブジェクトのメソッドを対応づけ、 そのメソッド呼び出す例を示します。

```
...
      PROCEDURE     DIVISION USING WKIND.
         *>インスタンスを生成・設定
         IF WKIND = “A” THEN
           INVOKE DELEGATE-INT-ADD  USING BY VALUE INT-LIST-OBJ
                                    BY VALUE “ADD”
                                RETURNING ADD-DELEGATER
         ELSE
           INVOKE DELEGATE-INT-ADD  USING BY VALUE INT-STACK-OBJ
                                    BY VALUE “PUSH”
                                RETURNING ADD-DELEGATER
         END-IF
...
         *> デリゲートオブジェクトに対応づけられたメソッドの呼び出し
         INVOKE ADD-DELEGATER "INVOKE" USING BY VALUE 10
...
```

NCOBU00861

構成型を継承/実装する

NetCOBOL for .NET では、総称クラスや総称インタフェースの構成型を継承あるいは実装した型を定義できます。構成型を継承あるいは実装して新しい型を定義する方法は、通常の型を使用する場合と同じです。

構成型を継承して型を定義する

総称クラスの構成型を継承して新しいクラスを定義した場合、そのクラスは構成型ではなく通常のクラスです。同じように総称インタフェースの構成型を継承して、新しいインタフェースを定義することができます。そのインタフェースは構成型ではなく通常のインタフェースです。

構成型を実装して型を定義する

次の例は、総称インタフェースの構成型を実装してクラスを定義しています。

```
 CLASS-ID.     INT-ENUMERATOR.
 ENVIRONMENT   DIVISION.
 CONFIGURATION  SECTION.
 REPOSITORY.
    CLASS    CLASS-OBJ AS "System.Object"
    CLASS    CLASS-INT AS "System.Int32"
    CLASS    CLASS-BOL AS "System.Boolean"
    INTERFACE IENUMERATOR0 AS "System.Collections.IEnumerator"
    INTERFACE IENUMERATOR1 AS "System.Collections.Generic.IEnumerator<>"
    INTERFACE INT-IENUMERATOR EXPANDS IENUMERATOR1
            USING CLASS-INT
…
 *> 実装するインタフェースの１つに IEnumerator<int>を指定。
 OBJECT.         IMPLEMENTS INT-IENUMERATOR IENUMERATOR0.
…
 METHOD-ID.    MoveNext.
 DATA         DIVISION.
 LINKAGE       SECTION.
 01 LK1       OBJECT REFERENCE CLASS-BOL.
 PROCEDURE     DIVISION RETURNING LK1.
…
 END CLASS   INT-ENUMERATOR.
```

このクラスは、構成型ではなく通常のクラスです。

構成型の一致と適合

2 つの構成型の間で型が一致するのは、次の場合だけです。

- テンプレートとなった総称型が一致する
- 型引数がすべて等しい

2 つの構成型の間で適合関係が生じるのは、次の場合だけです。

- テンプレートとなった総称型同士に適合関係がある
- 型引数がすべて等しい

型パラメタ付きメソッドを使用したプログラミング

メソッドの手続きが参照する型が、メソッド自身の持つ型パラメタによってパラメタ化されている場合のプログラミングについて説明します。

概要

総称型に含まれるメソッドは、総称型が持つ型パラメタにより、手続き中で参照する型がパラメタ化されています。このようなメソッドのふるまいは、総称型に型引数を指定して構成型を宣言した際に決定されます。 .NET Framework のジェネリックの機能では、これとは別にメソッドそのものが型パラメタを持つこともできます。このようなメソッドの場合、手続きの参照する型は、メソッド自身の持つ型パラメタによってパラメタ化されています。 型パラメタを持つメソッドは、通常の型あるいは総称型どちらのメンバーとしても定義できますが、メソッドの持つ型パラメタのスコープはそのメソッド内に閉じています。 どちらのメソッドも、型パラメタに指定した型に依存してふるまいが変わるメソッド(総称メソッド)です。 しかし、メソッド自身が型パラメタを持つ場合、実際にメソッドを呼び出す際に明または暗に型 引数を指定するまで、メソッドのふるまいは決定されません。このため、特にこれを“型パラメタ付きのメソッド”と呼びます。

型パラメタ付きメソッドを使用したプログラミングの手順

型パラメタ付きメソッドを利用したプログラミングは、次の手順で行います。

1. 型パラメタ付きメソッドを定義する
2. 型パラメタ付きメソッドの型引数を与えて実体化する
3. 実体化した型パラメタ付きメソッドを呼び出す

NetCOBOL for .NET では、型パラメタ付きメソッドに型引数を与えて実体化すること、および実体化された型パラメタ付きメソッドを呼び出すことはできます。しかし、型パラメタ付きメソッドそのものは定義できません。このため、NetCOBOL for .NET で型パラメタ付きメソッドを利用したプログラミングを行う場合、次のような総称型の定義を参照してプログラミングを行います。

- .NET Framework のクラスライブラリに含まれる型パラメタ付きメソッド (たとえば、System.Collections.Generic.List<T>.ConvertAll<>(TOutput)メソッド)
- C＃/VB.NET などでユーザが定義した型パラメタ付きメソッド

このため、以降の説明では型パラメタ付きメソッドについては C＃の記述形式を用いて説明します。 たとえば、C＃では、型パラメタを 1 つ持つ Compare メソッドは次のように表現されます。

```
int Compare<T>(T p1, T p2){ … }
```

T が型パラメタの名前で、複数の型パラメタがある場合は、記号“<”と“>”の間をカンマ(“,”)で区切って型パラメタの名前を列記します。

以下では、このような型パラメタ付きメソッドを COBOL から利用する方法について説明します。 NetCOBOL for .NET では実際に型パラメタ付きのメソッドを使用する方法として 2 つの方法が用意されています。

[型引数を指定しないで型パラメタ付きメソッドを呼び出す]

新しい構文を使用せずに型パラメタ付きのメソッドを呼び出す方法を説明します。

[型引数を明示的に指定して型パラメタ付きのメソッドを呼び出す]

新しい構文を使用して、明示的に型引数を指定してパラメタ付きのメソッドを呼び出す方法を説明します。

NCOBU00863

型引数を指定しないで型パラメタ付きメソッドを呼び出す

型パラメタ付きのメソッドを呼び出す前に、型引数を指定する必要があります。

NetCOBOL for .NET では、メソッド呼び出しに使用する通常の INVOKE 文で、型引数の指定とメソッドの呼び出しを同時に行うことができます。

型パラメタの実体化をともなうメソッド呼び出し

次のような型パラメタ付きメソッドを例にして説明します。

```
//  (C#)
public class Utilty {
    public int Compare<T>(T p1, T p2){ … }
}
```

このメソッドは次のようにして呼び出すことができます。

```
...
      REPOSITORY.
         CLASS CLASS-STR AS "System.String"
         CLASS CLASS-UTL AS "Utilty".
      DATA         DIVISION.
      WORKING-STORAGE SECTION.
      01 UTL-OBJ    OBJECT REFERENCE CLASS-UTL.
      01 RESULT     BINARY-LONG.
      01 INT1       BINARY-LONG.
      01 INT2       BINARY-LONG.
      01 STR1       OBJECT REFERENCE CLASS-STR.
      01 STR2       OBJECT REFERENCE CLASS-STR.
      PROCEDURE     DIVISION.
...
          *> USING 指定のデータ項目の型から
          *> 型引数は int32 型と見なされる
          INVOKE UTL-OBJ "Compare" USING INT1 INT2
                                 RETURNING RESULT
...
          *> USING 指定のデータ項目の型から
          *> 型引数は String 型と見なされる
          INVOKE UTL-OBJ "Compare" USING STR1 STR2
                                 RETURNING RESULT
...
```

これは、INVOKE 文の USING 指定に指定されたデータ項目の型から、型引数の型が推測可能なためです。上記の例では、それぞれ次のように型引数の型を推測しています。

```
public int Compare<int>(int p1, int p2){ … }
public int Compare<String>(String p1, String p2){ … }
```

メソッドの持つすべての型パラメタに対する、型引数の型の推測に矛盾がなければ、推測によって得られた型引数を使用した、メソッドの型パラメタの実体化は、コンパイラによって自動時に行われます。 なお、型の推測の過程で同じ型パラメタに対する型引数に複数の候補が見つかった場合、型の推測は失敗します。以下にその例を示します。

```
...
          *> USING 指定のデータ項目の型から
          *> 型引数の型の候補は int 型と String 型
          INVOKE UTL-OBJ "Compare" USING INT1 STR2
                                 RETURNING RESULT
...
```

デリゲートオブジェクト生成時の型パラメタの実体化

型または特定のオブジェクトのメソッドに対して、デリゲートオブジェクトを生成する場合も、通常の INVOKE 文の場合と同じように型パラメタを実体化することができます。

先の例で説明した、Utilty クラスの Compare<T>メソッドをデリゲートするデリゲートオブジェクトを生成する場合を例にして説明します。

```
      DELEGATE-ID.    INT2-DELEGATOR.
      ENVIRONMENT    DIVISION.
      DATA          DIVISION.
      LINKAGE        SECTION.
      01 LK1        BINARY-LONG.
      01 LK2        BINARY-LONG.
      01 LK3        BINARY-LONG.
      PROCEDURE     DIVISION USING BY VALUE  LK1
                             BY VALUE  LK2
                             RETURNING LK3
      END DELEGATE    INT2-DELEGATOR.
```

生成するデリゲートオブジェクトが参照するデリゲートが、上記のような内容である場合、次のようにしてデリゲートオブジェクトを生成できます。

```
...
      REPOSITORY.
         CLASS CLASS-UTL AS "Utilty"
         DELEGATE INT2-DELEGATOR.
         .
      DATA          DIVISION.
      WORKING-STORAGE SECTION.
      01 DLGT-OBJ    OBJECT REFERENCE INT2-DELEGATOR.
      01 UTY-OBJ     OBJECT REFERENCE CLASS-UTL.
...
         *> デリゲート定義 INT2-DELEGATOR の持つ USING パラメタのデータ型
         *> から型引数の型は int32 と推測される
         INVOKE INT2-DELEGATOR "NEW" USING UTY-OBJ
                                       "Compare"
...
```

この場合、生成するデリゲートオブジェクトの型が持つ USING パラメタのデータ型が、Compare<T>メソッドの型引数の推測に用いられます。

その結果、この INVOKE 文で生成されたデリゲートオブジェクト UTY-OBJ から呼び出すことができるメソッドは、次の形のものだけになります。

```
public int Compare<int>(int p1, int p2){ … }
```

型の推測実施の条件

通常の INVOKE 文で、型引数の型の推測と型パラメタの実体化が行えるのは、呼び出される型パラメタ付きのメソッドの呼び出しが、次の条件に当てはまらない場合です。

- メソッドの型パラメタが USING 仮引数に現れない場合
- USING 仮引数が構成型で、その型引数にメソッドの型パラメタが使用されている場合

また、呼び出し INVOKE 文が次の条件を満たす場合も、型の推測は行われません。

- USING に指定したデータ項目に、COBOL 独自型の項目が含まれている場合

メソッドのオーバロードと型パラメタ付きメソッド

従来より、.NET Framework では、パラメタの個数や型の違いから同名のメソッドを複数定義することができました(メソッドのオーバロード)。これと同じように、型パラメタの有無、個数の違いからメソッドをオーバロードすることができます。 たとえば、次の例では型パラメタの有無・個数が異なるメソッドが複数定義されています。

```
//  (C#)
public class Utilty{
    public int Compare(int p1, int p2){ … }        // (1)
    public int Compare<T>(T p1, T p2){ … }         // (2)
    public int Compare<T1, T2>(T1 p1, T2 p2){ … }   // (3)
}
```

このように、型パラメタの個数が異なる同名のメソッドがある場合、通常の INVOKE 文によってメソッドを呼び出すと、呼び出されるメソッドの束縛は次のように行われます。

1. 型パラメタを持たないメソッドを対象に、適合するメソッドがあるか検索します。 適合するメソッドがあれば、そのメソッドを呼び出します。
2. 型パラメタを持たないメソッドに適合するメソッドがないのなら、型パラメタを 1 つだけ持つメソッドを対象に型の推測を実施します。適合するメソッドがあれば、そのメソッドを呼び出します。
3. 見つからなければ、型パラメタを 2 つ持つメソッドを対象に 2.と同じことを繰り返します。

従って、ここで示した Compare メソッドを次のように呼び出す場合、常に(1)のメソッドが呼び出されます。

```
...
    01 INT1   BINARY-LONG.
    01 INT2   BINARY-LONG.
...
         INVOKE UTL-OBJ "Compare" USING INT1 INT2
                                RETURNING RESULT
...
```

関連トピックス

メソッドを利用する

型引数を明示的に指定して型パラメタ付きのメソッドを呼び出す

NetCOBOL for .NET では､型パラメタ付きのメソッドを呼び出す際に明示的に型引数に使用する型を指定して、それによって実体化されたメソッドを呼び出すことができます。 これは次のような場合に使用されます。

- 呼び出す型パラメタ付きメソッドが型引数の型の推測が可能な条件を満たさない
- 型引数の型を推測によって決定させたくない

メソッド識別名の宣言

メソッド識別名は、リポジトリ段落でメソッド指定子を使用して宣言します。 その方法は、総称型から構成型を宣言する場合と同じように、次の 1 組のメソッド指定子を使用する必要があります。

- 使用する型パラメタ付きのメソッドを宣言するメソッド指定子

  使用する型パラメタ付きのメソッドについて、その名前と型パラメタの個数を宣言します。

  ```
  METHOD メソッド識別名-1 AS “メソッド外部名”
  ```

  このメソッド指定子の AS 指定に定数で指定するメソッドの外部名は、次の形式になります。

  ```
  名前<[{, }…]>
  ```

  名前に続く記号“<”と“>”は、メソッド識別名が型パラメタ付きのメソッドのものであることを表します。また、記号“<”と“>”の間のカンマ(“,”)の数は、型パラメタの個数を表します(型パラメタの個数=カンマの個数+1)。

  このメソッド指定子で宣言したメソッド識別名-1 が、型パラメタ付きのメソッドのメソッド識別名になります。

- 型引数の並びを指定するメソッド指定子

  型パラメタ付きのメソッドと、その実体化のために指定する型引数の組み合わせに名前を付けてメソッド識別名を宣言します。このため、次のメソッド指定子を使用します。

  ```
  METHOD メソッド識別名-2 EXPANDS メソッド識別名-1
                USING{型名-1} …
  ```

  メソッド指定子の EXPANDS 指定に、実体化の対象となる型パラメタ付きのメソッドのメソッド識別名を指定し、USING 指定に型引数とする型の名前を指定します。

  このメソッド指定子で宣言したメソッド識別名-2 が、型引数の並びを指定するメソッド識別名となります。

  C#で定義された型パラメタ付きのメソッドを例にして､メソッド識別名を宣言する方法を次に示します。

  ```
  REPOSITORY.
     CLASS CLASS-INT AS "System.Int32"
     CLASS CLASS-LNG AS "System.Int64"
     CLASS CLASS-UTL AS "Utilty"
     METHOD COMPARE1 AS "Compare<>"
     METHOD COMPARE2 AS "Compare<,>"
     *> Compare<long>を宣言
     METHOD COMP-LNG EXPANDS COMPARE1
                  USING CLASS-LNG
     *> Compare<int,int>を宣言
     METHOD COMP-INT EXPANDS COMPARE2
                  USING CLASS-INT CLASS-INT
  ```

関連トピックス

COBOL 文法書の「11.6.1.3 リポジトリ段落（REPOSITORY）」

メソッド指定子の書き方について説明しています。

メソッド識別名を使用したメソッドの呼び出し

メソッド識別名を使用してメソッドを呼び出す場合、INVOKE 文で呼び出すメソッドを指定する際に、定数ではなくメソッド識別名を使用します。 “B.型引数の並びを指定するメソッド指定子”の例で宣言したメソッド識別名を使用してメソッドを呼び出す場合の例を、次に示します。

```
...
      WORKING-STORAGE SECTION.
      01 UTL-OBJ    OBJECT REFERENCE CLASS-UTL.
      01 RESULT     BINARY-LONG.
      01 INT1       BINARY-LONG.
      01 INT2       BINARY-LONG.
      01 LNG1       BINARY-DOUBLE.
      PROCEDURE     DIVISION.
...
         *> COMP-LNG の指定で実体化されたため、
         *> Compare<long>(long,long)
         INVOKE UTL-OBJ COMP-LNG USING LNG1
                              RETURNING RESULT
         *> COMP-INT の指定で実体化されたため、
         *> Compare<int,int>(int,int)
         INVOKE UTL-OBJ COMP-INT USING INT1 INT2
                              RETURNING RESULT
...
```

メソッド識別名を使用したデリゲートオブジェクトの生成

メソッド識別名は、デリゲートオブジェクトを生成する際に呼び出すメソッドを対応付けるためにも使用できます。 その例を次に示します。

```
...
      REPOSITORY.
         CLASS CLASS-UTL AS "Utilty"
         DELEGATE INT2-DELEGATOR
         CLASS CLASS-INT AS "System.Int32"
         METHOD COMPARE1 AS "Compare<>"
         METHOD COMP-INT EXPANDS COMPARE1
                       USING CLASS-INT.
      DATA          DIVISION.
      WORKING-STORAGE SECTION.
      01 DLGT-OBJ    OBJECT REFERENCE INT2-DELEGATOR.
      01 UTY-OBJ     OBJECT REFERENCE CLASS-UTL.
...
         INVOKE INT2-DELEGATOR "NEW" USING UTY-OBJ
                                           COMP-INT
...
```

入れ子になった型を使う

.NET プラットフォームでは、 クラスの中に型（クラス、インタフェース、構造体、列挙体、デリゲート）を定義することができます。 たとえば、System.Windows.Forms.ListBox クラスは Windows Forms の GUI 部品であるリストボックスを表すクラスですが、その中にリスト項目の集まりを管理する ObjectCollection クラスが定義されています。 このようなクラスの中に定義された型を「入れ子になった型」と呼びます。

NetCOBOL for .NET で入れ子になった型を使う場合、 リポジトリ段落に記述する型の外部名の書き方に特徴がある点を除いて、通常の型の使い方と変わりありません。

入れ子になった型の外部名の書き方

リポジトリ段落に入れ子になった型を記述する場合、型の外部名は、その型を定義しているクラスの外部名と入れ子の型の名前を区切り文字"+"で連結した名前になります。

```
...
      REPOSITORY.
         CLASS CLASS-LISTBOX AS "System.Windows.Forms.ListBox"
         CLASS CLASS-OBJECTCOLLECTION AS "System.Windows.Forms.ListBox+ObjectCollection"
         PROPERTY PROP-ITEMS AS "Items"
...
      WORKING-STORAGE SECTION.
      01 WK-LISTBOX         OBJECT REFERENCE CLASS-LISTBOX.
      01 WK-OBJECTCOLLECTION OBJECT REFERENCE CLASS-OBJECTCOLLECTION.
      PROCEDURE DIVISION.
         *> ListBox オブジェクトを作成します。
         INVOKE CLASS-LISTBOX "NEW" RETURNING WK-LISTBOX.

         *> ListBox オブジェクトの Items プロパティを取得します。
         *> Items プロパティは ListBox クラス内に定義された
         *> ObjectCollection クラスのオブジェクトです。
         SET WK-OBJECTCOLLECTION TO PROP-ITEMS OF WK-LISTBOX.
...
```

.NET Framework のドキュメントや Visual C#、Visual Basic などでは、 クラス名と入れ子になった型名の区切り文字として"."を使用しています。 IL では"+"を使用します。 NetCOBOL for .NET では、IL と同じ表現方法を使用します。

NCOBU00499

.NET 基本データ型のデータをオブジェクトとして扱う

.NET Framework はすべてのデータをオブジェクトとして扱います。 そのため、.NET 基本データ型は他のクラスと同様に型名をもち、メソッドやプロパティも定義されています。 一方、.NET 基本データ型の中には NetCOBOL for .NET ではオブジェクトとして表現されないものもあります。 たとえば、.NET Framework の System.Int32 型は NetCOBOL for .NET 上では用途が BINARY-LONG であるデータ項目であり、 オブジェクトとしては扱いません。 ここでは、このようなデータをオブジェクトとして扱う方法について説明します。

また、.NET Framework 上のすべての型は System.Object クラスから派生しています。 そのため、.NET Framework 上で「任意のデータ」を取り扱う場合、 System.Object オブジェクトの形でデータを扱います。 たとえば、動的に拡張可能な配列としての機能をもつ System.Collections.ArrayList オブジェクトは、 配列に格納する要素を System.Object オブジェクトの形で保持することによって、 任意の.NET Framework データを保持できるようになっています。 このため、場合によっては.NET 基本データ型のデータも System.Object オブジェクトとして扱う必要があります。 ここでは、 NetCOBOL for .NET ではオブジェクトとして表現されない.NET 基本データ型のデータと System.Object オブジェクトの変換についても説明します。

このセクションの内容

.NET 基本データ型のデータをオブジェクトとして扱う

COBOL 言語でオブジェクトとして扱われない.NET 基本データ型のデータを、オブジェクトとして扱う方法について説明します。

System.Object オブジェクトとの間で変換を行う

COBOL 言語でオブジェクトとして扱われない.NET 基本データ型のデータを、System.Object オブジェクトに変換する方法、および System.Object オブジェクトを.NET 基本データ型のデータに変換する方法について説明します。

数字定数を System.Object オブジェクトに変換する

数字定数を System.Object オブジェクトに変換する方法について説明します。

関連トピックス

.NET 基本データ型

COBOL のデータ型と.NET 基本データ型の対応について説明しています。

.NET 基本データ型のデータをオブジェクトとして扱う

COBOL 言語上ではオブジェクトとして表現されない BINARY-LONG 型などの.NET 基本データ型をオブジェクトとして扱うには、 以下の処理を行います。

1. .NET 基本データ型に対応するオブジェクト参照項目を定義する。
2. オブジェクト参照項目に.NET 基本データ型の値を設定する。
3. オブジェクト参照項目を通常のオブジェクトとして扱う。

以下の例では、BINARY-LONG 型のデータを System.Int32 型のオブジェクト参照項目に設定して、そのオブジェクト参照項目を使って ToString メソッドを呼び出しています。 ToString メソッドは System.Object クラスで定義されているため、 .NET Framework 上のすべてのオブジェクトで利用することができます。

```
...
      REPOSITORY.
         CLASS CLASS-INT32 AS "System.Int32"                      *> [1]
         CLASS CLASS-STRING AS "System.String"
...
      WORKING-STORAGE SECTION.
      01 WK       BINARY-LONG VALUE 123.
      01 WK-INT32  OBJECT REFERENCE CLASS-INT32.                  *> [1]
      01 WK-STRING OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION.
         *> BINARY-LONG 型データをオブジェクトとして扱います。
         SET WK-INT32 TO WK.                                     *> [2]

         *> メソッドを呼び出します。
         *> WK-STRING には"123"という文字列が設定されます。
         INVOKE WK-INT32 "ToString" RETURNING WK-STRING.          *> [3]
...
```

.NET 基本データ型に対応するオブジェクト参照項目を定義する

COBOL 言語上ではオブジェクトとして表現されない.NET 基本データ型であっても、 .NET Framework 上での型名があります。 その型名を用いてオブジェクト参照項目を定義します。 上の例では、 BINARY-LONG 型に対応する.NET Framework 上での型名は System.Int32 なので、 System.Int32 型のオブジェクト参照項目を定義しています。([1]) .NET 基本データ型と.NET Framework 上での型名の関係については、 「.NET 基本データ型」を参照してください。

オブジェクト参照項目に.NET 基本データ型の値を設定する

SET 文を使って、.NET 基本データ型と対応するオブジェクト参照項目の間で値を設定することができます ([2])。上の例では.NET 基本データ型からオブジェクト参照項目へ値を設定していますが、 逆にオブジェクト参照項目から.NET 基本データ型へ値を設定することもできます。

オブジェクト参照項目を通常のオブジェクトとして扱う

値が設定されたオブジェクト参照項目は、通常のオブジェクト参照項目と同じように利用できます。 上の例では、ToString メソッドを呼び出しています ([3])。

NCOBU00501

System.Object オブジェクトとの間で変換を行う

SET 文によって、 COBOL 言語上ではオブジェクトとして扱われない BINARY-LONG 型などの.NET 基本データ型を System.Object オブジェクトに変換することができます。 逆に、System.Object オブジェクトを.NET 基本データ型のデータに変換することもできます。後者の場合、 オブジェクト指定子によって.NET 基本データ型に対応する型のオブジェクト参照に変換した後、SET 文を使って.NET 基本データ型のデータに値を設定します。

```
...
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-INT32  AS "System.Int32"
...
      WORKING-STORAGE SECTION.
      01 WK     BINARY-LONG VALUE 123.
      01 WK-OBJ OBJECT REFERENCE CLASS-OBJECT.
      PROCEDURE DIVISION.
         *> BINARY-LONG 型のデータを System.Object オブジェクトに変換します。
         SET WK-OBJ TO WK.

         *> System.Object オブジェクトを BINARY-LONG 型のデータに変換します。
         *> ただし、WK-OBJ が参照するオブジェクトの実際の型は
         *> System.Int32 型でなければなりません。
         SET WK TO WK-OBJ AS CLASS-INT32.
...
```

ここで、 BINARY-LONG 型データから System.Object オブジェクトへの変換時にボックス化が、System.Object オブジェクトから BINARY-LONG 型データへの変換時にボックス化解除が発生しています。 ボックス化とボックス化解除については「ボックス化とボックス化解除」を参照してください。

System.Object オブジェクトを.NET 基本データ型に変換する場合、System.Object オブジェクトの実際の型は、.NET 基本データ型の型と同じでなければなりません。 そうでない場合は、オブジェクト指定子の処理の結果として例外が発生します。 この変換では、データの.NET Framework としての型は変更されません。 C/C++言語のキャストに慣れている方はご注意ください。

```
...
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-INT32  AS "System.Int32"
...
      WORKING-STORAGE SECTION.
      01 WK-LONG  BINARY-LONG.
      01 WK-SHORT BINARY-SHORT.
      01 WK-OBJ   OBJECT REFERENCE CLASS-OBJECT.
      PROCEDURE DIVISION.
         MOVE 30 TO WK-SHORT.
         SET WK-OBJ TO WK-SHORT.

         *> 以下のコードは実行時エラーになります。
         *> WK-OBJ は System.Int16 型のオブジェクトを参照しているため、
         *> System.Int32 型として扱うことはできません。
         SET WK-LONG TO WK-OBJ AS CLASS-INT32.
...
```

INVOKE 文、CALL 文およびメソッドの行内呼出しでは、 呼び出そうとしているメソッド（またはプログラム）の引数が System.Object 型の場合、 引数に COBOL 言語上ではオブジェクトとして扱わない.NET 基本データ型のデータを記述することができます。 この場合、NetCOBOL for .NET が、メソッドを呼び出す前に.NET 基本データ型のデータを System.Object オブジェクトに変換します。

```
...
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-INT32 AS "System.Int32"
         CLASS CLASS-ARRAYLIST AS "System.Collections.ArrayList"
...
      WORKING-STORAGE SECTION.
      01 WK          BINARY-LONG.
      01 WK-ARRAYLIST OBJECT REFERENCE CLASS-ARRAYLIST.
      01 WK-OBJECT    OBJECT REFERENCE CLASS-OBJECT.
      PROCEDURE DIVISION.
         *> ArrayList オブジェクトを作成します。
         INVOKE CLASS-ARRAYLIST "NEW" RETURNING WK-ARRAYLIST.

         *> ArrayList オブジェクトに要素を追加します。
         MOVE ZERO TO WK.
         PERFORM 3 TIMES
           *> Add メソッドの引数は System.Object 型ですが、
           *> NetCOBOL for .NET が BINARY-LONG 型のデータから System.Object オブジェクトへの
           *> 変換を自動的に行います。
           INVOKE WK-ARRAYLIST "Add" USING WK
           ADD 1 TO WK
         END-PERFORM.

         *> ArrayList オブジェクトの先頭の要素を取得します。
         *> get_Item メソッドは System.Object オブジェクトを返します。
         *> この復帰値を BINARY-LONG データに戻します。
         INVOKE WK-ARRAYLIST "get_Item" USING 0 RETURNING WK-OBJECT.
         SET WK TO WK-OBJECT AS CLASS-INT32.
...
```

NCOBU00502

数字定数を System.Object オブジェクトに変換する

数字定数を System.Object オブジェクトに変換すると、System.Int16 型、System.Int32 型、および System.Int64 型のうち、 その数字定数を格納することができる最小の型のオブジェクトに変換されます。

小数部分をもつ固定小数点定数または浮動小数点定数を System.Object オブジェクトに変換すると、System.Double 型のオブジェクトに変換されます。

```
      REPOSITORY.
         CLASS CLASS-OBJECT AS "System.Object"
...
      WORKING-STORAGE SECTION.
      01 WK-OBJ OBJECT REFERENCE CLASS-OBJECT.
      PROCEDURE DIVISION.
         *> WK-OBJ には System.Int16 型のオブジェクトが設定されます。
         SET WK-OBJ TO 123.
         *> WK-OBJ には System.Int32 型のオブジェクトが設定されます。
         SET WK-OBJ TO 1234567.
         *> WK-OBJ には System.Double 型のオブジェクトが設定されます。
         SET WK-OBJ TO 123.45.
```

System.Object 型の引数に数字定数を記述する場合は、 数字定数が意図した型に変換されるか注意が必要です。

たとえば、以下では、 動的に拡張可能な配列としての機能をもつ System.Collections.ArrayList オブジェクトに数値を格納しています。 System.Collections.ArrayList オブジェクトは任意のデータを保持できるようにするため、 要素を System.Object オブジェクトとして保持します。 このコードでは、System.Int32 型（BINARY-LONG）の値を追加したつもりでしたが、 実際に追加されるのは System.Int16 型（BINARY-SHORT）の値になります。

```
      REPOSITORY.
         CLASS CLASS-OBJECT    AS "System.Object"
         CLASS CLASS-INT32     AS "System.Int32"
         CLASS CLASS-ARRAY-LIST AS "System.Collections.ArrayList"
...
      WORKING-STORAGE SECTION.
      01 WK-LIST  OBJECT REFERENCE CLASS-ARRAY-LIST.
      01 WK-OBJ   OBJECT REFERENCE CLASS-OBJECT.
      01 WK-INT   BINARY-LONG.
      01 TEMP-INT BINARY-LONG.
      PROCEDURE DIVISION.
         *> System.Collections.ArrayList オブジェクトを作成します。
         INVOKE CLASS-ARRAY-LIST "NEW" RETURNING WK-LIST.

         *> 以下のコードは値が 123 の System.Int32 型のオブジェクト（BINARY-LONG）を
         *> 追加しているつもりですが、実際に追加されているのは、
         *> System.Int16 型のオブジェクト（BINARY-SHORT）です。
         INVOKE WK-LIST "Add" USING 123.

         *> System.Collections.ArrayList オブジェクトから System.Int32 型の
         *> 要素を取り出そうとすると失敗します。
         INVOKE WK-LIST "get_Item" USING 0 RETURNING WK-OBJ.
         SET WK-INT TO WK-OBJ AS CLASS-INT32. *> エラー！ WK-OBJ は System.Int16 型

         *> System.Int32 型オブジェクト（BINARY-LONG）を追加するには、
         *> 一度 BINARY-LONG 型のデータ項目に値を設定してから
         *> System.Object 型に変換します。
         MOVE 123 TO TEMP-INT.
         INVOKE WK-LIST "Add" USING TEMP-INT.
```

アセンブリを作成する

NetCOBOL for .NET を利用して.NET プログラムを作成すれば、 通常それはアセンブリになります。

ここでは、アセンブリ作成時に特に注意が場合について説明します。

このセクションの内容

NCOBU00943

厳密な名前付きアセンブリを作成する

アセンブリを作成する際に、キーペアを使って厳密な名前付きアセンブリとして署名を行うと、 そのアセンブリは厳密な名前付きアセンブリになります。

厳密な名前つきアセンブリを作成するための一般的な手順については、 「厳密な名前付きアセンブリの作成と使用」を参照してください。 ここでは、NetCOBOL for .NET を利用して厳密な名前つきアセンブリを作成する場合の注意点について説明します。

キーの指定法

厳密な名前付きアセンブリとして署名するためのキーを指定する方法としては、「方法 : 厳密な名前でアセンブリに署名する」 に説明されている以外に、以下の方法があります。

- Visual Studio 2017 を使って開発している場合は、プロジェクトデザイナーの「署名」ページからキーを指定することができます。
- コンパイラオプション/keyfile または/keyname を使ってキーを指定することができます。

厳密な名前の各要素の指定方法

厳密な名前の各要素は、以下の方法で指定します。

厳密な名前の要素	指定方法
単純テキスト名	アセンブリの出力ファイル名のベース名（拡張子を除いた名前）が単純テキスト名となります。 出力ファイル名は、コンパイラオプション/out で指定します。Visual Studio を使って開発している場合は、プロジェクトデザイナーの「アプリケーション」ページの「アセンブリ名」でアセンブリ名を直接指定することができます。
バージョン番号	以下のいずれかの方法で指定します。 ・ アセンブリに対して System.Reflection.AssemblyVersionAttribute カスタム属性を指定します。Visual Studio を使用して開発している場合は、プロジェクトデザイナーの「アプリケーション」ページから表示する「アセンブリ情報」ダイアログの「アセンブリバージョン」でこのカスタム属性を設定することができます。 ・ コンパイラオプション/version で指定します。
カルチャ情報	アセンブリに対して System.Reflection.AssemblyCultureAttribute カスタム属性を指定します。
公開キー	署名に用いたキーによって設定されます。
プラットフォーム情報	コンパイラオプション/platform の値のよって設定されます。 Visual Studio を使って開発している場合は、プロジェクトデザイナーの「ビルド」ページの「プラットフォームターゲット」で指定することができます。

型を定義する

NetCOBOL for .NET で、一般的な型を定義する場合、以下を参照してください。

型の種類	説明
クラス定義	クラス定義
	COBOL 文法書の「11.4.1.1 COBOL 翻訳集団」のクラス定義
	COBOL 文法書の「11.5.1.2 クラス名段落(CLASS-ID)」
インタフェース定義	インタフェース定義
	COBOL 文法書の「11.4.1.1 COBOL 翻訳集団」のインタフェース定義
	COBOL 文法書の「11.5.1.7 インタフェース名段落(INTERFACE-ID)」
ENUM 定義	列挙型定義
	COBOL 文法書の「11.4.1.1 COBOL 翻訳集団」の ENUM 定義
	COBOL 文法書の「11.5.1.8 ENUM 名段落(ENUM-ID)」
デリゲート定義	デリゲート定義
	COBOL 文法書の「11.4.1.1 COBOL 翻訳集団」のデリゲート定義
	COBOL 文法書の「11.5.1.6 デリゲート名段落(DELEGATE-ID)」

ここでは、特に注意が必要な型の定義について説明します。

このセクションの内容

カスタム属性を使う

カスタム属性について説明します。

NCOBU00504

カスタム属性を使う

カスタム属性を用いることによって、 .NET Framework のメタデータに独自の情報を含めることができます。

NetCOBOL for .NET を使って.NET Framework 向けアプリケーションを開発すると、 NetCOBOL for .NET コンパイラは.NET Framework の型情報（メタデータ）を生成します。 このメタデータには、 あらかじめ.NET Framework によって決められた情報だけではなく、 ツールやユーザが定義した情報を含めることができます。 メタデータに追加する独自の情報は、 カスタム属性という形式で格納されます。 カスタム属性そのものも.NET Framework 上のオブジェクトとして表現されています。 そのため、 メタデータにカスタム属性オブジェクトが格納されているように見えます。

カスタム属性は、 .NET Framework の型情報を取得する機能を通して利用できます。 詳細については、 .NET Framework のドキュメントの「.NET Framework のリフレクション」を参照してください。 しかし、 アプリケーションが直接カスタム属性を読み取ることはあまりありません。 多くの場合、 .NET Framework が提供するサービスや.NET Framework 対応のツールが、 メタデータからカスタム属性を読み取って利用します。 たとえば、 .NET Framework のリモーティングサービスでは、 オブジェクトをリモート環境へ転送する際に、 そのオブジェクトのクラスに System.SerializableAttribute カスタム属性が指定されているかどうか調べ、 転送可能かどうかチェックします。 また、ASP.NET は、 XML Web サービスとして設計されたクラスのうち、 System.Web.Services.WebMethodAttribute カスタム属性が指定されたメソッドのみを Web サービス上で公開します。 そのため、 各アプリケーションでカスタム属性を使う場合、 ツールの要求に応じてカスタム属性を指定することがほとんどです。

このセクションの内容

カスタム属性を指定する

ここでは、カスタム属性の指定方法について説明します。 以下の例は、Windows 版 NetCOBOL で開発したプログラムを呼び出すためのプログラム原型定義です。 各カスタム属性の役割については、他の環境および言語との相互運用を参照してください。 ここでは、カスタム属性を指定するための構文について解説します。

```
PROGRAM-ID. UNMANAGED-METHOD PROTOTYPE
          CUSTOM-ATTRIBUTE CA-RUNTIME-ENCODING CA-DLL-IMPORT. *> [1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    *> パラメタをもたないカスタム属性の例です。                        *> [2]
    CUSTOM-ATTRIBUTE CA-SUPPRESS-CONVERSION
      CLASS CLASS-SUPPRESS-CONVERSION

    *> パラメタをもつカスタム属性の例です。                            *> [3]
    CUSTOM-ATTRIBUTE CA-RUNTIME-ENCODING
      CLASS CLASS-RUNTIME-ENCODING
      USING PROP-ACP OF ENUM-RUNTIME-ENCODING-MODE

    *> 名前つきパラメタをもつカスタム属性の例です。                    *> [4]
    CUSTOM-ATTRIBUTE CA-DLL-IMPORT
      CLASS CLASS-DLL-IMPORT
      USING  "Sample.dll"
      PROPERTY PROP-ENTRYPOINT IS "Method1"
      PROPERTY PROP-EXACTSPELLING IS B"1"
    .
REPOSITORY.
    CLASS CLASS-DLL-IMPORT
      AS "System.Runtime.InteropServices.DllImportAttribute"
    CLASS CLASS-SUPPRESS-CONVERSION
      AS "Fujitsu.COBOL.InteropServices.Win32.SuppressEncodingConversionAttribute"
    CLASS CLASS-RUNTIME-ENCODING
      AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingAttribute"
    ENUM ENUM-RUNTIME-ENCODING-MODE
      AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode"
    PROPERTY PROP-ACP AS "ACP"
    PROPERTY PROP-ENTRYPOINT AS "EntryPoint"
    PROPERTY PROP-EXACTSPELLING AS "ExactSpelling"
    .
DATA DIVISION.
LINKAGE SECTION.
01 PARAM-1 PIC X(32) CUSTOM-ATTRIBUTE CA-SUPPRESS-CONVERSION.       *> [1]
01 PARAM-2 PIC N(8).
PROCEDURE DIVISION USING PARAM-1 PARAM-2.
END PROGRAM UNMANAGED-METHOD.
```

カスタム属性の指定

カスタム属性は、指定したい対象に対して CUSTOM-ATTRIBUTE 句を記述することによって指定します。 ここでは、以下のカスタム属性を指定しています。([1])

- プログラム原型定義に対して、CA-RUNTIME-ENCODING と CA-DLL-IMPORT
- 引数 PARAM-1 に対して、CA-SUPPRESS-CONVERSION

ここで、CA-RUNTIME-ENCODING、CA-DLL-IMPORT、および、CA-SUPPRESS-CONVERSION は特殊名段落で定義されたカスタム属性の名前です。 どの対象にカスタム属性を指定することができるかは、COBOL 文法書の「11.5.1.9 CUSTOM-ATTRIBUTE 句(見出し部)」および「11.7.3.2 CUSTOM-ATTRIBUTE 句(データ部)」を参照してください。

パラメタをもたないカスタム属性の定義

カスタム属性は、特殊名段落で CUSTOM-ATTRIBUTE 句を用いて定義します。 パラメタをもたないカスタム属性を指定するには、 予約語 CUSTOM-ATTRIBUTE に続いて、カスタム属性名、予約語 CLASS、カスタム属性のクラス名を記述します（[2]）。 カスタム属性自体もオブジェクトとして表現されるため、 カスタム属性のク

ラスを指定する必要があります。 カスタム属性のクラス名はリポジトリ段落で宣言されていなければなりません。

.NET Framework の慣習として、 カスタム属性を表すクラスの名前の末尾は"Attribute"がつきます。

パラメタをもつカスタム属性の定義

パラメタをもつカスタム属性を指定するには、 "パラメタをもたないカスタム属性の定義"の記述に続いて、 予約語 USING、パラメタ（複数可）を記述します。([3]) パラメタには以下を指定することができます。

- 数字定数
- 英数字定数
- 日本語定数
- 長さ 1 のブール定数
- 列挙体の列挙値
- 組込み関数の ENUM-OR、ENUM-AND、ENUM-NOT
- TYPE OF 特殊レジスタ

パラメタに、データ名やここに挙げた以外の組込み関数を記述することはできません。

カスタム属性のパラメタは、 カスタム属性のクラスのコンストラクタに対応しています。 .NET Framework の型情報取得機能を使ってカスタム属性を読み取ると、 対応するコンストラクタによって生成されたカスタム属性オブジェクトが取得されます。

名前つきパラメタをもつカスタム属性の定義

名前つきパラメタをもつカスタム属性を指定するには、 "パラメタをもつカスタム属性の定義"の記述に続いて、予約語 PROPERTY、プロパティ名、予約語 IS、パラメタのセットを各名前つきパラメタに対して記述します。([4]) 名前つきパラメタとして指定できる値は、 名前なしパラメタとして指定できるものと同じです。 プロパティ名はリポジトリ段落で宣言されていなければなりません。

カスタム属性の名前つきパラメタは、 カスタム属性の公開プロパティや公開フィールドに対応しています。 .NET Framework の型情報取得機能を使ってカスタム属性を読み取ると、 公開プロパティや公開フィールドに指定したパラメタが設定されたカスタム属性オブジェクトが取得されます。

特殊名段落で CUSTOM-ATTRIBUTE 句を定義する場合の書き方の詳細については、COBOL 文法書の「11.6.1.2 CUSTOM-ATTRIBUTE 句(環境部)」を参照してください。

COBOL の要素に指定したカスタム属性は、これらがメタデータから読み出されるときに有効になります。 そうでない場合は有効にならないので注意してください。 たとえば、可変個数の引数を示す ParamArrayAttribute を持たせたメソッドを COBOL で定義し、同ソース内あるいは同時に翻訳される COBOL ソース内からこれを呼び出そうとしても、ParamArrayAttribute が有効にならないため翻訳エラーになります。

カスタム属性を読み取る

メタデータ中のカスタム属性は、 .NET Framework の型情報取得機能を使って読み取ることができます。 詳細については、.NET Framework のドキュメントの「.NET Framework のリフレクション」を参照してください。

NCOBU00507

カスタム属性を定義する

独自のカスタム属性クラスは、 一般のクラスを定義するのと同様に定義することができます。 ただし、以下の点に注意してください。

- カスタム属性クラスは System.Attribute クラスを継承しなければなりません。
- カスタム属性クラスには、指定対象を示すために System.AttributeUsageAttribute カスタム属性を指定しなければなりません。

以下は、 クラスとインタフェースを対象としたカスタム属性クラスを定義する例です。

```
CLASS-ID. MY-ATTRIBUTE AS "MyAttribute" INHERITS CLASS-ATTRIBUTE
         CUSTOM-ATTRIBUTE IS CA-ATTRIBUTE-USAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    *> カスタム属性クラスには System.AttributeUsageAttribute カスタム属性を
    *> 指定する必要があります。
    CUSTOM-ATTRIBUTE CA-ATTRIBUTE-USAGE
      CLASS CLASS-ATTRIBUTE-USAGE
      USING FUNCTION ENUM-OR (PROP-CLASS OF ENUM-ATTRIBUTE-TARGETS,
            PROP-INTERFACE OF ENUM-ATTRIBUTE-TARGETS)
    .
REPOSITORY.
    CLASS CLASS-ATTRIBUTE AS "System.Attribute"
    CLASS CLASS-ATTRIBUTE-USAGE AS "System.AttributeUsageAttribute"
    ENUM ENUM-ATTRIBUTE-TARGETS AS "System.AttributeTargets"
    PROPERTY PROP-CLASS AS "Class"
    PROPERTY PROP-INTERFACE AS "Interface"
    .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 MYCOUNT BINARY-LONG PROPERTY.
PROCEDURE DIVISION.

METHOD-ID. NEW.
DATA DIVISION.
LINKAGE SECTION.
01 PARAM-1 BINARY-LONG.
PROCEDURE DIVISION USING BY VALUE PARAM-1.
    MOVE PARAM-1 TO MYCOUNT OF SELF.
END METHOD NEW.

END OBJECT.

END CLASS MY-ATTRIBUTE.
```

このバージョンの NetCOBOL for .NET では、 指定するカスタム属性は外部モジュールで定義されていなければなりません。 カスタム属性クラスの定義とそのカスタム属性を指定するコードを同一モジュールへ翻訳することはできません。

.NET リモート処理を使う

.NET Framework には、 異なるアプリケーションドメイン間（異なるプロセス間やマシン間を含みます）でオブジェクトのメソッド呼出しができる.NET リモート処理の仕組みが標準で含まれています。 .NET リモート処理では、IDL を記述しなくても、少しの設定を行うだけで多くのメソッドをリモート呼出しできます。 これは.NET プラットフォーム上のアプリケーションが共通のメタデータで記述されていることを利用し、 .NET Framework がメタデータを参照して必要な処理を自動的に行うためです。

NetCOBOL for .NET で開発したクラスも、 .NET リモート処理で利用することができます。 NetCOBOL for .NET で開発したクラスはそのまま.NET Framework 上のクラスになるので、 .NET Framework のドキュメントの「.NET リモート処理」に記述されている方法によって、 .NET リモート処理ができるようになります。

ただし、 リモート呼出しを行うメソッドの引数に COBOL 独自データ型を指定する場合など、 NetCOBOL for .NET で.NET リモート処理を利用する場合の注意点がいくつかあります。 ここでは、これらの注意点について説明します。

.NET リモート処理は、オブジェクトやオブジェクトメソッドを対象にしています。 クラスの静的メソッドやプログラムは、.NET リモート処理で呼び出すことはできません。

このセクションの内容

NCOBU00509

リモート呼出しで COBOL 独自データ型の引数を使用する

ここでは、リモート呼出しができるメソッドに対して、そのメソッドの引数に COBOL 独自データ型を使用する場合の注意点について説明します。

このセクションの内容

COBOL 独自データ型の引数を使用する

.NET リモート処理で、リモートからメソッド呼出しができるオブジェクトを開発する場合、 そのメソッドの引数や復帰項目に COBOL 独自データ型を指定することができます。

COBOL 独自データ型の内容は、呼び出す側と呼び出される側の環境を越えて複写されます。 この結果、 実行環境に依存するデータの内容は相手の環境では無効な値になります。 このようなデータはリモート呼出しの引数に含めないでください。 実行環境に依存するデータには、以下があります。

- ポインタデータ項目

このバージョンの NetCOBOL for .NET では、リモート呼出しができるメソッドに、参照渡しの COBOL 独自データ型の引数を指定した場合、 メソッドで変更された引数の内容は、呼び出す側に反映されません。 呼び出される側から呼び出す側へ値を返すには、復帰項目を利用します。

NCOBU00511

効率的な COBOL 独自データ型の引数をデザインする

リモート呼出しができるメソッドの引数や復帰項目に COBOL 独自データ型を使用すると、 COBOL 独自データ型の内容が呼び出す側と呼び出される側の環境を越えて複写されます。 引数として巨大な項目を定義した場合、大量のデータの複写が発生し、 リモート処理のオーバーヘッドが大きくなることがあります。

ここでは、 データの複写を最小限に抑えるような引数をデザインする方法について説明します。

```
      CLASS-ID. SAMPLE-1 INHERITS CLASS-MARSHALBYREFOBJECT.
...
      REPOSITORY.
         CLASS CLASS-MARSHALBYREFOBJECT AS "System.MarshalByRefObject"
...
      METHOD-ID. REMOTABLE-METHOD.
      DATA DIVISION.
      LINKAGE SECTION.
      01 PARAM.
        02 ERROR-CODE PIC 9.
        02 USER-ID    PIC 999.
        02 USER-NAME  PIC N(16).
      PROCEDURE DIVISION USING PARAM.
         *> 結果コードを初期化します
         MOVE ZERO TO ERROR-CODE.

         *> USER-ID から名前を求めます
         EVALUATE USER-ID
           WHEN 1
             MOVE N"鈴木" TO USER-NAME
           WHEN 2
             MOVE N"佐藤" TO USER-NAME
           WHEN OTHER
             MOVE 1 TO ERROR-CODE
             MOVE SPACE TO USER-NAME
         END-EVALUATE
      END METHOD REMOTABLE-METHOD.
...
```

メソッドのリモート呼出しをサポートするクラスは、System.MarshalByRefObject を継承しなければなりません。

このバージョンの NetCOBOL for .NET では、 上の例は意図した動作をしません。 リモート呼出しができるメソッドに、参照渡しの COBOL 独自データ型の引数を指定した場合、 メソッドで変更された引数の内容は呼出し側に反映されません。 ここでは、概念を説明するためにこの例を用いています。

上の例では、一つの集団項目を引数とするメソッドを定義しています。 このメソッドをリモート呼出しすると、利用状況から見て無駄なデータの複写が発生します。 この例では、説明のために小さな集団項目を使っているため、オーバーヘッドは問題にはならないでしょうが、 引数のサイズによってはオーバーヘッドが無視できなくなるでしょう。

各項目の利用状況を見てみると、 以下のことがわかります。

- USER-ID は参照しているだけで、値を設定しない。したがって、USER-ID を呼び出される環境から呼

び出す環境に書き戻す必要はない。

- ERROR-CODE と USER-NAME には値を設定しているが、メソッド呼出し時に渡された値を参照していない。 したがって、呼び出す環境から呼び出される環境に ERROR-CODE と USER-NAME を複写する必要はない。

したがって、 USER-ID を値渡しの引数に、 ERROR-CODE と USER-NAME を復帰項目にすれば、 リモート呼出し時のデータの複写が最小限に抑えられます。

```
     METHOD-ID. REMOTABLE-METHOD.
     DATA DIVISION.
     LINKAGE SECTION.
     01 USER-ID     PIC 999.
     01 RET-VAL.
       02 ERROR-CODE PIC 9.
       02 USER-NAME  PIC N(16).
     PROCEDURE DIVISION USING BY VALUE USER-ID RETURNING RET-VAL.
        *> 結果コードを初期化します
        MOVE ZERO TO ERROR-CODE.

        *> USER-ID から名前を求めます
        EVALUATE USER-ID
          WHEN 1
            MOVE N"鈴木" TO USER-NAME
          WHEN 2
            MOVE N"佐藤" TO USER-NAME
          WHEN OTHER
            MOVE 1 TO ERROR-CODE
            MOVE SPACE TO USER-NAME
        END-EVALUATE
     END METHOD REMOTABLE-METHOD.
...
```

このバージョンの NetCOBOL for .NET では、メソッドに COBOL 独自データ型の引数を指定する場合、数字項目のときだけ値渡しにすることができます。

NCOBU00512

NetCOBOL for .NET をインストールしていない環境からリモート呼出しを行う

ここでは、.NET リモート処理プロキシを用いて、 NetCOBOL for .NET をインストールしていない環境からリモート呼出しを行う方法について説明します。

このセクションの内容

.NET リモート処理プロキシ作成が必要となる場面

.NET リモート処理プロキシとは何か、また、.NET リモート処理プロキシが必要になる場面について説明します。

.NET リモート処理プロキシを作成する場合の注意点

.NET リモート処理プロキシを使用する場合の注意点について説明します。

関連トピックス

.NET リモート処理プロキシ作成ツール（genrp.exe）

.NET リモート処理プロキシ作成ツールについて説明しています。

.NET リモート処理プロキシ作成が必要となる場面

NetCOBOL for .NET を使ってリモートからメソッド呼出しができるクラスを開発しているとします。 通常、開発したアセンブリは、 リモート呼出しされる環境だけでなく、 リモート呼出しする環境にも配置します。 これは、クラスのコードが実行されるのはリモート呼出しされる環境ですが、 リモート呼出しする環境でも.NET Framework が.NET リモート処理を行うためにクラスのメタデータ（型情報）を参照する必要があるからです。

NetCOBOL for .NET で開発したアセンブリを配置するには、 その環境に NetCOBOL for .NET のランタイムがインストールされていなければなりません。 リモート呼出しする環境にすでに NetCOBOL for .NET がインストールされていれば問題はありません。 しかし、そうでない場合、 この環境では実際に COBOL による処理が行われないにもかかわらず NetCOBOL for .NET をインストールしなければならないことになります。 このような場合に、 .NET リモート処理プロキシを利用すると、 NetCOBOL for .NET のランタイムがインストールされていない環境からもリモート呼出しができるようになります。

.NET リモート処理プロキシは、 アセンブリから処理コードを取り除き、.NET リモート処理に必要な最小限の型情報だけを取り出したアセンブリです。 開発したアセンブリに対応するプロキシアセンブリをリモート呼出しする環境に配置することによって、 NetCOBOL for .NET ランタイムがインストールされていない環境からも、開発したクラスのメソッドをリモート呼出しができるようになります。

NetCOBOL for .NET に付属する.NET リモート処理プロキシ作成ツール（genrp.exe）を使うと、 アセンブリから簡単に.NET リモート処理プロキシを作成することができます。

NCOBU00514

.NET リモート処理プロキシを作成する場合の注意点

.NET リモート処理プロキシの名前

.NET リモート処理プロキシのファイル名は本体アセンブリのファイル名と同じでなければなりません。 また、本体アセンブリが厳密な名前を持つアセンブリである場合、 .NET リモート処理プロキシも同じ厳密名を持つ必要があります。 つまり、本体アセンブリと.NET リモート処理プロキシは同じ単純テキスト名、バージョン番号、カルチャ情報、公開キーを持つ必要があります。 同じ厳密名をつけるためには、本体アセンブリと.NET リモート処理プロキシを同じ秘密キーを使ってビルドする必要があります。 したがって、厳密な名前を持つアセンブリの.NET リモート処理プロキシを作成できるのは、 本体アセンブリを開発したベンダーだけです。

厳密な名前付きのアセンブリの詳細は.NET Framework のドキュメント「厳密な名前付きアセンブリの作成と使用」を参照してください。

.NET リモート処理プロキシが NetCOBOL for .NET を参照しないようにする

せっかく.NET リモート処理プロキシを作成して必要最小限の型情報を抜き出したとしても、 その型情報の中に NetCOBOL for .NET ランタイムを必要とする型が含まれていれば、 リモート呼出しする環境とリモート呼出しされる環境の両方に NetCOBOL for .NET ランタイムが必要になってしまいます。

たとえば、COBOL 独自データ型は NetCOBOL for .NET ランタイムを必要とします。 このため、リモート呼出しができるメソッドに COBOL 独自データ型の引数や復帰項目があると、 .NET リモート処理プロキシは NetCOBOL for .NET ランタイムを参照します。

.NET リモート処理プロキシを使って NetCOBOL for .NET ランタイムがインストールされていない環境からリモート呼出しを行いたい場合、 .NET リモート処理プロキシが NetCOBOL for .NET ランタイムを参照しないようにします。 そのためには、以下の点に注意します。

- リモート呼出しするメソッドの引数や復帰項目に以下を含まないようにします。
  - COBOL 独自データ型のデータ項目
  - NetCOBOL for .NET で開発した型のオブジェクト参照項目
- リモート呼出しをサポートする以下に対して、NetCOBOL for .NET で開発したカスタム属性を設定しないようにします。
  - クラス
  - プロパティ
  - メソッドおよびその引数と復帰項目

.NET リモート処理プロキシの実装

.NET リモート処理プロキシには実装コードが含まれていません。 そのため、.NET リモート処理の設定を行わずに、 .NET リモート処理プロキシ内のクラスのオブジェクトをローカル環境上で利用すると、 例外(System.NotImplementedException)が発生します。

値渡しでマーシャリングされるオブジェクトを作成する

ここでは、値渡しでマーシャリングされるリモート処理が可能なオブジェクトを、NetCOBOL for .NET で開発する場合の注意点について説明します。

値渡しでマーシャリングされるリモート処理が可能なクラスを作成するための最も簡単な方法は、 クラスに System.SerializableAttribute カスタム属性を指定することです。 これによって、.NET Framework が既定のマーシャリング方法によってオブジェクトを自動的に相手先の環境へ複製します。

ただし、以下の NetCOBOL for .NET の機能を利用しているクラスは、 内部的な制御領域などがマーシャリングをサポートしていないため、 .NET Framework の既定の方法でマーシャリングすることはできません。

- ファイル機能(印刷ファイルを含む)
- 整列併合機能
- SQL

以下の例は、.NET Framework の既定の方法によって値渡しでマーシャリングされるリモート処理が可能なクラスの例です。 この例では、PROP1、PROP2 という 2 つのデータを保持するオブジェクトを、値渡しでマーシャリングできるようにします。

```
*> 値渡しでマーシャリングできるクラスを定義します。
CLASS-ID. SAMPLE-1 CUSTOM-ATTRIBUTE CA-SERIALIZABLE.          *> [1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-SERIALIZABLE
      CLASS CLASS-SERIALIZABLE-ATTRIBUTE                      *> [1]
    .
REPOSITORY.
    CLASS CLASS-BOOLEAN  AS "System.Boolean"
    CLASS CLASS-SERIALIZABLE-ATTRIBUTE
      AS "System.SerializableAttribute"                       *> [1]
    .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
*> オブジェクトが保持するデータです。
01 PROP1 BINARY-LONG PROPERTY.
01 PROP2 OBJECT REFERENCE CLASS-BOOLEAN PROPERTY.
PROCEDURE DIVISION.
END OBJECT.
END CLASS SAMPLE-1.
```

System.SerializableAttribute カスタム属性の設定

このクラスが.NET リモート処理ができることを示すため、 クラスに対して System.SerializableAttribute カスタム属性を設定します（[1]）。

NetCOBOL for .NET を使って、値渡しでマーシャリングされるクラスを開発し、それを.NET リモート処理で使用した場合、リモート処理で呼び出す環境と呼び出される環境の両方に NetCOBOL for .NET ランタイムが必要になります。

一方、.NET Framework の既定の方法を用いずに、System.Runtime.Serialization.ISerializable を実装してシリアライズ処理をカスタマイズすることができます。 この方法の詳細については、.NET Framework のドキュメントの「カスタムのシリアル化」を参照してください。

以下の例は、シリアライズ処理をカスタマイズしているクラスの例です。

```
*> 値渡しでマーシャリング可能なクラスを定義します。
CLASS-ID. SAMPLE-1 CUSTOM-ATTRIBUTE CA-SERIALIZABLE.          *> [1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-SERIALIZABLE
      CLASS CLASS-SERIALIZABLE-ATTRIBUTE                      *> [1]
    .
REPOSITORY.
    CLASS CLASS-SERIALIZABLE-ATTRIBUTE
      AS "System.SerializableAttribute"                       *> [1]
    CLASS CLASS-BOOLEAN AS "System.Boolean"
    INTERFACE INT-SERIALIZABLE
      AS "System.Runtime.Serialization.ISerializable"
    CLASS CLASS-SERIALIZATION-INFO
      AS "System.Runtime.Serialization.SerializationInfo"
    CLASS CLASS-STREAMING-CONTEXT
      AS "System.Runtime.Serialization.StreamingContext"
    CLASS CLASS-OBJECT AS "System.Object"
    CLASS CLASS-TYPE AS "System.Type"
    CLASS ARRAY-BYTE AS "System.Byte[]"
    .

OBJECT. IMPLEMENTS INT-SERIALIZABLE.                          *> [2]
DATA DIVISION.
WORKING-STORAGE SECTION.
*> オブジェクトが保持するデータです。
01 PROP1 BINARY-LONG PROPERTY.
01 PROP2 OBJECT REFERENCE CLASS-BOOLEAN PROPERTY.
01 DATA1 PIC X(16).
PROCEDURE DIVISION.

*> ISerializable インタフェースのメソッドです。
*> このメソッドはマーシャリングのシリアル化の時に
*> 呼び出されます。
*> SerializationInfo オブジェクトへ
*> オブジェクトの内容を保存します。
METHOD-ID. GET-OBJECT-DATA AS "GetObjectData".                *> [2]
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK OBJECT REFERENCE ARRAY-BYTE.
LINKAGE SECTION.
01 PARAM-INFO    OBJECT REFERENCE CLASS-SERIALIZATION-INFO.
01 PARAM-CONTEXT OBJECT REFERENCE CLASS-STREAMING-CONTEXT.
PROCEDURE DIVISION USING BY VALUE PARAM-INFO PARAM-CONTEXT.
    *> オブジェクトのデータを保存します。
    INVOKE PARAM-INFO "AddValue" USING N"PROP1" PROP1.
    INVOKE PARAM-INFO "AddValue" USING N"PROP2" PROP2.

    *> COBOL 独自データ型は、バイト配列として保存します。
    SET WK TO DATA1.
    INVOKE PARAM-INFO "AddValue" USING N"DATA1" WK.
END METHOD GET-OBJECT-DATA.


*> 通常のコンストラクタです。
METHOD-ID. NEW.
DATA DIVISION.
LINKAGE SECTION.
01 PARAM-INIT-VAL BINARY-LONG.
PROCEDURE DIVISION USING BY VALUE PARAM-INIT-VAL.
    MOVE PARAM-INIT-VAL TO PROP1.
END METHOD NEW.

*> マーシャリングの逆シリアル化の時に
*> 呼び出されるコンストラクタです。
*> SerializationInfo オブジェクトの内容から
*> オブジェクトを復元します。
METHOD-ID. NEW.                                               *> [3]
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-OBJECT OBJECT REFERENCE CLASS-OBJECT.
01 WK-TYPE   OBJECT REFERENCE CLASS-TYPE.
LINKAGE SECTION.
01 PARAM-INFO    OBJECT REFERENCE CLASS-SERIALIZATION-INFO.
```

```
       01 PARAM-CONTEXT OBJECT REFERENCE CLASS-STREAMING-CONTEXT.
       PROCEDURE DIVISION USING BY VALUE PARAM-INFO PARAM-CONTEXT.
          *> オブジェクトのデータを再生します。
          INVOKE PARAM-INFO "GetInt32"   USING N"PROP1" RETURNING PROP1.
          INVOKE PARAM-INFO "GetBoolean" USING N"PROP2" RETURNING PROP2.

          *> COBOL 独自データ型は、バイト配列として保存しています。
          SET WK-TYPE TO TYPE OF ARRAY-BYTE.
          SET WK-OBJECT TO PARAM-INFO::"GetValue" (N"DATA1", WK-TYPE).
          SET DATA1 TO WK-OBJECT AS ARRAY-BYTE.
       END METHOD NEW.

       END OBJECT.
       END CLASS SAMPLE-1.
```

System.Runtime.Serialization.ISerializable インタフェースの実装

マーシャリングの際のシリアル化をカスタマイズするため、System.Runtime.Serialization.ISerializable インタフェースを実装し、 このインタフェースの GetObjectData メソッドを実装します（[2]）。 GetObjectData はオブジェクトのデータを保存するときに呼び出されます。 ここでは、PROP1, PROP2, DATA1 の値を SerializationInfo オブジェクトに保存しています。

オブジェクトデータを保存または再生するために用いる System.Runtime.Serialization.SerializationInfo オブジェクトは、 .NET Framework の型の保存または再生しかサポートしていません。 そのため、COBOL 独自データ型はバイト配列に変換して保存しています。

逆シリアル化のためのコンストラクタの定義

マーシャリングの際の逆シリアル化に対応するため、 専用のコンストラクタを定義します（[3]）。 このコンストラクタは GetObjectData と同じ引数を取ります。 このコンストラクタは、 GetObjectData で保存したデータからオブジェクトを再生するときに呼び出されます。 ここでは、PROP1, PROP2, DATA1 の値を SerializationInfo オブジェクトから再生しています。

オブジェクトデータを保存または再生するために用いる System.Runtime.Serialization.SerializationInfo オブジェクトは、 .NET Framework の型の保存または再生しかサポートしていません。 そのため、COBOL 独自データ型はバイト配列に変換して保存していました。 ここでは、バイト配列を再生し、それを COBOL 独自データ型へ変換しています。

- NetCOBOL for .NET を使って値渡しでマーシャリングされるクラスを開発し、それを.NET リモート処理で使用した場合、 リモート処理で呼び出す環境と呼び出される環境の両方に NetCOBOL for .NET ランタイムが必要になります。
- 「シリアル化のカスタマイズ」を行ったクラスを.NET リモート処理で使用する場合、 自動逆シリアル化のレベルとして Full が必要になります。 自動逆シリアル化のレベルの詳細については、 .NET Framework のドキュメントの「.NET Framework リモート処理での自動逆シリアル化」を参照してください。

NCOBU00516

Web アプリケーションの開発

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

NetCOBOL for .NET で Web アプリケーションを開発する場合、ASP.NET 技術を利用して Web フォームページを作成します。Web フォームページは、コードと Web コントロールとの組み合わせで構成されるページです。

NetCOBOL for .NET では、ページのコード部分を記述するプログラム言語として COBOL をサポートしています。

NetCOBOL for .NET で作成した ASP.NET Web アプリケーションは、以下の動作をします。

1. ページにアクセスするたびに、コードを呼び出し、
2. コードの実行によって出力された html を要求されたクライアントに返します。

ここで、コードはアクセスのたびに実行できる、コンパイルされたコードを指しています。

ここで説明する内容は、局所的な情報を寄せ集めたものになっています。必要に応じて適宜参照してください。

なお、NetCOBOL for .NET での ASP.NET Web アプリケーションの概要や開発の全体像を知りたい場合は、チュートリアルの ASP.NET Web アプリケーションの開発を参照してください。

また、ASP.NET Web アプリケーションのアーキテクチャの詳細については、Visual Studio のドキュメントの、「ASP.NET Web ページの概要」および「ASP.NET Web ページのコード モデル」を参照してください。

このセクションの内容

COBOL コードの配置

Web アプリケーションの開発では、ASP.NET ページ(Web フォームページ)を作成します。ASP.NET ページは、コントロールの配置(html)と手続きコードから構成されます。手続きコードは、.cobx ファイルに記述します。このトピックでは COBOL コードの配置方法について説明します。

ASP.NET AJAX 機能を利用する

.NET Framework 3.5 以降では、ASP.NET AJAX 機能が標準化され、AJAX 対応の Web アプリケーションを簡単に作成することができます。このトピックでは、ASP.NET AJAX の概要について説明します。

関連トピックス

Visual Studio のドキュメントの ASP.NET Web ページの概要

Web フォームの目的、およびフォーム内のコードの機能をわかりやすく紹介します。

COBOL コードの配置

ASP.NET Web アプリケーションを作成する場合、その手続きを記述する方法には、以下の方法があります。

- 分離コードファイルに COBOL コードを配置する

ASP.NET ページの言語として NetCOBOL for .NET を利用する場合、@ Page ディレクティブの Language 属性(ページのスクリプト言語)に"Fujitsu.COBOL"を指定する必要があります。

@ Page ディレクティブの詳細については、.NET Framework のドキュメントの@ Page を参照してください。

分離コードファイルに COBOL コードを配置する

.aspx ファイルとは別のファイル(.aspx.cobx ファイル)に COBOL コードを配置したい場合、@ Page ディレクティブの CodeFile 属性にそのファイル名を指定することで、分離コードファイルに COBOL コードを配置できます。

また、分離コードファイルに COBOL コードを記述する場合、以下の点に注意して記述します。

- Page クラスを継承するクラスを定義します。
- クラス名はその外部名が、@ Page ディレクティブの CodeFile 属性に指定したクラス名と同じ名前になるように指定します。

たとえば、MyPage.aspx.cobx ファイルに MyPageCOBOLCode クラスのコードを配置したい場合、.aspx ファイルの@ Page ディレクティブの指定は以下のようになります。

```
【.aspx ファイル】
<% Page Language="Fujitsu.COBOL" Inherits="MyNameSpace.MyPageCOBOLCode"
       Codebehind="MyPage.aspx.cobx" … %>
```

```
【MyPage.aspx.cobx】
     CLASS-ID. MyPageCOBOLCode AS "MyNameSpace.MyPageCOBOLCode" INHERITS PageClass IS PARTIAL.
     ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
     SPECIAL-NAMES.
     REPOSITORY.
         CLASS PageClass AS "System.Web.UI.Page"
           :
```

関連トピックス

Visual Studio のドキュメントの ASP.NET Web ページのコード モデル

注意：このトピックを参照する場合は、.NET Framework のドキュメントを参考にしてください。 分離コードモデル、およびスクリプトブロックモデルで、コードがどうやって Web フォームに統合されるのかを説明します。

.NET Framework のドキュメントの ASP.NET ページの構文

Web フォームページ(.aspx ファイル)を構成する構文について、説明しているトピックにリンクしています。 ディレクティブ構文やコード表示ブロックなどの構文を調査する場合に、役立ちます。

NCOBU01049

ASP.NET AJAX 機能を利用する

.NET Framework 3.5 以降では、ASP.NET AJAX 機能が標準化され、AJAX 対応の Web アプリケーションを簡単に作成することができます。 ASP.NET AJAX 機能を使用することで、応答性に優れた使いやすいユーザインタフェース (UI) をもつ Web ページを作成することができます。 ASP.NET AJAX 機能については、Visual Studio のドキュメントの「ASP.NET AJAX の概要」を参照してください。

サーバーコントロール

ASP.NET AJAX サーバーコントロールを使用することにより、部分ページ レンダリングや非同期ポストバック時の進行状況の表示などを、クライアント スクリプトをほとんど使用せずに作成できるようになります。 ASP.NET AJAX サーバーコントロールについては、Visual Studio のドキュメントの「AJAX サーバー コントロール」を参照してください。

.NET Framework 4 では、以下のサーバーコントロールを標準で使用することができます。 このサーバーコントロールは、ツールボックスの[AJAX Extensions]タブにあり、作成する Web ページに貼り付けて使用することができます。

ScriptManager コントロール

ASP.NET AJAX 機能を使用するためのクライアントスクリプトを管理します。 ScriptManager コントロールは、ASP.NET AJAX 機能を使用するのに必要となるコントロールで、UpdatePanel、UpdateProgress、および Timer など ASP.NET AJAX サーバー コントロールを使用するには、このコントロールを Web ページに配置する必要があります。

UpdatePanel コントロール

ページ全体を最新の情報に更新するのではなく、ページの選択した部分を最新の情報に更新できます。 既存のコントロールを UpdatePanel コントロール内に配置するだけで、パネル内のページの一部だけを更新できます。

UpdateProgress コントロール

UpdatePanel コントロールでの部分ページ更新に関するステータス情報を表示できます。

Timer コントロール

定義された間隔でポストバックを実行できます。 Timer コントロールを使用してページ全体をポストするか、UpdatePanel コントロールと共に使用して、定義された間隔で部分ページ更新を実行します。

XML Web サービスの開発

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

NetCOBOL for .NET で XML Web サービスを開発する場合、 ASP.NET 技術を利用して XML Web サービスクラスを作成します。 ここでは、ASP.NET 技術には含まれない、 COBOL に特化した部分の XML Web サービスの作成方法について説明します。

ここで説明する内容は、局所的な情報を寄せ集めたものになっています。 必要に応じて適宜参照してください。

なお、XML Web サービスの概要や開発の全体像を知りたい場合は、 チュートリアルの XML Web サービスの開発を参照してください。

このセクションの内容

COBOL コードの配置

XML Web サービスの開発では、 XML Web サービスクラス作成します。 XML Web サービスクラスは.cob ファイルに記述するのが一般的ですが、 別のアセンブリ内に記述して、 .asmx ファイルでそのアセンブリを関連付けるといった方法も選択できます。 これらの COBOL コードの配置方法について説明します。

WebService クラスの継承

XML Web サービスクラスは、 必ず System.Web.Service.WebService クラスを継承して作成します。 NetCOBOL for .NET でこのクラスを継承する場合の、 コーディング例を説明しています。

WebMethodAttribute カスタム属性を付加する

XML Web サービスメソッドには、 WebMethodAttribute カスタム属性を付ける必要があります。 このコーディング例について説明しています。

COBOL 独自データ型のパラメタ

XML Web サービスメソッドに COBOL 独自データ型をパラメタとして渡す場合の注意事項について説明しています。

デバッグ

XML Web サービスのデバッグでは、 外部プロセスで実行中のプログラムにアタッチして、 任意の XML Web サービスメソッドをデバッグしたい場合があります。 このデバッグ方法について説明しています。

関連トピックス

.NET Framework のドキュメントの ASP.NET と XML Web サービス クライアントを使用して作成した XML Web サービス

XML Web サービスを作成する場合に、必要となる知識を提供しています。

MSDN ライブラリの UDDI: XML Web サービス

.NET Framework のドキュメントの XML Web サービス ディレクトリ

UDDI に関する情報を提供します。UDDI は、Web におけるポータブルサントのように、利用者がウェブサービスを探索するためのディレクトリサービスです。

COBOL コードの配置

XML Web サービスクラスを作成する場合、その手続きを記述する方法には、以下の方法があります。

- 分離コードファイルに COBOL コードを配置する
- 個別のアセンブリに COBOL コードを配置する

上記のいずれの場合も、.asmx ファイルに含まれている@ WebService ディレクティブの指示に従い、XML Web サービスの URL アドレスをその実装(手続き)に関連付けます。

@ WebService ディレクティブの詳細については、 .NET Framework のドキュメントのチュートリアル : ASP.NET を使用する基本的な XML Web サービスの構築および XML Web サービスでの @ WebService ディレクティブを参照してください。

@ WebService ディレクティブで提供されるサンプルを NetCOBOL for .NET で使用する場合は、言語属性 Language に"Fujitsu.COBOL"を設定します。

分離コードファイルに COBOL コードを配置する

.asmx ファイルとは別のファイル(.asmx.cob ファイル)に COBOL コードを配置したい場合、@ WebService ディレクティブの CodeBehind にそのファイル名を指定することで、分離コードファイルに COBOL コードを配置できます。

たとえば、仮想ディレクトリルートの App_Code サブフォルダに格納した SimpleService.cob ファイルに SimpleService クラスのコードを配置したい場合、.asmx ファイルの@ WebService ディレクティブの指定は以下のようになります。

```
【.asmx ファイル】
<%@ WebService Language="Fujitsu.COBOL"
            CodeBehind="~/App_Code/SimpleService.cob"
            Class="MyName.SimpleService" %>
```

```
【App_Code¥SimpleService.cob ファイル】
     CLASS-ID. SimpleService AS "MyName.SimpleService" INHERITS WebServiceClass.
     ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
     REPOSITORY.
        CLASS WebServiceClass AS "System.Web.Services.WebService"
          :
```

「~/」は仮想ディレクトリのルートを表します。また、 分離コードファイルは、.asmx ファイルと一緒に、Web サーバ上に配置する必要があります。

個別のアセンブリに COBOL コードを配置する

COBOL コードを個別のアセンブリ(.dll)に含め、.asmx ファイルの Class に、そのアセンブリ名を記述することで、個別のアセンブリに COBOL コードを配置することができます。

たとえば、MyAssembly.dll に含まれる Myname.SimpleService クラスを参照する場合、.asmx ファイルの@ WebService ディレクティブの指定は以下のようになります。

```
【.asmx ファイル】
<%@ WebService Language="COBOL"
            Class="MyName.SimpleService,MyAssembly" %>
```

```
【MyAssembly.dll】
     CLASS-ID. SimpleService AS "MyName.SimpleService" INHERITS WebServiceClass.
     ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
     REPOSITORY.
        CLASS WebServiceClass AS "System.Web.Services.WebService"
          :
```

個別のアセンブリ（.dll ファイル）は、仮想ディレクトリルートの¥Bin サブフォルダに配置する必要があります。

WebService クラスの継承

NetCOBOL for .NET で XML Web サービスクラスを作成する場合、 必ず System.Web.Services.WebService クラスを継承する必要があります。 このクラスは、コードから ASP.NET Web サービスメソッドを呼び出すために必要になります。

プロジェクト作成時に ASP.NET Web サービスのテンプレートを選択して生成された .asmx.cob ファイルのクラス定義では、 System.Web.Services.WebService クラスを自動的に継承しています。

System.Web.Services.WebService クラスを継承する場合、以下のように記述します。

```
 IDENTIFICATION DIVISION.
CLASS-ID. MyWebService AS "MyWebServiceClasse" INHERITS WebService.   *>[1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS WebService AS "System.Web.Services.WebService"  *>[2]
```

[1]：MyWebService クラスは、外部名として MyWebServiceClasse という名前を持ちます。INHRITS 句には継承するクラス名を指定します。ここでは、WebService クラスを指定しています。

[2]：REPOSITORY 段落では、内部名と外部名を関連付けます。ここではクラス指定子で内部名"WebService"を外部名"System.Web.Services.WebService"に関連付けます。

関連トピックス

.NET Framework のドキュメントの WebService クラス

System.Web.Services.WebService クラスについて説明します。

NCOBU00523

WebMethodAttribute カスタム属性を付加する

XML Web サービスとして公開するメソッドには、 カスタム属性の WebMethodAttribute を付ける必要があります。

メソッド定義に、 WebMethodAttribute カスタム属性を付加するには、 以下のように記述します。

```
CLASS-ID. MyWebService AS "MyWebServiceClasse" INHERITS WebService.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE WebMethod CLASS WebMethodAttribute  *>[2]
   .
REPOSITORY.
   CLASS WebService AS "System.Web.Services.WebService"
   CLASS WebMethodAttribute AS "System.Web.Services.WebMethodAttribute"  *>[3]

METHOD-ID. MYMETHOD AS "MyMethod" CUSTOM-ATTRIBUTE IS WebMethod.  *>[1]
DATA DIVISION.
LINKAGE SECTION.
01 RET-VAL OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION RETURNING RET-VAL.
END METHOD MYMETHOD.
```

[1] : XML Web サービスメソッドとして公開するメソッドのメソッド名段落に CUSTOM-ATTRIBUTE 句を指定します。CUSTOM-ATTRIBUTE 句にはそのメソッドに付加するカスタム属性名を指定します。ここでは、WebMethod を指定しています。

[2] : 特殊名段落(SPECIAL-NAMES)でカスタム属性を定義します。CUSTOM-ATTRIBUTE 句の CLASS 指定には、カスタム属性が定義されているクラス名を指定します。ここでは、クラス名として WebMethodAttribute を指定しています。なお、クラス名以外の属性が省略された場合、そのカスタム属性のデフォルト値が有効になります。

[2] : REPOSITORY 段落では、内部名と外部名を関連付けます。ここではクラス指定子で内部名"WebMethodAttribute"を外部名"System.Web.Services.WebMethodAttribute"に関連付けます。

関連トピックス

.NET Framework のドキュメントの WebMethodAttribute クラス

WebMethodAttribute カスタム属性を定義している
System.Web.Services.WebMethodAttribute クラスについて説明しています。

COBOL 独自データ型のパラメタ

Web サービスメソッドのパラメタを、.NET データ型にマッピングできない COBOL データ型(これを COBOL 独自データ型といいます)で定義した場合、COBOL 以外の言語で記述されたプログラムから、この Web サービスメソッドを呼び出した時、パラメタは SOAP によって自然にマーシャリングされません。このため、追加作業をする必要があります。必要事項に関するガイドラインとして、.NET Framework のドキュメントの SOAP メッセージの書式のカスタマイズを参照してください。

NCOBU00525

デバッグ

デバッグの概要

Visual Studio 上で作業している場合、XML Web サービスデバックは、以下の方法で簡単にできます。

1. XML Web サービスのプロジェクトを開きます。
2. プロジェクトのプロパティページを表示させ、[開始オプション]を選択します。
3. デバッグ開始時に最初に実行したいページを指定します。
4. [デバッグ]メニューから、[デバッグの開始]をクリックすると、プロジェクトが必要な設定を行い、デバッガーが自動的に開始されます。COBOL コードにブレークポイントを設定しておけば、テストページから起動された後、ブレークポイントで実行が中断します。

なお、Visual Studio の外部プロセス内で実行中のプログラムにアタッチして、XML Web サービスメソッドに制御が渡されたところでデバッグを行いたい場合は、上記の方法ではデバッグできないため、以下に説明する方法でデバッグします。

アタッチが必要なデバッグの手順

アタッチしてデバッグするには、以下の手順で行います。

- XML Web サービスをデバッグ可能に設定する
- ASP.NET ワーカープロセスを選択して実行する

以下に、IIS 上で動作している XML Web サービスにアタッチしてデバッグする場合を例にして手順を説明します。

1. XML Web サービスをデバッグ可能に設定します。

   Web サイトの ASP.NET 構成ファイル（Web.config）の compiler 要素の debug 属性を true に設定します。

   ```
   <configuration>
   ...
     <system.web>
     ...
       <compilation debug="true" />
     ...
   ```

   Web サイトをビルドし、IIS に配置します。

2. Web ブラウザに、テストを行う.asmx ファイルの名前を、ローカルホストと仮想ディレクトリ名を含んだ名前で指定します。以下は入力の例です。

   ```
   http://localhost/COBOL/Web/SimpleService.asmx
   ```

   これにより、ASP ワーカープロセス(w3wp.exe)が開始され、ASP.NET のサービスヘルプページが表示されます。

3. Visual Studio で、コードファイルを開きます。
4. Visual Studio の[デバッグ]メニューから、[プロセスにアタッチ]を選択します。

   これにより、使用しているマシン上で、実行中のプロセスの一覧を含んだ、プロセスダイアログが表示されます。プロセスの 1 つが、「w3wp.exe」です。

5. w3wp.exe プロセスを選択して、「アタッチ」をクリックします。
6. コードにブレークポイントを設定します。
7. 使用している Web ブラウザに戻り、XML Web サービスを呼び出すか、ASP.NET ページを更新します。

これにより、設定したブレークポイントに到達するまでコードが実行されます。ブレークポイントに達すると、制御が Visual Studio デバッガーに渡され、中断した行を表示します。

関連トピックス

[ブレークポイントの設定]

NetCOBOL for .NET で Visual Studio を使ってデバッグする場合の、ブレークポイントの設定方法について説明しています。

NCOBU00865

SQL CLR データベースオブジェクトの開発

このセクションの内容

[ランタイムアセンブリをセットアップする]

SQL CLR でのランタイムアセンブリのセットアップ方法について説明します。

[ユーザ定義メッセージを送信する]

SQL CLR でのユーザ定義メッセージを送信する機能について説明します。

[SQL CLR データベースオブジェクトで使用できない機能]

SQL CLR でのプログラミングで使用できない機能について説明します。

[SQL CLR データベースオブジェクト作成時の注意点]

SQL CLR データベースオブジェクトを作成する際の注意点について説明します。

[SQL CLR データベースオブジェクト配置時の注意点]

SQL CLR データベースオブジェクトをデータベースに配置する際の注意点について説明します。

[SQL CLR データベースオブジェクトのアクセス時の注意点]

SQL CLR でのデータアクセスにおける注意事項について説明します。

関連トピックス

.NET Framework のドキュメントの SQL Server の共通言語ランタイム統合

SQL CLR データベースオブジェクトを開発するために必要となる基本的な情報を提供しています。

SQL Server オンラインブックの CLR (共通言語ランタイム) 統合のプログラミング概念

SQL CLR データベースオブジェクトのプログラミングについて詳細に説明しています。

ランタイムアセンブリをセットアップする

ここでは、NetCOBOL for .NET で SQL CLR データベースオブジェクトを開発・実行するために必要な、ランタイムアセンブリのセットアップ方法について説明します。

このセクションの内容

ランタイムアセンブリを登録する

ランタイムアセンブリを登録する方法について説明します。

ランタイムアセンブリを削除する

ランタイムアセンブリを削除する方法について説明します。

ランタイムアセンブリを更新する

ランタイムアセンブリを更新する方法について説明します。

SQL Server オンラインブックでの関連トピック

- アセンブリの実装
- CLR 統合アセンブリの管理

NCOBU00867

ランタイムアセンブリを登録する

NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトを配置、実行するためには、あらかじめランタイムアセンブリをデータベースに登録しておく必要があります。 ここでは、ランタイムアセンブリをデータベースに登録する方法について説明します。

ランタイムアセンブリを登録する SQL Server がインストールされているコンピュータには、登録するランタイムアセンブリと同じバージョンの NetCOBOL for .NET 製品がインストールされている必要があります。

登録するランタイムアセンブリは次のファイルです。

- Fujitsu.COBOL.dll
- Fujitsu.COBOL.resources.dll

ランタイムアセンブリは Windows がインストールされているドライブの以下の場所にインストールされます。

アセンブリファイル	Windows の種類	インストールフォルダ（Windows がインストールされているドライブ上のパス）
Fujitsu.COBOL.dll	32-bit	¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime
	64-bit	¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime
Fujitsu.COBOL.resouces.dll	32-bit	¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime¥ja
	64-bit	¥Program Files (x86)¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime¥ja

また、ランタイムアセンブリをインストールする前に該当するアセンブリのポリシーファイルを退避させてください。

```
C:¥temporary_dir>MOVE C:¥Windows¥Microsoft.NET¥assembly¥GAC_MSIL¥policy.8.0.Fujitsu.COBOL .
```

CREATE ASEMBLY コマンドでランタイムアセンブリをインストール後、 該当するアセンブリのポリシーファイルを元に戻してください。

```
C:¥temporary_dir>MOVE policy.8.0.Fujitsu.COBOL C:¥Windows¥Microsoft.NET¥assembly¥GAC_MSIL¥
```

ポリシーファイルがある場合は、以下のエラーが発生します。

```
SQL72014: .Net SqlClient Data Provider: Msg 6586, Level 16, State 1, Line 1 Assembly 'Fujitsu.COBOL' could not be installed because existing policy would keep it from being used.
```

ランタイムアセンブリを登録する場合は Transact-SQL の CREATE ASSEMBLY 文を使用します。

以下は、sqlcmd ユーティリティを使用して CREATE ASSEMBLY でランタイムアセンブリを登録する例です。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB -d COBOLSample
1> CREATE ASSEMBLY [Fujitsu.COBOL]
2> FROM 'C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime
V8.0¥Runtime¥Fujitsu.COBOL.dll'
3> WITH PERMISSION_SET=SAFE
4> GO
1> CREATE ASSEMBLY [Fujitsu.COBOL.resources]
2> FROM 'C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime
V8.0¥Runtime¥ja¥Fujitsu.COBOL.resources.dll'
3> WITH PERMISSION_SET = SAFE
4> GO
```

この例では、 Microsoft SQL Server 2016 Express LocalDB が動作しているコンピュータで、"C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime"にインストールされているランタイムアセンブリを COBOLSample というデータベースに登録しています。

- ランタイムアセンブリを登録する場合、 CREATE ASSEMBLY 文で WITH PERMISSION_SET を指定する場合は、SAFE 以外を指定しないでください。 SAFE 以外を指定した場合は動作保証されません。
- ランタイムアセンブリの登録は、NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトを配置するデータベース毎に行う必要があります。
- 1 つのデータベースには、1 組のランタイムアセンブリだけが登録できます。 バージョンの異なる同名のランタイムアセンブリを複数登録することはできません。
- リモートコンピュータの SQL Server に接続してランタイムアセンブリを登録する場合は、CREATE ASSEMBLY 文に指定するパスは、 リモートコンピュータにインストールされているランタイムアセンブリのパスを指定する必要があります。

詳細については、SQL Server オンラインブックの以下のトピックを参照してください。

sqlcmd ユーティリティの使い方について

sqlcmd ユーティリティの使用

CREATE ASSEMBLY 文について

CREATE ASSEMBLY (Transact-SQL)

NCOBU00868

ランタイムアセンブリを削除する

ここでは、データベースに登録されているランタイムアセンブリを削除する方法について説明します。

NetCOBOL for .NET 製品をコンピュータからアンインストールした場合は、必ずデータベースに登録されているランタイムアセンブリも削除しなければなりません。

ランタイムアセンブリを削除する場合は Transact-SQL の DROP ASSEMBLY 文を使用します。

NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトがデータベースに存在する場合は、 ランタイムアセンブリと依存関係をもつため、その SQL CLR データベースオブジェクトおよびアセンブリをランタイムアセンブリより先に削除する必要があります。

例えば、COBOLSample データベースに NetCOBOL for .NET で作成した"MyLib"というアセンブリが存在し、そのアセンブリ内のメソッド"StoredProcedure1"、"Function1"、"Trigger1"がそれぞれ、ストアドプロシージャ、ユーザ定義関数、トリガとして登録されている場合は、Transact-SQL の DROP PROCEDURE 文、DROP FUNCTION 文および DROP TRIGGER 文を使用して各 SQL CLR データベースオブジェクトを削除し、最後に DROP ASSSEMBLY 文で"MyLib"を削除します。

以下の例は、Microsoft SQL Server 2016 Express LocalDB が動作しているコンピュータで、sqlcmd ユーティリティを使用して各 SQL CLR データベースオブジェクトおよびアセンブリを削除しています。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB -d COBOLSample
1> DROP PROCEDURE StoredProcedure1
2> GO
1> DROP FUNCTION Function1
2> GO
1> DROP TRIGGER Trigger1
2> GO
1> DROP ASSEMBLY MyLib
2> GO
```

SQL CLR データベースオブジェクトの削除はどの順番でもかまいません。

ランタイムアセンブリと依存関係をもつ SQL CLR データベースオブジェクトおよびアセンブリをすべて削除した後は、ランタイムアセンブリを削除します。

以下は、sqlcmd ユーティリティを使用して DROP ASSEMBLY 文でランタイムアセンブリを削除する例です。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB -d COBOLSample
1> DROP ASSEMBLY [Fujitsu.COBOL]
2> GO
1> DROP ASSEMBLY [Fujitsu.COBOL.resources]
2> GO
```

- ランタイムアセンブリの削除は、ランタイムアセンブリが登録されているデータベース毎に行う必要があります。

詳細については、SQL Server オンラインブックの以下のトピックを参照してください。

sqlcmd ユーティリティの使い方について

sqlcmd ユーティリティの使用

DROP PROCEDURE 文について

DROP PROCEDURE (Transact-SQL)

DROP FUNCTION 文について

DROP FUNCTION (Transact-SQL)

DROP TRIGGER 文について

DROP TRIGGER (Transact-SQL)

DROP ASSEMBLY 文について

DROP ASSEMBLY (Transact-SQL)

NCOBU00869

ランタイムアセンブリを更新する

ここでは、データベースに登録されているランタイムアセンブリを更新する方法について説明します。

NetCOBOL for .NET 製品をバージョンアップやレベルアップなどした場合は、 必ずデータベースに登録されているランタイムアセンブリも更新しなければなりません。

NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトが存在する場合、 SQL CLR データベースオブジェクトを再ビルドせずに、ランタイムアセンブリだけ更新したいときは、Transact-SQL の ALTER ASSEMBLY 文を使用します。

以下は、sqlcmd ユーティリティを使用して ALTER ASSEMBLY でランタイムアセンブリを更新する例です。

```
C:¥> sqlcmd -E -S (localdb)¥MSSQLLocalDB -d COBOLSample
1> ALTER ASSEMBLY [Fujitsu.COBOL]
2> FROM 'C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime
V8.0¥Runtime¥Fujitsu.COBOL.dll'
3> GO
1> ALTER ASSEMBLY [Fujitsu.COBOL.resources]
2> FROM 'C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime
V8.0¥Runtime¥ja¥Fujitsu.COBOL.resources.dll'
3> GO
```

この例では、Microsoft SQL Server 2016 Express LocalDB が動作しているコンピュータで、COBOLSample というデータベースに登録されているランタイムアセンブリを"C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime"にインストールされた新しいランタイムアセンブリに更新しています。

- ランタイムアセンブリの更新を行う前に、必ずデータベースのバックアップをしておくことをお薦めします。
- ランタイムアセンブリの更新は、ランタイムアセンブリが登録されているデータベース毎に行う必要があります。
- ランタイムの下位バージョンへの変更は、以下の手順で行います。
    1. NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトを削除します。
    2. 現在のランタイムアセンブリを削除します。
    3. 下位バージョンのランタイムアセンブリを登録します。
    4. NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトを再登録します。

詳細については、SQL Server オンラインブックの以下のトピックを参照してください。

sqlcmd ユーティリティの使い方について

sqlcmd ユーティリティの使用

ALTER ASSEMBLY 文について

ALTER ASSEMBLY (Transact-SQL)

NCOBU00870

ユーザ定義メッセージを送信する

SQL CLR データベースオブジェクトで、 クライアント側にユーザ定義メッセージを送信したい場合は、DISPLAY 文(コンソール出力)を使用することができます。 これは、Transact-SQL の PRINT 文と同等の機能を提供します。

SQL CLR データベースオブジェクトで、DISPLAY 文を使用する場合の書き方は以下のとおりです。

【書き方】

```
          ì 一意名  ü
DISPLAY   í         ý … [UPON 呼び名]
          î 定数    þ
```

(注 1)WITH NO ADVANCING 指定は無視されます。

(注 2)呼び名には CONSOLE/SYSOUT/SYSERR のいずれかを指定することができますが、SSOUT オプションで環境変数情報名を指定して出力先を変更することはできません。

ユーザ定義関数では、ユーザ定義メッセージを送信することはできません。 SQL CLR ユーザ定義関数で DISPLAY 文を使用した場合、以下の実行時エラーとなります。

- UPON 指定を省略した場合、または、呼び名に CONSOLE または SYSOUT をのいずれかを指定した場合

```
JMP0016I-U ファイル'SYSOUT'の読み込みまたは書き込みに失敗しました.
```

- 呼び名に SYSERR を指定した場合

```
JMP0088I-U メッセージの出力に失敗しました.
```

Transact-SQL の PRINT 文については、PRINT (Transact-SQL)を参照してください。

送信されたユーザ定義メッセージをクライアント側で受け取るには

COBOL のデータベース機能で送信されたユーザ定義メッセージをクライアント側で受け取る場合、以下の方法があります。

- ADO.NET 接続でユーザ定義メッセージをトレース出力する
- ODBC 接続でユーザ定義メッセージを受け取る

関連トピック: SQL Server オンラインブックのエラーとメッセージの処理

ADO.NET 接続でユーザ定義メッセージをトレース出力する

ADO.NET 接続(SqlClient データプロバイダー利用)による COBOL のデータベース機能を利用したクライアントアプリケーションでは、 Transact-SQL で作成されたストアドプロシージャやトリガなどのデータベースオブジェクト、または、SQL CLR データベースオブジェクトから送信されたユーザ定義メッセージを.NET Framework のトレースリスナーに転送することができます。

データベースオブジェクトから送信されたメッセージを.NET Framework のトレースリスナーに送るには、環境変数@CBR_SQL_MESSAGE_TRACE をアプリケーション構成ファイルの<environmentSettings>要素に追加し、値に"YES"を設定することでトレース出力機能が有効になります。

SQL Server から送信されたユーザ定義メッセージのトレース出力機能は、COBOL のデータベース機能を SqlClient データプロバイダーを利用して ADO.NET 接続で使用した場合だけ有効です。以下の場合は利用できません。

- ADO.NET 接続で SqlClient データプロバイダー以外のプロバイダーを使用した場合
- ODBC 接続でデータベース機能を使用した場合
- COBOL のデータベース機能を使用せずに SQL Server に接続した場合

.NET Framework のトレースリスナーは、デフォルトではデバッグ出力が設定されています。 (例えば、Visual Studio のデバッガ上でクライアントアプリケーションをデバッグ実行している場合は、Visual Studio の[出力]ウィンドウに表示されます。) 特定のファイルやコンソールに出力したい場合は、アプリケーション構成ファイルに<system.diagnostics>要素-<trace>要素-<listeners>要素を追加し、利用したいトレースリスナーを設定することが可能です。

トレースリスナーは、System.Diagnostics 名前空間で提供されています。 ファイルに出力する場合は、TextWriterTraceListener を使用します。コンソールに出力する場合には、ConsoleTraceListener を使用します。アプリケーション構成ファイルには以下のように記述します。

```
<configuration>
  <system.diagnostics>
    <trace autoflush="true" indentsize="4">
      <listeners>
        <add name="tracelog"
          type="System.Diagnostics.TextWriterTraceListener"
          initializeData="C:¥trace.log" />
        <add name="consoleout"
          type="System.Diagnostics.ConsoleTraceListener" />
              :
      </listeners>
    </trace>
  </system.diagnostics>
              :
  <fujitsu.cobol>
    <runtime>
      <environmentSettings>
        <add key="@CBR_SQL_MESSAGE_TRACE" value="YES"/>
              :
      </environmentSettings>
              :
    </runtime>
  </fujitsu.cobol>
</configuration>
```

アプリケーション構成ファイルの<system.diagnostics>要素は、実行環境設定ユーティリティを使用して編集することはできません。 トレースリスナーを追加する場合はテキストエディターなどでアプリケーション構成ファイルを開いて編集してくだい。

.NET Framework のトレースリスナーについての詳細は、 .NET Framework のドキュメントのトレース リスナーを参照してください。

ODBC 接続でユーザ定義メッセージを受け取る

ODBC 接続では、Transact-SQL で作成されたストアドプロシージャやトリガなどのデータベースオブジェクト、または、SQL CLR データベースオブジェクトから送信されたユーザ定義メッセージは、例外事象としてホスト変数 SQLMSG で受け取ることができます。

- ホスト変数 SQLSTATE には、"01000"が設定されます。
- ユーザ定義メッセージはデータオブジェクトから最初に送信されたユーザ定義メッセージのみ受信できます。

NCOBU00873

SQL CLR データベースオブジェクトで使用できない機能

SQL CLR で使用できない COBOL 機能

SQL CLR データベースオブジェクトで使用できない COBOL 機能について示します。

- 以下の機能は/optionset:sqlclr コンパイラオプションによって翻訳エラーとなります。
  - スタティック定義のデータ部で宣言されたデータのスタティックコンストラクタ以外での書き換え
  - スタティック定義のデータ部でのファイル宣言
  - スタティック定義のデータ部での埋込み SQL 宣言
  - スタティック定義のデータ部での WITH NO SET 指定のないプロパティ句
  - 外部属性(EXTERNAL 指定)のデータおよびファイル

  詳細は、INITONLY を参照してください。
- 以下の機能を使用した場合は実行時メッセージ(JMP0170I-U)が出力されます。
  - ACCEPT 文(コンソールまたは SSIN 翻訳オプションのファイル指定によるデータ入力)
  - DISPLAY 文(SSOUT 翻訳オプションのファイル指定によるデータ出力)
  - コマンド行引数操作
  - 環境変数操作
  - 行順、順、相対、索引ファイル
  - 印刷、表示ファイル
  - 整列併合
  - メモリ獲得クラス
  - 動的プログラム構造(CALL 一意名/CANCEL 一意名)
- 以下の関数・サブルーチンを呼び出した場合はエラーで復帰します。
  - COBOL ファイルユーティリティ関数
  - データロックサブルーチン
  - CBL サブルーチン
  - AcuCOBOL サブルーチン
- 以下の機能については動作保証していません。
  - プログラム定義
  - コンストラクタメソッドにおける親クラスのコンストラクタの呼び出し
  - 要素も配列である多次元配列（ジャグ配列）

SQL CLR で使用できない.NET Framework 機能

SQL CLR データベースオブジェクトで使用できない.NET Framework 機能は次のとおりです。

Windows の GUI 機能などの利用

SQL CLR データベースオブジェクトのプログラミングにおいて、使用が制限されている機能があります。 詳細については、.NET Framework のドキュメントの SQL Server プログラミングとホスト保護属性および、 SQL Server オンラインブックのアセンブリのデザインの「アセンブリに関する制限事項」を参照してください。

XML Web サービスの参照/WCF サービスの参照

NetCOBOL for .NET では XML Web サービスや WCF サービスの参照をした場合、 複数のモジュールから構成されるアセンブリとなるため、SQL Server に配置することができません。

プラットフォーム呼び出し

Windows 版 NetCOBOL で作成した COBOL 独自項目のパラメタを持つプログラムをプラットフォーム呼び出し機能を使用して呼び出すことはできません。

NCOBU00874

SQL CLR データベースオブジェクト作成時の注意点

ここでは、 NetCOBOL for .NET で SQL CLR データベースオブジェクトを作成する場合の注意点について説明します。

このセクションの内容

全般的な SQL CLR データベースオブジェクト作成時の注意点

SQL CLR データベースオブジェクトを作成する場合、以下の注意が必要です。

- SQL CLR データベースオブジェクトを動作させるためには、ビルド時に/optionset:sqlclr コンパイラオプションを指定してください。このオプションを指定しなかった場合、または、/optionset:sqlclr コンパイラオプションの後に以下のいずれかの翻訳オプションを指定した場合は動作保証されません。
  - RCS 翻訳オプションに SJIS-UCS2 以外を指定した場合
  - NOINITONLY 翻訳オプションを指定した場合
  - SQLSCOPE 翻訳オプションに OBJECT を指定した場合

- SQL CLR データベースオブジェクトを実行するためには、クラスライブラリとして コンパイルする必要があります。 ビルド時に/target:library コンパイラオプションを指定してください。
- SQL CLR データベースオブジェクトのパラメタや戻り値には、COBOL 独自データ型を指定することはできません。SQL Server オンラインブックの CLR パラメーター データのマッピングで示す.NET データ型のみ使用できます。

  なお、 SQL CLR ストアドプロシージャに戻り値を指定する場合、 以下のいずれかの.NET データ型のみ戻り値のデータ型として使用することができます。
  - System.Int16 型
  - System.Int32 型
  - System.Data.SqlTypes.SqlInt16 型
  - System.Data.SqlTypes.SqlInt32 型

  .NET データ型の COBOL 表現については、.NET データ型の COBOL 表現を参照してください。

- 固定長の有効桁数と小数点以下桁数を持つ数値データ型(decimal 型および numeric 型)のパラメタを使用する場合は、カスタム属性でパラメタの有効桁数(Precision)と小数点以下桁数(Scale)を指定する必要があります。 設定方法については、SQL CLR データベースオブジェクトのパラメタ型とホスト変数の対応の注意事項である“固定長の有効桁数と小数点以下桁数を持つ数値データ型のパラメタを使用する場合”を参照してください。
- 実行環境の設定をすることはできません。
- SEPARATE 指定のない符号付外部 10 進項目の符号部の表現形式は富士通形式のみ使用できます。
- NetCOBOL for .NET ランタイムシステムが実行時に出力する重大コード U 以外のメッセージは、クライアントに送信されます。ただし、SQL CLR ユーザ定義関数は除きます。詳細については、SQL CLR ユーザ定義関数作成時の注意点を参照して下さい。
- SQL CLR データベースオブジェクトで STOP RUN 文を使用した場合、 実行時例外 (Fujitsu.COBOL.StopRunException)が発生します。

NCOBU00919

SQL CLR ユーザ定義関数作成時の注意点

SQL CLR ユーザ定義関数を作成する場合は、SQL CLR データベースオブジェクト作成時の注意点に加えて以下の注意が必要となります。

- ユーザ定義関数では、データベースへのアクセスはデータの参照のみ可能であり、データを変更することはできません。 データを参照する場合、Microsoft.SqlServer.Server 名前空間で提供されている、SqlFunction カスタム属性の DataAccess 名前付きパラメタに Read を指定する必要があります。また、システムデータをアクセスする場合は、SystemDataAccess 名前付きパラメタに Read を指定します。
- ユーザ定義関数に指定可能なパラメタは、入力パラメタ(値渡し)だけです。 メソッド定義での手続き部の見出しでは、必ず BY VALUE を指定してください。
- NetCOBOL for .NET ランタイムシステムが実行時にメッセージを出力する場合、重大度コードが I または W のメッセージは送信されません。重大度コードが E のメッセージは、実行時例外(Fujitsu.COBOL.COBOLRuntimeException)になります。

ユーザ定義関数を設計する上でのガイドラインや注意点についての詳細は、 SQL Server オンラインブックのユーザー定義関数を参照してください。

COBOL のデータベース機能を使用する場合の注意点

SQL CLR データベースオブジェクトでデータベース機能を使用する場合は、自動的に ADO.NET 接続(SqlClient データプロバイダー利用)のアクセスとなります。 ADO.NET 接続での注意点については、ADO.NET データプロバイダー使用時の注意事項を参照してください。

SQL CLR データベースオブジェクトでデータベース機能を使用する場合、以下の注意が必要です。

- データベースへの接続は、SQL CLR データベースオブジェクトが実行されるデータベースだけ接続することができます。 この接続は、埋込み SQL 文が実行されるときに SQL Server 内部で直接クライアント側から接続されているコンテキストに対し自動的に接続します。 (SQL CLR ではこの接続をコンテキスト接続と呼んでいます。) SQL 文の実行が終了すると、この接続は自動的に閉じられます。 従って、SQL CLR データベースオブジェクトでは接続および接続の有効範囲が影響するトランザクションに関連する操作を行うことはできません。以下の SQL 文を使用した場合は、実行時例外が発生します。
  - CONNECT 文
  - DISCONNECT 文
  - SET CONNECTION 文
  - COMMIT 文
  - ROLLBACK 文

コンテキスト接続については、SQL Server オンラインブックのコンテキスト接続を参照してください。

- SQL 情報は設定・変更することはできません。 なお、SQL CLR データベースオブジェクトにおける SQL 情報については、以下の表に示す値が設定された場合と同等の動作をします。

情報名	値
@SQL_VALUE_N_PADDING	SPACE_OF_SPECIFICATION
@SQL_TARGET_N_PADDING	SPACE_OF_SPECIFICATION
@SQL_ADONET_CURSOR_DEFAULT	READ_WRITE
@SQL_ADONET_FORUPDATE_CLAUSE	ENABLE

- SQL CLR ストアドプロシージャや SQL CLR ユーザ定義関数の連絡節にパラメタや戻り値として宣言したデータ項目は、直接ホスト変数として使用することはできません。 入力パラメタは、ホスト変数に転記して使用してください。 出力パラメタや戻り値は、ホスト変数から転記して設定してください。 詳細については、SQL CLR データベースオブジェクトのパラメタ型とホスト変数の対応を参照してください。
- SQL CLR データベースオブジェクトでは、カーソル宣言で FOR UPDATE が指定されたカーソルを使用することはできません。
- SQL CLR ユーザ定義関数で、ストアドプロシージャを呼び出す場合は、SqlFunction カスタム属性の SystemDataAccess 名前付きパラメタに Read を指定してください。

NCOBU00875

SQL CLRデータベースオブジェクトのパラメタ型とホスト変数の対応

SQL CLR データベースオブジェクトでは、メソッドのパラメタや戻り値に指定可能なデータ型は、 SQL Server オンラインブックの CLR パラメーター データのマッピングで示す.NET データ型に限られているため、連絡節に COBOL 独自データ型を指定することはできません。 ここでは、ホスト変数とメソッドの連結節に指定されたデータ項目との間でデータを受け渡す方法について説明します。

以下の表は、各 SQL Server 型に対応するホスト変数と連結節に指定されたデータ項目の受け渡し方法を示しています。

SQL Server 型	ホスト変数のデータ型	連絡節に指定可能なデータ型	受け渡し方法
nchar(1) nvarchar(1)	PIC N(1)	PIC N(1)	MOVE 文による転記
		System.String 型オブジェクト参照	SET 文による暗黙の型変換
		System.Char[]型オブジェクト参照	SET 文による暗黙の型変換
		System.Data.SqlTypes.SqlString 型オブジェクト参照	別項を参照
		System.Data.SqlTypes.SqlChars 型オブジェクト参照	別項を参照
nchar(n) nvarchar(n)	PIC X(n) PIC N(n)	System.String 型オブジェクト参照	SET 文による暗黙の型変換
		System.Char[]型オブジェクト参照	SET 文による暗黙の型変換
		System.Data.SqlTypes.SqlString 型オブジェクト参照	別項を参照
		System.Data.SqlTypes.SqlChars 型オブジェクト参照	別項を参照
smallint	PIC S9(4) BINARY PIC S9(4) COMP-5	BINARY-SHORT SIGNED PIC S9(4) COMP-5	MOVE 文による転記
		System.Data.SqlTypes.SqlInt16 型オブジェクト参照	別項を参照
int	PIC S9(9) BINARY PIC S9(9) COMP-5	BINARY-LONG SIGNEDPIC S9(9) COMP-5	MOVE 文による転記
		System.Data.SqlTypes.SqlInt32 型オブジェクト参照	別項を参照

numeric decimal	PIC S9(p) PACKED-DECIMAL PIC S9(p)V9(q) PACKED-DECIMAL PIC S9(p) DISPLAY PIC S9(p)V9(q) DISPLAY	System.Decimal 型オブジェクト参照	SET 文による暗黙の型変換
		System.Data.SqlTypes.SqlDecimal 型オブジェクト参照	別項を参照
real	COMP-1	COMP-1	MOVE 文による転記
		System.Data.SqlTypes.SqlSingle 型オブジェクト参照	別項を参照
float	COMP-2	COMP-2	MOVE 文による転記
		System.Data.SqlTypes.SqlDouble 型オブジェクト参照	別項を参照

注: 外部 10 進(DISPLAY)データ項目の符号は、以下のいずれかを指定します。

- SIGN IS LEADING SEPARATE CHARACTER
- SIGN IS TRAILING

.NET プログラミングでの代入時の適合については、COBOL 文法書の「11.8.2.6.1 代入時の適合」を参照してください。

SQL Server データ型パラメタとホスト変数との間でデータを受け渡す方法

System.Data.SqlTypes 名前空間で提供されている SQL Server データ型のパラメタとホスト変数の間でデータの受け渡しをする場合、MOVE 文や SET 文を使用してデータを直接受け渡すことはできません。 以下に SQL Server データ型のパラメタとホスト変数の間でデータを受け渡す方法について説明します。

SQL Server データ型のパラメタからホスト変数に値を格納する

SQL Server データ型のパラメタからホスト変数に値を格納する場合は、 SQL Server データ型オブジェクト参照の Value プロパティの値をホスト変数に格納します。 Value プロパティのデータ型は SQL Server データ型の種類によって異なります。 詳細は、.NET Framework のドキュメントの各 SQL Server データ型のリファレンスを参照してください。

以下の例では、SqlChars 型のパラメタの値を英数字データ項目のホスト変数に格納しています。 SqlChars 型の Value プロパティは System.Char[]型であり、日本語データ項目への型変換が可能であるため、SET 文を使用して値を格納しています。 代入時の暗黙の型変換については、COBOL 文法書の「11.8.2.6.1 代入時の適合」を参照してください。

```
REPOSITORY.
        ...
    CLASS CLASS-SQLCHARS AS "System.Data.SqlTypes.SqlChars"
    PROPERTY PROP-VALUE AS "Value"
        ...
DATA DIVISION.
WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE PIC X(5).
        ...
01 DATA1 PIC N(20).
        ...
    EXEC SQL END DECLARE SECTION END-EXEC.
        ...
LINKAGE SECTION.
        ...
01 P1 OBJECT REFERENCE CLASS-SQLCHARS.
        ...

PROCEDURE DIVISION USING BY VALUE P1 BY REFERENCE P2.
        ...

    SET DATA1 TO PROP-VALUE OF P1.
        ...
```

ホスト変数から SQL Server データ型のパラメタに値を格納する

ホスト変数から SQL Server データ型のパラメタに値を格納する場合は、 SQL Server データ型のコンストラクタを呼び出して新しいインスタンスを初期化します。コンストラクタのパラメタにはホスト変数の値を指定します。 詳細は、.NET Framework のドキュメントの各 SQL Server データ型のリファレンスを参照してください。

以下の例では、英数字データ項目のホスト変数を SqlString 型のパラメタに格納しています。 SqlString 型は System.String 型を指定するコンストラクタを呼び出して新しいインスタンスを初期化することが可能であり、System.String 型のパラメタは英数字データ項目からの暗黙の型変換が可能であるため、コンストラクタにホスト変数を直接指定して SqlString 型のパラメタを初期化しています。 パラメタでの暗黙の型変換については、COBOL 文法書の「11.8.2.6.2 引数の適合」を参照してください。

```
REPOSITORY.
        ...
    CLASS CLASS-SQLSTRING AS "System.Data.SqlTypes.SqlString"
        ...
DATA DIVISION.
WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE PIC X(5).
        ...
01 DATA2 PIC X(20).
        ...
    EXEC SQL END DECLARE SECTION END-EXEC.
        ...
LINKAGE SECTION.
        ...
01 P2 OBJECT REFERENCE CLASS-SQLSTRING.
        ...
PROCEDURE DIVISION USING BY VALUE P1 BY REFERENCE P2.
        ...

    SET P2 TO CLASS-SQLSTRING::"NEW" (DATA2).

        ...
```

固定長の有効桁数と小数点以下桁数を持つ数値データ型のパラメタを使用する場合

パラメタに固定長の有効桁数と小数点以下桁数を持つ数値データ型(decimal 型および numeric 型)を使用する場合は、System.Data.SqlTypes.SqlDecimal 型および、System.Decimal 型を使用し、Microsoft.SqlServer.Server.SqlFacetAttribute のカスタム属性でパラメタの有効桁数(Precision)と小数点以下桁数(Scale)を設定する必要があります。

以下の例では、SQL Server データ型が decimal(5, 2)となるパラメタを使用するために、System.Data.SqlTypes.SqlDecimal 型の有効桁数が 5、小数点以下桁数が 2 の入出力パラメタを作成しています。

```
SPECIAL-NAMES.
        ...
    CUSTOM-ATTRIBUTE CA-DECIMAL CLASS CLASS-SQLFACET
     PROPERTY PROP-SCALE IS 2
     PROPERTY PROP-PRECISION IS 5
        ...
REPOSITORY.
        ...
    CLASS CLASS-SQLDECIMAL AS "System.Data.SqlTypes.SqlDecimal"
    CLASS CLASS-SQLFACET AS "Microsoft.SqlServer.Server.SqlFacetAttribute"
    PROPERTY PROP-SCALE AS "Scale"
    PROPERTY PROP-PRECISION AS "Precision"
        ...
LINKAGE SECTION.
01 DEC1 OBJECT REFERENCE CLASS-SQLDECIMAL CUSTOM-ATTRIBUTE CA-DECIMAL.
PROCEDURE DIVISION USING BY REFERENCE DEC1.
        ...
```

関連トピック

- SQL Server オンラインブックの CLR パラメーター データのマッピング
- .NET Framework のデータ型と COBOL のデータ型の対応
- ADO.NET で扱うデータ型との対応

NCOBU00876

SQL CLR データベースオブジェクト配置時の注意点

SQL CLR データベースオブジェクトをデータベースに配置する際は、以下の点に注意してください。

- NetCOBOL for .NET で開発した SQL CLR データベースオブジェクトをデータベースに配置する際、以下のメッセージが表示された場合は、配置先のデータベースにランタイムアセンブリが登録されていません。

  > アセンブリ、`fujitsu.cobol, version=8.0.124.0, culture=neutral,publickeytoken=fac0fe3cab973246' が SQL カタログに見つかりませんでした。

  この場合は、配置先のデータベースにランタイムアセンブリを登録しておく必要があります。ランタイムアセンブリの登録方法についてはランタイムアセンブリを登録するを参照してください。

- SQL CLR データベースオブジェクトが含まれる同一アセンブリ内でSQL CLR データベースオブジェクトで使用できない機能を使用している場合は、アクセス許可のレベルを SAFE に設定して配置できない場合があります。

- SQL CLR データベースオブジェクトのビルド時に/platform コンパイラオプションに配置先の SQL Server のプラットフォームと異なるプラットフォームを指定した場合は配置することはできません。

- 配置する SQL CLR データベースオブジェクトが参照するランタイムアセンブリと配置先のデータベースに登録されているランタイムアセンブリのバージョンが一致しない場合、配置することができません。ランタイムアセンブリのバージョンが一致しない場合、以下のメッセージが表示されます。

  > アセンブリ 'xxxxxxxxxx' ではアセンブリ 'fujitsu.cobol, version=7.0.120.0, culture=neutral, publickeytoken=fac0fe3cab973246.' が参照されていますが、現在のデータベースに存在しません。SQL Server では、参照元のアセンブリと同じ場所から参照先のアセンブリを探して自動的に読み込もうとしましたが、操作は失敗しました (理由: 2(指定されたファイルが見つかりません。))。参照先のアセンブリを現在のデータベースに読み込み、要求を再試行してください。

  また、Visual Studio や SQL Server Management Studio などからリモートのサーバに配置しようとした場合は、以下のメッセージが表示されることがあります。

  > このクエリを実行するには、システム メモリが不足しています。

SQL CLR データベースオブジェクトのアクセス時の注意点

COBOL のクライアントプログラムから、データベース機能を使用して SQL CLR アプリケーションをアクセスする際は、以下の点に注意してください。

- NetCOBOL for .NET で開発した SQL CLR データベースオブジェクトを実行した際、以下のメッセージが表示された場合は、SQL CLR 機能が有効になっていません。

  > .NET Framework でのユーザー コードの実行は無効です。`clr enabled' 構成オプションを有効にしてください。

- NetCOBOL for .NET ランタイムが表示するエラーメッセージが英語で表示される場合は、Fujitsu.COBOL.resources.dll を配置先のデータベースに登録してください。Fujitsu.COBOL.resources.dll の登録方法については、ランタイムアセンブリを登録するを参照してください。
- ODBC 接続では、 SQL CLR ストアドプロシージャを呼び出すことはできません。
- SQL CLR データベースオブジェクトをアクセスする場合は、SQLERRD(3)による処理行数の確認は使用できません。

NCOBU00997

Windows Communication Foundation を利用したアプリケーションの開発

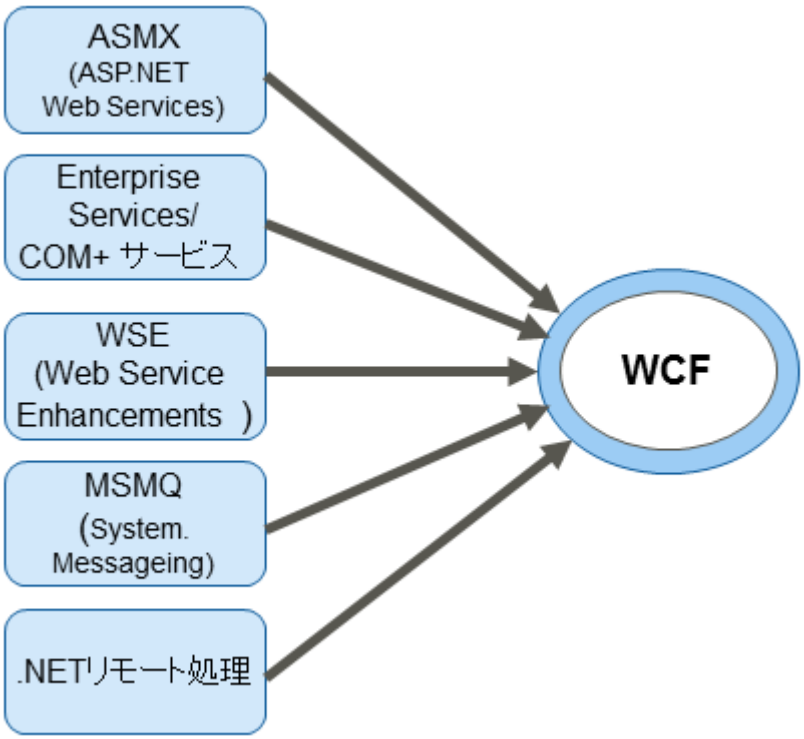
NetCOBOL for .NET では、Windows Communication Foundation(WCF)を利用したアプリケーションを作成することができます。

WCF とは

WCF は.NET Framework 3.0 で追加された新しい分散テクノロジです。

これまで.NET Framework で分散テクノロジを利用する場合、通信要件に応じて、通信要件に合ったコンポーネントを選んで開発する必要がありました。 例えば、ASP.NET XML Web サービス(ASMX)を利用する場合と、.NET リモート処理を利用する場合では、プログラミングモデルがまったく異なっていました。

WCF ではこれらのプログラミングモデルを統一することで、同じプログラミングから、様々な通信方法の分散テクノロジを利用したアプリケーションの開発ができるようになりました。



ここでは、WCF を利用したアプリケーションの開発方法について説明します。

なお、WCF サービスの開発についてはチュートリアルの Windows Communication Foundation サービスの開発 もあわせて参照してください。

このセクションの内容

WCF を利用するための準備

WCF を利用するための条件について説明します。

WCF サービスを作成する

WCF サービスを作成する方法を説明します。

[WCF サービスを利用する]

WCF サービスを利用するクライアントアプリケーションの作成方法について説明します。

[NetCOBOL for .NET で WCF を利用するための注意事項]

NetCOBOL for .NET で WCF サービスを利用したアプリケーションを開発する場合の注意事項について説明します。

関連トピックス

Visual Studio のドキュメント の Windows Communication Foundation

WCF アプリケーションを開発するために必要となる基本的な情報を提供しています。

NCOBU00998

WCF を利用するための準備

開発および運用に必要なコンポーネント

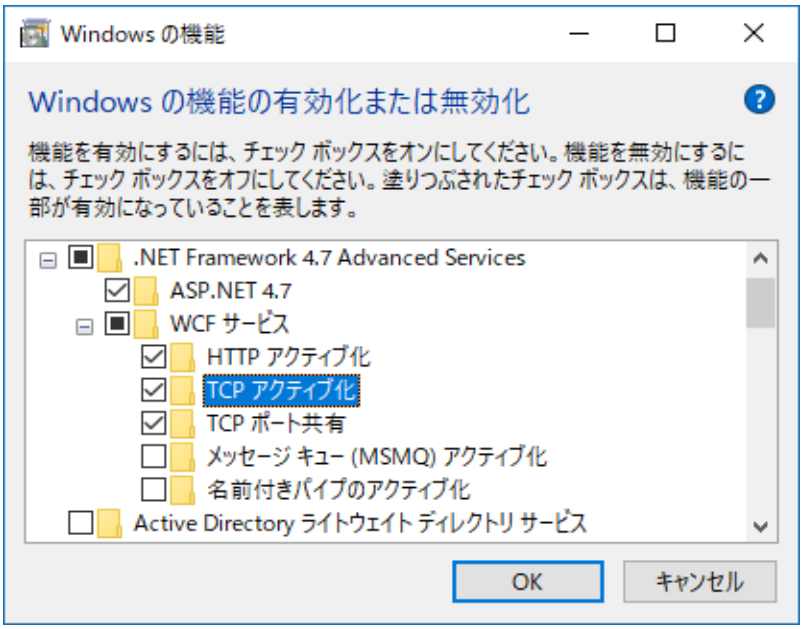
WCF を利用したアプリケーションを開発および運用する場合は、以下のいずれかのコンポーネントが必要です。

- .NET Framework 4
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2
- .NET Framework 4.6
- .NET Framework 4.6.1
- .NET Framework 4.6.2
- .NET Framework 4.7
- .NET Framework 4.7.1

Visual Studio 2017 がインストールされている場合は .NET Framework 4.6.1 は標準でインストールされています。

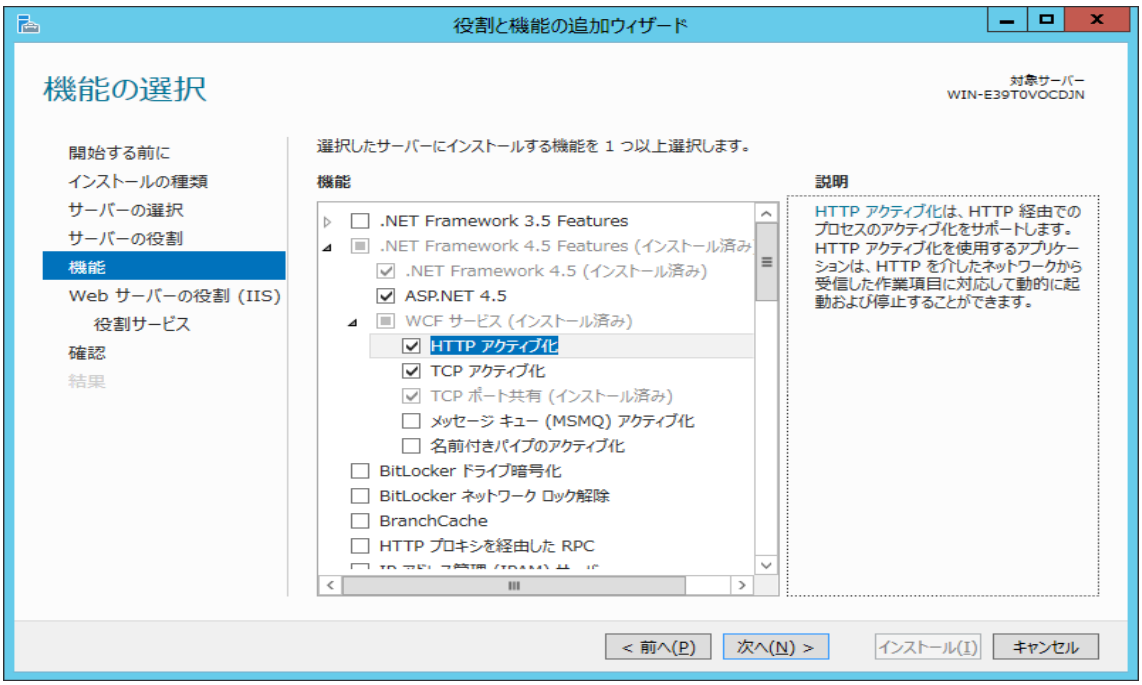
Windows 8.1 以降で WCF を利用するためには[コントロールパネル]-[プログラムと機能]の[Windows の機能の有効化または無効化]で[WCF サービス]を有効化する必要があります。 以下は、Windows 10 の例です。

- .NET Framework 4.7 Advanced Services
  - WCF サービス
    - HTTP アクティブ化
    - TCP アクティブ化



Windows Server 2012 R2 以降では、WCF を利用するためには[サーバー マネージャー]の[役割と機能の追加ウィザード]で[WCF サービス]を有効化する必要があります。 以下は、Windows Server 2012 R2 の例です。

- .NET Framework 4.5 Features
  - .NET Framework 4.5
  - WCF サービス
    - HTTP アクティブ化
    - TCP アクティブ化



[サーバー マネージャー]の[役割の選択]で「アプリケーション サーバー」の役割を追加する場合も、同等の役割サービス(「Windows プロセス アクティブ化サービス サポート」など)の追加が必要となります。

NCOBU00999

WCF サービスを作成する

ここでは、WCF サービスを作成する方法について説明します。

WCF サービスは、以下の手順で作成します。

1. WCF サービスをホストするためのアプリケーションを用意します。
2. どんなメッセージを交換するのかを定義します。(サービスコントラクトを定義する)
3. サービスコントラクトを実装したクラスを定義します。(サービスクラスを作成する)
4. WCF サービスを公開するために必要な情報をエンドポイントとして設定します。(サービスのエンドポイントを設定する)

このセクションの内容

[サービスをホストするためのアプリケーションを用意する]

サービスを公開するために、サービスをホストするアプリケーションを作成します。ここでは、WCF サービスを IIS でホストする方法、Windows アプリケーションのマネージコードからホストする方法、Windows サービスでホストする方法について説明します。

[コントラクトを定義する]

コントラクトは、どんなメッセージを交換するのかを示します。サービスコントラクト、オペレーションコントラクト、データコントラクト、メッセージコントラクトの定義方法について説明します。

[サービスクラスを作成する]

サービスコントラクトはインタフェース定義であるため、そのインタフェースを実装したクラスを定義します。ここでは、サービスコントラクトを実装したクラス定義について説明します。

[サービスのエンドポイントを設定する]

エンドポイントを定義して、WCF サービスを公開するために必要な情報を設定します。エンドポイントは、サービスとクライアントの間でやりとりするメッセージの出入口点となるもので、アドレス、通信手段、コントラクトの情報を持ちます。

サービスをホストするためのアプリケーションを用意する

WCF サービスを公開するためには、サービスをホストする必要があります。

NetCOBOL for .NET では、以下をサポートしています。

- IIS でホストする
- Windows アプリケーションのマネージコードからホストする
- Windows サービスでホストする

それぞれの方法でホストする場合の開発方法について説明します。以下の開発手順に従うことで、アプリケーションの構築に必要なファイル一式を用意できます。

このセクションの内容

Web サイトで WCF サービスをホストする

IIS でホストする Web サイトを作成します。

Windows アプリケーションのマネージコードでホストする

Windows アプリケーションのマネージコードからホストするアプリケーションを作成します。

Windows サービスでホストする

Windows サーバからホストするアプリケーションを作成します。

NCOBU01001

Web サイトで WCF サービスをホストする

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。NetCOBOL for .NET で Web サイトを作成する手順については、 NetCOBOL 製品の技術情報ページ(http://www.fujitsu.com/jp/software/cobol/ の「技術情報」>「注意事項」)を参照してください。

IIS でホストすると、Web ベースのサービスを運用することができます。ここでは Web ベースのサービスを運用するための WCF サービスの開発方法について説明します。

NetCOBOL for .NET で Web ベースの WCF サービスを開発する場合は、 XML Web サービスと同様に Web サイトアプリケーションとして WCF サービスを作成します。

WCF サービスの Web サイトの作成

テンプレートを利用した WCF サービスの作成方法について説明します。

NetCOBOL for .NET では、WCF サービスを IIS でホストするための「WCF サービス」テンプレートを提供しています。このテンプレートを使うことで、 IIS でホストする WCF サービスを開発することができます。

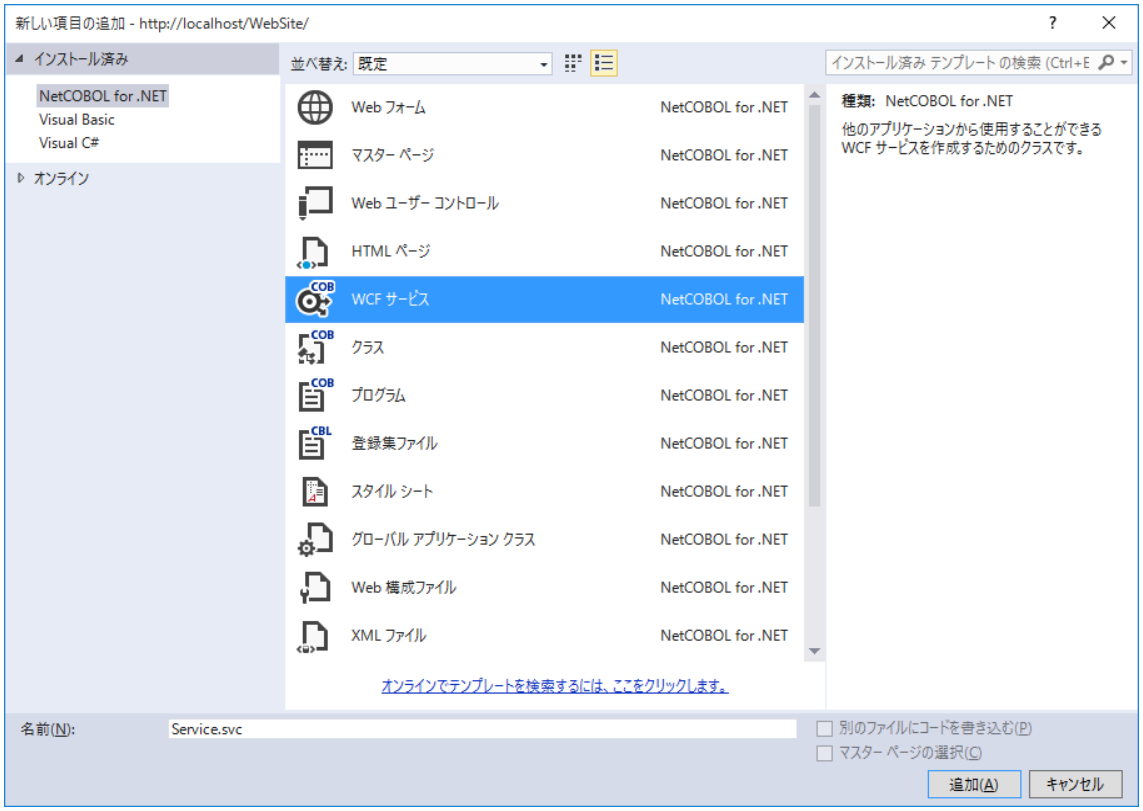
以下に作成手順を示します。

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから[新規作成]-[Web サイト]を選択します。
3. [新しい Web サイト]ダイアログが表示されます。[Web の場所]ボックスは、構築場所(ローカル IIS の URL など)を指定し、左ペインは[NetCOBOL for .NET]を選択します。対象のフレームワークは[.NET Framework 4.7]を選択します。
4. 右ペインで、[WCF サービス]を選択します。
5. [OK]ボタンをクリックすると、アプリケーションの構築に必要なファイル一式が作成されます。

デフォルトでは以下の構成の Web サイトが作成されます。

- [http://localhost/WCFService1/]: Web サイトフォルダ
  - [App_Code]
    - IService.cob: サービスコントラクト(インタフェース定義)のソースコード
    - Service.cob: WCF サービスクラスのソースコード
    - ServiceData.cob: データコントラクト(クラス定義)のソースコード
  - [App_Data]
  - Service.svc: WCF サービスホストファイル
  - Web.config: Web サイトおよび WCF サービスの構成ファイル

なお、既存の Web サイトに WCF サービスを追加したい場合は、 [Web サイト]メニューから[新しい項目の追加]で「WCF サービス」を選択して、新しい WCF サービスを追加します。



空の Web サイトに「WCF サービス」を追加した場合は、以下のような構成になります。

- [http://localhost/WebSite]: Web サイトフォルダ
  - [App_Code]
    - IService.cob: サービスコントラクト(インタフェース定義)のソースコード
    - Service.cob: WCF サービスクラスのソースコード
    - ServiceData.cob: データコントラクト(クラス定義)のソースコード
  - Service.svc: WCF サービスホストファイル
  - Web.Config: WCF サービスの構成が設定された Web 構成ファイル

COBOL コードの配置

WCF サービスホストファイル(以降では.svc ファイルと記述します)の@ServiceHost ディレクティブで、WCF サービスを実装するコード(WCF サービスクラス)を関連づけます。

以下はテンプレートから作成される.svc ファイルの例です。

```
<%@ ServiceHost Language="Fujitsu.COBOL" Debug="true"
           CodeBehind="~/App_Code/WCFService.cob" Service="WCFService" %>
```

この例では、@ServiceHost ディレクティブの CodeBehind 属性に、WCF サービスを実装する COBOL ソースファイル名([App_Code]フォルダ配下の WCFService.cob)を指定しています。

これは、XML Web サービスの@WebService ディレクティブに COBOL コードを配置する場合とよく似ています。@ServiceHost ディレクティブも@WebService ディレクティブと同様に、COBOL コードを個別のアセンブリ(.dll)に含めて指定することもできます。

```
<%@ServiceHost Language="Fujitsu.COBOL" Debug="true" Service="WCFService" %>
```

この場合、Service 属性に指定した WCFServiceSite クラスを含むアセンブリは、仮想ディレクトリの Bin フォルダに配置する必要があります。

デバッグ

IIS でホストされた WCF サービスは XML Web サービスと同様の方法でデバッグすることができます。

XML Web サービスのデバッグ方法については、XML Web サービスの開発のデバッグを参照してください。

Windows プロセス アクティブ化サービスでのホスティング

IIS 7.0 以降では、 Windows プロセス アクティブ化サービスを使用して HTTP 以外のプロトコルでの WCF サービスのホストが可能です。

Windows プロセス アクティブ化サービスについては、Visual Studio のドキュメント の Windows プロセス アクティブ化サービスでのホスティングを参照して下さい。

関連するトピックス

Visual Studio のドキュメント のインターネット インフォメーション サービスでのホスティング

IIS でホストする場合の様々な情報を提供しています。

Windows アプリケーションのマネージコードでホストする

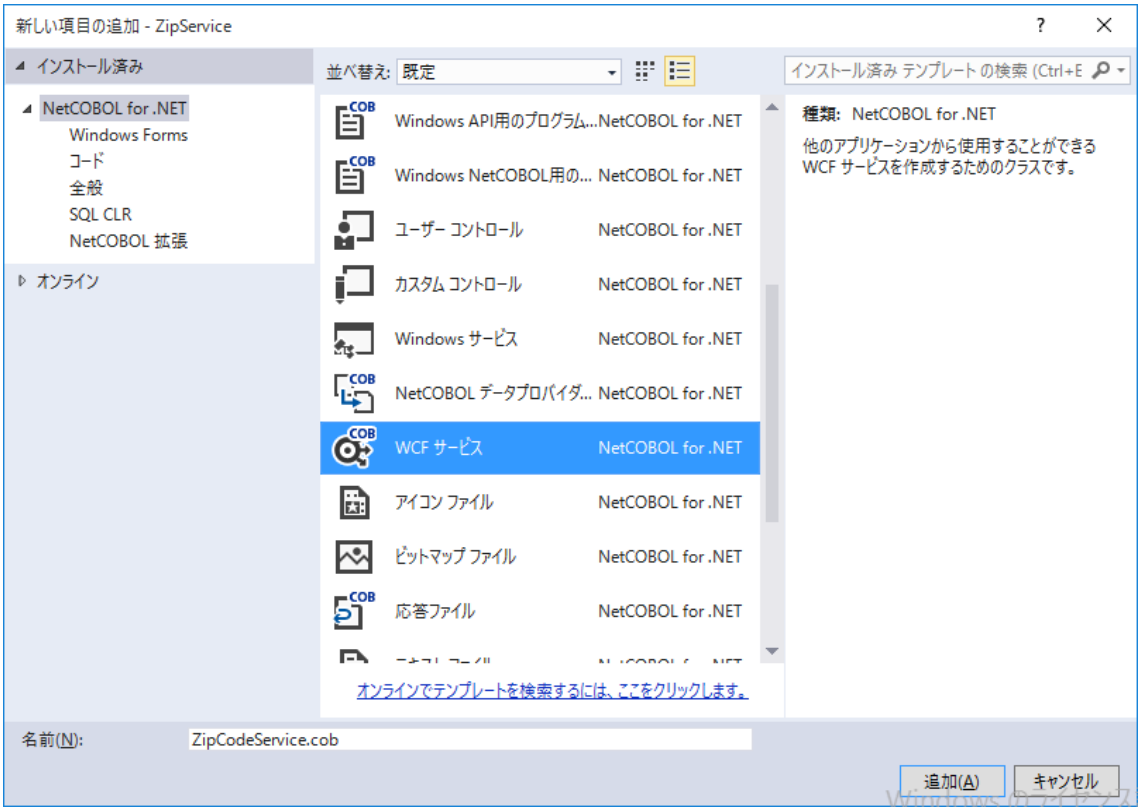
Windows アプリケーションやコンソールアプリケーションなどのマネージコードから WCF サービスをホストすることができます。

この場合は、WCF サービスをホストしたいアプリケーションのメインプログラムなどで、System.ServiceModel.ServiceHost クラスをインスタンス化し、Open メソッドを呼び出すことで関連づけられた WCF サービスを開始し、指定されたエンドポイントに従いクライアントからの要求を待ちます。

NetCOBOL for .NET では、サービスクラス、サービスコントラクトおよびサービスホストプログラムを既存のプロジェクトに追加するためのテンプレートを用意しています。追加する手順は以下のとおりです。

1. Visual Studio を起動して、既存のプロジェクトを開きます。
2. プロジェクトの[プロパティページ]を開き、[アプリケーション]ページで[対象のフレームワーク]として、[.NET Framework 4.7]が選択されていることを確認します。
3. ソリューションエクスプローラーの[ソースファイル]にカーソルを合わせてから、Visual Studio の[プロジェクト]メニューから[新しい項目の追加]を選択します。
4. [新しい項目の追加]ダイアログボックスが表示されます。
5. 右ペインで、[WCF サービス]を選択します。
6. [名前]に、サービスクラス名を入力します。

   例えば、“ZipCodeService”という名前のサービスクラスを作成したい場合は、“ZipCodeService.cob”と入力します。



7. [追加]ボタンをクリックすると、アプリケーションの構築に必要なファイル一式(サービスクラス、サービスコントラクト、サービスホストプログラムなど)が追加されます。

   この例では、以下のファイルが追加されます。

- ZipService: プロジェクト名
  - [Properties]
    - AssemblyInfo.cob: アセンブリ情報
  - [参照]
    - System
    - System.Core
    - System.Data
    - System.Data.DataSetExtensions
    - System.ServiceModel: WCF 用アセンブリ
    - System.Runtime.Serialization: WCF 用アセンブリ
    - System.Xml
    - System.Xml.Linq
  - [ソースコード]
    - IZipCodeService.cob: サービスコントラクト(インタフェース定義)のソースコード
    - Main.cob: メインプログラム
    - ZipCodeService.cob: WCF サービスクラスのソースコード
    - ZipCodeServiceHost.cob: サービスホストプログラムのソースコード
    - ZipCodeServiceData.cob: データコントラクト(クラス定義)のソースコード
  - [登録集ファイル]
  - App.config: WCF サービスの構成が設定されたアプリケーション構成ファイル

NetCOBOL for .NET では、サービスホストプログラムを簡単に記述できるように、サービスコントラクトに対応したカスタムのサービスホストクラスを展開します。

以下は、テンプレートから展開されたサービスホストプログラム(ZipCodeServiceHost.cob)のソースコードをベースに記述したコード例です。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-HOST AS "ZipService.ZipServiceHost" INHERITS CLASS-SERVICEHOST.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-SERVICEHOST AS "System.ServiceModel.ServiceHost"
   CLASS CLASS-ZipCodeSerivce AS "ZipServer.ZipCodeService"
   CLASS CLASS-URI AS "System.Uri"
   CLASS CLASS-TYPE AS "System.Type"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. NEW.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 URI OBJECT REFERENCE CLASS-URI.
01 SERVICE-TYPE OBJECT REFERENCE CLASS-TYPE.
PROCEDURE DIVISION.
   *> デフォルトのエンドポイントを指定する
   SET URI TO CLASS-URI :: "NEW"("http://localhost:8080/ZipCodeService").
   SET SERVICE-TYPE TO TYPE OF CLASS-ZipCodeSerivce.
   INVOKE SUPER "NEW" USING SERVICE-TYPE URI
```

```
    END METHOD NEW.

    END OBJECT.

    END CLASS CLASS-HOST.
```

このサービスホストクラス(ZipService.ZipServiceHost)をメインプログラムから呼び出す場合は以下のように記述することができます。

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN AS "ZipServer.Main".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-SERVICEHOST AS "ZipService.ZipServiceHost"
   .
DATA DIVISION.
WORKING-STORAGE SECTION.
01 SERVICE-HOST OBJECT REFERENCE CLASS-SERVICEHOST.
01 A PIC X.
PROCEDURE DIVISION.
   TRY
      *> サービスホストの初期化
      SET SERVICE-HOST TO CLASS-SERVICEHOST :: "NEW"
      *> サービスホストの開始
      INVOKE SERVICE-HOST "Open"
      DISPLAY "サービスを終了するには何れかの文字を入力して Enter を押してください。"
      ACCEPT A
   FINALLY
      *> サービスホストの終了
      IF SERVICE-HOST NOT = NULL
         INVOKE SERVICE-HOST "Close"
      END-IF
   END-TRY.
END PROGRAM MAIN.
```

サービスクラス、サービスコントラクトおよびサービスホストプログラムを既存のプロジェクトに追加する場合、WCF サービス ライブラリを利用することもできます。WCF サービス ライブラリの使い方については、Windows サービスでホストするで詳しく説明します。

関連するトピックス

Visual Studio のドキュメント のマネージ アプリケーションのホスト

マネージアプリケーションでホストする場合の関連情報を提供しています。

構造化例外処理(TRY 文)

NetCOBOL for .NET で try-catch-finally の各ブロックを定義する場合の TRY 文について説明しています。

NCOBU01003

Windows サービスでホストする

Windows サービスで WCF サービスをホストするアプリケーションを作成することができます。

ここでは、WCF サービス ライブラリを組み合わせた Windows サービスを作成する例を説明します。

WCF サービスを開始するには、System.ServiceModel.ServiceHost クラスをインスタンス化して Open メソッドを呼び出し、Close メソッドを呼び出すことで WCF サービスを終了させます。

これは、Windows サービスのベースクラス(System.ServiceProcess.ServiceBase クラス) からオーバーライドされた OnStart メソッドと OnStop メソッドに記述しやすい構造となっているため、比較的簡単に Windows サービスに組み込むことができます。

Windows サービスアプリケーションについての詳細は Visual Studio のドキュメントの Windows サービス アプリケーションの開発を参照してください。

WCF サービスは、HTTP 以外のトランスポートプロトコルを使用することができるため、Windows サービスでもサービスを提供することができます。

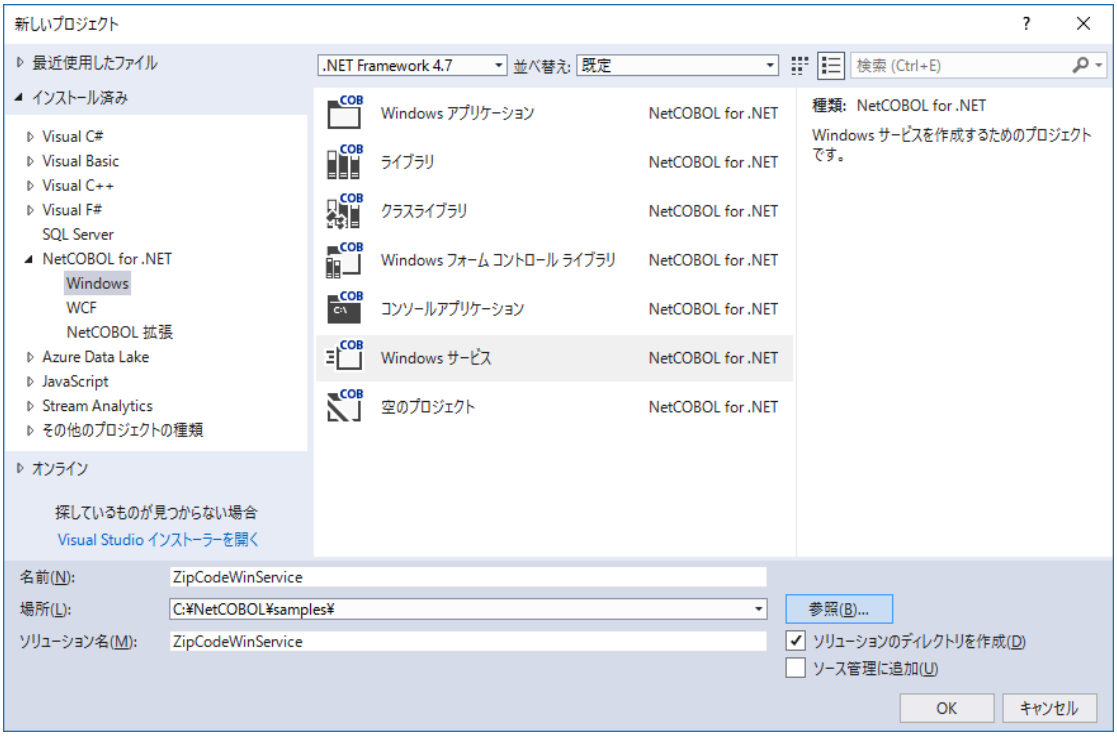
以下に、アプリケーションの作成手順を示します。

Windows サービスプロジェクトの作成

まず、最初に Windows サービスのプロジェクトを作成します。

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから[新規作成]-[プロジェクト]を選択します。
3. [新しい プロジェクト]ダイアログが表示されます。
4. 対象のフレームワークとして、 [.NET Framework 4.7] を選択します。
5. 左ペインで、[NetCOBOL for .NET]-[Windows]を選択します。
6. 右ペインで、[Windows サービス]を選択します。
7. 任意の[名前]、[場所]、[ソリューション名]を指定します。

   以下の例では、“ZipCodeWinService”という名前の プロジェクトを作成しています。



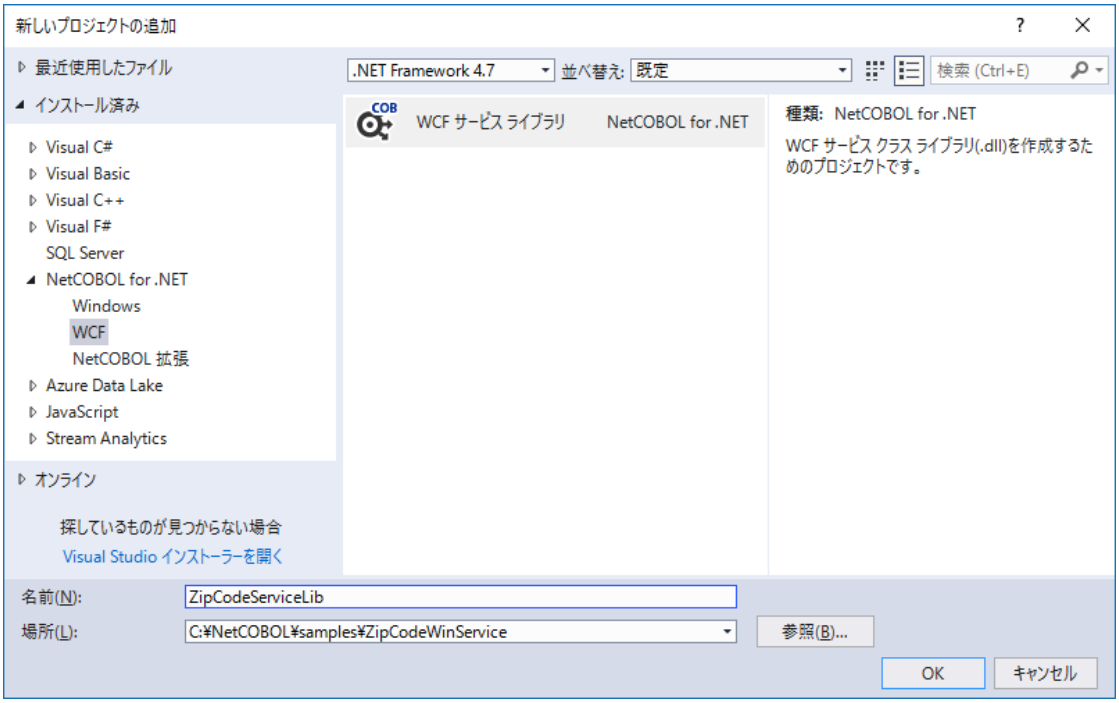
8. [OK]ボタンをクリックすると、アプリケーションの構築に必要なファイル一式が作成されます。

WCF サービス ライブラリの作成

次に、WCF サービス ライブラリをソリューションに追加します。

1. [ファイル]メニューから[追加]-[新しいプロジェクト]を選択すると、 [新しいプロジェクトの追加]ダイアログが表示されます。

2. 対象のフレームワークに [.NET Framework 4.7]が選択されていることを確認し、左ペインで、[NetCOBOL for .NET]-[WCF]を選択します。



3. [WCF サービス ライブラリ]を選択します。ここでは、“ZipCodeServiceLib”という名前の プロジェク

トを作成します。

4. [OK]ボタンをクリックすると、WCF サービス ライブラリのプロジェクトがソリューションに追加されます。

5. 追加されたファイルのファイル名をそれぞれ変更し、各コントラクトを定義します。 ここでは、“ZipCodeService”という名前のサービスを定義するため、次のように各ファイルの名前を変更します。

   - “IService1.cob”のファイル名を“IZipCodeService.cob”に変更します。サービスコントラクトのインタフェース名は、“ZipCodeServiceLib.IZipCodeSerivce”です。
   - “Service1Data.cob”のファイル名を“ZipAddress.cob”に変更します。データコントラクトのクラス名は、“ZipCodeServiceLib.ZipAddress”です。
   - “Service1.cob”のファイル名を“ZipCodeService.cob”に変更します。サービスクラスのクラス名は、“ZipCodeServiceLib.ZipCodeService”です。
   - “Service1Host.cob”のファイル名を“ZipCodeServiceHost.cob”に変更します。サービスホストのクラス名は“ZipCodeServiceLib.ZipCodeServiceHost”です。
   - 変更したサービスコントラクトのインタフェース名やサービスのクラス名にあわせて、“App.config”の内容を変更します。以下は、変更した例です。

```
<?xml version="1.0" encoding="utf-8"?><configuration>
  <!-- -->
  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" />
  </system.web>
  <!-- サービス ライブラリ プロジェクトの展開時に、構成ファイルの内容をホストの app.config ファイルに追加する
  必要があります。System.Configuration は、ライブラリの構成ファイルをサポートしていません。 -->
  <system.serviceModel>
    <services>
      <service name="ZipCodeServiceLib.ZipCodeService"">
        <host>
          <baseAddresses>
            <add
baseAddress="http://localhost:8733/Design_Time_Addresses/ZipCodeServiceLib/ZipCodeService/" />
          </baseAddresses>
        </host>
        <!-- Service Endpoints -->
        <!-- アドレスは、完全修飾でない限り、上で指定されたベース アドレスに相対的なものとなります -->
        <endpoint address="" binding="basicHttpBinding"
                  contract="ZipCodeServiceLib.IZipCodeService">
          <!--
              展開時に、次の ID 要素を削除または置換して、展開されたサービスが
              実行されている ID が反映されるようにする必要があります。削除されると、WCF は適切な ID を自動的に
              推測します。
          -->
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <!-- Metadata Endpoints -->
        <!-- Metadata Exchange エンドポイントは、サービスが、そのサービス自体をクライアントに記述するために使
用されます。 -->
        <!-- このエンドポイントは、セキュリティで保護されたバインドを使用していないため、展開する前にセキュリティ
で保護するか、削除する必要があります -->
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- メタデータ情報の開示を避けるには、
          展開する前に下の値を false に設定します -->
          <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True" />
          <!-- デバッグ目的で障害発生時の例外の詳細を受け取るには、
          下の値を true に設定します。例外情報の開示を避けるには、
          展開する前に false に設定します -->
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
```

```
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>

  <!-- -->
</configuration>
```

追加されたプロジェクトは以下の構成となります。

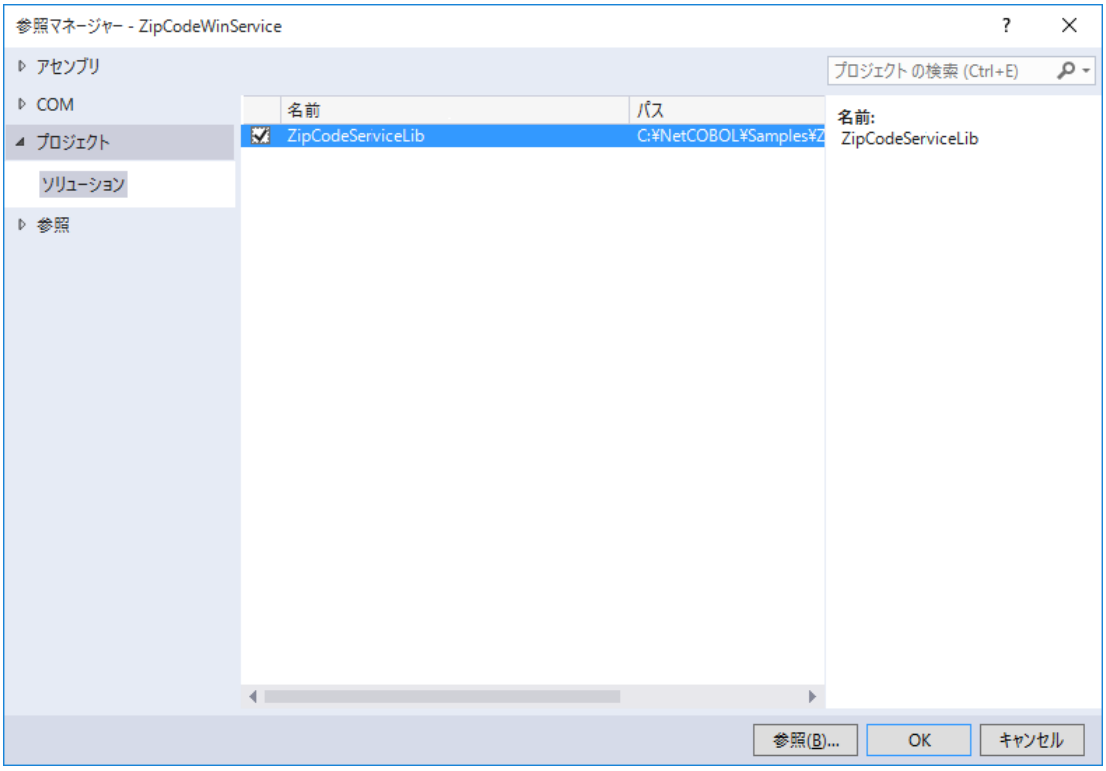
- ZipCodeServiceLib: プロジェクト名
  - [Properties]
    - AssemblyInfo.cob: アセンブリ情報
  - [参照]
  - [ソースコード]
    - IZipCodeService.cob: サービスコントラクト(インタフェース定義)のソースコード
    - ZipCodeService.cob: WCF サービスクラスのソースコード
    - ZipAddress.cob: データコントラクト(クラス定義)のソースコード
    - ZipServiceHost.cob: サービスホストプログラムのソースコード
  - [登録集ファイル]
  - App.config: WCF サービスの構成を定義する

WCF サービス ライブラリプロジェクトでは、WCF サービス ホスト (WcfSvcHost.exe)を使用して WCF サービス ライブラリをデバッグ実行することができます。 このとき、WCF サービス ライブラリプロジェクトに追加されたアプリケーション構成ファイル(“App.config”)が使用されます。 なお、WCF サービス ライブラリを他の実行可能なアプリケーションでホストする場合は、このアプリケーション構成ファイルは無効となります。この場合は、ホストアプリケーション側のアプリケーション構成ファイルで WCF サービス ライブラリの構成を再構成する必要があります。 WCF サービスの構成設定については、サービスのエンドポイントを設定するを参照して下さい。

WCF サービス ライブラリの参照

続いて、“ZipCodeWinService” プロジェクトから、“ZipCodeServiceLib” を WCF サービス ライブラリのプロジェクトを参照するように設定します。

1. ソリューションエクスプローラーで、“ZipCodeWinService” プロジェクトを選択します。
2. [プロジェクト]メニューから[参照の追加]を選択し、[参照マネージャー]ダイアログを表示します。
3. [プロジェクト]タブを表示し、リストから“ZipCodeServiceLib”を選んで[OK]ボタンをクリックします。



WCF サービス ライブラリの呼び出し

Windows サービスコンポーネントのソースコード(Service1.cob)を開き、“ZipCodeServiceLib”の“ZipCodeServiceHost”クラスに定義されているコードを呼び出します。ここでは、“OnStart”メソッドに WCF サービスの開始を示すコード、“OnStop”メソッドに WCF サービスの終了を示すコードをそれぞれ記述します。

以下のサンプルコードは、Windows サービスの起動時に、WCF サービスホストクラスの“ZipServer.ZipServiceHost”をインスタンス化してサービスを開始し、Windows サービスの停止時にサービスを終了させる例です。

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "ZipCodeWinService.Service1"
   INHERITS CLASS-SERVICEBASE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-SERVICEBASE AS "System.ServiceProcess.ServiceBase"
   CLASS CLASS-STRING AS "System.String"
   CLASS ARRAY-STRING AS "System.String[]"
   PROPERTY PROP-SERVICENAME AS "ServiceName"
   CLASS CLASS-SERVICEHOST AS "ZipCodeServiceLib.ZipCodeServiceHost"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
01 SERVICE-HOST OBJECT REFERENCE CLASS-SERVICEHOST.
PROCEDURE DIVISION.

METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
PROCEDURE DIVISION USING BY VALUE disposing.
```

```
        IF disposing NOT = B"0" AND components NOT = NULL THEN
          INVOKE components "Dispose"
        END-IF.
        INVOKE SUPER "Dispose" USING disposing.
    END METHOD DISPOSE.

    .....

    METHOD-ID. NEW.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    PROCEDURE DIVISION.
        INVOKE SELF "InitializeComponent".
    END METHOD NEW.

    METHOD-ID. ONSTART AS "OnStart" OVERRIDE PROTECTED.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    LINKAGE SECTION.
    01 ARGS OBJECT REFERENCE ARRAY-STRING.
    PROCEDURE DIVISION USING BY VALUE ARGS.
        *> サービスを開始します。
        SET SERVICE-HOST TO CLASS-SERVICEHOST :: "NEW".
         INVOKE SERVICE-HOST "Open".

    END METHOD ONSTART.

    METHOD-ID. ONSTOP AS "OnStop" OVERRIDE PROTECTED.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    PROCEDURE DIVISION.
        *> サービスを停止します。
        INVOKE SERVICE-HOST "Close"
    END METHOD ONSTOP.

    END OBJECT.
    END CLASS CLASS-THIS.
```

Windows サービスの詳細な開発方法については、Visual Studio のドキュメントの方法 : Windows サービスを作成するを参照してください。

Windows サービスのインストール

Windows サービスを起動するためには、システムに登録する必要があります。

詳細については Visual Studio のドキュメントを参照してください。

- 方法 : サービス アプリケーションにインストーラを追加する
- 方法 : サービスをインストールおよびアンインストールする

Windows サービスのデバッグ

Windows サービスをデバッグする場合は、サービスを開始しプロセスにデバッガーをアタッチする必要があります。

詳細については Visual Studio のドキュメントを参照してください。

- 方法 : サービスを開始する
- 方法 : Windows サービス アプリケーションをデバッグする

関連するトピックス

Visual Studio のドキュメント の Windows サービス アプリケーションのホスト

Windows サービスでホストする場合の関連情報を提供しています。

コントラクトを定義する

コントラクトとは、以下のサービスなどが何であるかを示すものです。

- 外に公開するサービス(サービスコントラクト=インタフェース)
- 利用するサービス(オペレーションコントラクト=メソッド)
- 交換するメッセージ(データコントラクト/メッセージコントラクト=データクラス)

ここでは、 WCF サービスとして公開するインタフェースやデータクラスをコントラクトとして定義する方法を説明します。

このセクションの内容

サービスコントラクトを作成する

WCF サービスとして公開するインタフェース(サービスコントラクト)やメソッド(オペレーションコントラクト)を定義します。ここでは、WCF サービスのテンプレートで追加されるサービスコントラクトのインタフェース定義を使って作成します。

データコントラクトを作成する

WCF サービスで扱うメッセージデータのクラスを定義します。ここでは、WCF サービスのテンプレートで追加されるデータコントラクトのクラス定義を使って作成します。

関連するトピックス

Visual Studio のドキュメント のサービスの設計と実装

コントラクトの定義に関する情報を提供しています。

NCOBU01005

サービスコントラクトを作成する

WCF サービスとして公開するインタフェース(サービスコントラクト)やメソッド(オペレーションコントラクト)を定義します。 ここでは、WCF サービスのテンプレートで追加されるサービスコントラクトのインタフェース定義を使って作成します。

サービスコントラクトはインタフェース定義を記述し、サービスコントラクトであることを示すために、System.ServiceModel.ServiceContractAttribute のカスタム属性を付けます。

また、 WCF サービスで公開するメソッド定義にはオペレーションコントラクトを示す System.ServiceModel.OperationContractAttribute のカスタム属性を付けます。

WCF サービスをテンプレートから作成する場合は、これらの属性はあらかじめ記述されているため、提供したいサービスの形にあわせてコントラクトを編集します。

以下にサービスコントラクトの例を示します。

```
IDENTIFICATION  DIVISION.
INTERFACE-ID. INTERFACE-1 AS "ZipCodeServiceLib.IZipCodeService"
              CUSTOM-ATTRIBUTE IS CA-SERVICECONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-SERVICECONTRACT CLASS CLASS-SERVICECONTRACT
     CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT CLASS CLASS-OPERATIONCONTRACT
    .
REPOSITORY.
    CLASS CLASS-STRING AS "System.String"
    CLASS CLASS-SERVICECONTRACT AS "System.ServiceModel.ServiceContractAttribute"
   CLASS CLASS-OPERATIONCONTRACT AS
          "System.ServiceModel.OperationContractAttribute"
    CLASS CLASS-ZipAddress AS "ZipCodeServiceLib.ZipAddress"
    .

PROCEDURE DIVISION.

METHOD-ID. GETADDRESS AS "GetAddress" CUSTOM-ATTRIBUTE IS CA-OPERATIONCONTRACT.
DATA DIVISION.
LINKAGE SECTION.
01 ZIPCODE OBJECT REFERENCE CLASS-STRING.
01 RETVAL OBJECT REFERENCE CLASS-ZipAddress.
PROCEDURE DIVISION USING BY VALUE ZIPCODE RETURNING RETVAL.

END METHOD GETADDRESS.

END INTERFACE INTERFACE-1.
```

上記のサンプルコードでは、サービスコントラクトとして ZipCodeServiceLib.IZipCodeService インタフェースを定義しています。

データコントラクトを作成する

WCF サービスで扱うメッセージデータのクラスを定義します。 ここでは、WCF サービスのテンプレートで追加されるデータコントラクトのクラス定義を使って作成します。

データコントラクトの種類

サービスコントラクトを作成するのサンプルコードではオペレーションコントラクトとなる GetAddress メソッドは、パラメタと戻り値の型にそれぞれ System.String 型と ZipCodeServiceLib.ZipAddress クラスを使用していますが、これらのデータ型はサービス側とクライアント側の間でやりとりするメッセージデータとして利用できなければなりません。

WCF サービスによる通信では、オペレーションコントラクトのパラメタや戻り値は、SOAP メッセージとして扱われるため、XML Web サービスや.NET リモート処理と同様に、 WCF サービスで扱うデータはシリアライズできなければなりません。

WCF サービスのメッセージとして利用できるデータ型には以下があります。

型の種類	説明
CLR 基本型	CLR 基本型は、 シリアライズできるためそのまま利用することができます。
データコントラクト	System.Runtime.Serialization.DataContractAttribute のカスタム属性が付いたユーザ定義型(クラス)です。 データコントラクトのメンバーとして公開したいプロパティやフィールドには、System.Runtime.Serialization.DataMemberAttribute のカスタム属性を付けます。 ただし、DataMemberAttribute 属性が指定されない場合は、 パブリックメンバーが自動的にデータコントラクトのメンバーとして公開されます。
メッセージコントラクト	CLR 基本型やデータコントラクトは、SOAP メッセージのヘッダ部分が省略された形式です。 SOAP メッセージのヘッダとボディを意識した形式にしたい場合は、 ユーザ定義クラスに System.ServiceModel.MessageContractAttribute のカスタム属性を指定します。 そして、メッセージヘッダとなるデータメンバーには、System.ServiceModel.MessageHeaderAttribute のカスタム属性を付け、 メッセージボディとなるデータメンバーには System.ServiceModel.MessageBodyMemberAttribute のカスタム属性を付けます。 メッセージのヘッダおよびボディのそれぞれのデータ型としてデータコントラクトを併用することができます。
型なしメッセージ	パラメタの型として System.ServiceModel.Channels.Message クラスを使用することで、素の SOAP メッセージをそのまま XML 形式で扱うことができます。
その他のシリアライズ可能なデータ	XML Web サービスや.NET リモート処理で使用した、 以下のいずれかの条件を満たした従来のシリアライズ可能なデータをデータコントラクトとして使用することができます。 ・ System.SerializableAttribute のカスタム属性が付けられたクラス ・ System.SerializableAttribute のカスタム属性が付けられたクラスを使用し、System.Xml.Serialization.XmlSerializer を使用したクラス ・ System.Xml.Serialization.IXmlSerializable インタフェースを実装したクラス

ユーザ定義のデータコントラクトを作成する

CLR 基本データ型である System.String 型は、シリアライズできるためそのまま利用できます。ユーザ定義クラスである ZipCodeServiceLib.ZipAddress クラスは、そのままではシリアライズすることができません。この場合は、ZipCodeServiceLib.ZipAddress クラスをシリアライズできるようにする必要があります。

ユーザ定義クラスをシリアライズできるようにする場合は、クラス定義に従来の System.SerializableAttribute のカスタム属性を指定することもできますが、WCF では、データコントラクトとして示すことが一般的です。

この場合は、クラス定義に System.Runtime.Serialization.DataContractAttribute のカスタム属性を付けます。更に、サービスで公開する必要があるプロパティやフィールドなどの各データメンバーには、System.Runtime.Serialization.DataMemberAttribute のカスタム属性を付けます。以下に ZipCodeServiceLib.ZipAddress をデータコントラクトとして示すサンプルコードを示します。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-1 AS "ZipCodeServiceLib.ZipAddress" CUSTOM-ATTRIBUTE IS CA-DATACONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-DATACONTRACT CLASS CLASS-DATACONTRACT
    CUSTOM-ATTRIBUTE CA-DATAMEMBER CLASS CLASS-DATAMEMBER
    .
REPOSITORY.
    CLASS CLASS-STRING AS "System.String"
    CLASS CLASS-DATACONTRACT AS "System.Runtime.Serialization.DataContractAttribute"
    CLASS CLASS-DATAMEMBER AS "System.Runtime.Serialization.DataMemberAttribute"
    .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ZipCode OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER.
01 PrefectureName OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER.
01 CtiyName OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER.
01 AreaName OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER.
END OBJECT.

END CLASS CLASS-1.
```

WCF サービスをテンプレートから作成する場合は、これらの属性はあらかじめ記述されているため、提供したいサービスの形にあわせてコントラクトを編集してください。

関連トピックス

Visual Studio のドキュメント のデータ転送とシリアル化

データ処理に関する情報を提供しています。

カスタム属性を使う

NetCOBOL for .NET でのカスタム属性の使い方について説明しています。

サービスクラスを作成する

サービスコントラクトはインタフェース定義であるため、サービスコントラクトを実装したクラスを定義してサービスの実態を作成する必要があります。 サービスクラスはサービスをホストするプログラムに指定することによってサービスとして公開することができます。

以下のサンプルコードでは、サービスコントラクトを作成するで作成した、ZipCodeServiceLib.IZipCodeServiceインタフェース定義を実装したクラスを定義しています。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-1 AS "ZipCodeServiceLib.ZipCodeService".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
    CLASS CLASS-STRING AS "System.String"
    INTERFACE INTERFACE-IZipCodeService AS "ZipCodeServiceLib.IZipCodeService"
    CLASS CLASS-ZipAddress AS "ZipCodeServiceLib.ZipAddress"

    PROPERTY PROP-ZIPCODE AS "ZipCode"
    PROPERTY PROP-PREFECTURENAME AS "PrefectureName"
    PROPERTY PROP-CTIYNAME AS "CtiyName"
    PROPERTY PROP-AREANAME AS "AreaName".

OBJECT. IMPLEMENTS INTERFACE-IZipCodeService.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. GETADDRESS AS "GetAddress".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 郵便番号データ.
  02 郵便番号   PIC X(7).
  02 都道府県名 PIC N(100).
  02 市区町村名 PIC N(100).
  02 町域名     PIC N(100).
LINKAGE SECTION.
01 ZIPCODE OBJECT REFERENCE CLASS-STRING.
01 RETVAL OBJECT REFERENCE CLASS-ZipAddress.
PROCEDURE DIVISION USING BY VALUE ZIPCODE RETURNING RETVAL.
    SET RETVAL TO CLASS-ZipAddress::"NEW".

    SET 郵便番号 TO ZIPCODE.
    CALL "ZipCodeServiceLib.GetZipAddress" USING 郵便番号データ.

    SET PROP-ZIPCODE OF RETVAL TO 郵便番号.
    SET PROP-PREFECTURENAME OF RETVAL TO 都道府県名.
    SET PROP-CTIYNAME OF RETVAL TO 市区町村名.
    SET PROP-AREANAME OF RETVAL TO 町域名.

END METHOD GETADDRESS.

END OBJECT.

END CLASS CLASS-1.
```

NCOBU01008

サービスのエンドポイントを設定する

WCF サービスは、クライアントとサービスが通信を行い、メッセージを交換することで分散アプリケーションが構成されていきます。このメッセージの出入り口点を、エンドポイント(Endpoint)と呼び、このエンドポイントの定義が重要になります。エンドポイントは、以下の“ABC”の概念によって定義されます。“ABC”とは、A:Address(アドレス)、 B:Binding(バインディング)、C:Contract(コントラクト)の略称です。

アドレス(Addess)

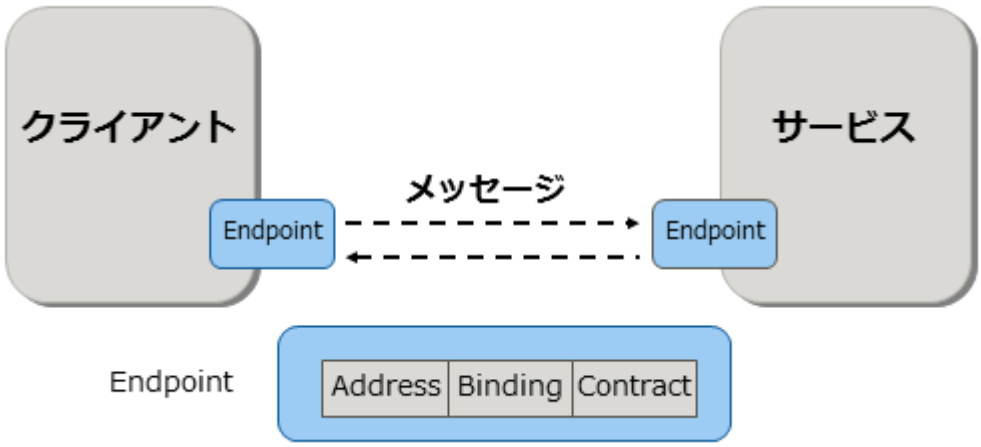
WCF サービスを公開するアドレス(URL)を示します。

バインディング(Binding)

WCF サービスをどのトランスポート・プロトコルで通信するのかを示します。

コントラクト(Contract)

WCF サービスのサービスコントラクトを示します。



WCF ではこの“ABC”の概念により、サービスとクライアントの双方のエンドポイントを設定することでサービスとクライアントの間の通信が可能になります。

エンドポイントは以下の方法で設定することができます。

- 構成ファイル(.config ファイル)に記述する
- サービスホストプログラムに記述する

構成ファイルに記述する

以下に構成ファイルにエンドポイントを定義する例を示します。

```
<configuration>
  <!-- ..... -->
  <system.serviceModel>
    <services>
      <service name="ZipCodeServiceLib.ZipCodeService"
               behaviorConfiguration="ZipCodeServiceLib.ZipCodeServiceBehavior" >
        <endpoint contract="ZipCodeServiceLib.IZipCodeService" binding="basicHttpBinding" />
        <endpoint contract="ZipCodeServiceLib.IZipCodeService" binding="netTcpBinding"
                  address="net.tcp://localhost:8081/ZipCodeService" />
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="ZipCodeServiceLib.ZipCodeServiceBehavior" >
          <serviceDebug includeExceptionDetailInFaults="true" />
          <serviceMetadata httpGetEnabled="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

構成ファイルは、テキストエディターまたは Visual Studio 2017 や Microsoft Windows SDK に同梱されている構成エディター ツール (SvcConfigEditor.exe)を使用して編集できます。

サービスホストプログラムに記述する

エンドポイントの指定はサービスホストプログラムに記述することができます。 以下にその例を示します。

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. MAIN AS "ZipServer.Main".
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
          CLASS CLASS-SERVICEHOST AS "System.ServiceModel.ServiceHost"
          CLASS CLASS-ZipCodeSerivce AS "ZipCodeServiceLib.ZipCodeService"
          CLASS CLASS-URI AS "System.Uri"
          CLASS CLASS-TYPE AS "System.Type"
          CLASS CLASS-NETTCPBINDING AS "System.ServiceModel.NetTcpBinding"
          .
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 SERVICE-HOST OBJECT REFERENCE CLASS-SERVICEHOST.
      01 URI OBJECT REFERENCE CLASS-URI.
      01 SERVICE-TYPE OBJECT REFERENCE CLASS-TYPE.
      01 BINDING OBJECT REFERENCE CLASS-NETTCPBINDING.
      01 A PIC X.
      PROCEDURE DIVISION.
          TRY
              *> デフォルトのエンドポイントを ServiceHost のコンストラクタに指定
              SET URI TO CLASS-URI::"NEW"("http://localhost:8080/ZipCodeService")
               SET SERVICE-TYPE TO TYPE OF CLASS-ZipCodeSerivce
               SET SERVICE-HOST TO CLASS-SERVICEHOST::"NEW"(SERVICE-TYPE URI)

              *> エンドポイントを追加する
              SET URI TO CLASS-URI::"NEW"("net.tcp://localhost:8081/ZipCodeService")
               SET BINDING TO CLASS-NETTCPBINDING::"NEW"
               INVOKE SERVICE-HOST "AddServiceEndpoint" USING SERVICE-TYPE BINDING URI

              INVOKE SERVICE-HOST "Open"
              DISPLAY "サービスを終了するには何れかの文字を入力して Enter を押してください。"
              ACCEPT A
          FINALLY
              IF SERVICE-HOST NOT = NULL
                  INVOKE SERVICE-HOST "Close"
              END-IF
          END-TRY.
      END PROGRAM MAIN.
```

関連するトピックス

Visual Studio のドキュメント のサービスの構成

WCF サービスの構成に関連する情報を提供しています。

WCF サービスを利用する

ここでは、 WCF サービスを利用するクライアントアプリケーションの作成方法について説明します。

このセクションの内容

NCOBU01010

サービスを参照する

クライアントアプリケーションからサービスを参照するためには、サービス側から提供されるサービスコントラクトやデータコントラクトのメタデータ(型)を参照できなければなりません。 ここでは、 クライアントアプリケーションでコントラクトのメタデータを参照する方法について説明します。

サービス側のコントラクトのソースコードをそのまま使用する

クライアントアプリケーションがサービス側と同一の言語で記述されていて、そのサービスのコントラクトを定義したソースコードを参照できる場合は、クライアントアプリケーションのプロジェクトに含めることができます。 このため、コントラクトのインタフェース定義やクラス定義をアプリケーション内部の型として参照することができます。

サービス側のコントラクトが含まれるアセンブリを参照する

サービス側のコントラクトが含まれるアセンブリ(.dll)を参照できる場合は、そのアセンブリをクライアント環境に配置します。クライアントアプリケーションからそのアセンブリを参照することで、サービス側と同一のメタデータを直接使用することができます。

プロキシコードを使用する

クライアントアプリケーションでコントラクトのメタデータを参照したい場合、コントラクトが定義されたソースコードもアセンブリも参照できない場合があります。 この場合は、必要とするサービス側のコントラクトと同じ構造のインタフェース定義やクラス定義をコピーしたコード(プロキシコード)の作成が必要となります。

プロキシコードは、メタデータ ユーティリティ ツール(Svcutil.exe)を使用することで、 WCF サービスから生成することができます。

プロキシコードを使用したクライアントアプリケーションの作成方法については、プロキシコードを利用してクライアントを作成するで説明します。

Visual Studio の[サービス参照の追加]を使用する

Visual Studio のプロジェクトで、対象フレームワークとして“.NET Framework 4“が選択されている場合は、WCF サービスのサービスクライアント(プロキシ)コードをプロジェクトに追加することができます。

詳細については、Visual Studio のドキュメントの方法 : サービス参照を追加、更新、または削除するを参照して下さい。

サービスクライアントコードをプロジェクトに追加する方法については、チュートリアルの WCF サービスのクライアントからの利用でも説明しています。

[サービス参照の追加]で生成されるサービスクライアントは C#によるコードが生成されます。

サービスに接続する

ここでは、 クライアントアプリケーションからサービスに接続する方法について説明します。

WCF サービスを利用するクライアントプログラムは、サービス側の実装と同様に、ターゲットアドレス、通信手段となるバインディングおよびサービスコントラクトを含むエンドポイントが構成されたチャネルファクトリによって、対応するサービス側のエンドポイントへの通信が可能になります。

サービスコントラクトが関連付けられたチャネルファクトリ (System.ServiceModel.ChannelFactory ジェネリッククラス)にエンドポイントを設定して初期化することによりサービスとのセッションを可能にし、チャネルファクトリからチャネル(サービスコントラクトのインスタンス)を取得することで、セッションが開始されます。

以下に NetCOBOL for .NET によるチャネルファクトリを使用したクライアントプログラムのサンプルコードを示します。

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN AS "ZipCodeClient.Main".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
    CLASS CLASS-CHANNELFACTORY AS "System.ServiceModel.ChannelFactory<>"
     INTERFACE INTERFACE-IZipCodeService AS "ZipCodeServiceLib.IZipCodeService"
     CLASS CLASS-FACTORY EXPANDS CLASS-CHANNELFACTORY USING INTERFACE-IZipCodeService
     INTERFACE INTERFACE-ICHANNEL AS "System.ServiceModel.Channels.IChannel"
    CLASS CLASS-ZipAddress AS "ZipCodeServiceLib.ZipAddress"
    PROPERTY PROP-PREFECTURENAME AS "PrefectureName"
    PROPERTY PROP-CITYNAME AS "CityName"
    PROPERTY PROP-AREANAME AS "AreaName".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CHANNEL-FACTORY OBJECT REFERENCE CLASS-FACTORY.
01 ZIPCODE-SERVICE OBJECT REFERENCE INTERFACE-IZipCodeService.
01 ZIPADDRESS-DATA OBJECT REFERENCE CLASS-ZipAddress.
01 郵便番号データ.
  02 郵便番号   PIC X(7).
  02 都道府県名 PIC N(100).
  02 市区町村名 PIC N(100).
  02 町域名     PIC N(100).
01 YN PIC X VALUE SPACE.
01 CHANNEL  OBJECT REFERENCE INTERFACE-ICHANNEL.
PROCEDURE DIVISION.
    TRY
        *> チャネルファクトリの作成
        SET CHANNEL-FACTORY TO CLASS-FACTORY::"NEW"("ZipCodeClientConfig")

        TRY
            *> チャネル(サービスコントラクト)の取得
            SET ZIPCODE-SERVICE TO CHANNEL-FACTORY::"CreateChannel"

            PERFORM UNTIL YN = "N" OR YN = "n"
                DISPLAY "郵便番号を入力してください: " NO ADVANCING
                ACCEPT 郵便番号
                DISPLAY " "

                SET ZIPADDRESS-DATA TO ZIPCODE-SERVICE::"GetAddress"(郵便番号)
                SET 都道府県名 TO PROP-PREFECTURENAME OF ZIPADDRESS-DATA
                SET 市区町村名 TO PROP-CITYNAME OF ZIPADDRESS-DATA
                SET 町域名     TO PROP-AREANAME OF ZIPADDRESS-DATA

                DISPLAY "  都道府県名: " 都道府県名(1:50)
                DISPLAY "  市区町村名: " 市区町村名(1:50)
                DISPLAY "  町域名    : " 町域名(1:50)

                DISPLAY "続けますか? (Y/N)" NO ADVANCING
                ACCEPT YN
                DISPLAY " "
            END-PERFORM
```

```
            FINALLY
                *> チャネルを閉じる
                SET CHANNEL TO ZIPCODE-SERVICE AS INTERFACE-ICHANNEL
                 INVOKE CHANNEL "Close"
            END-TRY
        FINALLY
            *> チャネルファクトリを閉じる
            INVOKE CHANNEL-FACTORY "Close"
        END-TRY.
     END PROGRAM MAIN.
```

エンドポイントはクライアントアプリケーションの構成ファイルに記述します。 上記のサンプルコードでチャネルファクトリのコンストラクタの引数に指定している“ZipCodeClientConfig”はエンドポイントの構成名を示しているため、この名前を持つエンドポイントの構成を定義する必要があります。

例えば、“ZipCodeServiceLib.IZipCodeService”というサービスコントラクトが“http://localhost:8080/ZipCodeService”のアドレスで、“WsHttpBinding バインディング”によって公開されているサービスを利用する場合は、 “ZipCodeClientConfig”のエンドポイントの構成は以下のように記述します。

```
<configuration>
  <system.serviceModel>
    <client>
      <endpoint name="ZipCodeClientConfig"
              contract="ZipCodeServiceLib.IZipCodeService"
              binding ="wsHttpBinding"
              address ="http://localhost:8080/ZipCodeService">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

構成ファイルは、構成エディター ツール (SvcConfigEditor.exe)を使用して編集することができます。

チャネルファクトリもサービスホストプログラムと同様にエンドポイントをソースコードに記述することができます。 上記のサンプルと同じ構成をコードで表すと以下のようになります。

```
    .....
     PROCEDURE DIVISION.
         TRY
             *> チャネルファクトリの作成
             SET BINDING TO CLASS-WSHTTPBINDING::"NEW"
              SET ENDPOINTADDRESS TO
                  CLASS-ENDPOINTADDRESS::"NEW"("http://localhost:8080/ZipCodeService")
              SET CHANNEL-FACTORY TO CLASS-FACTORY::"NEW"(BINDING ENDPOINTADDRESS)
             TRY
                 *> チャネル(サービスコントラクト)の取得
                 SET ZIPCODE-SERVICE TO CHANNEL-FACTORY::"CreateChannel"
    .....
```

関連するトピックス

Visual Studio のドキュメント の方法 : ChannelFactory を使用する

チャネルファクトリに関連する情報を提供しています。

プロキシコードを利用してクライアントを作成する

ここでは、 プロキシコードを利用したクライアントアプリケーションの作成方法について説明します。

ServiceModel メタデータ ユーティリティ ツール (Svcutil.exe)は、WCF サービスを参照してプロキシや構成ファイルを自動生成する機能を持ちます。SvcUtil によって生成されるプロキシコードは、WCF サービスからサービスコントラクト(インタフェース)のコピーとなるソースコードのほか、 チャネルファクトリに相当するインタフェースを実装したクライアントチャネルクラスのソースコードと、 クライアントのエンドポイントが定義された構成ファイルも生成します。

Svcutil はオプション省略時では C#ソースとしてプロキシコードを生成します。 NetCOBOL for .NET で記述されたプロキシコードを生成するには、 以下のオプションを指定しなければなりません。

オプション	指定内容
/language	NetCOBOL for .NET の CodeDomProvider として「Fujitsu.COBOL.COBOLCodeProvider, Fujitsu.COBOL.CodeDom, Version=8.0.124.0, Culture=neutral, PublicKeyToken=fac0fe3cab973246」を指定します。
/reference	NetCOBOL for .NET ランタイムアセンブリとして「C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime V8.0¥Runtime¥Fujitsu.COBOL.dll」を指定します。

以下は、Svcutil を使用して、アドレス「http://wcfservice.example.org/SampleService」から NetCOBOL for .NET によるプロキシコードを生成する例です。

```
Svcutil.exe /syncOnly /language:"Fujitsu.COBOL.COBOLCodeProvider, Fujitsu.COBOL.CodeDom,
Version=8.0.124.0, Culture=neutral, PublicKeyToken=fac0fe3cab973246"
/reference:"C:¥Program Files¥Common Files¥Fujitsu NetCOBOL for .NET Runtime
V8.0¥Runtime¥Fujitsu.COBOL.dll"
http://wcfservice.example.org/SampleService
```

(上記の例では複数行に分けていますが実際には 1 行で記述してください。)

Svcutil コマンドは、デフォルトでは同期メソッドシグネチャとタスクベースの非同期メソッドシグネチャを生成します。 NetCOBOL for .NET ではタスクベースの非同期メソッドシグネチャをサポートしないため、次のいずれかのオプションを必ず指定してください。

- /syncOnly (同期メソッドシグネチャのみ生成します)
- /async (同期メソッドシグネチャと begin/end 非同期メソッドシグネチャの両方を生成します)

生成される COBOL ソースコードの拡張子は、“.cobx”となります。

また、ソースコードの文字コードは UTF-8 です。

生成しようとするプロキシコードの内容によっては、 NetCOBOL for .NET でのプロキシコード生成に失敗することがあります。 プロキシコードの生成に失敗する例としては以下の場合があります。

- プロキシコードが NetCOBOL for .NET のコード生成機能がサポートしていない.NET 機能を必要とする場合。
- プロキシコードが NetCOBOL for .NET のコード生成機能の定量制限を越えるコード生成を必要とする場合。 特に、50 文字を超える文字定数を生成しなければならない場合。

この場合は、 C#でプロキシコードを作成し、 それを翻訳したアセンブリを NetCOBOL for .NET から利用するようにしてください。

生成したプロキシコードは以下の手順でプロジェクトに追加します。

1. WCF サービスを利用するための構成ファイルを組み込みます。 Svcutil はソースコードとともに構成ファイル（既定の名前は“output.config”）を生成します。 この内容をプロジェクトに組み込みます。

   まだプロジェクトに構成ファイル（“App.config”）が追加されていない場合は、 以下の手順で構成ファイルを組み込みます。

   1. 生成した構成ファイルをプロジェクトフォルダにコピーします。
   2. 構成ファイルのファイル名を“App.config”に変更します。
   3. [プロジェクト]メニューから[既存項目の追加]を選択し、 [既存項目の追加]ダイアログで“App.config”をプロジェクトへ追加します。

   プロジェクトに既に構成ファイル（“App.config”）が追加されている場合は、 以下の手順で構成ファイルの内容を組み込みます。

   4. 生成した構成ファイルの<configuration>要素の中身（<configuration>要素の開始・終了タグは含みません）を “App.config”の<configuration>の子要素として追加します 。

   プロジェクトに“App.config”が既に存在する場合は、 以下のオプションと共に Svcutil.exe を実行すると、 構成ファイルを生成する代わりに、 既存の“App.config”の中に構成を直接追加することができます。

   ```
   Svcutil.exe /syncOnly /config:既存の App.config のパス /mergeConfig /language:.......
   ```

2. NetCOBOL for .NET ソースとしてプロキシコードを生成した場合は、 以下の手順でプロキシコードをプロジェクトに追加します。

   1. 生成した COBOL ソースをプロジェクトフォルダにコピーします。
   2. [プロジェクト]メニューから[既存項目の追加]を選択し、 [既存項目の追加]ダイアログでコピーした COBOL ソースをプロジェクトへ追加します。
   3. [プロジェクト]メニューから[参照の追加]を選択し、[参照マネージャー]ダイアログで、以下のアセンブリを追加します。
      - § System.ServiceModel
      - § System.Runtime.Serialization

   C#ソースとしてプロキシコードを生成した場合は、以下の手順でプロキシコードを外部アセンブリとしてプロジェクトに追加します。

   4. 生成した C#ソースをアセンブリに翻訳します。
   5. [プロジェクト]メニューから[参照の追加]を選択し、[参照マネージャー]ダイアログで、翻訳したアセンブリを追加します。

以下は Svcutil で生成されたプロキシコードを利用したクライアントプログラムの例です。

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN AS "ZipCodeClient.Main".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
    CLASS CLASS-ZipAddress AS "ZipCodeServiceLib.ZipAddress"
     CLASS CLASS-CLIENT AS "ZipCodeServiceClient"
    PROPERTY PROP-PREFECTURENAME AS "PrefectureName"
    PROPERTY PROP-CTIYNAME AS "CtiyName"
    PROPERTY PROP-AREANAME AS "AreaName".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CLIENT OBJECT REFERENCE CLASS-CLIENT.
01 ZIPADDRESS-DATA OBJECT REFERENCE CLASS-ZipAddress.
01 郵便番号データ.
  02 郵便番号   PIC X(7).
  02 都道府県名 PIC N(100).
  02 市区町村名 PIC N(100).
  02 町域名     PIC N(100).
01 YN PIC X VALUE SPACE.
PROCEDURE DIVISION.
    TRY
        *> クライアントチャネルの生成
        SET CLIENT TO CLASS-CLIENT::"NEW"("ZipCodeClientConfig")

        PERFORM UNTIL YN = "N" OR YN = "n"
            DISPLAY "郵便番号を入力してください: " NO ADVANCING
            ACCEPT 郵便番号
            DISPLAY " "

            SET ZIPADDRESS-DATA TO CLIENT::"GetAddress"(郵便番号)
            SET 都道府県名 TO PROP-PREFECTURENAME OF ZIPADDRESS-DATA
            SET 市区町村名 TO PROP-CTIYNAME OF ZIPADDRESS-DATA
            SET 町域名     TO PROP-AREANAME OF ZIPADDRESS-DATA

            DISPLAY "  都道府県名: " 都道府県名(1:50)
            DISPLAY "  市区町村名: " 市区町村名(1:50)
            DISPLAY "  町域名    : " 町域名(1:50)

            DISPLAY "続けますか? (Y/N)" NO ADVANCING
            ACCEPT YN
            DISPLAY " "
        END-PERFORM
    FINALLY
        *> クライアントチャネルを閉じる
        INVOKE CLIENT "Close"
    END-TRY.
END PROGRAM MAIN.
```

ZipCodeServiceClient クラスはクライアントチャネルを示す、SvcUtil によって生成されたプロキシコードに含まれるクラスです。

このように、Svcutil によって生成されたプロキシコードを使用した場合は、 一般のオブジェクト参照と同様にクライアントチャネルクラスをインスタンス化して使用するため、 コードは簡素化されます。

NCOBU01013

NetCOBOL for .NET で WCF を利用するための注意事項

ここでは、 NetCOBOL for .NET で WCF を利用するための注意事項について説明します。

このセクションの内容

COBOL 独自データ型の引数を使用する

WCF を利用したアプリケーションでの COBOL 独自データ型を引数として使用する場合の注意事項について説明します。

使用できるデータ項目

WCF ではオペレーションコントラクトの引数に COBOL 独自データ型を指定することができます。 COBOL 独自データ型の内容は、サービス側とクライアント側の環境を超えて複写されます。 このため、サービス側の実行環境に依存するデータの内容はクライアント側では無効な値となります。 実行環境に依存するデータには以下があります。

- ポインタデータ項目

COBOL 独自データ型によるデータの受け渡し

このバージョンの NetCOBOL for .NET では、 サービス側でオペレーションコントラクトの参照渡しの COBOL 独自データ型の引数に値を指定しても、 引数の内容はクライアント側に反映されません。 ただし、復帰値によるデータの返却は可能です。

プロキシコードでの注意事項

オペレーションコントラクトで COBOL 独自データ型を使用している場合、 Visual Studio の[サービス参照の追加]で WCF サービスのサービスクライアントコードを追加するときや、Svcutil ツールを使用してプロキシコードを作成するときは COBOL 独自データ型を扱うことができません。 この場合は、手動でサービスコントラクトおよびデータコントラクトを記述してください。

- COBOL 独自データ型の引数を使用する

NCOBU01015

NetCOBOL for .NET でデータコントラクトを作成する

ここでは、 NetCOBOL for .NET でデータコントラクトを作成する場合の注意事項について説明します。

データメンバーとして利用できない COBOL 機能

WCF サービスで扱うことのできるクラスは、 データコントラクトとしてシリアライズできなければなりませんが、 以下の機能を使用している場合は COBOL 独自の内部的な制御域を持つためデータコントラクトのデータメンバーとして扱うことができません。

- ファイル機能(印刷/表示ファイルを含む)
- 整列併合機能
- SQL 機能

データメンバーにプロパティを使用する

COBOL で作成したデータコントラクトのデータメンバーにプロパティを使用している場合、 Visual Studio の[サービス参照の追加]で WCF サービスのサービスクライアントコードを追加するときや、Svcutil ツールを使用してプロキシコードを作成するときは、 クライアント側のプロキシコードにはデータコントラクトのデータメンバーとして生成されません。 この場合は、手動でサービスコントラクトおよびデータコントラクトを記述してください。

- 値渡しでマーシャリングされるオブジェクトを作成する

プログラム定義を使用する

ここでは、 サービス側のオペレーションコントラクト(メソッド)からプログラム定義を呼び出す場合の注意事項について説明します。

サービスホストはクライアントからセッションが開始されると、サービスクラスをインスタンス化します。そしてそのサービスは、サービスホストとは別のスレッドで動作します。 このため、サービス側のインスタンスメソッドとして定義されるオペレーションコントラクトから、プログラム定義を呼び出す場合は注意が必要です。

マルチスレッドアプリケーションの詳細については、マルチスレッドアプリケーションを参照してください。

サービスクラスは省略値ではセッション単位にインスタンス化されるため、複数のサービスインスタンスから同時にプログラムが呼び出される場合が想定されます。 サービスインスタンスをサービスホストで一貫したひとつのインスタンスで実行させたい場合は、 サービスクラスに System.ServiceModel.ServiceBehaviorAttribute のカスタム属性を付け、InstanceContextMode プロパティに InstanceContextMode.Single を設定してサービスホストを起動する必要があります。InstanceContextMode プロパティはサービスインスタンスの生成方法を指定します。

更に、サービスクラスに System.ServiceModel.ServiceBehaviorAttribute のカスタム属性の、ConcurrencyMode プロパティを指定することで同時実行の設定をすることができます。

System.ServiceModel.ServiceBehaviorAttribute のカスタム属性に指定する、InstanceContextMode プロパティと ConcurrencyMode プロパティについて、以下に説明します。

InstanceContextMode プロパティ

PerSession	InstanceContextMode.PerSession が指定された場合(省略値)は、 クライアントチャネルからセッションが開始された場合にサービスインスタンスが作成され、 チャネルが閉じられたときにサービスインスタンスは閉じられます。
PerCall	InstanceContextMode.PerCall が指定された場合は、クライアントからオペレーションコントラクト(サービスのメソッド)が呼び出される度にサービスインスタンスが生成されます。
Single	InstanceContextMode.Single が指定された場合は、 最初のセッションが開始された場合にサービスインスタンスが作成され、 サービスホストが終了するまでインスタンスは維持されます。サービスクラスに InstanceContextMode.Single を指定している場合は、ServiceHost のコンストラクタにサービスクラスの Type オブジェクトではなく、 サービスインスタンスを予め生成したオブジェクトを指定することもできます。

ConcurrencyMode プロパティ

Single	シングルスレッドで実行されます。省略値はこのモードです。
Reentrant	シングルスレッドで実行されます。この指定は再入可能となります。
Multiple	マルチスレッドで実行されます。

ConcurrencyMode.Single または ConcurrencyMode.Reentrant に設定した場合、 サービスはシングルスレッドで動作しますが、 これはひとつのスレッドでサービスオペレーションが実行されている間は、別のスレッドのサービスオペレーションが実行されないというだけであり、 サービスホストのアプリケーションドメインで一貫したスレッドにはならないので注意が必要です。

また、ConcurrencyMode.Reentrant を指定した場合は、コールバック機能を利用する双方向型のメッセージングを行うようなときは再入可能となりますが、再入処理は別のスレッドで実行されます。

以下は、 System.ServiceModel.ServiceBehaviorAttribute のカスタム属性に InstanceContextMode プロパティと ConcurrencyMode プロパティ設定したサンプルコードです。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-1 AS "ZipServer.ZipCodeService" CUSTOM-ATTRIBUTE IS CA-SERVICEBEHAVIOR.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-SERVICEBEHAVIOR
               CLASS CLASS-SERVICEBEHAVIOR
               PROPERTY PROP-INSTANCECONTEXTMODE IS PROP-SINGLE OF
                        ENUM-INSTANCECONTEXTMODE
               PROPERTY PROP-CONCURRENCYMODE IS PROP-SINGLE OF ENUM-CONCURRENCYMODE
   .
REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   INTERFACE INTERFACE-IZipCodeService AS "ZipServer.IZipCodeService"
   CLASS CLASS-ZipAddress AS "ZipServer.ZipAddress"
   CLASS CLASS-SERVICEBEHAVIOR AS "System.ServiceModel.ServiceBehaviorAttribute"
    ENUM ENUM-INSTANCECONTEXTMODE AS "System.ServiceModel.InstanceContextMode"
    ENUM ENUM-CONCURRENCYMODE AS "System.ServiceModel.ConcurrencyMode"

    PROPERTY PROP-INSTANCECONTEXTMODE AS "InstanceContextMode"
    PROPERTY PROP-CONCURRENCYMODE AS "ConcurrencyMode"
    PROPERTY PROP-SINGLE AS "Single"
   PROPERTY PROP-ZIPCODE AS "ZipCode"
   PROPERTY PROP-PREFECTURENAME AS "PrefectureName"
   PROPERTY PROP-CITYNAME AS "CityName"
   PROPERTY PROP-AREANAME AS "AreaName"
   .
.....
```

以下は、サービスホストのコンストラクタにサービスクラスの Type オブジェクトではなく、 サービスインスタンスをあらかじめ生成したオブジェクトを指定する例です。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "ZipServer.ZipServiceHost" INHERITS CLASS-SERVICEHOST.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-SERVICEHOST AS "System.ServiceModel.ServiceHost"
   CLASS CLASS-ZipCodeSerivce AS "ZipServer.ZipCodeService"
   CLASS CLASS-URI AS "System.Uri"
   CLASS CLASS-TYPE AS "System.Type"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. NEW.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 URI OBJECT REFERENCE CLASS-URI.
01 SERVICE-TYPE OBJECT REFERENCE CLASS-TYPE.
01 SVC OBJECT REFERENCE CLASS-ZipCodeSerivce.
```

```
      PROCEDURE DIVISION.
         SET URI TO CLASS-URI::"NEW"("http://localhost:8080/ZipCodeService").
         SET SVC TO CLASS-ZipCodeSerivce::"NEW"
         INVOKE SUPER "NEW" USING SVC URI

      END METHOD NEW.

      END OBJECT.

      END CLASS CLASS-THIS.
```

関連するトピックス

Visual Studio のドキュメント のセッション、インスタンス化、および同時実行

WCF でのセッション、インスタンス化および同時実行に関連する情報を提供しています。

NCOBU01017

データベース機能を使用する

ここでは、 WCF サービスでデータベース機能を使用する場合の注意事項について説明します。

オペレーションコントラクト(メソッド)の実装で埋込み SQL を使用する

オペレーションコントラクトの実装で埋込み SQL を使用する場合、実行時のコネクション有効範囲に注意しなければなりません。

- オペレーションコントラクトから埋込み SQL を記述したプログラム定義を呼び出す場合は、@SQL_CONNECTION_SCOPE (コネクションの有効範囲の指定)に THREAD を指定します。
- オペレーションコントラクトに埋込み SQL を直接記述する場合 (同一インスタンスまたはサービスインスタンスと寿命が同期しているオブジェクトの埋込み SQL が記述されたメソッドを呼び出す場合も含みます)は、 @SQL_CONNECTION_SCOPE には OBJECT_INSTANCE を指定してください。 ただし、サービスクラスの System.ServiceModel.ServiceBehaviorAttribute のカスタム属性の ConcurrencyMode プロパティに ConcurrencyMode.Reentrant や ConcurrencyMode.Multiple を指定している場合は、同一インスタンス上のメソッドが複数のスレッドから同時に呼び出される可能性があるため、プログラム定義を使用するかスレッドの同期制御を行う必要があります。

サービスクラスの System.ServiceModel.ServiceBehaviorAttribute のカスタム属性の ConcurrencyMode プロパティに ConcurrencyMode.Single または ConcurrencyMode.Reentrant が指定されている場合はシングルスレッドモードとなりますが、 サービスが実行されるスレッドはサービスホストのアプリケーションドメインで一貫されない場合があるので@SQL_CONNECTION_SCOPE に APPLICATION_DOMAIN を指定した場合の動作は保証しません。

マルチスレッドアプリケーションでのデータベース機能を使う場合の注意事項については、マルチスレッドの使用方法(基本編)のリモートデータベースアクセスの利用を参照してください。

WCF の分散トランザクション機能を使用する

WCF の分散トランザクションが有効な場合、 ADO.NET 接続による埋込み SQL 機能では、自動的にトランザクションフローに参加することができます。

WCF の分散トランザクションを有効にするためには、サーバ側のオペレーションコントラクトに System.ServiceModel.TransactionFlowAttribute のカスタム属性を付けます。

以下のサンプルコードでは、System.ServiceModel.TransactionFlowAttribute のカスタム属性に TransactionFlowOption の Mandatory を指定することで、クライアントでのトランザクションへの参加を義務づけています。

```
IDENTIFICATION DIVISION.
INTERFACE-ID. INTERFACE-SERVICE AS "WCFManageStock.IManageStockService"
   CUSTOM-ATTRIBUTE CA-SERVICECONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-SERVICECONTRACT CLASS CLASS-SERVICECONTRACT
   CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT CLASS CLASS-OPERATIONCONTRACT
```

```
       CUSTOM-ATTRIBUTE CA-TRANSACTIONFLOW CLASS CLASS-TRANSACTIONFLOW
                                    USING PROP-MANDATORY OF ENUM-TRANSACTIONFLOWOPTION
       .
REPOSITORY.
       CLASS CLASS-SERVICECONTRACT AS "System.ServiceModel.ServiceContractAttribute"
       CLASS CLASS-OPERATIONCONTRACT AS "System.ServiceModel.OperationContractAttribute"
       CLASS CLASS-TRANSACTIONFLOW AS "System.ServiceModel.TransactionFlowAttribute"
        ENUM  ENUM-TRANSACTIONFLOWOPTION AS "System.ServiceModel.TransactionFlowOption"
        PROPERTY PROP-MANDATORY AS "Mandatory".

PROCEDURE DIVISION.

METHOD-ID. OPERATION-1 AS "Stock" CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT
                                            CA-TRANSACTIONFLOW.
DATA DIVISION.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
END METHOD OPERATION-1.

METHOD-ID. OPERATION-2 AS "Delivery" CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT
                                              CA-TRANSACTIONFLOW.
DATA DIVISION.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
END METHOD OPERATION-2.

END INTERFACE INTERFACE-SERVICE.
```

サービスクラスでは、実装したオペレーションコントラクトに System.ServiceModel.OperationBehaviorAttribute のカスタム属性を付け、TransactionScopeRequired プロパティおよび TransactionAutoComplete プロパティをそれぞれ有効にします。これにより、オペレーションをトランザクションに自動的に組み込むことができ、オペレーションが例外を発生せずに実行された場合にコミットがマークされるようになります。

また、データベースアクセスルーチンではエラーが発生した場合は現在のアンビエントトランザクションを取得し、アンビエントトランザクションが有効な場合はトランザクションのロールバックを行っています。

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "WCFManageStock.ManageStockService".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
       CUSTOM-ATTRIBUTE CA-OPERATIONBEHAVIOR CLASS CLASS-OPERATIONBEHAVIOR
                              PROPERTY PROP-TRANSACTIONSCOPEREQUIRED IS B"1"
                              PROPERTY PROP-TRANSACTIONAUTOCOMPLETE IS B"1"
       .

REPOSITORY.
       CLASS CLASS-STRING AS "System.String"
       INTERFACE INTERFACE-SERVICECONTRACT AS "WCFManageStock.IManageStockService"
       CLASS CLASS-OPERATIONBEHAVIOR AS
             "System.ServiceModel.OperationBehaviorAttribute"
        PROPERTY PROP-TRANSACTIONSCOPEREQUIRED AS "TransactionScopeRequired"
        PROPERTY PROP-TRANSACTIONAUTOCOMPLETE AS "TransactionAutoComplete"
       .
OBJECT. IMPLEMENTS INTERFACE-SERVICECONTRACT.
PROCEDURE DIVISION.

METHOD-ID. OPERATION-1 AS "Stock" CUSTOM-ATTRIBUTE CA-OPERATIONBEHAVIOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 KIND    PIC S9(4) COMP-5.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
       MOVE 1 TO KIND. *> 1: 入庫
```

```
           *> データベースアクセスルーチンの呼び出し
              CALL "WCFManageStock.ManageStock" USING KIND GOODSNO QUANTITY UPDATED.
       END METHOD OPERATION-1.

       METHOD-ID. OPERATION-2 AS "Delivery" CUSTOM-ATTRIBUTE CA-OPERATIONBEHAVIOR.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 KIND    PIC S9(4) COMP-5.
       LINKAGE SECTION.
       01 GOODSNO  PIC S9(4) COMP-5.
       01 QUANTITY PIC S9(9) COMP-5.
       01 UPDATED  PIC S9(9) COMP-5.
       PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
           MOVE 2 TO KIND. *> 2: 出庫
           *> データベースアクセスルーチンの呼び出し
           CALL "WCFManageStock.ManageStock" USING KIND GOODSNO QUANTITY UPDATED.
       END METHOD OPERATION-2.

       END OBJECT.
       END CLASS CLASS-THIS.
      ***
      *** データベースアクセスルーチン
      ***
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PROGRAM-1 AS "WCFManageStock.ManageStock".
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
       REPOSITORY.
           CLASS CLASS-TRANSACTION AS "System.Transactions.Transaction"
            PROPERTY PROP-CURRENT AS "Current"
           .
       DATA DIVISION.
       WORKING-STORAGE SECTION.
           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01 SQLSTATE        PIC X(5).
       01 SQLINFOA.
         02 SQLERRD       PIC S9(9) COMP-5 OCCURS 6 TIMES.
       01 PRODUCT_NUMBER   PIC S9(4) COMP-5.
       01 INOUT_KIND       PIC S9(4) COMP-5.
       01 INOUT_QUANTITY   PIC S9(9) COMP-5.
       01 UPDATED_QUANTITY PIC S9(9) COMP-5.
       01 ERROR_MESSAGE    PIC X(256).
           EXEC SQL END   DECLARE SECTION END-EXEC.
       01 CURRENT-TRANSACTION OBJECT REFERENCE CLASS-TRANSACTION.
       LINKAGE SECTION.
       01 KIND    PIC S9(4) COMP-5.
       01 GOODSNO  PIC S9(4) COMP-5.
       01 QUANTITY PIC S9(9) COMP-5.
       01 UPDATED  PIC S9(9) COMP-5.
       PROCEDURE DIVISION USING KIND GOODSNO QUANTITY UPDATED.

           EXEC SQL CONNECT TO DEFAULT END-EXEC.

           MOVE GOODSNO  TO PRODUCT_NUMBER.
           MOVE KIND     TO INOUT_KIND.
           MOVE QUANTITY TO INOUT_QUANTITY.

           *> ストアドプロシージャの呼び出し
           EXEC SQL
              CALL ManageStock (:PRODUCT_NUMBER, :INOUT_KIND, :INOUT_QUANTITY,
                                :UPDATED_QUANTITY, :ERROR_MESSAGE)
           END-EXEC.

           MOVE UPDATED_QUANTITY TO UPDATED.

           *> ストアドプロシージャでエラーが発生した場合や
           *> 在庫がマイナスになった場合はロールバックする
           IF SQLERRD(1) NOT = 0 OR UPDATED_QUANTITY < 0
              *> 現在のアンビエントトランザクションを取得
              SET CURRENT-TRANSACTION TO PROP-CURRENT OF CLASS-TRANSACTION
               IF CURRENT-TRANSACTION NOT = NULL
                  INVOKE CURRENT-TRANSACTION "Rollback"
               END-IF
           END-IF.

           EXEC SQL DISCONNECT DEFAULT END-EXEC
       END PROGRAM PROGRAM-1.
```

構成ファイルでは、バインディングの構成設定で transactionFlow 属性を true に設定します。

```
<configuration>
  ......
   <system.serviceModel>
      <bindings>
         <wsHttpBinding>
            <binding name="ManageStockBinding" transactionFlow="true" />
         </wsHttpBinding>
      </bindings>
      <services>
         <service behaviorConfiguration="WCFManageStock.ManageStockServiceBehavior"
            name="WCFManageStock.ManageStockService">
            <endpoint binding="wsHttpBinding" bindingConfiguration="ManageStockBinding"
               contract="WCFManageStock.IManageStockService" />
         </service>
      </services>
      <behaviors>
         <serviceBehaviors>
            <behavior name="WCFManageStock.ManageStockServiceBehavior">
               <serviceMetadata httpGetEnabled="true" />
               <serviceDebug includeExceptionDetailInFaults="true" />
            </behavior>
         </serviceBehaviors>
      </behaviors>
   </system.serviceModel>
  ......
</configuration>
```

クライアント側では、System.Transactions.TransactionScope を使用したトランザクションスコープから、サービス側のオペレーション(メソッド)を呼び出します。 NetCOBOL for .NET ではトランザクションスコープは構造化例外処理の TRY 文で記述することができます。

```
      @OPTIONS NOALPHAL
      IDENTIFICATION DIVISION.
      CLASS-ID. CLASS-THIS AS "WCFManageStockClient.Form1"
         INHERITS CLASS-FORM.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
          .....
         CLASS CLASS-OBJECT AS "System.Object"
         CLASS CLASS-STRING AS "System.String"
         CLASS CLASS-MANAGESTOCKPROXY AS "WCFManageStockClient.ManageStockProxy"
         CLASS CLASS-TRANSACTIONSCOPE AS "System.Transactions.TransactionScope"
         CLASS CLASS-EXCEPTION AS "System.Exception"
          .....

      OBJECT.
          .....
      METHOD-ID. button1_Click PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 PROXY OBJECT REFERENCE CLASS-MANAGESTOCKPROXY.
      01 SCOPE  OBJECT REFERENCE CLASS-TRANSACTIONSCOPE.
      01 KIND    PIC S9(4) COMP-5.
      01 GOODSNO  PIC S9(4) COMP-5.
      01 QUANTITY PIC S9(9) COMP-5.
      01 UPDATED  PIC S9(9) COMP-5.
      01 WK-VAL OBJECT REFERENCE CLASS-STRING.
      01 WK-EXP OBJECT REFERENCE CLASS-EXCEPTION.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.

         *> チャネルを開く
         SET PROXY TO CLASS-MANAGESTOCKPROXY::"NEW"(N"SampleEndpoint")
         TRY
            TRY
               *> トランザクションスコープの開始
               SET SCOPE TO CLASS-TRANSACTIONSCOPE::"NEW"

               SET WK-VAL TO PROP-TEXT OF textBox1
               SET GOODSNO TO CLASS-INT16::"Parse"(WK-VAL)
               SET WK-VAL TO PROP-TEXT OF textBox2
```

```
                SET QUANTITY TO CLASS-INT32::"Parse"(WK-VAL)

                *> オペレーションの呼び出し
                IF PROP-CHECKED OF radioButton1 = B"1" THEN
                    INVOKE PROXY "Stock" USING GOODSNO QUANTITY RETURNING UPDATED
                ELSE
                    INVOKE PROXY "Delivery" USING GOODSNO QUANTITY RETURNING UPDATED
                END-IF

                SET WK-VAL TO CLASS-STRING::"Format"(N"{0}" UPDATED)
                SET PROP-TEXT OF textBox3 TO WK-VAL

                *> トランザクションを完了する
                INVOKE SCOPE "Complete"
            FINALLY
                *> トランザクションスコープの終了
                INVOKE SCOPE "Dispose"
            END-TRY
        CATCH WK-EXP
            INVOKE SELF "ShowMessage" USING  N"データ処理中にエラーが発生しました"
        END-TRY.

        *> チャネルを閉じる
        INVOKE PROXY "Close".

    END METHOD button1_Click.
        .....
    END OBJECT.
    END CLASS CLASS-THIS.
```

クライアント側の構成ファイルもサービス側の構成ファイルと同様に、バインディングの構成設定で transactionFlow 属性を true に設定します。

```
<configuration>
    <system.serviceModel>
        <bindings>
            <wsHttpBinding>
                <binding name="SampleBinding" transactionFlow="true" />
            </wsHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:8080/ManageStockService"
                binding="wsHttpBinding" bindingConfiguration="SampleBinding"
                contract="WCFManageStock.IManageStockService" name="SampleEndpoint">
            </endpoint>
        </client>
    </system.serviceModel>
</configuration>
```

クライアント側のトランザクションスコープから呼び出されることにより、サービス側のオペレーションは自動的にトランザクションフローに参加するため、ADO.NET によるデータベースの接続では暗黙に分散トランザクションに昇格します。

埋込み SQL と分散トランザクションを併用する場合は、サーバ情報の@SQL_COMMIT_MODE (COMMIT モードの指定)は AUTO に設定しなければなりません。詳細については、分散トランザクションを併用する場合の注意事項を参照してください。

関連するトピックス

Visual Studio のドキュメント のトランザクション

WCF のトランザクションに関連する情報を提供しています。

データプロバイダー拡張機能を使用したデータベースアクセス

ここでは、データプロバイダー拡張機能を使用したデータベースアクセス方法について説明します。 データプロバイダー拡張機能は、以下の.NET Framework 標準データプロバイダー以外を使用したデータベース接続をサポートします。

.NET Framework 標準データプロバイダー：

- .NET Framework Data Provider for SQL Server
- .NET Framework Data Provider for OLE DB
- .NET Framework Data Provider for ODBC

上記の標準データプロバイダーは、データプロバイダー拡張機能を使用しなくても高速に ADO.NET 接続ができます。

このセクションの内容

データプロバイダー拡張機能概要

データプロバイダー拡張機能の概要を説明しています。

データプロバイダー拡張ライブラリの作成手順

データプロバイダー拡張ウィザードを使用して、 使用するデータプロバイダー拡張ライブラリを作成する手順について説明しています。

データプロバイダー拡張機能を使用したデータベースアクセス実行方法

データプロバイダー拡張機能を使用して、データベースアクセスの実行方法について説明しています。

NCOBU01174

データプロバイダー拡張機能概要

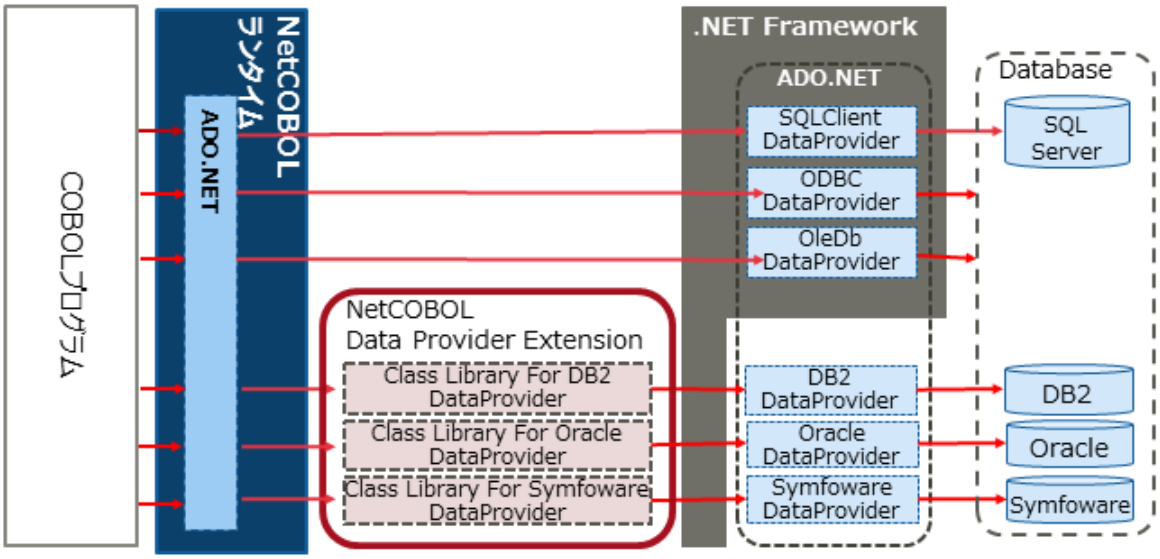
ここでは、データプロバイダー拡張機能の概要を説明します。

NetCOBOL を使用して ADO.NET 接続でデータベースアクセスできるデータプロバイダーは、ADO.NET2.0 に準拠している必要があります。 ADO.NET2.0 に準拠している場合でも、 データプロバイダー独自の仕様が一部存在します。 以下がその機能です。

- ストアドプロシージャのパラメタ情報を取得する機能
- パラメタマーカーとなる文字列を作成する機能
- データベース固有のエラー情報を作成する機能

これらの機能は、 NetCOBOL for .NET ランタイムが提供するデータプロバイダー拡張機能(ICobolSqlProviderInfo インタフェース)を使用して、補完することができます。データプロバイダー拡張機能は、Visual Studio の「データプロバイダー拡張ウィザード」によって、 データプロバイダー拡張ライブラリのプロジェクトを自動で作成することができます。

データプロバイダーとデータプロバイダー拡張ライブラリを使用して ADO.NET 接続する場合の構成図を以下に示します。



NetCOBOL で ADO.NET 接続するためには、 データプロバイダー拡張ライブラリの格納先を環境変数または、SQL 情報設定で指定することで、 参照追加することなく利用できます。

データプロバイダー拡張ライブラリを使用したデータベース接続の手順です。

1. 使用するデータベースを作成する。
2. 使用するデータベースドライバをアプリケーションが実行する環境にインストールする。
3. データプロバイダー拡張ライブラリを作成する。
4. データベースアクセスする COBOL プログラムのプロジェクトにデータプロバイダー拡張ライブラリを使用できるようにする。
5. COBOL プログラムを実行する。

このセクションでの内容

NCOBU01175

データプロバイダー拡張機能の必要条件

データプロバイダー拡張機能が使用できるデータプロバイダーの必要条件は以下です。

- ADO.NET2.0 に準拠している
- System.Data.Common 名前空間のクラスを継承した実装
- Factory モデルの実装
- データプロバイダーは、共有アセンブリとして厳密名（Strong Name）で署名されている
- ストアドプロシジャの呼び出しが.NET シンタックスである

ICobolSqlProviderInfo インタフェース

データプロバイダー拡張のための機能を提供します。

```
名前空間: Fujitsu.COBOL.Interfaces
アセンブリ: Fujitsu.COBOL (Fujitsu.COBOL 内)
```

ICobolSqlProviderInfo インタフェースで公開されるメソッドは以下のとおりです。

- ExecDeriveParameter
- ParameterMarker
- SetErrorInformation

NCOBU01189

ExecDeriveParameter

書式

C#

```
public void ExecDeriveParameter(DbCommand  command);
```

VB

```
Public Sub ExecDeriveParameter(command As DbCommand)
```

COBOL

```
METHOD-ID. EXECDERIVEPARAMETER AS "ExecDeriveParameter" IS PUBLIC.
DATA DIVISION.
LINKAGE SECTION.
01  P-COMMAND OBJECT REFERENCE CLASS-DBCOMMAND.
PROCEDURE DIVISION USING BY VALUE P-COMMAND RAISING CLASS-EXCEPTION.
```

概要

指定された DbCommand からパラメタ情報を取得します。

パラメタ

パラメタ	種別	意味
command	INPUT	データプロバイダーの DeriveParameters のパラメタとして使用する値

ParameterMarker

書式

C#

```
public string ParameterMarker(string paramString)
```

VB

```
Public Function ParameterMarker(paramString As String) As String
```

COBOL

```
METHOD-ID. PARAMETERMARKER AS "ParameterMarker" IS PUBLIC.
DATA DIVISION.
LINKAGE SECTION.
01  P-PARAMSTRING OBJECT REFERENCE CLASS-STRING.
01  R-VALUE OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION USING BY VALUE P-PARAMSTRING
                RETURNING R-VALUE RAISING CLASS-EXCEPTION.
```

概要

データプロバイダに関連付けられている SQL 文で使用する固有のパラメタマーカーを返します。

パラメタ

パラメタ	種別	意味
paramString	INPUT	データプロバイダーが使用するパラメタ

戻り値

データプロバイダーが使用するパラメタマーカー

NCOBU01178

SetErrorInformation

書式

C#

```
public void SetErrorInformation(Exception exception,
                            ref string SQLSTATE,
                            ref int SQLCODE,
                            ref string SQLMSG,
                            ref bool embeddingMsg)
```

VB

```
Public Sub SetErrorInformation(exception As Exception,
                           ByRef SQLSTATE As String,
                           ByRef SQLCODE As Integer,
                           ByRef SQLMSG As String,
                           ByRef embeddingMsg As Boolean)
```

COBOL

```
METHOD-ID. SETERRORINFORMATION AS "SetErrorInformation" IS PUBLIC.
DATA DIVISION.
LINKAGE SECTION.
01 P-EXCEPTION OBJECT REFERENCE CLASS-EXCEPTION.
01 P-SQLSTATE OBJECT REFERENCE CLASS-STRING.
01 P-SQLCODE BINARY-LONG.
01 P-SQLMSG OBJECT REFERENCE CLASS-STRING.
01 P-EMBEDDINGMSG OBJECT REFERENCE CLASS-BOOLEAN.
PROCEDURE DIVISION USING BY VALUE P-EXCEPTION
                      BY REFERENCE P-SQLSTATE
                                   P-SQLCODE
                                   P-SQLMSG
                                   P-EMBEDDINGMSG RAISING CLASS-EXCEPTION.
```

概要

データプロバイダーのエラー情報を COBOL のエラー情報にマッピングする。

パラメタ

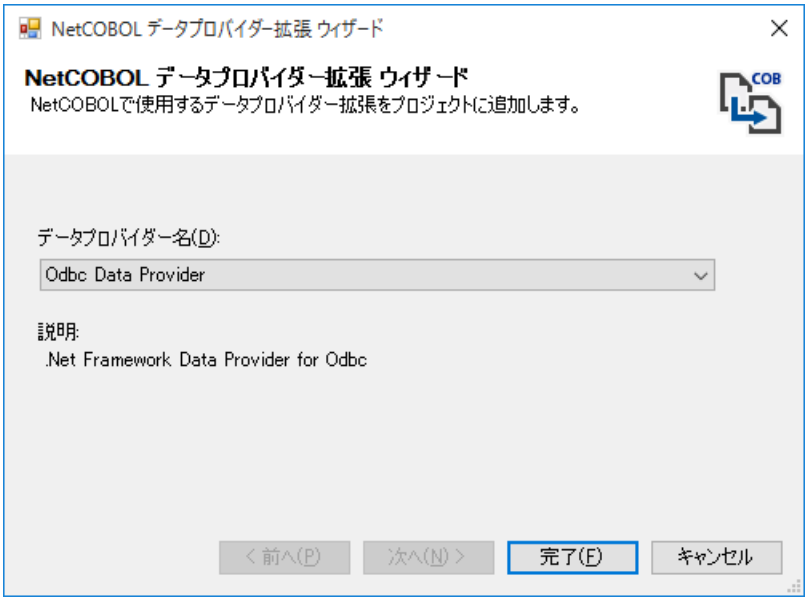
パラメタ	種別	意味
exception	INPUT	データプロバイダーが使用するパラメタ
SQLSTATE	OUTPUT	NetCOBOL の ADO.NET のエラー"9999F"を指定
SQLCODE	OUTPUT	NetCOBOL の ADO.NET のエラー"-1"を指定
SQLMSG	OUTPUT	データプロバイダーの例外メッセージを設定
embeddingMsg	OUTPUT	データプロバイダーの例外オブジェクトを指定したか否かを指定。 True:データプロバイダーのエラーを指定。かならず True を設定してください。

データプロバイダー拡張ライブラリの作成手順

ここでは、データプロバイダー拡張ウィザードを使用して、 使用するデータプロバイダー拡張ライブラリを作成する手順について説明します。

データプロバイダー拡張ライブラリの作成手順：

1. Visual Studio を起動します。
2. Visual Studio の[ファイル]メニューから、 [新規作成]-[プロジェクト]を選択します。
3. [新しいプロジェクト]ダイアログボックスが表示されます。 左側のペインで、 データプロバイダー拡張ライブラリを作成する言語(“Visual C#”、“Visual Basic” および “NetCOBOL for .NET” のいずれか)、[NetCOBOL 拡張]の順に選択します。
4. [NetCOBOL データプロバイダ拡張 ライブラリ]を選択し、名前、場所、ソリューション名を入力して 、[OK]ボタンをクリックします。
5. [NetCOBOL データプロバイダー拡張 ウィザード]ダイアログが表示されます。 ここでは、使用できるデータプロバイダーの一覧が表示されます。 使用するデータプロバイダーを選択し[完了]ボタンをクリックします。



6. 選択したデータプロバイダーを参照したデータプロバイダー拡張ライブラリプロジェクトが作成されます。 デフォルトでは、ICobolSqlProviderInfo インタフェースが実装され DataProviderInfo1 というクラスが 自動で生成されます。必要に応じてクラス名を変更してください。

データプロバイダー拡張は.NET Framework の Managed Extensibility Framework(MEF)によって COBOL ランタイムから込みこまれます。このため、実装クラスには Export 属性および ExportMetadata 属性を 指定しなければなりません。

- Export 属性: パラメタには ICobolSqlProviderInfo 型の Type オブジェクトを指定します。
- ExportMetaData 属性: 第 1 パラメタには文字列“ProviderName”を、 第 2 パラメタにはデータプロバイダーに関連づけられた不変名を指定します。

7. 自動生成された DataProviderInfo1 クラスで、各メソッドの内容を確認します。

NetCOBOL データプロバイダー拡張ウィザードで生成されたソースコードの内容は、 実際のデータプロバイダーの仕様と異なる場合があります。 データプロバイダーの仕様を確認し、必要であれば変更を行ってください。

- ExecDeriveParameter メソッド: データプロバイダー拡張ウィザードは選択されたデータプロバイダーの CommandBuilder クラスから “DeriveParameters” という名前のスタティックメソッドを探し、 検出された場合はその呼び出しを自動で生成します。検出されなかった場合は、例外(NotImplementedException)を発生させるコードを生成します。
- ParameterMarker メソッド: データプロバイダー拡張ウィザードは選択されたデータプロバイダーから プレースホルダーやパラメタ名を推測し、パラメタマーカーを作成するコードを生成します。 推測が正しく行われなかった場合は、例外(NotImplementedException)を発生させるコードを生成します。
- SetErrorInformation メソッド: データプロバイダー拡張ウィザードはデフォルト以下の値を設定するコードを生成します。
  - SQLSTATE : “9999F”
  - SQLCODE: -1
  - SQLMSG: exception(System.Exception クラス)の Message プロパティ

パラメタの exception オブジェクトがデータプロバイダー独自の例外である場合、 必要に応じて適切な SQL エラー情報を設定してください。

8. プロジェクトをビルドします。

既存のプロジェクトにデータプロバイダー拡張を追加する場合：

1. Visual Studio のソリューションエクスプローラーでデータプロバイダー拡張を追加するプロジェクトを選択します。
2. コンテキストメニューから[追加]-[新しい項目]を選択します。
3. [新しい項目の追加]ダイアログボックスの左側のペインで、 [NetCOBOL 拡張]を選択します。
4. [NetCOBOL データプロバイダ拡張]を選択し、名前を入力して [OK]ボタンをクリックします。ICobolSqlProviderInfo インタフェースが実装されたデータプロバイダー拡張クラスが追加されます。

データプロバイダー拡張機能を使用したデータベースアクセス実行方法

ここでは、データプロバイダー拡張機能を使用して、データベースアクセスの実行方法について説明します。

データプロバイダー拡張機能を使用する場合でも従来のデータベース接続と手順と変更ありません。

データベースアクセス実行手順

1. 実行環境設定ユーティリティの サーバ情報の設定方法で、 接続するデータベースを指定します。
   - 接続タイプは ADO.NET を選択します。
   - 接続文字列の設定方法(ADO.NET)により接続文字列を作成します。 接続文字列で選択するプロバイダ名は、データプロバイダー拡張機能で選択したプロバイダ名を選びます。
2. 実行環境設定ユーティリティを使用して、 データプロバイダー拡張機能を使用して作成したデータプロバイダー拡張ライブラリを選択します。 ライブラリは、以下の SQL 情報または環境変数を使用して指定します。

   SQL 情報

   ```
   @SQL_DATAPROVIDER_EXTENSION_DLL=DLL 名
   ```

   コネクション単位にデータプロバイダーを指定することができます。

   環境変数

   ```
   @CBR_DATAPROVIDER_EXTENSION_FOLDER=DLL が格納されているフォルダ
   ```

   注意）フォルダ指定は、セミコロンによる複数個指定はサポートしていません。

   両方が指定された場合は、@SQL_DATAPROVIDER_EXTENSION_DLL に指定された値が優先されます。

NCOBU01181

データプロバイダー拡張ライブラリの検索順序

ここでは、データプロバイダー拡張ライブラリの検索順序について説明します。

以下の順番でデータプロバイダー拡張ライブラリを検索します。

1. アプリケーション構成ファイル(machine.config、web.config または applicationName.config)で指定しているデータプロバイダー拡張情報
2. @SQL_DATAPROVIDER_EXTENSION_DLL に指定された DLL
3. @CBR_DATAPROVIDER_EXTENSION_FOLDER に指定されたフォルダの DLL
4. 実行中のアセンブリ

- 以下の.NET Framework 標準データプロバイダーを使用している場合は、 データプロバイダー拡張ライブラリを使用しないでください。性能が劣化する場合があります。 これらのデータプロバイダは、データプロバイダー拡張機能を使用しなくても高速に ADO.NET 接続ができます。
  - .NET Framework Data Provider for SQL Server
  - .NET Framework Data Provider for OLE DB
  - .NET Framework Data Provider for ODBC
- @SQL_DATAPROVIDER_EXTENSION_DLL に値が指定されていない場合、次の検索が行われます。
- @SQL_DATAPROVIDER_EXTENSION_DLL に指定した DLL が存在しない、または、 検索された DLL が “データプロバイダー拡張ライブラリの作成手順” に従って作成されていない場合、 Data Provider Extension 機能が利用できず、以下のエラーになります。

```
JMP0372I-U SQL 文を実行するための環境開設でエラーが発生しました．'DataProvider Extension[プロバイダ名]'
```

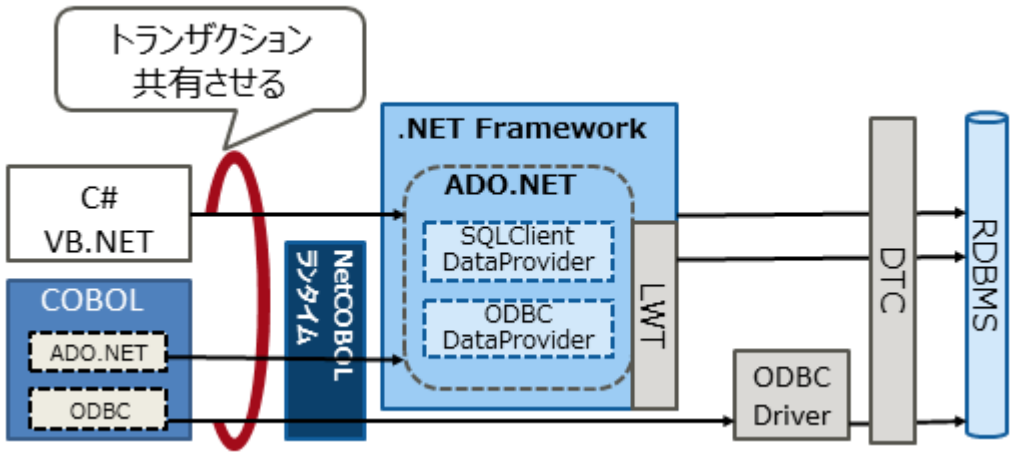
- @CBR_DATAPROVIDER_EXTENSION_FOLDER に値が指定されていない場合、次の検索が行われます。
- @CBR_DATAPROVIDER_EXTENSION_FOLDER に指定したフォルダが存在しない、または、検索されたライブラリが “データプロバイダー拡張ライブラリの作成手順” に従って作成されていない場合、 Data Provider Extension 機能が利用できず、以下のエラーになります。

```
JMP0372I-U SQL 文を実行するための環境開設でエラーが発生しました．'DataProvider Extension[プロバイダ名]'
```

- @CBR_DATAPROVIDER_EXTENSION_FOLDER に指定したフォルダに、データプロバイダー拡張機能で作成したデータプロバイダー拡張ライブラリ以外が含まれている場合、 検索に時間がかかります。 指定するフォルダには、データプロバイダー拡張機能で使用するライブラリのみを格納することをおすすめします。

.NET 言語とトランザクション連携

ここでは、NetCOBOL for .NET と.NET 言語において、トランザクションを共有する方法について説明します。分散トランザクションを使用して、COBOL プログラムと C#や VB などの.NET 言語とトランザクションの共有を行います。



分散トランザクションを使用するためには、.NET Framework が提供する System.Transactions namespace を使用します。

ODBC 接続の COBOL アプリケーションと、.NET 言語でトランザクションを連携する場合は、MS DTC が使用されます。 MS DTC の設定を行ってください。 また、ADO.NET 接続の場合でも、データベースへの設定などに MS DTC へ昇格する場合があります。

このセクションの内容

[明示的トランザクションを使用した.NET 言語連携]

NetCOBOL for .NET と.NET 言語において、明示的トランザクションを使用する方法について説明します。

[暗黙的トランザクションを使用した.NET 言語連携]

NetCOBOL for .NET と.NET 言語において、暗黙的トランザクションを使用する方法について説明します。

[分散トランザクション指定方法]

ADO.NET 接続と ODBC 接続について、それぞれの設定方法を説明します。

NCOBU01187

明示的トランザクションを使用した.NET 言語連携

ここでは、NetCOBOL for .NET と.NET 言語において、明示的トランザクションを使用する方法について説明します。

明示的トランザクションを使用するには、System.Transactions namespace の CommittableTransaction クラスを使用します。

呼び出し側 C#プログラム

```
class Program
 {
    static void Main(string[] args)
    {
       try
       {
          CommittableTransaction ct = new CommittableTransaction();
          Transaction.Current = ct;

          // COBOL PROGRAM
          CobolDataAccess.CobWrapper.COBDBACCESS();             [1]
          // C# PROGRAM
          CshDataAccess csh = new CshDataAccess();
          csh.CshInsert();                                      [2]

          ct.Commit();                                          [3]
          ct.Dispose();
        }
       catch (Exception)
       {
          Console.WriteLine("COBOL and C# threw exception: ROLLBACK");
       }
    }
 }
 // C#によるデータベースアクセス
 public class CshDataAccess
 {
    string dbConnectionString = null;

    public CshDataAccess()
    {
       dbConnectionString =
           ConfigurationManager.ConnectionStrings["LocalSqlServer"].ConnectionString;
    }

    public void CshInsert()
    {
       SqlConnection conn = new SqlConnection(dbConnectionString);
       conn.Open();

       SqlCommand cmd = conn.CreateCommand();
       cmd.CommandText = "INSERT INTO TESTTBL (COL1) VALUES (@val)";
       cmd.Parameters.AddWithValue("@val", "HELLO C#");

       cmd.ExecuteNonQuery();
       conn.Close();
    }
 }
```

[1] COBOL プログラムからデータベース更新を実行 [2] C＃プログラムからデータベース更新を実行 [3][1],[2]の更新が COMMIT される

C#と COBOL プログラムの連携クラス

```
     IDENTIFICATION DIVISION.
     CLASS-ID. CLASS-1 AS "CobolDataAccess.CobWrapper".
     ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
     SPECIAL-NAMES.
```

```
      REPOSITORY.

      STATIC.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      PROCEDURE DIVISION.

      METHOD-ID. METHOD-1 AS "COBDBACCESS".
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      PROCEDURE DIVISION.
          CALL "CobolDataAccess.CobDbAccess".
      END METHOD METHOD-1.
      END STATIC.

      END CLASS CLASS-1.
```

呼び出し先 COBOL プログラム

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. PROGRAM-1 AS "CobolDataAccess.CobDbAccess".
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
      REPOSITORY.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
          EXEC SQL BEGIN DECLARE SECTION END-EXEC.
      01 SQLSTATE PIC X(5).
      01 SQLCODE  PIC S9(9) COMP-5.
      01 SQLMSG   PIC X(256).
          EXEC SQL END DECLARE SECTION END-EXEC.
      PROCEDURE DIVISION.

          EXEC SQL  CONNECT TO DEFAULT END-EXEC.

          EXEC SQL
              INSERT INTO TESTTBL(COL1) VALUES ('HELLO COBOL')
          END-EXEC.

          EXEC SQL  DISCONNECT DEFAULT END-EXEC.

      END PROGRAM PROGRAM-1.
```

NCOBU01190

暗黙的トランザクションを使用した.NET 言語連携

ここでは、NetCOBOL for .NET と.NET 言語において、暗黙的トランザクションを使用する方法について説明します。

暗黙的トランザクションを使用するには、System.Transactions namespace の TransactionScope クラスを使用します。

呼び出し側 C#プログラム

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            using (TransactionScope ts = new TransactionScope())
            {
                // COBOL PROGRAM
                CobolDataAccess.CobWrapper.COBDBACCESS();

                // C# PROGRAM
                CshDataAccess csh = new CshDataAccess();
                csh.CshInsert();

                 ts.Complete();                          [1]
              }                                          [2]
         }
        catch (Exception)
        {
            Console.WriteLine("COBOL and C# threw exception: ROLLBACK");
        }
    }
}
// C#によるデータベースアクセス
public class CshDataAccess
{
    string dbConnectionString = null;

    public CshDataAccess()
    {
        dbConnectionString =
            ConfigurationManager.ConnectionStrings["LocalSqlServer"].ConnectionString;
    }

    public void CshInsert()
    {
        SqlConnection conn = new SqlConnection(dbConnectionString);
        conn.Open();

        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "INSERT INTO TESTTBL (COL1) VALUES (@val)";
        cmd.Parameters.AddWithValue("@val", "HELLO C#");

        cmd.ExecuteNonQuery();
        conn.Close();
    }
}
```

[1] TransactionScope 内のすべての操作が正常に終了。

[2] TransactionScope スコープを出ると、COBOL と C#の処理が反映される。

C#と COBOL プログラムの連携クラス

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-1 AS "CobolDataAccess.CobWrapper".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.

STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. METHOD-1 AS "COBDBACCESS".
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
   CALL "CobolDataAccess.CobDbAccess".
END METHOD METHOD-1.
END STATIC.

END CLASS CLASS-1.
```

呼び出し先 COBOL プログラム

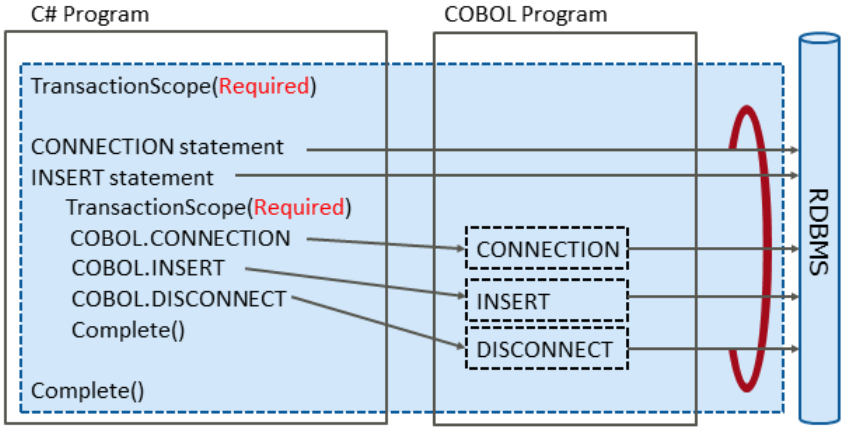
```
IDENTIFICATION DIVISION.
PROGRAM-ID. PROGRAM-1 AS "CobolDataAccess.CobDbAccess".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
DATA DIVISION.
WORKING-STORAGE SECTION.
   EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 SQLSTATE PIC X(5).
01 SQLCODE  PIC S9(9) COMP-5.
01 SQLMSG   PIC X(256).
   EXEC SQL END DECLARE SECTION END-EXEC.
PROCEDURE DIVISION.

   EXEC SQL  CONNECT TO DEFAULT END-EXEC.

   EXEC SQL
      INSERT INTO TESTTBL(COL1) VALUES ('HELLO COBOL')
   END-EXEC.

   EXEC SQL  DISCONNECT DEFAULT END-EXEC.

END PROGRAM PROGRAM-1.
```

TRANSACTOPMSCOPE の設定値

TransactionScope では、TransactionScopeOption 列挙体によって、 トランザクションのスコープを指定することができます。 使用できるメンバは、TransactionScopeOption 列挙体を参照してください。

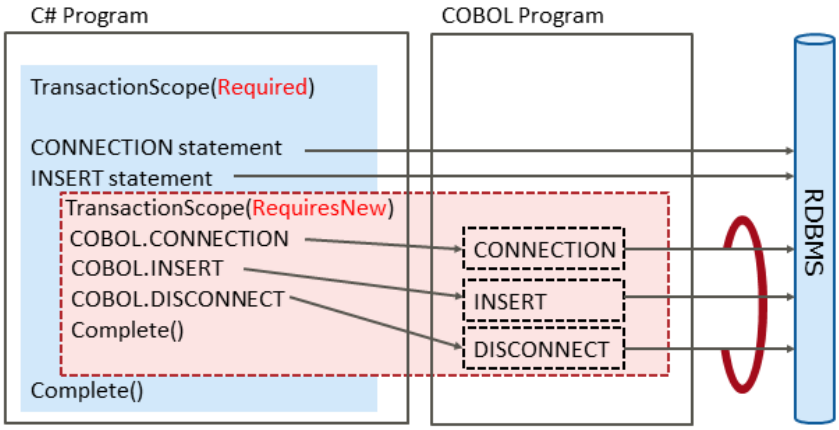
1.REQUIRED の動作

Ambient トランザクションがある場合、Ambient トランザクションに参加します。 Ambient トランザクションがない場合は、新しいトランザクションが作成されます。 C#と COBOL は同じトランザクションとして管理されます。



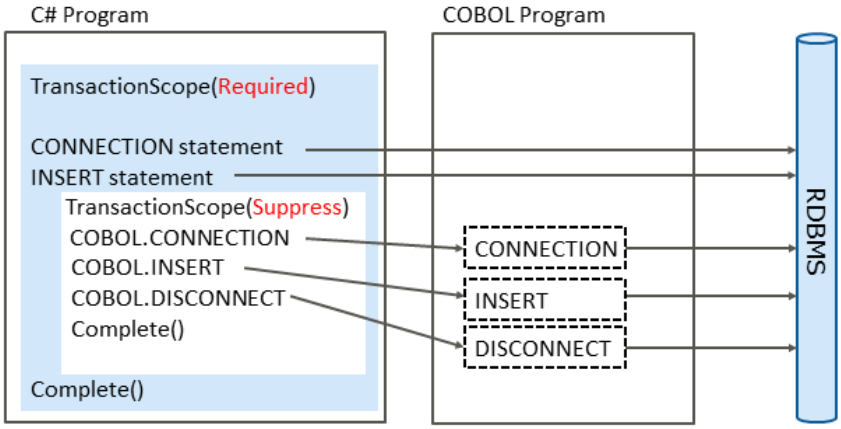
2.REQUIRESNEW の動作

Ambient トランザクションとは別に新しいトランザクションが作成されます。 赤色のコネクションが同じトランザクション管理を示します。



3.SUPRESS の動作

Ambient トランザクションとは別に新しいトランザクションが作成されます。



分散トランザクション指定方法

明示的トランザクションと暗黙的トランザクションの設定方法は同じです。 以下に ADO.NET 接続と ODBC 接続について、それぞれの設定方法を説明します。

ADO.NET 接続で設定

ADO.NET の場合は、@SQL_COMMIT_MODE に ENLIST を指定します。

```
<configuration>
 <connectionStrings>
   <add name="LocalSqlServer"
       connectionString="Data Source=xxx;Initial Catalog=XXX;User ID=xxx;
                         Password=xxxxxx;Enlist=true;"             [1]
       providerName="System.Data.SqlClient"/>
 </connectionStrings>
          :
  <fujitsu.cobol>
      <runtime>
         <sqlSettings>
             <serverList>
                <server name="SERVER1" type="adonet" description="SERVER1">
                   <add key="@SQL_DATASRC" value="LocalSqlServer"/>
                   <add key="@SQL_COMMIT_MODE" value="ENLIST"/>    [2]
         :
```

ADO.NET の場合は、以下のように設定してください。

- 使用するデータベースプロバイダの接続文字列に、Enlist=true を指定してください。([1])
- SQL_COMMIT_MODE に、ENLIST または AUTO を指定してください。([2])

SQL Client Data Provider の場合は、デフォルトが Enlist=true であるため、明に指定する必要ありません。 使用するデータプロバイダーの仕様を確認してください。

ODBC 接続で設定

ODBC 接続の場合は、@SQL_COMMIT_MODE に ENLIST を指定します。

```
<configuration>
          :
   <fujitsu.cobol>
       <runtime>
          <sqlSettings>
              <serverList>
                 <server name="SERVER1" type="odbc" description="SERVER1">
                    <add key="@SQL_DATASRC" value="DataSourceName"/>
                    <add key="@SQL_COMMIT_MODE" value="ENLIST"/>
          :
```

NCOBU01196

XML ドキュメントコメント

NetCOBOL for .NET では、COBOL ソースプログラム内に XML ドキュメントコメントを記述することができます。

XML ドキュメントコメントは *>> で始まる行内注記のコメントフィールドに XML 要素を配置することで XML ドキュメントコードを作成できます。

記述例:

```
*>> <summary>
*>> Sample
*>> </summary>
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-1 AS "MyLibrary.MyClass1".
```

正書法が固定形式および可変形式の場合、XML ドキュメントコメントは一連番号域(カラム 1～6)および標識領域(カラム 7)から記述できません。 XML ドキュメントコメントが標識領域から記述されたとき、通常のコメント行と解釈されます。

XML ドキュメントコメントは以下の場所に記述することができます。

- プログラム定義の見出し部(省略時は PROGRAM-ID 段落)の直前の行
- クラス定義の見出し部(省略時は CLASS-ID 段落)の直前の行
- インターフェイス定義の見出し部(省略時は INTERFACE-ID 段落)の直前の行
- メソッド定義の見出し部(省略時は METHOD-ID 段落)の直前の行
- デリゲート定義の見出し部(省略時は DELEGATE-ID 段落)の直前の行
- ENUM 定義の見出し部(省略時は ENUM-ID 段落)の直前の行
- OBJECT 段落や STATIC 段落の作業場所節に記述された基本データ項目(01 および 77 レベルのデータ項目)の直前の行

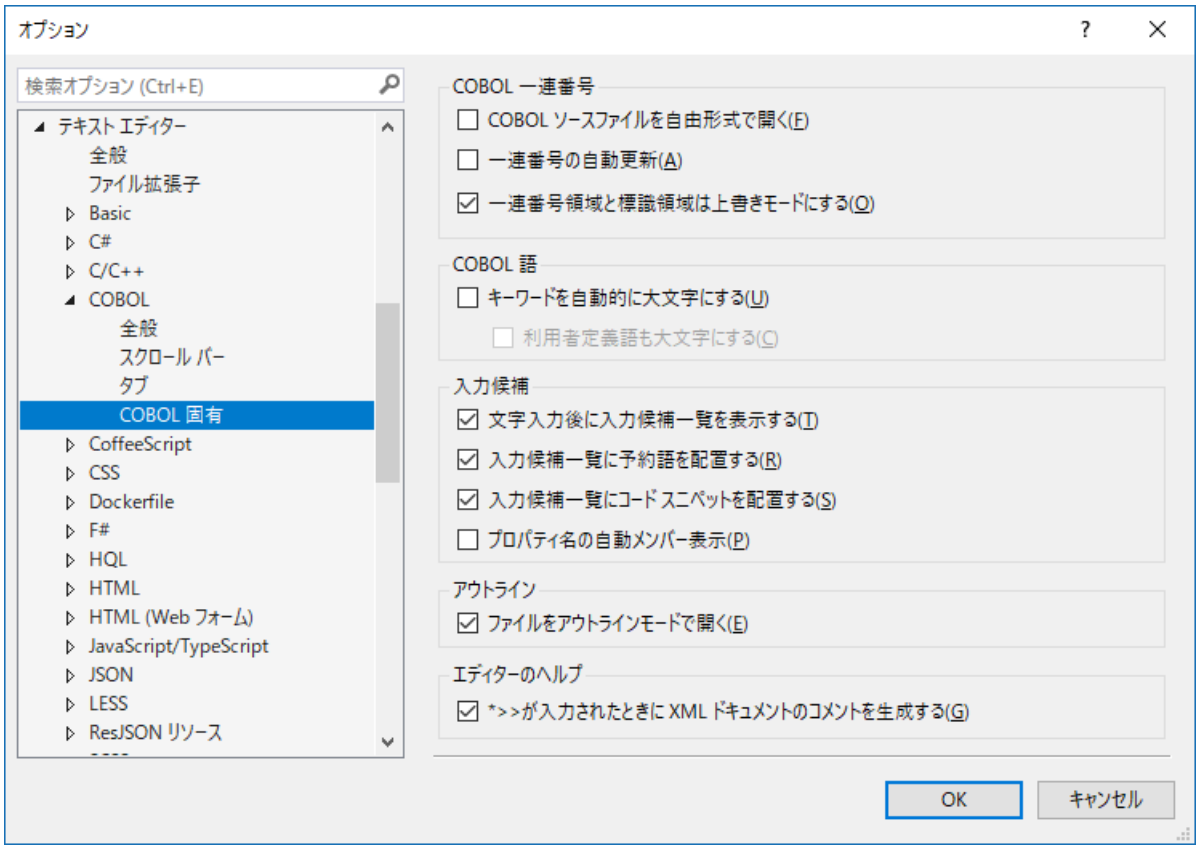
- XML ドキュメントファイルを作成する場合、プライベートメンバーや内部プログラムの XML ドキュメントコメントは出力されません。
- 部分型で、それぞれの型に XML ドキュメントコメントを記述している場合は、マージされた内容が XML ドキュメントファイルに出力されます。
- 同一のプロパティ名のプロパティメソッドの定義で、 GET PROPERTY メソッドと SET PROPERTY メソッドの両方に XML ドキュメントコメントを記述している場合は、 マージされた内容が、XML ドキュメントファイルに出力されます。

cobolc コマンドに/doc オプションを指定してコンパイルすると、 ソースコード内の XML ドキュメントコメントを検索して XML ドキュメントファイルを作成します。

XML ドキュメントファイルは出力ファイル(.exe ファイルまたは.dll ファイル)と同じフォルダに、 出力ファイルのベース名に.xml の拡張子を付けたファイル名で作成されます。

型およびメンバーに XML ドキュメントコメントを追加する

[オプション] ([ツール] メニュー) ダイアログボックスの左ペインから、[テキスト エディター] - [COBOL] - [COBOL 固有]を選択することで表示される[COBOL 固有]プロパティページにおいて、[*>>が入力されたときに XML ドキュメントのコメントを生成する]を選択している場合、 型およびメンバーの XML ドキュメントコメントの<summary>開始タグと終了タグを自動的に追加します。



手順:

1. コードエディターで、XML ドキュメントコメントを挿入する対象の型またはメンバーの上の行にカーソルを置きます。

2. *>> を入力します。 型やメンバーに対する XML ドキュメントコメントの<summary>開始タグと終了タグが追加されます。 メソッド定義やプログラム定義の場合は、手続き部に USING 句が記述されているときは、 パラメーター毎に<param>開始タグと終了タグが追加されます。 RETURNING 句が記述されている場合は<returns>開始タグと終了タグが追加されます。

3. XML ドキュメントコメントに適切な説明情報を追加します。 *>> で始まる XML ドキュメントコメント行が連続している場合、 XML ドキュメントコメントブロックと見なします。 XML ドキュメントブロックの途中に、XML ドキュメントコメント行を追加する場合も、先頭は*>>である必要があります。以下は、メソッド定義の例です。METHOD-ID の直前で「*>>」を入力すると、

```
*>>
METHOD-ID. METHOD-1 AS "SomeMethod".
DATA DIVISION.
LINKAGE SECTION.
01  L-TEXT PIC X(80).
01  R-VAL  BINARY-LONG.
PROCEDURE DIVISION USING L-TEXT RETURNING R-VAL.
```

XML ドキュメントコメントが自動的に展開されます。

```
     *>> <summary>
     *>>
     *>> </summary>
     *>> <param name="L-TEXT"></param>
     *>> <returns></returns>
METHOD-ID. METHOD-1 AS "SomeMethod".
DATA DIVISION.
LINKAGE SECTION.
01  L-TEXT PIC X(80).
01  R-VAL  BINARY-LONG.
ROCEDURE DIVISION USING L-TEXT RETURNING R-VAL.
```

なお、自動展開後にメソッドのパラメーターの個数や名前を変更しても、自動的に反映されません。 この場合は、手動で変更する必要があります。

注意事項

NetCOBOL for .NET の XML ドキュメントコメント機能は、 C#や Visual Basic の XML ドキュメントコメント機能の仕様に準拠します。

- XML Documentation Comments (C# Programming Guide)
- Documenting Your Code with XML (Visual Basic)

ただし、NetCOBOL for .NET ではいくつかの追記事項があります。

- コンパイラは cref 属性で指定されている型を検索するときに、 REPOSITORY 段落に記述した型指定子(CLASS,INTERFACE,ENUM,DELEGATE)を考慮します。
- エディターは XML ドキュメントコメント内の XML コードに対する IntelliSense 機能は対応していません。

以下のケースでは XML ドキュメントコメント機能は、XML ドキュメントコメントの自動展開および XML ドキュメントファイルの出力が正しく行われない場合があります。

- COBOL ソースプログラムに誤りがある場合。
- COBOL ソースプログラム(翻訳単位)に REPLACE 文が含まれる場合。 この場合、XML ドキュメントコメントの自動展開および XML ドキュメントファイルへの出力は行われません。
- ソースコードに COPY 文が含まれる場合、COBOL のコード要素の解析では COPY 文は単純に読み飛ばされます。 読み飛ばした結果、プログラムの解析ができなくなった場合は、 XML ドキュメントコメントの自動展開および XML ドキュメントファイルへの出力は行われません。 また、連絡節のデータ記述が不明となる場合は、XML ドキュメントファイルへの出力が意図どおりとならないことがあります。

なお、ビルド時に XML ドキュメントファイルの出力が正しく行われない場合でも、COBOL ソースコードに誤りがなければビルドは正常に終了します。

他の環境および言語との相互運用

ここでは、NetCOBOL for .NET アプリケーションを、他の環境で作成されたアプリケーションおよび他の.NET 言語で作成されたアプリケーションと、どのように相互運用していくのかについて説明します。

このセクションの内容

他の.NET 言語との相互運用

他の.NET 言語と相互に作用する方法について説明します。

.NET アプリケーションから COM モジュールへのアクセス

.NET は、優れた COM の相互運用機能を備えており、 相互運用の実装を助けるいくつかのユーティリティを提供しています。 これらのユーティリティの使用方法について説明します。

COBOL からアンマネージコードの呼出し

アンマネージコード(.NET 実行環境外部のコード) を呼び出すための、 .NET プラットフォーム呼出しサービスの使用方法について説明します。

Windows 版 NetCOBOL で作成したプログラムの呼出し

Windows 版 NetCOBOL で作成したプログラムの呼出しについて説明します。

COBOL データのマーシャリング

マーシャリングとは、スレッドやプロセス、ネットワークの境界を越えて 他のオブジェクトを使えるようにするメカニズムです。 ここでは、COBOL データのマーシャリングについて説明します。

NCOBU00527

他の.NET 言語との相互運用

ここでは、NetCOBOL for .NET アプリケーションを、他の.NET 言語で作成されたアプリケーションと相互運用する方法について説明します。

このセクションの内容

他の言語／環境との相互運用についての概要

他の言語／環境との相互運用の概要について説明します。

CLR のコード管理

共通言語ランタイム(Common Language Runtime)の役割について説明します。

.NET 環境でのアンマネージコード

アンマネージコードについて説明します。

.NET Framework クラスへのアクセス

.NET Framework で提供されるクラスの使用方法について説明します。

他のプログラミング言語から COBOL プログラム定義の呼出し

COBOL のプログラム定義を呼び出す場合の注意事項について説明します。

他の言語／環境との相互運用についての概要

複数の言語で記述されたアプリケーション部品が相互にやりとりする場合、さまざまな問題が発生します。直面する問題の大部分は、データ型の違い、データ型がコンポーネント間で渡される方法の違い、またはさまざま言語ランタイム間での衝突が中心でした。

.NET 対応の言語コンパイラが生成する共通中間言語コードは、Microsoft Intermediate Language("MSIL" や "IL")と呼ばれます。すべての.NET 対応のコンパイラは、MSIL を生成するため、言語間の相互運用性が非常に優れています。また、異なる言語で作成された.NET 向けのコンポーネントは、この環境でシームレスに接続できます。

.NET の提供する全ての機能は、MSIL 形式の基本クラス(システムクラスライブラリ)として提供されています。これらは、.NET に対応する多様な言語間で、シームレスに使用することができます。

.NET は、共通言語ランタイム(CLR)、および共通言語仕様(CLS)を備えています。

CLR は、.NET 対応の言語コンパイラによって生成されたすべての managed コードの管理、および実行の責任を持っています。CLR は、.NET アプリケーションと OS の間に位置し、それらの対話処理を管理します。

CLS は、.NET で作成されたコンポーネント部品に対する、最低限必要な外部条件を記述した仕様です。その中には、それらのコンポーネント、および基本データ型にアクセスする規則も備わっています。言語ベンダーが特定の外部インタフェースの規則、および基本データ型へのアクセスをサポートする限り、言語機能を実施する方法を決定する自由があるという意味では、非常に柔軟性の高い仕様です。

これら CLS への互換性を必要とする仕様部分は、言語モジュールが CLR と接続し、対話する技術であることを意味します。 同様に、CLS と CLR の組み合わせは、高度なレベルのプラットフォーム独立を提供します。.NET アプリケーションは、CLR をサポートしているすべてのプラットフォーム上で実行できます。

NCOBU00529

CLR のコード管理

共通言語ランタイム(CLR)は、従来の感覚の単純な言語ランタイムではありません。CLR は、プログラムによって呼び出される有用なルーチンを備えているだけでなく、アプリケーションの実ライフサイクル（翻訳、実行、およびセキュリティに関連した様々な特権の管理）の重要な一部です。さらに、CLR は、必要に応じて、より優れた実施を可能にするために、様々な言語コンパイラによって生成された固有の中間言語を、マシン特有のコードにコンパイルする、ジャストインタイム（JIT）コンパイラを管理します。CLR は、権限のない操作を実行しようとしたり、メモリリークが拡大していないかを確認するために、このコードを厳重に監視します。これは、CLR 内の大変優れたガベージコレクタの実施によって、実現します。 このガベージコレクタは、メモリヒープに含まれた非参照オブジェクトを時々チェックして、これらのオブジェクトが使用するメモリを解放します。

.NET 環境でのアンマネージコード

アンマネージコードとは、CLR を利用せずに機能することができる、CLR で管理されないプログラムコードです。たとえば、NetCOBOL for Windows コンパイラで作られたコードがこれに該当します。

アンマネージコードは、MSIL 形式ではなく、マシン特有のコード(ネイティブコード)です。

CLR は、マネージコードからアンマネージコードの呼出しをサポートしています。

詳細については、COBOL からアンマネージコードの呼出しを参照してください。

NCOBU00531

.NET Framework クラスへのアクセス

.NET Framework の最も役立つ、強力な点の 1 つに、開発者に提供された多くのクラス、関数、および API があげられます。この優れたフレームワークを最適に使用するには、従来の手続き型 COBOL プログラムではなく、クラスとしての COBOL プログラムを構造化する方法を学ぶ必要があります。 それは、従来の手続き型プログラムに数行を追加することでオブジェクト指向の振る舞いをするようになります。

以下にある簡単なサンプル、「HELLO WORLD」プログラムで検証してみます。

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
   DISPLAY "HELLO WORLD".
   EXIT PROGRAM.
```

上記のプログラムは、コンソール画面に文字列「HELLO WORLD」を表示します。このプログラムを、クラスを使ったプログラムに変更するには、以下のように行います。

```
IDENTIFICATION DIVISION.
CLASS-ID. HELLO.
ENVIRONMENT DIVISION.
STATIC.
PROCEDURE DIVISION.
METHOD-ID. MAIN.
DATA DIVISION.
PROCEDURE DIVISION.
   DISPLAY "HELLO WORLD".
   EXIT METHOD.
END METHOD MAIN.
END STATIC.
END CLASS HELLO.
```

さらに .NET Windows フォーム(GUI)アプリケーションになるように、プログラムを変更するには、以下のように行います。

```
CLASS-ID. HELLO INHERITS FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   PROPERTY WIN-TEXT AS "Text"
   CLASS APPLICATION AS "System.Windows.Forms.Application"
   CLASS FORM AS "System.Windows.Forms.Form".
STATIC.
PROCEDURE DIVISION.
METHOD-ID. MAIN.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 APP-OBJ USAGE OBJECT REFERENCE FORM.
PROCEDURE DIVISION.
   INVOKE HELLO "NEW" RETURNING APP-OBJ.
   SET WIN-TEXT OF APP-OBJ TO "HELLO WORLD".
   INVOKE APPLICATION "Run" USING BY VALUE APP-OBJ.
END METHOD MAIN.
END STATIC.
END CLASS HELLO.
```

これで、GUI フォームの作成、および文字列「HELLO WORLD」の表示のための変更がすべて終了しました。上記の変更は、以下の点に注目してください。

- クラス定義とリポジトリ段落の追加
- HELLO クラスのインスタンスを生成
- Text プロパティに「HELLO WORLD」文字列を設定

・　実行手続コードの追加

これらは、今後ビルドする、ほとんどの NetCOBOL for .NET アプリケーションにあてはまります。

NCOBU00532

他のプログラミング言語から COBOL プログラム定義の呼出し

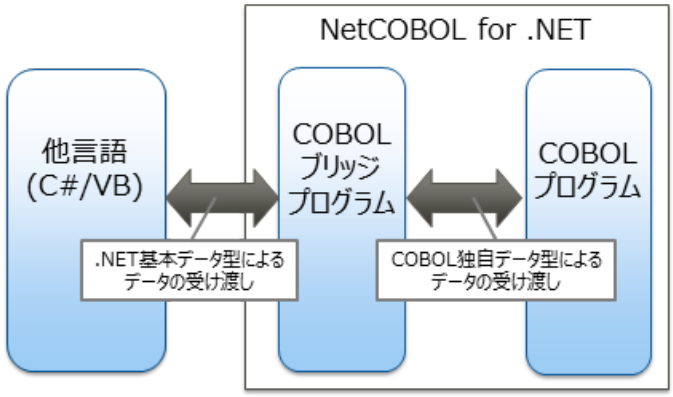
従来の手続き型 プログラム(COBOL のプログラム定義)を、CLR アセンブリにコンパイルするために、NetCOBOL for .NET の COBOL コンパイラを使用してビルドすることができます。

このプログラムは、COBOL プログラムから CALL 文によって呼び出すことができますが、他のプログラミング言語から 直接呼び出すことはできません。

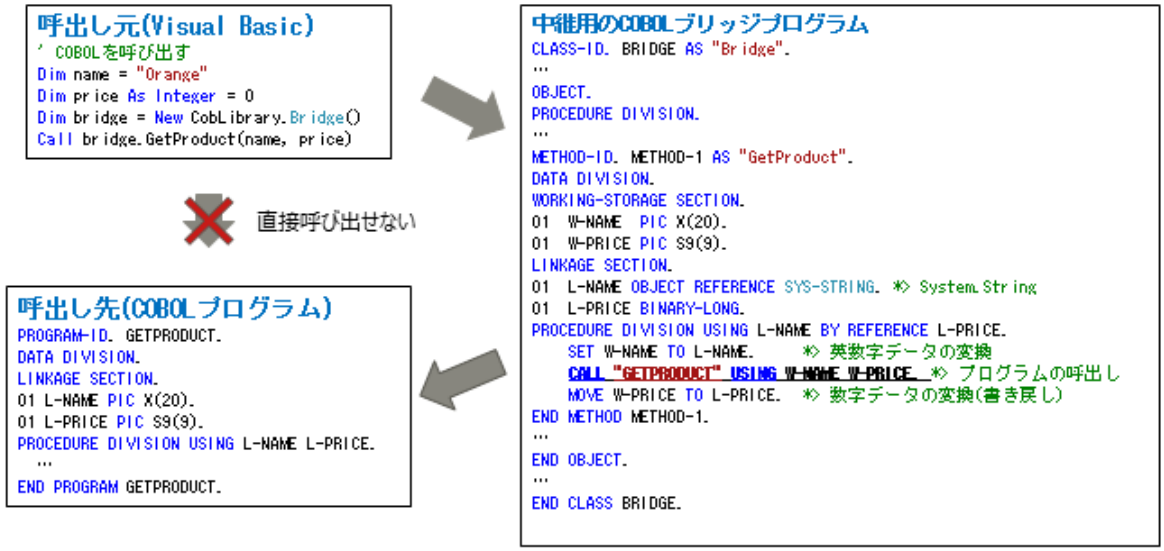
他のプログラミング言語から COBOL プログラムを呼び出したい場合は、オブジェクト指向 COBOL のクラス定義として COBOL プログラムを構造化するか、オブジェクト指向 COBOL によるブリッジプログラムを介して従来の COBOL プログラム定義を呼び出してください。

オブジェクト指向 COBOL によるブリッジプログラム

オブジェクト指向 COBOL によるブリッジプログラムは、 他のプログラミング言語(例えば、C#や Visual Basic)と COBOL の手続き型プログラムを中継し、 .NET 基本データ型と COBOL 独自データ型のデータの変換をすることで相互的なデータの受渡しを行ないます。



また、ブリッジプログラムをオブジェクト指向 COBOL で記述することによって、 他のプログラミング言語からは一般の.NET Framework クラスライブラリのように扱うことができます。



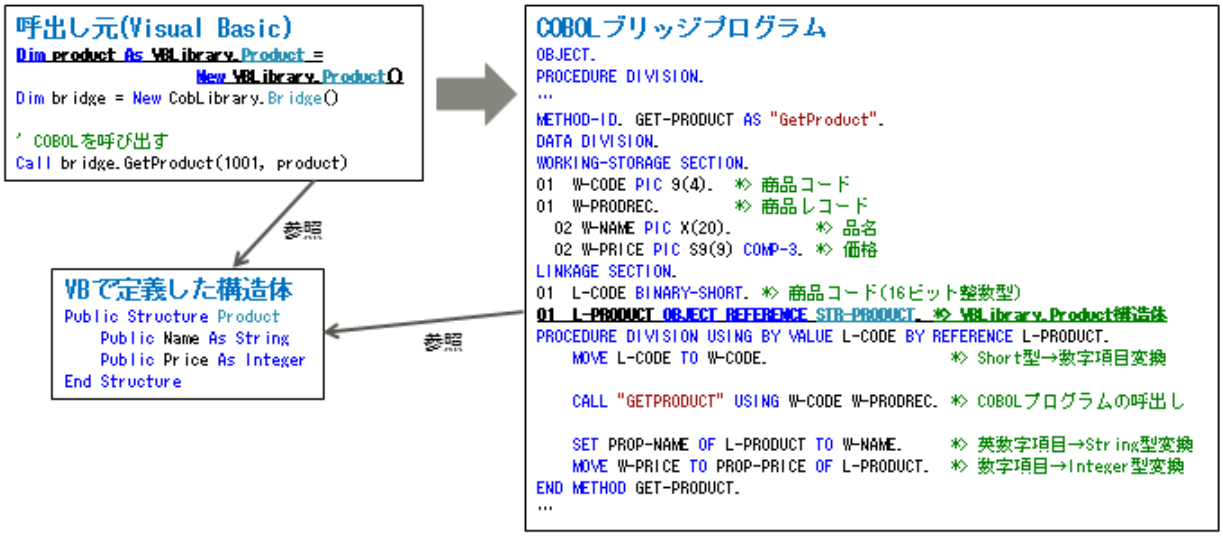
これは、Visual Basic から COBOL プログラムを呼び出すために、 ブリッジプログラムとしてオブジェクト指向 COBOL のメソッド定義を利用した例です。

この例では、ブリッジプログラムであるメソッド定義のパラメタで受け取った String 型のデータを COBOL の英

数字データ項目に変換して COBOL プログラムに渡しています。 また、COBOL プログラムから書き戻された数字データ項目をパラメタの Integer 型に変換して返しています。

COBOL プログラムのパラメタで集団項目を扱うケース

COBOL プログラムのパラメタで集団項目を扱う場合、 上記で説明した方法ではブリッジプログラムのパラメタ型として、 集団項目に対応したクラスや構造体を用意する必要があります。



しかし、集団項目の構造が複雑な場合やパラメタの数が多いような場合は、 オブジェクト指向 COBOL のクラス構造をそのままブリッジプログラムとして利用する方法もあります。

以下は集団項目のパラメタを持つ COBOL プログラムの例です。

呼出し先(COBOLプログラム)

```
PROGRAM-ID. GETSTOCK.
DATA DIVISION.
LINKAGE SECTION.
01  L-STOCKREC. *> STOCKレコード
  02  L-STOCKS OCCURS 10.
    03 L-NAME     PIC X(20).
    03 L-QUANTITY PIC S9(9).
01  L-COUNT PIC 99.
PROCEDURE DIVISION USING L-STOCKREC RETURNING L-COUNT.
  ...
END PROGRAM GETSTOCK.
```

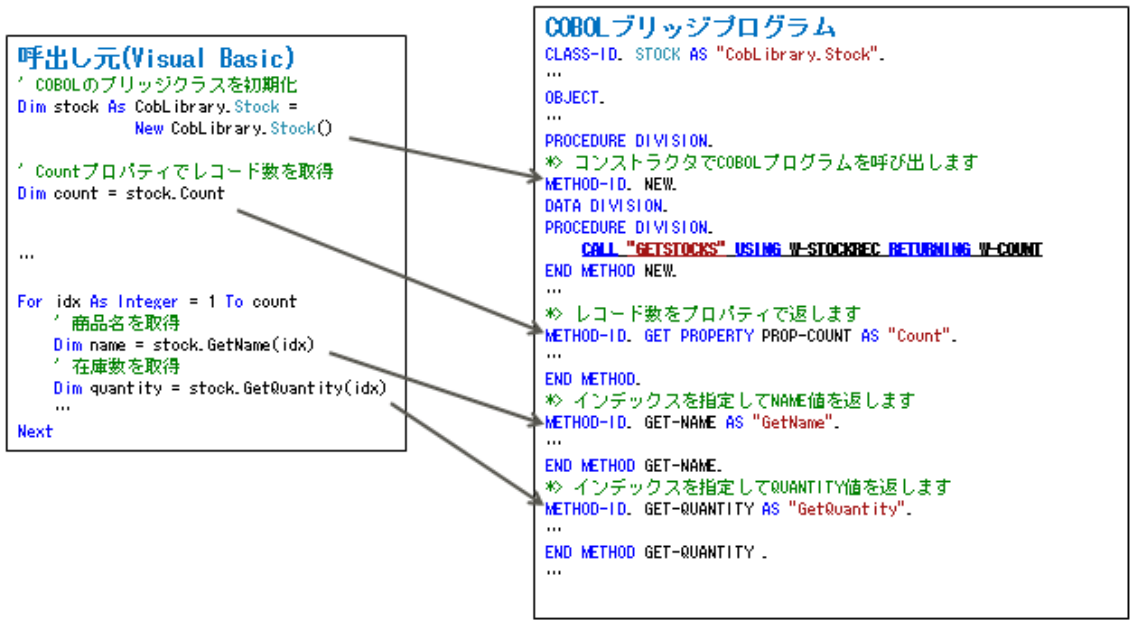
以下のブリッジプログラムの例では、集団項目はインスタンス定義のメンバーデータとして宣言し、 コンストラクタ等で COBOL プログラムを呼び出して集団項目に値を設定しています。

ブリッジプログラムをクラス構造として定義し、 各従属項目に対応したメソッドやプロパティを定義することで、他のプログラミング言語からデータのアクセスが容易になります。

COBOLブリッジプログラム

```
CLASS-ID. STOCK AS "CobLibrary.Stock".
…
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  W-STOCKREC. *> STOCKレコード
  02  W-STOCKS OCCURS 10.
    03 W-NAME     PIC X(20).
    03 W-QUANTITY PIC S9(9).
01  W-COUNT PIC 99.
…
PROCEDURE DIVISION.
*> コンストラクタでCOBOLプログラムを呼び出します
METHOD-ID. NEW.
DATA DIVISION.
PROCEDURE DIVISION.
    CALL "GETSTOCKS" USING W-STOCKREC RETURNING W-COUNT
END METHOD NEW.
…
*> レコード数をプロパティで返します
METHOD-ID. GET PROPERTY PROP-COUNT AS "Count".
DATA DIVISION.
LINKAGE SECTION.
01  RETVAL BINARY-LONG.
PROCEDURE DIVISION RETURNING RETVAL.
    MOVE W-COUNT TO RETVAL.
END METHOD.
*> インデックスを指定して値を返します
METHOD-ID. GET-NAME AS "GetName".
DATA DIVISION.
WORKING-STORAGE SECTION.
01  W-IDX PIC 99.
LINKAGE SECTION.
01  IDX BINARY-LONG.
01  RETVAL OBJECT REFERENCE SYS-STRING.
PROCEDURE DIVISION USING BY VALUE IDX RETURNING RETVAL.
    MOVE IDX TO W-IDX.
    SET RETVAL TO W-NAME( W-IDX ).
END METHOD GET-NAME.
…
```

以下は、Visual Basic から上記の COBOL ブリッジプログラム(クラス)のプロパティや メソッドを使用して COBOL のデータにアクセスする例を示しています。



.NET アプリケーションから COM モジュールへのアクセス

.NET プラットフォームが開発される以前は、 COM（Component Object Module）および後には COM+が言語に依存しない独立したアプリケーションモジュールを作成するための標準的な技術でした。 .NET は COM を置き換えていくことになるかも知れませんが、 現状では COM が Windows プラットフォームの大きな部分を占めています。 そのため、COM と.NET プリケーションを混在させることが可能でなければなりません。

.NET は、優れた COM との相互運用機能を備えており、 相互運用の実装を助けるいくつかのユーティリティを提供しています。 相互運用には二つの方向があります。 ひとつは COM コンポーネントが.NET コンポーネントやサービスを呼び出すというものであり、 もうひとつは.NET コンポーネントが COM コンポーネントやサービスを呼び出すというものです。

COM はバイナリ標準であり、.NET アプリケーションは CLR(共通言語ランタイム)を必要とします。 それに加えて、COM コンポーネントは Windows のレジストリに登録しなければならないという重大な違いがあります。また、COM モジュールは各モジュールで用いられるインタフェースやパラメタを記述した型ライブラリを必要とします。 これらは、 型情報を双方向に公開するための何らかのメカニズムが必要であることを意味しています。

.NET は、これらの問題を解決するために以下のユーティリティを提供しています。

ユーティリティ	説明
AxImp.exe	ActiveX コントロールを Windows Forms アプリケーションで利用できるようにするためのツールです。ActiveX コントロールの型ライブラリから Windows Forms コントロールとして利用できるプロクシを作成します。
RegAsm.exe	.NET アセンブリ内のメタデータをもとに Windows レジストリに必要な情報を設定し、COM クライアントが.NET クラスを COM コンポーネントとして利用できるようにするツールです。
TlbExp.exe	.NET アセンブリに定義された型を記述する型ライブラリを生成します。
TlbImp.exe	COM の型ライブラリに記述されている型を.NET アセンブリに変換するツールです。

COM コンポーネントを NetCOBOL for .NET で利用する場合の制限

COM コンポーネントを NetCOBOL for .NET で利用する場合、COM コンポーネントの使用には制限があります。次の条件を満たす COM コンポーネントのプロパティの参照は正しく処理できません。(注 1)

1. IDispatch を実装しており（つまり Automation をサポートしており）、その IDispatch インタフェースの型記述において、プロパティをメソッド形式ではなくプロパティ形式でしている COM コンポーネントに対して、以下の 2.または 3.の処理を行なう場合。
2. Microsoft Windows SDK 付属の tlbimp ツールを使ってラッパアセンブリを作成し、それを翻訳時に参照していた場合。
3. COBOL プロジェクトの「参照」に条件に該当する COM コンポーネントを追加した場合。

1. の条件に該当する COM コンポーネントの具体例としては、以下のものがあります。

- PowerBSORT OCX
- PowerCOBOL で作成した COM コンポーネント

ただし、問題となる COM コンポーネントが ActiveX コントロールである場合、3.の操作の代わりに、Windows Form 上に直接コントロールを貼り付ける場合に限り、この制限を回避することができます。

注 1: 診断メッセージ JMN5744I-S を出力して、翻訳処理を停止します。

このセクションの内容

Tlbexp.exe ユーティリティの使用

Tlbexp.exe(タイプ ライブラリ エクスポータ)は、.NET CLR アセンブリを記述する、型ライブラリを作成します。これにより、COM コンポーネントなどの型ライブラリを使用している、.NET 以外のアプリケーションから.NET アセンブリにアクセスできるようになります。

以下のコマンドは、アセンブリと同一名で、拡張子.tlb の付いた型ライブラリを生成します。

```
tlbexp cobhello.dll
```

以下のコマンドは、cobtl.tlb という名前の型ライブラリを生成します。

```
tlbexp cobhello.dll /out:cobtl.tlb
```

以下は、Tlbimp.exe を使用してインポートされたアセンブリを参照するアセンブリから、型ライブラリをエクスポートする例です。

1. まず、型ライブラリ、cobLib.tlb をインポートし、それを cobLib.dll として保存します。 インポートには、Tlbimp.exe を使用します。

   ```
   tlbimp cobLib.tlb /out:cobLib.dll
   ```

2. 次に COBOL コンパイラを使って前の例で作成した cobLib.dll を参照する cobhello.dll を作成します。

   ```
   cobolc cobhello.cob /MAIN:COBHELLO /reference:cobLib.dll /out:cobhello.dll
   ```

3. 最後に tlbexp.exe を使用して、cobLib.dll を参照する cobhello.dll 用の型ライブラリを生成します。

   ```
   tlbexp cobhello.dll
   ```

Tlbexp.exe については、.NET Framework のドキュメントのタイプ ライブラリ エクスポーター (Tlbexp.exe) を参照してください。

NCOBU00535

RegAsm.exe ユーティリティの使用

RegAsm.exe ユーティリティは、.NET アセンブリからメタデータを読込み、Windows レジストリに COM 互換エントリを作成します。これにより、COM プログラムから.NET Framework アセンブリが使用できるようになります。

以下のコマンドは、cobhello.dll に含まれた、すべての公開クラスを登録します。

```
regasm cobhello.dll
```

以下のコマンドは、cobhello.reg を生成します。このファイルには、すべての必要なレジストリエントリが含まれています。 このコマンドは、レジストリを更新しません。生成された.reg ファイルは Windows レジストリにインポートできます。

```
regasm cobhello.dll /regfile:cobhello.reg
```

以下のコマンドは、cobhello.dll に含まれた、すべての公開クラスを登録し、型ライブラリである cobhello.tlb を生成、および登録します。この型ライブラリは cobhello.dll で定義した、すべての公開型の定義を含みます。

```
regasm cobhello.dll /tlb:cobhello.tlb
```

RegAsm.exe については、.NET Framework のドキュメントのアセンブリ登録ツール (Regasm.exe)を参照してください。

Tlbimp.exe ユーティリティの使用

COM コンポーネントを.NET で利用するためには、Tlbimp.exe ユーティリティを使用します。Tlbimp.exe は、COM コンポーネントからそれと同等なアセンブリを作成します。作成されたアセンブリは、.NET から通常のクラスとして利用することができます。

以下のコマンドは、cobhello.dll にある型ライブラリと同一名で、.dll という拡張子の付いたメタデータを生成します。

```
tlbimp cobhello.tlb
```

以下のコマンドは、cobhello2.dll という名前のメタデータを生成します。

```
tlbimp cobhello.tlb /out:cobhello2.dll
```

Tlbimp.exe の詳細については、.NET Framework のドキュメントのタイプ ライブラリ インポーター (Tlbimp.exe)を参照してください。

NCOBU00989

Windows 版 NetCOBOL の COM 機能で利用する際の注意点

NetCOBOL for .NET で作成されたクラスにインタフェースが実装されている時、そのクラスを Windows 版 NetCOBOL の COM 機能のアーリバインドクラスで利用する場合は、そのインタフェースを COM のデフォルトインタフェースとして公開する必要があります。

そのためには、COM 公開するクラスに以下のカスタム属性を指定する必要があります。

System.Runtime.InteropServices.ClassInterface 属性

COM 公開するクラスに明示的にインタフェースを実装するような場合、この属性に ClassInterface.None を指定することで、クラスに対するクラスインタフェースを暗黙に生成しなくなります。この属性を省略した場合は、暗黙的にクラスインタフェース(メンバーを持たないディスパッチインタフェース)が公開され、CoClass のデフォルトインタフェースとなります。

System.Runtime.InteropServices.ComDefaultInterface 属性

この属性は、COM 公開する既定のインタフェースを指定します。 (ComDefaultInterface 属性は、.NET Framework 2.0 で追加された属性です。ClassInterface 属性に ClassInterface.None を指定した場合、実装されているインタフェースが単一であれば、この属性は省略しても問題ありません。)

これらの属性が指定されていない場合は、クラスに明示的にインタフェース を実装したとしても、メンバーを持たない空のディスパッチインタフェースがデフォルトインタフェースとして公開されてしまうため、Windows 版 NetCOBOL で作成したプログラムから COM 公開されたクラスのメソッドやプロパティを参照すると JMN5505I-S の翻訳エラーが発生してしまいます。

以下は COM 公開するインタフェース定義を示しています。

```
*
* COM 公開インタフェース(ICobHello)の定義
*
 IDENTIFICATION  DIVISION.
 INTERFACE-ID. INTERFACE-1 AS "CobHelloLib.ICobHello"
    CUSTOM-ATTRIBUTE IS CA-GUID CA-COMVISIBLE CA-INTERFACETYPE.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
*    COM インタフェースとして公開するための IID を指定します
     CUSTOM-ATTRIBUTE CA-GUID CLASS CLASS-GUID USING "446accef-d3ba-4611-9236-02aa30a10d9c"
*    COM 公開することを示します
     CUSTOM-ATTRIBUTE CA-COMVISIBLE CLASS CLASS-COMVISIBLE USING B"1"
*    デュアルインタフェースであることをマークします
     CUSTOM-ATTRIBUTE CA-INTERFACETYPE CLASS CLASS-INTERFACETYPE
                                 USING PROP-INTERFACEISDUAL OF ENUM-COMINTERFACETYPE
     .
 REPOSITORY.
     CLASS CLASS-GUID AS "System.Runtime.InteropServices.GuidAttribute"
     CLASS CLASS-COMVISIBLE AS "System.Runtime.InteropServices.ComVisibleAttribute"
     CLASS CLASS-INTERFACETYPE AS "System.Runtime.InteropServices.InterfaceTypeAttribute"
     ENUM ENUM-COMINTERFACETYPE AS "System.Runtime.InteropServices.ComInterfaceType"
     PROPERTY PROP-INTERFACEISDUAL AS "InterfaceIsDual"
     .
 PROCEDURE DIVISION.
* Hello メソッドの定義
 METHOD-ID. Hello AS "Hello".
 DATA DIVISION.
 LINKAGE SECTION.
 PROCEDURE DIVISION.

 END METHOD Hello.
 END INTERFACE INTERFACE-1.
```

上記のインタフェース定義をデフォルトインタフェースに指定して、COM 公開するクラス定義を以下に示します。

```
*
* COM 公開クラス(CobHello)の実装
*
 IDENTIFICATION DIVISION.
 CLASS-ID. CLASS-1 AS "CobHelloLib.CobHello"
    CUSTOM-ATTRIBUTE IS CA-GUID CA-COMVISIBLE CA-CLASSINTERFACE CA-COMDEFAULTINTERFACE.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
*    COM クラスとして公開するための CLSID を指定します
     CUSTOM-ATTRIBUTE CA-GUID CLASS CLASS-GUID USING "4fb93edd-b959-4c8d-8fdf-bd6dad29d064"
*    COM 公開することを示します
     CUSTOM-ATTRIBUTE CA-COMVISIBLE CLASS CLASS-COMVISIBLE USING B"1"
*    クラスインタフェースが生成されないようにマークします。
     CUSTOM-ATTRIBUTE CA-CLASSINTERFACE CLASS CLASS-CLASSINTERFACE
                                  USING PROP-NONE OF ENUM-CLASSINTERFACETYPE
*    デフォルトインタフェースとなるインタフェース定義のの Type オブジェクトを指定します。
     CUSTOM-ATTRIBUTE CA-COMDEFAULTINTERFACE CLASS CLASS-COMDEFAULTINTERFACE
                                  USING TYPE OF INTERFACE-ICOBHELLO

     .
 REPOSITORY.
     INTERFACE INTERFACE-ICOBHELLO AS "CobHelloLib.ICobHello"
     CLASS CLASS-GUID AS "System.Runtime.InteropServices.GuidAttribute"
     CLASS CLASS-COMVISIBLE AS "System.Runtime.InteropServices.ComVisibleAttribute"
     CLASS CLASS-CLASSINTERFACE AS
           "System.Runtime.InteropServices.ClassInterfaceAttribute"
     CLASS CLASS-COMDEFAULTINTERFACE AS
           "System.Runtime.InteropServices.ComDefaultInterfaceAttribute"
     ENUM ENUM-CLASSINTERFACETYPE AS
           "System.Runtime.InteropServices.ClassInterfaceType"
     PROPERTY PROP-NONE AS "None"
     .
 OBJECT. IMPLEMENTS INTERFACE-ICOBHELLO.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.

* Hello メソッドの実装
 METHOD-ID. Hello AS "Hello".
 DATA DIVISION.
 LINKAGE SECTION.
 PROCEDURE DIVISION.
    DISPLAY "Hello".
 END METHOD Hello.

 END OBJECT.

 END CLASS CLASS-1.
```

なお、Windows 版 NetCOBOL のアーリバインドで利用可能なインタフェースは、ディスパッチインタフェースであるため、公開するインタフェースに、System.Runtime.InteropServices.InterfaceType 属性を指定する場合は、ディスパッチインタフェース、デュアルインタフェースのいずれかを指定してください。 (既定ではデュアルインタフェースです)

NCOBU00537

COBOL からアンマネージコードの呼出し

ここでは、プラットフォーム呼出しサービスを使用して、COBOL からアンマネージコードを呼び出すしくみについて説明します。

このセクションの内容

プラットフォーム呼出しサービスの必要性

.NET プラットフォーム内で動作している場合、 .NET システムは実行しているメソッドや使用されているデータの情報を取り出すことができ、 メソッドの呼出し関係を管理することができます。 しかし、 .NET プラットフォームの外部のコード(アンマネージコード)を呼び出した場合、 .NET システムはこのコードやデータの情報にアクセスすることはできません。

アンマネージコードへのアクセスを可能にするため、 .NET はマネージコードから非.NET プログラムや API を呼び出すためのプラットフォーム呼出し(PINVOKE)サービスを提供しています。 プラットフォーム呼出しの機能については、.NET Framework のドキュメントのプラットフォーム呼び出しの詳細を参照してください。

プラットフォーム呼出しを利用するには、 プログラム原型定義（手続きを持たないプログラム定義）を記述します。 プログラム原型定義は、DLL ファイルの名前やパラメタのデータ型など、 アンマネージコードを呼び出すために必要な情報を.NET システムのために記述します。 プログラム原型定義を CALL すると、 プラットフォーム呼出しサービスがそれをアンマネージコード呼出しに変換して、 目的のアンマネージコードを呼び出すことができます。

NCOBU00539

データマーシャリング

プラットフォーム呼び出しの機能のひとつに、データマーシャリングがあります。 これは、 .NET プラットフォームとアンマネージコードの間で形式が異なるデータを適切に変換する機能です。.NET データ型は CLR が変換します。 COBOL 独自データ型は COBOL データマーシャラーがマーシャリングします。 プログラム原型定義に記述する設定によって、 どのようにデータをマーシャリングするかを選択することができます。 詳細については、COBOL データのマーシャリングを参照してください。

DLLImportAttribute クラス

DLLImportAttribute クラスはプログラム原型定義に指定するカスタム属性です。 このクラスのコンストラクタとメンバーは、プラットフォーム呼び出しに必要な情報を記述します。DLLImportAttribute クラスの使い方は、プログラム原型定義のサンプルと COBOL データのマーシャリングを参照してください。

DLLImportAttribute コンストラクタのパラメタは、呼び出されるアンマネージコードを含む.DLL の名前です。NetCOBOL for .NET では、 CUSTOM-ATTRIBUTE 句の USING 指定に DLL 名を定数で記述することによって指定します。

DLLImportAttribute クラスのメンバーは、使用する呼出し規約、 string オブジェクトのマーシャリングで用いる文字の符号化などの情報をプラットフォーム呼び出しに提供します。

このクラスの詳細については、 .NET Framework のドキュメントの DllImportAttribute クラスを参照してください。

NCOBU00541

プログラム原型定義

ここでは、プログラム原型定義について説明します。プログラム原型定義に定義されたアンマネージコードを呼び出したい場合は、このプログラム原型定義を呼び出します。詳細については、以下のサンプルで説明します。

プログラム原型定義のサンプル

以下は、USER32.DLL に含まれるネイティブ Windows API 関数 MessageBox のインタフェースを定義するプログラム原型定義のサンプルです。MessageBox はダイアログを作成し、パラメタで指定されたテキストメッセージを表示します。

```
PROGRAM-ID. MESSAGEBOX AS "MessageBox"
          IS PROTOTYPE CUSTOM-ATTRIBUTE IS PINVOKE.                        *>[1]
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.

   CUSTOM-ATTRIBUTE PINVOKE                                               *>[2]
      CLASS DLLIMPORT USING "USER32.DLL"                                  *>[3]
         PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION  *>[4]
         PROPERTY CHARSET IS ANSI OF E-CHARSET.                          *>[5]

REPOSITORY.
   CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute" *>[6]
   ENUM E-CALLINGCONVENTION
     AS "System.Runtime.InteropServices.CallingConvention"                 *>[7]
   PROPERTY P-CALLINGCONVENTION AS "CallingConvention"                     *>[8]
   PROPERTY STDCALL AS "StdCall"                                           *>[9]
   PROPERTY CHARSET AS "CharSet"                                           *>[9]
   ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"              *>[7]
   PROPERTY ANSI AS "Ansi"                                                 *>[9]
   CLASS SYS-STRING AS "System.String".                                    *>[10]
DATA DIVISION.

LINKAGE SECTION.
01 HWND USAGE BINARY-LONG.                                               *>|
01 MESSAGE-TEXT OBJECT REFERENCE SYS-STRING.                              *>|[11]
01 CAPTION-TEXT OBJECT REFERENCE SYS-STRING.                              *>|
01 MSGBOX-TYPE USAGE BINARY-LONG.                                         *>|
PROCEDURE DIVISION USING BY VALUE HWND MESSAGE-TEXT
   CAPTION-TEXT MSGBOX-TYPE.                                             *>[12]
END PROGRAM MessageBox.
```

[1]: プログラム名段落には以下を記述します。

- プログラム名は"MessageBox"です。

  これは、次の 2 つの目的で使用されます。まず、マネージコード内でアンマネージコードを呼び出したい場合、この名前を使います。次に、この名前をアンマネージ DLL 内でのエントリポイントとみなします。

  DllImportAttribute カスタム属性に EntryPoint フィールドが定義されない場合、エントリポイントとしてプログラム名が用いられます。このデフォルトの動作を利用すると、記述するコードがアンマネージコードを直接呼び出す場合と同様になるので便利です。

  DllImportAttribute の EntryPoint フィールドを用いてエントリポイントを指定したい場合は、以下のように特殊名段落の CUSTOM-ATTRIBUTE 句に以下の指定を記述します。

  ```
  PROPERTY EntryPoint IS "<entry point name>"
  ```

- PROTOTYPE 指定を記述します。

  この指定は、これがプログラム原型定義であることを示します。

- CUSTOM-ATTRIBUTE 句を指定します。

これは、特殊名段落の CUSTOM-ATTRIBUTE 句で定義したカスタム属性を参照します。ここでは、プラットフォーム呼出し(PINVOKE)に必要な情報を記述していることから"PINVOKE"という名前にしていますが、この名前である必要はありません。

[2]: 特殊名段落に CUSTOM-ATTRIBUTE 句を記述します。CUSTOM-ATTRIBUTE 句には、設定したいカスタム属性の内容を定義します。"PINVOKE"はカスタム属性名です。カスタム属性性名はプログラム名段落で参照されます。

[3]: CUSTOM-ATTRIBUTE 句の CLASS 指定はカスタム属性のクラスを指定します。この場合は、"DLLIMPORT"を指定しています。この名前は内部名と呼ばれ、リポジトリ段落で外部名"DllImportAttribute"と対応付けられています。そのため、この場合はカスタム属性のクラスとして、DllImportAttribute クラスを指定したことになります。

CUSTOM-ATTRIBUTE 句の USING 指定はカスタム属性クラスのコンストラクターに渡すパラメタを記述します。 DllImportAttribute の場合、アンマネージプログラムを含む DLL ファイルの名前を指定します。 DllImportAttribute コンストラクターの詳細については、 .NET Framework のドキュメントの DllImportAttribute コンストラクターを参照してください。

[4]: PROPERTY P-CALLINGCONVENTION という記述は、 DllImportAttribute クラスの DllImportAttribute.CallingConvention フィールドの値を設定しています。 この場合、フィールドの型は列挙型です。 この値は列挙型の列挙子"STDCALL"を記述することによって設定しています。 ここには、 フィールドの名前として外部名"CallingConvention"ではなく内部名"P-CALLINGCONVENTION"を記述します。 外部名と内部名はリポジトリ段落の PROPERTY 指定子によって関連付けられています。

[5]: PROPERTY CHARSET という記述は、 同様に DllImportAttribute クラスの DllImportAttribute.CharSet フィールドを設定しています。

これらのフィールドがデフォルト値のままでよい場合、 これらのフィールドを設定する必要がないことに注意してください。

[6]: リポジトリ段落の CLASS DLLIMPORT という記述は、 "DLLIMPORT"がクラスの名前であるということを示し、 その外部名"System.Runtime.InteropServices.DllImportAttribute"と対応付けています。 この外部名は.NET Framework リファレンスのクラスライブラリにある DllImportAttribute クラスです。

[7]: リポジトリ段落の ENUM 指定子は、 DllImportAttribute フィールドに値を設定するために使用するふたつの列挙型の内部名の外部名を定義します。

[8]: PROPERTY P-CALLINGCONVENTION という記述は、 これがプロパティ名またはフィールド名であることを示し、 その内部名と外部名を関連付けています。

[9]: STDCALL、CHARSET、および ANSI の 3 つの PROPERTY 指定子は、 同様にプロパティ名またはフィールド名を定義しています。

[10]: MessageBox API の文字列パラメタとして System.String 型を使用するため、 リポジトリ段落で System.String 型を宣言します。

[11]: 全てのパラメタは.NET データ型として宣言します。 このように記述することで、 データのマーシャリングを.NET システムに任せることができます。 パラメタに COBOL 独自データ型が含まれる場合にどのように記述するかについての詳細は、COBOL データのマーシャリングを参照してください。

[12]: 手続き部の見出しにパラメタの順序とパラメタ渡しの方法を宣言します。 この場合は、値渡しです。

プログラム呼出しのサンプル

```
CLASS-ID. HELLO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS SYS-STRING AS "System.String"
   PROGRAM MESSAGEBOX AS "MessageBox".               *>[1]
STATIC.
PROCEDURE DIVISION.
METHOD-ID. MAIN.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 HANDLE USAGE BINARY-LONG.                          *>|
01 MESSAGE-TEXT OBJECT REFERENCE SYS-STRING.          *>|[2]
01 CAPTION-TEXT OBJECT REFERENCE SYS-STRING.          *>|
01 MSGBOX-TYPE USAGE BINARY-LONG.                     *>|
PROCEDURE DIVISION.
   MOVE 0 TO HANDLE
   SET MESSAGE-TEXT TO "Hello world!"
   SET CAPTION-TEXT TO "Calling Unmanaged Code"
   MOVE 0 TO MSGBOX-TYPE
   CALL MESSAGEBOX USING HANDLE MESSAGE-TEXT         *>[3]
      CAPTION-TEXT MSGBOX-TYPE
   END-CALL.
END METHOD MAIN.
END STATIC.
END CLASS HELLO.
```

[プログラムの説明]

[1]: プログラム原型を使用する場合、リポジトリ段落に PROGRAM 指定子を記述して、プログラム原型を宣言する必要があります。

[2]; パラメタは、呼び出すアンマネージプログラムに対応した.NET データ型で定義します。

[3]: アンマネージプログラムの呼出しは、通常の CALL 呼出しと似ています。通常の CALL ではプログラム名を定数で指定しますが、プログラム原型を通してプログラムを呼び出す場合はリポジトリ段落で宣言したプログラム名を指定します。

アンマネージコード呼び出しの注意事項

.NET アプリケーションではない、アンマネージのコードの中で使用する環境変数は、アプリケーション構成ファイルに指定しても有効になりません。以下の方法で指定する必要があります。

- SET コマンドでの設定
- OS のユーザ環境変数に設定

NCOBU00542

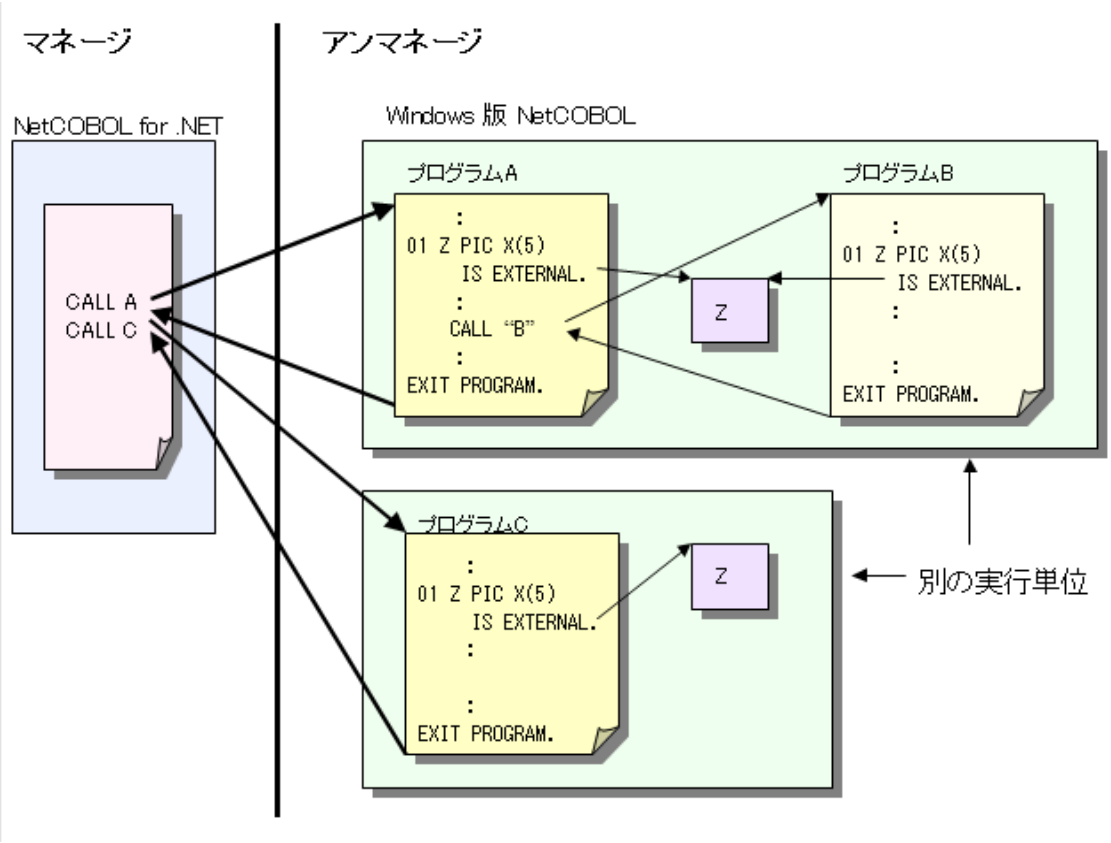
Windows 版 NetCOBOL で作成したプログラムの呼出し

ここでは、Windows 版 NetCOBOL で作成したプログラムの呼出しについて説明します。

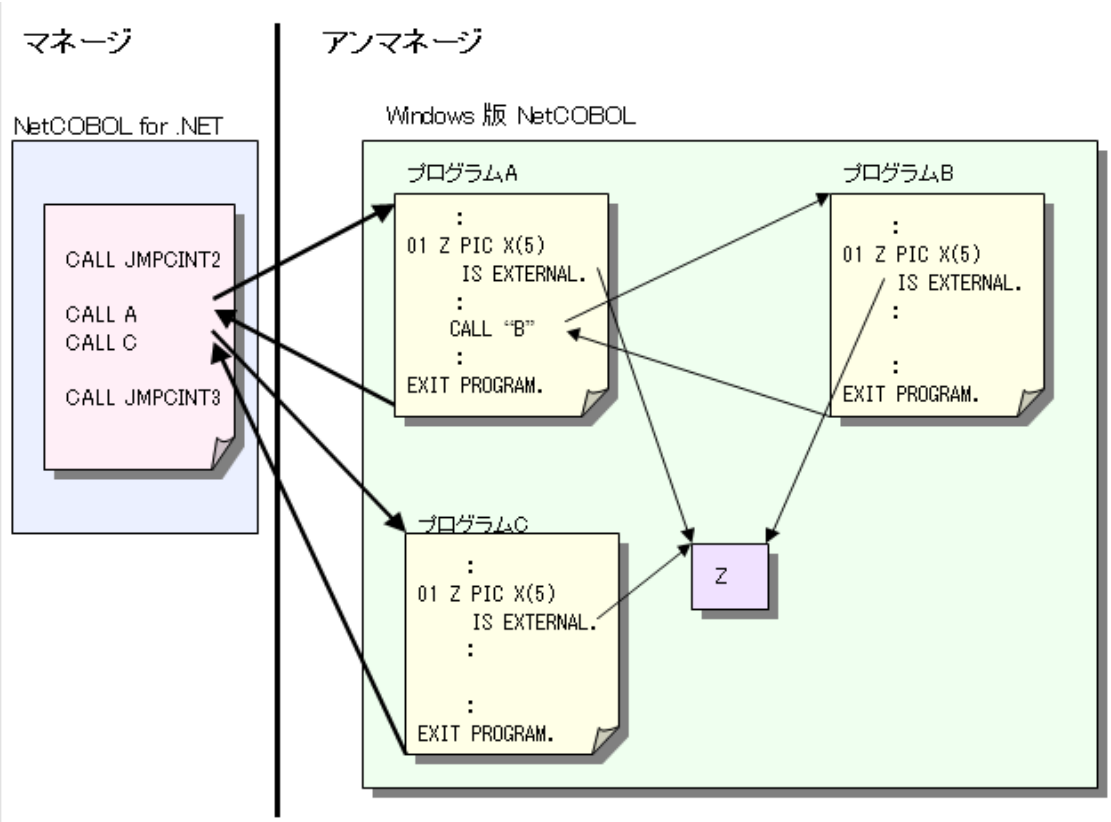
このセクションの内容

概要

NetCOBOL for .NET のプログラムから Windows 版 NetCOBOL のプログラムを呼び出す場合、プログラム原型定義を使用して呼び出します。 しかし、この呼出しは、呼出し毎に COBOL の単位が開始・終了します。そのため、複数の呼出しからなる処理を呼び出した場合、先の呼出しから次の呼出しまでの間が、別の実行単位になるため、スレッド単位で管理される資源は共用されません。たとえば、プログラム定義に宣言されたデータ（スレッド間共有外部データ/外部ファイルは除きます）は、スレッド単位で管理されます。このため、次のようなスレッド内共有外部データを持つプログラムは、正しく動作しません。 また、スレッド単位で管理される資源の獲得と解放が呼出し毎に行われるため、実行性能も劣化します。



これは、Windows 版 NetCOBOL が他言語連携用に提供している「JMPCINT2(実行単位の開始サブルーチン)」および「JMPCINT3(実行単位の終了サブルーチン)」を呼び出すことで解決できます。



Windows 版 NetCOBOL の実行環境、および、実行単位については、「Windows 版 NetCOBOL ユーザーズガイド」を参照してください。

JMPCINT2/JMPCINT3 サブルーチンの呼出し形式

JMPCINT2 サブルーチンの呼出し形式

呼出しの記述

```
CALL "[Fujitsu.COBOL.Subroutines.]JMPCINT2".
```

インタフェース

呼出し時にパラメタは必要ありません。

復帰コード

常に 0 が設定されます。

注意事項

ここでは、JMPCINT2(実行単位の開始サブルーチン)使用時の注意事項について説明します。

- 当サブルーチンを呼び出した場合、JMPCINT3(実行単位の終了サブルーチン)を必ず呼び出してください。
- 当サブルーチンを使用する場合は、“Fujitsu.COBOL.InteropServices.Win32.dll”を参照する必要があります。
- 名前空間“Fujitsu.COBOL.Subroutines”は省略できます。
- 当サブルーチンを使用する場合、対応するプラットフォームの「NetCOBOL Base Edition クライアント運用パッケージ」などがインストールされている必要があります。インストールされていない場合、例外(System.PlatformNotSupportedException)が発生します。

JMPCINT3 サブルーチンの呼出し形式

呼出しの記述

```
CALL "[Fujitsu.COBOL.Subroutines.]JMPCINT3".
```

インタフェース

呼出し時にパラメタは必要ありません。

復帰コード

常に 0 が設定されます。

注意事項

ここでは、JMPCINT3(実行単位の終了サブルーチン)使用時の注意事項について説明します。

- 当サブルーチンを呼び出す前に、JMPCINT2(実行単位の開始サブルーチン)を呼び出してください。
- 当サブルーチンを使用する場合は、“Fujitsu.COBOL.InteropServices.Win32.dll”を参照する必要があります。
- 名前空間“Fujitsu.COBOL.Subroutines”は省略できます。
- 当サブルーチンを使用する場合、対応するプラットフォームの「NetCOBOL Base Edition クライアント運用パッケージ」などがインストールされている必要があります。インストールされていない場合、例外(System.PlatformNotSupportedException)が発生します。

以下が、呼び出す側のプログラムのサンプルです。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
*    PROGRAM …        ここに呼び出すプログラムの名前を定義する
     .
PROCEDURE DIVISION.
*     :
     CALL "JMPCINT2".
     *> ここにプログラムの呼出しを記述する。
     CALL "JMPCINT3".
```

- 呼び出すプログラムのプログラム原型定義が必要です。
- 呼び出すプログラムの引数(データ)のマーシャリングについては、COBOL データのマーシャリングを参照してください。

注意事項

実行環境の注意事項

- 本製品は、Windows 版 NetCOBOL で作成したプログラムの動作に必要な、Windows 版の「NetCOBOL Base Edition クライアント運用パッケージ」などの環境を含んでいません。従って、これを呼び出すアプリケーションの動作には、環境となる製品が別途必要です。
- NetCOBOL for .NET の実行環境と、Windows 版 NetCOBOL の実行環境はそれぞれ独立しています。そのため、それぞれの環境が正しくなるように設定してください。
- 環境設定ファイルは NetCOBOL for .NET 用のものと、Windows 版 NetCOBOL 用のものがそれぞれ必要です。
- "JMPCINT2"と"JMPCINT3"の間に呼び出すプログラムの実行コード系(Unicode/シフト JIS)を統一してください。

環境変数の注意事項

NetCOBOL for .NET のプログラムから設定した環境変数は、Windows 版 NetCOBOL からは参照できません。引数として渡すなど、別の手段で伝達する必要があります。

IIS 配下で Windows 版 NetCOBOL のプログラムを呼び出す場合の注意事項

IIS 配下で実行される、WebForm や WebService から呼び出される Windows 版 NetCOBOL のプログラムは以下の注意が必要です。

- 翻訳オプション THREAD(MULTI)をつけて作成してください。

さらに、同じプロセス配下で実行されるサービスがある場合は以下の注意が必要です。

- 実行コード系(Unicode/シフト JIS)を統一してください。
- 呼出しの順序関係が保証できないので、"JMPCINT2"/"JMPCINT3"は呼び出さないでください。

NCOBU00546

COBOL データのマーシャリング

プラットフォーム呼び出しを用いてアンマネージコードを呼び出す場合、 .NET データ型のパラメタは.NET システムが適切にマーシャリングします。 一方、.NET システムには COBOL 独自データ型のパラメタをマーシャリングする機能はありません。 COBOL 独自データ型のマーシャリング機能は、 NetCOBOL for .NET が提供します。

多くの場合、マネージ COBOL プログラムとアンマネージ COBOL プログラムでは実行時文字コードが異なります。 COBOL 独自データ型のマーシャリングの主な目的は、 実行時文字コードを変換することです。

NetCOBOL for .NET では、プラットフォーム呼び出しを用いて、 従来の富士通 COBOL 製品で翻訳された Windows プラットフォーム向けプログラムの多くを呼び出すことができます。 しかし、Windows プラットフォーム向けプログラムのパラメタや復帰項目に以下のデータが含まれている場合、 そのプログラムはプラットフォーム呼び出しで呼び出すことができません。

- COBOL 独自データ型の復帰項目
- オブジェクト参照
- ポインタデータ項目
- 英字項目、英数字項目、英数字編集項目、日本語項目、日本語編集項目が REDEFINES 句によって再定義されているデータ
- 英字項目、英数字項目、英数字編集項目、日本語項目、日本語編集項目を含み、かつ、OCCURS DEPENDING ON 句が記述されている項目を含むデータ

以下では、 プラットフォーム呼び出しを用いて Windows プラットフォーム向けプログラムを呼び出す方法を、サンプルを用いて説明します。

- 単純なケース
- 文字コード系を指定する
- 特定のパラメタの文字コード変換を抑止する
- アンマネージ COBOL プログラムを呼び出す際の注意事項
- Windows 版 NetCOBOL V11.0 以降で作成したプログラムを呼び出す際の制限事項

単純なケース

以下は最も単純な場合のサンプルです。

翻訳オプション RCS の指定なし、または RCS(SJIS)を指定して翻訳された Windows 向け COBOL プログラムを呼び出すためのプログラム原型定義のサンプルです。

```
      PROGRAM-ID. NATIVE-METHOD AS "Native-Method" PROTOTYPE
                CUSTOM-ATTRIBUTE IS DLL-IMPORT.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
          CUSTOM-ATTRIBUTE DLL-IMPORT
            CLASS DLL-IMPORT-ATTRIBUTE USING "Win32app.dll".
      REPOSITORY.
          CLASS DLL-IMPORT-ATTRIBUTE
            AS "System.Runtime.InteropServices.DllImportAttribute"
     *     :
           .
      END PROGRAM NATIVE-METHOD.
```

これはプログラム原型定義で説明したアンマネージコードのためのプログラム原型定義そのままです。特に設定されていない場合、NetCOBOL for .NET はアンマネージ COBOL プログラムの実行時文字コード系が ANSI コードページ(日本語版 Windows の場合、これはシフト JIS と同じです)であるとみなしてマーシャリングを行います。

マネージコードからこのアンマネージ COBOL プログラムを呼び出す場合は COBOL からアンマネージコードの呼出しで説明されているように、プログラム原型を呼び出す形式で CALL 文を記述します。

NCOBU00548

文字コード系を指定する

以下は、Windows 向け COBOL プログラムの実行時文字コード系を指定する場合のサンプルです。

これは翻訳オプション RCS(UTF8-UCS2)を指定して翻訳された Windows 向け COBOL プログラムを呼び出すためのプログラム原型定義のサンプルです。

以下のプログラムを翻訳するには、参照するアセンブリファイルとして"Fujitsu.COBOL.InteropServices.Win32.dll"を指定します。

```
      PROGRAM-ID. NATIVE-METHOD AS "Native-Method" PROTOTYPE
                CUSTOM-ATTRIBUTE IS DLL-IMPORT RUNTIME-ENCODING.              *>[1]
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SPECIAL-NAMES.
          CUSTOM-ATTRIBUTE DLL-IMPORT
            CLASS DLL-IMPORT-ATTRIBUTE USING  "Win32App.dll"
          CUSTOM-ATTRIBUTE RUNTIME-ENCODING                                  *>[2]
            CLASS RUNTIME-ENCODING-ATTRIBUTE                                 *>[3]
               USING E-UNICODE-ZP OF RUNTIME-ENCODING-MODE.                  *>[4]
      REPOSITORY.
          CLASS DLL-IMPORT-ATTRIBUTE
            AS "System.Runtime.InteropServices.DllImportAttribute"
          CLASS RUNTIME-ENCODING-ATTRIBUTE                                     *>|
            AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingAttribute"  *>|[5]
          PROPERTY E-UNICODE-ZP AS "UnicodeZenkakuPadding"                     *>|
          ENUM RUNTIME-ENCODING-MODE                                           *>|
            AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode"
      *     :
          .
      END PROGRAM NATIVE-METHOD.
```

[プログラムの説明]

[1]: DLL-IMPORT カスタム属性に加えて、RUNTIME-ENCODING カスタム属性を参照しています。RUNTIME-ENCODING カスタム属性は特殊名段落で定義されています。

[2]: RUNTIME-ENCODING カスタム属性を定義しています。

[3]: RUNTIME-ENCODING カスタム属性の型は Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingAttribute です。ここでは、その型の内部名 RUNTIME-ENCODING-ATTRIBUTE を指定します。

[4]: RUNTIME-ENCODING カスタム属性のコンストラクタ引数には Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode 列挙型の UnicodeZenkakuPadding 列挙子を指定しています。この値がアンマネージ COBOL プログラムの実行時文字コードを表します。

[5]: RUNTIME-ENCODING カスタム属性の定義で参照されている型やフィールド名を宣言しています。

このサンプルのように、Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingAttribute カスタム属性をプログラム原型定義に設定することによって、呼び出すアンマネージ COBOL プログラムの実行時文字コードを指定することができます。実行時文字コードを表す Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode 列挙型には、以下の列挙子が定義されています。

Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode 列挙型の列挙子

列挙子	説明
ACP	アンマネージ COBOL プログラムの実行時文字コードは ANSI コードページです。日本語版 Windows の ANSI コードページはシフト JIS なので、日本語版 Windows 上では実行時文字コードがシフト JIS であるということと同じです。翻訳オプション RCS を指定せずに、または RCS(SJIS)を指

	定して翻訳したWin32向けCOBOLプログラムを呼び出す場合に使用します。
Unicode	アンマネージ COBOL プログラムの実行時文字コードは、英数字項目がUTF-8 符号化の Unicode、日本語項目がリトルエンディアン UTF-16 符号化の Unicode です。 ただし、Unicode 文字 U+0020（いわゆる半角空白）を日本語項目の空白とします。このモードは、通常、日本語版の COBOL では使用しません。
UnicodeZenkakuPadding	アンマネージ COBOL プログラムの実行時文字コードは、英数字項目がUTF-8 符号化の Unicode、日本語項目がリトルエンディアン UTF-16 符号化の Unicode です。 ただし、Unicode 文字 U+3000（いわゆる全角空白）を日本語項目の空白とします。RCS(UTF16)を指定して翻訳した Win32 向け COBOL プログラムを呼び出す場合に使用します。
UnicodeBE	アンマネージ COBOL プログラムの実行時文字コードは、英数字項目がUTF-8 符号化の Unicode、日本語項目がビッグエンディアン UTF-16 符号化の Unicode です。 ただし、Unicode 文字 U+0020（いわゆる半角空白）を日本語項目の空白とします。このモードは、通常、日本語版の COBOL では使用しません。
UnicodeBEZenkakuPadding	アンマネージ COBOL プログラムの実行時文字コードは、英数字項目がUTF-8 符号化の Unicode、日本語項目がビッグエンディアン UTF-16 符号化の Unicode です。 ただし、Unicode 文字 U+3000（いわゆる全角空白）を日本語項目の空白とします。RCS(UTF16,BE)を指定して翻訳した Win32 向け COBOL プログラムを呼び出す場合に使用します。
ACPTripleSized	アンマネージ COBOL プログラムの実行時文字コードは ANSI コードページです。日本語版 Windows の ANSI コードページはシフト JIS なので、日本語版 Windows 上では実行時文字コードがシフト JIS であるということと同じです。 ただし、アンマネージ COBOL プログラムを呼び出す際に、パラメタの長さを 3 分の 1 にします。 このモードの利用方法は、この後のアンマネージ COBOL プログラムを呼び出す際の注意事項を参照してください。

NCOBU00549

特定のパラメタの文字コード変換を抑止する

あるパラメタの文字コード変換を抑止したい場合があります。英数字項目に実際にはバイナリデータが入っている場合などです。このような場合にパラメタの文字コード変換を抑止するには、Fujitsu.COBOL.InteropServices.Win32.SuppressEncodingConversionAttribute カスタム属性を、プログラム原型定義ではなく、対象パラメタに設定します。

```
PROGRAM-ID. NATIVE-METHOD AS "Native-Method" PROTOTYPE
          CUSTOM-ATTRIBUTE IS DLL-IMPORT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE DLL-IMPORT
      CLASS DLL-IMPORT-ATTRIBUTE USING "Win32app.dll"
    CUSTOM-ATTRIBUTE SUPPRESS-CONVERSION                    *>[1]
      CLASS SUPPRESS-CONVERSION-ATTRIBUTE.
REPOSITORY.
    CLASS DLL-IMPORT-ATTRIBUTE
      AS "System.Runtime.InteropServices.DllImportAttribute"
    CLASS SUPPRESS-CONVERSION-ATTRIBUTE                     *>[2]
      AS "Fujitsu.COBOL.InteropServices.Win32.SuppressEncodingConversionAttribute".

DATA DIVISION.
LINKAGE SECTION.
01 PARAM1 CUSTOM-ATTRIBUTE IS SUPPRESS-CONVERSION.          *>[3]
  02 ITEM1 PIC X(16).
  02 ITEM2 PIC X(32).
PROCEDURE DIVISION USING PARAM1.
END PROGRAM NATIVE-METHOD.
```

[プログラムの説明]

[1]: カスタム属性 SUPPRESS-CONVERSION を定義しています。その型は Fujitsu.COBOL.InteropServices.Win32.SuppressEncodingConversionAttribute クラスです。

[2]: Fujitsu.COBOL.InteropServices.Win32.SuppressEncodingConversionAttribute クラスの内部名を定義しています。

[3]: 文字コード変換を抑止したいパラメタに SUPPRESS-CONVERSION カスタム属性を設定します。

アンマネージ COBOL プログラムを呼び出す際の注意事項

COBOL 独自データ型パラメタをマーシャリングする際に、 相手側の文字コードへ変換することができない文字があった場合、 NetCOBOL for .NET は Fujitsu.COBOL.InteropServices.Win32.InvalidCharException を発生させます。

文字コード変換の結果、 データのサイズが変わる場合があります。 したがって、 PICTURE 句で指定した書式の英数字編集項目または日本語編集項目のパラメタをマーシャリングすると、相手側の環境では PICTURE 句で指定した書式になっていない場合があります。

また、文字コード変換の結果、 データの長さが領域長よりも長くなった場合、 NetCOBOL for .NET は Fujitsu.COBOL.InteropServices.Win32.TextOverflowException を発生させます。

実行時文字コードがシフト JIS のアンマネージ COBOL の英数字項目に、 英数字以外のデータ(半角カタカナなど)が格納されている場合、 それをマネージ COBOL プログラムへマーシャリングすると文字サイズが大きく増加して TextOverflowException が発生することがあります。 たとえば、以下のようなアンマネージ COBOL プログラム（実行時文字コードはシフト JIS）があるとします。

```
PROGRAM-ID. NATIVE-METHOD AS "Native-Method".
DATA DIVISION.
LINKAGE SECTION.
01 PARAM1 PIC X.
PROCEDURE DIVISION USING PARAM1.
   MOVE "ｱ" TO PARAM1.
END PROGRAM NATIVE-METHOD.
```

これに対してマネージコードで以下のプログラム原型を定義して呼び出すと、 TextOverflowException が発生します。

```
PROGRAM-ID. NATIVE-METHOD AS "Native-Method" PROTOTYPE
          CUSTOM-ATTRIBUTE IS DLL-IMPORT RUNTIME-ENCODING.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE DLL-IMPORT
     CLASS DLL-IMPORT-ATTRIBUTE USING "Win32App.dll"
   CUSTOM-ATTRIBUTE RUNTIME-ENCODING
     CLASS RUNTIME-ENCODING-ATTRIBUTE
       USING E-ACP OF RUNTIME-ENCODING-MODE.
REPOSITORY.
   CLASS DLL-IMPORT-ATTRIBUTE
     AS "System.Runtime.InteropServices.DllImportAttribute"
   CLASS RUNTIME-ENCODING-ATTRIBUTE
     AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingAttribute"
   PROPERTY E-ACP AS "ACP"
   ENUM RUNTIME-ENCODING-MODE
     AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode".
DATA DIVISION.
LINKAGE SECTION.
01 PARAM1 PIC X.
PROCEDURE DIVISION USING PARAM1.
END PROGRAM NATIVE-METHOD.
```

これは、シフト JIS では 1 バイトであった半角カタカナが、 UTF-8(マネージコードの英数字項目の文字コード)では 3 バイトで表現されるため、 1 バイトの領域に収まりきれなくなったためです。

このような場合のために、 NetCOBOL for .NET では ACPTripleSized モードのマーシャリングを用意しています。

ACPTripleSized モードでは、 プログラム原型定義に記述された英字項目、英数字項目、英数字編集項目のパラメタは、 その長さの 3 分の 1 のアンマネージ COBOL の領域へマーシャリングされます。 そのため、英字項目、英数字項目、英数字編集項目のパラメタの長さは 3 の倍数でなければなりません。日本語項目、日本語編集項目は長さを変更しません。

たとえば、 上の例は以下のように ACPTripleSized モードを利用したプログラム原型定義を記述すれば、

TextOverflowException を発生させることなく半角カタカナを受け取ることができます。

```
PROGRAM-ID. NATIVE-METHOD AS "Native-Method" PROTOTYPE
        CUSTOM-ATTRIBUTE IS DLL-IMPORT RUNTIME-ENCODING.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE DLL-IMPORT
     CLASS DLL-IMPORT-ATTRIBUTE USING "Win32App.dll"
   CUSTOM-ATTRIBUTE RUNTIME-ENCODING
     CLASS RUNTIME-ENCODING-ATTRIBUTE
       USING E-ACP-TRIPLE OF RUNTIME-ENCODING-MODE.
REPOSITORY.
   CLASS DLL-IMPORT-ATTRIBUTE
     AS "System.Runtime.InteropServices.DllImportAttribute"
   CLASS RUNTIME-ENCODING-ATTRIBUTE
     AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingAttribute"
   PROPERTY E-ACP-TRIPLE AS "ACPTripleSized"
   ENUM RUNTIME-ENCODING-MODE
     AS "Fujitsu.COBOL.InteropServices.Win32.RuntimeEncodingMode".
DATA DIVISION.
LINKAGE SECTION.
01 PARAM1 PIC X(3).
PROCEDURE DIVISION USING PARAM1.
END PROGRAM NATIVE-METHOD.
```

PARAM1 に PIC X ではなく PIC X(3)と記述していることに注意してください。 アンマネージ COBOL プログラムを実際に呼び出す際には、PARAM1 はアンマネージ環境内の 1 バイトの領域としてマーシャリングされます。

ACPTripleSized モードでは、データ領域のサイズを変更するため、 パラメタを編集項目にする意味がありません。 そのため、全てのパラメタを長さが 3 の倍数の英字項目、英数字項目、日本語項目として記述してください。

たとえば、以下のようなアンマネージ COBOL プログラム（実行時文字コードはシフト JIS）があったとします。

```
PROGRAM-ID. NATIVE-METHOD AS "Native-Method".
DATA DIVISION.
LINKAGE SECTION.
01 PARAM1 PIC X(3)BX(2).
 01 PARAM2 PIC BN.
PROCEDURE DIVISION USING PARAM1 PARAM2.
*    :
END PROGRAM NATIVE-METHOD.
```

このアンマネージ COBOL プログラムを呼び出すための ACPTripleSized モードのプログラム原型定義は以下のように記述します。

```
PROGRAM-ID. NATIVE-METHOD AS "Native-Method" PROTOTYPE
          CUSTOM-ATTRIBUTE IS DLL-IMPORT RUNTIME-ENCODING.
DATA DIVISION.
LINKAGE SECTION.
01 PARAM1 PIC X(18).
 01 PARAM2 PIC N(6).
PROCEDURE DIVISION USING PARAM1 PARAM2.
END PROGRAM NATIVE-METHOD.
```

Windows版 NetCOBOL V11.0以降で作成したプログラムを呼び出す際の制限事項

以下の Windows プラットフォーム向けプログラムに対しては、COBOL 独自データ型のマーシャリングによる文字コード変換はサポートしていません。

- ENCODE オプションのサブオペランドに UTF32 を指定して翻訳されたプログラム
- パラメタのデータ項目に ENCODING 句を指定しているプログラム

NCOBU00551

リファレンス

ここでは、NetCOBOL for .NET でプログラミングする場合に、必要な参照情報をとりあげます。

このセクションの内容

言語リファレンス

COBOL 文法書の参照方法について説明します。

.NET データ型の COBOL 表現

COBOL データと.NET データ型の対応を説明します。

プロジェクト デザイナー

NetCOBOL for .NET プロジェクトに設定できるプロジェクトオプションについて説明します。

MSBuild タスク リファレンス (Fujitsu.COBOL.Build)

NetCOBOL for .NET が提供する MSBuild タスクについて説明します。

コンパイラオプション・翻訳オプション

NetCOBOL for .NET で、使用可能なコンパイラオプションについて説明します。

実行環境変数

NetCOBOL for .NET が使用する実行環境変数について説明します。

ファイル入出力状態一覧

FILE STATUS に返却された値について説明します。

組込み関数

NetCOBOL for .NET でサポートしている組込み関数を一覧で説明します。

SQL 情報

SQL に関連する情報(サーバ情報、デフォルトコネクション情報、コネクション有効範囲)について説明します。

埋込み SQL 文

埋込み SQL 文に関連する情報(埋込み SQL のキーワード一覧、埋込み SQL 文で使用可能なホスト変数、SQLSTATE/SQLCODE/SQLMSG)について説明します。

データ型の対応

データベース対応で扱うデータ対応について説明します。

制限事項・仕様変更

今回の NetCOBOL for .NET 製品のリリースにおける制限および仕様の変更について説明します。

NCOBU00552

言語リファレンス

COBOL 文法書は、PDF ファイルで提供されます。

[スタート] - アプリ一覧(*) - お使いの NetCOBOL for .NET 製品名 - [オンラインマニュアル]から参照できます。

*：ご使用の Windows によって、以下の操作をすることで同様の画面になります。

- Windows 7 : [スタート]メニュー - [すべてのプログラム]
- Windows 8.1 および Windows Server 2012 R2 : [スタート]画面 - [↓] - [アプリ]

.NET データ型の COBOL 表現

NetCOBOL for .NET での、 .NET 基本データ型と COBOL 言語としてのデータ型との対応を以下に示します。

.NET プラットフォーム対応の多くの言語では、 .NET 基本データ型をその言語の組み込みデータ型に対応付けています。 NetCOBOL for .NET では、 いくつかの.NET 基本データ型は非オブジェクトデータ型として扱いますが、 オブジェクトとして扱うデータ型もあります。

.NET 基本データ型と COBOL データ型の対応

.NET Framework での型名	COBOL 言語上の扱い	説明	CLS 準拠	値型
System.Byte	USAGE BINARY-CHAR UNSIGNED	8 ビット符号なし整数	Yes	Yes
System.SByte	"System.SByte"型オブジェクト参照	8 ビット符号付き整数	No	Yes
System.Int16	USAGE BINARY-SHORT SIGNED	16 ビット符号付き整数	Yes	Yes
System.Int32	USAGE BINARY-LONG SIGNED	32 ビット符号付き整数	Yes	Yes
System.Int64	USAGE BINARY-DOUBLE SIGNED	64 ビット符号付き整数	Yes	Yes
System.UInt16	"System.UInt16"型オブジェクト参照	16 ビット符号なし整数	No	Yes
System.UInt32	"System.UInt32"型オブジェクト参照	32 ビット符号なし整数	No	Yes
System.UInt64	"System.UInt64"型オブジェクト参照	64 ビット符号なし整数	No	Yes
System.Single	USAGE COMP-1	単精度浮動小数点	Yes	Yes
System.Double	USAGE COMP-2	倍精度浮動小数点	Yes	Yes
System.Boolean	"System.Boolean"型オブジェクト参照	ブール値（真または偽）	Yes	Yes
System.Char	PIC N	Unicode 文字(16 ビット)	Yes	Yes
System.Decimal	"System.Decimal"型オブジェクト参照	96 ビット 10 進値	Yes	Yes
System.IntPtr	"System.IntPtr"型オブジェクト参照	基になるプラットフォームによってサイズが決まる符号付き整数（32 ビットのプラットフォームでは 32 ビット値、64 ビットのプラットフォームでは 64 ビット値）	Yes	Yes
System.UIntPtr	"System.UIntPtr"型オブジェクト参照	基になるプラットフォームによってサイズが決まる符号なし整数（32 ビットのプラットフォームでは 32 ビット値、64 ビッ	No	Yes

		トのプラットフォームでは 64 ビット値)		
System.Object	"System.Object"型オブジェクト参照	.NET Framework のすべての型のルート	Yes	No
System.String	"System.String"型オブジェクト参照	Unicode 文字の文字列	Yes	No

表中にある CLS 準拠についての説明は CLS 準拠データ型を参照してください。 また、表中にある値型についての説明は参照型と値型を参照してください。

用途の BINARY-SHORT SIGNED, BINARY-LONG SIGNED, BINARY-DOUBLE SIGNED において、SIGNED は省略可能です。

データ型の詳細については、.NET 基本データ型を参照してください。

プロジェクト デザイナー

プロジェクトオプションは、プロジェクトのプロパティページで設定します。 プロパティページは、ソリューションエクスプローラーでプロジェクト名を右クリックし、コンテキストメニューから[プロパティ]を選択することで表示されます。

ここでは、NetCOBOL for .NET プロジェクト固有のページについて説明します。 他のページについては、Visual Studio のドキュメントの「プロジェクトのプロパティのリファレンス」を参照してください。

このトピックの内容

NCOBU01054

[アプリケーション] ページ (プロジェクト デザイナー)

[アプリケーション] ページでは、プロジェクトのアプリケーション設定およびプロパティを設定します。

アセンブリ名

アセンブリ マニフェストを格納する出力ファイルの名前を指定します。

既定の名前空間

新しい項目の追加やカスタムツールの実行で作成されるクラスの名前空間を指定します。

対象のフレームワーク

アプリケーションが対象とする .NET Framework のバージョンを指定します。 このオプションには、以下の値を指定できます。

- .NET Framework 4
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2
- .NET Framework 4.6
- .NET Framework 4.6.1
- .NET Framework 4.6.2
- .NET Framework 4.7
- .NET Framework 4.7.1

詳細については、Visual Studio のドキュメントの「方法 : 特定の .NET Framework を対象にする」および「Visual Studio のマルチ ターゲットの概要」を参照してください。

対象のフレームワークを変更した場合、プロジェクトで参照しているアセンブリが設定後のフレームワークをサポートしていることを確認し、必要ならば適切なバージョンのアセンブリを参照するように再設定してください。

出力の種類

ビルドするアプリケーションの種類を指定します。 このオプションには、以下の値を指定できます。

- Windows アプリケーション
- コンソール アプリケーション
- クラス ライブラリ

詳細については、/target コンパイラオプションを参照してください。

スタートアップ オブジェクト

アプリケーションの読込み時に呼び出されるエントリポイントを指定します。 詳細については、/main コンパイラオプションを参照してください。

アセンブリ情報

.NET Framework グローバルアセンブリ属性の値を指定するための[アセンブリ情報] ダイアログ ボックスを表示します。 グローバルアセンブリ属性は、プロジェクトと一緒に自動的に作成される AssemblyInfo.cob ファイルに格納されます。 詳細については、Visual Studio のドキュメントの「[アセンブリ情報] ダイアログ ボックス」を参照してください。

アイコン

アプリケーションのアイコンとして使用されるアイコンファイル(.ico)を指定します。 詳細については、/win32icon コンパイラオプションを参照してください。

マニフェスト

独自のアプリケーションマニフェストを指定する場合に、マニフェストファイルを指定します。 このオプションには、以下の値を指定できます。

- マニフェストを既定の設定で埋め込みます

  要求実行レベル（requestedExecutionLevel）に"asInvoker"を指定して、アプリケーションの実行ファイルにマニフェスト情報を埋め込みます。

- マニフェストなしでアプリケーションを作成します

  マニフェストファイルをアプリケーションの実行ファイルと同じフォルダに配置する場合や仮想化に従う場合に指定します。

- ＜マニフェストファイル＞

  アプリケーションの実行可能ファイルに独自のマニフェストを埋め込む場合に、プロジェクトに追加されているマニフェストファイルを指定します。

詳細については、/win32manifest コンパイラオプションおよび/nowin32manifest コンパイラオプションを参照してください。

リソース ファイル

独自の Win32 リソースを埋め込む場合に、Win32 リソースファイル(.res)を指定します。 詳細については、/win32res コンパイラオプションを参照してください。

NCOBU01055

[ビルド] ページ (プロジェクト デザイナー)

[ビルド] ページでは、プロジェクトのビルド構成プロパティを設定します。 ビルド構成プロパティは、各構成およびプラットフォームごとに設定することができます。

構成

このページで表示または変更する構成を指定します。 既定では、以下の値が指定できます。

- アクティブ (Debug)
- Debug
- Release
- すべての構成

プラットフォーム

このページで表示または変更するプラットフォームを指定します。 既定の指定は、[アクティブ (Any CPU)] です。

追加オプション

コンパイラで使用される追加のオプションを指定します。 ここで指定された追加オプションは、他のオプションよりあとに追加され、COBOL コンパイラに渡されます。 このオプションには、/wc コンパイラオプションを使用して、翻訳オプションを指定することができます。 翻訳オプションについては、/wc コンパイラオプションおよび翻訳オプション - アルファベット順一覧を参照してください。

追加オプションへ応答ファイル（@コンパイラオプション）を指定している場合は無視されます。 Cobolc コマンドまたは Cobolc タスクを使用して応答ファイルを指定してください。

プラットフォーム ターゲット

出力ファイルがターゲットとするプロセッサの種類を指定します。 利用可能なプロセッサの種類として、以下の値を指定できます。

- Any CPU
- x86
- x64

詳細については、/platform コンパイラオプションを参照してください。

32 ビットの優先

このチェックボックスを選択した場合、プラットフォーム ターゲットとして"Any CPU"が指定されているとき、32 ビットプラットフォームを優先します。64 ビットと 32 ビットの両方のアプリケーションをサポートするシステムでは、32 ビットプラットフォームを優先するファイルを出力します。

詳細については、/platform コンパイラオプションの anycpu32bitpreferred を参照してください。

このオプションは、対象のフレームワークが.NET Framework 4.5 以降である場合、かつ、出力の種類が"Windows アプリケーション"または"コンソール アプリケーション"の場合のみ有効です。

出力パス

出力ファイルの場所を指定します。

デバッグ情報

デバッグに必要な情報を出力します。このオプションには以下の値を指定できます。

- none
- full
- pdb-only
- minimal

full、pdb-only、または minimal を指定した場合、 COBOL コンパイラはデバッグ情報を含むデバッグデータベースファイル(.pdb)ファイルを生成します。詳細については、/debug コンパイラオプションを参照してください。

対象のフレームワークに、.NET Framework 4 より前のバージョンを指定している場合は、 pdb-only または minimal を指定しても、full が指定されたものとみなします。

リストファイルの生成

翻訳リストファイルの生成を指定します。 このオプションを選択した場合、COBOL コンパイラは翻訳リストファイル(.lst)を生成します。 詳細については、/print コンパイラオプションを参照してください。

XML ドキュメント ファイルの生成

XML ドキュメントファイルの生成を指定します。 このオプションを選択した場合、COBOL コンパイラは XML ドキュメントファイル(.xml)を生成します。 詳細については/doc コンパイラオプションを参照してください。

NCOBU01056

[ビルド イベント] ページ (プロジェクト デザイナー)

[ビルド イベント] ページでは、ビルドの開始前またはビルドの終了後に実行するコマンドを設定します。

ビルド前に実行するコマンドライン

ビルド開始前に実行する任意のコマンドを指定します。 長いコマンドを入力するには、[ビルド前の編集] をクリックして表示されるダイアログ ボックスを使用して入力します。

ビルド後に実行するコマンドライン

ビルド終了後に実行する任意のコマンドを指定します。 長いコマンドを入力するには、[ビルド後の編集] をクリックして表示されるダイアログ ボックスを使用して入力します。

ビルド後のコマンドラインの実行条件

ビルド後のイベントを実行する条件を指定します。 このオプションには、以下の値を指定できます。

- 常に行う

  ビルド後のイベントは、ビルドが成功したかどうかに関係なく実行されます。

- ビルドが成功したとき

  ビルド後のイベントは、ビルドが成功した場合に実行されます。このため、ビルドが成功した場合は、最新のプロジェクトについてもイベントが実行されます。

- ビルドがプロジェクト出力を更新したとき

  ビルド後のイベントは、コンパイラの出力ファイル (.exe または .dll) が以前のコンパイラの出力ファイルと異なる場合にだけ実行されます。 このため、ビルド後のイベントは、プロジェクトが最新の場合は実行されません。

[参照パス] ページ (プロジェクト デザイナー)

[参照パス] ページでは、プロジェクトで使用されるアセンブリ参照のフォルダのパスを設定します。 詳細については、参照の解決を参照してください。

フォルダ

[参照パス(ユーザー)]一覧または[参照パス(プロジェクト)]一覧に追加または更新を行うためのフォルダを指定します。

参照パス(ユーザー)

参照するアセンブリファイルのあるフォルダの一覧です。 各環境で参照するアセンブリのフォルダが異なる場合に、この一覧にフォルダを追加します。 この一覧は、プロジェクトのユーザーオプションファイル(.cobproj.user)に保存されます。

- フォルダの追加: [フォルダ]ボックスで指定されたフォルダを[参照パス(ユーザー)]一覧に追加します。
- 更新: [参照パス(ユーザー)]一覧で選択されているフォルダを[フォルダ]ボックスで指定されたフォルダに変更します。
- 上へ (): [参照パス(ユーザー)]一覧で選択されているフォルダを１つ上に移動します。
- 下へ (): [参照パス(ユーザー)]一覧で選択されているフォルダを１つ下に移動します。
- 削除 (): [参照パス(ユーザー)]一覧で選択されているフォルダを一覧から削除します。

参照パス(プロジェクト)

参照するアセンブリファイルのあるフォルダの一覧です。 各環境で参照するアセンブリのフォルダが同じ場合に、この一覧にフォルダを追加します。 この一覧は、プロジェクトファイル(.cobproj)に保存されます。

- フォルダの追加: [フォルダ]ボックスで指定されたフォルダを[参照パス(プロジェクト)]一覧に追加します。
- 更新: [参照パス(ユーザー)]一覧で選択されているフォルダを[フォルダ]ボックスで指定されたフォルダに変更します。
- 上へ (): [参照パス(プロジェクト)]一覧で選択されているフォルダを１つ上に移動します。
- 下へ (): [参照パス(プロジェクト)]一覧で選択されているフォルダを１つ下に移動します。
- 削除 (): [参照パス(プロジェクト)]一覧で選択されているフォルダを一覧から削除します。

NCOBU01058

[登録集パス] ページ (プロジェクト デザイナー)

[登録集パス] ページでは、COBOL コンパイラで使用される登録集ファイルのフォルダのパスを設定します。 詳細については、/copypath コンパイラオプションを参照してください。

フォルダ

[登録集パス(ユーザー)]一覧または[登録集パス(プロジェクト)]一覧に追加または更新を行うためのフォルダを指定します。

登録集パス(ユーザー)

登録集ファイルのフォルダの一覧です。 各環境で登録集ファイルのあるフォルダが異なっている場合に、この一覧にフォルダを追加します。 この一覧は、プロジェクトのユーザーオプションファイル(.cobproj.user)に保存されます。

- フォルダの追加: [フォルダ]ボックスで指定されたフォルダを[登録集パス(ユーザー)]一覧に追加します。
- 更新: [登録集パス(ユーザー)]一覧で選択されているフォルダを[フォルダ]ボックスで指定されたフォルダに変更します。
- 上へ (): [登録集パス(ユーザー)]一覧で選択されているフォルダを１つ上に移動します。
- 下へ (): [登録集パス(ユーザー)]一覧で選択されているフォルダを１つ下に移動します。
- 削除 (): [登録集パス(ユーザー)]一覧で選択されているフォルダを一覧から削除します。

登録集パス(プロジェクト)

登録集ファイルのフォルダの一覧です。 各環境で登録集ファイルのあるフォルダが同じ場合に、この一覧にフォルダを追加します。 この一覧は、プロジェクトファイル(.cobproj)に保存されます。

- フォルダの追加: [フォルダ]ボックスで指定されたフォルダを[登録集パス(プロジェクト)]一覧に追加します。
- 更新: [登録集パス(ユーザー)]一覧で選択されているフォルダを[フォルダ]ボックスで指定されたフォルダに変更します。
- 上へ (): [登録集パス(プロジェクト)]一覧で選択されているフォルダを１つ上に移動します。
- 下へ (): [登録集パス(プロジェクト)]一覧で選択されているフォルダを１つ下に移動します。
- 削除 (): [登録集パス(プロジェクト)]一覧で選択されているフォルダを一覧から削除します。

MSBuild タスク リファレンス (Fujitsu.COBOL.Build)

NetCOBOL for .NET が提供する MSBuild を使用するためのタスクです。 このタスクを使用するには、以下のファイルを Import 要素を使用しインポートする必要があります。 $(MSBuildExtensionsPath)は、MSBuild 用の拡張フォルダを示すプロパティで、通常は"C:¥Program Files¥MSBuild"です。

```
"$(MSBuildExtensionsPath)¥Fujitsu¥NetCOBOL for .NET¥V8.0¥Fujitsu.COBOL.Build.Tasks"
```

MSBuild で共通に使用できるタスクや要素などについては、 Visual Studio のドキュメントの MSBuild リファレンスを参照してください。

タスク	用途
Cobolc タスク	COBOL コンパイラ(cobolc.exe)を実行するタスクです。

NCOBU00879

Cobolc タスク

COBOL コンパイラ(cobolc.exe)を実行する MSBuild のタスクです。

パラメタ

パラメタ	説明
AdditionalLibPaths	省略可能な String[]型のパラメタです。 参照するアセンブリファイルへの検索パスを指定します。 詳細については、/libpath コンパイラオプションを参照してください。
AdditionalOptions	省略可能な String 型のパラメタです。 コンパイラに指定する追加のオプションを指定します。 このパラメタで指定されたオプションは、他で指定されたオプションのあとに指定されます。
AddModules	省略可能な String[]型のパラメタです。 アセンブリに含めるモジュールファイルを指定します。 詳細については、/addmodule コンパイラオプションを参照してください。
AssemblyCompany	省略可能な String 型のパラメタです。 アセンブリのカスタム属性に会社名を設定します。 詳細については、/company コンパイラオプションを参照してください。
AssemblyConfiguration	省略可能な String 型のパラメタです。 アセンブリのカスタム属性に構成を示す文字列を設定します。 詳細については、/configuration コンパイラオプションを参照してください。
AssemblyCopyright	省略可能な String 型のパラメタです。 アセンブリのカスタム属性に著作権を設定します。詳細については、/copyright コンパイラオプションを参照してください。
AssemblyDescription	省略可能な String 型のパラメタです。 アセンブリのカスタム属性に説明文を設定します。 詳細については、/description コンパイラオプションを参照してください。
AssemblyProduct	省略可能な String 型のパラメタです。 アセンブリのカスタム属性に製品名を設定します。 詳細については、/product コンパイラオプションを参照してください。
AssemblyTitle	省略可能な String 型のパラメタです。 アセンブリのカスタム属性にタイトルを設定します。 詳細については、/title コンパイラオプションを参照してください。
AssemblyTrademark	省略可能な String 型のパラメタです。 アセンブリのカスタム属性に商標を設定します。 詳細については、/trademark コンパイラオプションを参照してください。

AssemblyVersion	省略可能な String 型のパラメタです。 アセンブリのカスタム属性にバージョンを設定します。 詳細については、/version コンパイラオプションを参照してください。
CompilerDirectives	省略可能な String 型のパラメタです。 翻訳オプションを指定します。 詳細については、/wc コンパイラオプションを参照してください。
CompilerListingFile	省略可能な String 型のパラメタです。 翻訳リストファイルの出力を指定します。 詳細については、/print コンパイラオプションを参照してください。
CopyBookExtensions	省略可能な String[]型のパラメタです。 登録集ファイルの拡張子を指定します。 詳細については、/copyext コンパイラオプションを参照してください。
CopyBookNames	省略可能な String[]型のパラメタです。 登録集名、および位置を指定します。 詳細については、/copyname コンパイラオプションを参照してください。
CopyBookPaths	省略可能な String[]型のパラメタです。 登録集ファイルへの検索パスを指定します。 詳細については、/copypath コンパイラオプションを参照してください。
DebugType	省略可能な String 型のパラメタです。 デバッグ情報の種類を指定します。DebugType には、full、pdbonly、minimal、および none を指定します。 空文字列またはすべて空白が指定された場合は、none が指定されたとみなします。 このパラメタが省略され、かつ、EmitDebugInformation パラメタが true の場合は、full が指定されたとみなします。 このパラメタが指定されている場合、EmitDebugInformation パラメタは無視されます。 例えば、EmitDebugInformation パラメタが true であっても、DebugType に none が指定されている場合は、デバッグ情報は出力されません。 詳細については、/debug コンパイラオプションを参照してください。
DelaySign	省略可能な Boolean 型のパラメタです。 アセンブリへの署名の遅延を指定します。 詳細については、/delaysign コンパイラオプションを参照してください。
DocumentationFile	省略可能な String 型のパラメタです。 XML ドキュメントコメントを XML ファイルに出力します。 詳細については、/doc コンパイラオプションを参照してください。
EmitDebugInformation	省略可能な Boolean 型のパラメタです。 デバッグのための情報を出力します。 詳細については、/debug コンパイラオプションを参照してください。
FormDescriptExtensions	省略可能な String[]型のパラメタです。

	帳票定義体ファイルの拡張子を指定します。 詳細については、/formext コンパイラオプションを参照してください。
FormDescriptPaths	省略可能な String[]型のパラメタです。 帳票定義体ファイルへの検索パスを指定します。 詳細については、/formpath コンパイラオプションを参照してください。
KeyContainer	省略可能な String 型のパラメタです。 キーコンテナの名前を指定します。 詳細については、/keyname コンパイラオプションを参照してください。
KeyFile	省略可能な String 型のパラメタです。 キーを含んでいるファイル名を指定します。 詳細については、/keyfile コンパイラオプションを参照してください。
LinkResources	省略可能な ITaskItem[]型のパラメタです。 リンクするリソースファイルを指定します。 このパラメタのアイテムには、LogicalName および Access メタデータエントリを指定することができます。LogicalName は、/linkresource コンパイラオプションの name パラメータに対応しています。 Access は、/linkresource コンパイラオプションの access パラメータに対応しています。 詳細については、/linkresource コンパイラオプションを参照してください。
MainEntryPoint	省略可能な String 型のパラメタです。 アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定します。 詳細については、/main コンパイラオプションを参照してください。
NoLogo	省略可能な Boolean 型のパラメタです。 コンパイラの著作権情報を表示しないようにします。 詳細については、/nologo コンパイラオプションを参照してください。
NoWin32Manifest	省略可能な Boolean 型のパラメタです。 マニフェストを出力ファイルに埋め込まないように指示します。 詳細については、/nowin32manifest コンパイラオプションを参照してください。
OutputAssembly	省略可能な String 型のパラメタです。 出力ファイルのファイル名を指定します。 詳細については、/out コンパイラオプションを参照してください。
Platform	省略可能な String 型のパラメタです。 出力ファイルのターゲットとなるプラットフォームを指定します。 詳細については、/platform コンパイラオプションを参照してください。
Prefer32Bit	省略可能な Boolean 型のパラメタです。 Platfrom パラメタに anycpu が指定されている場合に、32 ビットプラットフォーム を優先するファイルを出力することを指示します。 詳細については、/platform コンパイラオプションを参照してください。
References	省略可能な ITaskItem[]型のパラメタです。

	参照するメタデータを含むアセンブリファイルを指定します。 詳細については、/reference コンパイラオプションを参照してください。
Resources	省略可能な ITaskItem[]型のパラメタです。 出力ファイルに埋め込むリソースファイルを指定します。 このパラメータのアイテムには、LogicalName および Access メタデータエントリを指定することができます。 LogicalName は、/resource コンパイラオプションの name パラメータに対応しています。Access は、/resource コンパイラオプションの access パラメータに対応しています。 詳細については、/resource コンパイラオプションを参照してください。
ResponseFiles	省略可能な ITaskItem[]型のパラメタです。 コンパイラのコマンド引数を応答ファイル(.rsp)で指定します。 詳細については、@コンパイラオプションを参照してください。
Sources	省略可能な ITaskItem[]型のパラメタです。 COBOL ソースファイルを指定します。
StackReserveSize	省略可能な Int32 型のパラメタです。 スタックに使用する仮想メモリのサイズを設定します。詳細については、/stack コンパイラオプションを参照してください。
TargetFrameworkProfile	省略可能な String 型のパラメタです。 対象とする.NET Framework のプロファイルを指定します。 省略した場合は空文字列が指定されたとみなします。 このバージョンの NetCOBOL for .NET では、 この値は空文字列でなければなりません。
TargetFrameworkVersion	省略可能な String 型のパラメタです。 対象とする.NET Framework のバージョンを指定します。 指定可能な値は 'v4.0', 'v4.5' , 'v4.5.1' , 'v4.5.2' , 'v4.6' , 'v4.6.1' , 'v4.6.2' , 'v4.7' , 'v4.7.1' で、 それぞれ.NET Framework のバージョンに対応しています。 対象とする.NET Framework のバージョンによって、使用する NetCOBOL for .NET のコンパイラ・ランタイムのバージョンが決まります。 詳細は、特定のバージョンの.NET Framework 向けアプリケーションを開発する際の注意事項を参照してください。 省略した場合は、このバージョン(V8.0)の NetCOBOL for .NET コンパイラ・ランタイムを使用します。
TargetType	省略可能な String 型のパラメタです。 出力ファイルの種類を指定します。 詳細については、/target コンパイラオプションを参照してください。
ToolPath	省略可能な String 型のパラメタです。 コンパイラ(cobolc.exe)へのパスを指定します。 このパラメタを省略した場合、このタスクに対応したバージョンのコンパイラを使用します。
TypeLibGUID	省略可能な String 型のパラメタです。 タイプライブラリの GUID を設定します。 詳細については、/typelibguid コンパイラオプションを参照してください。
Verifiable	省略可能な Boolean 型のパラメタです。 検証可能なタイプセーフコードを生成します。 詳細については、/verifiable コ

	ンパイラオプションを参照してください。
Win32Icon	省略可能な String 型のパラメタです。 出力ファイルにデフォルトアイコンを挿入します。 詳細については、/win32icon コンパイラオプションを参照してください。
Win32Manifest	省略可能な String 型のパラメタです。 出力ファイルに埋め込むマニフェストファイルを指定します。 詳細については、/win32manifest コンパイラオプションを参照してください。
Win32Resource	省略可能な String 型のパラメタです。 出力ファイルに Win32 リソースを挿入します。 詳細については、/win32res コンパイラオプションを参照してください。

説明

COBOL コンパイラ(cobolc.exe)を実行するタスクです。 Cobolc タスクが出力するエラーなどのメッセージは、MSBuild のタスクの出力形式に合わせて、 以下のような形式になります。

```
ファイル名(行番号): [warning|error] メッセージ番号: メッセージ文
```

COBOL のソースファイルに記述されている内容に対するメッセージでない場合、Cobolc タスクが記述されているファイルのファイル名および行番号となります。

メッセージ番号およびメッセージ文については、NetCOBOL for .NET メッセージ集 の cobolc コマンドメッセージおよび翻訳時メッセージを参照してください。

例

以下の例では、Compile アイテムで指定されている COBOL ソースファイルをコンパイルし、実行可能なアプリケーション(MyProg.exe)を作成します。

```
<Cobolc
    Sources="@(Compile)"
    TargetType="Exe"
    MainEntryPoint="MyClass,Main"
    OutputAssembly="MyProg.exe" />
```

コンパイラオプション・翻訳オプション

ここでは、コンパイラオプション、および翻訳オプションの、一覧と詳細を紹介します。

このトピックの内容

関連トピックス

NCOBU00556

コンパイラオプション - トピック順一覧

応答ファイルに関するもの	コンパイラオプション
コンパイラのコマンド引数を応答ファイル(.rsp)で指定	@
入口点に関するもの	コンパイラオプション
アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定	/main
登録集ファイルに関するもの	コンパイラオプション
登録集ファイルの拡張子を指定	/copyext
登録集名、および位置を指定	/copyname
登録集ファイルへのパスを指定	/copypath
帳票定義体に関するもの	コンパイラオプション
帳票定義体ファイルの拡張子を指定	/formext
帳票定義体ファイルへのパスを指定	/formpath
出力に関するもの	コンパイラオプション
XML ドキュメントファイルの出力を指定	/doc
出力ファイルのファイル名を指定	/out
出力ファイルのプラットフォームを指定	/platform
翻訳リストファイルの出力を指定	/print
出力ファイルに埋め込むリソースファイルを指定	/resource
出力ファイルの種類を指定	/target
出力ファイルにデフォルトアイコンを挿入	/win32icon
出力ファイルに Win32 リソースを挿入	/win32res
タイプライブラリの GUID を設定	/typelibguid
スタックに使用する仮想メモリのサイズを設定	/stack
Win32 マニフェストファイルの埋め込み	/win32manifest

Win32 マニフェストファイルの埋め込みの禁止	/nowin32manifest
アセンブリ・メタデータに関するもの	コンパイラオプション
アセンブリに含めるモジュールファイルの指定	/addmodule
参照するアセンブリファイルへのパスを指定	/libpath
参照するメタデータを含むアセンブリファイルを指定	/reference
リンクするリソースファイルを指定	/linkresource
アセンブリのカスタム属性に会社名を設定	/company
アセンブリのカスタム属性に構成を示す文字列を設定	/configuration
アセンブリのカスタム属性に著作権を設定	/copyright
アセンブリのカスタム属性に説明文を設定	/description
アセンブリのカスタム属性に製品名を設定	/product
アセンブリのカスタム属性にタイトルを設定	/title
アセンブリのカスタム属性に商標を設定	/trademark
アセンブリのカスタム属性にバージョンを設定	/version
公開キーに関するもの	コンパイラオプション
アセンブリへの署名の遅延を指定	/delaysign
キーファイルから公開キーの設定および署名	/keyfile
キーコンテナから公開キーの設定および署名	/keyname
その他	コンパイラオプション
デバッグのための情報を出力	/debug
コマンドラインでの使用方法を表示	/help,/?
コンパイラの著作権情報を表示しない	/nologo
オプションのセットを指定	/optionset
検証可能なタイプセーフコードを生成	/verifiable
翻訳オプションを指定	/wc

NCOBU00557

コンパイラオプション - 詳細

コンパイラオプションの詳細について説明します。

コンパイラオプション	用途
@	コンパイラのコマンド引数を応答ファイル(.rsp)で指定します。
/addmodule	アセンブリに含めるモジュールファイルを指定します。
/company	アセンブリのカスタム属性に会社名を設定します。
/configuration	アセンブリのカスタム属性に構成を示す文字列を設定します。
/copyext	登録集ファイルの拡張子を指定します。
/copyname	登録集名、および位置を指定します。
/copypath	登録集ファイルへのパスを指定します。
/copyright	アセンブリのカスタム属性に著作権を設定します。
/debug	デバッグのための情報を出力します。
/delaysign	アセンブリへの署名の遅延を指定します。
/description	アセンブリのカスタム属性に説明文を設定します。
/doc	XML ドキュメントコメントを XML ファイルに出力します。
/formext	帳票定義体ファイルの拡張子を指定します。
/formpath	帳票定義体ファイルへのパスを指定します。
/help,/?	コマンドラインでの使用方法を表示します。
/keyfile	キーファイルから公開キーの設定および署名を行います。
/keyname	キーコンテナから公開キーの設定および署名を行います。
/libpath	参照するアセンブリファイルへのパスを指定します。
/linkresource	リンクするリソースファイルを指定します。
/main	アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定します。
/nologo	コンパイラの著作権情報を表示しないようにします。

/nowin32manifest	出力ファイルにマニフェストファイルの埋め込みを禁止します。
/optionset	指定されたオプションのセットを指定された位置に展開します。
/out	出力ファイルのファイル名を指定します。
/platform	出力ファイルのターゲットとなるプラットフォームを指定します。
/print	翻訳リストファイルの出力を指定します。
/product	アセンブリのカスタム属性に製品名を設定します。
/reference	参照するメタデータを含むアセンブリファイルを指定します。
/resource	出力ファイルに埋め込むリソースファイルを指定します。
/stack	スタックに使用する仮想メモリのサイズを設定します。
/target	出力ファイルの種類を指定します。
/title	アセンブリのカスタム属性にタイトルを設定します。
/trademark	アセンブリのカスタム属性に商標を設定します。
/typelibguid	タイプライブラリの GUID を設定します。
/verifiable	検証可能なタイプセーフコードを生成します。
/version	アセンブリのカスタム属性にバージョンを設定します。
/wc	翻訳オプションを指定します。
/win32icon	出力ファイルにデフォルトアイコンを挿入します。
/win32manifest	出力ファイルにマニフェストファイルを埋め込みます。
/win32res	出力ファイルに Win32 リソースを挿入します。

NCOBU00558

@ (コンパイラのコマンド引数を応答ファイル(.rsp)で指定)

コンパイラのコマンド引数を応答ファイル(.rsp)で指定します。

形式

```
@<filename>
```

filename

応答ファイルの名前を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。

説明

コンパイラのコマンド引数を応答ファイル(.rsp)で指定します。 応答ファイルは、コンパイラオプションやソースファイル名がリストされているテキストファイルです。 応答ファイルには、一行に複数のコンパイラオプションを記述できます。 応答ファイルでは、空白やタブを含むパス名など、コマンドラインと同様に二重引用符で囲む必要があります。 行の先頭文字が#の場合、その行はコメントとして扱われます。 例えば、source1.cob から MyProg.exe という名前のアプリケーションを作成する、応答ファイルの記述は、以下のようになります。

```
# build the MyProg file
/target:exe
/main:Main
/out:MyProg.exe
source1.cob
```

@コンパイラオプションは、指定された位置に応答ファイル内で指定されたコンパイラオプションおよびソースファイル名を展開します。 例えば、上記の応答ファイル(MyProg.rsp)を次のように指定した場合、

```
cobolc @MyProg.rsp /debug
```

そのコマンドラインは、次のコマンドラインが指定されたことになります。

```
cobolc /target:exe /main:Main /out:MyProg.exe source1.cob /debug
```

@コンパイラオプションは応答ファイル内で指定することはできません。 複数の応答ファイルを指定する場合は、コマンドラインで@コンパイラオプションを複数指定します。

```
@option1.rsp @option2.rsp
```

応答ファイルの文字コードは ACP(Shift JIS)でなければなりません。

例

以下のコマンドラインでは、option.rsp を使用しコンパイルを行います。

```
cobolc @option.rsp
```

NCOBU00559

/addmodule (アセンブリに含めるモジュールファイルの指定)

アセンブリに含めるモジュールファイルを指定します。

形式

```
/addmodule:<filename>[,<filename>]...
```

filename[,filename]...

モジュールファイルのファイル名を指定します。 空白を含むファイル名を指定する場合、そのファイル名を二重引用符(")で囲む必要があります。 複数のファイルを指定する場合、カンマ(,)で区切り指定します。

説明

指定されたモジュールファイルのメタデータ（型情報）を出力ファイルのアセンブリに含めます。 アプリケーションのフォルダ以外にあるモジュールファイルを指定することもできますが、実行時には指定したモジュールファイルが、アプリケーションのフォルダに必要になります。 モジュールファイルは、/target:module コンパイラオプションによって作成することができます。 モジュールファイルはアセンブリの情報を持ちません。 /target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

このコンパイラオプションは、複数指定することができます。 指定されたすべてのモジュールファイルがアセンブリに含められます。

例

以下のコマンドラインでは、source.cob をコンパイルし、meta1.netmodule と meta2.netmodule をアセンブリに含む、myprog.exe を作成します。

```
cobolc /addmodule:meta1.netmodule,meta2.netmodule /main:Main /out:myprog.exe source.cob
```

/company (アセンブリのカスタム属性に会社名を設定)

アセンブリのカスタム属性に会社名を設定します。

形式

```
/company: <text>
```

text

会社名を示す文字列を指定します。 空白を含む会社名を指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性に会社名を示す文字列を設定します。 また、/win32res コンパイラオプションを指定しない場合、指定した会社名は、バージョン情報リソースの会社名項目として Windows エクスプローラーなどで表示されます。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、会社名を「Fujitsu Limited」にした myprog.exe を出力します。

```
cobolc /company:"Fujitsu Limited" /main:Main /out:myprog.exe source.cob
```

NCOBU00561

/configuration (アセンブリのカスタム属性に構成を示す文字列を設定)

アセンブリのカスタム属性に構成を示す文字列を設定します。

形式

```
/configuration: <text>
```

text

Retail や Debug などのアセンブリの構成を示す文字列を指定します。 空白を含む構成文字列を指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性に構成を示す文字列を設定します。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、アセンブリの構成を「Retail」にした myprog.exe を出力します。

```
cobolc /configuration:"Retail" /main:Main /out:myprog.exe source.cob
```

/copyext (登録集ファイルの拡張子を指定)

登録集ファイルの拡張子を指定します。

形式

```
/copyext:<extname>[,<extname>]...
```

extname[,extname]...

登録集ファイルの拡張子を指定します。

説明

登録集ファイルの拡張子を指定します。拡張子には、任意の文字列を指定することができます。ファイル名に拡張子がない場合は、拡張子の代わりに、文字列"None"を指定します。

拡張子を複数指定した場合、指定された順序で検索が行われます。

コンパイラオプション/copyext の指定がない場合、 (1)拡張子(cbl)、(2)拡張子(cob)、(3)拡張子(cobol) の順で検索が行われます。

登録集ファイルの拡張子に"None"を使用することはできません。

例

以下のコマンドラインでは、登録集ファイルとして拡張子が abc のファイルが取り込まれ、myprog.exe を作成します。

```
cobolc /copyext:abc /main:Main /out:myprog.exe source.cob
```

NCOBU00563

/copyname (登録集名および位置の指定)

登録集名、および位置を指定します。

形式

```
/copyname:<name>=<pathname>[,<name>= <pathname>]...
```

name

登録集名を指定します。

pathname

登録集ファイルの格納フォルダを指定します。 空白を含むパスを指定する場合、そのパス名を二重引用符(")で囲む必要があります。

説明

登録集名と登録集ファイルの格納フォルダを関係付けます。 登録集名を指定した COPY 文を使った COBOL ソースプログラムを翻訳する場合、登録集名と登録集ファイルのフォルダを関係付けます。 カンマ(,)で区切ることによって、複数の関係付けを指定できます。

このコンパイラオプションは、複数指定することができます。指定したすべての関連付けが有効になります。

例

以下のコマンドラインでは、COPY 文で登録集名に data1 を指定することにより、c:¥copy フォルダにある登録集ファイルが取り込まれた myprog.exe を作成します。

source.cob

```
     COPY cp1 OF data1.
```

cp1.cbl が c:¥copy にある場合、以下のように指定します。

```
cobolc /copyname:data1=c:¥copy /main:Main /out:myprog.exe source.cob
```

/copypath (登録集ファイルへのパスの指定)

登録集ファイルへのパスを指定します。

形式

```
/copypath:<pathname>[,<pathname>]...
```

pathname[,pathname]...

登録集ファイルへのパスを指定します。 空白を含むパスを指定する場合、そのパス名を二重引用符(")で囲む必要があります。 複数のパスを指定する場合、カンマ(,)で区切り指定します。

説明

COPY 文で記述されている登録集ファイルへのパスを指定します。 COPY 文で参照するファイルは、指定されたパス名を使用して、指定順に検索されます。

このコンパイラオプションは、複数指定することができます。 指定されたすべてのパスが登録集ファイルへのパスとして参照されます。

例

以下のコマンドラインでは、c:¥copy にある登録集ファイルが source.cob に記述されている COPY 文により取り込まれ、myprog.exe を作成します。

```
cobolc /copypath:c:¥copy /main:Main /out:myprog.exe source.cob
```

NCOBU00565

/copyright (アセンブリのカスタム属性に著作権を設定)

アセンブリのカスタム属性に著作権を設定します。

形式

```
/copyright: <text>
```

text

著作権を示す文字列を指定します。 空白を含む著作権文字列を指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性に著作権を示す文字列を設定します。 また、/win32res コンパイラオプションを指定しない場合、指定した著作権文字列は、バージョン情報リソースの著作権項目として Windows エクスプローラーなどで表示されます。モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、著作権文字列を「Copyright (C) Fujitsu Limited」にした myprog.exe を出力します。

```
cobolc /copyright:"Copyright (C) Fujitsu Limited" /main:Main /out:myprog.exe source.cob
```

/debug (デバッグのための情報を出力)

デバッグのための情報を出力します。

形式

```
/debug[:full] /debug:pdbonly /debug:minimal
```

full

省略可能です。実行中のプログラムにデバッガをアタッチできます。

pdbonly

デバッガから実行を開始した場合、ソースコードをデバッグできます。 実行中のプログラムにデバッガをアタッチした場合、アセンブリコードだけ表示されます。

minimal

デバッガから実行を開始した場合でも、ソースコードをデバッグすることはできません。 例外が発生した場合、スタックトレースにメソッド内の行番号を表示させることができます。

説明

デバッグに必要な情報を出力します。 このオプションが指定されなかった場合、出力ファイルに対してデバッグを行うことができません。

/debug:full では、デバッグ情報が利用できることを JIT コンパイラに指示するため、 パフォーマンスに影響することがあります。

リリースバージョンのコンパイルでは、/debug:pdbonly または/debug:minimal を指定するか、 /debug オプションを指定しないことを推奨します。

例

以下のコマンドラインでは、myprog.exe のデバッグ情報を出力します。

```
cobolc /debug:full /main:Main /out:debug¥myprog.exe source.cob
```

```
cobolc /debug:pdbonly /main:Main /out:release¥myprog.exe source.cob
```

NCOBU00567

/delaysign (アセンブリへの署名の遅延を指定)

アセンブリへの署名の遅延を指定します。

形式

```
/delaysign
```

説明

アセンブリへの署名を遅延し公開キーだけをアセンブリに設定する場合、このコンパイラオプションを指定します。/keyfile コンパイラオプションまたは/keyname コンパイラオプションを指定しない場合、公開キーの設定は行われません。 また、モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、公開キーだけが設定された myassembly.dll を出力します。

```
cobolc /delaysign /keyfile:mykey.snk /target:library /out:myassembly.dll source.cob
```

/description (アセンブリのカスタム属性に説明文を設定)

アセンブリのカスタム属性に説明文を設定します。

形式

```
/description: <text>
```

text

説明文を指定します。 空白を含むタイトルを指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性に説明文を設定します。 また、/win32res コンパイラオプションを指定しない場合、指定した説明は、バージョン情報リソースのコメント項目として Windows エクスプローラーなどで表示されます。モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、アセンブリの説明を「This program is the example.」にした myprog.exe を出力します。

```
cobolc /description:"This program is an example." /main:Main /out:myprog.exe source.cob
```

NCOBU01197

/doc (XML ドキュメントファイルの出力を指定)

XML ドキュメントコメントを XML ファイルに出力します。

形式

```
/doc[: <filename>]
```

filename

XML ドキュメントファイルの名前を指定します。 空白を含むファイル名を指定する場合は二重引用符(")で囲む必要があります。 省略した場合は、出力ファイルと同じフォルダに、出力ファイル名の拡張子を「xml」に変更したファイルに出力されます。

説明

XML ドキュメントファイルを指定します。 ソースコードから XML ドキュメントコメントを検索して、XML ドキュメントファイルを生成します。

例

以下の例では、c:¥sample¥bin¥mylibrary.xml に XML ドキュメントファイルを作成します。

```
cobolc /doc:c:¥sample¥bin¥mylibrary.xml /target:library /out:c:¥sample¥bin¥mylibrary.dll
myclass.cob
```

/formext (帳票定義体ファイルの拡張子を指定)

帳票定義体ファイルの拡張子を指定します。

形式

```
/formext:<extname>[,<extname>]...
```

extname[,extname]...

帳票定義体ファイルの拡張子を指定します。

説明

帳票定義体ファイルの拡張子を指定します。拡張子には、任意の文字列を指定することができます。ファイル名に拡張子がない場合は、拡張子の代わりに、文字列"None"を指定します。

拡張子を複数指定した場合、指定された順序で検索が行われます。

備考

コンパイラオプション/formext の指定がない場合、 (1)拡張子(pmd)、(2)拡張子(pxd)、(3)拡張子(smd) の順で検索が行われます。 拡張子 PXD は、「帳票クラスライブラリ」で使用される専用定義体の拡張子です。

- 帳票定義体ファイルの拡張子として"None"を使用することはできません。

例

以下のコマンドラインでは、帳票定義体ファイルとして拡張子が abc のファイルが取り込まれ、myprog.exe を作成します。

```
cobolc /formext:abc /main:Main /out:myprog.exe source.cob
```

NCOBU00572

/formpath (帳票定義体ファイルへのパスを指定)

帳票定義体ファイルへのパスを指定します。

形式

```
/formpath:<pathname>[,<pathname>]...
```

pathname[,pathname]...

帳票定義体ファイルのパスを指定します。 空白を含むパスを指定する場合、そのパス名を二重引用符(")で囲む必要があります。 複数のパスを指定する場合、カンマ(,)で区切り指定します。

説明

IN/OF XMDLIB 指定の COPY 文により帳票定義体からレコード定義を取り込む場合、帳票定義体ファイルのフォルダを指定します。

このコンパイラオプションは、複数指定することができます。 指定されたすべてのパスが帳票定義体ファイルへのパスとして参照されます。

例

以下のコマンドラインでは、`c:¥form` にある帳票定義体ファイルが取り込まれ、myprog.exe を作成します。

```
cobolc /formpath:c:¥form /main:Main /out:myprog.exe source.cob
```

/help,/? (コマンドラインでの使用方法を表示)

コマンドラインでの使用方法を表示します。

形式

```
/help


/?
```

説明

コンパイラのコマンドラインでの指定方法およびコンパイラオプションを表示します。 このオプションが指定された場合、他の指定は無視されます。

例

以下のコマンドラインでは、コマンドラインの使用方法が表示されます。

```
cobolc /help
```

NCOBU00574

/keyfile (キーファイルから公開キーの設定および署名)

キーファイルから公開キーの設定および署名を行います。

形式

```
/keyfile: <filename>
```

filename

キーファイルの名前を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。

説明

公開、および秘密キーの両方を含んだキーファイルは、Microsoft Windows SDK に付属の sn.exe ユーティリティを使用して生成されます。 キーまたはキーのペアをもつキーファイルから公開キーの設定および署名を行います。 指定されたキーファイルの公開キーを使用しアセンブリに公開キーが設定されます。 また、秘密キーを使用してアセンブリに署名を行います。 署名を遅延させる場合には、/delaysign コンパイラオプションを指定し、公開キーをもつキーファイルを指定する必要があります。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、mykey.snk にあるキーのペアにより署名を行った myassembly.dll を作成します。

```
cobolc /keyfile:mykey.snk /target:library /out:myassembly.dll source.cob
```

/keyname (キーコンテナから公開キーの設定および署名)

キーコンテナから公開キーの設定および署名を行います。

形式

```
/keyname: <name>
```

name

暗号化サービスプロバイダ(CSP)が提供するコンテナの名前を指定します。 空白を含む名前を指定する場合、二重引用符(")で囲む必要があります。

説明

キーまたはキーのペアをもつキーのコンテナから公開キーの設定および署名を行います。 指定されたキーのコンテナがもつ公開キーを使用しアセンブリに公開キーが設定されます。 また、秘密キーを使用してアセンブリに署名を行います。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、mycontainer にあるキーのペアにより署名を行った myassembly.dll を作成します。

```
cobolc /keyname:mycontainer /target:library /out:myassembly.dll source.cob
```

NCOBU00576

/libpath (参照するアセンブリファイルへのパスを指定)

参照するアセンブリファイルへのパスを指定します。

形式

```
/libpath:<pathname>[,<pathname>]...
```

pathname[,pathname]...

アセンブリファイルへのパスを指定します。 空白を含むパスを指定する場合、そのパス名を二重引用符(")で囲む必要があります。 複数のパスを指定する場合、カンマ(,)で区切り指定します。

説明

/reference コンパイラオプションで指定されるアセンブリへのパスを指定します。

このコンパイラオプションは、複数指定することができます。指定されたすべてのパスがアセンブリへのパスとして参照されます。

例

以下のコマンドラインでは、c:¥lib にある meta1.dll のアセンブリを参照し、source.cob から myprog.exe を作成します。

```
cobolc /libpath:c:¥lib /reference:meta1.dll /main:Main /out:myprog.exe source.cob
```

/linkresource (リンクするリソースファイルを指定)

リンクするリソースファイルを指定します。

形式

```
/linkresource:<filename>[,<name>[,<access>]]
```

filename

リンクするリソースファイルのファイル名を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。

name

リソースのロードに使用される名前を指定します。 指定されなかった場合、デフォルトでファイル名と同じ名前になります。 空白を含む名前を指定する場合、二重引用符(")で囲む必要があります。 また、次の文字を含む名前を指定することはできません。

¥ / : , ; * ? " < > |

access

リソースへのアクセス属性 public または private を指定します。 デフォルトは public です。

説明

リンクするリソースファイルの名前を指定します。 アプリケーションのフォルダ以外にあるリソースファイルを指定することもできますが、実行時には指定したリソースファイルが、アプリケーションのフォルダに必要になります。

このコンパイラオプションは、複数指定することができます。指定されたすべてのリソースファイルがリンクされます。

例

以下のコマンドラインでは、myres.resource ファイルをリンクした myprog.exe を作成します。実行時には、myres.resource ファイルにあるリソースが使用されます。

```
cobolc /linkresource:myres.resource /main:Main /out:myprog.exe source.cob
```

NCOBU00578

/main (アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定)

アプリケーションのエントリポイントとなるプログラム名またはメソッド名を指定します。

形式

```
/main: <methodname>
```

methodname

エントリポイントとなるプログラム名またはメソッド名を指定します。 メソッドの場合には、クラス名とメソッド名をカンマ(,)で区切り指定します。

説明

出力ファイルがアプリケーション(.exe)の場合、最初に呼び出されるエントリポイントを指定します。 エントリポイントとして指定できるプログラムまたはメソッドは、以下の条件を満たしている必要があります。

- 引数なし。
- 戻り値なし。
- メソッドの場合、STATIC である。

例

以下のコマンドラインでは、myprog.exe のエントリポイントを MyClass,Main メソッドにします。

```
cobolc /main:MyClass,Main /out:myprog.exe source.cob
```

/nologo (コンパイラの著作権情報の表示を抑止)

コンパイラの著作権情報を表示しないようにします。

形式

```
/nologo
```

説明

コンパイル時にコンパイラの著作権情報が表示されなくなります。

例

以下のコマンドラインでは、コンパイラの著作権が表示されません。

```
cobolc /nologo source.cob
```

NCOBU01051

/nowin32manifest (Win32 マニフェストファイルの埋め込み禁止)

出力ファイルにアプリケーションマニフェストを埋め込まないように指示します。

形式

```
/nowin32manifest
```

説明

出力ファイルにアプリケーションマニフェストの埋め込みを禁止します。 マニフェストファイルをアプリケーションの実行ファイルと同じフォルダに 配置する場合や仮想化に従う場合に、このオプションを 指定します。仮想化については、Visual Studio のドキュメントの ClickOnce 配置を参照してください。

アプリケーションマニフェストの埋め込みについては、/win32manifest コンパイラオプションを参照してください。

例

以下のコマンドラインでは、アプリケーションマニフェストが埋め込まれていない myprog.exe を出力します。

```
cobolc /nowin32manifest /main:Main /out:myprog.exe source.cob
```

/optionset (特定のオプションのセットを指定)

指定されたオプションのセットを指定された位置に展開します。

形式

```
/optionset:sqlclr
```

sqlclr

SQL Server の内部で動作するアセンブリ(SQL CLR データベースオブジェクト)を作成するためのオプションセットです。 次のオプションが展開されます。

- /verifiable
- /wc:RCS(SJIS-UCS2)
- /wc:INITONLY
- /wc:SQLSCOPE(METHOD)

説明

指定されたオプションのセットを指定された位置に展開します。 例えば、次のコマンドラインを指定した場合、

```
cobolc /target:library /optionset:sqlclr class1.cob
```

そのコマンドラインは、次のコマンドラインが指定されたことになります。

```
cobolc /target:library /verifiable /wc:RCS(SJIS-UCS2),INITONLY,SQLSCOPE(METHOD) class1.cob
```

展開される各オプションについては、各オプションの説明を参照してください。

例

以下のコマンドラインでは、SQL CLR データベースオブジェクト用のライブラリ(mylib.dll)を作成します。

```
cobolc /optionset:sqlclr /target:library /out:mylib.dll source.cob
```

NCOBU00580

/out (出力ファイルのファイル名を指定)

出力ファイルのファイル名を指定します。

形式

```
/out: <filename>
```

filename

出力ファイルのファイル名を指定します。 空白を含むファイル名を指定する場合は二重引用符(")で囲む必要があります。

説明

出力ファイルのファイル名を指定します。 このコンパイラオプションが指定されなかった場合、出力ファイルの名前は、先頭に指定されているソースファイルの名前から拡張子を変えた名前となります。

例

以下のコマンドラインでは、source.cob から myprog.exe を作成します。

```
cobolc /main:Main /out:myprog.exe source.cob
```

/platform (プラットフォームの指定)

出力ファイルのターゲットとなるプラットフォームを指定します。

形式

```
/platform:anycpu

/platform:anycpu32bitpreferred

/platform:x86

/platform:x64
```

anycpu

どのプラットフォームでも動作するアセンブリを作成します。(デフォルト) アプリケーションが 64 ビットプロセスとして実行される場合は、64 ビットモードで動作します。

anycpu32bitpreferred

どのプラットフォームでも動作するアセンブリを作成します。 64 ビットと 32 ビットの両方のシステムをサポ ートするシステムでは、32 ビットモードで動作します。 anycpu32bitpreferred は、.NET Framework 4.5 以降で動作する実行可能ファイル(.EXE)のみ有効です。

x86

x86 互換の 32 ビット環境で動作するアセンブリを作成します。 64 ビット環境では、WOW64 で動作します。

x64

AMD64 または Intel64 命令をサポートするコンピュータの 64 ビット環境で動作するアセンブリを作成します。

説明

出力ファイルのターゲットとなるプラットフォームを指定します。 このオプションを指定しない場合、/platform:anycpu が指定されたとみなします。

例

以下のコマンドラインでは、32 ビット環境で動作するアプリケーション(myprog.exe)を作成します。

```
cobolc /platform:x86 /main:MyClass,Main /out:myprog.exe source.cob
```

NCOBU00581

/print (翻訳リストファイルの出力を指定)

翻訳リストファイルの出力を指定します。

形式

```
/print[:<filename>]
```

filename

出力する翻訳リストのファイル名を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。 この指定を省略した場合、出力ファイル名から拡張子を「lst」に変更したファイルに出力されます。

説明

翻訳リストファイルの出力および出力ファイル名を指定します。 翻訳リストに出力される項目は、翻訳オプション(/wc コンパイラオプション)により指定します。

例

以下のコマンドラインでは、`c:¥print.lst` に翻訳リストを出力します。

```
cobolc /print:c:¥print.lst /wc:MESSAGE /main:Main /out:myprog.exe source.cob
```

/product (アセンブリのカスタム属性に製品名を設定)

アセンブリのカスタム属性に製品名を設定します。

形式

```
/product: <text>
```

text

製品名を示す文字列を指定します。 空白を含む製品名を指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性に製品名を示す文字列を設定します。 また、/win32res コンパイラオプションを指定しない場合、指定した製品名は、バージョン情報リソースの製品名項目として Windows エクスプローラーなどで表示されます。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、製品名を「Fujitsu Examples」にした myprog.exe を出力します。

```
cobolc /product:"Fujitsu Examples" /main:Main /out:myprog.exe source.cob
```

NCOBU00583

/reference (参照するメタデータを含むアセンブリファイルを指定)

参照するメタデータを含むアセンブリファイルを指定します。

形式

```
/reference:<filename>[,<filename>]...
```

filename[,filename]...

アセンブリファイル名を指定します。 空白を含むファイル名を指定する場合、そのファイル名を二重引用符(")で囲む必要があります。 複数のファイルを指定する場合、カンマ(,)で区切り指定します。

説明

手続き内で使用しているクラスなどの情報を含むアセンブリファイルを指定します。 アセンブリファイルがパス無しのファイル名で指定された場合、以下のフォルダを順番に検索して、最初に見つかった同名のアセンブリが使用されます。

- カレントフォルダ
- /libpath コンパイラオプションで指定されたフォルダ
- .NET Framework のフォルダ

このコンパイラオプションは、複数指定することができます。指定されたすべてのアセンブリファイルが参照の対象となります。

例

以下のコマンドラインでは、meta1.dll と meta2.dll のアセンブリを参照し、source.cob から myprog.exe を作成します。

```
cobolc /reference:meta1.dll,meta2.dll /main:Main /out:myprog.exe source.cob
```

/resource (出力ファイルに埋め込むリソースファイルを指定)

出力ファイルに埋め込むリソースファイルを指定します。

形式

```
/resource:<filename>[,<name>[,<access>]]
```

filename

リソースファイルのファイル名を指定します。空白を含むファイル名を指定する場合は、二重引用符(")で囲む必要があります。

name

リソースのロードに使用される名前を指定します。 指定されなかった場合、デフォルトでファイル名と同じ名前になります。 空白を含む名前を指定する場合、二重引用符(")で囲む必要があります。 また、次の文字を含む名前を指定することはできません。

```
¥ / : , ; * ? " < > |
```

access

リソースへのアクセス属性 public または private を指定します。 デフォルトは public です。

説明

出力ファイルに埋め込むリソースファイルの名前を指定します。

このコンパイラオプションは、複数指定することができます。指定されたすべてのリソースが埋め込まれます。

例

以下のコマンドラインでは、myres.resource ファイルが埋め込まれた myprog.exe を作成します。実行時には、myprog.exe に埋め込まれているリソースが使用されます。

```
cobolc /resource:myres.resource /main:Main /out:myprog.exe source.cob
```

NCOBU00882

/stack (スタックサイズの設定)

スタックに使用する仮想メモリのサイズを設定します。

形式

```
/stack:<size>
```

size

スタックサイズをバイト単位の数値で指定します。

説明

出力ファイルがアプリケーション(.exe)の場合に、スタックに使用する仮想メモリのサイズを設定します。 このオプションを指定しない場合、/platform コンパイラオプションで指定されたプラットフォームによりスタックサイズは以下の値となります。

プラットフォーム	スタックサイズ
anycpu	1047586 (1MB)
x86	1047586 (1MB)
x64	4194304 (4MB)

例

以下のコマンドラインでは、myprog.exe が使用するスタックサイズを 2MB に設定します。

```
cobolc /main:MyClass,Main /stack:2097152 /out:myprog.exe source.cob
```

/target (出力ファイルの種類を指定)

出力ファイルの種類を指定します。

形式

```
/target:exe

/target:library

/target:module

/target:winexe
```

exe

コンソールアプリケーション(.exe)を作成します。（デフォルト）

library

ライブラリ(.dll)を作成します。

module

モジュール(.netmodule)を作成します。

winexe

Windows アプリケーションを(.exe)を作成します。

説明

出力ファイルの種類を指定します。このコンパイラオプションを指定しない場合、デフォルトで/target:exe が指定されたとみなします。

例

以下のコマンドラインでは、source.cob からライブラリ mylib.dll を作成します。

```
cobolc /target:library /out:mylib.dll source.cob
```

NCOBU00586

/title (アセンブリのカスタム属性にタイトルを設定)

アセンブリのカスタム属性にタイトルを設定します。

形式

```
/title: <text>
```

text

タイトル文字列を指定します。 空白を含むタイトルを指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性にタイトルを示す文字列を設定します。 また、/win32res (出力ファイルに Win32 リソースを挿入)コンパイラオプションを指定しない場合、指定したタイトルは、バージョン情報リソースの説明項目として Windows エクスプローラーなどで表示されます。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、アセンブリのタイトルを「My program」にした myprog.exe を出力します。

```
cobolc /title:"My program" /main:Main /out:myprog.exe source.cob
```

/trademark (アセンブリのカスタム属性に商標を設定)

アセンブリのカスタム属性に商標を設定します。

形式

```
/trademark: <text>
```

text

商標を示す文字列を指定します。 空白を含む商標文字列を指定する場合、二重引用符(")で囲む必要があります。

説明

アセンブリのカスタム属性に商標を示す文字列を設定します。 また、/win32res コンパイラオプションを指定しない場合、指定した商標文字列は、バージョン情報リソースの商標項目として Windows エクスプローラーなどで表示されます。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

NCOBU00588

/typelibguid (タイプライブラリの GUID を設定)

タイプライブラリの GUID を設定します。

形式

```
/typelibguid:<guid>
```

guid

タイプライブラリの GUID を指定します。

説明

タイプライブラリの GUID を設定します。この GUID は、プロジェクトが COM に公開された場合、タイプ ライブラリの ID になります。

例

以下のコマンドラインでは、タイプライブラリの GUID を{6BAA0193-E411-40E2-9502-5500CA53D316}に設定したライブラリを出力します。

```
cobolc /typelibguid:6BAA0193-E411-40E2-9502-5500CA53D316 /target:library /out:mylib.dll source.cob
```

/verifiable (検証可能なタイプセーフコードを生成)

検証可能なタイプセーフコードを生成します。

形式

```
/verifiable
```

説明

検証可能なタイプセーフコードを生成する場合、/verifiable を指定します。

このオプションを指定しない場合は性能重視のコードを生成します。デフォルトは指定なしとみなします。

検証可能なタイプセーフコードについては、.NET Framework のドキュメントの検証可能なタイプ セーフ コードの作成を参照してください。

例

以下のコマンドラインでは、検証可能なタイプセーフコードからライブラリ mylib.dll を作成します。

```
cobolc /verifiable /target:library /out:mylib.dll source.cob
```

以下の機能を使用すると、/verifiable は無効になります。

- LINAGE 句
- コンストラクタメソッドにおける親クラスのコンストラクタの呼出し
- プラットフォーム呼び出し(PINVOKE)
- 要素も配列である多次元配列（ジャグ配列）

NCOBU00590

/version (アセンブリのカスタム属性にバージョンを設定)

アセンブリのカスタム属性にバージョンを設定します。

形式

```
/version: <version>
```

version

バージョンを指定します。 バージョンは、major.minor.build.revision の形式で指定します。 major、minor、build、revision には、それぞれ 0 から 65534 までの値を指定することができます。

説明

アセンブリのカスタム属性にバージョンを設定します。 このコンパイラオプションを指定しなかった場合、バージョンは「0.0.0.0」となります。 また、/win32res コンパイラオプションを指定しない場合、指定したバージョンは、バージョン情報リソースのファイルバージョンおよび製品バージョン項目として Windows エクスプローラーなどで表示されます。 モジュールファイルはアセンブリに関する情報を持ちません。/target:module コンパイラオプションが指定されている場合、このコンパイラオプションの指定は無効となります。

例

以下のコマンドラインでは、バージョンを「1.0.0.0」にした myprog.exe を出力します。

```
cobolc /version:1.0.0.0 /main:Main /out:myprog.exe source.cob
```

/wc (翻訳オプションを指定)

翻訳オプションを指定します。

形式

```
/wc:<directive>[,<directive>]...
```

directive[,directive]...

翻訳オプションを指定します。 複数の翻訳オプションを指定する場合、カンマ(,)で区切り指定します。

説明

翻訳オプションを指定します。詳細は、翻訳オプション - アルファベット順一覧を参照してください。

このコンパイラオプションは、複数指定することができます。指定されたすべての翻訳オプションが有効となります。 同じ種類の翻訳オプションが複数指定されている場合には、最後に指定された翻訳オプションが有効となります。

NetCOBOL for .NET V2.1 までは、/wc コンパイラオプションを複数指定した場合には、最後に指定した /wc コンパイラオプションのみ有効でした。

例

以下のコマンドラインでは、通貨編集用文字として$を指定するための翻訳オプション CURRENCY と CHECK 機能を使用するための翻訳オプション CHECK を指定しています。

```
cobolc /wc:"CURRENCY($),CHECK(ALL)" /main:Main /out:myprog.exe source.cob
```

NCOBU00592

/win32icon (出力ファイルにデフォルトアイコンを挿入)

出力ファイルにデフォルトアイコンを挿入します。

形式

```
/win32icon: <filename>
```

filename

アイコンファイル(.ico)のファイル名を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。

説明

出力ファイルにデフォルトアイコンとして、アイコンを挿入します。 挿入されたアイコンは、Windows エクスプローラーなどで使用されます。/target:module コンパイラオプションまたは/win32res コンパイラオプションが指定されている場合、このコンパイラオプションは無効となります。

例

以下のコマンドラインでは、アイコン app.ico が挿入された myprog.exe を作成します。

```
cobolc /win32icon:app.ico /main:Main /out:myprog.exe source.cob
```

/win32manifest (Win32 マニフェストファイルの埋め込み)

出力ファイルにマニフェストファイルを埋め込みます。

形式

```
/win32manifest: <filename>
```

filename

マニフェストファイル(.manifest)のファイル名を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。

説明

出力ファイルがアプリケーション(.exe)の場合、既定では、 "asInvoker"要求実行レベルを指定するアプリケーションマニフェストが埋め込まれます。 独自のマニフェストファイルを作成し、出力ファイルに埋め込む場合に、このオプションを指定します。/nowin32manifest コンパイラオプション、/target:module コンパイラオプションまたは/win32res コンパイラオプションが指定されている場合、このコンパイラオプションは無効になります。

例

以下のコマンドラインでは、myapp.manifest が埋め込まれた myprog.exe を出力します。

```
cobolc /win32manifest:myapp.manifest /main:Main /out:myprog.exe source.cob
```

NCOBU00593

/win32res (出力ファイルに Win32 リソースを挿入)

出力ファイルに Win32 リソースを挿入します。

形式

```
/win32res: <filename>
```

filename

Win32 リソースファイル(.res)のファイル名を指定します。 空白を含むファイル名を指定する場合、二重引用符(")で囲む必要があります。

説明

出力ファイルに Win32 リソースを挿入します。Win32 リソースは、Windows エクスプローラーなどで表示されるアイコンやバージョン情報を含めることができます。このコンパイラオプションが指定されなかった場合は、アセンブリ情報からバージョン情報を生成し出力ファイルに挿入します。

例

以下のコマンドラインでは、myres.res が挿入された myprog.exe を作成します。

```
cobolc /win32res:myres.res /main:Main /out:myprog.exe source.cob
```

翻訳オプション - アルファベット順一覧

指示	説明
ALPHAL	ソースプログラム中の英小文字を英大文字と等価に扱うか、扱わないかを指定します。
ALPHAL(ALL)	COBOL の語は、英小文字と英大文字が等価に扱われます。また、プログラム名定数、CALL 文、CANCEL 文、ENTRY 文および INVOKE 文の定数中の英小文字も英大文字と等価に扱われます。
ALPHAL(AUTO)	(省略値)COBOL の語は、英小文字と英大文字が適切な表現に自動判定されます。 予約語、利用者語の参照時、REPOSITORY 段落、CALL 文、CANCEL 文、および、INVOKE 文の定数の中の英小文字は英大文字と等価に扱われます。 また、利用者語の定義時および見出し部に指定された定数の中の英小文字は英大文字と区別されます。
ALPHAL(WORD)	COBOL の語は、英小文字と英大文字が等価に扱われます。プログラム名定数、CALL 文、CANCEL 文、ENTRY 文および INVOKE 文の定数を含む、定数中の英小文字は英大文字と区別されます。
NOALPHAL	COBOL の語および定数中の英小文字は、英大文字と区別されます。
APOST	表意定数 QUOTE および QUOTES としてアポストロフィ(')を使用します。
ASCOMP5(NONE)	(省略値)宣言されたとおりに解釈します。
ASCOMP5(ALL)	USAGE BINARY および USAGE COMP、USAGE COMPUTATIONAL と宣言された項目は USAGE COMP-5 が指定されたとみなします。
ASCOMP5(BINARY)	USAGE BINARY と宣言された項目は USAGE COMP-5 が指定されたとみなします。
ASCOMP5(COMP)	USAGE COMP および USAGE COMPUTATIONAL と宣言された項目は USAGE COMP-5 が指定されたとみなします。
BINARY(WORD,MLBON)	(省略値)2 進データの基本項目が、桁数より求められるワード単位の領域長(2,4,8)に割り付けられ、符号なし 2 進項目の最左端ビットは符号として扱われます。
BINARY(WORD,MLBOFF)	2 進データの基本項目が、桁数より求められるワード単位の領域長(2,4,8)に割り付けられ、符号なし 2 進項目の最左端ビットは数値として扱われます。
BINARY(BYTE)	2 進データの基本項目が、桁数より求められるバイト単位の領域長（1～8 バイト）に割り付けられ、符号なし 2 進項目の最左端ビットは数値として扱われます。
CHECK(ALL)	NUMERIC および BOUND の検査を行います。
CHECK(NUMERIC)	データ例外(属性形式に合った値が数字項目に入っているかおよび除数がゼロでないか)の検査を行います。
CHECK(BOUND)	添字・指標および部分参照の範囲外検査を行います。
NOCHECK	(省略値)CHECK 機能を使用しません。

COPY	ソースプログラムリスト内に、COPY 文によって組み込まれる登録集原文を表示します。また、翻訳オプション SOURCE を指定した場合だけ意味を持ちます。
NOCOPY	(省略値)ソースプログラムリスト内に、COPY 文によって組み込まれる登録集原文を表示しません。
CURRENCY(¥)	(省略値)通貨編集用文字として、¥を使用します。
CURRENCY($)	通貨編集用文字として、$を使用します。
DECIMAL(FJ)	(省略値)SEPARATE 指定なしの外部 10 進項目の内部表現を富士通形式とします。
DECIMAL(MF)	SEPARATE 指定なしの外部 10 進項目の内部表現を MF 互換形式とします。
DECIMAL(88)	SEPARATE 指定なしの外部 10 進項目の内部表現を 88 コンソーシアム形式とします。
DLOAD	プログラム名を定数で指定した CALL 文および CANCEL 文のプログラムを動的構造にします。
DUPCHAR(EXT)	コンパイラが付加／置換する以下の JIS 非漢字の負号を、拡張文字とします。
DUPCHAR(STD)	コンパイラが付加／置換する以下の JIS 非漢字の負号を、標準文字とします。
NODLOAD	(省略値)プログラム名を定数で指定した CALL 文および CANCEL 文のプログラムを動的構造にしません。
EQUALS	実行時に SORT 文の入力中に同一キーを持つレコードが複数個存在する場合、それらに関して SORT 文の出力レコード順を入力レコード順と同じにすることを保証します。
NOEQUALS	(省略値)実行時に SORT 文の入力中に同一キーを持つレコードが複数個存在する場合、それらに関して SORT 文の出力レコード順を入力レコード順と同じにすることを保証しません。 この時、出力レコード順は規定されません。
FLAG(I)	(省略値)すべての診断メッセージを表示します。
FLAG(W)	W レベル以上の診断メッセージだけ表示します。
FLAG(E)	E レベル以上の診断メッセージだけ表示します。
FLAGSW(STDM)	COBOL 文法の言語要素に対して、COBOL 規格の下位レベル外の指摘メッセージを表示します。
FLAGSW(STDI)	COBOL 文法の言語要素に対して、COBOL 規格の中位レベル外の指摘メッセージを表示します。
FLAGSW(STDH)	COBOL 文法の言語要素に対して、COBOL 規格の上位レベル外の指摘メッセージを表示します。
FLAGSW(SIA)	COBOL 文法の言語要素に対して、富士通システム統合アーキテクチャ(SIA)の範囲外の指摘メッセージを表示します。
NOFLAGSW	(省略値)COBOL 文法の言語要素に対する指摘メッセージを表示しません。

INITONLY	クラス定義のスタティックデータすべてに、初期化だけ可能な INITOLY 属性を付加します。
NOINITONLY	(省略値)初期化だけ可能な INITOLY 属性を付加しません。
INITVALUE(xx)	作業場所節データの VALUE 句なし項目を指定値で初期化します。 xx は 2 桁の 16 進数を指定してください。
NOINITVALUE	(省略値)作業場所節データの VALUE 句なし項目を初期化しません。
LINECOUNT(nnn)	翻訳リストの 1 ページあたりの行数を指定します。nnn は、3 桁以内の整数を指定してください。 nnn を省略した場合、0 が指定されたものとみなします。0 から 12 までの値を指定すると、ページ替えのないベタ打ち表示となります。
LINESIZE(nnn)	翻訳リストの 1 行あたりの最大文字数(リスト上に表示される A/N 文字換算の値)を指定します。 nnn は、80 および 120 以上の 3 桁の整数を指定することができます。 nnn を省略した場合、136 が指定されたものとみなします。
MESSAGE	オプション情報リストおよび翻訳単位統計情報リストを出力します。
NOMESSAGE	(省略値)オプション情報リストおよび翻訳単位統計情報リストを出力しません。
MODE(CCVS)	ACCEPT 文の“ACCEPT 一意名 [FROM 呼び名]”の書き方で、受取り側項目に数字項目を指定した ACCEPT 文を実行する場合、受取り側項目に右詰めの文字転記を行います。数字項目としては、外部 10 進項目だけが ACCEPT 文の受取り側項目として指定できます。
MODE(STD)	(省略値)ACCEPT 文の“ACCEPT 一意名 [FROM 呼び名]”の書き方で、受取り側項目に数字項目を指定した ACCEPT 文を実行する場合、受取り側項目に右詰めの数字転記を行います。
NCW(STD)	(省略値)利用者語に指定できる日本語文字集合をシステム共通な日本語文字集合とします。
NCW(SYS)	利用者語に指定できる日本語文字集合を計算機の日本語文字集合とします。
NUMBER	翻訳時および実行時の各種リストで、ソースプログラム中の各行を識別するための行情報の行番号に、ソースプログラムの一連番号領域の値を使用します。一連番号領域に数字以外の文字が含まれている場合および一連番号が昇順になっていない場合、その行の行番号は、直前の正しい一連番号に 1 を加えた値に変更されます。ただし、一連番号が降順となる場合、一意の補正された番号が COPY 修飾値と同じ形式で付加されます。
NONUMBER	(省略値)翻訳時および実行時の各種リストで、ソースプログラム中の各行を識別するための行情報の行番号に、コンパイラが生成した値を使用します。行番号は、1 から 1 きざみに昇順に与えられます。
OPTIMIZE	広域最適化された目的プログラムを作成します。
NOOPTIMIZE	(省略値)広域最適化された目的プログラムを作成しません。
PGMNAME(ALL)	(省略値)帳票定義体をプログラミング名で展開します。
PGMNAME(MED)	帳票定義体だけプログラミング名で展開します。

PGMNAME(NO)	帳票定義体を英数字項目名で展開します。
QUOTE	(省略値)表意定数 QUOTE および QUOTES としてクォーテーションマーク(”) を使用します。
RCS(SJIS)	実行時データのコード系としてシフト JIS を使用します。
RCS(SJIS-UCS2)	実行時データのコード系としてシフト JIS+Unicode を使用します。
RCS(UTF8-UCS2)	(省略値)実行時データのコード系として Unicode を使用します。
RSV(ALL)	(省略値)本バージョンの予約語を使用します。
RSV(V111)	OSIV COBOL85 V11L11 用の予約語を使用します。
RSV(V112)	OSIV COBOL85 V11L20 用の予約語を使用します。
RSV(V122)	OSIV COBOL85 V12L20 用の予約語を使用します。
RSV(V125)	COBOL85 V12L50 用の予約語を使用します。
RSV(V30)	COBOL85 V30 用の予約語を使用します。
RSV(V40)	COBOL97 V40 用の予約語を使用します。
RSV(V61)	COBOL97 V61 用の予約語を使用します。
RSV(V70)	NetCOBOL for .NET V2.0 および Windows 版 NetCOBOL V70 用の予約語を使用します。
RSV(V81)	NetCOBOL for .NET V2.1 および Windows 版 NetCOBOL V80 用の予約語を使用します。
RSV(V90)	NetCOBOL for .NET V3.0 および Windows 版 NetCOBOL V90 用の予約語を使用します。
RSV(V91)	NetCOBOL for .NET V3.1 および V4.0 用の予約語を使用します。
RSV(V1050)	NetCOBOL for .NET V4.1、V4.2、V5.0 および Windows 版 NetCOBOL NetCOBOL V10 用の予約語を使用します。
RSV(V1100)	NetCOBOL for .NET V7.0 および Windows 版 NetCOBOL NetCOBOL V11.0 用の予約語を使用します。
RSV(VSR2)	VS COBOL II REL2.0 用の予約語を使用します。
RSV(VSR3)	VS COBOL II REL3.0 用の予約語を使用します。
SCS(ACP)	(省略値)COBOL ソースファイルおよび登録集ファイルのコード系を ACP とみなします。
SCS(UTF8)	COBOL ソースファイルおよび登録集ファイルのコード系を UTF8 とみなします。

SDS	(省略値)符号付き内部 10 進項目から符号付き内部 10 進項目への転記で、送出し側項目の符号をそのまま転記します。
NOSDS	符号付き内部 10 進項目から符号付き内部 10 進項目への転記で、送出し側項目の符号を整形して転記します。負号には X‘B'および X‘D'の 2 種類があり、その他は正号として扱われます。ここでいう符号の整形とは、送出し側項目の符号が正ならば X‘C'に、負ならば X‘D'に変換することです。
SHREXT	外部属性(EXTERNAL 指定)のデータおよびファイルをスレッド間で共有します。
NOSHREXT	(省略値)外部属性(EXTERNAL 指定)のデータおよびファイルをスレッド間で共有しません。
SMSIZE(nK)	PowerSORT が使用するメモリ容量をキロバイト単位で指定します。 n を省略された場合、 "0" が指定されたものとします。
SOURCE	ソースプログラムリストを/print コンパイラオプションで指定されたフォルダに出力します。/print コンパイラオプションが指定されていない場合、本オプションを指定しても、ソースプログラムリストは出力されません。
NOSOURCE	(省略値)ソースプログラムリストは出力しません。
SQLSCOPE(METHOD)	カーソル・文識別子は、これを定義したメソッド内のスコープを持ち、メソッドから実行の制御が戻るまで有効となります。
SQLSCOPE(OBJECT)	(省略値)オブジェクトメソッド中で定義されたカーソル・文識別子は、オブジェクト定義内のスコープを持ち、オブジェクトが有効な限り有効となります。また、スタティックメソッド中で定義されたカーソル・文識別子は、スタティック定義内のスコープを持ち、実行単位の終了まで有効となります。
SRF(FIX,FIX)	COBOL ソースプログラムおよび登録集ファイルの正書法を固定形式とします。
SRF(FIX,VAR)	COBOL ソースプログラムの正書法を固定形式、登録集ファイルの正書法を可変形式とします。
SRF(FIX,FREE)	COBOL ソースプログラムの正書法を固定形式、登録集ファイルの正書法を自由形式とします。
SRF(VAR,FIX)	COBOL ソースプログラムの正書法を可変形式、登録集ファイルの正書法を固定形式とします。
SRF(VAR,VAR)	(省略値)COBOL ソースプログラムおよび登録集ファイルの正書法を可変形式とします。
SRF(VAR,FREE)	COBOL ソースプログラムの正書法を可変形式、登録集ファイルの正書法を自由形式とします。
SRF(FREE,FIX)	COBOL ソースプログラムの正書法を自由形式、登録集ファイルの正書法を固定形式とします。
SRF(FREE,VAR)	COBOL ソースプログラムの正書法を自由形式、登録集ファイルの正書法を可変形式とします。
SRF(FREE,FREE)	COBOL ソースプログラムおよび登録集ファイルの正書法を自由形式とします。

SSIN(環境変数情報名)	小入出力機能の ACCEPT 文のデータの入力先としてファイルを使用します。環境変数情報名には、実行時にファイルのパス名を設定します。環境変数情報名は英大文字(A～Z)で始まる 8 文字以内の英大文字および数字でなければなりません。また、環境変数情報名は、他のファイルで使用する環境変数情報名(ファイル識別名)と一致しないようにする必要があります。
SSIN(SYSIN)	(省略値)小入出力機能の ACCEPT 文のデータの入力先としてコンソールウィンドウを使用します。
SSOUT(環境変数情報名)	小入出力機能の DISPLAY 文のデータの出力先としてファイルを使用します。環境変数情報名には、実行時にファイルのパス名を設定します。環境変数情報名は英大文字(A～Z)で始まる 8 文字以内の英大文字または数字でなければなりません。また、環境変数情報名は、他のファイルで使用する環境変数情報名(ファイル識別名)と一致しないようにする必要があります。
SSOUT(SYSOUT)	(省略値)小入出力機能の DISPLAY 文のデータの出力先としてコンソールウィンドウを使用します。
TAB(4)	タブの扱いを 4 カラム単位とします。ただし、値としてのタブは、タブ値そのものです。
TAB(8)	(省略値)タブの扱いを 8 カラム単位とします。ただし、値としてのタブは、タブ値そのものです。
TRUNC	2 進項目を受取り側項目とする数字転記で、結果の値が受取り側項目の PICTURE 句の記述に従って、上位桁が桁落としされ、受取り側項目に格納されます。
NOTRUNC	(省略値)2 進項目を受取り側項目とする数字転記で、桁落としを行わないほうが実行が速くなる場合には、桁落としは行われません。
USEEXTRET	例外オブジェクトに対する USE 手続き終了後、USE への移行原因の文の直後に復帰し処理を継続します。
NOUSEEXTRET	(省略値)例外オブジェクトに対する USE 手続き終了後、暗の EXIT METHOD(または、EXIT PROGRAM)を実行します。
XREF	相互参照リストを出力します。/print コンパイラオプションが指定されていない場合、本オプションを指定しても無効となります。
NOXREF	(省略値)相互参照リストを出力しません。
ZWB	(省略値)符号付き外部 10 進項目を英数字フィールドと比較するときに、外部 10 進項目の符号部を無視して比較します。ここで、英数字とは、英数字項目、英字項目、英数字編集項目、数字編集項目、文字定数および ZERO 以外の表意定数のことです。
NOZWB	符号付き外部 10 進項目を英数字フィールドと比較するときに、外部 10 進項目の符号部を含めて比較します。ここで、英数字とは、英数字項目、英字項目、英数字編集項目、数字編集項目、文字定数および ZERO 以外の表意定数のことです。

翻訳オプション - トピック順一覧

翻訳リストに関するもの	翻訳オプション
登録集原文の表示	COPY
翻訳リストの 1 ページあたりの行数	LINECOUNT
翻訳リストの 1 行あたりの文字数	LINESIZE
オプション情報リスト、翻訳単位統計情報リストの出力の可否	MESSAGE
ソースプログラムの一連番号領域の指定	NUMBER
ソースプログラムリストの出力の可否	SOURCE
相互参照リストの出力の可否	XREF
翻訳時メッセージに関するもの	翻訳オプション
診断メッセージのレベル	FLAG
COBOL 文法の言語要素に対しての指摘メッセージ表示の可否	FLAGSW
COBOL プログラムの解釈に関するもの	翻訳オプション
プログラム中の英小文字の扱い	ALPHAL
2 進項目の扱い	BINARY
通貨編集用文字の扱い	CURRENCY
重複文字の扱い	DUPCHAR
作業場所節での VALUE 句なし項目の扱い	INITVALUE
日本語利用者語の文字集合の指定	NCW
帳票定義体の項目名の展開方法の指定	PGMNAME
表意定数 QUOTE の扱い	APOST
予約語の種類	RSV
ソースプログラムのコード系	SCS
符号付き 10 進項目の符号の整形の可否	SDS
外部属性に関する扱い	SHREXT

正書法の種類	SRF
タブの扱い	TAB
符号付き外部 10 進項目と英数字項目の比較	ZWB
目的プログラムの作成に関するもの	翻訳オプション
ACCEPT 文の動作の指定	MODE
2 進項目の解釈の指定	ASCOMP5
SEPARATE 指定なしの外部 10 進項目の内部表現形式の指定	DECIMAL
INITONLY 属性の指定	INITONLY
広域最適化の扱い	OPTIMIZE
実行時データのコード系の指定	RCS
埋込み SQL 文のカーソル・文識別子のスコープと寿命の指定	SQLSCOPE
プログラムの動的構造の可否	DLOAD
実行時の処理に関するもの	翻訳オプション
SORT 文での同一キーデータの処理方法	EQUALS
桁落とし処理の可否	TRUNC
USE 手続き終了後の動作指定	USEEXTRET
実行時の資源に関するもの	翻訳オプション
PowerSORT が使用するメモリ容量を指定	SMSIZE
ACCEPT 文のデータの入力先	SSIN
DISPLAY 文のデータの出力先	SSOUT
実行時のデバッグ機能に関するもの	翻訳オプション
CHECK 機能の使用の可否	CHECK

翻訳オプション - 詳細

翻訳オプションの詳細について説明します。

翻訳オプション	用途
ALPHAL	プログラム中の英小文字の扱い
APOST	表意定数 QUOTE の扱い
ASCOMP5	2 進項目の解釈
BINARY	2 進項目の扱い
CHECK	CHECK 機能の使用の可否
COPY	登録集原文の表示
CURRENCY	通貨編集用文字の扱い
DECIMAL	SEPARATE 指定なしの外部 10 進項目の内部表現形式の指定
DUPCHAR	重複文字の扱い
DLOAD	プログラムの動的構造の可否
EQUALS	SORT 文での同一キーデータの処理方法
FLAG	診断メッセージのレベル
FLAGSW	COBOL 文法の言語要素に対しての指摘メッセージ表示の可否
INITONLY	INITONLY 属性の指定
INITVALUE	作業場所節での VALUE 句なし項目の扱い
LINECOUNT	翻訳リストの 1 ページあたりの行数
LINESIZE	翻訳リストの 1 行あたりの文字数
MESSAGE	オプション情報リスト、翻訳単位統計情報リストの出力の可否
MODE	ACCEPT 文の動作の指定
NCW	日本語利用者語の文字集合の指定
NUMBER	ソースプログラムの一連番号領域の指定
OPTIMIZE	広域最適化の扱い

PGMNAME	帳票定義体の項目名の展開方法の指定
QUOTE	表意定数 QUOTE の扱い
RSV	予約語の種類
RCS	実行時のコード系
SCS	ソースプログラムのコード系
SQLSCOPE	埋込み SQL 文のカーソル・文識別子のスコープと寿命の指定
SDS	符号付き 10 進項目の符号の整形の可否
SHREXT	外部属性に関する扱い
SMSIZE	PowerSORT が使用するメモリ容量を指定
SOURCE	ソースプログラムリストの出力の可否
SRF	正書法の種類
SSIN	ACCEPT 文のデータの入力先
SSOUT	DISPLAY 文のデータの出力先
TAB	タブの扱い
TRUNC	桁落とし処理の可否
USEEXTRET	USE 手続き終了後の動作指定
XREF	相互参照リストの出力の可否
ZWB	符号付き外部 10 進項目と英数字項目の比較

ALPHAL (プログラム中の英小文字の扱い)

ソースプログラム中の英小文字を英大文字と等価に扱う(ALPHAL)か、扱わない(NOALPHAL)かを指定します。

```
ì             ì ALL    ü ü
ï ALPHAL[(í AUTO   ý)]ï
í             î WORD   þ ý
ï                      ï
î NOALPHAL             þ
```

ALPHAL が有効な場合、文字定数は以下のように扱われます。

ALPHAL(ALL):

英小文字と英大文字が等価に扱われます。 また、プログラム名定数、AS 指定の定数、CALL 文、CANCEL 文、ENTRY 文および INVOKE 文の定数中の英小文字も英大文字と等価に扱われます。

ALPHAL(AUTO):

英小文字と英大文字が適切な表現に自動判定されます。 予約語、利用者語の参照時、REPOSITORY 段落、CALL 文、CANCEL 文、および、INVOKE 文の定数中の英小文字は英大文字と等価に扱われます。 また、利用者語の定義時および見出し部に指定された定数中の英小文字は英大文字と区別されます。

ALPHAL(WORD):

英小文字と英大文字が等価に扱われます。 プログラム名定数、AS 指定の定数、CALL 文、CANCEL 文、ENTRY 文および INVOKE 文の定数を含む、定数中の英小文字は英大文字と区別されます。

NOALPHAL

COBOL の語および定数中の英小文字は、英大文字と区別されます。 COBOL の語については、COBOL 文法書「1.2.2 COBOL の語」を参照してください。

NCOBU00598

APOST/QUOTE (表意定数 QUOTE の扱い)

表意定数 QUOTE および QUOTES としてクォーテーションマーク(") を使う(QUOTE)か、アポストロフィ(')を使う(APOST)かを指定します。

```
ì QUOTE ü
í       ý
î APOST þ
```

ソースプログラム中の引用符は、このオプションの指定に関係なく、クォーテーションマークとアポストロフィのどちらでも使用できます。 ただし、左側の引用符と右側の引用符は、同じでなくてはなりません。

ASCOMP5 (2 進項目の解釈)

2 進項目の解釈を指定します。

```
              ì NONE  ü
              ï ALL   ï
ASCOMP5(ï     í BINARY ý)
              ï COMP  ï
              î       þ
```

ASCOMP5(NONE):

宣言されたとおりに解釈します。

ASCOMP5(ALL):

USAGE BINARY および USAGE COMP、USAGE COMPUTATIONAL と宣言された項目は USAGE COMP-5 が指定されたとみなします。

ASCOMP5(BINARY):

USAGE BINARY と宣言された項目は USAGE COMP-5 が指定されたとみなします。

ASCOMP5(COMP):

USAGE COMP、USAGE COMPUTATIONAL と宣言された項目は USAGE COMP-5 が指定されたとみなします。

本オプションは実行性能の向上を目的に使用します。

データの内部表現が変わるため、内部表現を意識したコーディングが含まれている場合には注意してください。

CBL ルーチンを使用する場合は、NONE を指定してください。

NCOBU00600

BINARY (2 進項目の扱い)

2 進データの基本項目が､桁数より求められるワード単位の領域長(2,4,8)に割り付けられる(BINARY(WORD))か､バイト単位の領域長(1～8)に割り付けられる(BINARY(BYTE))かを指定します。 なお、符号なし 2 進項目の最左端ビットの扱いも指定できます。

```
          ì          ì MLBON  ü ü
          ï WORD[,í           ý]ï
BINARY(í          î MLBOFFþ ý)
          ï                  ï
          î BYTE             þ
```

BINARY(WORD,MLBON):

最左端ビットは符号

BINARY(WORD,MLBOFF):

最左端ビットは数値

BINARY(BYTE)

最左端ビットは数値

宣言した桁数と、割り当てられる領域長の関係は、下表のとおりです。

PIC 桁数から割り当てられる領域長

PIC の桁数		割り当てられる領域長	
符号付き	符号なし	BINARY(BYTE)	BINARY(WORD)
1 - 2	1 - 2	1	2
3 - 4	3 - 4	2	2
5 - 6	5 - 7	3	4
7 - 9	8 - 9	4	4
10 - 11	10 - 12	5	8
12 - 14	13 - 14	6	8
15 - 16	15 - 16	7	8
17 - 18	17 - 18	8	8

CHECK (CHECK 機能の使用の可否)

CHECK 機能を使用する(CHECK)か、しない(NOCHECK)かを指定します。

```
ì CHECK[([ALL][,NUMERIC][,BOUND])]ü
í                                 ý
î NOCHECK                         þ
```

CHECK(ALL):

NUMERIC および BOUND の検査を行います。

CHECK(NUMERIC):

データ例外(属性形式に合った値が数字項目に入っているかおよび除数がゼロでないか)の検査を行います。

CHECK(BOUND):

添字・指標および部分参照の範囲外検査を行います。

- CHECK を指定すると、上記の検査をするための処理が目的プログラム中に組み込まれるため、実行性能が低下します。デバッグ終了時には、NOCHECK を指定して再翻訳してください。
- ON SIZE ERROR 指定または NOT ON SIZE ERROR 指定の算術文では、ON SIZE ERROR の除数のゼロ検査が行われ、CHECK(NUMERIC)の除数のゼロ検査は行われません。
- CHECK(NUMERIC)のデータ例外検査は、外部 10 進項目または内部 10 進項目が参照で使用される場合、および、英数字項目または集団項目から、外部 10 進項目または内部 10 進項目へ転記される場合に行われます。 ただし次は、チェックの対象とはなりません。
    1. 添字として ALL が指定されている表要素
    2. SEARCH ALL 文におけるキー項目(ただしキー項目に対する添字が 1 次元かつ WHEN 条件がひとつだけである場合は除く)
    3. SORT/MERGE 文におけるキー項目
    4. SQL 文で使用されているホスト変数
    5. CALL 文、INVOKE 文および行内呼出しの BY REFERENCE パラメタ
    6. 次の組込み関数の引数
        - FUNCTION ADDR
        - FUNCTION LENG
        - FUNCTION LENGTH
    7. 英数字項目または集団項目から、外部 10 進項目または内部 10 進項目のオブジェクトプロパティへの転記

- CHECK 機能は、CHECK オプションを指定して翻訳したプログラムにだけ有効になります。複数のプログラムをリンクしている場合に特定のプログラムだけを CHECK 機能の対象にするには、対象にするプログラムは CHECK オプションを指定して翻訳し、対象にしないプログラムは CHECK オプションを指定しないで翻訳してください。

COPY (登録集原文の表示)

ソースプログラムリスト内に、COPY 文によって組み込まれる登録集原文を表示する(COPY)か、しない(NOCOPY)かを指定します。

```
¡COPY    ü
í        ý
îNOCOPYþ
```

COPY は、翻訳オプション SOURCE を指定した場合だけ意味を持ちます。

NCOBU00603

CURRENCY (通貨編集用文字の扱い)

通貨編集用文字として使用している文字に、¥を使用する(CURRENCY(¥))か、$を使用する(CURRENCY($))かを指定します

```
            ì¥ü
CURRENCY(í  ý)
            î$þ
```

DECIMAL (SEPARATE 指定なしの外部 10 進項目の内部表現形式の指定)

SEPARATE 指定なしの外部 10 進項目の内部表現を、富士通形式(DECIMAL(FJ))とするか、 MF 互換形式(DECIMAL(MF))とするか、88 コンソーシアム形式(DECIMAL(88))とするかを指定します。

```
              ì FJ ü
DECIMAL(í     MF ý)
              î 88 þ
```

NCOBU00991

DLOAD (プログラムの動的構造の可否)

プログラム名を定数で指定した CALL 文および CANCEL 文のプログラムを動的構造にする(DLOAD)か、動的構造にしない(NODLOAD)かを指定します。

```
¡DLOAD    ü
í         ý
îNODLOADþ
```

- NODLOAD の場合、ビルド時に呼び出すプログラムを同一の実行可能ファイルに含めるか、コンパイラオプション/reference に指定する必要があります。翻訳時にパラメタの個数と長さをチェックします。
- DLOAD の場合、翻訳時に呼び出すプログラムを参照しないため、呼び出すプログラムが存在しなくても実行可能ファイルが作成できるメリットがあります。ただし、翻訳時にパラメタをチェックしないため、パラメタ誤りの場合は、実行時エラーやデータ領域破壊となるので注意が必要です。また、NODLOAD と比べ実行性能は低下します。
- DLOAD の場合、実行時にエントリ情報の指定が必要になります。

DUPCHAR (重複文字の扱い)

ソースファイルが Unicode の場合、コンパイラが付加／置換する以下の JIS 非漢字の負号を、拡張文字(DUPCHAR(EXT))とするか、標準文字(DUPCHAR(STD))とするかを指定します。

- 3 バイト項目制御部を指定した帳票定義体
- COPY 文の書き方 2 と書き方 3 の日本語利用者語

```
              ì EXTü
DUPCHAR(í       ý)
              î STDþ
```

JIS 非漢字の負号の扱い

オプションの指定	文字コード(UTF-8)
DUPCHAR(EXT)	－ (X"EFBC8D")
DUPCHAR(STD)	- (X"E28892")

JIS 非漢字の負号について

COPY 文の書き方 2 と書き方 3(DISJOINING/JOINING 指定)において日本語利用者語の場合に分離符とみなすのは、“JIS 非漢字の負号”です。 Unicode には、見た目で“負号”と識別できるコードが、2 つ割り振られています。

	UTF-8 表現	UTF16 表現
MINUS SIGN	X"E28892"	X"2212"
FULLWIDTH HYPHEN-MINUS	X"EFBC8D"	X"FF0D"

NetCOBOL において、ソースおよび登録集を UTF-8 で作成する場合、 分離符として上記の「FULLWIDTHHYPHEN-MINUS」を採用しています。 このため、日本語利用者語に「MINUS SIGN」を使用していた場合は、意図したとおり動作しません。

ただし、これは翻訳オプション DUPCHAR によって変更することができます。

- DUPCHAR(STD) ： MINUS SIGN
- DUPCHAR(EXT) ： FULLWIDTH HYPHEN-MINUS(省略値)

上記の場合、DUPCHAR(STD)を指定して翻訳してください。

NCOBU00604

EQUALS (SORT 文での同一キーデータの処理方法)

実行時に、SORT 文の入力中に同一キーを持つレコードが複数個存在する場合、それらに関して、SORT 文の出力レコードの順序を SORT 文の入力レコードと同じにすることを保証する(EQUALS)か、しない(NOEQUALS)かを指定します。

```
¡ EQUALS    ü
í           ý
î NOEQUALSþ
```

NOEQUALS を指定すると、SORT 文で同一キーを持つレコードの出力順序は規定されません。

EQUALS を指定すると、整列操作で入力順序を保証するための特別な処理が行われるため、実行性能が低下します。

FLAG (診断メッセージのレベル)

表示する診断メッセージを指定します。

```
        ì I ü
FLAG(í W ý)
        î E þ
```

FLAG(I):

すべての診断メッセージを表示します。

FLAG(W):

W レベル以上の診断メッセージだけ表示します。

FLAG(E):

E レベル以上の診断メッセージだけ表示します。

NCOBU00606

FLAGSW (COBOL 文法の言語要素に対しての指摘メッセージ表示の可否)

COBOL 文法の言語要素に対しての指摘メッセージを表示する(FLAGSW)か、しない(NOFLAGSW)かを指定します。

```
FLAGSW(STDM)
       STDI
       STDH
       SIA
NOFLAGSW
```

以下に指定する言語要素を示します。

FLAGSW(STDM):

COBOL 規格の下位レベル外

FLAGSW(STDI):

COBOL 規格の中位レベル外

FLAGSW(STDH):

COBOL 規格の上位レベル外

FLAGSW(SIA):

富士通システム統合アーキテクチャ(SIA)の範囲外

FLAGSW(SIA) は、他システムで動作させるプログラムを作成する場合に有効です。

INITONLY (INITONLY 属性の指定)

クラス定義のスタティックデータすべてに、初期化だけ可能な INITONLY 属性を付加する(INITONLY)か、付加しない(NOINITONLY)かを指定します。

```
¡ INITONLY    ü
í             ý
î NOINITONLYþ
```

- プログラム定義に対して、このオプションを指定しても意味を持ちません。
- INITONLY を指定する場合、以下の記述はできません。
  - スタティック定義の環境部
    - 入出力節
  - スタティック定義のデータ部
    - ファイル記述項、および、整列併合用ファイル記述項
    - 埋込み SQL 宣言節
    - WITH NO SET 指定がない PROPERTY 句
  - メソッド定義のデータ部
    - 外部属性を持つレコード記述項
    - 外部属性を持つデータ記述項

NCOBU00607

INITVALUE (作業場所節での VALUE 句なし項目の扱い)

作業場所節データの VALUE 句なし項目を指定値で初期化する(INITVALUE)か、しない(NOINITVALUE)かを指定します。 xx は、2 桁の 16 進数を指定してください。xx は省略できません。

```
¡ INITVALUE(xx)ü
í                ý
î NOINITVALUE   þ
```

LINECOUNT (翻訳リストの 1 ページあたりの行数)

翻訳リストの 1 ページあたりの行数を指定します。n は、3 桁以内の整数を指定してください。 n を省略した場合、0 が指定されたものとみなします。

```
LINECOUNT(n)
```

0 から 12 までの値を指定すると、ページ替えのないベタ打ち表示となります。

NCOBU00609

LINESIZE (翻訳リストの 1 行あたりの文字数)

翻訳リストの 1 行あたりの最大文字数(リスト上に表示される A/N 文字換算の値)を指定します。 n は、80 および 120 以上の 3 桁の整数を指定することができます。 n を省略した場合、136 が指定されたものとみなします。

```
LINESIZE(n)
```

- ソースプログラムリスト、オプション情報リスト、診断メッセージリストおよび翻訳単位統計情報リストは、翻訳オプション LINESIZE に指定した最大文字数に関係なく固定の文字数(120)で出力されます。
- 文字数として有効な最大の値は 136 です。 翻訳オプション LINESIZE に 136 より大きい値を指定した場合、136 として扱われます。

MESSAGE (オプション情報リスト、翻訳単位統計情報リストの出力の可否)

オプション情報リストおよび翻訳単位統計情報リストを出力する(MESSAGE)か、しない(NOMESSAGE)かを指定します。

```
¡MESSAGE    ü
í           ý
îNOMESSAGEþ
```

NCOBU00611

MODE (ACCEPT 文の動作の指定)

ACCEPT 文の"ACCEPT 一意名 [FROM 呼び名]"の書き方で、受取り側項目に数字項目を指定した ACCEPT 文を実行する場合、受取り側項目に右詰めの数字転記を行う(MODE(STD))か、左詰めの文字転記を行う(MODE(CCVS))かを指定します。

```
      ì STD  ü
MODE(í       ý)
      î CCVS þ
```

MODE(CCVS)を指定する場合、数字項目としては、外部 10 進項目だけが ACCEPT 文の受取り側項目として指定できます。

NCW (日本語利用者語の文字集合の指定)

利用者語に指定できる日本語文字集合をシステム共通な日本語文字集合とする(NCW(STD))か、計算機の日本語文字集合とする(NCW(SYS))かを指定します。

```
      ì STDü
NCW(í       ý)
      î SYSþ
```

STD を指定すると、次の日本語文字集合が日本語利用者語として利用できます。

- JIS 第一水準
- JIS 第二水準
- JIS 非漢字(以下の文字)
  - ０、１、…、９
  - Ａ、Ｂ、…、Ｚ
  - ａ、ｂ、…、ｚ
  - ぁ、あ、ぃ、い、…、ん
  - ァ、ア、ィ、イ、…、ン、ヴ、ヵ、ヶ
  - ー（長音）、‐（ハイフン）、－（負号）、々

SYS を指定すると、次の日本語文字集合が日本語利用者語として使用できます。

- STD 指定の文字集合
- 拡張文字
- 拡張非漢字
- 利用者定義文字
- JIS 非漢字(以下の文字は使用不可)
  - 、 。 ， ． ・ ： ； ？ ！ ゛
  - ° ゜ ´ ˆ ￣ ＿ ／ ＼ ｜ （ ）
  - ［ ］ ｛ ｝ 「 」 ＋ ＝ ＜ ＞
  - ￥ ＄ ￠ ￡ ％ ＃ ＆ ＊ ＠

NCOBU00613

NUMBER (ソースプログラムの一連番号領域の指定)

翻訳時および実行時の各種リストで、ソースプログラム中の各行を識別するための行情報の行番号に、ソースプログラムの一連番号領域の値を使用する(NUMBER)か、コンパイラが生成した値を使用する(NONUMBER)かを指定します。

```
¡ NUMBER    ü
í           ý
î NONUMBERþ
```

NUMBER:

一連番号領域に数字以外の文字が含まれている場合および一連番号が昇順になっていない場合、その行の行番号は、直前の正しい一連番号に 1 を加えた値に変更されます。

一連番号領域	例	行数出力	例
連続した数字の場合	0100 0200 0300 0400	同様に連続した数字として出力します。	0100 0200 0300 0400
数字以外の文字だけで書かれている場合	xxx xxx yyy zzz	コンパイラによって 1 からはじまる連続する数字として出力します	1 2 3 4
数字と数字以外の文字で書かれている場合	0100 xxx ccc zzz	数字以外で書かれている行に関して、現在かかれている数字から 1 づつ増加した数字が出力されます。	0100 0101 0102 0103
ある行は、数字でかかれているが、ある行は、数字以外の文字で書かれている場合。 連続する数字の間に数字以外の行が挿入されたような場合	0100 xxx ccc 0101	現在使用している数字が使われ、それ以降で数字以外の行について、数字部分から 1 づつ増加した数字が書かれる。 重複する数字がある場合は、COPY ファイルのフォーマットで使われているように、臨時の数字を追加する。	0100 0101 0102 1-0101
行に降順で数字が書かれている場合	0400 0300 0200 0100	COPY ファイルにラベルするような感じで、昇順である数字があるように、現在ある数字にラベルを追加する。	0400 1-0300 2-0200 3-0100

NONUMBER:

行番号は、1 から 1 きざみに昇順に与えられます。

- NUMBER が指定されているときには、同一の一連番号が連続していても誤りではないものとみなされます。

- NUMBER を指定した場合、ビルダのエラージャンプ機能は使用できません。
- NUMBER を指定した場合、翻訳オプション SRF に FREE は指定できません。 翻訳オプション SRF に FREE を指定した場合、プログラムの動作は保証されません。

NCOBU00614

OPTIMIZE (広域最適化の扱い)

広域最適化された目的プログラムを作成する(OPTIMIZE)か、しない(NOOPTIMIZE)かを指定します。

```
¡ OPTIMIZE    ü
í             ý
î NOOPTIMIZEþ
```

PGMNAME (帳票定義体の項目名の展開方法の指定)

帳票定義体から COPY 文を使ってレコード定義を取り込むときに展開される項目名を、プログラミング名で展開するか、英数字項目名で展開するかを指定します。

```
          ¡ALL ü
          ï    ï
PGMNAME(í MEDý)
          ï    ï
          î NO  þ
```

PGMNAME(ALL):

帳票定義体をプログラミング名で展開します。

PGMNAME(MED):

帳票定義体だけプログラミング名で展開します。

PGMNAME(NO):

帳票定義体を英数字項目名で展開します。

- プログラミング名での展開を指定した場合、項目定義にプログラミング名が存在しなければ、英数字項目名で展開します。
- 翻訳オプション PGMNAME は、COBOL G 資産の移行を支援する目的で提供しています。COBOL G ではプログラム定義だけしか存在しないため、プログラム定義以外(クラス定義やメソッド定義など)の翻訳では、翻訳オプション PGMNAME は指定できません。

NCOBU00616

RCS (実行時データのコード系)

実行時データのコード系を指定します。

- 英数字字類および日本語字類ともにシフト JIS にする場合、RCS(SJIS)を指定します。
- 英数字字類をシフト JIS、日本語字類を Unicode(UCS-2)にする場合、RCS(SJIS-UCS2)を指定します。
- 英数字字類を Unicode(UTF-8)、日本語字類を Unicode(UCS-2)にする場合、RCS(UTF8-UCS2)を指定します。

```
    ì SJIS       ü
RCS(í SJIS-UCS2 ý)
    î UTF8-UCS2 þ
```

翻訳オプション RCS を指定した場合のアプリケーションの動作については、Unicode を参照してください。

SCS(UTF8)を指定した場合、RCS(SJIS)および RCS(SJIS-UCS2)は指定できません。 この場合は、RCS(UTF8-UCS2)と見なされます。

RSV (予約語の種類)

COBOL コンパイラが使用する予約語の種類を指定します。 新規に追加された予約語と利用者語の名前がかち合った場合に旧バージョンと同じ予約語を使用して翻訳します。

```
      ì ALL  ü
      ï V111 ï
      ï V112 ï
      ï V122 ï
      ï V125 ï
      ï V30  ï
      ï V40  ï
      í V61  ý
RSV(ï V70  ï)
      ï V81  ï
      ï V90  ï
      ï V91  ï
      ï V1050ï
      ï V1100ï
      ï VSR2 ï
      î VSR3 þ
```

以下に予約語集合名の意味を示します。

RSV(ALL):

本バージョン用

RSV(V111):

OSIV COBOL85 V11L11 用

RSV(V112):

OSIV COBOL85 V11L20 用

RSV(V122):

OSIV COBOL85 V12L20 用

RSV(V125):

COBOL85 V12L50 用

RSV(V30):

COBOL85 V30 用

RSV(V40):

COBOL97 V40 用

RSV(V61):

COBOL97 V61 用

RSV(V70):

NetCOBOL for .NET V2.0 用、 Windows 版 NetCOBOL V70 用

RSV(V81):

NetCOBOL for .NET V2.1 用、 Windows 版 NetCOBOL V80 用

RSV(V90):

NetCOBOL for .NET V3.0 用、 Windows 版 NetCOBOL V90 用

RSV(V91):

NetCOBOL for .NET V3.1 および V4.0 用

RSV(V1050):

NetCOBOL for .NET V4.1、V4.2、V5.0 および Windows 版 NetCOBOL V10 用

RSV(V1100):

NetCOBOL for .NET V7.0 および Windows 版 NetCOBOL V11.0 用

RSV(VSR2):

VS COBOL II REL2.0 用

RSV(VSR3):

VS COBOL II REL3.0 用

SCS (ソースプログラムのコード系)

COBOL ソースプログラムおよび登録集ファイルのコード系を ACP にする(ACP)か、UTF-8 にする(UTF8)かを指定します。

```
     ì ACP  ü
SCS(í       ý)
     î UTF8þ
```

このオプションはソースプログラム中の翻訳指示文(@OPTIONS)に指定できません。

翻訳オプション SCS に UTF8 を指定する場合、COBOL ソースプログラムおよび登録集ファイルは、UTF-8 の BOM(シグネチャとも言う)を付加した UTF-8 コードのファイルとして作成する必要があります。

NCOBU00619

SDS (符号付き 10 進項目の符号の整形の可否)

符号付き内部 10 進項目から符号付き内部 10 進項目への転記で、送出し側項目の符号をそのまま転記する(SDS)か、整形された符号を転記する(NOSDS)かを指定します。

```
ì SDS    ü
í        ý
î NOSDSþ
```

負号には X'B'および X'D'の 2 種類があり、その他は正号として扱われます。 ここでいう整形された符号とは、送出し側項目の符号が正ならば X'C'に、負ならば X'D'に変換することです。

SHREXT (外部属性に関する扱い)

外部属性(EXTERNAL 指定)のデータおよびファイルをスレッド間で共有する(SHREXT)か、共有しない(NOSHREXT)かを指定します。

```
¡SHREXT    ü
í          ý
îNOSHREXTþ
```

SHREXT でコンパイルしたプログラムの外部属性のついたデータおよびファイルは、NOSHREXT でコンパイルしたプログラムの同じ名前の外部属性のついたデータおよびファイルと共有されません。 データおよびファイルを共有する場合には同じオプションを指定してください。

NCOBU00621

SMSIZE (PowerSORT が使用するメモリ容量を指定)

PowerSORT が使用するメモリ容量をキロバイト単位の数字で指定します。

```
SMSIZE(nK)
```

このオプションは、別製品 'PowerSORT' をインストールしている場合に有効であり、インストールされていない場合は無効になります。

PowerSORT がインストールされている場合に、SORT 文および MERGE 文から呼び出される PowerSORT が使用するメモリ空間の容量を指定したい場合に指定します。 指定する値は、キロバイト単位の数字です。 指定された値は PowerSORT に通知します。 指定された値が実際に有効になるかについては、PowerSORT の 'オンラインマニュアル' を参照してください。

このオプションは、実行時オプション 'smsize' および特殊レジスタ 'SORT-CORE-SIZE' に指定する値の意味と等価ですが、同時に指定された場合の優先順位は以下のようになっています。

強い　　弱い

特殊レジスタ'SORT-CORE-SIZE'　>　実行時オプション'smsize'　>　翻訳オプション'SMSIZE(nK)'

例

特殊レジスタ	MOVE 102400 TO SORT-CORE-SIZE *> (102400=100 キロです)
翻訳オプション	SMSIZE(500K)
実行時オプション	smsize300k

この場合、一番強い特殊レジスタ 'SORT-CORE-SIZE' の値 100 キロバイトが優先します。

SOURCE (ソースプログラムリストの出力の可否)

ソースプログラムリストを出力する(SOURCE)か、しない(NOSOURCE)かを指定します。 SOURCE 指定された場合、コンパイラオプション/print で指定されたファイルに出力されます。

```
¡SOURCE    ü
í          ý
îNOSOURCEþ
```

コンパイラオプション/print が指定されていない場合、本オプションを指定しても、ソースプログラムリストは出力されません。

NCOBU00885

SQLSCOPE (埋込み SQL 文のカーソル・文識別子のスコープと寿命の指定)

埋込み SQL 文のカーソル・文識別子のスコープと寿命を指定します。

```
          ¡METHODü
SQLSCOPE(í        ý)
          îOBJECT þ
```

SQLSCOPE(METHOD)

カーソル・文識別子はこれを定義したメソッド内のスコープを持ち、メソッドから実行の制御が戻るまで有効となります。

SQLSCOPE(OBJECT)

オブジェクトメソッド中で定義されたカーソル・文識別子はオブジェクト定義内のスコープを持ち、オブジェクトが有効な限り有効となります。 また、スタティックメソッド中で定義されたカーソル・文識別子はスタティック定義内のスコープを持ち、実行単位の終了まで有効となります。

プログラム定義に対して、このオプションを指定しても意味を持ちません。

SRF (正書法の種類)

COBOL ソースプログラムおよび登録集ファイルの正書法の種類を、固定形式にする(FIX)か、自由形式にする(FREE)か、可変形式にする(VAR)かを指定します。

```
     ì FIX  ü ì FIX  ü
SRF(í FREEý[,í FREEý])
     î VAR  þ î VAR  þ
```

正書法の種類は、最初に COBOL ソースプログラムを、次に登録集を指定します。

登録集の指定を省略した場合、COBOL ソースプログラムに指定した正書法の種類となります。

このオプションはソースプログラムファイルに翻訳単位が 1 つだけの場合、翻訳指示文(@OPTIONS)に指定可能です。

翻訳オプション SRF に FREE を指定した場合、翻訳オプション NUMBER は指定できません。 翻訳オプション NUMBER を指定した場合、プログラムの動作は保証されません。

ソースファイルと登録集ファイルの正書法が同じ場合、登録集ファイルの正書法の指定は省略することができます。

NCOBU00624

SSIN (ACCEPT 文のデータの入力先)

小入出力機能の ACCEPT 文のデータの入力先を指定します。

```
      ì環境変数情報名ü
SSIN(í               ý)
      îSYSIN         þ
```

SSIN(環境変数情報名):

データの入力先としてファイルを使用します。 環境変数情報名には、実行時にファイルのパス名を設定します。

SSIN(SYSIN):

データの入力先としてコンソールウィンドウを使用します。

- 環境変数情報名は英大文字(A～Z)で始まる 8 文字以内の英大文字および数字でなければなりません。 また、環境変数情報名は、他のファイルで使用する環境変数情報名(ファイル識別名)と一致しないようにする必要があります。

SSOUT (DISPLAY 文のデータの出力先)

小入出力機能の DISPLAY 文のデータの出力先を指定します。

```
        ¡環境変数情報名ü
SSOUT(í                 ý)
        îSYSOUT         þ
```

SSOUT(環境変数情報名):

データの出力先としてファイルを使用します。 環境変数情報名には、実行時にファイルのパス名を設定します。 出力するファイルのコード系を指定するには、実行環境変数 @CBR_DISPLAY_SYSOUT_CODE を指定してください。

SSOUT(SYSOUT):

データの出力先としてコンソールウィンドウを使用します。

環境変数情報名は英大文字(A～Z)で始まる 8 文字以内の英大文字または数字でなければなりません。 また、環境変数情報名は、他のファイルで使用する環境変数情報名(ファイル識別名)と一致しないようにする必要があります。

[参考] 実行環境変数 -@CBR_DISPLAY_SYSOUT_CODE (DISPLAY 文の出力ファイルのコード系の指定)

NCOBU00626

TAB (タブの扱い)

タブの扱いを 4 カラム単位にする(TAB(4))か 8 カラム単位にする(TAB(8))かを指定します。 ただし、値としてのタブは、タブ値そのものです。

```
      ì 8ü
TAB(í  ý)
      î 4þ
```

このオプションはソースプログラムファイルに翻訳単位が 1 つだけの場合、翻訳指示文(@OPTIONS)に指定可能です。

TRUNC (桁落とし処理の可否)

2 進項目を受取り側項目とする数字転記で、上位桁の桁落としに関する処理方法を指定します。

```
¡TRUNC    ü
í         ý
îNOTRUNCþ
```

TRUNC:

結果の値が受取り側項目の PICTURE 句の記述に従って、上位桁が桁落としされ、受取り側項目に格納されます。 なお、送出し側項目の整数部の桁数が、受取り側項目の整数部の桁数よりも大きい場合だけ、上記のような桁落としが行われます。

NOTRUNC:

目的プログラムの実行速度を優先します。 桁落としを行わないほうが実行が速くなる場合には、桁落としは行われません。

PICTURE 句の記述で、

- S999V9（整数部３桁）を S99V99（整数部２桁）に転記: 桁落としあり
- S9V999（整数部１桁）を S99V99（整数部２桁）に転記: 桁落としなし

- TRUNC を指定したとき、送出し側項目の整数領域に入る値は、ハードウェアに依存します。
- NOTRUNC を指定する場合には、桁落としが行われなくても、受取り側項目に PICTURE 句に記述した桁を超える値が格納されないように、プログラムを設計しなければなりません。
- NOTRUNC で桁落としを行うか行わないかの基準は、コンパイラによって異なります。 したがって、NOTRUNC によって桁落としが行われないことを利用したプログラムは他システムへの互換が保証されないので注意してください。

NCOBU01199

USEEXTRET (USE 手続き終了後の動作指定)

例外オブジェクトに対する USE 手続き終了後、USE への移行原因の文の直後に復帰し処理を継続する（USEEXTRET）か、 暗の EXIT METHOD(または、EXIT PROGRAM)を実行する（NOUSEEXTRET）かを選択します。

```
¡USEEXTRET    ü
í             ý
îNOUSEEXTRETþ
```

COBOL 標準の例外処理の概要

XREF (相互参照リストの出力の可否)

相互参照リストを出力する(XREF)か、しない(NOXREF)かを指定します。

```
ì XREF    ü
í         ý
î NOXREFþ
```

相互参照リストには、利用者語や特殊レジスタなどが文字の大小順序の昇順に表示されます。

XREF 指定された場合、コンパイラオプション/print で指定されたファイルに出力されます。

- コンパイラオプション/print が指定されていない場合、本オプションを指定しても、相互参照リストは出力されません。
- 翻訳の結果、最大重大度コードが S レベル以上の場合、相互参照リストの出力は抑止されます。

NCOBU00629

ZWB (符号付き外部 10 進項目と英数字項目の比較)

符号付き外部 10 進項目を英数字フィールドと比較するときに、外部 10 進項目の符号部を無視して比較する(ZWB)か、符号部を含めて比較する(NOZWB)かを指定します。 ここで、英数字とは、英数字項目、英字項目、英数字編集項目、数字編集項目、文字定数および ZERO 以外の表意定数のことです。

```
ì ZWB    ü
í        ý
î NOZWBþ
```

例

- 77 ED PIC S9(3) VALUE +123.
- 77 AN PIC X(3) VALUE "123".

上記で、条件式 ED = AN の真偽は、以下のようになります。

- ZWB を指定した場合 … 真
- NOZWB を指定した場合 … 偽

翻訳オプションの指定方法と優先順位

翻訳オプションの指定方法には、以下の 2 種類があります。

1. /wc オプションによる指定
2. ソースプログラム中の翻訳指示文(@OPTIONS)による指定

オプションの優先順位は、2.>1.となります。

NCOBU00632

翻訳オプション ALPHAL に関する注意事項

翻訳オプションで、最も重要なものの 1 つに、ALPHAL があります。 COBOL 言語自体は、構文上、パラメタ名の使用に、大文字と小文字の区別をしません。 また他のプログラム名、クラス名、およびメソッド名も、大文字と小文字の区別がありません。 ほとんどの COBOL プログラムは、COBOL プログラム名、クラス名、およびメソッド名など、すべて大文字に変換し使用されるため、大文字と小文字の区別を意識する必要がありません。

しかし、.NET サービスは、大文字と小文字の区別が行われるため、NetCOBOL for .NET コンパイラは、ALPHAL(AUTO)を省略値として設定します。 これにより、COBOL の語は、英大文字と英小文字が適切な表現に自動判定されます。 また、明に ALPHAL(WORD)を指定した場合、プログラム名、クラス名、およびメソッド名の大文字への変換は行われません。 正常に動作させるためには、.NET Framework 内で使用する名前について、大文字と小文字を区別するルールに従ったプログラム名、クラス名、およびメソッド名をプログラミングすることを覚えておく必要があります。

.NET Framework 外部にある、COBOL プログラムの起動が必要な場合、プログラム名を大文字にコード化する必要があります。

実行環境変数

ここでは、実行環境変数について説明します。

このトピックの内容

実行環境変数を設定する場合、指定するフォルダ名またはファイル名にコンマ（,）が含まれている場合、そのフォルダ名またはファイル名は、二重引用符（"）内に指定されている必要があります。

NCOBU00634

実行環境変数 - トピック順一覧

実行時オプションに関するもの	実行環境変数
実行時オプションの指定	@GOPT
小入出力に関するもの	実行環境変数
Micro Focus COBOL 仕様による ACCEPT 文動作の指定	@CBR_ACCEPT_COMPATIBILITY
コンソールウィンドウの種別の指定	@CBR_CONSOLE
DISPLAY UPON CONSOLE のイベントログ出力時のイベント種類指定	@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL
DISPLAY UPON CONSOLE のイベントログ出力指定	@CBR_DISPLAY_CONSOLE_OUTPUT
DISPLAY UPON SYSERR イベントログ出力時のイベント種類指定	@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL
DISPLAY UPON SYSERR のイベントログ出力指定	@CBR_DISPLAY_SYSERR_OUTPUT
DISPLAY UPON SYSOUT イベントログ出力時のイベント種類指定	@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL
DISPLAY UPON SYSOUT のイベントログ出力指定	@CBR_DISPLAY_SYSOUT_OUTPUT
任意の日付を取得	@CBR_JOBDATE
メッセージに関するもの	実行環境変数
実行時メッセージに例外情報を追加する指定	@CBR_APPEND_EXCEPTION_MESSAGE
実行時メッセージの出力先の指定	@CBR_MESSAGE
実行時メッセージの重大度を指定	@CBR_MESS_LEVEL_CONSOLE
実行時メッセージの重大度を指定	@CBR_MESS_LEVEL_EVENTLOG
U レベルエラーの出力有無の指定	@CBR_OUTPUT_UNRECOVERABLE_MESSAGE
SYSERR 出力情報の拡張の指定	@CBR_SYSERR_EXTEND
メッセージを出力するファイルの指定	@MessOutFile

メッセージを出力するファイルのコード系の指定	@MessOutFile_CODE
実行時メッセージの抑止指定	@NoMessage
ファイルに関するもの/SQL に関するもの	実行環境変数
ファイルの排他処理の指定	@AllFileExclusive
実行用の初期化ファイルの指定	@CBR_CBRFILE
エントリ情報ファイルの指定	@CBR_ENTRYFILE
ファイルのコード系の指定	@CBR_FILE_CODE_SET
COBOL ファイルのファイルサイズの指定	@CBR_FILE_LFS_ACCESS
外部ファイルハンドラで結合するファイルシステムの入り口名の指定	@CBR_EXFH_API
外部ファイルハンドラで結合するファイルシステムの DLL 名の指定	@CBR_EXFH_LOAD
ファイルの高速処理の一括指定	@CBR_FILE_SEQUENTIAL_ACCESS
入出力エラーの実行時メッセージの出力	@CBR_FILE_USE_MESSAGE
SQL Server から送信されたユーザ定義メッセージのトレース出力の指定	@CBR_SQL_MESSAGE_TRACE
行順ファイルのレコード内後置空白を取り除くまたは有効にする指定	@CBR_TRAILING_BLANK_RECORD
ODBC 情報ファイルの指定	@ODBC_Inf
データプロバイダー拡張ライブラリの格納フォルダ指定	@CBR_DATAPROVIDER_EXTENSION_FOLDER
プログラムで使用するファイルの指定	ファイル識別名
小入出力機能の入力ファイルの指定	SYSIN のアクセス名
小入出力機能の出力ファイルの指定	SYSOUT のアクセス名
プリンタに関するもの	実行環境変数
I 制御レコードによる文書名の指定	@CBR_DocumentName_xxxx
I 制御レコードのとじしろ方向、とじしろ幅および印刷原点位置指定をフォームオーバレイに対して有効または無効にする指定	@CBR_OverlayPrintOffset
ANK 文字サイズの指定	@CBR_PrinterANK_Size

印刷ファイルで使用するフォントテーブルの指定	@CBR_PrintFontTable
ASSIGN 句に PRINTER を指定したファイルに対して有効な印刷情報ファイルの指定	@CBR_PrintInfoFile
文字配置座標の計算方法の指定	@CBR_PrintTextPosition
文字行内配置時の上端/下端合わせの指定	@CBR_TextAlign
デフォルト FCB 名の指定	@DefaultFCB_Name
印刷ファイルで使用するフォントの指定	@PrinterFontName
用紙名の指定	@PRN_FormName_xxx
表示ファイルから使用する情報ファイルの指定	ファイル識別名
プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定	ファイル識別名
プログラムで使用するプリンタおよび各種パラメタの指定	ファイル識別名
FCB 制御文の指定	FCBxxxx
フォームオーバレイパターンのフォルダの指定	FOVLDIR
フォームオーバレイパターンのファイル名の形式の指定	FOVLTYPE
フォームオーバレイパターンのファイル名の指定	FOVLNAME
フォームオーバレイパターンのファイルの拡張子の指定	OVD_SUFFIX
CSV 形式データの操作に関するもの	実行環境変数
CSV 形式データ操作時のメッセージ抑止指定	@CBR_CSV_OVERFLOW_MESSAGE
生成する CSV 形式のバリエーション	@CBR_CSV_TYPE
その他	実行環境変数
ファイル名の空白処理	@CBR_AC_FILENAME_SPACES
簡略化した動作状態の出力	@CBR_CBRINFO
外部属性の RCS オプションチェックの指定	@CBR_EXTERNAL_RCS_OPTION
NATIONAL 関数の変換モードの指定	@CBR_FUNCTION_NATIONAL
スレッド同期制御サブルーチンの待ち時間の指定	@CBR_THREAD_TIMEOUT

DECIMAL オプションチェックの指定	@CBR_DECIMAL_OPTION

NCOBU00635

実行環境変数 - 詳細

環境変数の詳細について説明します。

実行環境変数	用途
@AllFileExclusive	ファイルの排他処理の指定
@CBR_ACCEPT_COMPATIBILITY	Micro Focus COBOL 仕様による ACCEPT 文動作の指定
@CBR_AC_FILENAME_SPACES	ファイル名の空白処理
@CBR_APPEND_EXCEPTION_MESSAGE	実行時メッセージに例外情報を追加する指定
@CBR_CBRFILE	実行用の初期化ファイルの指定
@CBR_CBRINFO	簡略化した動作状態の出力
@CBR_CONSOLE	コンソールウィンドウの種別の指定
@CBR_CSV_OVERFLOW_MESSAGE	CSV 形式データ操作時のメッセージ抑止指定
@CBR_CSV_TYPE	生成する CSV 形式のバリエーション
@CBR_DECIMAL_OPTION	DECIMAL オプションチェックの指定
@CBR_DATAPROVIDER_EXTENSION_FOLDER	データプロバイダー拡張ライブラリの格納フォルダ指定
@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL	DISPLAY UPON CONSOLE のイベントログ出力時のイベント種類指定
@CBR_DISPLAY_CONSOLE_OUTPUT	DISPLAY UPON CONSOLE のイベントログ出力指定
@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL	DISPLAY UPON SYSERR のイベントログ出力時のイベント種類指定
@CBR_DISPLAY_SYSERR_OUTPUT	DISPLAY UPON SYSERR のイベントログ出力指定
@CBR_DISPLAY_SYSOUT_CODE	DISPLAY 文の出力ファイルのコード系の指定
@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL	DISPLAY UPON SYSOUT のイベントログ出力時のイベント種類指定
@CBR_DISPLAY_SYSOUT_OUTPUT	DISPLAY UPON SYSOUT のイベントログ出力指定

@CBR_DocumentName_xxxx	I 制御レコードによる文書名の指定
@CBR_ENTRYFILE	エントリ情報ファイルの指定
@CBR_EXFH_API	外部ファイルハンドラで結合するファイルシステムの入り口名の指定
@CBR_EXFH_LOAD	外部ファイルハンドラで結合するファイルシステムの DLL 名の指定
@CBR_EXTERNAL_RCS_OPTION	外部属性の RCS オプションチェックの指定
@CBR_FILE_CODE_SET	ファイルのコード系の指定
@CBR_FILE_LFS_ACCESS	COBOL ファイルのファイルサイズを拡張する指定
@CBR_FILE_SEQUENTIAL_ACCESS	ファイルの高速処理の一括指定
@CBR_FILE_USE_MESSAGE	入出力エラーの実行時メッセージの出力
@CBR_FUNCTION_NATIONAL	NATIONAL 関数の変換モードの指定
@CBR_JOBDATE	任意の日付を取得
@CBR_MESSAGE	実行時メッセージの出力先の指定
@CBR_MESS_LEVEL_CONSOLE	実行時メッセージの重大度を指定
@CBR_MESS_LEVEL_EVENTLOG	実行時メッセージの重大度を指定
@CBR_OUTPUT_UNRECOVERABLE_MESSAGE	U レベルエラーの出力有無の指定
@CBR_OverlayPrintOffset	I 制御レコードのとじしろ方向、とじしろ幅および印刷原点位置指定をフォームオーバレイに対して有効または無効にする指定
@CBR_PrintFontTable	印刷ファイルで使用するフォントテーブルの指定
@CBR_PrintInfoFile	ASSIGN 句に PRINTER を指定したファイルに対して有効な印刷情報ファイルの指定
@CBR_PrintTextPosition	文字配置座標の計算方法の指定
@CBR_PrinterANK_Size	ANK 文字サイズの指定
@CBR_SQL_MESSAGE_TRACE	SQL Server から送信されたユーザ定義メッセージのトレース出力の指定
@CBR_SYSERR_EXTEND	SYSERR 出力情報の拡張の指定
@CBR_TRAILING_BLANK_RECORD	行順ファイルのレコード内後置空白を取り除くま

	たは有効にする指定
@CBR_TextAlign	文字行内配置時の上端/下端合わせの指定
@DefaultFCB_Name	デフォルト FCB 名の指定
FCBxxxx	FCB 制御文の指定
FOVLDIR	フォームオーバレイパターンのフォルダの指定
FOVLNAME	フォームオーバレイパターンのファイル名の指定
FOVLTYPE	フォームオーバレイパターンのファイル名の形式の指定
ファイル識別名	表示ファイルから使用する情報ファイルの指定
ファイル識別名	プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定
ファイル識別名	プログラムで使用するファイルの指定
ファイル識別名	プログラムで使用するプリンタおよび各種パラメタの指定
@GOPT	実行時オプションの指定
@MessOutFile	メッセージを出力するファイルの指定
@MessOutFile_CODE	メッセージを出力するファイルのコード系の指定
@NoMessage	実行時メッセージの抑止指定
@ODBC_Inf	ODBC 情報ファイルの指定
OVD_SUFFIX	フォームオーバレイパターンのファイルの拡張子の指定
@PRN_FormName_xxx	用紙名の指定
@PrinterFontName	印刷ファイルで使用するフォントの指定
SYSIN のアクセス名	小入出力機能の入力ファイルの指定
SYSOUT のアクセス名	小入出力機能の出力ファイルの指定

@AllFileExclusive (ファイルの排他処理の指定)

ファイルの排他処理の指定

```
                    ì YESü
@AllFileExclusive=í      ý
                    î NO  þ
```

プログラムを実行するときに、COBOL ソースプログラムに記述したファイルの排他に関する処理を無視して、無条件にすべてのファイルを排他処理する(YES)か、COBOL ソースプログラムに記述したファイルの排他に関する処理どおりに実行する(NO)かを指定します。排他制御を指定した場合、プログラムで使用しているファイルは、他のプログラムからはアクセス不可能(オープンエラー)となります。 本指定を省略した場合、"NO"が有効となります。

ファイルを排他処理することにより、ファイルアクセス時間が短縮される場合があります。

NCOBU01194

@CBR_ACCEPT_COMPATIBILITY（Micro Focus COBOL 仕様による ACCEPT 文動作の指定）

Micro Focus COBOL 仕様による ACCEPT 文動作の指定

```
@CBR_ACCEPT_COMPATIBILITY=MF
```

ACCEPT 文に FROM 指定がない、または機能名 SYSIN に対応付けた呼び名を指定した場合、 実行キーまたはリターンキーが入力されるまで入力要求が行われます。

@CBR_AC_FILENAME_SPACES（ファイル名の空白処理）

ファイル名の空白処理

```
@CBR_AC_FILENAME_SPACES=NO
```

AcuCOBOL サブルーチンでファイル処理を行う際、処理対象のファイル名に空白が含まれる場合の動作を指定します。 "NO"を指定した場合、ファイル名に空白を含めることはできません。 ファイル名に指定した文字列のうち、空白以降の文字は無視されます。

本環境変数に"0", "OFF", "FALSE" が指定された場合、"NO"が指定されたものとして動作します。

本環境変数の指定を省略した場合、ファイル名に空白を含めることができます。

[参照]NetCOBOL for .NET AcuCOBOL サブルーチン ユーザーズガイド

NCOBU00637

@CBR_APPEND_EXCEPTION_MESSAGE (実行時メッセージに例外情報を追加する指定)

実行時メッセージに例外情報を追加する指定

```
                                    ì YESü
@CBR_APPEND_EXCEPTION_MESSAGE=í     ý
                                    î NO þ
```

"YES" を指定した場合に、実行時に例外が発生すると、例外に関する情報が実行時エラーメッセージの終わりに追加されます。 "NO" を指定した場合は、例外情報は追加されません。本指定を省略した場合、例外情報は追加されません。

@CBR_CBRFILE (実行用の初期化ファイルの指定)

実行用の初期化ファイルの指定

```
@CBR_CBRFILE=実行用の初期化ファイル名
```

使用する実行用の初期化ファイル名を指定します。 実行時の初期化ファイルを他で設定する場合は、実行用の初期化ファイルの指定方法を参照して下さい。 実行用の初期化ファイル名には、絶対パスと相対パスを指定できます。 相対パスで指定された場合、ベースディレクトリからの相対パスとなります。 EXE ファイルのベースディレクトリは、EXE ファイルを含むフォルダとなり、Web フォームや Web サービスのベースディレクトリは、仮想ディレクトリとなります。

[参照]実行環境設定ツール

NCOBU00639

@CBR_CBRINFO (簡略化した動作状態の出力)

簡略化した動作状態の出力

```
@CBR_CBRINFO=YES
```

COBOL プログラムの実行時の情報を実行環境の開設時に実行時メッセージ(JMP0070I-I)として出力します。 出力される情報には、ランタイムシステムのバージョンレベル、実行用の初期化ファイル名などがあります。

@CBR_CONSOLE (コンソールウィンドウの種別の指定)

コンソールウィンドウの種別の指定

```
                ì SYSTEMü
@CBR_CONSOLE=í          ý
                î COBOL  þ
```

小入出力機能でコンソールウィンドウを使用する場合、および実行時メッセージの出力先として、システムのコンソールを使用する(SYSTEM)か、COBOL のコンソールウィンドウおよびメッセージボックスを使用する(COBOL)かを指定します。 本指定を省略した場合、"COBOL"が有効となります。 .NET では、小入出力機能のコンソールウィンドウは、システムのコンソールだけなので、実行時メッセージの出力先にだけ有効です。

NCOBU01171

@CBR_CSV_OVERFLOW_MESSAGE（CSV 形式データ操作時のメッセージ抑止指定）

CSV 形式データ操作時のメッセージ抑止指定

```
@CBR_CSV_OVERFLOW_MESSAGE=NO
```

STRING 文(書き方 2)および UNSTRING 文(書き方 2)の実行時に出力される以下のメッセージを抑止します。

- JMP0262I-W
- JMP0263I-W

@CBR_CSV_TYPE（生成する CSV 形式のバリエーション）

生成する CSV 形式のバリエーション

```
                    ì MODE-1ü
                    ï MODE-2ï
@CBR_CSV_TYPE=í MODE-3ý
                    ï       ï
                    î MODE-4þ
```

STRING 文(書き方 2)で生成する CSV 形式のバリエーションを指定します。 この指定は、TYPE 指定を省略した STRING 文だけに有効です。 生成される CSV 形式の詳細は、CSV 形式のバリエーションを参照してください。

NCOBU01182

@CBR_DATAPROVIDER_EXTENSION_FOLDER（データプロバイダー拡張ライブラリのフォルダ指定）

```
@CBR_DATAPROVIDER_EXTENSION_FOLDER=フォルダ
```

データプロバイダー拡張ライブラリの格納先フォルダを指定します。

```
例：@CBR_DATAPROVIDER_EXTENSION_FOLDER=C:¥EXTENSIONS
```

[注意]

- フォルダ指定は、セミコロンによる複数個指定はサポートしていません。
- 絶対パスで指定されていない場合、アプリケーションドメインからの相対パスとして検索されます。
- @SQL_DATAPROVIDER_EXTENSION_DLL が指定されている場合、
  @SQL_DATAPROVIDER_EXTENSION_DLL に指定された値が優先されます。
- @CBR_DATAPROVIDER_EXTENSION_FOLDER に値が指定されていない場合、 “データプロバイダー拡張ライブラリの検索順序”に従い、 次の検索が行われます。
- 指定したフォルダが存在しない、または、検索されたライブラリが “データプロバイダー拡張ライブラリの作成手順”に従って作成されていない場合、 Data Provider Extension 機能が利用できず、以下のエラーになります。

  ```
  JMP0372I-U SQL 文を実行するための環境開設でエラーが発生しました. 'DataProvider Extension[プロバイダ名]'
  ```

- 指定したフォルダに、 データプロバイダー拡張機能で作成したデータプロバイダー拡張ライブラリ以外が含まれている場合、 検索に時間がかかります。 指定するフォルダには、データプロバイダー拡張機能で使用するライブラリのみを格納することをおすすめします。
- 指定したフォルダに、 データプロバイダー拡張機能で作成したデータプロバイダー拡張ライブラリが複数含まれている場合、 ライブラリ名の昇順で最初に検出されたデータプロバイダー拡張ライブラリが使用されます。

@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL(DISPLAY UPON CONSOLE のイベントログ出力時のイベント種類指定)

DISPLAY UPON CONSOLE のイベントログ出力時のイベント種類指定

```
                                                  ì I ü
@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL=í W ý
                                                  î E þ
```

DISPLAY UPON CONSOLE の出力をイベントログに出力するときのイベント種類を指定します。

@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL に指定できるパラメタは以下のとおりです。

パラメタ	意味
I	情報イベント。イベント種類には"情報"と表示されます。
W	警告イベント。イベント種類には"警告"と表示されます。
E	エラーイベント。イベント種類には"エラー"と表示されます。

例：イベント種類を"警告"としてイベントログに出力する

```
@CBR_DISPLAY_CONSOLE_EVENTLOG_LEVEL=W
```

NCOBU01079

@CBR_DISPLAY_CONSOLE_OUTPUT (DISPLAY UPON CONSOLE のイベントログ出力指定)

DISPLAY UPON CONSOLE のイベントログ出力指定

```
@CBR_DISPLAY_CONSOLE_OUTPUT=EVENTLOG[(コンピュータ名)]
```

DISPLAY UPON CONSOLE の出力先をイベントログにします。

コンピュータ名には、イベントログの出力先コンピュータ名を指定します。ネットワーク上の他のコンピュータ名を指定することもできます。コンピュータ名を省略した場合は、プログラムを実行しているコンピュータのイベントログに出力します。 また、指定したコンピュータのイベントログに出力対象メッセージを出力できない以下のような場合、実行中のコンピュータのイベントログに、エラーメッセージと出力対象メッセージを出力します。

- ネットワーク上に存在しないコンピュータ名を指定した場合
- 指定したコンピュータで Windows が動作していない場合

- リモートコンピュータに.NET Framework がインストールされていない場合、期待どおりに動作しません。
- 出力先の Windows システム上でイベントログへの書き込み権限の設定(ユーザアカウント、ファイアウォールなど)が必要な場合があります。設定方法の詳細については、Microsoft のドキュメントを参照してください。

@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL(DISPLAY UPON SYSERR のイベントログ出力時のイベント種類指定)

DISPLAY UPON SYSERR のイベントログ出力時のイベント種類指定

```
                                                  ì I ü
@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL=í W ý
                                                  î E þ
```

DISPLAY UPON SYSERR の出力をイベントログに出力するときのイベント種類を指定します。

@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL に指定できるパラメタは以下のとおりです。

パラメタ	意味
I	情報イベント。イベント種類には"情報"と表示されます。
W	警告イベント。イベント種類には"警告"と表示されます。
E	エラーイベント。イベント種類には"エラー"と表示されます。

例：イベント種類を"警告"としてイベントログに出力する

```
@CBR_DISPLAY_SYSERR_EVENTLOG_LEVEL=W
```

NCOBU01081

@CBR_DISPLAY_SYSERR_OUTPUT (DISPLAY UPON SYSERR のイベントログ出力指定)

DISPLAY UPON SYSERR のイベントログ出力指定

```
@CBR_DISPLAY_SYSERR_OUTPUT=EVENTLOG[(コンピュータ名)]
```

DISPLAY UPON SYSERR の出力先をイベントログにします。

コンピュータ名には、イベントログの出力先コンピュータ名を指定します。ネットワーク上の他のコンピュータ名を指定することもできます。コンピュータ名を省略した場合は、プログラムを実行しているコンピュータのイベントログに出力します。 また、指定したコンピュータのイベントログに出力対象メッセージを出力できない以下のような場合、実行中のコンピュータのイベントログに、エラーメッセージと出力対象メッセージを出力します。

- ネットワーク上に存在しないコンピュータ名を指定した場合
- 指定したコンピュータで Windows が動作していない場合

- リモートコンピュータに.NET Framework がインストールされていない場合、期待どおりに動作しません。
- 出力先の Windows システム上でイベントログへの書き込み権限の設定(ユーザアカウント、ファイアウォールなど)が必要な場合があります。設定方法の詳細については、Microsoft のドキュメントを参照してください。

@CBR_DISPLAY_SYSOUT_CODE (DISPLAY 文の出力ファイルのコード系の指定)

```
                                ¡ACP       ü
@CBR_DISPLAY_SYSOUT_CODE=í                 ý
                                îUNICODEþ
```

DISPLAY 文を実行したときのデータの出力先となるファイルのコード系を指定します。

出力ファイルのコード系を ASCII/シフト JIS として扱う(ACP)か、 Unicode として扱う(UNICODE)かを指定します。 本指定を省略した場合は、"UNICODE"が有効になります。

本指定は、翻訳オプション SSOUT に環境変数情報名が指定された場合だけ有効になります。

[参照]

- 翻訳オプション -SSOUT (DISPLAY 文のデータの出力先)
- 実行環境変数 -SYSOUT のアクセス名 (小入出力機能の出力ファイルの指定)

NCOBU01084

@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL(DISPLAY UPON SYSOUT のイベントログ出力時のイベント種類指定)

DISPLAY UPON SYSOUT のイベントログ出力時のイベント種類指定

```
                                                  ì I ü
@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL=í W ý
                                                  î E þ
```

DISPLAY UPON SYSOUT の出力をイベントログに出力するときのイベント種類を指定します。

@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL に指定できるパラメタは以下のとおりです。

パラメタ	意味
I	情報イベント。イベント種類には"情報"と表示されます。
W	警告イベント。イベント種類には"警告"と表示されます。
E	エラーイベント。イベント種類には"エラー"と表示されます。

例：イベント種類を"警告"としてイベントログに出力する

```
@CBR_DISPLAY_SYSOUT_EVENTLOG_LEVEL=W
```

@CBR_DISPLAY_SYSOUT_OUTPUT (DISPLAY UPON SYSOUT のイベントログ出力指定)

DISPLAY UPON SYSOUT のイベントログ出力指定

```
@CBR_DISPLAY_SYSOUT_OUTPUT=EVENTLOG[(コンピュータ名)]
```

DISPLAY UPON SYSOUT の出力先をイベントログにします。

コンピュータ名には、イベントログの出力先コンピュータ名を指定します。ネットワーク上の他のコンピュータ名を指定することもできます。コンピュータ名を省略した場合は、プログラムを実行しているコンピュータのイベントログに出力します。 また、指定したコンピュータのイベントログに出力対象メッセージを出力できない以下のような場合、実行中のコンピュータのイベントログに、エラーメッセージと出力対象メッセージを出力します。

- ネットワーク上に存在しないコンピュータ名を指定した場合
- 指定したコンピュータで Windows が動作していない場合

- リモートコンピュータに.NET Framework がインストールされていない場合、期待どおりに動作しません。
- 出力先の Windows システム上でイベントログへの書き込み権限の設定(ユーザアカウント、ファイアウォールなど)が必要な場合があります。設定方法の詳細については、Microsoft のドキュメントを参照してください。

NCOBU00886

@CBR_DECIMAL_OPTION（DECIMAL オプションチェックの指定）

DECIMAL オプションチェックの指定

```
                          ì CHECK    ü
@CBR_DECIMAL_OPTION=í                ý
                          î NOCHECKþ
```

プログラムが呼び出されたとき、DECIMAL オプションの指定をチェックする(CHECK)か、チェックしない(NOCHECK)かを指定します。本指定を省略した場合、"CHECK"が有効となります。

"CHECK"を指定した場合、呼び出されたプログラムの DECIMAL オプションのチェックを行います。呼び出されたプログラムの DECIMAL オプションの指定が異なっている場合､実行時メッセージを出力し異常終了します。

"NOCHECK"を指定した場合、呼び出されたプログラムの DECIMAL オプションのチェックは行いません。

[参照] "DECIMAL (SEPARATE 指定なしの外部 10 進項目の内部表現形式の指定)"

@CBR_DocumentName_xxxx (I 制御レコードによる文書名の指定)

I 制御レコードによる文書名の指定

```
@CBR_DocumentName_xxxx=文書名
```

FORMAT 句なし印刷ファイルで、I 制御レコードの DOC-INFO(文書名識別情報)フィールドに指定した任意の 4 文字以内の文字列(英数字)を本環境変数情報名の xxxx 部分に置き換え、本環境変数情報名に対して文書名を対応付けて指定します。 このとき、指定可能な文書名は、128 バイト以内の英字/数字/カナ/日本語の組合せでなければなりません。 [参照] "I 制御レコード"/"S 制御レコード"

なお、ここで指定した文書名は、Windows システムが提供するプリントマネージャーまたはプリンタの画面に表示されます。

例

```
〔I 制御レコードの DOC-INFO フィールドの指定〕
"ABCD"
〔xxxx 部分置換後の環変数情報名〕
@CBR_DocumentName_ABCD
〔環境変数情報名と文書名の対応付け〕
@CBR_DocumentName_ABCD=富士通パソコン売上伝票
```

NCOBU00643

@CBR_ENTRYFILE (エントリ情報ファイルの指定)

エントリ情報ファイルの指定

```
@CBR_ENTRYFILE=エントリ情報ファイル名
```

エントリ情報を指定したい場合、エントリ情報ファイルを作成し、本環境変数情報に指定します。

エントリ情報ファイル名には、絶対パスと相対パスを指定できます。 相対パスで指定された場合、ベースディレクトリからの相対パスとなります。 EXE ファイルのベースディレクトリは、EXE ファイルを含むフォルダとなり、Web フォームや Web サービスのベースディレクトリは、仮想ディレクトリとなります。

エントリ情報の詳細についてはエントリ情報ファイルを参照してください。

@CBR_EXTERNAL_RCS_OPTION(外部属性の RCS オプションチェックの指定)

外部属性の RCS オプションチェックの指定

```
                                ¡CHECK    ü
@CBR_EXTERNAL_RCS_OPTION=í              ý
                                îNOCHECKþ
```

外部データおよび外部ファイルの RCS オプションの違いをチェックする(CHECK)か、チェックしない(NOCHECK)かを指定します。 本指定を省略した場合、"CHECK"が有効となります。

"CHECK"を指定した場合は、外部データおよび外部ファイルの RCS オプションのチェックを行ないます。 RCS オプションが異なっているとき、実行時メッセージを出力し異常終了します。

"NOCHECK"を指定した場合、外部データおよび外部ファイルの RCS オプションのチェックを行ないません。

[参照] "RCS オプション"

NCOBU01059

@CBR_EXFH_API (外部ファイルハンドラで結合するファイルシステムの入口名の指定)

外部ファイルハンドラで結合するファイルシステムの入口名の指定

```
@CBR_EXFH_API=入口名
```

外部ファイルハンドラを使用する時、結合するファイルシステムの入口名を指定します(必須)。

入口名を指定する時、大文字と小文字を区別して記述してください。

[参照]外部ファイルハンドラ

@CBR_EXFH_LOAD (外部ファイルハンドラで結合するファイルシステムの DLL 名の指定)

外部ファイルハンドラで結合するファイルシステムの DLL 名の指定

```
@CBR_EXFH_LOAD=DLL 名
```

外部ファイルハンドラを使用する時、結合するファイルシステムの DLL 名を指定します。 ファイルのパスには、絶対パス、相対パスのどちらも指定することができます。相対パスを用いた場合、カレントディレクトリからの相対パスとなります。

[参照]外部ファイルハンドラ

NCOBU00645

@CBR_FILE_CODE_SET (ファイルのコード系の指定)

ファイルのコード系の指定

```
@CBR_FILE_CODE_SET=ASCII
```

ファイルのコード系を指定します。 "ASCII" を指定した場合、行順ファイルおよび索引キーに日本語項目を指定した索引ファイルのコード系を ASCII/シフト JIS として扱います。 本指定を省略した場合は、Unicode として扱います。

注意

本指定は、レコードデータのコード変換は行われません。 コード変換は "UNICODE-OF" 関数および、"ACP-OF" 関数を利用してください。

@CBR_FILE_LFS_ACCESS (COBOL ファイルのサイズを拡張する指定)

COBOL ファイルのファイルサイズを拡張する指定

```
@CBR_FILE_LFS_ACCESS=YES
```

COBOL ファイルのファイルサイズをシステム制限まで拡張することができます。

ファイルサイズについては、他のファイルシステムの使用法を参照してください。

- 以下の環境下において、本環境変数を指定する場合、全ての処理で有効となるようにしてください。
  - 他の COBOL プログラムとファイルを共有している
  - 対象となる COBOL ファイルをツールおよび他製品からアクセスしている

  本環境変数が有効でない場合、COBOL ファイルの排他制御の動作は保証しません。
- 相対ファイルには本指定は有効となりません。指定した場合、ファイルサイズは拡張されません。
- 本環境変数は、アプリケーション構成ファイルに指定しても、有効になりません。以下の方法で設定する必要があります。
  - SET コマンドでの設定
  - OS のユーザ環境変数に設定

NCOBU01086

@CBR_FILE_SEQUENTIAL_ACCESS (ファイルの高速処理の一括指定)

ファイルの高速処理の一括指定

```
@CBR_FILE_SEQUENTIAL_ACCESS=BSAM
```

ファイルの高速処理を有効とする場合、BSAM を指定します。

本環境変数の使用方法については、ファイルの高速処理の一括の指定方法を参照してください。

@CBR_FILE_USE_MESSAGE (入出力エラーの実行時メッセージの出力)

入出力エラーの実行時メッセージの出力

```
@CBR_FILE_USE_MESSAGE=YES
```

有効な誤り処理手続きがある場合に、入出力エラーの実行時メッセージを出力します。

NCOBU00646

@CBR_FUNCTION_NATIONAL (NATIONAL 関数の変換モードの指定)

NATIONAL 関数の変換モードの指定

```
                                  ì MODE1ü   ì MODE3ü
@CBR_FUNCTION_NATIONAL=[  í          ý][,í          ý]
                                  î MODE2þ   î MODE4þ
```

NATIONAL 関数の変換モードを指定します。

NATIONAL 関数の変換モードには、第 1 オペランドに従来どおりの変換を行う(MODE1)か、視覚的により近い変換を行う(MODE2)かを指定します。 本指定を省略した場合、“MODE1”が指定されたものとみなします。

翻訳オプション RCS(UTF8-UCS2)または RCS(SJIS-UCS2)が有効な場合、第 2 オペランドに UNIX 系システムに近い変換を行う(MODE3)か、Windows 系システムに近い変換を行う(MODE4)かを指定します。 本指定を省略した場合、“MODE4”が指定されたものとみなします。

本指定に誤りがある場合、省略したものとみなします。翻訳オプションが RCS(SJIS)の場合に“MODE3”および“MODE4”を指定した場合も誤りとみなします。“MODE3”および“MODE4”を省略した場合の動作はプラットフォームにより異なります。

MODE1 と MODE2 の違いを以下に示します。

- 翻訳オプション RCS(SJIS)が有効な場合

指定	変換方法
MODE1	(0xB0) を(0x815C)に変換します。 (0x60)を(0x8165)に変換します。
MODE2	(0xB0) を(0x815B)に変換します。 (0x60)を(0x814D)に変換します。

- 翻訳オプション RCS(UTF8-UCS2)または RCS(SJIS-UCS2)が有効な場合

指定	変換方法
MODE1	(0xB0) を(0x1520)に変換します。 (0x60)を(0x1820)に変換します。
MODE2	(0xB0) を(0xFC30)に変換します。 (0x60)を(0x40FF)に変換します。

MODE3 と MODE4 の違いを以下に示します。

- 翻訳オプション RCS(UTF8-UCS2)または RCS(SJIS-UCS2)が有効な場合

指定	変換方法
MODE3	(0x2D) を(0x1222)に変換します。 (0x7E)を(0x1C30)に変換します。
MODE4	(0x2D) を(0x0DFF)に変換します。 (0x7E)を(0x5EFF)に変換します。

NCOBU00887

@CBR_JOBDATE（任意の日付を取得）

任意の日付を取得

```
@CBR_JOBDATE=年.月.日
```

ACCEPT 文（FROM 指定に DATE を指定）または組み込み関数 CURRENT-DATE で、任意の日付を取得する場合に指定します。年.月.日は以下の形式で指定します。

- 年：(00-99)または(1900-2099)
- 月：(01-12)
- 日：(01-31)

“年”の値は、西暦 1900 年代ならば、西暦年の下 2 けた、または 4 けたの西暦年です。西暦 2000 年代ならば、4 けたの西暦年です。

@CBR_MESSAGE (実行時メッセージの出力先の指定)

実行時メッセージの出力先の指定

```
                    ì CBLERROUT                ü
@CBR_MESSAGE=í EVENTLOG[(コンピュータ名)]ý
                    î ALL[(コンピュータ名)]    þ
```

メッセージの出力先を指定します。

CBLERROUT:

メッセージボックス、システムのコンソール(コマンドプロンプト)またはファイルにメッセージを出力します。

@MessOutFile を指定した場合は、ファイルに出力します。@CBR_CONSOLE=SYSTEM を指定した場合は、システムのコンソールに出力します。@MessOutFile と @CBR_CONSOLE を同時に指定した場合、@MessOutFile が有効になり、@CBR_CONSOLE は無効になります。

EVENTLOG:

イベントログにメッセージを出力します。

ALL:

"CBLERROUT"の出力先とイベントログの両方にメッセージを出力します。

省略された場合、"CBLERROUT"の出力先にメッセージを出力します。

コンピュータ名には、イベントログの出力先コンピュータ名を指定します。ネットワーク上の他のコンピュータ名を指定することもできます。コンピュータ名を省略した場合は、プログラムを実行しているコンピュータに出力します。 また、指定したコンピュータのイベントログに出力対象メッセージを出力できない以下のような場合、実行中のコンピュータのイベントログに、エラーメッセージと出力対象メッセージを出力します。

- ネットワーク上に実在しないコンピュータ名を指定した場合
- 指定したコンピュータで Windows が動作していない場合

注意

- リモートコンピュータに.NET Framework がインストールされていない場合、期待どおりに動作しません。
- 出力先の Windows システム上でイベントログへの書き込み権限の設定(ユーザアカウント、ファイアウォールなど)が必要な場合があります。設定方法の詳細については、Microsoft のドキュメントを参照してください。

NCOBU00890

@CBR_MESS_LEVEL_CONSOLE(実行時メッセージの重大度指定)

実行時メッセージの重大度指定

```
                                  ì   NO   ü
                                  ï   I    ï
@CBR_MESS_LEVEL_CONSOLE=          í   W    ý
                                  ï   E    ï
                                  î   U    þ
```

ランタイムシステムが出力する実行時メッセージの出力を抑止したり、実行時メッセージを出力する重大度コードを指定します。 @CBR_MESSAGE の宛先 CBLERROUT が有効でない場合、本環境変数の値は意味を持ちません。

@CBR_MESS_LEVEL_CONSOLE に指定できるパラメタは以下の通りです。

NO :

実行時メッセージを出力しません。

I :

重大度コードが I 以上のメッセージを出力します。

W :

重大度コードがW以上のメッセージを出力します。

E :

重大度コードがE以上のメッセージを出力します。

U :

重大度コードがUのメッセージを出力します。

例

[重大度コードが I 以上の実行時メッセージを出力する]

```
@CBR_MESS_LEVEL_CONSOLE=I
```

- @CBR_MESS_LEVEL_CONSOLE の指定がない場合、または上記以外のパラメタが右辺に指定された場合、重大度コードが I 以上のメッセージが出力されます。（I 指定と同意）
- @CBR_MESS_LEVEL_CONSOLE の指定は、環境変数情報@MessOutFile で指定したファイルに出力される実行時メッセージにも有効となります。
- @CBR_MESS_LEVEL_CONSOLE に NO を指定した場合、オペレータの応答が不要な実行時メッセージはすべて出力されません。実行時エラーの原因特定が困難になりますので、目的を十分理解した上で指定してください。
- @CBR_MESS_LEVEL_CONSOLE に NO を指定した場合でも、オペレータの応答が必要な実行時メッセージは常に出力されます。
- @NoMessage=YES を同時に指定している場合、@CBR_MESS_LEVEL_CONSOLE に指定した重大度が有効となります。

NCOBU00891

@CBR_MESS_LEVEL_EVENTLOG(実行時メッセージの重大度指定)

実行時メッセージの重大度指定

```
                                  NO
                                ì     ü
                                ï I   ï
@CBR_MESS_LEVEL_EVENTLOG=       í W   ý
                                ï E   ï
                                î U   þ
```

ランタイムシステムが出力する実行時メッセージのイベントログへの出力を抑止したり、実行時メッセージを出力する重大度コードを指定します。

@CBR_MESS_LEVEL_EVENTLOG に指定できるパラメタは以下の通りです。

NO :

実行時メッセージを出力しません。

I :

重大度コードが I 以上のメッセージを出力します。

W :

重大度コードがW以上のメッセージを出力します。

E :

重大度コードがE以上のメッセージを出力します。

U :

重大度コードがUのメッセージを出力します。

例

[重大度コードが I 以上の実行時メッセージをイベントログに出力する]

```
@CBR_MESS_LEVEL_EVENTLOG=I
```

- @CBR_MESS_LEVEL_EVENTLOG の指定がない場合、または上記以外のパラメタが右辺に指定された場合、重大度コードが I 以上のメッセージが出力されます。（I 指定と同意）
- @CBR_MESS_LEVEL_EVENTLOG に NO を指定した場合、すべての実行時メッセージが出力されません。実行時エラーの原因特定が困難になりますので、目的を十分理解した上で指定してください。
- @NoMessage=YES を同時に指定している場合、@CBR_MESS_EVENTLOG に指定した重大度が有効となります。

NCOBU00648

@CBR_OUTPUT_UNRECOVERABLE_MESSAGE (U レベルエラーの出力有無の指定)

U レベルエラーの出力有無の指定

```
                                          ì YESü
@CBR_OUTPUT_UNRECOVERABLE_MESSAGE=í       ý
                                          î NO þ
```

"YES" を指定した時に U レベルエラーが発生した場合、システムは、エラーメッセージを出力します。 "NO" を指定した場合は、エラーメッセージを出力しません。 本指定を省略した場合、"NO"が有効となります。

@CBR_OverlayPrintOffset (I 制御レコードのとじしろ方向、とじしろ幅および印刷原点位置指定をフォームオーバレイに対して有効または無効にする指定)

I 制御レコードのとじしろ方向、とじしろ幅および印刷原点位置指定をフォームオーバレイに対して有効または無効にする指定

```
                              ¡VALID   ü
@CBR_OverlayPrintOffset=í             ý
                              îINVALIDþ
```

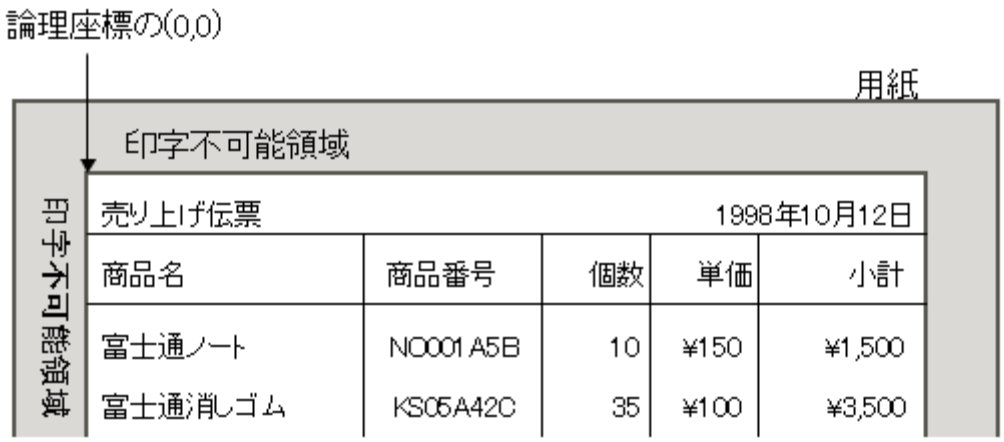
FORMAT 句なし印刷ファイルで、I 制御レコードに指定されたとじしろ方向(BIND)、とじしろ幅(WIDTH)および印刷原点位置(OFFSET)機能をフォームオーバレイに対しても有効にする(VALID)か、無効にする(INVALID)かを指定します。 本指定の省略時は、"INVALID"が指定されたものとみなされます。 なお、本指定の有効範囲は実行単位内です。ファイル単位に有効な指定については、 "印刷情報ファイル" を参照してください。 I 制御レコードについては、 "I 制御レコード/S 制御レコード" を参照してください。

参考

以下に、原点オフセットを設けない通常の印刷結果と、上方向および左方向にそれぞれ 1 インチの原点オフセットを設けた場合の印刷結果を例にとり、VALID/INVALID 指定時(または省略時)の印刷結果の相違について図示します。

※ 以下の図では、罫線=オーバレイ、文字=行レコードを示しています。

[印刷原点位置を指定しない場合の印刷結果]



[X,Y 座標にそれぞれ 1 インチ原点オフセットを設けた場合の印刷結果]

- VALID 指定の場合

・ INVALID 指定の場合

@CBR_PrintFontTable (印刷ファイルで使用するフォントテーブルの指定)

印刷ファイルで使用するフォントテーブルの指定

```
@CBR_PrintFontTable=フォントテーブル名
```

印刷ファイルで使用するフォントテーブルのファイル名を絶対パスで指定します。

ここで指定されたフォントテーブルは、同じ実行環境で動作するすべての印刷ファイルに対して有効になります。ファイルごとのフォントテーブルの指定については、"ファイル識別名 (プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定)"、 "ファイル識別名 (プログラムで使用するプリンタおよび各種パラメタの指定)" を参照してください。

また、フォントテーブルの詳細については、 "印字文字" の "印字文字の書体" および "印字文字のスタイル"、 "フォントテーブル" を参照してください。

NCOBU00652

@CBR_PrintInfoFile (ASSIGN 句に PRINTER を指定したファイルに対して有効な印刷情報ファイルの指定)

ASSIGN 句に PRINTER を指定したファイルに対して有効な印刷情報ファイルの指定

```
@CBR_PrintInfoFile=印刷情報ファイル名
```

FORMAT 句なし印刷情報ファイルで、ASSIGN 句に PRINTER を指定したファイルに対して有効な印刷情報ファイルの名前を絶対パスで指定します。

印刷情報ファイルの詳細については、"印刷情報ファイル"を参照してください。

例

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRINTER
〔実際に使用する印刷情報ファイル名とそのパス〕
 C:¥PRINT.INF
〔環境変数情報〕
 @CBR_PrintInfoFile=C:¥PRINT.INF
```

@CBR_PrintTextPosition (文字配置座標の計算方法の指定)

文字配置座標の計算方法の指定

```
                          ¡TYPE1ü
@CBR_PrintTextPosition=í        ý
                          îTYPE2þ
```

FORMAT 句なし印刷ファイルで、印字する文字を配置する座標(x,y)の計算方法を指定します。なお、本指定の有効範囲は実行単位内です。

文字配置の座標の補正を行わない(TYPE1)か、行う(TYPE2)かを指定します。 本指定を省略した場合、"TYPE2"が有効となります。

TYPE1:

印刷するプリンタ装置の解像度をアプリケーションで指定した行間隔(LPI)や文字間隔(CPI)で除算し、余りは切り捨てた値で文字と文字の間隔を決定し配置します。

TYPE2:

TYPE1 指定時と同様、印刷するプリンタ装置の解像度をアプリケーションで指定した行間隔(LPI)や文字間隔(CPI)で除算した値で文字と文字の間隔を決定し配置します。しかし、割り切れないような解像度を持つプリンタ装置に出力する場合、1 インチ単位内で座標の補正処理を行います。

NCOBU00654

@CBR_PrinterANK_Size (ANK 文字サイズの指定)

ANK 文字サイズの指定

```
                          ì TYPE-M ü
@CBR_PrinterANK_Size=í TYPE-PC ý
                          î TYPE-G þ
```

印刷ファイルで、デフォルトの ANK 文字サイズ(CHARACTER TYPE 句および PRINTING POSITION 句を指定していない英数字項目の文字サイズ)を変更する場合に指定します。なお、本指定の有効範囲は実行単位内です。

TYPE-M:

デフォルト ANK 文字サイズを OSIV 系システムでのサイズに近づけ、9.6 ポイントで印字します。

TYPE-PC:

デフォルト ANK 文字サイズを PC で標準的なサイズとし、10.5 ポイントで印字します。

TYPE-G:

デフォルト ANK 文字サイズを FMG シリーズでのサイズに近づけ、10.8 ポイントで印字します。

省略された場合、7.0 ポイントで印字します。

注意

固定サイズのフォント(ラスタフォント)が印刷用フォントとして選択されている場合、本指定は有効とならない場合があります。 この場合、そのフォントがサポートするサイズのうち、一番近いサイズで印字されます。

一般的な例として、シリアルプリンタに搭載されているデバイスフォントは、通常、10.5 ポの固定サイズでしか印字できません。 このようなフォントが選択された場合、本実行環境情報の指定にかかわらず、つねに 10.5 ポで印字されます。

@CBR_SQL_MESSAGE_TRACE (SQL Server から送信されたユーザ定義メッセージのトレース出力の指定)

SQL Server から送信されたユーザ定義メッセージのトレース出力の指定

```
                                ¡YESü
@CBR_SQL_MESSAGE_TRACE=í        ý
                                îNO þ
```

"YES" を指定した場合に、SQL Server 上で実行されているストアドプロシージャやトリガなどのデータベースオブジェクトから送信されたユーザ定義メッセージの出力を、.NET Framework のトレースリスナーに転送します。 "NO" を指定した場合、および、本指定を省略した場合は、ユーザ定義メッセージはトレースリスナーに転送されません。

この機能は、ADO.NET 接続(SqlClient データプロバイダー利用)で COBOL のデータベース機能を使用したクライアントアプリケーションを実行した場合に有効となる機能です。

- ユーザ定義メッセージは以下の場合だけ、トレースリスナーに転送されます。
  - クライアントの COBOL アプリケーションから直接呼び出されたストアドプロシージャから送信された場合
  - クライアントの COBOL アプリケーションでテーブルのデータの更新操作、挿入操作、および削除操作によって発生したトリガから送信された場合
- NetCOBOL for .NET で作成した SQL CLR データベースオブジェクトを実行している場合に、NetCOBOL for .NET ランタイムシステムから重大コードが U 以外のメッセージが送信されたときも、トレースリスナーに転送されます。

.NET Framework のトレースリスナーについての詳細は、 .NET Framework のドキュメントのトレース リスナーを参照してください。

NCOBU00655

@CBR_SYSERR_EXTEND (SYSERR 出力情報の拡張の指定)

SYSERR 出力情報の拡張の指定

```
@CBR_SYSERR_EXTEND=YES
```

DISPLAY 文で SYSERR に出力するとき、プロセス ID、スレッド ID の情報を付加します。

@CBR_TRAILING_BLANK_RECORD (行順ファイルのレコード内後置空白を取り除くまたは有効にする指定)

行順ファイルのレコード内後置空白を取り除くまたは有効にする指定

```
                                  ¡REMOVEü
@CBR_TRAILING_BLANK_RECORD=í        ý
                                  îVALID  þ
```

行順ファイルの WRITE 文の実行時に、 レコード内の後置空白を取り除く(REMOVE)か、有効にする(VALID)かを指定します。

本指定を省略した場合、"VALID"が指定されたものとみなします。

NCOBU00657

@CBR_THREAD_TIMEOUT (スレッド同期制御サブルーチンの待ち時間の指定)

スレッド同期制御サブルーチンの待ち時間の指定

```
@CBR_THREAD_TIMEOUT=待ち時間（秒）
```

スレッド同期制御サブルーチンで無限待ちを指定した場合、待ち時間を変更する際に指定します。 待ち時間は、0 から最大 2147483(秒)の数字で指定します(注 1)。 指定しない場合、無限待ちとなります。 [参照] "スレッド同期制御サブルーチン"

注 1: 範囲を越える値を指定した場合、動作は保障されません。

@CBR_TextAlign (文字行内配置時の上端/下端合わせの指定)

文字行内配置時の上端/下端合わせの指定

```
                  ì TOP      ü
@CBR_TextAlign=í             ý
                  î BOTTOMþ
```

FORMAT 句なし印刷ファイルで、行内に配置する印字文字の位置を指定します。なお、本指定の有効範囲は実行単位内です。

印字する文字を文字セルの左上端を基準点として行内に上端合わせで配置する(TOP)か、左下端を基準点として行内に下端合わせで配置する(BOTTOM)かを指定します。 本指定を省略した場合、"BOTTOM"が有効となります。

NCOBU00659

@DefaultFCB_Name (デフォルト FCB 名の指定)

デフォルト FCB 名の指定

```
@DefaultFCB_Name=FCBxxxx
```

デフォルトで使用する FCB の名前を指定します。ここで指定する FCB 名(FCBxxxx)は、I 制御レコードから FCB を動的に変更する場合と同様、FCB 制御文をあらかじめ定義しておく必要があります。

以下に、FCB 制御文を複数定義しておき、行間隔=6LPI、ページ内行数=66 行、ページ内印刷開始行位置=1 行目、用紙縦長=11 インチと定義されている FCB をデフォルト FCB に設定する例を示します。

例

```
@DefaultFCB_Name=FCB6LPI
FCB6LPI=LPI((6,66)),CH1(1),SIZE(110)
FCB8LPI=LPI((8,88)),CH1(4),SIZE(110)
FCBA4LD=LPI((12)),CH1(1),FORM(A4,LAND)
```

FCBxxxx (FCB 制御文の指定)

FCB 制御文の指定

```
FCBxxxx=FCB 制御文
```

xxxx には、I 制御レコードに指定した FCB 名を指定します。 FCB 制御文の指定形式については、 "FCB" を参照してください。

NCOBU00661

FOVLDIR (フォームオーバレイパターンのフォルダの指定)

フォームオーバレイパターンのフォルダの指定

```
FOVLDIR=フォルダ
```

フォームオーバレイパターンの格納先フォルダを絶対パス名で指定します。省略された場合、フォームオーバレイパターンの焼付けは行われません。

なお、複数のフォルダを指定することはできません。

FOVLNAME (フォームオーバレイパターンのファイル名の指定)

フォームオーバレイパターンのファイル名の指定

```
FOVLNAME=ファイル名
```

フォームオーバレイパターンのファイル名から先頭 4 文字の形式部分を除いた、後半のファイル名を 4 文字以内で指定します。

例

フォームオーバレイパターンファイルが "¥C:¥FOVLDATA¥KOL5FOVL.OVD" の場合

```
FOVLDIR=C:¥FOVLDATA
FOVLTYPE=KOL5
FOVLNAME=FOVL
OVD_SUFFIX=OVD
```

NCOBU00663

FOVLTYPE (フォームオーバレイパターンのファイル名の形式の指定)

フォームオーバレイパターンのファイル名の形式の指定

```
FOVLTYPE=形式
```

フォームオーバレイパターンのファイル名の先頭 4 文字が "KOL5" (省略値)以外の場合、形式に、ファイル名の先頭 4 文字を指定します。 ファイル名に形式を持たない場合は、 "None" を指定してください。

ファイル識別名 (表示ファイルから使用する情報ファイルの指定)

表示ファイルから使用する情報ファイルの指定

```
ファイル識別名=情報ファイル名
```

表示ファイルを利用する場合のファイル識別名の指定方法について説明します。

ファイル識別名には、COBOL ソースプログラムの ASSIGN 句に記述したファイル識別名を指定します。プログラム中でファイル識別名を英小文字で記述した場合でも、環境変数情報のファイル識別名は英大文字で指定してください。

情報ファイル名の指定は、MeFt を使用する場合は必須となります。情報ファイルには、実際に使用する MeFt のプリンタ情報ファイル名を指定します。

例

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO GS-PRTFILE
〔COBOL ソースプログラムの SYMBOLIC DESTINATION 句の記述〕
 SYMBOLIC DESTINATION IS "PRT"
〔実際に使用する情報ファイルの名前〕
 C:¥WORK.WRC
〔環境変数情報〕
 PRTFILE=C:¥WORK.WRC
```

NCOBU00665

ファイル識別名 (プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定)

プログラムで使用するプリンタ情報ファイルおよび各種パラメタの指定

```
ファイル識別名=プリンタ情報ファイル名 [,[,FONT(フォントテーブル名)]]
```

FORMAT 句付き印刷ファイルを利用する場合のファイル識別名の指定方法について説明します。

ファイル識別名には、COBOL ソースプログラムの ASSIGN 句に記述したファイル識別名を指定します。 プログラム中でファイル識別名を英小文字で記述した場合でも、環境変数情報のファイル識別名は英大文字で指定してください。

プリンタ情報ファイルには、実際に使用する MeFt のプリンタ情報ファイル名を指定します。

フォントテーブル名には、フォントテーブルのファイル名を絶対パスで指定します。 フォントテーブルには、PRINTING MODE 句の FONT-nnn(書体番号)指定に対応するフォントフェース名や印字スタイルの情報を定義します。 ここで指定されたフォントテーブルは、そのファイルに対してだけ有効なフォントテーブルとなります。すべての印刷ファイルに共通なフォントテーブル(実行環境単位で有効なフォントテーブル)の指定については、"@CBR_PrintFontTable (印刷ファイルで使用するフォントテーブルの指定)" を参照してください。

また、フォントテーブルの詳細については、 "印字文字" の "印字文字の書体" および "印字文字のスタイル" 、 "フォントテーブル" を参照してください。

なお、実行環境単位で有効なフォントテーブル名とファイル単位で有効なフォントテーブル名が異なる場合、ファイル単位の指定が優先されます。

例 1

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRTFILE
〔実際に使用するプリンタ情報ファイルの名前〕
 C:¥WORK.WRC
〔環境変数情報〕
 PRTFILE=C:¥WORK.WRC
```

例 2

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRTFILE
〔実際に使用するプリンタ情報ファイルの名前〕
 C:¥WORK.PRC
〔実際に使用するフォントテーブルの名前〕
 C:¥DEFAULT.FTB
〔環境変数情報〕
 PRTFILE=C:¥WORK.PRC,,FONT(C:¥DEFAULT.FTB)
```

ファイル識別名 (プログラムで使用するファイルの指定)

プログラムで使用するファイルの指定

```
                                  ì BSAM ü
ファイル識別名=[ファイル名][,[í BTRV ý][,MOD | CONCAT(ファイル名１ ファイル名２...)]]
                                  î RDM  þ
```

ファイル識別名には、COBOL ソースプログラムの ASSIGN 句に記述したファイル識別名を指定し、ファイル名には実際に処理を行うファイルの名前を指定します。 この指定は、プログラム中のファイルと実際に処理するファイルを関連付けるために行います。

- BSAM は、ファイルの高速処理を実現するオプションです。 詳細は、ファイルの高速処理を参照してください。
- BTRV は、Btrieve ファイルを処理します。 詳細は、Btrieve ファイルを参照してください。
- RDM は、RDM ファイルを処理します。 詳細は、PowerRDBconnector を参照してください。
- MOD を指定すると、OPEN OUTPUT 文の実行で、既に存在するファイルにレコードを追加することができます。 詳細は、ファイル追加書きを参照してください。
- CONCAT を指定すると、複数のファイルを連結して、レコードを参照・更新することができます。 詳細は、ファイルの連結を参照してください。
- ファイル名にファイルパスを指定した場合、絶対パスと相対パスを指定できます。 相対パスが指定された場合は、カレントフォルダからの相対パスになります。
- プログラム中でファイル識別名を英小文字で記述した場合でも、環境変数情報は英大文字で指定してください。

例

- COBOL ソースプログラムの ASSIGN 句の記述

  ASSIGN TO OUTFILE

- 実際に処理を行うファイルの名前

  ```
  F:¥WORK.DAT
  ```

- 環境変数情報

  ```
  OUTFILE=F:¥WORK.DAT
  ```

NCOBU00667

ファイル識別名 (プログラムで使用するプリンタおよび各種パラメタの指定)

プログラムで使用するプリンタおよび各種パラメタの指定

```
                  ì LPTn:                    ü
ファイル識別名=   í COMn:                    ý[,[,INF(印刷情報ファイル名)[,FONT(フォントテーブル名)]]]
                  î PRTNAME:プリンタ名       þ
```

FORMAT 句なし印刷ファイルを利用する場合のファイル識別名の指定方法について説明します。

ファイル識別名には、COBOL ソースプログラムの ASSIGN 句に記述したファイル識別名を指定します。 プログラム中でファイル識別名を英小文字で記述した場合でも、環境変数情報のファイル識別名は英大文字で指定してください。

"LPTn:"/"COMn:"/"PRTNAME:プリンタ名"には、実際に帳票出力を行うプリンタが接続されているローカルプリンタポート名(LPTn:)/通信ポート名(COMn:)/プリンタの名前(PRTNAME:プリンタ名)を指定します。 ローカルプリンタポート名(LPTn:)/通信ポート名(COMn:)/プリンタの名前(PRTNAME:プリンタ名)の指定は、省略することもできます。 省略した場合、印刷情報ファイルの"PRTOUT"キーでこれらを指定する必要があります。 ファイル識別名および印刷情報ファイルに、それぞれ異なるローカルプリンタポート名(LPTn:)/通信ポート名(COMn:)/プリンタの名前(PRTNAME:プリンタ名)が指定された場合、印刷情報ファイルの指定が優先されます。 どちらも省略された場合または指定に誤りがある場合は、実行時エラーとなります。

印刷情報ファイル名には、印刷情報ファイル名を絶対パスで指定します。 印刷情報ファイルには、出力される帳票に関する状態制御を行ういくつかの情報を定義します。 印刷情報ファイルの詳細については、 "印刷情報ファイル" を参照してください。

フォントテーブル名には、フォントテーブルのファイル名を絶対パスで指定します。 フォントテーブルには、PRINTING MODE 句の FONT-nnn(書体番号)指定に対応するフォントフェース名や印字スタイルの情報を定義します。 ここで指定されたフォントテーブルは、そのファイルに対してだけ有効なフォントテーブルとなります。すべての印刷ファイルに共通なフォントテーブル(実行環境単位で有効なフォントテーブル)の指定については、"@CBR_PrintFontTable (印刷ファイルで使用するフォントテーブルの指定)" を参照してください。

また、フォントテーブルの詳細については、 "印字文字" の "印字文字の書体" および "印字文字のスタイル"、 "フォントテーブル" を参照してください。

なお、実行環境単位で有効なフォントテーブル名とファイル単位で有効なフォントテーブル名が異なる場合、ファイル単位の指定が優先されます。

印刷情報ファイル名とフォントテーブル名の指定順序は、どちらが先でもかまいません。

例 1

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRTFILE
〔実際に帳票を出力するプリンタの名前〕
 FUJITSU FMLBP
〔実際に使用する印刷情報ファイル名とそのパス〕
 C:¥PRINT.INF
〔実際に使用するフォントテーブルとそのパス〕
 C:¥DEFAULT.FTB
〔環境変数情報〕
 PRTFILE=PRTNAME:FUJITSU FMLBP,,INF(C:¥PRINT.INF),FONT(C:¥DEFAULT.FTB)
 または
 PRTFILE=PRTNAME:FUJITSU FMLBP,,FONT(C:¥DEFAULT.FTB),INF(C:¥PRINT.INF)
```

例 2

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRINTER-1
〔実際に帳票を出力するローカルプリンタポート名〕
 LPT1:
〔実際に使用する印刷情報ファイル名とそのパス〕
 使用しない
〔実際に使用するフォントテーブルとそのパス〕
 C:¥DEFAULT.FTB
〔環境変数情報〕
 PRINTER-1=LPT1:,,FONT(C:¥DEFAULT.FTB)
```

例 3

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRINTER-2
〔実際に帳票を出力するローカルプリンタポート名〕
 COM1:
〔実際に使用する印刷情報ファイル名とそのパス〕
 使用しない
〔実際に使用するフォントテーブルとそのパス〕
 使用しない
〔環境変数情報〕
 PRINTER-2=COM1:
```

例 4

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRTFILE
〔実際に帳票を出力するローカルプリンタポート名〕
 FUJITSU FMLBP
〔実際に使用する印刷情報ファイル名とそのパス〕
 C:¥PRINT.INF
〔実際に使用するフォントテーブルとそのパス〕
 使用しない
〔環境変数情報〕
 PRTFILE=,,INF(C:¥PRINT.INF)
〔印刷情報ファイル(C:¥PRINT.INF)の指定〕
 PRTOUT=PRTNAME:FUJITSU FMLBP
```

例 5

```
〔COBOL ソースプログラムの ASSIGN 句の記述〕
 ASSIGN TO PRTFILE
〔実際に帳票を出力するローカルプリンタポート名〕
 FUJITSU FMLBP
〔実際に使用する印刷情報ファイル名とそのパス〕
 C:¥PRINT.INF
〔実際に使用するフォントテーブルとそのパス〕
 使用しない
〔環境変数情報〕
 PRTFILE=PRTNAME:FUJITSU XL-5600,,INF(C:¥PRINT.INF)
〔印刷情報ファイル(C:¥PRINT.INF)の指定〕
 PRTOUT=PRTNAME:FUJITSU FMLBP
```

NCOBU00668

@GOPT (実行時オプションの指定)

実行時オプションの指定

```
@GOPT=実行時オプションの並び
```

実行時オプションの並びには、実行時オプションを指定します。 実行時オプションの指定形式については、 "実行時オプションの形式" を参照してください。

例

```
      @GOPT=s01010101
```

@MessOutFile (メッセージを出力するファイルの指定)

メッセージを出力するファイルを指定します。

```
@MessOutFile=ファイル名
```

実行時メッセージおよび UPON SYSERR 指定の DISPLAY 文による出力を､指定されたファイルに出力します。ファイル名を指定した場合、メッセージボックスは画面に表示されません。

ファイル名にファイルパスを指定した場合､絶対パスと相対パスを指定できます｡相対パスが指定された場合は、ベースフォルダからの相対パスとなります。 EXE ファイルのベースフォルダは、EXE ファイルを含むフォルダとなり、Web フォームや Web サービスのベースフォルダは、仮想フォルダとなります。

すでに、同一のファイルが存在する場合は、そのファイルに情報を追加します

出力するファイルのコード系を指定するには、実行環境変数@MessOutFile_CODE を指定してください。

各種アプリケーションサーバ配下で動作するプログラムの実行時メッセージは、@MessOutFile を必ず指定して出力させてください。

- 入出力機能の出力ファイルおよび小入出力の出力ファイルとして指定したファイルを @MessOutFile に指定した場合、出力されるファイルの内容は保証されません。
- ファイルに出力できない場合、実行時メッセージはメッセージボックスに出力されます。

[参照] 実行環境変数-@MessOutFile_CODE (メッセージを出力するファイルのコード系の指定)

NCOBU00670

@MessOutFile_CODE (メッセージを出力するファイルのコード系の指定)

```
                    ¡ACP       ü
@MessOutFile_CODE=í            ý
                    îUNICODEþ
```

実行時メッセージおよび UPON SYSERR 指定の DISPLAY 文の出力先となるファイルのコード系を指定します。

出力ファイルのコード系を ASCII/シフト JIS として扱う(ACP)か、 Unicode として扱う(UNICODE)かを指定します。 本指定を省略した場合、"UNICODE"が有効になります。

- 本指定は、実行環境変数@MessOutFile が指定された場合だけ有効になります。
- すでにファイルが存在する場合は、本指定は有効になりません。

[参照] 実行環境変数 -@MessOutFile (メッセージを出力するファイルの指定)

@NoMessage (実行時メッセージの抑止指定)

実行時メッセージの抑止指定

```
@NoMessage=YES
```

以下の実行時メッセージを抑止します。

- U レベル以外の実行時メッセージ
- オペレータの応答を必要としない実行時メッセージ

NCOBU00672

@ODBC_Inf (ODBC 情報ファイルの指定)

```
@ODBC_Inf=ODBC 情報ファイル名
```

ランタイムシステムが ODBC を使用するために必要な SQL 情報は、ODBC 情報ファイルに指定します。 ODBC 情報ファイルに指定された情報は、主にクライアントとサーバ間のコネクション(CONNECT 文などで指定する)を確立するために使用されます。

例: @ODBC_Inf=C:¥DBMSACS.INF

NetCOBOL for .NET では、アプリケーション構成ファイルを使用して SQL 情報を設定することを推奨しています。ただし、実行環境変数@ODBC_Inf に指定された ODBC 情報ファイルとアプリケーション構成ファイルの両方がある場合は、ODBC 情報ファイルに設定された SQL 情報が優先されます。

アプリケーション構成ファイルに SQL 情報を設定する方法については、アプリケーション構成ファイルを使用した SQL 情報の設定を参照してください。また、ODBC 情報ファイルの内容については、ODBC 情報ファイルの形式を参照してください。

OVD_SUFFIX (フォームオーバレイパターンのファイルの拡張子の指定)

フォームオーバレイパターンのファイルの拡張子の指定

```
OVD_SUFFIX=拡張子
```

フォームオーバレイパターンのファイルの拡張子が"OVD"(省略値)以外の場合、拡張子に、拡張子の文字列を指定します。ファイル名に拡張子がない場合には、文字列"None"を指定してください。

NCOBU00674

@PRN_FormName_xxx (用紙名の指定)

用紙名の指定

```
@PRN_FormName_xxx=用紙名
```

FORMAT 句なし印刷ファイルで、I 制御レコードの SIZE(用紙サイズ)フィールドに指定した任意の 3 文字以内の文字列を本環境変数情報名の xxx 部分に置き換え、本環境変数情報名に対応して用紙名を対応付けて指定します。

用紙名について、以下に説明します。

各プリンタの〔プロパティ〕ダイアログの〔デバイスの設定〕シートの"給紙方法と用紙の割り当て"で、各給紙方法の右側にあるリストボックスに表示される用紙名のことをいいます。

- 用紙名は、プリンタまたはプリンタドライバによりサポート範囲が異なります。 プリントマネージャーまたは"プリンタ"の"プロパティ"で、目的のプリンタドライバがサポートしている用紙サイズを確認するか、プリンタの取扱い説明書を参照してください。
- 作成した用紙名を指定する場合、用紙定義後、システムを再起動する必要があります。 システムを再起動しないと、定義した用紙名が有効とならない場合がありますので注意してください。

  用紙の作成を行う場合、以下の手順で行います。(Windows 8.1 の場合)

  コントロールパネルから[デバイスとプリンター]を選択します。表示されたプリンタの一覧からプリンタを選択し、[プリントサーバー プロパティ]シートの[用紙]ページで作成します。

例

```
〔I 制御レコードの SIZE フィールドの指定〕
 ABC
〔xxx 部分置換後の環境変数情報名〕
 @PRN_FormName_ABC
〔環境変数情報名と用紙名の対応付け〕
 @PRN_FormName_ABC=15x11
```

@PrinterFontName (印刷ファイルで使用するフォントの指定)

印刷ファイルで使用するフォントの指定

```
@PrinterFontName=(明朝体フォント名，ゴシック体フォント名)
```

印刷ファイルで使用するフォントのフォントフェース名を指定します。

- フォントフェース名は、32 バイト以内の英数字または日本語文字で指定してください。
- フォントフェース名の前後に、全角または半角の空白が含まれる場合、これらの空白もフォントフェース名の一部とみなされます。
- フォントフェース名に半角のコンマ(,)が含まれるフォント名は指定できません。
- フォントフェース名に指定できるフォント名は、コントロールパネルのフォントを選択し、表示されるフォントの一覧から選択してください。
    1. コントロールパネルの“フォント”を選択します。
    2. 表示されるフォントの一覧から該当フォントを選択し、ダブルクリックします。
    3. 表示ウィンドウの「書体名： 」のアンダーライン箇所に書体名が表示されます。

備考

- 明朝体フォントの検索順序は、(1)本環境変数情報で指定した明朝体フォント、(2)明朝、(3)MS 明朝となります。
- ゴシック体フォントの検索順序は、(1)本環境変数情報で指定したゴシック体フォント、(2)ゴシック、(3)MS ゴシックとなります。
- 検索の結果、上記のどのフォントもシステムにインストールされていない場合には、アプリケーション実行時に JMP0320I 'FONT'エラーとなります。

NCOBU00676

SYSIN のアクセス名 (小入出力機能の入力ファイルの指定)

小入出力機能の入力ファイルの指定

```
SYSIN のアクセス名=ファイル名
```

SYSIN のアクセス名には、翻訳オプション SSIN に指定した環境変数情報名を指定し、ファイル名には、小入出力機能の ACCEPT 文を実行したときのデータの入力先となるファイルの名前を指定します。 [参照] "SSIN (ACCEPT 文のデータの入力先)"

ファイル名にファイルパスを指定した場合、絶対パスと相対パスを指定できます。相対パスが指定された場合は、カレントフォルダからの相対パスになります。

例

- COBOL ソースプログラムを翻訳するときの翻訳オプション SSIN の指定

  SSIN(INFILE)

- ACCEPT 文の入力先ファイルの名前

  ```
  A:¥INDATA.TXT
  ```

- 環境変数情報

  ```
  INFILE=A:¥INDATA.TXT
  ```

SYSOUT のアクセス名 (小入出力機能の出力ファイルの指定)

小入出力機能の出力ファイルの指定

```
SYSOUT のアクセス名=ファイル名
```

SYSOUT のアクセス名には、翻訳オプション SSOUT に指定した環境変数情報名を指定し、ファイル名には、小入出力機能の DISPLAY 文を実行したときのデータの出力先となるファイルの名前を指定します。 [参照] "SSOUT (DISPLAY 文のデータの出力先)"

ファイル名にファイルパスを指定した場合、絶対パスと相対パスを指定できます。 相対パスが指定された場合は、カレントフォルダからの相対パスになります。

すでに、同一のファイルが存在する場合は、ファイルは上書きされます。

例

- COBOL ソースプログラムを翻訳するときの翻訳オプション SSOUT の指定

  SSOUT(OUTFILE)

- DISPLAY 文の出力先ファイルの名前

  ```
  A:¥OUTDATA.TXT
  ```

- 環境変数情報

  ```
  OUTFILE=A:¥OUTDATA.TXT
  ```

NCOBU00678

ファイル入出力状態一覧

ここでは、入出力文を実行したときに、ファイル管理記述項の FILE STATUS 句に記述したデータ名に設定される値(入出力状態値)と詳細情報の意味を "表: 入出力状態値" に記述します。

入出力状態値

分類	値	詳細	意 味
成功	00	--	入出力文の実行が成功しました。
	02	--	入出力文の実行は成功しましたが、以下の状態のどちらかです ・ READ 文の実行で、読み込んだレコードの参照キーの値が、次のレコードの参照キーの値と等しい。 ・ WRITE 文または REWRITE 文の実行で、書き出すレコードと同じレコードキーの値を持つレコードが、すでにファイル中に存在しています。ただし、そのレコードキーは、重複した値が許されているので、誤りではありません。
	04	--	READ 文の実行は成功しましたが、入力したレコードの長さが最大レコード長よりも大きい。
	05	--	OPTIONAL 句指定のファイルで、入出力文の実行は成功しましたが、以下の状態のどれかになりました。 ・ ファイルに対して INPUT/I-O/EXTEND モードの OPEN 文を実行しましたが、ファイルが未創成状態でした。 ・ ファイルに対して INPUT モードの OPEN 文を実行しましたが、ファイルが存在しませんでした。ファイルは生成されず、最初の READ 文実行時にファイル終了条件（入出力状態値="10" ）となります。 ・ ファイルに対して I-O または EXTEND モードの OPEN 文を実行しましたがファイルが存在しませんでした。このとき、ファイルは生成されます。
	07	--	入出力文の実行は成功しましたが、以下の方法のどれかで参照したファイルは非リール／ユニット媒体上にありました。 ・ NO REWIND 指定の OPEN 文または CLOSE 文 ・ REEL/UNIT 指定の CLOSE 文
ファイル終了条件（不成功）	10	--	順呼出しの READ 文でファイル終了条件となりました。 ・ ファイルの終わりに達しました。 ・ 存在しない不定入力ファイルに対して初めての READ 文を実行しました。存在しない不定入力ファイルとは、OPTIONAL 句指定のファイルを INPUT モードで開いたが、そのファイルが未創成状態である場合のことです。
	14	--	順呼出しの READ 文でファイル終了条件となりました。 ・ 読み込んだレコードの相対レコード番号の有効桁が、そのファイルの相

			対キー項目の大きさより大きい。
無効キー条件（不成功）	21	--	レコードキーの順序誤り。次の状態のどれかです。 ・ 順呼出しにおいて、READ 文とそれに続く REWRITE 文との間で、主レコードキーの値が変更されました。 ・ 乱呼出しまたは動的呼出しにおいて、主キーに DUPLICATES 指定の記述があるファイルで、READ 文とそれに続く REWRITE 文または DELETE 文との間で、主レコードキーの値が変更されました。 ・ 順呼出しにおいて、WRITE 文の実行のときに主レコードキーの値が昇順になっていません。
	22	--	WRITE 文または REWRITE 文の実行時に、書こうとしたレコードの主レコードキー または副レコードキーの値がすでにファイル中に存在しています。 ただし、主レコードキーまたは副レコードキーに DUPLICATES 指定が記述されている場合を除きます。
	23	--	レコードが見つかりません。 ・ START 文または乱呼出しの READ/REWRITE/DELETE 文の実行で、指定されたキー値をもつレコードがファイル中に存在しません。 ・ 相対レコード番号に 0 が指定されました。
	24	--	次の状態のどれかです。 ・ WRITE 文または CLOSE 文の実行で、領域不足が発生しました。 ・ WRITE 文の実行で、指定されたキーがキーレンジ外です。 ・ 区域外書出し発生後、さらに WRITE 文を実行しようとしました。 ・ WRITE 文の実行で、ファイルの最大サイズを超えました。
永続誤り条件（不成功）	30	--	物理的なエラーが発生しました。
	34	--	・ WRITE 文または CLOSE 文の実行で、領域不足が発生しました。 ・ WRITE 文の実行で、ファイルの最大サイズを超えました。
	35		OPTIONAL 句指定のないファイルに対して、INPUT/I-O/EXTEND モードの OPEN 文を実行しましたが、ファイルが未創成状態でした。
	37	--	指定された機能は、未サポートです。
	38		以前に CLOSE LOCK 文を実行したファイルに対して OPEN 文を実行しました。
	39	--	OPEN 文の実行時に、プログラム中でそのファイルに指定した属性と矛盾するファイルが割り当てられました。
論理誤り条件（不成功）	41	--	すでに開いたファイルに対して、OPEN 文を実行しました。
	42	--	開いていないファイルに対して、CLOSE 文を実行しました。
	43	--	順呼出しまたは主キーに DUPLICATES 指定を記述したファイルに対する DELETE 文または REWRITE 文の実行で、先行する入出力文が成功した READ

			文ではありませんでした。
	44	--	以下の状態のどちらかです。 ・ WRITE/REWRITE 文実行時のレコード長がプログラムの記述で決められた最大レコード長より大きいかまたはレコード長として誤った数値が指定されました。 ・ REWRITE 文実行時にそのレコードは書き換えるレコードの長さと等しくありませんでした。
	46	--	順呼出しの READ 文の実行で、次のどちらかの理由でファイル位置指示子が不定です。 ・ 先行する START 文が不成功です。 ・ 先行する READ 文が不成功（ファイル終了条件も含む）です。
	47	--	INPUT/I-O モードで開かれていないファイルに対して READ 文または START 文を実行しました。
	48	--	OUTPUT/EXTEND（順・相対・索引）または I-O（相対・索引）モードで開かれていないファイルに対して WRITE 文を実行しました。
	49	--	I-O モードで開かれていないファイルに対して REWRITE 文または DELETE 文を実行しました。
その他の誤り（不成功）	90	--	他のどれにも含まれないエラーです。以下のような状態が考えられます。 ・ ファイル情報が不完全またはその情報に誤りがあります。 ・ OPEN/CLOSE 文の実行時に、関数で何らかのエラーが発生しました。 ・ 以前に CLOSE 文が入出力状態値 90 で不成功になったファイルに対して、入出力文を実行しようとしました。 ・ 主記憶などの資源が利用できません。 ・ 正しく閉じられていないファイルに対して、OPEN 文を実行しました。 ・ 区域外書出しによる誤りの発生後、レコードを書き出そうとしました。 ・ no-space 状態発生後、レコードを書き出そうとしました。 ・ テキストファイルのレコードに不当な文字があります。 ・ コードセットに変換できない文字があります。 ・ 同一のファイルに対し、多数のアプリケーションから OPEN 要求がありました。その結果、ロックテーブルに不足が発生しました。 ・ 必要な関連製品のローディングに失敗しました。 ・ 上記以外の誤りが存在します。その入出力動作に関してはそれ以上の情報がありません。 ・ システムエラーが発生しました。
		MeFt	MeFt がエラーを検出しました。
	91	--	・ ファイルが割り当てられていません。 ・ OPEN 文実行時に、ファイル識別名と物理ファイル名の対応付けがあり

			ません。
	92	--	排他エラーが発生しました。(RDM ファイル)
	93	--	排他エラーが発生しました。(ファイルロック)
	99	--	排他エラーが発生しました。(レコードロック)
		MeFt	システム異常が発生しました。

FORMAT 句付き印刷ファイルおよび表示ファイルの場合は、入出力状態値と詳細情報が通知されます。詳細情報の "MeFt" は MeFt の通知コードを参照してください。 "--" は詳細情報に 0000 が通知されます。

NCOBU00679

組込み関数

NetCOBOL for .NET で使用できる組込み関数の一覧を以下に示します。

組込み関数一覧

分類	関数	用途	関数の型
長さ	LENGTH	データ項目や定数の長さを求めます。	整数
	LENG	バイト数を求めます。	整数
	STORED-CHAR-LENGTH	有効文字の長さを求めます。	整数
大きさ	MAX	最大値を求めます。	整数、数字または英数字
	MIN	最小値を求めます。	整数、数字または英数字
	ORD-MAX	最大値の順序位置を求めます。	整数
	ORD-MIN	最小値の順序位置を求めます。	整数
変換	REVERSE	文字列の順序を逆にします。	英数字
	LOWER-CASE	大文字を小文字に変換します。	英数字
	UPPER-CASE	小文字を大文字に変換します。	英数字
	NUMVAL	数字文字を数値に変換します。	数字
	NUMVAL-C	コンマや通貨記号のある数字文字を数値に変換します。	数字
	NATIONAL	日本語文字に変換します。	日本語
	NATIONAL-OF	文字のエンコード、項類および字類を日本語に変換します。	日本語
	DISPLAY-OF	文字のエンコード、項類および字類を英数字に変換します。	英数字
	CAST-ALPHANUMERIC	項類および字類を英数字に変換します。	英数字
	UCS2-OF	エンコード方式を UCS-2 に変換します。	日本語
	UTF8-OF	エンコード方式を UTF-8 に変換します。	英数字
	UNICODE-OF	Unicode 文字に変換します。	英数字または日

			本語
	ACP-OF	ACP(Ansi Code Page)文字に変換します。	英数字または日本語
文字操作	CHAR	プログラムの文字の大小順序において、指定した位置にある１文字を求めます。	英数字
	ORD	プログラムの文字の大小順序において、指定した文字の順序位置を求めます。	整数
数値操作	INTEGER	指定した値を超えない最大の整数を求めます。	整数
	INTEGER-PART	整数部を求めます。	整数
	RANDOM	乱数を求めます。	数字
金利計算	ANNUITY	元金を１とし、利率と期間から均等払いの比率の近似値を求めます。	数字
	PRESENT-VALUE	減価率による現在の価格を求めます。	数字
日付操作	CURRENT-DATE	現在の日付、時刻、グリニッジ標準時からの時差を求めます。	英数字
	DATE-OF-INTEGER	通日に対応する年月日を求めます。	整数
	DAY-OF-INTEGER	通日に対応する年日を求めます。	整数
	INTEGER-OF-DATE	年月日に対応する通日を求めます。	整数
	INTEGER-OF-DAY	年日に対応する通日を求めます。	整数
	WHEN-COMPILED	プログラムが翻訳された日時を求めます。	英数字
算術計算	SQRT	平方根の近似値を求めます。	数字
	FACTORIAL	階乗を求めます。	整数
	LOG	自然対数を求めます。	数字
	LOG10	常用対数を求めます。	数字
	MEAN	平均値を求めます。	数字
	MEDIAN	中央値を求めます。	数字
	MIDRANGE	最小値と最大値の平均値を求めます。	数字
	RANGE	最大値と最小値の差を求めます。	整数または数字
	STANDARD-DEVIATION	標準偏差を求めます。	数字

	MOD	指定した法での指定した値の値を求めます。	整数
	REM	余りを求めます。	数字
	SUM	和を求めます。	整数または数字
	VARIANCE	分散を求めます。	数字
三角関数	SIN	正弦の近似値を求めます。	数字
	COS	余弦の近似値を求めます。	数字
	TAN	正接の近似値を求めます。	数字
	ASIN	逆正弦の近似値を求めます。	数字
	ACOS	逆余弦の近似値を求めます。	数字
	ATAN	逆正接の近似値を求めます。	数字
ENUM 処理	ENUM-OR	変数リストの OR 結果を求めます。	オブジェクト参照
	ENUM-AND	変数リストの AND 結果を求めます。	オブジェクト参照
	ENUM-NOT	変数リストの NOT 結果を求めます。	オブジェクト参照

SQL 情報

SQL に関連する情報について説明します。

このトピックの内容

NCOBU00681

サーバ情報

サーバ情報は、データベースのコネクション単位に設定する情報を指定します。この情報には、データベースの接続方法と、サーバごとに指定する情報名があります。

データベースの接続方法は、アプリケーション構成ファイルの<server>要素の type 属性に指定します。サーバごとに指定する情報名は、アプリケーション構成ファイルの<server>要素内に指定します。

> ODBC 情報ファイルには、データベースの接続方法を指定できません。 アプリケーション構成の形式については、アプリケーション構成ファイルの形式を参照してください。

データベースの接続方法を以下に示します。

データベース接続方法

type 属性	設定内容
adonet	ADO.NET 接続
odbc	ODBC 接続

type 属性を省略した場合は ODBC 接続になります。

サーバ毎に指定できる情報名と、その情報が利用できる接続を以下の表に示します。

サーバ情報の定義内容

情報名	設定内容	接続方法	
		ADO.NET	ODBC
@SQL_DATASRC	接続文字列名またはデータソース名注 1)	○	○
@SQL_DATASRC_KIND	データソース種別	×	○
@SQL_USERID	ユーザ ID	○	○
@SQL_PASSWORD	パスワード	○	○
@SQL_ACCESS_MODE	アクセスモード	×	○
@SQL_COMMIT_MODE	COMMIT モード	○	○
@SQL_QUERY_TIMEOUT	タイムアウト時間（秒）	○	○
@SQL_ISOLATION	トランザクション分離レベル	○	○
@SQL_MULTIPLE_ROWS	複数行操作のサポート	×	○
@SQL_CONCURRENCY	カーソルの同時実行	×	○

@SQL_ODBC_CURSORS	ODBC カーソルライブラリ	×	○
@SQL_VALUE_N_PADDING	日本語項目の空白づめ(値指定)	○	○
@SQL_TARGET_N_PADDING	日本語項目の空白づめ(相手指定)	○	○
@SQL_ADONET_CURSOR_DEFAULT	カーソル動作省略時のカーソル動作	○	×
@SQL_ADONET_FORUPDATE_CLAUSE	カーソル宣言の FOR UPDATE 句の無効化	○	×
@SQL_ODBC_CURSOR_DEFAULT	カーソル動作省略時の SQL オプション情報	×	○
@SQL_ODBC_CURSOR_UPDATE	更新カーソルの SQL オプション情報	×	○
@SQL_ODBC_MARS	ODBC MARS 機能	×	○
@SQL_AUTO_CURSOR_CLOSE	カーソルの自動クローズ機能	×	○
@SQL_ADONET_DATETIME_FORMAT	日付型の書式指定	○	×
@SQL_SPACE_TRUNC	ホスト変数の入力値の後置空白の無効化	○	○
@SQL_CODE_MAPPING	文字データのマッピングコードを指定	○	○
@SQL_STOREDPROCEDURE_CACHE	ストアドプロシージャのキャッシュ指定	×	○
@SQL_X_NULL_TERM	英数字項目に含まれるデータの NULL 値の無効化	○	○
@SQL_DATAPROVIDER_EXTENSION_DLL	コネクション単位のデータプロバイダー拡張ライブラリ指定	○	×

○：利用可×：利用不可

注 1)@SQL_DATASRC には、ADO.NET 接続の場合は接続文字列名を、ODBC 接続の場合はデータソース名を指定してください。

NCOBU00682

@SQL_DATASRC (接続文字列名またはデータソース名の指定)

接続文字列名またはデータソース名を指定します。

データベースの接続方法として ADO.NET が指定されている場合

接続文字列名を設定します。接続文字列名は、アプリケーション構成ファイルに設定した接続文字列情報の名前です。接続文字列情報の設定方法については、接続文字列の設定方法(ADO.NET)を参照してください。

データベースの接続方法として ODBC が指定されている場合

システムの ODBC データソースアドミニストレータで指定したデータソース名を設定します。

@SQL_DATASRC_KIND に FILE_DS が指定されている場合は、データソース名にファイルデータソース名を記述することができます。

```
@SQL_DATASRC={接続文字列名 | データソース名}
```

- データソース名の最大長は、コネクションを確立するデータソースの仕様により異なります。したがって、ODBC ドライバと関係する環境のマニュアルを参照してください。
- コンピューティング エミュレーター上でデバッグする場合、 ファイルデータソース名に approot フォルダからの相対パスを指定することはできません。 任意のフォルダにコピーし、その絶対パスを指定してください。

ファイルデータソース

ファイルデータソースの拡張子は「.dsn」です。 ファイルデータソースの形式は、例えば以下のようになります。

```
[ODBC]
DRIVER=SQL Server Native Client 10.0
DATABASE=AzureSQL1
UID=FujitsuTaro@Jxxxxxxxxx.database.windows.net
APP=Microsoft Data Access Components
SERVER=Jxxxxxxxxx.database.windows.net
```

ファイルデータソース名は、アプリケーション構成ファイルに以下のように設定されます。

例: ファイルデータソース名が「SetAzure.dsn」の場合

```
<fujitsu.cobol>
  <runtime>
    <sqlSettings>
      <serverList>
        <server name="SQLAZURE" type="odbc" description="SQL Azure Connection">
          <add key="@SQL_DATASRC" value="SetAzure.dsn" />
          <add key="@SQL_DATASRC_KIND" value="FILE_DS" />
        </server>
      </serverList>
    </sqlSettings>
  </runtime>
</fujitsu.cobol>
```

NCOBU00683

@SQL_DATASRC_KIND (データソース種別の指定)

データソースの種別を指定します。 デフォルトは MACHINE_DS です。 文字列 MACHINE_DS を指定した場合はマシンデータソースを使用します。 マシンデータソースはユーザデータソースとシステムデータソースの総称です。 文字列 FILE_DS を指定した場合はファイルデータソースを使用します。

```
                          ì MACHINE_DS   ü
@SQL_DATASRC_KIND=í                      ý
                          î FILE_DS      þ
```

データソース種別に FILE_DS を指定したとき、サーバ情報とファイルデータソースの両方にユーザ ID、パスワードが指定されている場合は、サーバ情報のユーザ ID、パスワードが有効になります。

@SQL_USERID (ユーザ ID の指定)

データベースを操作するためのユーザ ID を指定します。

データベースの接続方法として ADO.NET が指定されている場合は、接続文字列中のユーザ ID よりも、@SQL_USERID に指定したユーザ ID の方が有効になります。

```
@SQL_USERID=ユーザ ID
```

- 接続文字列や ODBC データソースで、Windows 認証(注)を設定している場合は、@SQL_USERID にはユーザ ID を指定しないでください。 ユーザ ID が指定されている場合、ADO.NET データプロバイダーや ODBC ドライバによっては、Windows 認証が行われないことがあります。

  (注)「Windows 認証」はデータベースによって呼び方が異なります。「Windows 統合セキュリティ」や「信頼済み接続」などと呼ばれている場合があります。

- 設定する値の最大長は、コネクションを確立するデータソースの仕様により異なります。 したがって、ADO.NET データプロバイダーまたは、ODBC ドライバと関係する環境のマニュアルを参照してください。

NCOBU00685

@SQL_PASSWORD (パスワードの指定)

データベースを操作するためのパスワードを指定します。

データベースの接続方法として ADO.NET が指定されている場合は、接続文字列中のパスワードよりも、@SQL_PASSWORD に指定したパスワードの方が有効になります。

```
@SQL_PASSWORD=パスワード
```

- 接続文字列や ODBC データソースで、Windows 認証(注)を設定している場合は、@SQL_PASSWORD にはパスワードを指定しないでください。 パスワードが指定されている場合、ADO.NET データプロバイダーや ODBC ドライバによっては、Windows 認証が行われないことがあります。

  (注)「Windows 認証」はデータベースによって呼び方が異なります。「Windows 統合セキュリティ」や「信頼済み接続」などと呼ばれている場合があります。

- このパスワードは、実行環境設定ユーティリティまたは ODBC 情報設定ユーティリティを使用して暗号化してください。
- 設定する値の最大長は、コネクションを確立するデータソースの仕様により異なります。 したがって、ADO.NET データプロバイダーまたは、ODBC ドライバと関係する環境のマニュアルを参照してください。

@SQL_ACCESS_MODE (アクセスモードの指定)

データ ベースに対するアクセスモードを指定します。 デフォルトは READ_WRITE です。 文字列 READ_ONLY を指定した場合は読込み専用となります。 文字列 READ_WRITE を指定した場合は読込みおよび書込みが可能となります。 ただし、ODBC ドライバによって、指定に対する動作が異なる場合があります。

```
                      ¡READ_ONLY ü
@SQL_ACCESS_MODE=í               ý
                      îREAD_WRITEþ
```

NCOBU00687

@SQL_COMMIT_MODE (COMMIT モードの指定)

データベースに対するトランザクションの動作を指定します。デフォルトは MANUAL です。 文字列 MANUAL を指定した場合は、COBOL ソースプログラムに COMMIT 文または ROLLBACK 文を記述することで、SQL の操作を確定します。 文字列 AUTO を指定した場合は、COBOL ソースプログラムの記述に関係なく、SQL 文ごとにその操作が確定します。 文字列 ENLIST が指定された場合は、Ambient トランザクション(現在参加しているトランザクション)がある場合は、そのトランザクションに参加します。 Ambient トランザクションがない場合は、AUTO 指定と同様の動きをします。

```
                        ì MANUAL  ü
@SQL_COMMIT_MODE=í AUTO    ý
                        î ENLIST  þ
```

- COMMIT 文または ROLLBACK 文によって、オープン中のカーソルがクローズする場合があります。COMMIT 文または ROLLBACK 文の実行によるカーソルの状態は、データベースのドキュメントを参照してください。
- AUTO を指定した場合は、SQL 文を実行した時点でデータベースに、その処理が反映されるため、ROLLBACK 文でデータベースを元の状態に戻すことができません。 MANUAL の指定をおすすめします。ただし、 ADO.NET データプロバイダーまたは、 ODBC ドライバによって、指定に対する動作が異なる場合があります。
- ADO.NET 接続で ENLIST 指定を行う場合、接続文字列の Enlist に True を指定してください。SQL Client Data provider は、True がデフォルトであるため、指定する必要はありません。 使用するデータベースプロバイダの Enlist を確認してください。

@SQL_QUERY_TIMEOUT (タイムアウト時間の指定)

データ ベースに対するクエリーのタイムアウト時間を指定します。

@SQL_QUERY_TIMEOUT の指定範囲と、省略時の動作は以下のとおりです。

データベースの接続方法として ADO.NET が指定されている場合

指定範囲は、0 ～2147483647 です。省略時は、各 ADO.NET データプロバイダーのデフォルト値に従います。

データベースの接続方法として ODBC が指定されている場合

指定範囲は、0 ～4294967285 です。 省略時は、0 とみなして、タイムアウトはしません。 ただし、ODBC ドライバによって、 指定に対する動作が異なる場合があります。

```
@SQL_QUERY_TIMEOUT=タイムアウト時間（秒）
```

NCOBU00892

@SQL_ISOLATION (トランザクション分離レベルの指定)

トランザクション分離レベルを指定します。

@SQL_ISOLATION=í	READ_UNCOMMITTED	ü
	READ_COMMITTED	
	REPEATABLE_READ	ý
	SERIALIZABLE	
	SNAPSHOT	þ

- 指定値に対する動作については、"表: @SQL_ISOLATION の指定値に対する動作説明"を参照してください。 ただし、ADO.NET データプロバイダー、ODBC ドライバまたは、データベースによって指定に対する動作が異なる場合があります。 関連するドキュメントを参照してください。
- データベースによっては、指定するトランザクション分離レベルがサポートされていない場合があります。使用するデータベースの仕様を確認してください。
- サポートされていないトランザクション分離レベルが指定された場合または、トランザクション分離レベルを指定しなかった場合は、ADO.NET データプロバイダーまたは、ODBC ドライバのデフォルトのトランザクション分離レベルでトランザクションが開始されます。
- トランザクション分離レベルの指定は、@SQL_COMMIT_MODE が MANUAL の場合にだけ有効になります。

表: @ SQL_ISOLATION の指定値に対する動作説明

指定値	動作説明
READ_UNCOMMITTED	ダーティ読み出し、繰り返し不可読み出し、およびファントムが発生する可能性があります。
READ_COMMITTED	ダーティ読み出しは発生しません。繰り返し不可読み出しおよびファントムが発生する可能性があります。
REPEATABLE_READ	ダーティ読み出しと繰り返し不可読み出しは発生しません。ファントムが発生する可能性があります。
SERIALIZABLE	トランザクションの直列化が可能です。ダーティ読み出し、繰り返し不可読み出し、およびファントムは発生しません。
SNAPSHOT	Microsoft SQL Server のトランザクション分離レベル SNAPSHOT を使用します。

ダーティ読み出し、繰り返し不可読み出しまたは、ファントムについては、使用するデータベースの仕様書を確認してください。

SNAPSHOT 分離レベルを使用する際の注意事項

- 使用するデータベースが、Microsoft SQL Server 2005 以降のときに使用することができます。
- ODBC 接続の場合、データベースドライバが Microsoft SQL Server Native Client 10.0/11.0 を使用している場合に、このオプションを指定することができます。
- ADO.NET 接続の場合、使用するデータプロバイダーが、System.Data.SqlClient データプロバイダーを使用する必要があります。
- アプリケーション構成ファイルにだけ指定できます。
- SNAPSHOT 分離レベルは、データベースから取り出したデータが、他トランザクションによる変更が反映される前のデータである可能性があります。SNAPSHOT 分離レベルを使用する際には、SNAPSHOT 分離レベルの仕様をよく理解した上でご利用ください。 SNAPSHOT 分離レベルの詳細は MSDN の"データベース スナップショット"を参照してください。

NCOBU00906

@SQL_MULTIPLE_ROWS (複数行操作のサポートの指定)

FOR 句、複数行指定ホスト変数または、表指定ホスト変数をサポートしていない ODBC ドライバを使用する場合に OFF を指定します。

```
                          ¡OFFü
@SQL_MULTIPLE_ROWS=í      ý
                          îON þ
```

データベースの接続方法として ODBC が指定されている場合にだけ有効になります。

@SQL_ODBC_CURSORS (ODBC カーソルライブラリの指定)

ODBC カーソルライブラリは、通常 ODBC ドライバが行うカーソル処理を代替する機能を有します。 ODBC カーソルライブラリを使用することにより、 ODBC ドライバが位置付け UPDATE、位置付け DELETE をサポートしない場合でも、これらを実現することが可能になります。 デフォルトは USE_DRIVER です。 文字列 USE_DRIVER を指定した場合は、ODBC カーソルライブラリを使用しません。 文字列 USE_ODBC を指定した場合は、ODBC カーソルライブラリを使用します。

```
                        ì USE_DRIVER  ü
@SQL_ODBC_CURSORS=í                   ý
                        î USE_ODBC    þ
```

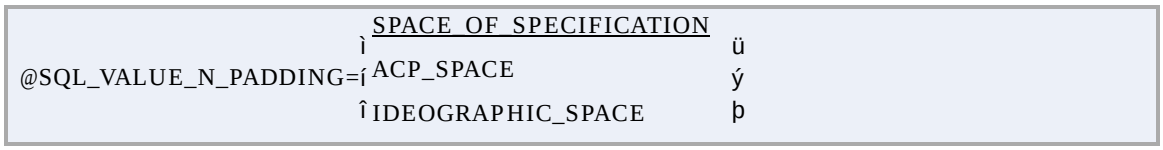
ODBC カーソルライブラリを使用する場合は、以下の注意事項があります。

- UPDATE 文(位置付け)または DELETE 文(位置付け)を使用する場合は、カーソル宣言で値が一意な列を必ず 1 つ以上選択してください。
- ODBC カーソルライブラリは、UPDATE 文(位置付け)および DELETE 文(位置付け)をそれぞれ UPDATE 文(探索)および DELETE 文(探索)にシミュレートして実行します。そのため、一意な値を持つ列が選択されていない場合、複数行に処理が影響する場合があります。
- ODBC カーソルライブラリを使用しない場合と比較して、実行性能が劣化する場合があります。
- カーソルの同時実行(@SQL_CONCURRENCY)に"VALUES"を指定しなければなりません。
- データベースの接続方法として ODBC が指定されている場合にだけ有効になります。

NCOBU00691

@SQL_VALUE_N_PADDING (日本語項目を渡す場合の後続空白種別の指定)

日本語項目のホスト変数が持つ値を、データベースに渡す場合の後続空白種別を指定します。@SQL_VALUE_N_PADDING が指定されない場合、デフォルトは SPACE_OF_SPECIFICATION です。SPACE_OF_SPECIFICATION を指定した場合は、COBOL システムの仕様に従った後続空白を入れます。ACP_SPACE を指定した場合は、後続空白として半角空白を入れます。 IDEOGRAPHIC_SPACE を指定した場合は、後続空白として全角空白を入れます。



@SQL_TARGET_N_PADDING (日本語項目受け取り時の後続空白種別の指定)

データベースから、日本語項目のホスト変数に値を受け取る場合の後続空白種別を指定します。@SQL_TARGET_N_PADDING が指定されない場合、デフォルトは SPACE_OF_SPECIFICATION です。SPACE_OF_SPECIFICATION を指定した場合は、データベースや COBOL システムの仕様に従った後続空白が入ります。 ACP_SPACE を指定した場合は、後続空白として半角空白が入ります。 IDEOGRAPHIC_SPACE を指定した場合は、後続空白として全角空白が入ります。

```
                          ì SPACE_OF_SPECIFICATION ü
@SQL_TARGET_N_PADDING=í   ACP_SPACE                ý
                          î IDEOGRAPHIC_SPACE      þ
```

NCOBU00893

@SQL_ADONET_CURSOR_DEFAULT(カーソル動作省略時のカーソル動作の指定)

データベースの接続方法が ADO.NET の場合に、FOR UPDATE 句が指定されていないカーソルが、読み取り専用カーソルか、更新可能カーソルかを指定します。

```
                                    ì READ_ONLY  ü
@SQL_ADONET_CURSOR_DEFAULT=í                     ý
                                    î READ_WRITE þ
```

表: @SQL_ADONET_CURSOR_DEFAULT の指定値に対する動作説明

指定値	動作説明
READ_ONLY	FOR UPDATE 句の指定がないカーソルを、読み取り専用カーソルとして扱います。
READ_WRITE	FOR UPDATE 句の指定がないカーソルを、更新可能カーソルとして扱います。

- ソースコード上のカーソル宣言で FOR UPDATE 句が指定されていない場合に有効です。@SQL_ADONET_FORUPDATE_CLAUSE に DISABLE を設定したことによって、FOR UPDATE 句が無効化されたカーソルの動作には影響は与えません。
- データベースの接続方法として ADO.NET が指定されている場合にだけ有効になります。
- FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文を実行する場合は、READ_WRITE を指定してください。 読み取り専用カーソルでは順方向専用カーソルとなり、FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文を実行することができません。

@SQL_ADONET_FORUPDATE_CLAUSE (カーソル宣言の FOR UPDATE 句の無効化の指定)

データベースの接続方法が ADO.NET の場合に、ソースコード上に記述されたカーソル宣言の SELECT 文で指定された文末の FOR UPDATE 句を無効にするかどうか指定します。

```
                                        ì ENABLE ü
@SQL_ADONET_FORUPDATE_CLAUSE=í           ý
                                        î DISABLEþ
```

表: @SQL_ADONET_FORUPDATE_CLAUSE の指定値に対する動作説明

指定値	動作説明
ENABLE	ソースコード上に記述されたカーソル宣言の文末の FOR UPDATE 句を有効にします。
DISABLE	ソースコード上に記述されたカーソル宣言の文末の FOR UPDATE 句を無効にします。

- DISABLE を指定した場合でも、カーソルは更新カーソルとして動作します。
- データベースの接続方法として ADO.NET が指定されている場合にだけ有効になります。
- SQL Server に接続する場合は、カーソル宣言に FOR UPDATE 句を指定することができません。このオプションを DISABLE に指定することで、文末の FOR UPDATE 句を無効にすることができます。

NCOBU00894

@SQL_ODBC_CURSOR_DEFAULT(カーソル動作省略時のSQL オプション情報の指定)

データベースの接続方法が ODBC の場合、FOR UPDATE 句の指定がないカーソルに設定する SQL オプション情報に関連付けされたオプション名を指定します。オプション名は、SQL オプション情報の<option>要素の name 属性に指定された値です。<option>要素については、SQL 情報設定のためのアプリケーション構成ファイルの形式を参照してください。

```
@SQL_ODBC_CURSOR_DEFAULT=オプション名
```

- アプリケーション構成ファイルにだけ指定できます。
- データベースの接続方法として ODBC が指定されている場合にだけ有効になります。
- @SQL_ODBC_CURSOR_DEFAULT が指定された場合、FOR UPDATE 句が指定されていないカーソルで、オプション名が示す SQL オプション情報が有効になります。

@SQL_ODBC_CURSOR_UPDATE(更新カーソルの SQL オプション情の指定)

データベースの接続方法が ODBC の場合、FOR UPDATE 句の指定があるカーソルに設定する SQL オプション情報に関連付けされたオプション名を指定します。オプション名は、SQL オプション情報の<option>要素の name 属性に指定された値です。<option>要素については、SQL 情報設定のためのアプリケーション構成ファイルの形式を参照してください。

```
@SQL_ODBC_CURSOR_UPDATE=オプション名
```

- アプリケーション構成ファイルにだけ指定できます。
- データベースの接続方法として ODBC が指定されている場合にだけ有効になります。
- @SQL_ODBC_CURSOR_UPDATE が指定された場合、FOR UPDATE 句が指定されているカーソルで、オプション名が示す SQL オプション情報が有効になります。

NCOBU00896

@SQL_ODBC_MARS (ODBC MARS 機能の指定)

データベースの接続方法が ODBC の場合に、SQL Server の MARS 機能を使用するかどうか指定します。

```
                    ì OFFü
@SQL_ODBC_MARS=í      ý
                    î ON þ
```

表: @SQL_ODBC_MARS の指定値に対する動作説明

指定値	動作説明
OFF	MARS 機能を有効にしません。
ON	MARS 機能を有効にします。

- データベースの接続方法として ODBC が指定されている場合にだけ有効になります。
- MARS 機能は、SQL Server Native Client 10.0/11.0 ODBC ドライバを使用する場合にだけ使用することができます。

- MARS 機能については、SQL Server オンラインブックの複数のアクティブな結果セット (MARS)を参照してください。

@SQL_AUTO_CURSOR_CLOSE (トランザクションによるカーソルの振る舞い指定)

トランザクションによるカーソルの振る舞いを指定します。

```
                                  DRIVER_DEFAULT
                                ì                ü
@SQL_AUTO_CURSOR_CLOSE=í        CLOSE            ý
                                î HOLD           þ
```

データベースの接続方法が ODBC の場合に、ROLLBACK 文または COMMIT 文の実行によるトランザクションの確定時に、該当するコネクションでオープンしているカーソルを自動的にクローズするかどうかを指定します。

@SQL_AUTO_CURSOR_CLOSE の指定値に対する動作説明

指定値	動作説明
DRIVER_DEFAULT	ROLLBACK 文または COMMIT 文実行時に該当するカーソルをクローズするかホールドするかはドライバの規定の動作に従います。
CLOSE	ROLLBACK 文または COMMIT 文実行時に該当するコネクションでオープンしているカーソルをクローズします。
HOLD	ROLLBACK 文または COMMIT 文実行時に該当するコネクションでオープンしているカーソルをホールドします。

- データベースに Microsoft SQL Server を使用している、かつ、データベースドライバが Microsoft SQL Server Native Client 10.0/11.0 を使用している場合にこのオプションを指定することができます。
- アプリケーション構成ファイルにだけ指定できます。
- データベースの接続方法が ODBC 接続の場合にだけ有効になります。
- サーバカーソルを使用しなければなりません。サーバカーソルへの変更は、@SQL_CONCURRENCY (カーソルの同時実行の指定)と@SQL_CURSOR_TYPE (カーソル種別の指定)によって変更できます。デフォルトでは、この値は、@SQL_CONCURRENCY が READ_ONLY で、@SQL_CURSOR_TYPE が FORWARD_ONLY になっているため、既定の結果セットになっています。

NCOBU01076

@SQL_ADONET_DATETIME_FORMAT (日付型の書式指定)

ADO.NET 経由の接続時に、日付型から取得するデータの書式を指定します。

```
@SQL_ADONET_DATETIME_FORMAT={書式指定文字列}
```

データベースの接続方法が ADO.NET 経由の場合に、日付型から取得するデータの書式を指定します。 このオプション省略時には、英数字型ホスト変数に現在のスレッドのカルチャ情報に依存した形式で日付が格納されます。

@SQL_ADONET_DATETIME_FORMAT の指定値に対する動作説明

指定値	動作説明
書式指定文字列	英数字型ホスト変数に指定した書式で日付が格納されます。

書式指定文字列には.NET Framework 標準の日付フォーマットの指定が可能です。.NET Framework 標準の日付フォーマットには形式指定文字の標準パターンおよびカスタムパターンがあります。書式指定文字列に指定する値の例を以下に示します。

標準パターンの出力例

書式指定文字列	取得するデータ	備考
G	2010/01/01 11:12:13	一般（当オプション省略時と同じ） 左記は例であり、格納される形式は現在のスレッドのカルチャ情報に依存します。
D	2010 年 1 月 1 日	LongDatePattern(長い日付の値の形式) 左記は例であり、格納される形式は現在のスレッドのカルチャ情報に依存します。
f	2010 年 1 月 1 日 11:12:13	AM 完全な日付と時刻 (長い日付と短い時刻) 左記は例であり、格納される形式は現在のスレッドのカルチャ情報に依存します。

カスタムパターンの出力例

書式指定文字列	取得するデータ	備考
yyyy/MM/dd	2010/01/01	
yyyy/%M/%d	2010/1/1	
yyyy-MM-dd	2010-01-01	
yyyy-MM-dd hh:MM:ss	2010-01-01 11:12:13	SQLServer の smalldate 型を使用、または、Oracle の DATE 型を使用し、ODBC 経由の接続と同様の結果を得たい場合に指定します。

yyyy-MM-dd hh:MM:ss.fff	2010-01-01 11:12:13.123	SQLServer の datetime 型、datetime2 型を使用し、ODBC 経由の接続と同様の結果を得たい場合に指定します。
yyyy-MM-dd hh:MM:ss.fffffff	2010-01-01 11:12:13.1234567	Oracle の TIMESTAMP 型を使用し、ODBC 経由の接続と同様の結果を得たい場合に指定します。

- 書式体系は、スレッドに関連付けられたカルチャ固有の情報（言語、サブ言語、国/地域、暦、文化的な慣習など）に依存します。
- このオプションは、コネクション単位で有効になります。
- このオプションは、取得対象データが System.DateTime 型にマッピングされ、使用するホスト変数が英数字項目の場合のみ有効になります。
- このオプションは、データベースの接続方法が ADO.NET 接続の場合にだけ有効になります。
- 書式指定文字列に指定可能な値については、MSDN ライブラリの"標準の日付と時刻の書式指定文字列"および"カスタムの日付と時刻の書式指定文字列"を参照してください。
- 指定した書式指定文字列が不正の場合は SQLSTATE=999SM のエラーが発生する場合があります。

NCOBU01091

@SQL_SPACE_TRUNC (ホスト変数の入力値の後置空白の無効化)

英数字項目または日本語項目の入力値のホスト変数の後置空白を無効にするかどうかを指定します。

```
                      ìNO ü
@SQL_SPACE_TRUNC=í    ý
                      îYESþ
```

例えば、以下のようなデータの場合

```
01 HOST-VAL PIC X(5) VALUE "ABC" .
```

変数 HOST-VAL には、"ABC__" (*1) が設定されますが、このオプションに YES を指定することで、 後置空白が削除され、データベースには"ABC"が通知されます。

*1: "_" は空白を表しています。

- このオプションは英数字項目および日本語項目だけに適用され、各データ項目に対応した後置空白が削除されます。
- ホスト変数の内容が全て空白の場合、1 字の空白データとして扱います。
- アプリケーション構成ファイルにだけ指定できます。
- @SQL_VALUE_N_PADDING(日本語項目を渡す場合の後続空白種別の指定) と一緒に指定できません。

@SQL_CODE_MAPPING　(文字データのマッピングコードを指定)

文字データのマッピングコードを指定します。

```
                          ì RCS       ü
@SQL_CODE_MAPPING =       í           ý
                          î UNICODE   þ
```

このオプションは、翻訳オプション RCS(SJIS)および RCS(SJIS-UCS2)で翻訳されたオブジェクトで、 Azure SQL データベースを使用する場合、SJIS のデータ型で ASCII 範囲以外の文字を扱う時に指定します。 このオプションに UNICODE を指定することにより、翻訳オプション RCS(UTF8-UCS2)が指定された場合と同じデータ形式を使用することができます。

ADO.NET または ODBC で扱う文字データと COBOL で扱う文字データの対応は、文字データの対応表（ADO.NET）または文字データの対応表(ODBC)を参照してください。

- このオプションはアプリケーション構成ファイルのみに指定できます。
- CONVERT 関数などで、文字列を文字列以外の型に変換する場合、結果が異なる場合があります。そのような処理を含む場合、このオプションは使用しないでください。

NCOBU01185

@SQL_STOREDPROCEDURE_CACHE（ストアドプロシージャキャッシュ指定）

データベースの接続が ODBC の場合に、ストアドプロシージャ情報をキャッシュするかどうか指定します。 この値を設定すると、同じストアドプロシージャの２回目以降の実行時間が短くなります。

```
                                    ì"ON"  ü
@SQL_STOREDPROCEDURE_CACHE=í       ý
                                    î"OFF" þ
```

例）ストアドプロシージャをキャッシュしたい場合

```
<add key="@SQL_STOREDPROCEDURE_CACHE" value="ON" />
```

[注意]

- デフォルト値は、ON です。この場合、ストアドプロシージャはキャッシュされます。
- アプリケーション構成ファイルにだけ指定できます。
- 複数コネクションで使用する場合は、各々のコネクションに対して設定する必要があります。
- データベースの接続方法が ODBC 接続の場合にだけ有効になります。

@SQL_X_NULL_TERM（英数字項目に含まれるデータの NULL 値の無効化）

英数字項目の出力値のホスト変数の NULL 値を無効にするかどうかを指定します。

```
                        ì"ON" ü
@SQL_X_NULL_TERM=í        ý
                        î"OFF"þ
```

英数字項目に、データベースの CHAR 型がマッピングしている場合、NULL 値は文字終端として扱われます。NetCOBOL においては、文字終端以降は、空白によってパディングします。 NULL 文字以降も空白パディングを行わず、データベースの値をそのままホスト変数に取得する場合は、OFF を指定してください。

[注意]

- ADO.NET 接続の場合は、アプリケーションサーバとデータベースサーバの文字コードを合わせてください。
- ADO.NET接続の場合は､アプリケーションサーバとデータベースサーバの文字コードが異なる場合に、正しく文字列が取得できない可能性があります｡その場合は､以下のいずれかにより回避してください。
  - COBOLプログラムで､CONVERT関数を使用し､BINARY型としてデータを取得するように、COBOL プログラムを修正してください。 SQL Server で、TABLE の COL1 が CHAR 型の場合に NULL 値を含めてデータを取得する例を示します。

```
修正前：SELECT COL1 INTO :X1 FROM TABLE
修正後：SELECT CONVERT(BINARY,COL1) INTO :X1 FROM TABLE
```

  - データベースのデータ型を CHAR 型ではなく、BINARY 型に変更して、データベースを再構築してください。

NCOBU01183

@SQL_DATAPROVIDER_EXTENSION_DLL（データプロバイダー拡張ライブラリの DLL 名指定）

データプロバイダー拡張ライブラリの DLL 名を指定します。

```
@SQL_DATAPROVIDER_EXTENSION_DLL=Data Provider クラスライブリ DLL
```

[注意]

- 絶対パスで指定されていない場合、アプリケーションドメインからの相対パスとして検索されます。
- @SQL_DATAPROVIDER_EXTENSION_DLL に値を指定しない場合、 “データプロバイダー拡張ライブラリの検索順序”に従い、 次の検索が行われます。
- @SQL_DATAPROVIDER_EXTENSION_DLL に指定した DLL が存在しない、または､検索された DLL が “データプロバイダー拡張ライブラリの作成手順” に従って作成されていない場合、Data Provider Extension 機能が利用できず、以下のエラーになります。

```
JMP0372I-U SQL 文を実行するための環境開設でエラーが発生しました. 'DataProvider Extension[プロバイダ名]'
```

デフォルトコネクション情報

デフォルトコネクション情報をアプリケーション構成ファイルに設定する場合は、<sqlDefaultInf>要素内に定義します。アプリケーション構成ファイルの形式については、SQL 情報設定のためのアプリケーション構成ファイルの形式を参照してください。

ODBC 情報ファイルに設定する場合は、[SQL_DEFAULT_INF]セクションに定義します。ODBC 情報ファイルの形式については、ODBC 情報ファイルの形式を参照してください。

デフォルトコネクション情報に関する情報の定義内容を、以下の表に示します。

サーバ情報の定義内容

情報名	設定内容	接続方法	
		ADO.NET	ODBC
@SQL_SERVER	サーバ名	○	○
@SQL_USERID	ユーザ ID	○	○
@SQL_PASSWORD	パスワード	○	○

ユーザ ID およびパスワードは、両方指定してください。 ユーザ ID またはパスワードのどちらか一方を省略することはできません。

NCOBU00694

@SQL_SERVER (サーバ名の指定)

デフォルトコネクションを確立する対象のサーバ名を指定します。 このサーバ名をもとにしてサーバごとの定義情報を検索し、サーバごとのデータソース名に対してコネクションを確立します。 したがって、サーバごとの定義情報を記述しておく必要があります。

```
@SQL_SERVER=サーバ名
```

- 設定する値の最大長は、コネクションを確立するデータソースの仕様により異なります。 したがって、ADO.NET データプロバイダーまたは、 ODBC ドライバと関係する環境のマニュアルを参照してください。

@SQL_USERID (ユーザ ID の指定)

デフォルトコネクションサーバに対応するデータソースを操作するためのユーザ ID を指定します。

ADO.NET 接続で@SQL_USERID が指定された場合、接続文字列で指定されたユーザ ID よりも、@SQL_USERID で指定したユーザ ID が有効になります。

```
@SQL_USERID=ユーザ ID
```

- 接続文字列や ODBC データソースで、Windows 認証(注)を設定している場合は、@SQL_USERID にはユーザ ID を指定しないでください。 ユーザ ID が指定されている場合、ADO.NET データプロバイダーや ODBC ドライバによっては、Windows 認証が行われないことがあります。

  (注)「Windows 認証」はデータベースによって呼び方が異なります。「Windows 統合セキュリティ」や「信頼済み接続」などと呼ばれている場合があります。

- 設定する値の最大長は、コネクションを確立するデータソースの仕様により異なります。 したがって、ODBC ドライバと関係する環境のマニュアルを参照してください。

NCOBU00696

@SQL_PASSWORD (パスワードの指定)

デフォルトコネクションサーバに対応するデータソースを操作するためのパスワードを指定します。

ADO.NET 接続で@SQL_PASSWORD が指定された場合、接続文字列で指定されたパスワードよりも、@SQL_PASSWORD で指定したパスワードが有効になります。

```
@SQL_PASSWORD=パスワード
```

- 接続文字列や ODBC データソースで、Windows 認証(注)を設定している場合は、@SQL_PASSWORD にはパスワードを指定しないでください。 パスワードが指定されている場合、ADO.NET データプロバイダーや ODBC ドライバによっては、Windows 認証が行われないことがあります。

  (注)「Windows 認証」はデータベースによって呼び方が異なります。「Windows 統合セキュリティ」や「信頼済み接続」などと呼ばれている場合があります。

- このパスワードは、実行環境設定ユーティリティまたは ODBC 情報設定ユーティリティを使用して暗号化してください。
- 設定する値の最大長は、コネクションを確立するデータソースの仕様により異なります。 したがって、ODBC ドライバと関係する環境のマニュアルを参照してください。

コネクション有効範囲

コネクション有効範囲をアプリケーション構成ファイルに設定する場合は、<connectionScope>要素内に定義します。アプリケーション構成ファイルの形式については、SQL 情報設定のためのアプリケーション構成ファイルの形式を参照してください。

ODBC 情報ファイルに設定する場合は、[CONNECTION_SCOPE]セクションに定義します。ODBC 情報ファイルの形式については、ODBC 情報ファイルの形式を参照してください。

コネクション有効範囲の定義内容を、以下の表に示します。

コネクション有効範囲の定義内容

情報名	設定内容	接続方法	
		ADO.NET	ODBC
@SQL_CONNECTION_SCOPE	コネクション有効範囲	○	○

NCOBU00698

@SQL_CONNECTION_SCOPE (コネクションの有効範囲の指定)

コネクションの有効範囲を指定します。 デフォルトは APPLICATION_DOMAIN です。 コネクションの有効範囲とは、接続したコネクションの利用可能範囲です。

```
                              ì APPLICATION_DOMAIN  ü
@SQL_CONNECTION_SCOPE=í THREAD                  ý
                              î OBJECT_INSTANCE     þ
```

指定値に対する動作については、"表:コネクション有効範囲の指定値に対する動作説明"を参照してください。 また、有効な利用方法については、"コネクションをオブジェクトインスタンス単位で利用する"を参照してください。

表: コネクション有効範囲の指定値に対する動作説明

指定値	動作説明
APPLICATION_DOMAIN	接続したコネクションは、実行環境内(APPLICATION_DOMAIN)で利用可能です。通常、シングルスレッドプログラムを動作させる場合に指定します。マルチスレッドプログラムを動作させる場合は、(注 3)を参照してください。
THREAD	接続したコネクションは、実行単位内（スレッド内）で利用可能です。 通常、シングルスレッドプログラムをマルチスレッドプログラムに移行する場合に指定します。 実行単位内で複数のコネクションを接続した場合、最後に実行された CONNECT 文または SET CONNECTION 文で指定されたコネクションが、実行単位の現コネクションになります。(注 1)
OBJECT_INSTANCE	接続したコネクションは、オブジェクトインスタンス内で利用可能です。通常、オブジェクト指向機能を利用したシングルスレッドプログラムをマルチスレッドプログラムに移行する場合に指定します。オブジェクトインスタンス内で複数のコネクションを接続した場合、最後に実行された CONNECT 文または SET CONNECTION 文で指定されたコネクションが、オブジェクトインスタンスの現コネクションになります。(注 2)

注 1:

クラス定義(オブジェクト指向プログラミング機能)に記述した埋込み SQL 文は動作しません。

注 2:

プログラム定義に記述した埋込み SQL 文は動作しません。

注 3:

マルチスレッドプログラムを動作させる場合の注意事項

- 複数のマルチスレッドプログラムが 1 つのコネクションを共有して利用する場合、いずれかのマルチスレッドプログラムでトランザクション処理を行ったとき、コネクションを共有するすべてのマルチスレッドプログラムのデータ操作に影響します。
- 複数のマルチスレッドプログラムが複数のコネクションを利用する場合、各マルチスレッドプログラムで最初に実行される埋込み SQL 文は CONNECT 文または SET CONNECTION 文でなければなりません。 CONNECT 文または SET CONNECTION 文以外の埋込み SQL 文を実行した場合は、どのコネクションに対して操作をするか保証できません。 これは、各マルチスレッドプログラムがそれぞれの現コネクションを利用するためです。 なお、複数のマルチスレッドプログラムが 1 つのコネクションを共有して利用する場合は必要ありません。
- カーソルを使用したマルチスレッドプログラムが複数のスレッドで実行された場合、それぞれのスレッドが自分自身のカーソルを持ちます。 したがって、カーソルをスレッド間で共有することはできません。

NCOBU00897

SQL オプション情報

SQL オプション情報には、カーソル単位の SQL 情報を指定します。

オプションをアプリケーション構成ファイルに設定する場合は、<sqlOptionInf>要素内に定義します。アプリケーション構成ファイルの形式については、SQL 情報設定のためのアプリケーション構成ファイルの形式を参照してください。

SQL オプション情報の定義内容を、以下の表に示します。

表: SQL オプション情報の定義内容

情報名	設定内容	接続情報	
		ADO.NET	ODBC
@SQL_CONCURRENCY	カーソルの同時実行	×	○
@SQL_ROW_ARRAY_SIZE	メモリにキャッシュする行数	×	○
@SQL_CURSOR_TYPE	カーソル種別	×	○

- アプリケーション構成ファイルにだけ指定できます。
- データベースの接続方法が ODBC 接続の場合にだけ有効になります。

@SQL_CONCURRENCY (カーソルの同時実行の指定)

カーソルの同時実行を指定します。 同時実行とは、複数のクライアント（複数のコネクションでも同様）が、同一データを同時に使用する機能を指します。 @SQL_CONCURRENCY が指定されない場合、デフォルトは READ_ONLY です。

```
                          ì READ_ONLY  ü
                          ï LOCK       ï
@SQL_CONCURRENCY=í        ï ROWVER     ý
                          ï            ï
                          î VALUE      þ
```

- この情報をサーバ情報に指定すると、コネクション単位に有効になり、SQL オプション情報に指定すると、カーソル単位に有効になります。サーバ情報と、SQL オプション情報の両方に指定した場合は、SQL オプション情報に指定した値が有効になります。
- 指定値に対する動作については、"表: @SQL_CONCURRENCY の指定値に対する動作説明"を参照してください。 ただし、ODBC ドライバによって指定に対する動作が異なる場合があります。また、ご使用になる前には、"各 ODBC ドライバ固有の留意事項"を参照してください。

表: @SQL_CONCURRENCY の指定値に対する動作説明

指定値	動作説明		
READ_ONLY	カーソル系データ操作文の位置付け・取出し(FETCH 文)が可能です。更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)は実行できません。(注 1)		
LOCK	カーソル系データ操作文の位置付け・取出し(FETCH 文)、更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)が可能です。(注 2)	NO CONCURRENCY	カーソルは最下位レベルのロックを使用して、更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)を行います。同一リソースを更新または削除することを防ぎます。
ROWVER	カーソル系データ操作文の位置付け・取出し(FETCH 文)、更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)が可能です。(注 2)	CONCURRENCY	カーソルは行の変更履歴を使用したオプティミスティック同時実行制御を使用して、更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)を行います。オプティミスティック同時実行では、行が更新または削除されるまでロックされません。したがって、他のユーザが同時に同一リソースを更新または削除した場合、処理が失敗する可能性があります。

VALUE	カーソル系データ操作文の位置付け・取出し(FETCH 文)、更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)が可能です。(注 2)	CONCURRENCY	カーソルは行の値を使用したオプティミスティック同時実行制御を使用して、更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)を行います。

注 1:

一部のデータソースでは、カーソル系データ操作文の更新(位置付け UPDATE 文)、削除(位置付け DELETE 文)が実行可能です。

注 2:

カーソルのロックレベルはデータソースに依存します。

@SQL_ROW_ARRAY_SIZE(メモリにキャッシュする行数の指定)

オープンしたカーソルのデータをメモリにキャッシュする行数を指定します。

このオプションが指定された場合、FETCH 文で取得する行数に関係なく、常に@SQL_ROW_ARRAY_SIZE に指定された行数のデータをカーソルから読み取り、メモリにキャッシュします。FETCH 文の実行は、このキャッシュされたデータから読み取ります。FETCH 文で、キャッシュされたデータをすべて読み取った場合は、再度、@SQL_ROW_ARRAY_SIZE に指定された行数分カーソルからデータを読み取り、メモリにキャッシュします。キャッシュされたデータは、カーソルの CLOSE 文により解放します。

このため、FETCH 文実行時のデータベースへのアクセス回数が減少し、性能向上が期待できます。 ただし、既に、メモリにキャッシュされた後で、変更されたカーソルのデータは、FETCH 文では取得されませんので注意してください。

```
@SQL_ROW_ARRAY_SIZE=行数
```

省略時または 0 や負の値が指定された場合は、行数は 1 とみなします。

- アプリケーション構成ファイルにだけ指定できます。
- データベースの接続方法が ODBC 接続の場合にだけ有効になります。
- 実際に取得する行数よりも小さい値を指定してください。大きい値を指定した場合、 使用しないメモリ領域を獲得することになり、性能が低下する場合があります。
- 行数には、1～2147483647 を指定することができます。しかし、指定された行数分のデータベース情報を メモリ上にキャッシュすることになるため、使用しているシステム環境または、ODBC ドライバの環境の制限を 受けますので注意してください。
- アプリケーション上に複数のカーソルを実装している場合、カーソル毎にデータベースのデータの 情報がメモリ上に確保されます。アプリケーション中のカーソル数と、取得しようとするデータが使用している システム環境のメモリ制限内になるように指定してください。カーソル毎につき、以下のメモリ領域が確保されます。

表 1：カーソル毎に使用するメモリ領域サイズ

データサイズ	(第 1 カラム長(注 1) + 第 2 カラム長 + … + 最終カラム長) * 行数
標識変数サイズ	(標識変数のバイト数[4 バイト] * 取得するテーブルカラム数) * 行数
管理領域サイズ	(行数+1) * 2

注 1: カラム長 = 取得するテーブル列のデータのバイト数+ 8 バイト境界補正

- 行数を指定する場合は、表 1 のデータサイズが指定できる範囲(1～2147483647)になるように指定してください。

- FETCH 文により複数行の取得した行(注 2)に対して、カーソルの UPDATE 文(位置付け)または、DELETE 文(位置付け)は、サポートしていません。UPDATE 文(位置付け)または、DELETE 文(位置付け)は、FETCH 文により単一行を取得して実行してください。

  注 2: 下記を指定した FETCH 文で取得した行を示します。

  - FOR 句による実行回数指定
  - 複数行指定ホスト変数の行数指定
  - 表指定ホスト変数の行数指定

- FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文を使用する場合、行数には 1 を指定するか、当オプションを指定しないでください。

@SQL_CURSOR_TYPE (カーソル種別の指定)

カーソル種別を指定します。

```
                        ì FORWARD_ONLY  ü
                        ï STATIC        ï
@SQL_CURSOR_TYPE=í      ï KEYSET_DRIVEN ý
                        ï               ï
                        î DYNAMIC       þ
```

カーソル種別は、性能に大きく影響します。読み取り専用のカーソルの場合は、FORWARD_ONLY を指定すると性能が向上します。

指定値に対する動作については、"表: @SQL_CURSOR_TYPE の指定値に対する動作説明"を参照してください。 ただし、ODBC ドライバによって指定に対する動作が異なる場合があります。 また、ご使用になる前には、"各 ODBC ドライバ固有の留意事項"を参照してください。

表: @SQL_CURSOR_TYPE の指定値に対する動作説明

指定値	動作説明
FORWARD_ONLY	カーソルを順方向専用カーソルでオープンします。
STATIC	カーソルを静的カーソルでオープンします。
KEYSET_DRIVEN	カーソルをキーセットドリブンカーソルでオープンします。
DYNAMIC	カーソルを動的カーソルでオープンします。

カーソル種別は、@SQL_CONCURRENCY の値に影響します。関連するドライバを参照してください。

- FORWARD_ONLY 指定では、位置付け UPDATE 文、位置付け DELETE 文、FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文は実行できません。
- カーソル種別は、データソース（ODBC ドライバ、データベース、データベース関連製品）に依存します。使用するデータベースに該当するカーソルタイプがあるか確認してください。
- アプリケーション構成ファイルにだけ指定できます。
- データベースの接続方法が ODBC 接続の場合にだけ有効になります。

NCOBU00699

埋込み SQL 文

埋込み SQL 文に関連する情報について説明します。

このトピックの内容

埋込み SQL 文で使用可能なホスト変数

埋込み SQL 文に指定できるホスト変数の形式を以下に示します。

○ :指定可

× :指定不可

埋込み SQL 文		１行ずつ操作		複数行を同時に操作	
		単一列指定ホスト変数	複数列指定ホスト変数	複数行指定ホスト変数	表指定ホスト変数
非カーソル系データ操作文					
	SELECT 文	○	○	○	○
	DELETE 文(探索)	○	×	○	×
	INSERT 文	○	○	○	○
	UPDATE 文(探索)	○	×	○	×
カーソル系データ操作文					
	OPEN 文	○	×	×	×
	CLOSE 文	×	×	×	×
	FETCH 文	○	○	○	○
	DELETE 文(位置付け)	×	×	×	×
	UPDATE 文(位置付け)	×	×	×	×
動的 SQL 文					
	PREPARE 文	○	×	×	×
	EXECUTE 文	○	○	○	○
	EXECUTE IMMEDIATE 文	○	×	×	×
	動的 SELECT 文	×	×	×	×
	動的カーソル宣言	×	×	×	×

	動的 OPEN 文	○	×	×	×
	動的 CLOSE 文	×	×	×	×
	動的 FETCH 文	○	○	○	○
	動的 DELETE 文(位置付け)	×	×	×	×
	動的 UPDATE 文(位置付け)	×	×	×	×
セッション制御文					
	COMMIT 文	×	×	×	×
	ROLLBACK 文	×	×	×	×
コネクション制御文					
	CONNECT 文	○	×	×	×
	SET CONNECTION 文	○	×	×	×
	DISCONNECT 文	○	×	×	×
その他					
	CALL 文	○	×	×	×
	FOR 句	○	×	×	×

埋込み SQL 文のキーワード一覧

埋込み SQL 文のキーワード一覧を以下に示します。

[A]			
ABSOLUTE	ADA	ADD	ALL
ALLOCATE	ALTER	AND	ANY
ARE	AS	ASC	ASSERTION
AT	AUTHORIZATION	AVG	
[B]			
BEGIN	BETWEEN	BIND	BIT
BIT_LENGTH	BY		
[C]			
CALL	CASCADE	CASCADED	CASE
CAST	CATALOG	CHAR	CHAR_LENGTH
CHARACTER	CHARACTER_LENGTH	CHARACTER_SET_CATALOG	CHARACTER_SET_NAME
CHARACTER_SET_SCHEMA	CHECK	CLOSE	COALESCE
COBOL	COLLATE	COLLATION	COLLATION_CATALOG
COLLATION_NAME	COLLATION_SCHEMA	COLUMN	COMMIT
CONNECT	CONNECTION	CONSTRAINT	CONSTRAINTS
CONTINUE	CONVERT	CORRESPONDING	COUNT
CREATE	CURRENT	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP_CURSOR			
[D]			
DATA	DATE	DATETIME_INTERVAL_CODE	DATETIME_INTERVAL_PRECISION
DAY	DEALLOCATE	DEC	DECIMAL

DECLARE	DEFAULT	DEFERRABLE	DEFERRED
DELETE	DESC	DESCRIBE	DESCRIPTOR
DIAGNOSTICS	DICTIONARY	DISCONNECT	DISPLACEMENT
DISTINCT	DOMAIN	DOUBLE	DROP
[E]			
ELSE	END	END-EXEC	ESCAPE
EXCEPT	EXCEPTION	EXEC	EXECUTE
EXISTS	EXTERNAL	EXTRACT	
[F]			
FALSE	FETCH	FIRST	FLOAT
FOR	FOREIGN	FORTRAN	FOUND
FROM	FULL		
[G]			
GET	GLOBAL	GO	GOTO
GRANT	GROUP		
[H]			
HAVING	HOUR		
[I]			
IDENTITY	IGNORE	IMMEDIATE	IN
INCLUDE	INDEX	INDICATOR	INITIALLY
INNER	INPUT	INSENSITIVE	INSERT
INTEGER	INTERSECT	INTERVAL	INTO
IS	ISOLATION		
[J]			
JOIN			
[K]			
KEY			

[L]			
LANGUAGE	LAST	LEFT	LENGTH
LEVEL	LIKE	LIST	LOCAL
LOWER			
[M]			
MATCH	MAX	MIN	MINUTE
MODULE	MONTH	MUMPS	
[N]			
NAME	NAMES	NATIONAL	NCHAR
NEXT	NONE	NOT	NULL
NULLABLE	NULLIF	NUMERIC	
[O]			
OCTET_LENGTH	OF	OFF	ON
ONLY	OPEN	OPTION	OR
ORDER	OUTER	OUTPUT	OVERLAPS
[P]			
PARTIAL	PASCAL	PLI	POSITION
PRECISION	PREPARE	PRESERVE	PREVIOUS
PRIMARY	PRIOR	PRIVILEGES	PROCEDURE
PUBLIC			
[R]			
RELATIVE	RESTRICT	REVOKE	RIGHT
ROLLBACK	ROWS		
[S]			
SCALE	SCHEMA	SCROLL	SECOND
SECTION	SELECT	SEQUENCE	SET
SIZE	SMALLINT	SOME	SQL

SQLCA	SQLCODE	SQLERARY	SQLERRD
SQLERROR	SQLSTATE	SQLWARNING	START
SUBSTRING	SUM	SYSTEM	
[T]			
TABLE	TEMPORARY	THEN	TIME
TIMESTAMP	TIMEZONE_HOUR	TIMEZONE_MINUTE	TO
TRANSACTION	TRANSLATE	TRANSLATION	TRUE
TYPE			
[U]			
UNION	UNIQUE	UNKNOWN	UPDATE
UPPER	USAGE	USER	USING
[V]			
VALUE	VALUES	VARCHAR	VARIABLES
VARYING	VIEW		
[W]			
WHEN	WHENEVER	WHERE	WITH
WORK			
[Y]			
YEAR			

SQLSTATE/SQLCODE/SQLMSG

ここでは、埋込み SQL 文を実行した時に通知される情報について説明します。

以下に、SQLSTATE、SQLCODE および SQLMSG に通知される情報を示します。

通知情報	通知される値	通知の意味	プログラマの処理
SQLSTATE	00000	正常終了	SQLCODE に付加情報が通知されることがあります。（注）
	表示可能な英数字５桁の組合せ	SQL 文の実行時に、何らかのエラーまたは警告などの付加情報が発生しました。	SQLCODE、SQLMSG の内容を参照し、エラーの場合は処置を行ってください。SQLSTATE の詳細情報については、Microsoft(R)社の ODBC SDK の資料または使用している ODBC の環境(ODBC ドライバ、DBMS など)のマニュアルを参照してください。
SQLCODE	0	正常終了	――――――――――――――
	０以外の正または負の整数値	SQL 文の実行時に、何らかのエラーまたは警告などの付加情報が発生しました。	使用している ODBC の環境のマニュアルから、原因を調査して対処してください。
SQLMSG	空白（出力なし）	正常終了	――――――――――――――
	メッセージ文字列	SQL 文実行時エラー、または警告など付加情報の内容が通知されます。	設定されたメッセージ文字列から原因を調査し、対処してください。

注:

SQLSTATE の値が 00000 かつ SQLCODE の値が 0 の場合が正常終了です。SQLSTATE の値が 00000 でも、SQLCODE に付加情報が通知されている場合は、データの内容は保証されません。

次に、埋込み SQL 文の実行時に COBOL が検出するエラーについて説明します。

以下は、SQLSTATE、SQLCODE、SQLMSG に通知される情報です。 表中の接続方法は、エラーが発生する可能性がある接続を示しています。

SQLSTATE	SQLCODE	SQLMSG	プログラマの処理	接続方法	
				ADO.NET	ODBC
99999	-999999999	同じコネクション名で接続しました	同じコネクション名を持つ複数のコネクションの指定は許されません。それぞれコネクション名が一意になるように修正してください。	○	○
9999A	-999999990	コネクションの最大設定数を超えました	COBOL システムで許されるコネクションの最大数を超えています。 コネクション数を減らす対処を行ってください。ただし、コネクションの最大数は、ODBC の環境により異なることがあります。ODBC の環境のマニュアルを参照して対処してください。	○	○
9999B	-999999800	指定したコネクションは存在しません	コネクションがアクティブになっていないため、SQL 文が実行できない状態にあります。 プログラムの SQL 文の順序を調査し対処してください。	○	○
9999D	-999999200	FOR 句に指定した繰り返し回数が大きすぎます	FOR 句に指定する繰り返し回数を OCCURS 句の繰り返し以下に修正してください。	○	○
9999E	-999999100	FOR 句に指定した値に誤りがあります	FOR 句に指定する繰り返し回数を 1 以上の値に修正してください。	○	○
9999F	-999999993 または、例外コード	SQL 文の実行時に例外が発生しました。’$1’’$2’	SQL 文の実行時に例外が発生しました。例外情報に従って対処してください。 $1: カーソル名(カーソル使用時のみ)$2: 例外情報 例外コードは、各 ADO.NET データプロバイダーに依存します。注 1 を参照してください。 発生する例外に関係する注意事項は、埋込み SQL 文の実行時の注意事項を参照してください。	○	×
9999G	-999999900	FOR 句、複数行指定ホスト変数または、表指定ホスト変数は使用できません	FOR 句、複数行指定ホスト変数または、表指定ホスト変数は使用できませんので修正してください。	×	○
9999H	-999999980	分散トランザクションの参加に失敗しました’$1’’$2’	分散トランザクションの参加に失敗しました。エラー情報に従って対処してください。 $1: カーソル名(カーソル使用時のみ) $2:	○	○

			エラー情報 エラー情報は、各 ADO.NET データプロバイダーまたは MSDTC に依存します。使用しているデータプロバイダーの仕様を確認してください。または、MSDTC のドキュメントを参照してください。		
999SA	-999999700	カーソルがオープンされていません	カーソルが使用可能状態ではありません。 カーソルを使用している SQL 文の順序を調査し対処してください。	○	○
999SB	-999999600	被準備文が準備されていません	動的 SQL 文のシーケンスを調査し対処してください。	○	○
999SC	-999999500	カーソルはすでにオープンされています	カーソルはすでに使用可能状態になっています。 カーソルを使用している SQL 文の順序を調査し対処してください。	○	○
999SE	-999999000	カーソルの状態が正しくありません	カーソルの状態が正しくありません。以下の原因が考えられます。 ADO.NET 接続の場合 ・ READ ONLY カーソルに対して更新が行われた ・ 更新カーソルに主キーが含まれていない ・ 文識別子名が、動的カーソルの文識別子名と同じ ・ READ ONLY カーソルに対して FETCH PRIOR 文、FETCH FIRST 文、または FETCH LAST 文が実行された ODBC 接続の場合 ・ カーソルタイプが FORWARD_ONLY の場合に、位置付け UPDATE 文、位置付け DELETE 文が実行された	○	○
999SF	-999999001	DELETE 文(位置付け)または UPDATE 文(位置付け)が不正です	位置付け更新の SQL 文が不正です。正しい SQL 文を指定してください。	○	×
999SG	-999999002	更新するカーソル行がありません	カーソルの更新をする前に FETCH 文を実行してください。	○	×
999SH	-999999003	カーソルは変更されるテーブルを含んでいま	位置付け更新の SQL 文が正しくありません。SQL 文を修正して	○	×

		せん	ください。		
999SI	-999999004	可変長の長さ部にホスト変数の長さより大きい値が指定されました. ホスト変数の長さで処理を行います	ホスト変数の可変長の長さ部に正しい値を設定してください。	○	○
999SJ	-999999005	ホスト変数より大きいデータのため、データを切り捨てました	ホスト変数の項目長を正しく設定してください。	○	×
999SK	-999999006	同時実行違反 : 処理予定の 1 レコードのうち 0 件が処理されました	データを更新するためには、カーソルをクローズし、再度カーソルのオープンから処理を行う必要があります。 同時実行の詳細については埋込み SQL 文の実行時の注意事項を参照してください。	○	×
999SL	-999999007	カーソルに処理できないオプションが指定されました	@SQL_ROW_ARRAY_SIZE に 1 より大きい値を設定し、カーソルのデータをメモリにキャッシュしている状態で FETCH PRIOR 文、FETCH FIRST 文、および FETCH LAST 文を実行することはできません。 @SQL_ROW_ARRAY_SIZE に 1 を指定してください。	×	○
999SM	-999999008	日付型の書式指定に誤りがあります	@SQL_ADONET_DATETIME_FORMAT に正しい値を設定してください。	○	×
999SN	-999999009	FETCH の開始行が検索範囲を超えています	FOR 句、複数行指定ホスト変数または表指定ホスト変数により、FETCH PRIOR 文の実行で、先頭行より前が取得対象になりました。データは先頭から取得されます。	○	×
?????	-999999992	不正な処理が発生しました	システムエラーです。 提供元担当者に連絡してください。	○	○

○： この接続では、このエラーが発生する可能性があります。
×： この接続では、このエラーは発生しません。

注 1) 例外コードは、使用している ADO.NET データプロバイダーに依存し、以下の表に示す例外コードが設定されます。

ADO.NET データプロバイダー	例外コード(.NET Framework のドキュメントを参照)
.NET Framework Data Provider for SQL Server	SqlException.Number プロパティ
.NET Framework Data Provider for OLE DB	OleDbError.NativeError プロパティ、または、 OleDbException.ErrorCode プロパティ

.NET Framework Data Provider for ODBC	OdbcError.NativeError プロパティ、または、ExternalException.ErrorCode プロパティ（ExternalException クラスから継承）。
.NET Framework Data Provider for Oracle	OracleException.Code プロパティ

データプロバイダー拡張機能を使用して、上記以外のデータプロバイダーを使用している場合は、使用しているデータプロバイダーの仕様を確認してください。

NCOBU00900

データ型の対応

このセクションの内容

データベースアクセスで扱うデータ対応について説明します。

ADO.NET で扱うデータ型との対応

ADO.NET で扱うデータ型との対応を示します。

ODBC で扱うデータとの対応

ODBC で扱うデータ型との対応を示します。

ADO.NET で扱うデータ型との対応

COBOL では、ADO.NET で扱うデータを対応付けて扱います。ADO.NET 経由のアクセス時に使用する ADO.NET データプロバイダーが SQL データ型をどのように扱うかについては、使用するデータプロバイダーのヘルプおよびマニュアルを参照してください。

このセクションの内容

NCOBU00902

算術データの対応表（ADO.NET）

以下は、ADO.NET で扱う算術データと COBOL で扱う算術データの対応表です。

表: 算術データの対応表

DbType 型		COBOL での表現
２進	Int16	PIC S9(4) BINARY または PIC S9(4) COMP-5
	Int32	PIC S9(9) BINARY または PIC S9(9) COMP-5
	Int64	PIC S9(18) BINARY または PIC S9(18) COMP-5
10 進	Decimal	PIC S9(p) PACKED-DECIMAL または PIC S9(p)V9(q) PACKED-DECIMAL 1 =< p =<15, 1=< q, p+q =<18
	Decimal	ì ü í PIC S9(p) ý í PIC S9(p)V9(q) ý î þ ì ü [SIGN IS í LEADING SEPARATE CHARACTER ý] í TRAILING ý î þ 1 =< p =<15, 1=< q, p+q =<18
内部浮動小数点	Single	COMP-1
	Double	COMP-2

注意

- 対応表に記述されていない対応付けを行った場合、データの内容は保証されません。

文字データの対応表（ADO.NET）

以下は、ADO.NET で扱う文字データと COBOL で扱う文字データの対応表です。

> RCS 翻訳オプションの省略値は、RCS(UTF8-UCS2)です。RCS 翻訳オプションの詳細については、RCS (実行時データのコード系)を参照してください。また、RCS 翻訳オプションの指定による動作の違いについては、Unicode を参照してください。

表: 文字データの対応表(RCS(UTF8-UCS2)翻訳オプションを指定した場合)

DbType 型		COBOL での表現
固定長	StringFixedLength	PIC X(n) (注 1)
		PIC N(n) (注 1)
可変長	String	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注 2) 49 データ名-3 PIC X(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注 2) 49 データ名-3 PIC X(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注 2) 49 データ名-3 PIC N(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注 2) 49 データ名-3 PIC N(n). (注 1) m = 4 または 9

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、 データベースドライバの仕様によって制限されます。例えば、ODBC2.0 を使用している場合は、 以下のような制限があります。 X(n) 1=< n =< 254 N(n) 1=< n =< 127

注 2: 文字データ可変長の長さ部は、文字数を指定します。

表: 文字データの対応表(RCS(SJIS-UCS2)翻訳オプションを指定した場合)

DbType 型		COBOL での表現
固定長	AnsiStringFixedLength	PIC X(n) (注1)
	StringFixedLength	PIC N(n) (注1)
可変長	AnsiString	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注2) 49 データ名-3 PIC X(n). (注1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注2) 49 データ名-3 PIC X(n). (注1) m = 4 または 9
	String	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注2) 49 データ名-3 PIC N(n). (注1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注2) 49 データ名-3 PIC N(n). (注1) m = 4 または 9

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、 データベースドライバの仕様によって制限されます。例えば、ODBC2.0 を使用している場合は、 以下のような制限があります。 X(n) 1=< n =< 254 N(n) 1=< n =< 127

注 2: 文字データ可変長の長さ部は、文字数を指定します。

表: 文字データの対応表(RCS(SJIS)翻訳オプションを指定した場合)

DbType 型		COBOL での表現
固定長	AnsiStringFixedLength	PIC X(n) (注1)
		PIC N(n) (注1)
可変長	AnsiString	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注2) 49 データ名-3 PIC X(n). (注1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注2) 49 データ名-3 PIC X(n). (注1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注2) 49 データ名-3 PIC N(n). (注1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注2) 49 データ名-3 PIC N(n). (注1) m = 4 または 9

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、 データベースドライバの仕様によって制限されます。例えば、ODBC2.0 を使用している場合は、 以下のような制限があります。 X(n) 1=< n =< 254 N(n) 1=< n =< 127

注 2: 文字データ可変長の長さ部は、文字数を指定します。

対応表に記述されていない対応付けを行った場合、データの内容は保証されません。

NCOBU00992

日付データの対応表（ADO.NET）

以下は、ADO.NET で扱う日付データと COBOL で扱う日付データの対応表です。

表: 日付データの対応表

DbType 型		COBOL での表現
日付データ	DateTime	PIC X(n) (注 1)

注 1: 文字数 n は、データプロバイダ、データベースドライバの仕様によって制限されます。

ODBC で扱うデータとの対応

COBOL では、ODBC で扱うデータを対応付けて扱います。 ODBC ドライバが ODBC の SQL データ型をどのように扱うかについては、使用する ODBC ドライバのヘルプおよびマニュアルを参照してください。

このセクションの内容

NCOBU00904

算術データの対応表（ODBC）

以下は、ODBC で扱う算術データと COBOL で扱う算術データの対応表です。

表: 算術データの対応表

ODBC SQL データ型		COBOL での表現
2進	SQL_SMALLINT (SMALLINT)	PIC S9(4) BINARY または PIC S9(4) COMP-5
	SQL_INTEGER (INTEGER)	PIC S9(9) BINARY または PIC S9(9) COMP-5
	SQL_BIGINT (BIGINT)	PIC S9(18) BINARY または PIC S9(18) COMP-5
10進	SQL_DECIMAL (DECIMAL)	PIC S9(p) PACKED-DECIMAL または PIC S9(p)V9(q) PACKED-DECIMAL 1 =< p =<15, 1=< q, p+q =<18
	SQL_NUMERIC (NUMERIC)	ì ü ´PIC S9(p) ý íPIC S9(p)V9(q) î þ ì ü [SIGN IS´LEADING SEPARATE CHARACTER ý] íTRAILING î þ 1 =< p =<15, 1=< q, p+q =<18
内部浮動小数点	SQL_REAL (REAL)	COMP-1
	SQL_DOUBLE (FLOAT)	COMP-2

- 対応表に記述されていない対応付けを行った場合、データの内容は保証されません。

文字データの対応表(ODBC)

以下は、ODBC で扱う文字データと COBOL で扱う文字データの対応表です。

> RCS 翻訳オプションの省略値は、RCS(UTF8-UCS2)です。RCS 翻訳オプションの詳細については、RCS (実行時データのコード系)を参照してください。また、RCS 翻訳オプションの指定による動作の違いについては、Unicode を参照してください。

表: 文字データの対応表(RCS(UTF8-UCS2)翻訳オプションを指定した場合)

ODBC SQL データ型		COBOL での表現
固定長	SQL_WCHAR (WCHAR)	PIC X(n) (注 1)
		PIC N(n) (注 1)
可変長	SQL_WVARCHAR (WVARCHAR)	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注 2) 49 データ名-3 PIC X(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注 2) 49 データ名-3 PIC X(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注 2) 49 データ名-3 PIC N(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注 2) 49 データ名-3 PIC N(n). (注 1) m = 4 または 9

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、 データベースドライバの仕様によって制限されます。例えば、ODBC2.0 を使用している場合は、 以下のような制限があります。 X(n) 1=< n =< 254 N(n) 1=< n =< 127

注 2: 文字データ可変長の長さ部は、文字数を指定します。

表: 文字データの対応表(RCS(SJIS-UCS2)翻訳オプションを指定した場合)

ODBC SQL データ型		COBOL での表現
固定長	SQL_CHAR (CHAR)	PIC X(n) (注 1)
	SQL_WCHAR (WCHAR)	PIC N(n) (注 1)
可変長	SQL_VARCHAR (VARCHAR)	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注 2) 49 データ名-3 PIC X(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注 2) 49 データ名-3 PIC X(n). (注 1) m = 4 または 9
	SQL_WVARCHAR (WVARCHAR)	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注 2) 49 データ名-3 PIC N(n). (注 1) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注 2) 49 データ名-3 PIC N(n). (注 1) m = 4 または 9

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、 データベースドライバの仕様によって制限されます。例えば、ODBC2.0 を使用している場合は、 以下のような制限があります。 X(n) 1=< n =< 254 N(n) 1=< n =< 127

注 2: 文字データ可変長の長さ部は、文字数を指定します。

表: 文字データの対応表(RCS(SJIS)翻訳オプションを指定した場合)

ODBC SQL データ型		COBOL での表現
固定長	SQL_CHAR (CHAR)	PIC X(n) (注１)
		PIC N(n) (注１)
可変長	SQL_VARCHAR (VARCHAR)	01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注２) 49 データ名-3 PIC X(n). (注１) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注２) 49 データ名-3 PIC X(n). (注１) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) COMP-5. (注２) 49 データ名-3 PIC N(n). (注１) m = 4 または 9
		01 データ名-1. 49 データ名-2 PIC S9(m) BINARY. (注２) 49 データ名-3 PIC N(n). (注１) m = 4 または 9

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、 データベースドライバの仕様によって制限されます。例えば、ODBC2.0 を使用している場合は、 以下のような制限があります。 X(n) 1=< n =< 254 N(n) 1=< n =< 127

注 2: 文字データ可変長の長さ部は、文字数を指定します。

対応表に記述されていない対応付けを行った場合、データの内容は保証されません。

NCOBU00993

日付データの対応表（ODBC）

以下は、ODBC で扱う日付データと COBOL で扱う日付データの対応表です。

表: 日付データの対応表

ODBC SQL データ型		COBOL での表現
日付データ	SQL_DATE	PIC X(n) (注 1)
	SQL_TIME	
	SQL_TIMESTAMP	

注 1: 文字数 n は、データベースドライバがサポートしている ODBC ドライバマネージャーのバージョン、データベースドライバの仕様によって制限されます。

制限事項・仕様変更

ここでは、NetCOBOL for .NET の制限事項および仕様変更について説明します。

このセクションの内容

NCOBU00705

NetCOBOL for .NET の制限事項

NetCOBOL for .NET の本バージョンにおける制限事項は、以下のとおりです。

- 画面節は、サポートされていません。 NetCOBOL for .NET は、よりユーザビリティの高い Windows アプリケーションや ASP.NET Web アプリケーションをサポートしています。
- 報告書作成機能は、サポートされていません。
- TYPEDEF STRONG は、サポートされていません。
- ENTRY 文は、サポートされていません。
- ALTER 文は、サポートされていません。 ALTER 文は、1985 年の COBOL 規格で、廃要素だと宣言されました。
- 手続き名(ALTER 文のターゲットとして使用される)のない GO TO 文は、サポートされていません。
- メッセージをオペレータに表示し、応答待ちで実行を一時停止する「STOP 定数」文はサポートされていません。
- .NET の例外処理は、規格 OO COBOL の例外処理といくつか異なります。 .NET の例外は、一致する例外ハンドラが発見されるまで、呼び出し元へ伝播します。 規格 OO COBOL では、例外が発生するソース単位でハンドラを定義する必要があります。
- 帳票印刷以外の表示ファイル機能は、サポートされていません。

NetCOBOL for .NET の仕様変更

ここでは、エラーチェック強化による V1.1 からの仕様の変更について説明します。

このセクションの内容

NCOBU00707

BY VALUE/BY REFERENCE 指定の違いだけでのオーバロードされたメソッドの呼出し

現象

同じ位置のパラメタの属性が一致して、この仮パラメタに指定された BY 指定の違いだけで、オーバロードされたメソッドを呼び出す際は実パラメタの指定に明示的な BY 指定を必要とします。

```
CLASS-ID. C1.
      …
METHOD-ID. M1.
      …
LINAKGE SECTION.
01 LK1  S9(9) COMP-5.
PROCEDURE DIVISION USING BY REFERECE LK1.
      …
END METHOD M1.
      …
METHOD-ID. M1.
      …
LINAKGE SECTION.
01 LK1  S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE   LK1.
      …
END METHOD M1.
```

```
PRORAM-ID. P1
      …
REPOSITORY.
    CLASS C1.
      …
01 OBJC1 OBJECT REFERENCE C1.
01 WINT   PIC S9(9) COMP-5.
      …
    INVOKE OBJC1 "M1" USING WINT *> ★
      …
```

V1.0L10/V1.1L10 では、★のような記述が翻訳エラーになりませんでしたが、V2.0L10 以降では翻訳時に診断メッセージ JMN5725I-S が出力されます。

対処方法

呼び出したいメソッドの定義に合わせて USING 指定に BY 指定を記述してください。

66/78/88 レベル項目の記述個所

現象

プログラム原型定義、デリゲート定義、インタフェース定義の連絡節にはレベル番号 66, 78, 88 の項目は記述できません。 記述されている場合、翻訳時に診断メッセージ JMN5265I-S が出力されます。

対処方法

該当する項目を削除してください。

NCOBU00709

System.Void クラスのオブジェクト参照

現象

System.Void クラスを参照するオブジェクト参照データ項目を宣言することはできません。 また、オブジェクト指定子に System.Void クラスを指定することはできません。

対処方法

該当する項目を削除してください。

PROTECTED メンバーのアクセス

現象

PROTECTED メンバーを、そのメンバーを宣言したクラスのオブジェクト修飾でアクセスすることを禁止しました。

```
CLASS-ID.   A.
              :
 01  X  ... PROTECTED.
```

```
CLASS-ID.   B  INHERITS  A.
              :
 01  OBJA  OBJECT REFERECE A.
              :
   SET  X OF OBJA ... *> ★
```

★印の箇所のように継承先の型を通さずに PROTECTED メンバーにアクセスすることは typesafe な用法ではないので、V2.0L10 からは翻訳エラーとしています。

対処方法

その PROTECTED メンバーが宣言されているクラスから派生したクラスのオブジェクトを使用するか、または、定義済み一意名の SELF または SUPER を使用してアクセスしてください。

NCOBU00711

転記の送出し側にUSAGE BINARY-CHAR UNSIGNEDを指定した場合の仕様

現象

USAGE BINARY-CHAR UNSIGNED のデータを英数字項目などに転記していると翻訳エラーとなります。

対処方法

一旦 int 型 2 進項目以外の整数項目に転記してから、英数字項目などに転記してください。

COBOL ファイルユーティリティ関数のインタフェースの変更

現象

COBOL ファイルユーティリティ関数のインタフェースが、次のように仕様変更になりました。

V1.1L10 以前 ：

プログラム原型の連絡節に、登録集 COBF-INF.cbl を COPY 文で取り込む。

V2.0L10 :

プログラム原型の連絡節に、登録集 COBF-PRO.cbl を COPY 文で取り込む。

プログラム原型の連絡節に、登録集 COBF-INF.cbl を COPY 文で取り込むと、翻訳時に次の診断メッセージが出力されます。

```
JMN5265I-S プログラム原型定義中に 88 レベル項目は記述できません
```

[対処方法]

この場合、COPY 文で取り込む登録集を“COBF-INF.cbl”から“COBF-PRO.cbl”に変更してください。

V2.1L10 以降

プログラム原型定義を用意せずに、通常の CALL 呼出しで COBOL ファイルユーティリティ関数を呼び出せます。

NCOBU00713

サンプルアプリケーション

ここでは、NetCOBOL for .NET を使用して、 プログラミングの実証を行う様々なサンプルアプリケーションの説明と、 リンクを提供します。

このトピックの内容

NetCOBOL for .NET サンプルアプリケーションのビルドと実行

サンプルのビルド、および実行の手順を説明します。

NetCOBOL for .NET コマンドラインサンプルアプリケーション

コマンドラインの例を説明し、リンクを提供します。

NetCOBOL for .NET Visual Studio プロジェクトサンプルアプリケーション

Visual Studio との統合を、 例をあげてサンプルの説明を行い、 そのサンプルのソースにリンクします。

NetCOBOL for .NET サンプルデータベースのセットアップ

サンプルで使用するデータベースのセットアップ方法を説明します。

NetCOBOL for .NET データベースアクセスサンプルアプリケーション

データベースアクセスの例を説明し、 そのサンプルのソースにリンクします。

NetCOBOL for .NET Web サンプル アプリケーション（ASP.NET および Web サービス）

ASP.NET および Web サービスを使用したサンプルの説明をします。

NetCOBOL for .NET Windows Communication Foundation サンプル アプリケーション

Windows Communication Foundation(WCF)を使用したサンプルの説明をします。

例題集

従来の COBOL の機能を使用したサンプルについて説明します。

NetCOBOL for .NET サンプルアプリケーションのビルドと実行

サンプルアプリケーションをすべてビルドする

NetCOBOL for .NET に含まれる、サンプルアプリケーションをすべてビルドするには、 NetCOBOL for .NET のサンプルフォルダにあるバッチファイル("BuildAll.bat")を以下の手順で実行します。

ビルドには、NMAKE ユーティリティを使用します。NMAKE ユーティリティは、Visual Studio 2017 のワークロード「C++ によるデスクトップ開発」がインストールされている場合に利用することができます。

1. NetCOBOL for .NET コマンドプロンプトウィンドウを起動します。 [参照] "コマンドプロンプトの起動"
2. コマンドプロンプトで、サンプルが位置するサブフォルダにカレントフォルダを移します。 そのとき、コマンドは以下のようになります。

```
CD "¥Program Files¥Fujitsu NetCOBOL for .NET V8.0¥Examples"
```

3. 「BuildAll」と入力します。

ただし、Visual Studio を使用するサンプルアプリケーションは、この操作でビルドされません。 各サンプルアプリケーションのフォルダにあるソリューションファイル(.sln)を Visual Studio で開き、ビルドしてください。

サンプルアプリケーションを実行する

コマンドラインサンプルアプリケーションまたは Visual Studio プロジェクトサンプルアプリケーションを実行するには、 ビルド処理によって出力された.exe ファイルをコマンドプロンプトで入力するか、 または Windows エクスプローラー上で.exe ファイルをダブルクリックします。

コマンドラインサンプルアプリケーションおよび Visual Studio プロジェクトサンプルアプリケーションには、データベースを使用するアプリケーションが含まれています。 サンプルで使用するデータベースの設定方法については、NetCOBOL for .NET サンプルデータベースのセットアップを参照してください。

Web サンプルアプリケーションの設定方法および実行方法については、NetCOBOL for .NET Web サンプル アプリケーション（ASP.NET および Web サービス）を参照してください。

サンプルアプリケーションをすべてクリーンアップする

サンプルアプリケーションをすべてクリーンアップ(ビルドで作成されたファイルをすべて削除すること)するには、 NetCOBOL for .NET のサンプルフォルダにあるバッチファイル("CleanAll.bat")を以下の手順で実行します。

1. NetCOBOL for .NET コマンドプロンプトウィンドウを起動します。 [参照] "コマンドプロンプトの起動"
2. コマンドプロンプトで、サンプルが位置するサブフォルダにカレントフォルダを移します。 そのとき、コマンドは以下のようになります。

```
CD "¥Program Files¥Fujitsu NetCOBOL for .NET V8.0¥Examples"
```

3. 「CleanAll」と入力します。

ただし、Visual Studio を使用するサンプルアプリケーションは、この操作でクリーンアップされません。 各サンプルアプリケーションのフォルダにあるソリューションファイル(.sln)を Visual Studio で開き、 [ビルド]-[ソリューションのクリーン]を選択してください。

サンプルアプリケーションを個別にビルドする

ビルドの方法は、サンプルアプリケーションの種類によって異なります。

コマンドラインサンプルアプリケーションのビルド

コマンドラインサンプルアプリケーションは、 NetCOBOL for .NET をインストールしたフォルダ内の Examples フォルダ配下の CommandLine というフォルダに格納されています。

コマンドラインサンプルアプリケーションのビルドでは、 NetCOBOL for .NET コンパイラをコマンドプロンプトから使用します。 NetCOBOL for .NET コマンドプロンプトを起動してください。 [参照] "コマンドプロンプトの起動"

各コマンドラインサンプルアプリケーションには、Makefile というメイクファイルを含んでいます。 サンプルアプリケーションを含むフォルダにカレントフォルダを変更し、「nmake」と入力することでビルドすることができます。 NetCOBOL for .NET コンパイラに渡されるオプションについて確認したい場合は、各 Makefile を参照してください。

Visual Studio プロジェクトサンプルアプリケーションのビルド

Visual Studio プロジェクトサンプルアプリケーションは、 NetCOBOL for .NET をインストールしたフォルダ内の Examples フォルダ配下の VisualStudio というフォルダに格納されています。

各サンプルアプリケーションのフォルダにあるソリューションファイル（.sln）を Visual Studio で開き、 [ビルド]-[ソリューションのビルド]を選択します。

Web サンプルアプリケーション（ASP.NET、および Web サービス）のビルド

Web サンプルアプリケーションは、 NetCOBOL for .NET をインストールしたフォルダ内の Examples フォルダ配下の Web というフォルダに格納されています。

各サンプルアプリケーションのフォルダにあるソリューションファイル（.sln）を Visual Studio で開き、 [ビルド]-[ソリューションのビルド]を選択します。

サンプルアプリケーションを個別にクリーンアップする

メイクファイルを使用してビルドされたサンプルサンプルアプリケーションをクリーンアップ(ビルドで作成されたファイルをすべて削除すること）するには、 各サンプルアプリケーションのフォルダにあるメイクファイルを以下の手順で実行します。

1. NetCOBOL for .NET コマンドプロンプトウィンドウを起動します。 [参照] "コマンドプロンプトの起動"
2. クリーンアップするサンプルアプリケーションのフォルダに、カレントフォルダを移動します。
3. 「nmake clean」と入力します。

また、Visual Studio を使用してビルドしたサンプルアプリケーションをクリーンアップするには、 各サンプルアプリケーションのフォルダにあるソリューションファイル(.sln)を Visual Studio で開き、[ビルド]-[ソリューションのクリーン]を選択します。

NetCOBOL for .NET コマンドラインサンプルアプリケーション

コマンドラインサンプルは、 NetCOBOL for .NET コンパイラをコマンドラインで利用します。 これらは、CommandLine というサブフォルダにあり、 各サンプルには Makefile が含まれています。 サンプルのビルドと実行については、NetCOBOL for .NET サンプルアプリケーションのビルドと実行を参照してください。

サンプルごとに、`Examples¥CommandLine` の下にあるサブフォルダ名、 実行可能なサンプルコマンド名、ソースコードへのリンク、および 簡単な説明を以下に示します。

1. Windows Forms "Hello World"

フォルダ	HelloWin
コマンドライン	HelloWin.exe
ソースコード	HelloWin.cob

説明

この Windows フォームアプリケーションはタイトルバーに、 "Hello COBOL World" という文字列を表示します。

2. PInvoke

フォルダ	PInvoke
コマンドライン	PInvoke.exe
ソースコード	PInvoke.cob

説明

これは、アンマネージコードの MessageBox Win32 API を呼び出す "Hello World"プログラムです。

3. Walk The Dog

フォルダ	WalkTheDog
コマンドライン	WalkTheDog.exe
ソースコード	walkthedog.cob

説明

このプログラムは、 犬の散歩を思い出させるメッセージボックスを定期的に表示するために、 Timer オブジェクトのイベントハンドラを使用する、 Windows フォームアプリケーションです。 このメッセージボックスが不要な場合、 Stop It ボタンで Timer オブ

ジェクトを止めることができます。

4. Print1

フォルダ	Print1
コマンドライン	Print1.exe
ソースコード	Print1.cob

説明

このプログラムは、 FORMAT 句なし印刷ファイルを使用して、 I 制御レコードを使用したページ形式の設定／変更と、 CHARACTER TYPE 句や PRINTING POSITION 句を使用して印字したい文字の修飾および配置(行／桁)を意識して印刷装置に出力するプログラムです。 詳細については、NetCOBOL for .NET 印刷ファイルを使ったサンプル 1 を参照してください。

5. Print2

フォルダ	Print2
コマンドライン	Print2.exe
ソースコード	Print2.cob

説明

このプログラムは、FORMAT 句付き印刷ファイルを使用して、 I 制御レコードを使用したページ形式の設定／変更と、 CHARACTER TYPE 句や PRINTING POSITION 句を使用して印字したい文字の修飾および配置(行／桁)を意識して印刷装置に出力するプログラムです。 詳細については、NetCOBOL for .NET 印刷ファイルを使ったサンプル 2 を参照してください。

6. Print3

フォルダ	Print3
コマンドライン	Print3.exe
ソースコード	Print3.cob

説明

このプログラムは、FORMAT 句付き印刷ファイルを使用して、 集計表を印刷装置に出力するプログラムです。 詳細については、NetCOBOL for .NET 印刷ファイルを使ったサンプル 3 を参照してください。

NetCOBOL for .NET 印刷ファイルを使ったサンプル 1

FORMAT 句なし印刷ファイルを使って、I 制御レコードを使用したページ形式の設定／変更と、 CHARACTER TYPE 句や PRINTING POSITION 句を使用して印字したい文字の修飾および配置(行／桁) を意識して印刷装置に出力するプログラムの例を示します。

FORMAT 句なし印刷ファイルを使用して帳票印刷を行う場合、 主に利用される機能を想定し、 以下の項目について印刷デモを行います。

FCB を使用した 6LPI、8LPI での帳票印刷

FCB を利用した任意の行間隔(6/8LPI)で帳票印刷を行うことを想定し、 I 制御レコードによる FCB(LPI)の切り替えを行う例を示しています。

このとき、CHARACTER TYPE 句、PRINTING POSITION 句 を利用して、行間隔(LPI)や文字 間隔(CPI) などの行・桁を意識して帳票の体裁を整えています。

- A4 用紙を横向きに使用し、 1 ページすべての行間隔を 6LPI とした場合の帳票をイメージし、6LPI/10CPI フォーマットのスペーシングチャート形式のフォームオーバレイと 重畳印刷します。
- A4 用紙を横向きに使用し、 1 ページすべての行間隔を 8LPI とした場合の帳票をイメージし、8LPI/10CPI フォーマットのスペーシングチャート形式のフォームオーバレイと重畳印刷します。

CHARACTER TYPE 句で指定する各種文字属性での印刷

次に、I 制御レコードを使用し、 用紙サイズを A4／横向きから B4／横向きに変更し、これにあわせて FCB も A4／横向き用から B4／横向き用に変更します。 その後、各種文字属性の印字サンプルを印刷装置に出力します。

- 文字サイズ

  1 文字ずつ 3、7.2、9、12、18、24、36、50、72、100、 200、300 ポイントの文字サイズを印字します。

  ※ ここでは、文字ピッチ指定を省略することによって、 文字サイズに合わせた最適な文字ピッチを NetCOBOL for .NET ランタイムシステムに自動算出させます。

- 文字ピッチ

  文字ピッチ 1CPI で 1 文字、2CPI で 2 文字、3CPI で 3 文字、5CPI で 5 文字、6CPI で 6 文字、7.5CPI で 15 文字、20CPI で 20 文字、24CPI で 24 文字指定します。

  ※ ここでは、文字サイズ指定を省略することによって、 文字ピッチに合わせた最 適な文字サイズを COBOL ランタイムシステムに自動算出させます。

- 文字書体

  ゴシック、ゴシック半角（文字形態半角）、明朝、明朝半角（文字形態半角）を 10 文字ずつ 2 回繰り返し印字します。

  ※ ここで指定した書体名は、以下の実行環境情報に関連付けられています。

```
@PrinterFontName=(FONT-NAME1,FONT-NAME2)
```

  "MINCHOU" 、"MINCHOU-HANKAKU" を指定したデータ項目は、"FONT-NAME1"に 指定されたフォントを用いて印字され、 "GOTHIC"、

"GOTHIC-HANKAKU"を指定したデータ項目は、 "FONT-NAME2"に指定されたフォントを用いて印字されます。

なお、本サンプルプログラムでは、アプリケーション構成ファイル(Print1.exe.config)内に "@PrinterFontName=(MS 明朝,MS ゴシック)"を指定しています。

・ 文字回転

縦書き（反時計回りに 90 度回転)、横書きを 10 文字ずつ繰り返し印字します。

・ 文字形態

全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、半各倍角の文字形態指定を 9 文字ずつ印刷します。

上記５つの文字属性を組み合わせた印刷を行います。

実行は、特に応答・操作する必要はなく自動的に終了します。 実行が終了するとサンプル帳票が、 "通常使うプリンタ"として設定されている印刷装置に出力されます。

NetCOBOL for .NET 印刷ファイルを使ったサンプル 2

FORMAT 句付き印刷ファイルを使って、I 制御レコードを使用したページ形式の設定／変更と、 CHARACTER TYPE 句や PRINTING POSITION 句を使用して印字したい文字の修飾および配置(行／桁) を意識して印刷装置に出力するプログラムの例を示します。 なお、このプログラムを実行する には、MeFt が必要です。

FORMAT 句付き印刷ファイルで行レコードを使用して帳票印刷を行う場合、 主に利用される機能を想定し、 以下の項目について印刷デモを行います。

FCB を使用した 6LPI、8LPI での帳票印刷

FCB を利用した任意の行間隔(6/8LPI)で帳票印刷を行うことを想定し、 I 制御レコードによる FCB(LPI)の切り替えを行う例を示しています。

このとき、CHARACTER TYPE 句、PRINTING POSITION 句 を利用して、行間隔(LPI)や文字 間隔(CPI) などの行・桁を意識して帳票の体裁を整えています。

- A4 用紙を横向きに使用し、 1 ページすべての行間隔を 6LPI とした場合の帳票をイメージし、6LPI/10CPI フォーマットのスペーシングチャート形式のフォームオーバレイと重畳印刷します。
- A4 用紙を横向きに使用し、 1 ページすべての行間隔を 8LPI とした場合の帳票をイメージし、8LPI/10CPI フォーマットのスペーシングチャート形式のフォームオーバレイと重畳印刷します。

CHARACTER TYPE 句で指定する各種文字属性での印刷

次に、I 制御レコードを使用し、用紙サイズを A4／横向きから B4／横向きに変更し、 これにあわせて FCB も A4／横向き用から B4／横向き用に変更します。 その後、各種文字属性の印字サンプルを印刷装置に出力します。

- 文字サイズ

  1 文字ずつ 3 ポ、7.2 ポ、9 ポ、12 ポ、18 ポ、24 ポ、36 ポ、50 ポ、72 ポ、100 ポ、 200 ポ、300 ポの文字サイズを印字します。

  ※ ここでは、文字ピッチ指定を省略することによって、 文字サイズに合わせた最適な文字ピッチを NetCOBOL for .NET ランタイムシステムに自動算出させます。

- 文字ピッチ

  文字ピッチ 1CPI で 1 文字、2CPI で 2 文字、3CPI で 3 文字、5CPI で 5 文字、6CPI で 6 文字、7.5CPI で 15 文字、20CPI で 20 文字、24CPI で 24 文字指定します。

  ※ ここでは、文字サイズ指定を省略することによって、 文字ピッチに合わせた最 適な文字サイズを NetCOBOL for .NET ランタイムシステムに自動算出させます。

- 文字書体

  ゴシック、ゴシック半角（文字形態半角）、明朝、明朝半角（文字形態半角） を 10 文字ずつ 2 回繰り返し印字します。

- 文字回転

  縦書き（反時計回りに 90 度回転）、横書きを 10 文字ずつ繰り返し印字します。

・ 文字形態

全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、 半各倍角の 文字形態指定を 9 文字ずつ印刷します。

上記５つの文字属性を組み合わせた印刷を行います。

実行は、特に応答・操作する必要はなく自動的に終了します。 実行が終了すると、 "通常使うプリンタ"として設定されている印刷装置にサンプル帳票が出力されます。

出力先プリンタを指定する場合は、 プリンタ情報ファイル(PRT.env)のキーワード PRTDRV に、 出力プリンタデバイス名を指定してください。

NetCOBOL for .NET 印刷ファイルを使ったサンプル 3

FORMAT 句付き印刷ファイルを使って、 集計表を印刷装置に出力するプログラムの例を示します。 なお、このプログラムを実行するには、MeFt が必要です。

FORMAT 句付き印刷ファイルを使用して、 商品コード、商品名、および単価が格納されているマスタファイル（索引ファイル）と受注日、 数量、および売上げ金額が格納されている売上げファイル（索引ファイル）を入力して、 売上集計表を印刷装置に出力します。

使用している帳票定義体

売上集計表(SYUUKEI.PMD)

形式	集計表形式	
用紙サイズ	A4	
用紙方向	縦	
行ピッチ	1/6 インチ	
パーティション	PH(ページ頭書き)	[固定パーティション、印刷開始位置:0 インチ(1 行目)、縦幅:1 インチ(6 行)]
	CH1(制御頭書き)	[浮動パーティション、縦幅:0.83 インチ(5 行)]
	DE(明細)	[浮動パーティション、縦幅:0.33 インチ(2 行)]
	CF1(制御脚書き)	[浮動パーティション、縦幅:0.83 インチ(5 行)]
	CF2(制御脚書き)	[浮動パーティション、縦幅:0.67 インチ(4 行)]
	PF(ページ脚書き)	[固定パーティション、印刷開始位置:10.48 インチ(63 行目)、縦幅:0.49 インチ(3 行)]

プログラムを作成する上でのポイント

- PH および PF は、 固定パーティション（固定の印刷位置情報を持っている）なので、 パーティションを出力すると、 必ず、パーティションに定義されている印刷開始位置 に出力されます。
- CH1、DE、CF1 および CF2 は、 浮動パーティション（固定の印刷位置情報を持たない）なので、 自由な位置に出力することができる反面、 パーティション出力時に印刷位置を制御する必要があります。
- 各パーティションに定義された出力項目は、 翻訳時に COPY 文で帳票定義体からレコードに展開されます。 このとき、定義した出力項目の項目名がデータ名になります。

プログラムを実行する前に

MeFt のセットアップを行い、使用できる状態にしておいてください。

実行は、特に応答・操作する必要はなく自動的に終了します。 実行が終了すると、"通常使うプリンタ"として設定されている印刷装置にサンプル帳票が出力されます。

出力先プリンタを指定する場合は、 プリンタ情報ファイル(PRT.env)のキーワード PRTDRV に、 出力プリンタデバイス名を指定してください。

ソースコード：HelloWin.cob

```
000010 CLASS-ID. HELLO INHERITS FORM.
000020 ENVIRONMENT DIVISION.
000030 CONFIGURATION SECTION.
000040 REPOSITORY.
000050    PROPERTY WIN-TEXT AS "Text"
000060    CLASS APPLICATION AS "System.Windows.Forms.Application"
000070    CLASS FORM AS "System.Windows.Forms.Form".
000080 STATIC.
000090 PROCEDURE DIVISION.
000100 METHOD-ID. MAIN.
000110 DATA DIVISION.
000120 WORKING-STORAGE SECTION.
000130 77 APP-OBJ USAGE OBJECT REFERENCE FORM.
000140 PROCEDURE DIVISION.
000150    INVOKE HELLO "NEW" RETURNING APP-OBJ.
000160    SET WIN-TEXT OF APP-OBJ TO "Hello COBOL World".
000170    INVOKE APPLICATION "Run" USING BY VALUE APP-OBJ.
000180 END METHOD MAIN.
000190 END STATIC.
000200 END CLASS HELLO.
```

NCOBU00721

ソースコード：PInvoke.cob

```
000270 CLASS-ID. HELLO.
000280 ENVIRONMENT DIVISION.
000290 CONFIGURATION SECTION.
000300 REPOSITORY.
000310    CLASS SYS-STRING AS "System.String"
000320    PROGRAM MessageBox.
000330 STATIC.
000340 PROCEDURE DIVISION.
000350 METHOD-ID. MAIN.
000360 DATA DIVISION.
000370 WORKING-STORAGE SECTION.
000380 01 HANDLE BINARY-LONG.
000390 01 MESSAGE-TEXT OBJECT REFERENCE SYS-STRING.
000400 01 CAPTION-TEXT OBJECT REFERENCE SYS-STRING.
000410 01 MSGBOX-TYPE BINARY-LONG.
000420 PROCEDURE DIVISION.
000430    MOVE 0 TO HANDLE.
000440    SET MESSAGE-TEXT TO "Hello world!".
000450    SET CAPTION-TEXT TO "Calling Unmanaged Code".
000460    MOVE 0 TO MSGBOX-TYPE.
000470    CALL MessageBox USING BY VALUE HANDLE MESSAGE-TEXT
000480       CAPTION-TEXT MSGBOX-TYPE.
000490 END METHOD MAIN.
000500 END STATIC.
000510 END CLASS HELLO.
```

ソースコード：walkthedog.cob

```
000010 CLASS-ID. WALK INHERITS FORM.
000020 ENVIRONMENT DIVISION.
000030 CONFIGURATION SECTION.
000040 REPOSITORY.
000050    CLASS FORM AS "System.Windows.Forms.Form"
000060    CLASS REMINDER
000061    CLASS ABOUT
000070    CLASS RESOURCEMANAGER AS "System.Resources.ResourceManager"
000080    CLASS APPLICATION AS "System.Windows.Forms.Application"
000090    CLASS CONTAINER AS "System.ComponentModel.Container"
000100    CLASS BUTTON AS "System.Windows.Forms.Button"
000110    CLASS TIMER AS "System.Windows.Forms.Timer"
000120    CLASS PICTUREBOX AS "System.Windows.Forms.PictureBox"
000130    CLASS BITMAP AS "System.Drawing.Bitmap"
000140    CLASS IMAGE AS "System.Drawing.Image"
000150    DELEGATE EVENTHANDLER AS "System.EventHandler"
000160    CLASS EVENTARGS AS "System.EventArgs"
000170    CLASS POINT AS "System.Drawing.Point"
000180    CLASS DRAWING-SIZE AS "System.Drawing.Size"
000190    CLASS FONT AS "System.Drawing.Font"
000200    CLASS SYS-OBJECT AS "System.Object"
000210    CLASS SYS-STRING AS "System.String"
000220    CLASS SYS-BOOLEAN AS "System.Boolean"
000230    CLASS CONTROL-COLLECTION AS "System.Windows.Forms.Control+ControlCollection"
000240    CLASS MAIN-MENU AS "System.Windows.Forms.MainMenu"
000250    CLASS MENU-ITEM AS "System.Windows.Forms.MenuItem"
000260    CLASS PATH AS "System.IO.Path"
000261    INTERFACE ICOMPONENT AS "System.ComponentModel.IComponent"
000262    INTERFACE IWIN32-WINDOW AS "System.Windows.Forms.IWin32Window"
000270    PROPERTY PROP-INTERVAL AS "Interval"
000280    PROPERTY PROP-ENABLED AS "Enabled"
000290    PROPERTY PROP-LOCATION AS "Location"
000300    PROPERTY PROP-SIZE AS "Size"
000310    PROPERTY PROP-TABINDEX AS "TabIndex"
000320    PROPERTY PROP-FONT AS "Font"
000330    ENUM FONT-STYLE AS "System.Drawing.FontStyle"
000340    PROPERTY PROP-BOLD AS "Bold"
000350    PROPERTY PROP-TEXT AS "Text"
000360    PROPERTY PROP-INDEX AS "Index"
000380    PROPERTY APP-CONTROLS AS "Controls"
000390    PROPERTY PROP-TABSTOP AS "TabStop"
000400    PROPERTY PROP-AUTOSCALEBASESIZE AS "AutoScaleBaseSize"
000410    PROPERTY PROP-CLIENTSIZE AS "ClientSize"
000420    PROPERTY PROP-IMAGE AS "Image"
000430    ENUM PICTUREBOX-SIZEMODE AS "System.Windows.Forms.PictureBoxSizeMode"
000440    PROPERTY PROP-SIZE-MODE AS "SizeMode"
000450    PROPERTY PROP-CENTER-IMAGE AS "CenterImage"
000460    PROPERTY PROP-EXECUTABLE-PATH AS "ExecutablePath"
000461    PROPERTY PROP-MENU AS "Menu"
000470    PROPERTY PROP-MENU-ITEMS AS "MenuItems"
000480    .
000490 STATIC.
000500 PROCEDURE DIVISION.
000510 METHOD-ID. MAIN.
000520 DATA DIVISION.
000530 WORKING-STORAGE SECTION.
000540 01 OBJ OBJECT REFERENCE WALK.
000550 PROCEDURE DIVISION.
000560    INVOKE WALK "NEW" RETURNING OBJ.
000570    INVOKE APPLICATION "Run" USING BY VALUE OBJ.
000580 END METHOD MAIN.
000590 END STATIC.
000600 OBJECT.
000610 DATA DIVISION.
000620 WORKING-STORAGE SECTION.
000630 01 COMPONENTS OBJECT REFERENCE CONTAINER.
000640 01 BUTTON1 OBJECT REFERENCE BUTTON.
000650 01 TIMER1 OBJECT REFERENCE TIMER.
000660 01 PICTUREBOX1 OBJECT REFERENCE PICTUREBOX.
000670 01 MAINMENU1 OBJECT REFERENCE MAIN-MENU.
000680 01 FILE-MENU OBJECT REFERENCE MENU-ITEM.
```

```
000690 01 EXIT-MENU OBJECT REFERENCE MENU-ITEM.
000700 01 HELP-MENU OBJECT REFERENCE MENU-ITEM.
000710 01 ABOUT-MENU OBJECT REFERENCE MENU-ITEM.
000720 PROCEDURE DIVISION.
000730*
000740 METHOD-ID. NEW.
000750 DATA DIVISION.
000760 WORKING-STORAGE SECTION.
000770 01 APP-PATH OBJECT REFERENCE SYS-STRING.
000780 01 IMAGE-PATH OBJECT REFERENCE SYS-STRING.
000790 PROCEDURE DIVISION.
000800    INVOKE CONTAINER "NEW" RETURNING COMPONENTS.
000810    INVOKE SELF "InitializeComponent".
000820    SET APP-PATH TO PROP-EXECUTABLE-PATH OF APPLICATION.
000830    INVOKE PATH "GetDirectoryName" USING BY VALUE APP-PATH RETURNING APP-PATH.
000840    INVOKE PATH "Combine" USING BY VALUE APP-PATH "MAIN.JPG" RETURNING IMAGE-PATH.
000850    SET PROP-IMAGE OF PICTUREBOX1 TO IMAGE::"FromFile" (IMAGE-PATH).
000860 END METHOD NEW.
000870*
000880 METHOD-ID. DISPOSE AS "Dispose" PROTECTED OVERRIDE.
000890 DATA DIVISION.
000900 WORKING-STORAGE SECTION.
000910 01 FLAG PIC 1 USAGE BIT.
000920 LINKAGE SECTION.
000930 01 DISPOSING OBJECT REFERENCE SYS-BOOLEAN.
000940 PROCEDURE DIVISION USING BY VALUE DISPOSING.
000950    SET FLAG TO DISPOSING.
000960    IF FLAG = B"1" AND COMPONENTS NOT = NULL THEN
000970       INVOKE COMPONENTS "Dispose"
000980    END-IF.
000990    INVOKE SUPER "Dispose" USING BY VALUE DISPOSING.
001000 END METHOD DISPOSE.
001010*
001020 METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent".
001030 DATA DIVISION.
001040 WORKING-STORAGE SECTION.
001050 01 TICK-EVENT OBJECT REFERENCE EVENTHANDLER.
001060 01 CLICK-EVENT OBJECT REFERENCE EVENTHANDLER.
001070 01 EXIT-EVENT OBJECT REFERENCE EVENTHANDLER.
001080 01 ABOUT-EVENT OBJECT REFERENCE EVENTHANDLER.
001090 01 BOLD-FONT OBJECT REFERENCE FONT-STYLE.
001100 PROCEDURE DIVISION.
001110    INVOKE PICTUREBOX "NEW" RETURNING PICTUREBOX1.
001120    INVOKE BUTTON "NEW" RETURNING BUTTON1.
001130    INVOKE TIMER "NEW" USING BY VALUE COMPONENTS RETURNING TIMER1.
001140    INVOKE MAIN-MENU "NEW" RETURNING MAINMENU1.
001150    INVOKE MENU-ITEM "NEW" RETURNING FILE-MENU.
001160    INVOKE MENU-ITEM "NEW" RETURNING EXIT-MENU.
001170    INVOKE MENU-ITEM "NEW" RETURNING HELP-MENU.
001180    INVOKE MENU-ITEM "NEW" RETURNING ABOUT-MENU.
001190    INVOKE SELF "SuspendLayout".
001200*
001210    SET PROP-LOCATION OF PICTUREBOX1 TO POINT::"NEW" (24, 24).
001220    SET PROP-SIZE OF PICTUREBOX1 TO DRAWING-SIZE::"NEW" (240, 128).
001230    SET PROP-TABSTOP OF PICTUREBOX1 TO B'0'.
001240    SET PROP-SIZE-MODE OF PICTUREBOX1 TO PROP-CENTER-IMAGE OF PICTUREBOX-SIZEMODE.
001250    INVOKE APP-CONTROLS OF SELF "Add" USING BY VALUE PICTUREBOX1.
001260*
001270    SET PROP-LOCATION OF BUTTON1 TO POINT::"NEW" (40, 184).
001280    SET PROP-SIZE OF BUTTON1 TO DRAWING-SIZE::"NEW" (208, 48).
001290    MOVE 0 TO PROP-TABINDEX OF BUTTON1.
001300    SET PROP-TEXT OF BUTTON1 TO "&Stop It!".
001310    SET BOLD-FONT TO PROP-BOLD OF FONT-STYLE.
001320    SET PROP-FONT OF BUTTON1 TO
001330       FONT::"NEW" ("Microsoft Sans Serif", 12.0, BOLD-FONT).
001340    INVOKE EVENTHANDLER "NEW" USING BY VALUE SELF "Button1_Click"
001350       RETURNING CLICK-EVENT.
001360    INVOKE BUTTON1 "add_Click" USING BY VALUE CLICK-EVENT.
001370    INVOKE APP-CONTROLS OF SELF "Add" USING BY VALUE BUTTON1.
001380*
001390    MOVE 7000 TO PROP-INTERVAL OF TIMER1.
001400    SET PROP-ENABLED OF TIMER1 TO B'1'.
001410    INVOKE EVENTHANDLER "NEW" USING BY VALUE SELF "Timer1_Tick"
001420       RETURNING TICK-EVENT.
001430    INVOKE TIMER1 "add_Tick" USING BY VALUE TICK-EVENT.
001440*
001450    INVOKE PROP-MENU-ITEMS OF MAINMENU1 "Add" USING BY VALUE FILE-MENU.
001460    INVOKE PROP-MENU-ITEMS OF MAINMENU1 "Add" USING BY VALUE HELP-MENU.
001470*
001480    MOVE 0 TO PROP-INDEX OF FILE-MENU.
001490    INVOKE PROP-MENU-ITEMS OF FILE-MENU "Add" USING BY VALUE EXIT-MENU.
001500    SET PROP-TEXT OF FILE-MENU TO "&File".
```

```
001510*
001520    MOVE 0 TO PROP-INDEX OF EXIT-MENU.
001530    SET PROP-TEXT OF EXIT-MENU TO "E&xit".
001540    INVOKE EVENTHANDLER "NEW" USING BY VALUE SELF "Exit_Click"
001550       RETURNING EXIT-EVENT.
001560    INVOKE EXIT-MENU "add_Click" USING BY VALUE EXIT-EVENT.
001570*
001580    MOVE 1 TO PROP-INDEX OF HELP-MENU.
001590    INVOKE PROP-MENU-ITEMS OF HELP-MENU "Add" USING BY VALUE ABOUT-MENU.
001600    SET PROP-TEXT OF HELP-MENU TO "&Help".
001610*
001620    MOVE 0 TO PROP-INDEX OF ABOUT-MENU.
001630    SET PROP-TEXT OF ABOUT-MENU TO "&About".
001640    INVOKE EVENTHANDLER "NEW" USING BY VALUE SELF "About_Click"
001650       RETURNING ABOUT-EVENT.
001660    INVOKE ABOUT-MENU "add_Click" USING BY VALUE ABOUT-EVENT.
001670*
001671    SET PROP-MENU OF SELF TO MAINMENU1.
001680    SET PROP-TEXT OF SELF TO "My Dog".
001690    SET PROP-AUTOSCALEBASESIZE OF SELF TO DRAWING-SIZE::"NEW" (5, 13).
001700    SET PROP-CLIENTSIZE OF SELF TO DRAWING-SIZE::"NEW" (292, 274).
001710    INVOKE SELF "ResumeLayout" USING BY VALUE B'0'.
001720 END METHOD INITIALIZECOMPONENT.
001730*
001740 METHOD-ID. BUTTON1_CLICK AS "Button1_Click".
001750 DATA DIVISION.
001760 WORKING-STORAGE SECTION.
001770 01 ENABLED PIC 1 USAGE BIT.
001780 LINKAGE SECTION.
001790 01 SENDER OBJECT REFERENCE SYS-OBJECT.
001800 01 E OBJECT REFERENCE EVENTARGS.
001810 PROCEDURE DIVISION USING BY VALUE SENDER E.
001820    SET ENABLED TO PROP-ENABLED OF TIMER1.
001830    IF ENABLED = B'1' THEN
001840       SET PROP-ENABLED OF TIMER1 TO B'0'
001850       SET PROP-TEXT OF BUTTON1 TO "&Remind me"
001860    ELSE
001870       SET PROP-ENABLED OF TIMER1 TO B'1'
001880       SET PROP-TEXT OF BUTTON1 TO "&Stop it!"
001890    END-IF.
001900 END METHOD BUTTON1_CLICK.
001910*
001920 METHOD-ID. TIMER1_TICK AS "Timer1_Tick".
001930 DATA DIVISION.
001940 WORKING-STORAGE SECTION.
001950 01 REMINDER-OBJ OBJECT REFERENCE REMINDER.
001951 01 COMPONENT-OBJ OBJECT REFERENCE ICOMPONENT.
001960 LINKAGE SECTION.
001970 01 SENDER OBJECT REFERENCE SYS-OBJECT.
001980 01 E OBJECT REFERENCE EVENTARGS.
001990 PROCEDURE DIVISION USING BY VALUE SENDER E.
002000    INVOKE REMINDER "NEW" USING BY VALUE SELF RETURNING REMINDER-OBJ.
002010    INVOKE REMINDER-OBJ "Show".
002011    SET COMPONENT-OBJ TO REMINDER-OBJ AS ICOMPONENT.
002020    INVOKE COMPONENTS "Add" USING BY VALUE COMPONENT-OBJ.
002030 END METHOD TIMER1_TICK.
002040*
002050 METHOD-ID. EXIT_CLICK AS "Exit_Click".
002060 DATA DIVISION.
002070 LINKAGE SECTION.
002080 01 SENDER OBJECT REFERENCE SYS-OBJECT.
002090 01 E OBJECT REFERENCE EVENTARGS.
002100 PROCEDURE DIVISION USING BY VALUE SENDER E.
002110    INVOKE APPLICATION "Exit".
002120 END METHOD EXIT_CLICK.
002121*
002122 METHOD-ID. ABOUT_CLICK AS "About_Click".
002123 DATA DIVISION.
002124 WORKING-STORAGE SECTION.
002125 01 ABOUT-FORM OBJECT REFERENCE ABOUT.
002126 01 WIN32-WINDOW OBJECT REFERENCE IWIN32-WINDOW.
002127 LINKAGE SECTION.
002128 01 SENDER OBJECT REFERENCE SYS-OBJECT.
002129 01 E OBJECT REFERENCE EVENTARGS.
002130 PROCEDURE DIVISION USING BY VALUE SENDER E.
002131    INVOKE ABOUT "NEW" RETURNING ABOUT-FORM.
002132    SET WIN32-WINDOW TO SELF AS IWIN32-WINDOW.
002133    INVOKE ABOUT-FORM "ShowDialog" USING BY VALUE WIN32-WINDOW.
002134 END METHOD ABOUT_CLICK.
002141*
002142 METHOD-ID. DECREMENT-TIMER.
002143 DATA DIVISION.
```

```
002144 WORKING-STORAGE SECTION.
002145 01 INTERVAL COMP-2.
002146 PROCEDURE DIVISION.
002147    MOVE PROP-INTERVAL OF TIMER1 TO INTERVAL.
002148    IF INTERVAL > 1000 THEN
002149       SUBTRACT 1000 FROM INTERVAL
002150       MOVE INTERVAL TO PROP-INTERVAL OF TIMER1
002151    END-IF.
002152 END METHOD DECREMENT-TIMER.
002153*
002154 METHOD-ID. RESET-TIMER.
002155 PROCEDURE DIVISION.
002156    MOVE 7000 TO PROP-INTERVAL OF TIMER1.
002157 END METHOD RESET-TIMER.
002158*
002159 END OBJECT.
002160*
002161 END CLASS WALK.
002162
002170
```

ソースコード：Print1.cob

```
000010*================================================================
000020*「ＦＯＲＭＡＴ句なし印刷ファイルを使ったプログラム」
000030*    ＦＯＲＭＡＴ句なし印刷ファイルを使って様々な印字属性の文字を
000040*  プリンタ出力します。
000050*
000060* Copyright 2003-2010 FUJITSU LIMITED
000070*================================================================
000080 IDENTIFICATION         DIVISION.
000090 PROGRAM-ID.           PRINT1.
000100*－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－*
000110*【印刷サンプルでお見せする項目概要】                                *
000120*                                                                    *
000130*  １．ＦＣＢを使用した６ＬＰＩ／８ＬＰＩでの帳票サンプル印刷。      *
000140*                                                                    *
000150*    ◆Ａ４用紙を横向きに使用し、１ページ全ての行間隔を６ＬＰＩとした場合の帳票をイ  *
000160*      メージし、６ＬＰＩ／１０ＣＰＩフォーマットのスペーシングチャート形式のフォー  *
000170*      ムオーバレイと重畳印刷します。                                *
000180*                                                                    *
000190*    ◆Ａ４用紙を横向きに使用し、１ページ全ての行間隔を８ＬＰＩとした場合の帳票をイ  *
000200*      メージし、８ＬＰＩ／１０ＣＰＩフォーマットのスペーシングチャート形式のフォー  *
000210*      ムオーバレイと重畳印刷します。                                *
000220*                                                                    *
000230*  ２．ＣＨＡＲＡＣＴＥＲ　ＴＹＰＥ句で指定する様々な文字属性のサンプリング印刷。  *
000240*                                                                    *
000250*    ◆あらゆる文字サイズを使用した印刷                              *
000260*      １文字づつ３ポ、７．２ポ、９ポ、１２ポ、１８ポ、２４ポ、３６ポ、５０ポ、      *
000270*      ７２ポ、１００ポ、２００ポ、３００ポの文字サイズを指定します。          *
000280*      ※ここでは、文字ピッチ指定を省略し、文字サイズに合わせた最適な文字ピッチを  *
000290*        ＣＯＢＯＬランタイムシステムに自動算出・配置させます。      *
000300*                                                                    *
000310*    ◆あらゆる文字ピッチを使用した印刷                              *
000320*      文字ピッチ１ＣＰＩで１文字、２ＣＰＩで２文字、３ＣＰＩで３文字、          *
000330*      ５ＣＰＩで５文字、６ＣＰＩで６文字、７．５ＣＰＩで１５文字、              *
000340*      ２０ＣＰＩで２０文字、２４ＣＰＩで２４文字指定します。                  *
000350*      ※ここでは、文字サイズ指定を省略し、文字ピッチに合わせた最適な文字サイズを  *
000360*        ＣＯＢＯＬランタイムシステムに自動算出・印字させます。      *
000370*                                                                    *
000380*    ◆複数の書体を使用した印刷                                      *
000390*      ゴシック、ゴシック半角（文字形態半角）、明朝、明朝半角（文字形態半角）を１０  *
000400*      文字づつ２回繰り返します。                                    *
000410*      ※ここで指定した書体名は、以下の実行環境情報に関連づけられています。        *
000420*        ＠ＰｒｉｎｔｅｒＦｏｎｔＮａｍｅ＝（フォント名１，フォント名２）      *
000430*        「ＭＩＮＣＨＯＵ」／「ＭＩＮＣＨＯＵ－ＨＡＮＫＡＫＵ」を指定したデータ項目  *
000440*        は「フォント名１」に指定されたフォントを用いて印字され、              *
000450*        「ＧＯＴＨＩＣ」／「ＧＯＴＨＩＣ－ＨＡＮＫＡＫＵ」を指定したデータ項目は  *
000460*        「フォント名２」に指定されたフォントを用いて印字されます。            *
000470*        本サンプルでは、＠ＰｒｉｎｔｅｒＦｏｎｔＮａｍｅ＝（ＭＳ明朝，ＭＳゴシック）*
000480*        を指定しています。                                            *
000490*                                                                    *
000500*    ◆文字回転を使用した印刷                                        *
000510*      縦書き（反時計回りに９０度回転）、横書きを１０文字づつ繰り返します。      *
000520*                                                                    *
000530*    ◆あらゆる文字形態を使用した印刷                                *
000540*      全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、半各倍角の文字  *
000550*      形態指定を９文字づつ指定します。                              *
000560*                                                                    *
000570*    ◆上記５つの文字属性を組み合わせた印刷を行います。              *
000580*                                                                    *
000590*－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－*
000600 ENVIRONMENT            DIVISION.
000610 CONFIGURATION           SECTION.
000620 SPECIAL-NAMES.
000630*-------------------------------------------------------------------------------*
000640* 制御レコードを示す機能名とそれに対応する呼び名を宣言。
000650*-------------------------------------------------------------------------------*
000660     CTL IS PAGE-CNTL
```

```
000670*-------------------------------------------------------------------------------*
000680* ＦＣＢを使用した帳票サンプルに使用する機能名とその呼び名を宣言。
000690*-------------------------------------------------------------------------------*
000700* ６ＬＰＩの帳票に使用
000710* １２ポ／５ＣＰＩ／明朝／横書き／長体
000720    PRINTING MODE OOMIDASHI6 IS FOR ALL
000730                 IN   SIZE  012.0 POINT
000740                 AT   PITCH 05.00 CPI
000750                 WITH FONT  MINCHOU
000760                 AT   ANGLE 00 DEGREES
000770                 BY   FORM  F0102
000780* １２ポ／５ＣＰＩ／明朝／横書き／全角
000790    PRINTING MODE KOMIDASHI6 IS FOR ALL
000800                 IN   SIZE  012.0 POINT
000810                 AT   PITCH 05.00 CPI
000820                 WITH FONT  MINCHOU
000830                 AT   ANGLE 00 DEGREES
000840                 BY   FORM  F
000850* １０．５ポ／１０ＣＰＩ／ゴシック／横書き／全角
000860    PRINTING MODE MEISAIANK6 IS FOR ALL
000870                 IN   SIZE  010.5 POINT
000880                 AT   PITCH 10.00 CPI
000890                 WITH FONT  GOTHIC
000900                 AT   ANGLE 00 DEGREES
000910                 BY   FORM  F
000920* １０．５ポ／５ＣＰＩ／明朝／横書き／全角
000930    PRINTING MODE MEISAIJP16 IS FOR ALL
000940                 IN   SIZE  010.5 POINT
000950                 AT   PITCH 05.00 CPI
000960                 WITH FONT  MINCHOU
000970                 AT   ANGLE 00 DEGREES
000980                 BY   FORM  F
000990* ７．２ポ／１０ＣＰＩ／明朝／横書き／全角
001000    PRINTING MODE MEISAIJP26 IS FOR ALL
001010                 IN   SIZE  006.0 POINT
001020                 AT   PITCH 10.00 CPI
001030                 WITH FONT  MINCHOU
001040                 AT   ANGLE 00 DEGREES
001050                 BY   FORM  F0102
001060*
001070* ８ＬＰＩの帳票に使用
001080* ９ポ／５ＣＰＩ／明朝／横書き／長体
001090    PRINTING MODE OOMIDASHI8 IS FOR ALL
001100                 IN   SIZE  009.0 POINT
001110                 AT   PITCH 05.00 CPI
001120                 WITH FONT  MINCHOU
001130                 AT   ANGLE 00 DEGREES
001140                 BY   FORM  F0102
001150* ９ポ／５ＣＰＩ／明朝／横書き／全角
001160    PRINTING MODE KOMIDASHI8 IS FOR ALL
001170                 IN   SIZE  009.0 POINT
001180                 AT   PITCH 05.00 CPI
001190                 WITH FONT  MINCHOU
001200                 AT   ANGLE 00 DEGREES
001210                 BY   FORM  F
001220* ７．２ポ／１０ＣＰＩ／ゴシック／横書き／全角
001230    PRINTING MODE MEISAIANK8 IS FOR ALL
001240                 IN   SIZE  007.2 POINT
001250                 AT   PITCH 10.00 CPI
001260                 WITH FONT  GOTHIC
001270                 AT   ANGLE 00 DEGREES
001280                 BY   FORM  F
001290* ７．２ポ／５ＣＰＩ／明朝／横書き／全角
001300    PRINTING MODE MEISAIJP18 IS FOR ALL
001310                 IN   SIZE  007.2 POINT
001320                 AT   PITCH 05.00 CPI
001330                 WITH FONT  MINCHOU
001340                 AT   ANGLE 00 DEGREES
001350                 BY   FORM  F
001360* ７．２ポ／１０ＣＰＩ／明朝／横書き／全角
001370    PRINTING MODE MEISAIJP28 IS FOR ALL
001380                 IN   SIZE  007.2 POINT
001390                 AT   PITCH 10.00 CPI
001400                 WITH FONT  MINCHOU
001410                 AT   ANGLE 00 DEGREES
001420                 BY   FORM  F
001430*-------------------------------------------------------------------------------*
001440* 文字サイズを示す機能名とそれに対応する呼び名を宣言。
001450*-------------------------------------------------------------------------------*
```

```
001460* ３．０ポ
001470    PRINTING MODE PMSIZE01 IS FOR ALL
001480                  IN   SIZE   3   POINT
001490* ７．２ポ
001500    PRINTING MODE PMSIZE02 IS FOR ALL
001510                  IN   SIZE   7.2 POINT
001520* ９．０ポ
001530    PRINTING MODE PMSIZE03 IS FOR ALL
001540                  IN   SIZE   9   POINT
001550* １２．０ポ
001560    PRINTING MODE PMSIZE04 IS FOR ALL
001570                  IN   SIZE  12   POINT
001580* １８．０ポ
001590    PRINTING MODE PMSIZE05 IS FOR ALL
001600                  IN   SIZE  18   POINT
001610* ２４．０ポ
001620    PRINTING MODE PMSIZE06 IS FOR ALL
001630                  IN   SIZE  24   POINT
001640* ３６．０ポ
001650    PRINTING MODE PMSIZE07 IS FOR ALL
001660                  IN   SIZE  36   POINT
001670* ５０．０ポ
001680    PRINTING MODE PMSIZE08 IS FOR ALL
001690                  IN   SIZE  50   POINT
001700* ７２．０ポ
001710    PRINTING MODE PMSIZE09 IS FOR ALL
001720                  IN   SIZE  72   POINT
001730* １００．０ポ
001740    PRINTING MODE PMSIZE10 IS FOR ALL
001750                  IN   SIZE 100   POINT
001760* ２００．０ポ
001770    PRINTING MODE PMSIZE11 IS FOR ALL
001780                  IN   SIZE 200   POINT
001790* ３００．０ポ
001800    PRINTING MODE PMSIZE12 IS FOR ALL
001810                  IN   SIZE 300   POINT
001820*-------------------------------------------------------------------------------*
001830* 文字ピッチを示す機能名とそれに対応する呼び名を宣言。
001840*-------------------------------------------------------------------------------*
001850* １．００ＣＰＩ
001860    PRINTING MODE PMPICH01 IS FOR ALL
001870                  AT   PITCH  1   CPI
001880* ２．００ＣＰＩ
001890    PRINTING MODE PMPICH02 IS FOR ALL
001900                  AT   PITCH  2   CPI
001910* ３．００ＣＰＩ
001920    PRINTING MODE PMPICH03 IS FOR ALL
001930                  AT   PITCH  3   CPI
001940* ５．００ＣＰＩ
001950    PRINTING MODE PMPICH04 IS FOR ALL
001960                  AT   PITCH  5   CPI
001970* ６．００ＣＰＩ
001980    PRINTING MODE PMPICH05 IS FOR ALL
001990                  AT   PITCH  6   CPI
002000* ７．５０ＣＰＩ
002010    PRINTING MODE PMPICH06 IS FOR ALL
002020                  AT   PITCH  7.5 CPI
002030* ８．００ＣＰＩ
002040    PRINTING MODE PMPICH07 IS FOR ALL
002050                  AT   PITCH  8   CPI
002060* １０．００ＣＰＩ
002070    PRINTING MODE PMPICH08 IS FOR ALL
002080                  AT   PITCH 10   CPI
002090* １２．００ＣＰＩ
002100    PRINTING MODE PMPICH09 IS FOR ALL
002110                  AT   PITCH 12   CPI
002120* １５．００ＣＰＩ
002130    PRINTING MODE PMPICH10 IS FOR ALL
002140                  AT   PITCH 15   CPI
002150* ２０．００ＣＰＩ
002160    PRINTING MODE PMPICH11 IS FOR ALL
002170                  AT   PITCH 20   CPI
002180* ２４．００ＣＰＩ
002190    PRINTING MODE PMPICH12 IS FOR ALL
002200                  AT   PITCH 24   CPI
002210*-------------------------------------------------------------------------------*
002220* 書体を示す機能名とそれに対応する呼び名を宣言。
002230*-------------------------------------------------------------------------------*
```

```
002240* ＧＯＴＨＩＣ
002250    PRINTING MODE PMFONT01 IS FOR ALL
002260                  WITH FONT   GOTHIC
002270* ＧＯＴＨＩＣ－ＨＡＮＫＡＫＵ
002280    PRINTING MODE PMFONT02 IS FOR ALL
002290                  WITH FONT   GOTHIC-HANKAKU
002300                  BY   FORM   H
002310* ＭＩＮＣＨＯＵ
002320    PRINTING MODE PMFONT03 IS FOR ALL
002330                  WITH FONT   MINCHOU
002340* ＭＩＮＣＨＯＵ－ＨＡＮＫＡＫＵ
002350    PRINTING MODE PMFONT04 IS FOR ALL
002360                  WITH FONT   MINCHOU-HANKAKU
002370                  BY   FORM   H
002380*-----------------------------------------------------------------------------*
002390* 文字回転を示す機能名とそれに対応する呼び名を宣言。
002400*-----------------------------------------------------------------------------*
002410* 横書き
002420    PRINTING MODE PMANGL01 IS FOR ALL
002430                  AT   ANGLE  0   DEGREES
002440* 縦書き（反時計回りに９０度回転）
002450    PRINTING MODE PMANGL02 IS FOR ALL
002460                  AT   ANGLE  90  DEGREES
002470*-----------------------------------------------------------------------------*
002480* 文字形態を示す機能名とそれに対応する呼び名を宣言。
002490*-----------------------------------------------------------------------------*
002500* 全角
002510    PRINTING MODE PMFORM01 IS FOR ALL
002520                  BY   FORM   F
002530* 半角
002540    PRINTING MODE PMFORM02 IS FOR ALL
002550                  BY   FORM   H
002560                  WITH FONT   GOTHIC-HANKAKU
002570* 全角平体
002580    PRINTING MODE PMFORM03 IS FOR ALL
002590                  BY   FORM   F0201
002600* 半角平体
002610    PRINTING MODE PMFORM04 IS FOR ALL
002620                  BY   FORM   H0201
002630                  WITH FONT   GOTHIC-HANKAKU
002640* 全角長体
002650    PRINTING MODE PMFORM05 IS FOR ALL
002660                  BY   FORM   F0102
002670* 半角長体
002680    PRINTING MODE PMFORM06 IS FOR ALL
002690                  BY   FORM   H0102
002700                  WITH FONT   GOTHIC-HANKAKU
002710* 全角倍角
002720    PRINTING MODE PMFORM07 IS FOR ALL
002730                  BY   FORM   F0202
002740* 半角倍角
002750    PRINTING MODE PMFORM08 IS FOR ALL
002760                  BY   FORM   H0202
002770                  WITH FONT   GOTHIC-HANKAKU
002780*-----------------------------------------------------------------------------*
002790* あらゆる文字属性の組合せを示す機能名とそれに対応する呼び名を宣言。
002800*-----------------------------------------------------------------------------*
002810* ９ポ／８ＣＰＩ／ＭＩＮＣＨＯＵ／横書き／全角
002820    PRINTING MODE PRTMODE1 IS FOR ALL
002830                  IN   SIZE   9   POINT
002840                  AT   PITCH  8   CPI
002850                  WITH FONT   MINCHOU
002860                  AT   ANGLE  0   DEGREES
002870                  BY   FORM   F
002880* ７．２ポ／２０ＣＰＩ／ＧＯＴＨＩＣ－ＨＡＮＫＡＫＵ／横書き／半角
002890    PRINTING MODE PRTMODE2 IS FOR ALL
002900                  IN   SIZE   7.2 POINT
002910                  AT   PITCH  20  CPI
002920                  WITH FONT   GOTHIC-HANKAKU
002930                  AT   ANGLE  0   DEGREES
002940                  BY   FORM   H
002950* １２ポ／２．５ＣＰＩ／ＧＯＴＨＩＣ／縦書き／倍角
002960    PRINTING MODE PRTMODE3 IS FOR ALL
002970                  IN   SIZE   12  POINT
002980                  AT   PITCH  2.5 CPI
002990                  WITH FONT   GOTHIC
003000                  AT   ANGLE  90  DEGREES
003010                  BY   FORM   F0202.
```

```
003020*
003030 INPUT-OUTPUT           SECTION.
003040*
003050 FILE-CONTROL.
003060    SELECT 印刷ファイル ASSIGN TO PRINTER
003070          ORGANIZATION IS SEQUENTIAL
003080          ACCESS MODE  IS SEQUENTIAL
003090          FILE STATUS  IS 入出力状態.
003100*----------------------------------------------------------------------------*
003110 DATA                  DIVISION.
003120*
003130 FILE                  SECTION.
003140*
003150 FD  印刷ファイル.
003160 01  行レコード         PIC N(136).
003170 01  注釈レコード       PIC N(050).
003180 01  制御レコード       PIC X(100).
003190*
003200 WORKING-STORAGE         SECTION.
003210* ＦＣＢを使用した帳票サンプルに使用するデータ宣言
003220 01  大見出し６         CHARACTER TYPE   IS OOMIDASHI6
003230                       PRINTING POSITION IS 36
003240                       PIC N(020)
003250                       VALUE NC"ＦＣＢを使用した６ＬＰＩでの帳票サンプル".
003260 01  小見出し６         CHARACTER TYPE   IS KOMIDASHI6.
003270   02  FILLER          PIC N(002)
003280                       VALUE NC"項番".
003290   02  FILLER          PRINTING POSITION IS 11
003300                       PIC N(005)
003310                       VALUE NC"商品コード".
003320   02  FILLER          PRINTING POSITION IS 27
003330                       PIC N(003)
003340                       VALUE NC"商品名".
003350   02  FILLER          PRINTING POSITION IS 57
003360                       PIC N(002)
003370                       VALUE NC"単価".
003380   02  FILLER          PRINTING POSITION IS 67
003390                       PIC N(004)
003400                       VALUE NC"出荷本数".
003410   02  FILLER          PRINTING POSITION IS 81
003420                       PIC N(004)
003430                       VALUE NC"在庫数量".
003440   02  FILLER          PRINTING POSITION IS 95
003450                       PIC N(002)
003460                       VALUE NC"備考".
003470 01  明細６.
003480   02  項番            CHARACTER TYPE IS MEISAIANK6
003490                       PIC 9(004)
003500                       VALUE 9999.
003510   02  商品コード       CHARACTER TYPE IS MEISAIANK6
003520                       PRINTING POSITION IS 11
003530                       PIC X(010)
003540                       VALUE "XXXXXXXXXX".
003550   02  商品名           CHARACTER TYPE IS MEISAIJP16
003560                       PRINTING POSITION IS 27
003570                       PIC N(010)
003580                       VALUE NC"ＮＮＮＮＮＮＮＮＮＮ".
003590   02  単価            CHARACTER TYPE IS MEISAIANK6
003600                       PRINTING POSITION IS 53
003610                       PIC X(008)
003620                       VALUE "¥9999999".
003630   02  出荷本数         CHARACTER TYPE IS MEISAIANK6
003640                       PRINTING POSITION IS 67
003650                       PIC 9(008)
003660                       VALUE 99999999.
003670   02  在庫数量         CHARACTER TYPE IS MEISAIANK6
003680                       PRINTING POSITION IS 81
003690                       PIC 9(008)
003700                       VALUE 99999999.
003710   02  備考            CHARACTER TYPE IS MEISAIJP26
003720                       PRINTING POSITION IS 95
003730                       PIC N(016)
003740                       VALUE NC"この行は６ＬＰＩで印字されます。".
003750 01  大見出し８         CHARACTER TYPE   IS OOMIDASHI8
003760                       PRINTING POSITION IS 36
003770                       PIC N(020)
003780                       VALUE NC"ＦＣＢを使用した８ＬＰＩでの帳票サンプル".
```

```
003790 01  小見出し８          CHARACTER TYPE   IS KOMIDASHI8.
003800   02  FILLER            PIC N(002)
003810                        VALUE NC"項番".
003820   02  FILLER            PRINTING POSITION IS 11
003830                        PIC N(005)
003840                        VALUE NC"商品コード".
003850   02  FILLER            PRINTING POSITION IS 27
003860                        PIC N(003)
003870                        VALUE NC"商品名".
003880   02  FILLER            PRINTING POSITION IS 57
003890                        PIC N(002)
003900                        VALUE NC"単価".
003910   02  FILLER            PRINTING POSITION IS 67
003920                        PIC N(004)
003930                        VALUE NC"出荷本数".
003940   02  FILLER            PRINTING POSITION IS 81
003950                        PIC N(004)
003960                        VALUE NC"在庫数量".
003970   02  FILLER            PRINTING POSITION IS 95
003980                        PIC N(002)
003990                        VALUE NC"備考".
004000 01  明細８.
004010   02  項番              CHARACTER TYPE IS MEISAIANK8
004020                        PIC 9(004)
004030                        VALUE 9999.
004040   02  商品コード        CHARACTER TYPE IS MEISAIANK8
004050                        PRINTING POSITION IS 11
004060                        PIC X(010)
004070                        VALUE "XXXXXXXXXX".
004080   02  商品名            CHARACTER TYPE IS MEISAIJP18
004090                        PRINTING POSITION IS 27
004100                        PIC N(010)
004110                        VALUE NC"ＮＮＮＮＮＮＮＮＮＮ".
004120   02  単価              CHARACTER TYPE IS MEISAIANK8
004130                        PRINTING POSITION IS 53
004140                        PIC X(008)
004150                        VALUE "¥9999999".
004160   02  出荷本数          CHARACTER TYPE IS MEISAIANK8
004170                        PRINTING POSITION IS 67
004180                        PIC 9(008)
004190                        VALUE 99999999.
004200   02  在庫数量          CHARACTER TYPE IS MEISAIANK8
004210                        PRINTING POSITION IS 81
004220                        PIC 9(008)
004230                        VALUE 99999999.
004240   02  備考              CHARACTER TYPE IS MEISAIJP28
004250                        PRINTING POSITION IS 95
004260                        PIC N(016)
004270                        VALUE NC"この行は８ＬＰＩで印字されます。".
004280*
004290* 注釈として使用するデータ宣言
004300 01  見出しデータ１       PIC N(050) CHARACTER TYPE IS MODE-1
004310                        VALUE NC"富士通ＮｅｔＣＯＢＯＬ　ＦＯＲＭＡＴ句なし印刷ファイル".
004320 01  見出しデータ２       PIC N(050) CHARACTER TYPE IS PMPICH01
004330                        VALUE NC"各種文字属性印字サンプル集".
004340 01  注釈データ１         PIC N(050) CHARACTER TYPE IS MODE-1
004350                        VALUE NC"◆あらゆる文字サイズを使用した印刷例。".
004360 01  注釈データ２         PIC N(050) CHARACTER TYPE IS MODE-1
004370                        VALUE NC"◆あらゆる文字ピッチを使用した印刷例。".
004380 01  注釈データ３         PIC N(050) CHARACTER TYPE IS MODE-1
004390                        VALUE NC"◆複数の書体を使用した印刷例。".
004400 01  注釈データ４         PIC N(100) CHARACTER TYPE IS MODE-1
004410                        VALUE NC"◆文字回転を使用した印刷例。".
004420 01  注釈データ５         PIC N(100) CHARACTER TYPE IS MODE-1
004430                        VALUE NC"◆あらゆる文字形態を使用した印刷例。".
004440 01  注釈データ６         PIC N(100) CHARACTER TYPE IS MODE-1
004450                        VALUE NC"◆様々な文字属性を組み合わせた（混在）印刷例。".
004460* 文字サイズを指定したデータ宣言
004470 01  文字サイズデータ.
004480   02                   PIC N(001) CHARACTER TYPE IS PMSIZE01
004490                        VALUE NC"あ".
004500   02                   PIC N(001) CHARACTER TYPE IS PMSIZE02
004510                        VALUE NC"あ".
004520   02                   PIC N(001) CHARACTER TYPE IS PMSIZE03
004530                        VALUE NC"あ".
```

```
004540   02                  PIC N(001) CHARACTER TYPE IS PMSIZE04
004550                      VALUE NC"あ".
004560   02                  PIC N(001) CHARACTER TYPE IS PMSIZE05
004570                      VALUE NC"あ".
004580   02                  PIC N(001) CHARACTER TYPE IS PMSIZE06
004590                      VALUE NC"あ".
004600   02                  PIC N(001) CHARACTER TYPE IS PMSIZE07
004610                      VALUE NC"あ".
004620   02                  PIC N(001) CHARACTER TYPE IS PMSIZE08
004630                      VALUE NC"あ".
004640   02                  PIC N(001) CHARACTER TYPE IS PMSIZE09
004650                      VALUE NC"あ".
004660   02                  PIC N(001) CHARACTER TYPE IS PMSIZE10
004670                      VALUE NC"あ".
004680   02                  PIC N(001) CHARACTER TYPE IS PMSIZE11
004690                      VALUE NC"あ".
004700   02                  PIC N(001) CHARACTER TYPE IS PMSIZE12
004710                      VALUE NC"あ".
004720* 文字ピッチを指定したデータ宣言
004730 01  文字ピッチデータ.
004740   02                  PIC N(001) CHARACTER TYPE IS PMPICH01
004750                      VALUE    NC"い".
004760   02                  PIC N(002) CHARACTER TYPE IS PMPICH02
004770                      VALUE ALL NC"い".
004780   02                  PIC N(003) CHARACTER TYPE IS PMPICH03
004790                      VALUE ALL NC"い".
004800   02                  PIC N(005) CHARACTER TYPE IS PMPICH04
004810                      VALUE ALL NC"い".
004820   02                  PIC N(006) CHARACTER TYPE IS PMPICH05
004830                      VALUE ALL NC"い".
004840   02                  PIC N(015) CHARACTER TYPE IS PMPICH06
004850                      VALUE ALL NC"い".
004860   02                  PIC N(008) CHARACTER TYPE IS PMPICH07
004870                      VALUE ALL NC"い".
004880   02                  PIC N(010) CHARACTER TYPE IS PMPICH08
004890                      VALUE ALL NC"い".
004900   02                  PIC N(012) CHARACTER TYPE IS PMPICH09
004910                      VALUE ALL NC"い".
004920   02                  PIC N(015) CHARACTER TYPE IS PMPICH10
004930                      VALUE ALL NC"い".
004940   02                  PIC N(020) CHARACTER TYPE IS PMPICH11
004950                      VALUE ALL NC"い".
004960   02                  PIC N(024) CHARACTER TYPE IS PMPICH12
004970                      VALUE ALL NC"い".
004980* 書体を指定したデータ宣言
004990 01  文字フォントデータ.
005000   02                  PIC N(010) CHARACTER TYPE IS PMFONT01
005010                      VALUE ALL NC"う".
005020   02                  PIC N(010) CHARACTER TYPE IS PMFONT02
005030                      VALUE ALL NC"う".
005040   02                  PIC N(010) CHARACTER TYPE IS PMFONT03
005050                      VALUE ALL NC"う".
005060   02                  PIC N(010) CHARACTER TYPE IS PMFONT04
005070                      VALUE ALL NC"う".
005080   02                  PIC N(010) CHARACTER TYPE IS PMFONT01
005090                      VALUE ALL NC"う".
005100   02                  PIC N(010) CHARACTER TYPE IS PMFONT02
005110                      VALUE ALL NC"う".
005120   02                  PIC N(010) CHARACTER TYPE IS PMFONT03
005130                      VALUE ALL NC"う".
005140   02                  PIC N(010) CHARACTER TYPE IS PMFONT04
005150                      VALUE ALL NC"う".
005160* 文字回転を指定したデータ宣言
005170 01  文字回転データ.
005180   02                  PIC N(010) CHARACTER TYPE IS PMANGL01
005190                      VALUE ALL NC"え".
005200   02                  PIC N(010) CHARACTER TYPE IS PMANGL02
005210                      VALUE ALL NC"え".
005220   02                  PIC N(010) CHARACTER TYPE IS PMANGL01
005230                      VALUE ALL NC"え".
005240   02                  PIC N(010) CHARACTER TYPE IS PMANGL02
005250                      VALUE ALL NC"え".
005260   02                  PIC N(010) CHARACTER TYPE IS PMANGL01
005270                      VALUE ALL NC"え".
005280   02                  PIC N(010) CHARACTER TYPE IS PMANGL02
```

```
005290                      VALUE ALL NC"え".
005300   02                  PIC N(010) CHARACTER TYPE IS PMANGL01
005310                      VALUE ALL NC"え".
005320   02                  PIC N(010) CHARACTER TYPE IS PMANGL02
005330                      VALUE ALL NC"え".
005340* 文字形態を指定したデータ宣言
005350 01  文字形態データ.
005360   02                  PIC N(009) CHARACTER TYPE IS PMFORM01
005370                      VALUE ALL NC"お".
005380   02                  PIC N(009) CHARACTER TYPE IS PMFORM02
005390                      VALUE ALL NC"お".
005400   02                  PIC N(009) CHARACTER TYPE IS PMFORM03
005410                      VALUE ALL NC"お".
005420   02                  PIC N(009) CHARACTER TYPE IS PMFORM04
005430                      VALUE ALL NC"お".
005440   02                  PIC N(009) CHARACTER TYPE IS PMFORM05
005450                      VALUE ALL NC"お".
005460   02                  PIC N(009) CHARACTER TYPE IS PMFORM06
005470                      VALUE ALL NC"お".
005480   02                  PIC N(009) CHARACTER TYPE IS PMFORM07
005490                      VALUE ALL NC"お".
005500   02                  PIC N(009) CHARACTER TYPE IS PMFORM08
005510                      VALUE ALL NC"お".
005520* 様々な文字属性を組合せ指定したデータ宣言
005530 01  文字属性組合せデータ.
005540   02                  PIC N(010) CHARACTER TYPE IS PRTMODE1
005550                      VALUE ALL NC"か".
005560   02                  PIC N(010) CHARACTER TYPE IS PRTMODE2
005570                      VALUE ALL NC"か".
005580   02                  PIC N(010) CHARACTER TYPE IS PRTMODE3
005590                      VALUE ALL NC"か".
005600   02                  PIC N(010) CHARACTER TYPE IS PRTMODE1
005610                      VALUE ALL NC"か".
005620   02                  PIC N(010) CHARACTER TYPE IS PRTMODE2
005630                      VALUE ALL NC"か".
005640   02                  PIC N(010) CHARACTER TYPE IS PRTMODE3
005650                      VALUE ALL NC"か".
005660*Ｉ制御レコードのデータ宣言
005670 01  Ｉ制御データ.
005680   02  レコード識別子    PIC X(002) VALUE "I1".
005690   02  モード            PIC X(001) VALUE "1".
005700   02  オーバレイ.
005710     03  オーバレイ名    PIC X(004) VALUE SPACE.
005720     03  焼付け回数      PIC 9(003) VALUE 0.
005730   02  複写数            PIC 9(003) VALUE 0.
005740   02  ＦＣＢ名          PIC X(004) VALUE SPACE.
005750   02  帳票定義体名      PIC X(008) VALUE SPACE.
005760   02                    PIC X(030) VALUE SPACE.
005770   02  印刷形式          PIC X(002) VALUE SPACE.
005780   02  用紙サイズ        PIC X(003) VALUE SPACE.
005790   02                    PIC X(004) VALUE SPACE.
005800   02  印刷面            PIC X(001) VALUE SPACE.
005810   02  印刷開始位置づけ面 PIC X(001) VALUE SPACE.
005820   02  印字禁止域        PIC X(001) VALUE SPACE.
005830   02  綴じ代方向.
005840     03  ポート時表面    PIC X(001) VALUE SPACE.
005850     03  ポート時裏面    PIC X(001) VALUE SPACE.
005860     03  ランド時表面    PIC X(001) VALUE SPACE.
005870     03  ランド時裏面    PIC X(001) VALUE SPACE.
005880   02  綴じ代幅          PIC X(004) VALUE SPACE.
005890   02  印字開始原点位置.
005900     03  ポート時Ｘ座標  PIC X(004) VALUE SPACE.
005910     03  ポート時Ｙ座標  PIC X(004) VALUE SPACE.
005920     03  ランド時Ｘ座標  PIC X(004) VALUE SPACE.
005930     03  ランド時Ｙ座標  PIC X(004) VALUE SPACE.
005940   02  文書情報          PIC X(004) VALUE SPACE.
005950   02  予約域            PIC X(005) VALUE SPACE.
005960* エラーカウンタとして使用するデータ宣言
005970 01  エラーカウンタ      PIC 9(004)       VALUE ZERO.
005980* 入出力状態として使用するデータ宣言
005990 77  入出力状態          PIC X(002)       VALUE ZERO.
006000*-----------------------------------------------------------------------------*
006010 PROCEDURE             DIVISION.
```

```
006020*
006030* 印刷ファイルのＯＰＥＮ
006040    OPEN OUTPUT 印刷ファイル.
006050    IF 入出力状態 NOT = ZERO THEN
006060      ADD 1 TO エラーカウンタ
006070    END-IF.
006080*
006090*-------------------------------------------------------------------------------*
006100* １．ＦＣＢを使用した帳票印刷サンプルの出力処理                                *
006110*-------------------------------------------------------------------------------*
006120*
006130* ６ＬＰＩの帳票印刷処理
006140*
006150* オーバレイ名”Ａ４Ｌ６”（ＫＯＬ５Ａ４Ｌ６．ＯＶＤ）設定
006160* ６ＬＰＩ／１０ＣＰＩのスペーシングチャートイメージのオーバレイを出力
006170    MOVE "A4L6" TO オーバレイ名.
006180* 焼き付け回数１回設定
006190    MOVE １     TO 焼付け回数.
006200* 複写枚数１枚設定
006210    MOVE １     TO 複写数.
006220* ＦＣＢ名”Ａ４Ｌ６”（ＦＣＢＡ４Ｌ６）設定
006230    MOVE "A4L6" TO ＦＣＢ名.
006240* 印刷形式ランドスケープモード(横向き)設定
006250    MOVE "L"    TO 印刷形式.
006260* 用紙サイズＡ４設定
006270    MOVE "A4"   TO 用紙サイズ.
006280* 片面印刷設定
006290    MOVE "F"    TO 印刷面.
006300* 表面位置づけ設定
006310    MOVE "F"    TO 印刷開始位置づけ面.
006320* 文書情報名”ＤＯＣ１”（＠ＣＢＲ＿＿ＤｏｃｕｍｅｎｔＮａｍｅ＿＿ＤＯＣ１）設定
006330    MOVE "DOC1" TO 文書情報.
006340* Ｉ制御レコードを出力することによりページ属性を設定
006350    WRITE 制御レコード FROM Ｉ制御データ
006360                       AFTER ADVANCING PAGE-CNTL.
006370    IF 入出力状態 NOT = ZERO THEN
006380      ADD 1 TO エラーカウンタ
006390    END-IF.
006400*
006410* 大見出し出力
006420    MOVE SPACE  TO 注釈レコード.
006430    WRITE 注釈レコード AFTER ADVANCING PAGE.
006440    IF 入出力状態 NOT = ZERO THEN
006450      ADD 1 TO エラーカウンタ
006460    END-IF.
006470*
006480* 小見出し出力
006490    WRITE 行レコード FROM 大見出し６
006500                       AFTER ADVANCING 1 LINE.
006510    IF 入出力状態 NOT = ZERO THEN
006520      ADD 1 TO エラーカウンタ
006530    END-IF.
006540*
006550* 明細出力
006560    WRITE 行レコード FROM 小見出し６
006570                       AFTER ADVANCING 2 LINES.
006580    IF 入出力状態 NOT = ZERO THEN
006590      ADD 1 TO エラーカウンタ
006600    END-IF.
006610    WRITE 行レコード FROM 明細６
006620                       AFTER ADVANCING 2 LINES
006630    PERFORM 40 TIMES
006640      WRITE 行レコード FROM 明細６
006650                       AFTER ADVANCING 1 LINE
006660      IF 入出力状態 NOT = ZERO THEN
006670        ADD 1 TO エラーカウンタ
006680      END-IF
006690    END-PERFORM.
006700*
006710* ８ＬＰＩの帳票印刷処理
006720*
006730* オーバレイ名”Ａ４Ｌ８”（ＫＯＬ５Ａ４Ｌ８．ＯＶＤ）設定
006740* ８ＬＰＩ／１０ＣＰＩのスペーシングチャートイメージのオーバレイを出力
006750    MOVE "A4L8" TO オーバレイ名.
```

```
006760* 焼き付け回数１回設定
006770    MOVE １     TO 焼付け回数.
006780* 複写枚数１枚設定
006790    MOVE １     TO 複写数.
006800* ＦＣＢ名”Ａ４Ｌ６”（ＦＣＢＡ４Ｌ６）設定
006810    MOVE "A4L8" TO ＦＣＢ名.
006820* 印刷形式ランドスケープモード(横向き)設定
006830    MOVE "L"    TO 印刷形式.
006840* 用紙サイズＡ４設定
006850    MOVE "A4"   TO 用紙サイズ.
006860* 片面印刷設定
006870    MOVE "F"    TO 印刷面.
006880* 表面位置づけ設定
006890    MOVE "F"    TO 印刷開始位置づけ面.
006900* 文書情報名”ＤＯＣ１”（＠ＣＢＲ＿ＤｏｃｕｍｅｎｔＮａｍｅ＿ＤＯＣ１）設定
006910    MOVE "DOC1" TO 文書情報.
006920* Ｉ制御レコードを出力することによりページ属性を設定
006930    WRITE 制御レコード FROM Ｉ制御データ
006940                       AFTER ADVANCING PAGE-CNTL.
006950    IF 入出力状態 NOT = ZERO THEN
006960      ADD 1 TO エラーカウンタ
006970    END-IF.
006980*
006990* 大見出し出力
007000    MOVE SPACE  TO 注釈レコード.
007010    WRITE 注釈レコード AFTER ADVANCING PAGE.
007020    IF 入出力状態 NOT = ZERO THEN
007030      ADD 1 TO エラーカウンタ
007040    END-IF.
007050    WRITE 行レコード FROM 大見出し８
007060                       AFTER ADVANCING 1 LINE.
007070    IF 入出力状態 NOT = ZERO THEN
007080      ADD 1 TO エラーカウンタ
007090    END-IF.
007100*
007110* 小見出し出力
007120    WRITE 行レコード FROM 小見出し８
007130                       AFTER ADVANCING 2 LINES.
007140    IF 入出力状態 NOT = ZERO THEN
007150      ADD 1 TO エラーカウンタ
007160    END-IF.
007170*
007180* 明細出力
007190    WRITE 行レコード FROM 明細８
007200                       AFTER ADVANCING 2 LINES
007210    PERFORM 55 TIMES
007220      WRITE 行レコード FROM 明細８
007230                       AFTER ADVANCING 1 LINE
007240      IF 入出力状態 NOT = ZERO THEN
007250        ADD 1 TO エラーカウンタ
007260      END-IF
007270    END-PERFORM.
007280*
007290*-------------------------------------------------------------------------------*
007300* ２．各種文字属性印字サンプル集の出力処理                                      *
007310*-------------------------------------------------------------------------------*
007320*
007330* オーバレイ名”Ｂ４ＯＶ”（ＫＯＬ５Ｂ４ＯＶ．ＯＶＤ）設定
007340    MOVE "B4OV" TO オーバレイ名.
007350* 焼き付け回数１回設定
007360    MOVE １     TO 焼付け回数.
007370* 複写枚数１枚設定
007380    MOVE １     TO 複写数.
007390* ＦＣＢ名”ＬＰＩ６”（ＦＣＢ６ＬＰＩ）設定
007400    MOVE "LPI6" TO ＦＣＢ名.
007410* 印刷形式ランドスケープモード(横向き)設定
007420    MOVE "L"    TO 印刷形式.
007430* 用紙サイズＢ４設定
007440    MOVE "B4"   TO 用紙サイズ.
007450* 片面印刷設定
007460    MOVE "F"    TO 印刷面.
007470* 表面位置づけ設定
007480    MOVE "F"    TO 印刷開始位置づけ面.
```

```
007490* 文書情報名”ＤＯＣ２”（＠ＣＢＲ＿ＤｏｃｕｍｅｎｔＮａｍｅ＿ＤＯＣ２）設定
007500    MOVE "DOC2" TO 文書情報.
007510* Ｉ制御レコードを出力することによりページ属性を設定
007520    WRITE 制御レコード FROM Ｉ制御データ
007530                      AFTER ADVANCING PAGE-CNTL.
007540    IF 入出力状態 NOT = ZERO THEN
007550      ADD 1 TO エラーカウンタ
007560    END-IF.
007570*
007580* 見出し出力
007590    WRITE 注釈レコード FROM 見出しデータ１
007600                      AFTER ADVANCING PAGE.
007610    IF 入出力状態 NOT = ZERO THEN
007620      ADD 1 TO エラーカウンタ
007630    END-IF.
007640    WRITE 注釈レコード FROM 見出しデータ２
007650                      AFTER ADVANCING 30 LINES.
007660    IF 入出力状態 NOT = ZERO THEN
007670      ADD 1 TO エラーカウンタ
007680    END-IF.
007690*
007700* 文字サイズサンプル出力
007710    WRITE 注釈レコード FROM 注釈データ１
007720                      AFTER ADVANCING PAGE.
007730    WRITE 行レコード   FROM 文字サイズデータ
007740                      AFTER ADVANCING 40 LINES.
007750    IF 入出力状態 NOT = ZERO THEN
007760      ADD 1 TO エラーカウンタ
007770    END-IF.
007780*
007790* 文字ピッチサンプル出力
007800    WRITE 注釈レコード FROM 注釈データ２
007810                      AFTER ADVANCING PAGE.
007820    PERFORM 5 TIMES
007830      WRITE 行レコード FROM 文字ピッチデータ
007840                      AFTER ADVANCING 10 LINES
007850      IF 入出力状態 NOT = ZERO THEN
007860        ADD 1 TO エラーカウンタ
007870      END-IF
007880    END-PERFORM.
007890*
007900* 文字書体サンプル出力
007910    WRITE 注釈レコード FROM 注釈データ３
007920                      AFTER ADVANCING PAGE.
007930    PERFORM 18 TIMES
007940      WRITE 行レコード FROM 文字フォントデータ
007950                      AFTER ADVANCING 3 LINES
007960      IF 入出力状態 NOT = ZERO THEN
007970        ADD 1 TO エラーカウンタ
007980      END-IF
007990    END-PERFORM.
008000*
008010* 文字回転サンプル出力
008020    WRITE 注釈レコード FROM 注釈データ４
008030                      AFTER ADVANCING PAGE.
008040    PERFORM 18 TIMES
008050      WRITE 行レコード FROM 文字回転データ
008060                      AFTER ADVANCING 3 LINES
008070      IF 入出力状態 NOT = ZERO THEN
008080        ADD 1 TO エラーカウンタ
008090      END-IF
008100    END-PERFORM.
008110*
008120* 文字形態データ出力
008130    WRITE 注釈レコード FROM 注釈データ５
008140                      AFTER ADVANCING PAGE.
008150    PERFORM 18 TIMES
008160      WRITE 行レコード FROM 文字形態データ
008170                      AFTER ADVANCING 3 LINES
008180      IF 入出力状態 NOT = ZERO THEN
008190        ADD 1 TO エラーカウンタ
008200      END-IF
008210    END-PERFORM.
008220*
008230* 文字属性組合せサンプル出力
```

```
008240    WRITE 注釈レコード FROM 注釈データ６
008250                  AFTER ADVANCING PAGE.
008260    PERFORM 18 TIMES
008270      WRITE 行レコード FROM 文字属性組合せデータ
008280                    AFTER ADVANCING 3 LINES
008290      IF 入出力状態 NOT = ZERO THEN
008300        ADD 1 TO エラーカウンタ
008310      END-IF
008320    END-PERFORM.
008330*
008340* 印刷ファイルＣＬＯＳＥ
008350    CLOSE 印刷ファイル.
008360    IF 入出力状態 NOT = ZERO THEN
008370      ADD 1 TO エラーカウンタ
008380    END-IF.
008390* エラー判定
008400    IF エラーカウンタ > ZERO THEN
008410      DISPLAY NC"エラーが発生しました。"
008420      DISPLAY NC"実行時メーッセージを確認し、エラーの原因を取り除いてください。"
008430      DISPLAY SPACE
008440    END-IF.
008450*
008460    STOP RUN.
008470
```

ソースコード：Print2.cob

```
000010*================================================================
000020*「ＦＯＲＭＡＴ句付き印刷ファイルを使ったプログラム」
000030*    ＦＯＲＭＡＴ句付き印刷ファイルを使って様々な印字属性の文字を
000040*  プリンタ出力します。
000050*
000060*================================================================
000070****  注意事項  ****
000080*================================================================
000090*    このプログラムを実行するためには、ＭｅＦｔが必要です。
000100*
000110*     Copyright 2003-2010 FUJITSU LIMITED
000120*================================================================
000130 IDENTIFICATION         DIVISION.
000140 PROGRAM-ID.           PRINT2.
000150*----------------------------------------------------------------*
000160*【印刷サンプルでお見せする項目概要】                            *
000170*                                                                *
000180*  １．ＦＣＢを使用した６ＬＰＩ／８ＬＰＩでの帳票サンプル印刷。  *
000190*                                                                *
000200*    ◆Ａ４用紙を横向きに使用し、１ページ全ての行間隔を６ＬＰＩとした場合の帳票をイ  *
000210*      メージし、６ＬＰＩ／１０ＣＰＩフォーマットのスペーシングチャート形式のフォー  *
000220*      ムオーバレイと重畳印刷します。                            *
000230*                                                                *
000240*    ◆Ａ４用紙を横向きに使用し、１ページ全ての行間隔を８ＬＰＩとした場合の帳票をイ  *
000250*      メージし、８ＬＰＩ／１０ＣＰＩフォーマットのスペーシングチャート形式のフォー  *
000260*      ムオーバレイと重畳印刷します。                            *
000270*                                                                *
000280*  ２．ＣＨＡＲＡＣＴＥＲ　ＴＹＰＥ句で指定する様々な文字属性のサンプリング印刷。  *
000290*                                                                *
000300*    ◆あらゆる文字サイズを使用した印刷                          *
000310*      １文字づつ３ポ、７．２ポ、９ポ、１２ポ、１８ポ、２４ポ、３６ポ、５０ポ、      *
000320*      ７２ポ、１００ポ、２００ポ、３００ポの文字サイズを指定します。            *
000330*      ※ここでは、文字ピッチ指定を省略し、文字サイズに合わせた最適な文字ピッチを  *
000340*        ＣＯＢＯＬランタイムシステムに自動算出・配置させます。  *
000350*                                                                *
000360*    ◆あらゆる文字ピッチを使用した印刷                          *
000370*      文字ピッチ１ＣＰＩで１文字、２ＣＰＩで２文字、３ＣＰＩで３文字、            *
000380*      ５ＣＰＩで５文字、６ＣＰＩで６文字、７．５ＣＰＩで１５文字、              *
000390*      ２０ＣＰＩで２０文字、２４ＣＰＩで２４文字指定します。                    *
000400*      ※ここでは、文字サイズ指定を省略し、文字ピッチに合わせた最適な文字サイズを  *
000410*        ＣＯＢＯＬランタイムシステムに自動算出・印字させます。  *
000420*                                                                *
000430*    ◆複数の書体を使用した印刷                                  *
000440*      ゴシック、ゴシック半角（文字形態半角）、明朝、明朝半角（文字形態半角）を１０  *
000450*      文字づつ２回繰り返します。                                *
000460*                                                                *
000470*    ◆文字回転を使用した印刷                                    *
000480*      縦書き（反時計回りに９０度回転）、横書きを１０文字づつ繰り返します。        *
000490*                                                                *
000500*    ◆あらゆる文字形態を使用した印刷                            *
000510*      全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、半各倍角の文字  *
000520*      形態指定を９文字づつ指定します。                          *
000530*                                                                *
000540*    ◆上記５つの文字属性を組み合わせた印刷を行います。          *
000550*                                                                *
000560*----------------------------------------------------------------*
000570 ENVIRONMENT            DIVISION.
000580 CONFIGURATION           SECTION.
000590 SPECIAL-NAMES.
000600*--------------------------------------------------------------------------------*
000610* ＦＣＢを使用した帳票サンプルに使用する機能名とその呼び名を宣言。
000620*--------------------------------------------------------------------------------*
000630* ６ＬＰＩの帳票に使用
000640* １２ポ／５ＣＰＩ／明朝／横書き／長体
000650    PRINTING MODE OOMIDASHI6 IS FOR ALL
000660                 IN   SIZE  012.0 POINT
```

```
000670                AT   PITCH 05.00 CPI
000680                WITH FONT  MINCHOU
000690                AT   ANGLE 00 DEGREES
000700                BY   FORM  F0102
000710* １２ポ／５ＣＰＩ／明朝／横書き／全角
000720    PRINTING MODE KOMIDASHI6 IS FOR ALL
000730                IN   SIZE  012.0 POINT
000740                AT   PITCH 05.00 CPI
000750                WITH FONT  MINCHOU
000760                AT   ANGLE 00 DEGREES
000770                BY   FORM  F
000780* １０．５ポ／１０ＣＰＩ／ゴシック／横書き／全角
000790    PRINTING MODE MEISAIANK6 IS FOR ALL
000800                IN   SIZE  010.5 POINT
000810                AT   PITCH 10.00 CPI
000820                WITH FONT  GOTHIC
000830                AT   ANGLE 00 DEGREES
000840                BY   FORM  F
000850* １０．５ポ／５ＣＰＩ／明朝／横書き／全角
000860    PRINTING MODE MEISAIJP16 IS FOR ALL
000870                IN   SIZE  010.5 POINT
000880                AT   PITCH 05.00 CPI
000890                WITH FONT  MINCHOU
000900                AT   ANGLE 00 DEGREES
000910                BY   FORM  F
000920* ７．２ポ／１０ＣＰＩ／明朝／横書き／全角
000930    PRINTING MODE MEISAIJP26 IS FOR ALL
000940                IN   SIZE  006.0 POINT
000950                AT   PITCH 10.00 CPI
000960                WITH FONT  MINCHOU
000970                AT   ANGLE 00 DEGREES
000980                BY   FORM  F0102
000990*
001000* ８ＬＰＩの帳票に使用
001010* ９ポ／５ＣＰＩ／明朝／横書き／長体
001020    PRINTING MODE OOMIDASHI8 IS FOR ALL
001030                IN   SIZE  009.0 POINT
001040                AT   PITCH 05.00 CPI
001050                WITH FONT  MINCHOU
001060                AT   ANGLE 00 DEGREES
001070                BY   FORM  F0102
001080* ９ポ／５ＣＰＩ／明朝／横書き／全角
001090    PRINTING MODE KOMIDASHI8 IS FOR ALL
001100                IN   SIZE  009.0 POINT
001110                AT   PITCH 05.00 CPI
001120                WITH FONT  MINCHOU
001130                AT   ANGLE 00 DEGREES
001140                BY   FORM  F
001150* ７．２ポ／１０ＣＰＩ／ゴシック／横書き／全角
001160    PRINTING MODE MEISAIANK8 IS FOR ALL
001170                IN   SIZE  007.2 POINT
001180                AT   PITCH 10.00 CPI
001190                WITH FONT  GOTHIC
001200                AT   ANGLE 00 DEGREES
001210                BY   FORM  F
001220* ７．２ポ／５ＣＰＩ／明朝／横書き／全角
001230    PRINTING MODE MEISAIJP18 IS FOR ALL
001240                IN   SIZE  007.2 POINT
001250                AT   PITCH 05.00 CPI
001260                WITH FONT  MINCHOU
001270                AT   ANGLE 00 DEGREES
001280                BY   FORM  F
001290* ７．２ポ／１０ＣＰＩ／明朝／横書き／全角
001300    PRINTING MODE MEISAIJP28 IS FOR ALL
001310                IN   SIZE  007.2 POINT
001320                AT   PITCH 10.00 CPI
001330                WITH FONT  MINCHOU
001340                AT   ANGLE 00 DEGREES
001350                BY   FORM  F
001360*-----------------------------------------------------------------------------*
001370* 文字サイズを示す機能名とそれに対応する呼び名を宣言。
001380*-----------------------------------------------------------------------------*
001390* ３．０ポ
001400    PRINTING MODE PMSIZE01 IS FOR ALL
001410                IN   SIZE   3   POINT
001420* ７．２ポ
001430    PRINTING MODE PMSIZE02 IS FOR ALL
001440                IN   SIZE   7.2 POINT
001450* ９．０ポ
```

```
001460    PRINTING MODE PMSIZE03 IS FOR ALL
001470                  IN   SIZE   9   POINT
001480* １２．０ポ
001490    PRINTING MODE PMSIZE04 IS FOR ALL
001500                  IN   SIZE  12   POINT
001510* １８．０ポ
001520    PRINTING MODE PMSIZE05 IS FOR ALL
001530                  IN   SIZE  18   POINT
001540* ２４．０ポ
001550    PRINTING MODE PMSIZE06 IS FOR ALL
001560                  IN   SIZE  24   POINT
001570* ３６．０ポ
001580    PRINTING MODE PMSIZE07 IS FOR ALL
001590                  IN   SIZE  36   POINT
001600* ５０．０ポ
001610    PRINTING MODE PMSIZE08 IS FOR ALL
001620                  IN   SIZE  50   POINT
001630* ７２．０ポ
001640    PRINTING MODE PMSIZE09 IS FOR ALL
001650                  IN   SIZE  72   POINT
001660* １００．０ポ
001670    PRINTING MODE PMSIZE10 IS FOR ALL
001680                  IN   SIZE 100   POINT
001690* ２００．０ポ
001700    PRINTING MODE PMSIZE11 IS FOR ALL
001710                  IN   SIZE 200   POINT
001720* ３００．０ポ
001730    PRINTING MODE PMSIZE12 IS FOR ALL
001740                  IN   SIZE 300   POINT
001750*-------------------------------------------------------------------------------*
001760* 文字ピッチを示す機能名とそれに対応する呼び名を宣言。
001770*-------------------------------------------------------------------------------*
001780* １．００ＣＰＩ
001790    PRINTING MODE PMPICH01 IS FOR ALL
001800                  AT   PITCH  1   CPI
001810* ２．００ＣＰＩ
001820    PRINTING MODE PMPICH02 IS FOR ALL
001830                  AT   PITCH  2   CPI
001840* ３．００ＣＰＩ
001850    PRINTING MODE PMPICH03 IS FOR ALL
001860                  AT   PITCH  3   CPI
001870* ５．００ＣＰＩ
001880    PRINTING MODE PMPICH04 IS FOR ALL
001890                  AT   PITCH  5   CPI
001900* ６．００ＣＰＩ
001910    PRINTING MODE PMPICH05 IS FOR ALL
001920                  AT   PITCH  6   CPI
001930* ７．５０ＣＰＩ
001940    PRINTING MODE PMPICH06 IS FOR ALL
001950                  AT   PITCH  7.5 CPI
001960* ８．００ＣＰＩ
001970    PRINTING MODE PMPICH07 IS FOR ALL
001980                  AT   PITCH  8   CPI
001990* １０．００ＣＰＩ
002000    PRINTING MODE PMPICH08 IS FOR ALL
002010                  AT   PITCH 10   CPI
002020* １２．００ＣＰＩ
002030    PRINTING MODE PMPICH09 IS FOR ALL
002040                  AT   PITCH 12   CPI
002050* １５．００ＣＰＩ
002060    PRINTING MODE PMPICH10 IS FOR ALL
002070                  AT   PITCH 15   CPI
002080* ２０．００ＣＰＩ
002090    PRINTING MODE PMPICH11 IS FOR ALL
002100                  AT   PITCH 20   CPI
002110* ２４．００ＣＰＩ
002120    PRINTING MODE PMPICH12 IS FOR ALL
002130                  AT   PITCH 24   CPI
002140*-------------------------------------------------------------------------------*
002150* 書体を示す機能名とそれに対応する呼び名を宣言。
002160*-------------------------------------------------------------------------------*
002170* ＧＯＴＨＩＣ
002180    PRINTING MODE PMFONT01 IS FOR ALL
002190                  WITH FONT   GOTHIC
002200* ＧＯＴＨＩＣ－ＨＡＮＫＡＫＵ
002210    PRINTING MODE PMFONT02 IS FOR ALL
002220                  WITH FONT   GOTHIC-HANKAKU
002230                  BY   FORM   H
```

```
002240* ＭＩＮＣＨＯＵ
002250    PRINTING MODE PMFONT03 IS FOR ALL
002260                  WITH FONT   MINCHOU
002270* ＭＩＮＣＨＯＵ－ＨＡＮＫＡＫＵ
002280    PRINTING MODE PMFONT04 IS FOR ALL
002290                  WITH FONT   MINCHOU-HANKAKU
002300                  BY   FORM   H
002310*-------------------------------------------------------------------------------*
002320* 文字回転を示す機能名とそれに対応する呼び名を宣言。
002330*-------------------------------------------------------------------------------*
002340* 横書き
002350    PRINTING MODE PMANGL01 IS FOR ALL
002360                  AT   ANGLE  0   DEGREES
002370* 縦書き（反時計回りに９０度回転）
002380    PRINTING MODE PMANGL02 IS FOR ALL
002390                  AT   ANGLE  90  DEGREES
002400*-------------------------------------------------------------------------------*
002410* 文字形態を示す機能名とそれに対応する呼び名を宣言。
002420*-------------------------------------------------------------------------------*
002430* 全角
002440    PRINTING MODE PMFORM01 IS FOR ALL
002450                  BY   FORM   F
002460* 半角
002470    PRINTING MODE PMFORM02 IS FOR ALL
002480                  BY   FORM   H
002490                  WITH FONT   GOTHIC-HANKAKU
002500* 全角平体
002510    PRINTING MODE PMFORM03 IS FOR ALL
002520                  BY   FORM   F0201
002530* 半角平体
002540    PRINTING MODE PMFORM04 IS FOR ALL
002550                  BY   FORM   H0201
002560                  WITH FONT   GOTHIC-HANKAKU
002570* 全角長体
002580    PRINTING MODE PMFORM05 IS FOR ALL
002590                  BY   FORM   F0102
002600* 半角長体
002610    PRINTING MODE PMFORM06 IS FOR ALL
002620                  BY   FORM   H0102
002630                  WITH FONT   GOTHIC-HANKAKU
002640* 全角倍角
002650    PRINTING MODE PMFORM07 IS FOR ALL
002660                  BY   FORM   F0202
002670* 半角倍角
002680    PRINTING MODE PMFORM08 IS FOR ALL
002690                  BY   FORM   H0202
002700                  WITH FONT   GOTHIC-HANKAKU
002710*-------------------------------------------------------------------------------*
002720* あらゆる文字属性の組合せを示す機能名とそれに対応する呼び名を宣言。
002730*-------------------------------------------------------------------------------*
002740* ９ポ／８ＣＰＩ／ＭＩＮＣＨＯＵ／横書き／全角
002750    PRINTING MODE PRTMODE1 IS FOR ALL
002760                  IN   SIZE   9   POINT
002770                  AT   PITCH  8   CPI
002780                  WITH FONT   MINCHOU
002790                  AT   ANGLE  0   DEGREES
002800                  BY   FORM   F
002810* ７．２ポ／２０ＣＰＩ／ＧＯＴＨＩＣ－ＨＡＮＫＡＫＵ／横書き／半角
002820    PRINTING MODE PRTMODE2 IS FOR ALL
002830                  IN   SIZE   7.2 POINT
002840                  AT   PITCH  20  CPI
002850                  WITH FONT   GOTHIC-HANKAKU
002860                  AT   ANGLE  0   DEGREES
002870                  BY   FORM   H
002880* １２ポ／２．５ＣＰＩ／ＧＯＴＨＩＣ／縦書き／倍角
002890    PRINTING MODE PRTMODE3 IS FOR ALL
002900                  IN   SIZE   12  POINT
002910                  AT   PITCH  2.5 CPI
002920                  WITH FONT   GOTHIC
002930                  AT   ANGLE  90  DEGREES
002940                  BY   FORM   F0202.
002950*
002960 INPUT-OUTPUT           SECTION.
002970*
002980 FILE-CONTROL.
002990    SELECT 印刷ファイル ASSIGN TO PRTFILE
003000          ORGANIZATION IS SEQUENTIAL
003010          ACCESS MODE  IS SEQUENTIAL
```

```
003020          FORMAT      IS 定義体名
003030          GROUP       IS パーティション名
003040          FILE STATUS  IS 入出力状態 詳細情報.
003050*-----------------------------------------------------------------------------*
003060 DATA                  DIVISION.
003070*
003080 FILE                  SECTION.
003090*
003100 FD  印刷ファイル CONTROL RECORD IS 制御レコード.
003110 01  行レコード          PIC N(136).
003120 01  注釈レコード        PIC N(050).
003130 01  制御レコード        PIC X(100).
003140*
003150 WORKING-STORAGE         SECTION.
003160* ＦＣＢを使用した帳票サンプルに使用するデータ宣言
003170 01  大見出し６          CHARACTER TYPE   IS OOMIDASHI6
003180                       PRINTING POSITION IS 36
003190                       PIC N(020)
003200                       VALUE NC"ＦＣＢを使用した６ＬＰＩでの帳票サンプル".
003210 01  小見出し６          CHARACTER TYPE   IS KOMIDASHI6.
003220   02  FILLER           PIC N(002)
003230                       VALUE NC"項番".
003240   02  FILLER           PRINTING POSITION IS 11
003250                       PIC N(005)
003260                       VALUE NC"商品コード".
003270   02  FILLER           PRINTING POSITION IS 27
003280                       PIC N(003)
003290                       VALUE NC"商品名".
003300   02  FILLER           PRINTING POSITION IS 57
003310                       PIC N(002)
003320                       VALUE NC"単価".
003330   02  FILLER           PRINTING POSITION IS 67
003340                       PIC N(004)
003350                       VALUE NC"出荷本数".
003360   02  FILLER           PRINTING POSITION IS 81
003370                       PIC N(004)
003380                       VALUE NC"在庫数量".
003390   02  FILLER           PRINTING POSITION IS 95
003400                       PIC N(002)
003410                       VALUE NC"備考".
003420 01  明細６.
003430   02  項番             CHARACTER TYPE IS MEISAIANK6
003440                       PIC 9(004)
003450                       VALUE 9999.
003460   02  商品コード        CHARACTER TYPE IS MEISAIANK6
003470                       PRINTING POSITION IS 11
003480                       PIC X(010)
003490                       VALUE "XXXXXXXXXX".
003500   02  商品名           CHARACTER TYPE IS MEISAIJP16
003510                       PRINTING POSITION IS 27
003520                       PIC N(010)
003530                       VALUE NC"ＮＮＮＮＮＮＮＮＮＮ".
003540   02  単価             CHARACTER TYPE IS MEISAIANK6
003550                       PRINTING POSITION IS 53
003560                       PIC X(008)
003570                       VALUE "¥9999999".
003580   02  出荷本数         CHARACTER TYPE IS MEISAIANK6
003590                       PRINTING POSITION IS 67
003600                       PIC 9(008)
003610                       VALUE 99999999.
003620   02  在庫数量         CHARACTER TYPE IS MEISAIANK6
003630                       PRINTING POSITION IS 81
003640                       PIC 9(008)
003650                       VALUE 99999999.
003660   02  備考             CHARACTER TYPE IS MEISAIJP26
003670                       PRINTING POSITION IS 95
003680                       PIC N(016)
003690                       VALUE NC"この行は６ＬＰＩで印字されます。".
003700 01  大見出し８          CHARACTER TYPE   IS OOMIDASHI8
003710                       PRINTING POSITION IS 36
003720                       PIC N(020)
003730                       VALUE NC"ＦＣＢを使用した８ＬＰＩでの帳票サンプル".
003740 01  小見出し８          CHARACTER TYPE   IS KOMIDASHI8.
003750   02  FILLER           PIC N(002)
003760                       VALUE NC"項番".
003770   02  FILLER           PRINTING POSITION IS 11
003780                       PIC N(005)
```

```
003790                        VALUE NC"商品コード".
003800   02  FILLER            PRINTING POSITION IS 27
003810                        PIC N(003)
003820                        VALUE NC"商品名".
003830   02  FILLER            PRINTING POSITION IS 57
003840                        PIC N(002)
003850                        VALUE NC"単価".
003860   02  FILLER            PRINTING POSITION IS 67
003870                        PIC N(004)
003880                        VALUE NC"出荷本数".
003890   02  FILLER            PRINTING POSITION IS 81
003900                        PIC N(004)
003910                        VALUE NC"在庫数量".
003920   02  FILLER            PRINTING POSITION IS 95
003930                        PIC N(002)
003940                        VALUE NC"備考".
003950 01  明細８.
003960   02  項番              CHARACTER TYPE IS MEISAIANK8
003970                        PIC 9(004)
003980                        VALUE 9999.
003990   02  商品コード        CHARACTER TYPE IS MEISAIANK8
004000                        PRINTING POSITION IS 11
004010                        PIC X(010)
004020                        VALUE "XXXXXXXXXX".
004030   02  商品名            CHARACTER TYPE IS MEISAIJP18
004040                        PRINTING POSITION IS 27
004050                        PIC N(010)
004060                        VALUE NC"ＮＮＮＮＮＮＮＮＮＮ".
004070   02  単価              CHARACTER TYPE IS MEISAIANK8
004080                        PRINTING POSITION IS 53
004090                        PIC X(008)
004100                        VALUE "¥9999999".
004110   02  出荷本数          CHARACTER TYPE IS MEISAIANK8
004120                        PRINTING POSITION IS 67
004130                        PIC 9(008)
004140                        VALUE 99999999.
004150   02  在庫数量          CHARACTER TYPE IS MEISAIANK8
004160                        PRINTING POSITION IS 81
004170                        PIC 9(008)
004180                        VALUE 99999999.
004190   02  備考              CHARACTER TYPE IS MEISAIJP28
004200                        PRINTING POSITION IS 95
004210                        PIC N(016)
004220                        VALUE NC"この行は８ＬＰＩで印字されます。".
004230*
004240* 注釈として使用するデータ宣言
004250 01  見出しデータ１       PIC N(050) CHARACTER TYPE IS MODE-1
004260                        VALUE NC"富士通ＮｅｔＣＯＢＯＬ　ＦＯＲＭＡＴ句付き印刷ファイル".
004270 01  見出しデータ２       PIC N(050) CHARACTER TYPE IS PMPICH01
004280                        VALUE NC"各種文字属性印字サンプル集".
004290 01  注釈データ１         PIC N(050) CHARACTER TYPE IS MODE-1
004300                        VALUE NC"◆あらゆる文字サイズを使用した印刷例。".
004310 01  注釈データ２         PIC N(050) CHARACTER TYPE IS MODE-1
004320                        VALUE NC"◆あらゆる文字ピッチを使用した印刷例。".
004330 01  注釈データ３         PIC N(050) CHARACTER TYPE IS MODE-1
004340                        VALUE NC"◆複数の書体を使用した印刷例。".
004350 01  注釈データ４         PIC N(100) CHARACTER TYPE IS MODE-1
004360                        VALUE NC"◆文字回転を使用した印刷例。".
004370 01  注釈データ５         PIC N(100) CHARACTER TYPE IS MODE-1
004380                        VALUE NC"◆あらゆる文字形態を使用した印刷例。".
004390 01  注釈データ６         PIC N(100) CHARACTER TYPE IS MODE-1
004400                        VALUE NC"◆様々な文字属性を組み合わせた（混在）印刷例。".
004410* 文字サイズを指定したデータ宣言
004420 01  文字サイズデータ.
004430   02                   PIC N(001) CHARACTER TYPE IS PMSIZE01
004440                        VALUE NC"あ".
004450   02                   PIC N(001) CHARACTER TYPE IS PMSIZE02
004460                        VALUE NC"あ".
004470   02                   PIC N(001) CHARACTER TYPE IS PMSIZE03
004480                        VALUE NC"あ".
004490   02                   PIC N(001) CHARACTER TYPE IS PMSIZE04
004500                        VALUE NC"あ".
004510   02                   PIC N(001) CHARACTER TYPE IS PMSIZE05
004520                        VALUE NC"あ".
004530   02                   PIC N(001) CHARACTER TYPE IS PMSIZE06
```

```
004540                       VALUE NC"あ".
004550   02                   PIC N(001) CHARACTER TYPE IS PMSIZE07
004560                       VALUE NC"あ".
004570   02                   PIC N(001) CHARACTER TYPE IS PMSIZE08
004580                       VALUE NC"あ".
004590   02                   PIC N(001) CHARACTER TYPE IS PMSIZE09
004600                       VALUE NC"あ".
004610   02                   PIC N(001) CHARACTER TYPE IS PMSIZE10
004620                       VALUE NC"あ".
004630   02                   PIC N(001) CHARACTER TYPE IS PMSIZE11
004640                       VALUE NC"あ".
004650   02                   PIC N(001) CHARACTER TYPE IS PMSIZE12
004660                       VALUE NC"あ".
004670* 文字ピッチを指定したデータ宣言
004680 01  文字ピッチデータ.
004690   02                   PIC N(001) CHARACTER TYPE IS PMPICH01
004700                       VALUE    NC"い".
004710   02                   PIC N(002) CHARACTER TYPE IS PMPICH02
004720                       VALUE ALL NC"い".
004730   02                   PIC N(003) CHARACTER TYPE IS PMPICH03
004740                       VALUE ALL NC"い".
004750   02                   PIC N(005) CHARACTER TYPE IS PMPICH04
004760                       VALUE ALL NC"い".
004770   02                   PIC N(006) CHARACTER TYPE IS PMPICH05
004780                       VALUE ALL NC"い".
004790   02                   PIC N(015) CHARACTER TYPE IS PMPICH06
004800                       VALUE ALL NC"い".
004810   02                   PIC N(008) CHARACTER TYPE IS PMPICH07
004820                       VALUE ALL NC"い".
004830   02                   PIC N(010) CHARACTER TYPE IS PMPICH08
004840                       VALUE ALL NC"い".
004850   02                   PIC N(012) CHARACTER TYPE IS PMPICH09
004860                       VALUE ALL NC"い".
004870   02                   PIC N(015) CHARACTER TYPE IS PMPICH10
004880                       VALUE ALL NC"い".
004890   02                   PIC N(020) CHARACTER TYPE IS PMPICH11
004900                       VALUE ALL NC"い".
004910   02                   PIC N(024) CHARACTER TYPE IS PMPICH12
004920                       VALUE ALL NC"い".
004930* 書体を指定したデータ宣言
004940 01  文字フォントデータ.
004950   02                   PIC N(010) CHARACTER TYPE IS PMFONT01
004960                       VALUE ALL NC"う".
004970   02                   PIC N(010) CHARACTER TYPE IS PMFONT02
004980                       VALUE ALL NC"う".
004990   02                   PIC N(010) CHARACTER TYPE IS PMFONT03
005000                       VALUE ALL NC"う".
005010   02                   PIC N(010) CHARACTER TYPE IS PMFONT04
005020                       VALUE ALL NC"う".
005030   02                   PIC N(010) CHARACTER TYPE IS PMFONT01
005040                       VALUE ALL NC"う".
005050   02                   PIC N(010) CHARACTER TYPE IS PMFONT02
005060                       VALUE ALL NC"う".
005070   02                   PIC N(010) CHARACTER TYPE IS PMFONT03
005080                       VALUE ALL NC"う".
005090   02                   PIC N(010) CHARACTER TYPE IS PMFONT04
005100                       VALUE ALL NC"う".
005110* 文字回転を指定したデータ宣言
005120 01  文字回転データ.
005130   02                   PIC N(010) CHARACTER TYPE IS PMANGL01
005140                       VALUE ALL NC"え".
005150   02                   PIC N(010) CHARACTER TYPE IS PMANGL02
005160                       VALUE ALL NC"え".
005170   02                   PIC N(010) CHARACTER TYPE IS PMANGL01
005180                       VALUE ALL NC"え".
005190   02                   PIC N(010) CHARACTER TYPE IS PMANGL02
005200                       VALUE ALL NC"え".
005210   02                   PIC N(010) CHARACTER TYPE IS PMANGL01
005220                       VALUE ALL NC"え".
005230   02                   PIC N(010) CHARACTER TYPE IS PMANGL02
005240                       VALUE ALL NC"え".
005250   02                   PIC N(010) CHARACTER TYPE IS PMANGL01
005260                       VALUE ALL NC"え".
005270   02                   PIC N(010) CHARACTER TYPE IS PMANGL02
005280                       VALUE ALL NC"え".
```

```
005290* 文字形態を指定したデータ宣言
005300 01  文字形態データ.
005310   02                  PIC N(009) CHARACTER TYPE IS PMFORM01
005320                       VALUE ALL NC"お".
005330   02                  PIC N(009) CHARACTER TYPE IS PMFORM02
005340                       VALUE ALL NC"お".
005350   02                  PIC N(009) CHARACTER TYPE IS PMFORM03
005360                       VALUE ALL NC"お".
005370   02                  PIC N(009) CHARACTER TYPE IS PMFORM04
005380                       VALUE ALL NC"お".
005390   02                  PIC N(009) CHARACTER TYPE IS PMFORM05
005400                       VALUE ALL NC"お".
005410   02                  PIC N(009) CHARACTER TYPE IS PMFORM06
005420                       VALUE ALL NC"お".
005430   02                  PIC N(009) CHARACTER TYPE IS PMFORM07
005440                       VALUE ALL NC"お".
005450   02                  PIC N(009) CHARACTER TYPE IS PMFORM08
005460                       VALUE ALL NC"お".
005470* 様々な文字属性を組合せ指定したデータ宣言
005480 01  文字属性組合せデータ.
005490   02                  PIC N(010) CHARACTER TYPE IS PRTMODE1
005500                       VALUE ALL NC"か".
005510   02                  PIC N(010) CHARACTER TYPE IS PRTMODE2
005520                       VALUE ALL NC"か".
005530   02                  PIC N(010) CHARACTER TYPE IS PRTMODE3
005540                       VALUE ALL NC"か".
005550   02                  PIC N(010) CHARACTER TYPE IS PRTMODE1
005560                       VALUE ALL NC"か".
005570   02                  PIC N(010) CHARACTER TYPE IS PRTMODE2
005580                       VALUE ALL NC"か".
005590   02                  PIC N(010) CHARACTER TYPE IS PRTMODE3
005600                       VALUE ALL NC"か".
005610*Ｉ制御レコードのデータ宣言
005620 01  Ｉ制御データ.
005630   02  レコード識別子    PIC X(002) VALUE "I1".
005640   02  モード            PIC X(001) VALUE "1".
005650   02  オーバレイ.
005660     03  オーバレイ名    PIC X(004) VALUE SPACE.
005670     03  焼付け回数      PIC 9(003) VALUE 0.
005680   02  複写数            PIC 9(003) VALUE 0.
005690   02  ＦＣＢ名          PIC X(004) VALUE SPACE.
005700   02  帳票定義体名      PIC X(008) VALUE SPACE.
005710   02                    PIC X(030) VALUE SPACE.
005720   02  印刷形式          PIC X(002) VALUE SPACE.
005730   02  用紙サイズ        PIC X(003) VALUE SPACE.
005740   02  用紙供給口        PIC X(002) VALUE SPACE.
005750   02                    PIC X(002) VALUE SPACE.
005760   02  印刷面            PIC X(001) VALUE SPACE.
005770   02  印刷開始位置づけ面 PIC X(001) VALUE SPACE.
005780   02  印字禁止域        PIC X(001) VALUE SPACE.
005790   02  綴じ代方向.
005800     03  ポート時表面    PIC X(001) VALUE SPACE.
005810     03  ポート時裏面    PIC X(001) VALUE SPACE.
005820     03  ランド時表面    PIC X(001) VALUE SPACE.
005830     03  ランド時裏面    PIC X(001) VALUE SPACE.
005840   02  綴じ代幅          PIC X(004) VALUE SPACE.
005850   02  印字開始原点位置.
005860     03  ポート時Ｘ座標  PIC X(004) VALUE SPACE.
005870     03  ポート時Ｙ座標  PIC X(004) VALUE SPACE.
005880     03  ランド時Ｘ座標  PIC X(004) VALUE SPACE.
005890     03  ランド時Ｙ座標  PIC X(004) VALUE SPACE.
005900   02  予約域            PIC X(009) VALUE SPACE.
005910* エラーカウンタとして使用するデータ宣言
005920 01  エラーカウンタ      PIC 9(004)       VALUE ZERO.
005930* 入出力状態として使用するデータ宣言
005940 77  入出力状態          PIC X(002)       VALUE ZERO.
005950 77  詳細情報            PIC X(004)       VALUE ZERO.
005960 77  定義体名            PIC X(008)       VALUE SPACE.
005970 77  パーティション名    PIC X(008)       VALUE SPACE.
005980*-----------------------------------------------------------------------------*
005990 PROCEDURE              DIVISION.
006000*
006010* 印刷ファイルのＯＰＥＮ
```

```
006020    OPEN OUTPUT 印刷ファイル.
006030    IF 入出力状態 NOT = ZERO THEN
006040      ADD 1 TO エラーカウンタ
006050    END-IF.
006060*
006070*-------------------------------------------------------------------------------*
006080* １．ＦＣＢを使用した帳票印刷サンプルの出力処理                                *
006090*-------------------------------------------------------------------------------*
006100*
006110* ６ＬＰＩの帳票印刷処理
006120*
006130* オーバレイ名”Ａ４Ｌ６”（ＫＯＬ５Ａ４Ｌ６．ＯＶＤ）設定
006140* ６ＬＰＩ／１０ＣＰＩのスペーシングチャートイメージのオーバレイを出力
006150    MOVE "A4L6" TO オーバレイ名.
006160* 焼き付け回数１回設定
006170    MOVE １     TO 焼付け回数.
006180* 複写枚数１枚設定
006190    MOVE １     TO 複写数.
006200* ＦＣＢ名”Ａ４Ｌ６”（ＦＣＢＡ４Ｌ６）設定
006210    MOVE "A4L6" TO ＦＣＢ名.
006220* 印刷形式ランドスケープモード(横向き)設定
006230    MOVE "L"    TO 印刷形式.
006240* 用紙サイズＡ４設定
006250    MOVE "A4"   TO 用紙サイズ.
006260* 用紙供給口任意設定
006270    MOVE "P "   TO 用紙供給口.
006280* 片面印刷設定
006290    MOVE "F"    TO 印刷面.
006300* 表面位置づけ設定
006310    MOVE "F"    TO 印刷開始位置づけ面.
006320* Ｉ制御レコードを出力することによりページ属性を設定
006330    WRITE 制御レコード FROM Ｉ制御データ.
006340    IF 入出力状態 NOT = ZERO THEN
006350      ADD 1 TO エラーカウンタ
006360    END-IF.
006370*
006380* 大見出し出力
006390    MOVE SPACE  TO 注釈レコード.
006400    WRITE 注釈レコード AFTER ADVANCING PAGE.
006410    IF 入出力状態 NOT = ZERO THEN
006420      ADD 1 TO エラーカウンタ
006430    END-IF.
006440*
006450* 小見出し出力
006460    WRITE 行レコード FROM 大見出し６
006470                     AFTER ADVANCING 1 LINE.
006480    IF 入出力状態 NOT = ZERO THEN
006490      ADD 1 TO エラーカウンタ
006500    END-IF.
006510*
006520* 明細出力
006530    WRITE 行レコード FROM 小見出し６
006540                     AFTER ADVANCING 2 LINES.
006550    IF 入出力状態 NOT = ZERO THEN
006560      ADD 1 TO エラーカウンタ
006570    END-IF.
006580    WRITE 行レコード FROM 明細６
006590                     AFTER ADVANCING 2 LINES
006600    PERFORM 40 TIMES
006610      WRITE 行レコード FROM 明細６
006620                     AFTER ADVANCING 1 LINE
006630      IF 入出力状態 NOT = ZERO THEN
006640        ADD 1 TO エラーカウンタ
006650      END-IF
006660    END-PERFORM.
006670*
006680* ８ＬＰＩの帳票印刷処理
006690*
006700* オーバレイ名”Ａ４Ｌ８”（ＫＯＬ５Ａ４Ｌ８．ＯＶＤ）設定
006710* ８ＬＰＩ／１０ＣＰＩのスペーシングチャートイメージのオーバレイを出力
006720    MOVE "A4L8" TO オーバレイ名.
006730* 焼き付け回数１回設定
006740    MOVE １     TO 焼付け回数.
006750* 複写枚数１枚設定
```

```
006760    MOVE １     TO 複写数.
006770* ＦＣＢ名”Ａ４Ｌ６”（ＦＣＢＡ４Ｌ６）設定
006780    MOVE "A4L8" TO ＦＣＢ名.
006790* 印刷形式ランドスケープモード(横向き)設定
006800    MOVE "L"    TO 印刷形式.
006810* 用紙サイズＡ４設定
006820    MOVE "A4"   TO 用紙サイズ.
006830* 用紙供給口任意設定
006840    MOVE "P "   TO 用紙供給口.
006850* 片面印刷設定
006860    MOVE "F"    TO 印刷面.
006870* 表面位置づけ設定
006880    MOVE "F"    TO 印刷開始位置づけ面.
006890* Ｉ制御レコードを出力することによりページ属性を設定
006900    WRITE 制御レコード FROM Ｉ制御データ.
006910    IF 入出力状態 NOT = ZERO THEN
006920      ADD 1 TO エラーカウンタ
006930    END-IF.
006940*
006950* 大見出し出力
006960    MOVE SPACE  TO 注釈レコード.
006970    WRITE 注釈レコード AFTER ADVANCING PAGE.
006980    IF 入出力状態 NOT = ZERO THEN
006990      ADD 1 TO エラーカウンタ
007000    END-IF.
007010    WRITE 行レコード FROM 大見出し８
007020                     AFTER ADVANCING 1 LINE.
007030    IF 入出力状態 NOT = ZERO THEN
007040      ADD 1 TO エラーカウンタ
007050    END-IF.
007060*
007070* 小見出し出力
007080    WRITE 行レコード FROM 小見出し８
007090                     AFTER ADVANCING 2 LINES.
007100    IF 入出力状態 NOT = ZERO THEN
007110      ADD 1 TO エラーカウンタ
007120    END-IF.
007130*
007140* 明細出力
007150    WRITE 行レコード FROM 明細８
007160                     AFTER ADVANCING 2 LINES
007170    PERFORM 55 TIMES
007180      WRITE 行レコード FROM 明細８
007190                     AFTER ADVANCING 1 LINE
007200      IF 入出力状態 NOT = ZERO THEN
007210        ADD 1 TO エラーカウンタ
007220      END-IF
007230    END-PERFORM.
007240*
007250*-------------------------------------------------------------------------------*
007260* ２．各種文字属性印字サンプル集の出力処理                                      *
007270*-------------------------------------------------------------------------------*
007280*
007290* オーバレイ名”Ｂ４ＯＶ”（ＫＯＬ５Ｂ４ＯＶ．ＯＶＤ）設定
007300    MOVE "B4OV" TO オーバレイ名.
007310* 焼き付け回数１回設定
007320    MOVE １     TO 焼付け回数.
007330* 複写枚数１枚設定
007340    MOVE １     TO 複写数.
007350* ＦＣＢ名”ＬＰＩ６”（ＦＣＢ６ＬＰＩ）設定
007360    MOVE "LPI6" TO ＦＣＢ名.
007370* 印刷形式ランドスケープモード(横向き)設定
007380    MOVE "L"    TO 印刷形式.
007390* 用紙サイズＢ４設定
007400    MOVE "B4"   TO 用紙サイズ.
007410* 用紙供給口任意設定
007420    MOVE "P "   TO 用紙供給口.
007430* 片面印刷設定
007440    MOVE "F"    TO 印刷面.
007450* 表面位置づけ設定
007460    MOVE "F"    TO 印刷開始位置づけ面.
007470* Ｉ制御レコードを出力することによりページ属性を設定
007480    WRITE 制御レコード FROM Ｉ制御データ.
```

```
007490    IF 入出力状態 NOT = ZERO THEN
007500      ADD 1 TO エラーカウンタ
007510    END-IF.
007520*
007530* 見出し出力
007540    WRITE 注釈レコード FROM 見出しデータ１
007550                    AFTER ADVANCING PAGE.
007560    IF 入出力状態 NOT = ZERO THEN
007570      ADD 1 TO エラーカウンタ
007580    END-IF.
007590    WRITE 注釈レコード FROM 見出しデータ２
007600                    AFTER ADVANCING 30 LINES.
007610    IF 入出力状態 NOT = ZERO THEN
007620      ADD 1 TO エラーカウンタ
007630    END-IF.
007640*
007650* 文字サイズサンプル出力
007660    WRITE 注釈レコード FROM 注釈データ１
007670                    AFTER ADVANCING PAGE.
007680    WRITE 行レコード   FROM 文字サイズデータ
007690                    AFTER ADVANCING 40 LINES.
007700    IF 入出力状態 NOT = ZERO THEN
007710      ADD 1 TO エラーカウンタ
007720    END-IF.
007730*
007740* 文字ピッチサンプル出力
007750    WRITE 注釈レコード FROM 注釈データ２
007760                    AFTER ADVANCING PAGE.
007770    PERFORM 5 TIMES
007780      WRITE 行レコード FROM 文字ピッチデータ
007790                    AFTER ADVANCING 10 LINES
007800      IF 入出力状態 NOT = ZERO THEN
007810        ADD 1 TO エラーカウンタ
007820      END-IF
007830    END-PERFORM.
007840*
007850* 文字書体サンプル出力
007860    WRITE 注釈レコード FROM 注釈データ３
007870                    AFTER ADVANCING PAGE.
007880    PERFORM 18 TIMES
007890      WRITE 行レコード FROM 文字フォントデータ
007900                    AFTER ADVANCING 3 LINES
007910      IF 入出力状態 NOT = ZERO THEN
007920        ADD 1 TO エラーカウンタ
007930      END-IF
007940    END-PERFORM.
007950*
007960* 文字回転サンプル出力
007970    WRITE 注釈レコード FROM 注釈データ４
007980                    AFTER ADVANCING PAGE.
007990    PERFORM 18 TIMES
008000      WRITE 行レコード FROM 文字回転データ
008010                    AFTER ADVANCING 3 LINES
008020      IF 入出力状態 NOT = ZERO THEN
008030        ADD 1 TO エラーカウンタ
008040      END-IF
008050    END-PERFORM.
008060*
008070* 文字形態データ出力
008080    WRITE 注釈レコード FROM 注釈データ５
008090                    AFTER ADVANCING PAGE.
008100    PERFORM 18 TIMES
008110      WRITE 行レコード FROM 文字形態データ
008120                    AFTER ADVANCING 3 LINES
008130      IF 入出力状態 NOT = ZERO THEN
008140        ADD 1 TO エラーカウンタ
008150      END-IF
008160    END-PERFORM.
008170*
008180* 文字属性組合せサンプル出力
008190    WRITE 注釈レコード FROM 注釈データ６
008200                    AFTER ADVANCING PAGE.
008210    PERFORM 18 TIMES
008220      WRITE 行レコード FROM 文字属性組合せデータ
008230                    AFTER ADVANCING 3 LINES
008240      IF 入出力状態 NOT = ZERO THEN
```

```
008250       ADD 1 TO エラーカウンタ
008260     END-IF
008270   END-PERFORM.
008280*
008290* 印刷ファイルＣＬＯＳＥ
008300   CLOSE 印刷ファイル.
008310   IF 入出力状態 NOT = ZERO THEN
008320     ADD 1 TO エラーカウンタ
008330   END-IF.
008340* エラー判定
008350   IF エラーカウンタ > ZERO THEN
008360     DISPLAY NC"エラーが発生しました。"
008370     DISPLAY NC"実行時メーッセージを確認し、エラーの原因を取り除いてください。"
008380     DISPLAY SPACE
008390   END-IF.
008400*
008410   STOP RUN.
008420
```

ソースコード：Print3.cob

```
000010*================================================================
000020*「ＦＯＲＭＡＴ句付き印刷ファイルを使ったプログラム」
000030*    ＦＯＲＭＡＴ句付き印刷ファイルで帳票定義体を使って、集計表を
000040*  プリンタ出力します。
000050*
000060*================================================================
000070****  注意事項  ****
000080*================================================================
000090*    このプログラムを実行するためには、ＭｅＦｔが必要です。
000100*
000110*    Copyright 2003-2010 FUJITSU LIMITED
000120*================================================================
000130 IDENTIFICATION DIVISION.
000140  PROGRAM-ID. PRINT3.
000150*
000160 ENVIRONMENT DIVISION.
000170  CONFIGURATION SECTION.
000180   SPECIAL-NAMES.
000190     SYSERR IS メッセージ出力先.
000200  INPUT-OUTPUT SECTION.
000210   FILE-CONTROL.
000220     SELECT 帳票の印刷先 ASSIGN TO PRTFILE
000230       FORMAT          IS 定義体の名前
000240       GROUP           IS 項目群の名前
000250       FILE STATUS      IS 帳票の状態
000260                         詳細コード.
000270     SELECT 商品マスタ   ASSIGN TO DATAFILE
000280       FILE STATUS      IS 商品マスタの状態
000290       ORGANIZATION     IS INDEXED
000300       ACCESS MODE      IS SEQUENTIAL
000310       RECORD KEY       IS 商品コード OF 商品マスタレコード
000320       ALTERNATE RECORD KEY IS  商品名  OF  商品マスタレコード.
000330     SELECT 売上ファイル ASSIGN TO SALEFILE
000340       FILE STATUS      IS 売上ファイルの状態
000350       ORGANIZATION     IS INDEXED
000360       ACCESS MODE      IS DYNAMIC
000370       RECORD KEY       IS 商品コード OF 売上レコード
000380                        WITH DUPLICATES
000390       ALTERNATE RECORD KEY IS  売上日 OF 売上レコード
000400                        WITH DUPLICATES.
000410*
000420 DATA DIVISION.
000430  FILE SECTION.
000440   FD  帳票の印刷先.
000450     COPY  SYUUKEI OF XMDLIB.
000460   FD  商品マスタ.
000470   01  商品マスタレコード.
000480     COPY  SYOHINM.
000490   FD  売上ファイル.
000500     COPY  URIAGE.
000510*
000520  WORKING-STORAGE SECTION.
000530   77  W数量                PIC 9(08)  VALUE 0.
000540   77  W小計                PIC 9(13)  VALUE 0.
000550   77  W合計                PIC 9(13)  VALUE 0.
000560   77  ページカウンタ        PIC 9(03)  VALUE 0.
000570   77  行カウンタ            PIC 9(03)  VALUE 0.
000580   77  定義体の名前          PIC X(08)  VALUE "SYUUKEI".
000590   77  項目群の名前          PIC X(08).
000600   77  帳票の状態            PIC XX.
000610       88  帳票の印刷成功              VALUE "00".
000620   77  詳細コード            PIC X(04).
000630   77  商品マスタの状態      PIC XX.
000640       88  商品マスタのアクセス成功      VALUE "00".
000650   77  売上ファイルの状態    PIC XX.
000660       88  売上ファイルのアクセス成功    VALUE "00".
```

```
000670   77  処理の結果              PIC XX.
000680       88  エラー発生                    VALUE "77".
000690       88  正常終了                      VALUE "00".
000700   77  行制御の状態            PIC XX.
000710       88  商品データの切り換え          VALUE "99".
000720       88  売上げデータ継続              VALUE "00".
000730*  メッセージ（ファイルのアクセスに失敗しました）
000740    77  ファイルアクセス失敗    PIC N(20)  VALUE
000750            NC"ファイルのアクセスに失敗しました．".
000760*  メッセージ（帳票の印刷に失敗しました）
000770    77  帳票印刷失敗            PIC N(20)  VALUE
000780            NC"帳票の印刷に失敗しました．".
000790*
000800  CONSTANT SECTION.
000810   77  ページヘッダ            PIC 9     VALUE 6.
000820   77  制御ヘッダ              PIC 9     VALUE 5.
000830   77  詳細部                  PIC 9     VALUE 2.
000840   77  制御フッタ              PIC 9     VALUE 5.
000850   77  集計フッタ              PIC 9     VALUE 4.
000860   77  ページフッタの開始行    PIC 9(02)  VALUE 63.
000870*
000880 PROCEDURE DIVISION.
000890*
000900     SET 正常終了 TO TRUE.
000910*
000920     PERFORM 入力ファイルをオープンする.
000930      IF エラー発生 THEN GO TO 処理を終了する.
000940*
000950     PERFORM 帳票の印刷先をオープンする.
000960      IF エラー発生 THEN
000970        PERFORM  入力ファイルをクローズする
000980        GO TO 処理を終了する
000990      END-IF.
001000*
001010     PERFORM 帳票を印刷する.
001020*
001030     PERFORM 帳票の印刷先をクローズする.
001040     PERFORM 入力ファイルをクローズする.
001050*
001060  処理を終了する.
001070     EXIT PROGRAM.
001080*
001090*================================================================
001100*
001110  入力ファイルをオープンする  SECTION.
001120      OPEN INPUT 商品マスタ 売上ファイル.
001130      IF NOT 商品マスタのアクセス成功
001140        THEN DISPLAY ファイルアクセス失敗 UPON メッセージ出力先
001150             SET エラー発生 TO TRUE.
001160      IF NOT 売上ファイルのアクセス成功
001170        THEN PERFORM マスタファイルをクローズする
001180             DISPLAY ファイルアクセス失敗 UPON メッセージ出力先
001190             SET エラー発生 TO TRUE.
001200*
001210*============================================================
001220*
001230  マスタファイルをクローズする  SECTION.
001240      CLOSE  商品マスタ.
001250      IF NOT 商品マスタのアクセス成功
001260        THEN DISPLAY ファイルアクセス失敗 UPON メッセージ出力先
001270             SET エラー発生 TO TRUE.
001280*
001290*================================================================
001300*
001310  入力ファイルをクローズする  SECTION.
001320      CLOSE  商品マスタ 売上ファイル.
001330      IF NOT 商品マスタのアクセス成功 OR
001340         NOT 売上ファイルのアクセス成功
001350        THEN DISPLAY ファイルアクセス失敗 UPON メッセージ出力先
001360             SET エラー発生 TO TRUE.
001370*
001380*
001390*============================================================
001400*
```

```
001410  帳票の印刷先をオープンする  SECTION.
001420      OPEN OUTPUT 帳票の印刷先.
001430      IF NOT 帳票の印刷成功
001440        THEN DISPLAY 帳票印刷失敗 UPON メッセージ出力先
001450            SET エラー発生 TO TRUE.
001460*
001470*=============================================================
001480*
001490  帳票の印刷先をクローズする  SECTION.
001500      CLOSE 帳票の印刷先.
001510      IF NOT 帳票の印刷成功
001520        THEN DISPLAY 帳票印刷失敗 UPON メッセージ出力先
001530            SET エラー発生 TO TRUE.
001540*
001550*=============================================================
001560*
001570  帳票を印刷する  SECTION.
001580     MOVE ZERO      TO W合計.
001590*
001600    ページの先頭.
001610     PERFORM ページヘッダを印刷する.
001620*
001630   商品データを読み込む.
001640     READ 商品マスタ NEXT
001650       AT END
001660         PERFORM 集計を印刷する
001670         PERFORM ページフッタを印刷する
001680         GO TO 帳票印刷終了
001690     END-READ.
001700     START 売上ファイル FIRST RECORD
001710       INVALID KEY GO TO 商品データを読み込む
001720     END-START.
001730     MOVE 商品コード OF 商品マスタレコード
001740       TO 商品コード OF 売上レコード.
001750     START 売上ファイル
001760       KEY IS = 商品コード OF 売上レコード
001770       INVALID KEY GO TO 商品データを読み込む
001780     END-START.
001790*
001800   制御ヘッダを印刷する.
001810     IF 行カウンタ >= ページフッタの開始行 -
001820        (制御ヘッダ + 詳細部 + 制御フッタ) THEN
001830       PERFORM ページフッタを印刷する
001840       PERFORM ページヘッダを印刷する
001850     END-IF.
001860     MOVE  "CH1"     TO 項目群の名前.
001870     MOVE 商品コード OF 商品マスタレコード
001880       TO 商品コード OF SYUUKEI.
001890     MOVE 商品名     OF 商品マスタレコード
001900       TO 商品名     OF SYUUKEI.
001910     MOVE 単価       OF 商品マスタレコード
001920       TO 単価       OF SYUUKEI.
001930     WRITE SYUUKEI.
001940     IF NOT 帳票の印刷成功
001950       THEN GO TO 帳票印刷失敗処理
001960     END-IF.
001970     ADD  制御ヘッダ TO 行カウンタ.
001980     MOVE ZERO      TO W数量 W小計.
001990     READ 売上ファイル.
002000     SET 売上げデータ継続 TO TRUE.
002010     PERFORM WITH TEST BEFORE UNTIL
002020       行カウンタ >= ページフッタの開始行 - (制御フッタ + 詳細部)
002030*  詳細部を印刷する
002040       MOVE  "DE"      TO 項目群の名前
002050       MOVE 売上日     OF 売上レコード
002060         TO 受注日     OF SYUUKEI
002070       MOVE 数量       OF 売上レコード
002080         TO 数量       OF SYUUKEI
002090       MOVE 金額       OF 売上レコード
002100         TO 売上げ     OF SYUUKEI
002110       WRITE SYUUKEI
002120       IF NOT 帳票の印刷成功
002130         THEN GO TO 帳票印刷失敗処理
```

```
002140      END-IF
002150      ADD 詳細部             TO 行カウンタ
002160      ADD 数量 OF 売上レコード TO W数量
002170      ADD 金額 OF 売上レコード TO W小計
002180      READ 売上ファイル NEXT
002190        AT END
002200          SET 商品データの切り換え TO TRUE
002210          EXIT PERFORM
002220      END-READ
002230      IF 商品コード OF 売上レコード NOT =
002240         商品コード OF 商品マスタレコード
002250        SET 商品データの切り換え TO TRUE
002260        EXIT PERFORM
002270      END-IF
002280    END-PERFORM.
002290*
002300   制御フッタを印刷する.
002310     MOVE  "CF1"     TO 項目群の名前
002320     MOVE W数量     TO 数量小計   OF SYUUKEI
002330     MOVE W小計     TO 売上げ小計 OF SYUUKEI
002340     WRITE SYUUKEI
002350     IF NOT 帳票の印刷成功
002360       THEN GO TO 帳票印刷失敗処理
002370     END-IF
002380     ADD W小計      TO W合計
002390     ADD 制御フッタ  TO 行カウンタ
002400     IF 行カウンタ >= ページフッタの開始行 - 詳細部 THEN
002410       PERFORM ページフッタを印刷する
002420       PERFORM ページヘッダを印刷する
002430       IF 売上げデータ継続 THEN
002440         GO TO 制御ヘッダを印刷する
002450       END-IF
002460     END-IF
002470*
002480     GO TO 商品データを読み込む.
002490*
002500 ページヘッダを印刷する.
002510     INITIALIZE SYUUKEI.
002520     MOVE  "PH"     TO 項目群の名前.
002530     WRITE SYUUKEI AFTER ADVANCING PAGE.
002540     IF NOT 帳票の印刷成功
002550       THEN GO TO 帳票印刷失敗処理.
002560     MOVE ページヘッダ TO 行カウンタ.
002570     ADD  1           TO ページカウンタ.
002580*
002590   ページヘッダを印刷する終了.
002600     EXIT.
002610*
002620   ページフッタを印刷する.
002630     MOVE  "PF"     TO 項目群の名前.
002640     MOVE ページカウンタ TO ページ数 OF SYUUKEI.
002650     WRITE SYUUKEI.
002660     IF NOT 帳票の印刷成功
002670       THEN GO TO 帳票印刷失敗処理
002680     END-IF.
002690*
002700   ページフッタを印刷する終了.
002710     EXIT.
002720*
002730   集計を印刷する.
002740     MOVE  "CF2"    TO 項目群の名前.
002750     MOVE W合計    TO 合計 OF SYUUKEI.
002760     WRITE SYUUKEI.
002770     IF NOT 帳票の印刷成功
002780       THEN GO TO 帳票印刷失敗処理
002790     END-IF.
002800*
002810   集計を印刷する終了.
002820     EXIT.
002830*
002840   帳票印刷失敗処理.
002850     DISPLAY 帳票印刷失敗 UPON メッセージ出力先.
002860     SET エラー発生 TO TRUE.
002870*
002880   帳票印刷終了.
```

```
002890     EXIT.
002900
```

NCOBU00728

NetCOBOL for .NET Visual Studio プロジェクトサンプルアプリケーション

Visual Studio プロジェクトサンプルアプリケーション は、COBOL の Visual Studio への統合をデモンストレーションします。 サンプルは VisualStudio サブフォルダにあります。 サンプルを試すためには、 Visual Studio からソリューションファイル(.sln)を開いてください。

1. Hello World

フォルダ	Hello
ソリューションファイル	Hello.sln
ソースコード	Hello.cob

説明

コンソール上の"Hello World"プログラムです

2. Multi-language Dogs

フォルダ	MultiDog
ソリューションファイル	dogs.sln
ソースコード	bigdog.cob

説明

このサンプルでは、複数言語を使用した例を示します。 COBOL の BigDog クラスでは Visual Basic の Dog クラスを継承し、 Bark メソッドを上書きしています。 また、Visual C#の Main メソッドから、BigDog クラスの RollOver メソッドを呼び出すことによって、継承された Bark メソッドを呼び出します。 このサンプルを使用するためには、Microsoft Visual C#および Microsoft Visual Basic のパッケージが必要です。

3. Alphabet Table Lookup

フォルダ	AlphabetTable
ソリューションファイル	AlphabetTable.sln
ソースコード	AlphabetTable.cob

説明

COBOL の小入出力機能を使って、 コンソールウィンドウからデータを入力したり、 コンソールウィンドウにデータを出力するプログラムの例を示します。 このサンプルは、コンソールウィンドウからアルファベット 1 文字(小文字)を入力し、 入力したアルファベットで始まる単語をコンソールウィンドウに出力します。

4. Calculate Days Between Two Dates

フォルダ	DaysBetween
ソリューションファイル	DaysBetween.sln
ソースコード	DaysBetween.cob

説明

コマンドライン引数の操作機能を使って、 COBOL プログラムの実行時に指定された引数を受け取るプログラムの例を示します。 このサンプルは、開始年月日から終了年月日までの日数を求めます。 開始年月日および終了年月日は、コマンドの引数として次の形式で指定します。

```
コマンド名　開始年月日　終了年月日
```

開始年月日および終了年月日は、 1900 年 1 月 1 日～2172 年 12 月 31 日までの日付を YYYYMMDD で指定します。 西暦については 4 桁で指定します。 Visual Studio でデバッグするには、 DaysBetween プロジェクトのプロパティで、[デバッグ]タブにある[コマンドライン引数]プロパティに値を設定します。デバッグオプションについては、デバッグオプションの設定を参照して下さい。

5. Schedule

フォルダ	Schedule
ソリューションファイル	Schedule.sln
ソースコード	MainForm.cob, Registration.cob, Calendar.cob, Delegate.cob, Main.cob

説明

フォーム間のデータの受け渡しを行うためにメソッドやプロパティ、および デリゲートを使用した Windows アプリケーションの例を示します。 このサンプルは、スケジュール一覧のリストを表示し、 別のウィンドウから入力データを登録します。

6. CSV to Indexed File

フォルダ	CsvToIdx
ソリューションファイル	CsvToIdx.sln
ソースコード	MainForm.cob, Extraction.cob, IndicatorForm.cob, Main.cob

説明

カンマ区切り値(CSV)ファイル形式のデータを読込み、 索引ファイルを作成する

Windows フォームアプリケーションの例を示します。 それぞれのファイルを、Windows のコモンダイアログを使用して設定し、 [変換]ボタンをクリックすると、 バックグラウンドで CSV ファイルから索引ファイルへの変換が行われます。 CSV ファイルは行順編成ファイルとして参照し、CSV-FORMAT 指定の UNSTRING 文を利用してデータを読み込みます。 読み込まれたデータは ACP-OF 関数でシフト JIS 形式に変換して、 索引ファイルに書き込みます。 このサンプルで作成した索引ファイルは、 Web アプリケーションサンプルの郵便番号検索サービスで使用することができます。 CSV ファイルは、 日本郵便株式会社(http://www.post.japanpost.jp/)が提供している郵便番号データを使用しています。 最新の郵便番号データは http://www.post.japanpost.jp/zipcode/download.html からダウンロードできます。

7. Fibonacci

フォルダ	Fibonacci
ソリューションファイル	Fibonacci.sln
ソースコード	Fib_COBOL.cob

説明

数種のプログラミング言語に組み入れたクラスを使用して、 フィボナッチ数列を計算します。 呼び出すフィボナッチ計算関数の数を意味する引数が、 Driver プログラム ("Driver.exe") に渡されることに注意してください。 Visual Studio でデバッグするには、Driver プロジェクトのプロパティで、[デバッグ]タブにある[コマンドライン引数]プロパティに値を設定します。 フィボナッチに興味のある方は、 http://pass.maths.org.uk/issue3/fibonacci/index.html を参照してください。 このサンプルを使用するためには、 Microsoft Visual C#および Microsoft Visual Basic のパッケージが必要です。

ソースコード：Hello.cob

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. MAIN.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050 WORKING-STORAGE SECTION.
000060 PROCEDURE DIVISION.
000070    DISPLAY "Hello World.".
000080    DISPLAY "Done!".
000090 END PROGRAM MAIN.
```

NCOBU00730

ソースコード：bigdog.cob

```
CLASS-ID. BIGDOG AS "BigDog" INHERITS DOG.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS DOG AS "Dog".
OBJECT.
PROCEDURE DIVISION.
METHOD-ID. BARK AS "Bark" OVERRIDE.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I BINARY-LONG.
LINKAGE SECTION.
77 Bark-Count BINARY-LONG.
PROCEDURE DIVISION USING BY VALUE Bark-Count.
   PERFORM VARYING I FROM 1 BY 1 UNTIL I > Bark-Count
      DISPLAY "WOOF WOOF (COBOL)"
   END-PERFORM.
END METHOD BARK.
END OBJECT.
END CLASS BIGDOG.
```

ソースコード：AlphabetTable.cob

```
000100 IDENTIFICATION DIVISION.
000200  PROGRAM-ID. ALPHABETTABLE.
000300*
000400 DATA DIVISION.
000500  WORKING-STORAGE SECTION.
000600  01  単語一覧.
000700      02    PIC X(10) VALUE "apple".
000800      02    PIC X(10) VALUE "black".
000900      02    PIC X(10) VALUE "cobol".
001000      02    PIC X(10) VALUE "dog".
001100      02    PIC X(10) VALUE "eye".
001200      02    PIC X(10) VALUE "fault".
001300      02    PIC X(10) VALUE "good".
001400      02    PIC X(10) VALUE "high".
001500      02    PIC X(10) VALUE "idea".
001600      02    PIC X(10) VALUE "junior".
001700      02    PIC X(10) VALUE "king".
001800      02    PIC X(10) VALUE "love".
001900      02    PIC X(10) VALUE "medium".
002000      02    PIC X(10) VALUE "new".
002100      02    PIC X(10) VALUE "open".
002200      02    PIC X(10) VALUE "pig".
002300      02    PIC X(10) VALUE "queen".
002400      02    PIC X(10) VALUE "review".
002500      02    PIC X(10) VALUE "smile".
002600      02    PIC X(10) VALUE "tomorrow".
002700      02    PIC X(10) VALUE "understand".
002800      02    PIC X(10) VALUE "version".
002900      02    PIC X(10) VALUE "wood".
003000      02    PIC X(10) VALUE "xylophone".
003100      02    PIC X(10) VALUE "yesterday".
003200      02    PIC X(10) VALUE "zoo".
003300      02    PIC X(10) VALUE "**error**".
003400 01  単語表  REDEFINES  単語一覧.
003500    02  単語  OCCURS 27 TIMES.
003600       03  先頭文字  PIC X.
003700       03          PIC X(9).
003800 01  繰り返し回数     PIC 9(3).
003900 01  入力文字         PIC X.
004000 01  入力要求メッセージ  PIC N(30)
004100    VALUE  NC"アルファベットを１文字（小文字）入力してください。＝>".
004200*
004300 PROCEDURE DIVISION.
004400  データ入力  SECTION.
004500**（１）要求メッセージを出力します。出力後、改行は行いません。
004600      DISPLAY 入力要求メッセージ WITH NO ADVANCING.
004700**（２）アルファベット１文字を入力します。
004800      ACCEPT  入力文字.
004900*
005000  単語の検索  SECTION.
005100**（３）アルファベットに対応する単語を検索します。
005200      PERFORM TEST BEFORE
005300            VARYING 繰り返し回数 FROM 1 BY 1
005400            UNTIL 繰り返し回数 > 26
005500        IF 入力文字 = 先頭文字 (繰り返し回数)
005600          THEN EXIT PERFORM
005700        END-IF
005800      END-PERFORM.
005900*
006000  単語の表示  SECTION.
006100**（４）アルファベットに対応する単語を表示します。
006200      DISPLAY 単語 (繰り返し回数).
006300*
006400      EXIT PROGRAM.
006500   END PROGRAM ALPHABETTABLE.
```

NCOBU00732

ソースコード：DaysBetween.cob

```
000010*================================================================
000020*「コマンド行引数の受取り方」
000030*
000040*
000050* Copyright 2000-2010 FUJITSU LIMITED
000060*================================================================
000070 IDENTIFICATION DIVISION.
000080  PROGRAM-ID. DAYSBETWEEN.
000090*
000100 ENVIRONMENT DIVISION.
000110  CONFIGURATION SECTION.
000120  SPECIAL-NAMES.
000130     ARGUMENT-NUMBER IS 引数の番号
000140     ARGUMENT-VALUE  IS 引数の値.
000150*
000160 DATA DIVISION.
000170 WORKING-STORAGE SECTION.
000180   01  引数の数    PIC 9(4)  BINARY.
000190   01  日付の差    PIC S9(5) DISPLAY.
000200*
000210   01  通算日１    PIC S9(8) BINARY.
000220   01  引数１.
000230     02  年月日１.
000240        03  年   PIC 9999.
000250        03  月   PIC 99.
000260        03  日   PIC 99.
000270     02  日付情報１  REDEFINES 年月日１  PIC X(8).
000280*
000290   01  通算日２    PIC S9(8) BINARY.
000300   01  引数２.
000310     02  年月日２.
000320        03  年  PIC 9999.
000330        03  月  PIC 99.
000340        03  日  PIC 99.
000350     02  日付情報２  REDEFINES 年月日２  PIC X(8).
000360*
000370 PROCEDURE DIVISION.
000380     ACCEPT 引数の数 FROM 引数の番号.
000390     IF 引数の数 NOT = 2 THEN
000400       DISPLAY "引数の数に誤りがあります. "
000410       GO TO 終了処理
000420     END-IF
000430*
000440     ACCEPT 日付情報１ FROM 引数の値.
000450     IF 年 OF 年月日１ < 1900 THEN
000460        DISPLAY "入力日付は１９００年以降の日付を入力してください. "
000470        GO TO 終了処理
000480     END-IF.
000490     ACCEPT 日付情報２ FROM 引数の値.
000500     IF 年 OF 年月日２ < 1900 THEN
000510        DISPLAY "入力日付は１９００年以降の日付を入力してください. "
000520        GO TO 終了処理
000530     END-IF.
000540     CALL "SUMCALC"  USING 年月日１ 通算日１.
000550     CALL "SUMCALC"  USING 年月日２ 通算日２.
000560*
000570     COMPUTE 日付の差 = 通算日２ - 通算日１.
000580     DISPLAY "日数の差は  " 日付の差 "日です. ".
000590*
000600 終了処理.
000610     EXIT PROGRAM.
000620*
000630 END PROGRAM  DAYSBETWEEN.
000640
```

ソースコード：MainForm.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "Schedule.MainForm"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   DELEGATE DELEGATE-REGISTRATION AS "Schedule.Registration"
   CLASS FORM-REGISTRATION AS "Schedule.RegistrationForm"
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-DATETIME AS "System.DateTime"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   ENUM ENUM-ANCHORSTYLES AS "System.Windows.Forms.AnchorStyles"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-COLUMNHEADER AS "System.Windows.Forms.ColumnHeader"
   CLASS ARRAY-COLUMNHEADER AS "System.Windows.Forms.ColumnHeader[]"
   ENUM ENUM-COLUMNHEADERSTYLE AS "System.Windows.Forms.ColumnHeaderStyle"
   CLASS CLASS-CONTEXTMENUSTRIP AS "System.Windows.Forms.ContextMenuStrip"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   CLASS CLASS-LISTVIEW AS "System.Windows.Forms.ListView"
   CLASS CLASS-COLUMNHEADERCOLLECTION AS
               "System.Windows.Forms.ListView+ColumnHeaderCollection"
   CLASS COLL-LISTVIEWITEMS AS "System.Windows.Forms.ListView+ListViewItemCollection"
   CLASS COLL-SELECTEDLISTVIEWITEMS AS
               "System.Windows.Forms.ListView+SelectedListViewItemCollection"
   CLASS CLASS-LISTVIEWITEM AS "System.Windows.Forms.ListViewItem"
   CLASS CLASS-LISTVIEWSUBITEM AS "System.Windows.Forms.ListViewItem+ListViewSubItem"
   CLASS COLL-LISTVIEWSUBITEMS AS
               "System.Windows.Forms.ListViewItem+ListViewSubItemCollection"
   CLASS CLASS-MENUSTRIP AS "System.Windows.Forms.MenuStrip"
   ENUM ENUM-SORTORDER AS "System.Windows.Forms.SortOrder"
   CLASS ARRAY-TOOLSTRIPITEM AS "System.Windows.Forms.ToolStripItem[]"
   CLASS CLASS-TOOLSTRIPITEMCOLLECTION AS "System.Windows.Forms.ToolStripItemCollection"
   CLASS CLASS-TOOLSTRIPMENUITEM AS "System.Windows.Forms.ToolStripMenuItem"
   ENUM ENUM-VIEW AS "System.Windows.Forms.View"
   PROPERTY PROP-ANCHOR AS "Anchor"
   PROPERTY PROP-ASCENDING AS "Ascending"
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-BOTTOM AS "Bottom"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-COLUMNHEADER1 AS "columnHeader1"
   PROPERTY PROP-COLUMNHEADER2 AS "columnHeader2"
   PROPERTY PROP-COLUMNS AS "Columns"
   PROPERTY PROP-COMPONENTS AS "components"
   PROPERTY PROP-CONTEXTMENUSTRIP AS "ContextMenuStrip"
   PROPERTY PROP-CONTEXTMENUSTRIP1 AS "contextMenuStrip1"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-COUNT AS "Count"
   PROPERTY PROP-DETAILS AS "Details"
   PROPERTY PROP-DROPDOWNITEMS AS "DropDownItems"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-FULLROWSELECT AS "FullRowSelect"
   PROPERTY PROP-HEADERSTYLE AS "HeaderStyle"
   PROPERTY PROP-ITEMS AS "Items"
   PROPERTY PROP-LEFT AS "Left"
   PROPERTY PROP-LISTVIEW1 AS "listView1"
   PROPERTY PROP-LOCATION AS "Location"
   PROPERTY PROP-MAINMENUSTRIP AS "MainMenuStrip"
   PROPERTY PROP-MENUSTRIP1 AS "menuStrip1"
```

```
          PROPERTY PROP-MULTISELECT AS "MultiSelect"
          PROPERTY PROP-NAME AS "Name"
          PROPERTY PROP-NONCLICKABLE AS "Nonclickable"
          PROPERTY PROP-RIGHT AS "Right"
          PROPERTY PROP-SELECTEDITEMS AS "SelectedItems"
          PROPERTY PROP-SIZE AS "Size"
          PROPERTY PROP-SORTING AS "Sorting"
          PROPERTY PROP-SUBITEMS AS "SubItems"
          PROPERTY PROP-TABINDEX AS "TabIndex"
          PROPERTY PROP-TEXT AS "Text"
          PROPERTY PROP-TOOLSTRIPMENUITEM1 AS "toolStripMenuItem1"
          PROPERTY PROP-TOOLSTRIPMENUITEM10 AS "toolStripMenuItem10"
          PROPERTY PROP-TOOLSTRIPMENUITEM2 AS "toolStripMenuItem2"
          PROPERTY PROP-TOOLSTRIPMENUITEM3 AS "toolStripMenuItem3"
          PROPERTY PROP-TOOLSTRIPMENUITEM5 AS "toolStripMenuItem5"
          PROPERTY PROP-TOOLSTRIPMENUITEM6 AS "toolStripMenuItem6"
          PROPERTY PROP-TOOLSTRIPMENUITEM7 AS "toolStripMenuItem7"
          PROPERTY PROP-TOOLSTRIPMENUITEM8 AS "toolStripMenuItem8"
          PROPERTY PROP-TOOLSTRIPMENUITEM9 AS "toolStripMenuItem9"
          PROPERTY PROP-TOP AS "Top"
          PROPERTY PROP-USECOMPATIBLESTATEIMAGEBE AS "UseCompatibleStateImageBehavior"
          PROPERTY PROP-VIEW AS "View"
          PROPERTY PROP-WIDTH AS "Width"
          .

       OBJECT.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 toolStripMenuItem1 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 toolStripMenuItem3 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 toolStripMenuItem2 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 toolStripMenuItem5 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 menuStrip1 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 toolStripMenuItem6 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 toolStripMenuItem7 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 contextMenuStrip1 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       01 toolStripMenuItem8 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 toolStripMenuItem9 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 toolStripMenuItem10 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 listView1 OBJECT REFERENCE CLASS-LISTVIEW.
       01 columnHeader1 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 columnHeader2 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
       01 registrationForm1 OBJECT REFERENCE FORM-REGISTRATION.
       PROCEDURE DIVISION.

       METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING BY VALUE disposing.
          IF disposing NOT = B"0" AND components NOT = NULL THEN
            INVOKE components "Dispose"
          END-IF.
          INVOKE SUPER "Dispose" USING disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-CONTAINER.
       01 TEMP2 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP3 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP4 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP5 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP6 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP7 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP8 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP9 OBJECT REFERENCE INTERFACE-ICONTAINER.
       01 TEMP10 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       01 TEMP11 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP12 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP13 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP14 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP15 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP16 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP17 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP18 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       01 TEMP19 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP20 OBJECT REFERENCE CLASS-TOOLSTRIPITEMCOLLECTION.
       01 TEMP21 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
```

```
       01 TEMP22 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP23 BINARY-LONG.
       01 TEMP24 OBJECT REFERENCE ARRAY-TOOLSTRIPITEM.
       01 TEMP25 BINARY-LONG.
       01 TEMP26 BINARY-LONG.
       01 TEMP27 OBJECT REFERENCE CLASS-POINT.
       01 TEMP28 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP29 OBJECT REFERENCE CLASS-STRING.
       01 TEMP30 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP31 BINARY-LONG.
       01 TEMP32 BINARY-LONG.
       01 TEMP33 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP34 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP35 BINARY-LONG.
       01 TEMP36 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP37 OBJECT REFERENCE CLASS-STRING.
       01 TEMP38 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP39 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP40 OBJECT REFERENCE CLASS-TOOLSTRIPITEMCOLLECTION.
       01 TEMP41 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP42 BINARY-LONG.
       01 TEMP43 OBJECT REFERENCE ARRAY-TOOLSTRIPITEM.
       01 TEMP44 OBJECT REFERENCE CLASS-STRING.
       01 TEMP45 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP46 BINARY-LONG.
       01 TEMP47 BINARY-LONG.
       01 TEMP48 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP49 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP50 OBJECT REFERENCE CLASS-STRING.
       01 TEMP51 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP52 OBJECT REFERENCE CLASS-STRING.
       01 TEMP53 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP54 BINARY-LONG.
       01 TEMP55 BINARY-LONG.
       01 TEMP56 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP57 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP58 OBJECT REFERENCE CLASS-STRING.
       01 TEMP59 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP60 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP61 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP62 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP63 OBJECT REFERENCE CLASS-TOOLSTRIPITEMCOLLECTION.
       01 TEMP64 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP65 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP66 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP67 BINARY-LONG.
       01 TEMP68 OBJECT REFERENCE ARRAY-TOOLSTRIPITEM.
       01 TEMP69 OBJECT REFERENCE CLASS-STRING.
       01 TEMP70 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP71 BINARY-LONG.
       01 TEMP72 BINARY-LONG.
       01 TEMP73 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP74 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP75 OBJECT REFERENCE CLASS-STRING.
       01 TEMP76 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP77 OBJECT REFERENCE CLASS-STRING.
       01 TEMP78 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP79 BINARY-LONG.
       01 TEMP80 BINARY-LONG.
       01 TEMP81 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP82 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP83 OBJECT REFERENCE CLASS-STRING.
       01 TEMP84 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP85 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP86 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP87 OBJECT REFERENCE CLASS-STRING.
       01 TEMP88 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP89 BINARY-LONG.
       01 TEMP90 BINARY-LONG.
       01 TEMP91 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP92 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP93 OBJECT REFERENCE CLASS-STRING.
       01 TEMP94 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP95 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP96 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP97 OBJECT REFERENCE CLASS-STRING.
       01 TEMP98 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP99 BINARY-LONG.
       01 TEMP100 BINARY-LONG.
       01 TEMP101 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP102 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP103 OBJECT REFERENCE CLASS-STRING.
```

```
       01 TEMP104 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP105 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP106 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP107 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       01 TEMP108 OBJECT REFERENCE CLASS-TOOLSTRIPITEMCOLLECTION.
       01 TEMP109 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP110 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP111 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP112 BINARY-LONG.
       01 TEMP113 OBJECT REFERENCE ARRAY-TOOLSTRIPITEM.
       01 TEMP114 OBJECT REFERENCE CLASS-STRING.
       01 TEMP115 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       01 TEMP116 BINARY-LONG.
       01 TEMP117 BINARY-LONG.
       01 TEMP118 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP119 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       01 TEMP120 OBJECT REFERENCE CLASS-STRING.
       01 TEMP121 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP122 BINARY-LONG.
       01 TEMP123 BINARY-LONG.
       01 TEMP124 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP125 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP126 OBJECT REFERENCE CLASS-STRING.
       01 TEMP127 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP128 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP129 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP130 OBJECT REFERENCE CLASS-STRING.
       01 TEMP131 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP132 BINARY-LONG.
       01 TEMP133 BINARY-LONG.
       01 TEMP134 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP135 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP136 OBJECT REFERENCE CLASS-STRING.
       01 TEMP137 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP138 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP139 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP140 OBJECT REFERENCE CLASS-STRING.
       01 TEMP141 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP142 BINARY-LONG.
       01 TEMP143 BINARY-LONG.
       01 TEMP144 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP145 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP146 OBJECT REFERENCE CLASS-STRING.
       01 TEMP147 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP148 OBJECT REFERENCE CLASS-TOOLSTRIPMENUITEM.
       01 TEMP149 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP150 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP151 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP152 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP153 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP154 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP155 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP156 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP157 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP158 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP159 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP160 OBJECT REFERENCE CLASS-COLUMNHEADERCOLLECTION.
       01 TEMP161 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP162 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP163 BINARY-LONG.
       01 TEMP164 OBJECT REFERENCE ARRAY-COLUMNHEADER.
       01 TEMP165 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP166 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP167 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP168 BINARY-LONG.
       01 TEMP169 BINARY-LONG.
       01 TEMP170 OBJECT REFERENCE CLASS-POINT.
       01 TEMP171 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP172 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP173 OBJECT REFERENCE CLASS-STRING.
       01 TEMP174 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP175 BINARY-LONG.
       01 TEMP176 BINARY-LONG.
       01 TEMP177 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP178 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP179 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP180 BINARY-LONG.
       01 TEMP181 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP182 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP183 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP184 OBJECT REFERENCE CLASS-STRING.
       01 TEMP185 OBJECT REFERENCE CLASS-COLUMNHEADER.
```

```
       01 TEMP186 BINARY-LONG.
       01 TEMP187 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP188 OBJECT REFERENCE CLASS-STRING.
       01 TEMP189 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP190 BINARY-LONG.
       01 TEMP191 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP192 COMP-1.
       01 TEMP193 COMP-1.
       01 TEMP194 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP195 BINARY-LONG.
       01 TEMP196 BINARY-LONG.
       01 TEMP197 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP198 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP199 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP200 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP201 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP202 OBJECT REFERENCE CLASS-STRING.
       01 TEMP203 OBJECT REFERENCE CLASS-STRING.
       01 TEMP204 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP205 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP206 OBJECT REFERENCE CLASS-MENUSTRIP.
       01 TEMP207 OBJECT REFERENCE CLASS-CONTEXTMENUSTRIP.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="components" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.Container" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.MenuStrip" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem6" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem7" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ContextMenuStrip" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="components" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem9" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripMenuItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
```

```
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem10" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ToolStripMenuItem" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="listView1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ListView" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="columnHeader1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ColumnHeader" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ColumnHeader" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
```

```
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Items" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="AddRange" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem1" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="292" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="24" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripMenuItem1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem1" />
      *</prop>
```

```
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="DropDownItems" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="AddRange" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeArrayCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ToolStripItem" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Initializers" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem3" />
*</prop>
*</object>
*</prop>
*<prop name="Size">
*<int32 value="0" />
*</prop>
*<prop name="SizeExpression">
*<null />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="toolStripMenuItem1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
```

```
      *<prop name="FieldName">
      *<string value="toolStripMenuItem1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="66" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="ファイル(&amp;amp;F)" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="109" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="終了(&amp;amp;X)" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="toolStripMenuItem3_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
```

```
*<prop name="Text">
*<string value="toolStripMenuItem2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="DropDownItems" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="AddRange" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeArrayCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ToolStripItem" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Initializers" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem5" />
*</prop>
*</object>
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem6" />
*</prop>
*</object>
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem7" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="56" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="編集(&amp;amp;E)" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="toolStripMenuItem5" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem5" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="toolStripMenuItem5" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="234" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="新規に登録ウィンドウを開く(&amp;amp;N)" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="toolStripMenuItem5_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripMenuItem6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem6" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem6" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="234" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem6" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="選択された記事をコピーして開く(&amp;amp;C)" />
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem6" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="toolStripMenuItem6_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="toolStripMenuItem7" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem7" />
*</prop>
*</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem7" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem7" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="234" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem7" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
*<string value="選択された記事の削除(&amp;amp;D)" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem7" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="toolStripMenuItem7_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="contextMenuStrip1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Items" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="AddRange" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem9" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
```

```
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="219" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="70" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
```

```
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="218" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
```

```
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="新規に登録ウィンドウを開く" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem8" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="toolStripMenuItem5_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripMenuItem9" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem9" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem9" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="218" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem9" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="選択された記事をコピーして開く" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="toolStripMenuItem9" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="toolStripMenuItem6_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
```

```
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="218" />
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="選択された記事の削除" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripMenuItem10" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="toolStripMenuItem7_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Left" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
```

```
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Columns" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="AddRange" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ColumnHeader" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="ContextMenuStrip" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="FullRowSelect" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="HeaderStyle" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ColumnHeaderStyle" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Nonclickable" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="27" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MultiSelect" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="268" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="227" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="PropertyName">
      *<string value="Sorting" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.SortOrder" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Ascending" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseCompatibleStateImageBehavior" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
```

```
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="View" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.View" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Details" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="日付" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Width" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="80" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="記事" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Width" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="200" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="MainForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleDimensions" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
```

```
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.SizeF" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="6" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AutoScaleMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Font" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="292" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="266" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MainMenuStrip" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="MainForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="スケジュール" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Load" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
```

```
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="MainForm_Load" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="menuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="contextMenuStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-CONTAINER "NEW" RETURNING TEMP1
          SET PROP-COMPONENTS OF SELF TO TEMP1
          INVOKE CLASS-MENUSTRIP "NEW" RETURNING TEMP2
          SET PROP-MENUSTRIP1 OF SELF TO TEMP2
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP3
          SET PROP-TOOLSTRIPMENUITEM1 OF SELF TO TEMP3
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP4
          SET PROP-TOOLSTRIPMENUITEM3 OF SELF TO TEMP4
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP5
          SET PROP-TOOLSTRIPMENUITEM2 OF SELF TO TEMP5
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP6
          SET PROP-TOOLSTRIPMENUITEM5 OF SELF TO TEMP6
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP7
          SET PROP-TOOLSTRIPMENUITEM6 OF SELF TO TEMP7
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP8
```

```
          SET PROP-TOOLSTRIPMENUITEM7 OF SELF TO TEMP8
          SET TEMP9 TO PROP-COMPONENTS OF SELF
          INVOKE CLASS-CONTEXTMENUSTRIP "NEW" USING BY VALUE TEMP9 RETURNING TEMP10
          SET PROP-CONTEXTMENUSTRIP1 OF SELF TO TEMP10
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP11
          SET PROP-TOOLSTRIPMENUITEM8 OF SELF TO TEMP11
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP12
          SET PROP-TOOLSTRIPMENUITEM9 OF SELF TO TEMP12
          INVOKE CLASS-TOOLSTRIPMENUITEM "NEW" RETURNING TEMP13
          SET PROP-TOOLSTRIPMENUITEM10 OF SELF TO TEMP13
          INVOKE CLASS-LISTVIEW "NEW" RETURNING TEMP14
          SET PROP-LISTVIEW1 OF SELF TO TEMP14
          INVOKE CLASS-COLUMNHEADER "NEW" RETURNING TEMP15
          SET PROP-COLUMNHEADER1 OF SELF TO TEMP15
          INVOKE CLASS-COLUMNHEADER "NEW" RETURNING TEMP16
          SET PROP-COLUMNHEADER2 OF SELF TO TEMP16
          SET TEMP17 TO PROP-MENUSTRIP1 OF SELF
          INVOKE TEMP17 "SuspendLayout"
          SET TEMP18 TO PROP-CONTEXTMENUSTRIP1 OF SELF
          INVOKE TEMP18 "SuspendLayout"
          INVOKE SELF "SuspendLayout"
      *
      *menuStrip1
      *
          MOVE 2 TO TEMP23
          INVOKE ARRAY-TOOLSTRIPITEM "NEW" USING BY VALUE TEMP23 RETURNING TEMP24
          SET TEMP21 TO PROP-TOOLSTRIPMENUITEM1 OF SELF
          INVOKE TEMP24 "Set" USING BY VALUE 0 BY VALUE TEMP21
          SET TEMP22 TO PROP-TOOLSTRIPMENUITEM2 OF SELF
          INVOKE TEMP24 "Set" USING BY VALUE 1 BY VALUE TEMP22
          SET TEMP19 TO PROP-MENUSTRIP1 OF SELF
          SET TEMP20 TO PROP-ITEMS OF TEMP19
          INVOKE TEMP20 "AddRange" USING BY VALUE TEMP24
          MOVE 0 TO TEMP25
          MOVE 0 TO TEMP26
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP25 BY VALUE TEMP26 RETURNING TEMP27
          SET TEMP28 TO PROP-MENUSTRIP1 OF SELF
          SET PROP-LOCATION OF TEMP28 TO TEMP27
          SET TEMP29 TO N"menuStrip1"
          SET TEMP30 TO PROP-MENUSTRIP1 OF SELF
          SET PROP-NAME OF TEMP30 TO TEMP29
          MOVE 292 TO TEMP31
          MOVE 24 TO TEMP32
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP31 BY VALUE TEMP32 RETURNING TEMP33
          SET TEMP34 TO PROP-MENUSTRIP1 OF SELF
          SET PROP-SIZE OF TEMP34 TO TEMP33
          MOVE 0 TO TEMP35
          SET TEMP36 TO PROP-MENUSTRIP1 OF SELF
          MOVE TEMP35 TO PROP-TABINDEX OF TEMP36
          SET TEMP37 TO N"menuStrip1"
          SET TEMP38 TO PROP-MENUSTRIP1 OF SELF
          SET PROP-TEXT OF TEMP38 TO TEMP37
      *
      *toolStripMenuItem1
      *
          MOVE 1 TO TEMP42
          INVOKE ARRAY-TOOLSTRIPITEM "NEW" USING BY VALUE TEMP42 RETURNING TEMP43
          SET TEMP41 TO PROP-TOOLSTRIPMENUITEM3 OF SELF
          INVOKE TEMP43 "Set" USING BY VALUE 0 BY VALUE TEMP41
          SET TEMP39 TO PROP-TOOLSTRIPMENUITEM1 OF SELF
          SET TEMP40 TO PROP-DROPDOWNITEMS OF TEMP39
          INVOKE TEMP40 "AddRange" USING BY VALUE TEMP43
          SET TEMP44 TO N"toolStripMenuItem1"
          SET TEMP45 TO PROP-TOOLSTRIPMENUITEM1 OF SELF
          SET PROP-NAME OF TEMP45 TO TEMP44
          MOVE 66 TO TEMP46
          MOVE 20 TO TEMP47
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP46 BY VALUE TEMP47 RETURNING TEMP48
          SET TEMP49 TO PROP-TOOLSTRIPMENUITEM1 OF SELF
          SET PROP-SIZE OF TEMP49 TO TEMP48
          SET TEMP50 TO N"ファイル(&F)"
          SET TEMP51 TO PROP-TOOLSTRIPMENUITEM1 OF SELF
          SET PROP-TEXT OF TEMP51 TO TEMP50
      *
      *toolStripMenuItem3
      *
          SET TEMP52 TO N"toolStripMenuItem3"
          SET TEMP53 TO PROP-TOOLSTRIPMENUITEM3 OF SELF
          SET PROP-NAME OF TEMP53 TO TEMP52
          MOVE 109 TO TEMP54
          MOVE 22 TO TEMP55
```

```
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP54 BY VALUE TEMP55 RETURNING TEMP56
          SET TEMP57 TO PROP-TOOLSTRIPMENUITEM3 OF SELF
          SET PROP-SIZE OF TEMP57 TO TEMP56
          SET TEMP58 TO N"終了(&X)"
          SET TEMP59 TO PROP-TOOLSTRIPMENUITEM3 OF SELF
          SET PROP-TEXT OF TEMP59 TO TEMP58
          SET TEMP60 TO PROP-TOOLSTRIPMENUITEM3 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                         BY VALUE N"toolStripMenuItem3_Click" RETURNING TEMP61
          INVOKE TEMP60 "add_Click" USING BY VALUE TEMP61
      *
      *toolStripMenuItem2
      *
          MOVE 3 TO TEMP67
          INVOKE ARRAY-TOOLSTRIPITEM "NEW" USING BY VALUE TEMP67 RETURNING TEMP68
          SET TEMP64 TO PROP-TOOLSTRIPMENUITEM5 OF SELF
          INVOKE TEMP68 "Set" USING BY VALUE 0 BY VALUE TEMP64
          SET TEMP65 TO PROP-TOOLSTRIPMENUITEM6 OF SELF
          INVOKE TEMP68 "Set" USING BY VALUE 1 BY VALUE TEMP65
          SET TEMP66 TO PROP-TOOLSTRIPMENUITEM7 OF SELF
          INVOKE TEMP68 "Set" USING BY VALUE 2 BY VALUE TEMP66
          SET TEMP62 TO PROP-TOOLSTRIPMENUITEM2 OF SELF
          SET TEMP63 TO PROP-DROPDOWNITEMS OF TEMP62
          INVOKE TEMP63 "AddRange" USING BY VALUE TEMP68
          SET TEMP69 TO N"toolStripMenuItem2"
          SET TEMP70 TO PROP-TOOLSTRIPMENUITEM2 OF SELF
          SET PROP-NAME OF TEMP70 TO TEMP69
          MOVE 56 TO TEMP71
          MOVE 20 TO TEMP72
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP71 BY VALUE TEMP72 RETURNING TEMP73
          SET TEMP74 TO PROP-TOOLSTRIPMENUITEM2 OF SELF
          SET PROP-SIZE OF TEMP74 TO TEMP73
          SET TEMP75 TO N"編集(&E)"
          SET TEMP76 TO PROP-TOOLSTRIPMENUITEM2 OF SELF
          SET PROP-TEXT OF TEMP76 TO TEMP75
      *
      *toolStripMenuItem5
      *
          SET TEMP77 TO N"toolStripMenuItem5"
          SET TEMP78 TO PROP-TOOLSTRIPMENUITEM5 OF SELF
          SET PROP-NAME OF TEMP78 TO TEMP77
          MOVE 234 TO TEMP79
          MOVE 22 TO TEMP80
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP79 BY VALUE TEMP80 RETURNING TEMP81
          SET TEMP82 TO PROP-TOOLSTRIPMENUITEM5 OF SELF
          SET PROP-SIZE OF TEMP82 TO TEMP81
          SET TEMP83 TO N"新規に登録ウィンドウを開く(&N)"
          SET TEMP84 TO PROP-TOOLSTRIPMENUITEM5 OF SELF
          SET PROP-TEXT OF TEMP84 TO TEMP83
          SET TEMP85 TO PROP-TOOLSTRIPMENUITEM5 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                          BY VALUE N"toolStripMenuItem5_Click" RETURNING TEMP86
          INVOKE TEMP85 "add_Click" USING BY VALUE TEMP86
      *
      *toolStripMenuItem6
      *
          SET TEMP87 TO N"toolStripMenuItem6"
          SET TEMP88 TO PROP-TOOLSTRIPMENUITEM6 OF SELF
          SET PROP-NAME OF TEMP88 TO TEMP87
          MOVE 234 TO TEMP89
          MOVE 22 TO TEMP90
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP89 BY VALUE TEMP90 RETURNING TEMP91
          SET TEMP92 TO PROP-TOOLSTRIPMENUITEM6 OF SELF
          SET PROP-SIZE OF TEMP92 TO TEMP91
          SET TEMP93 TO N"選択された記事をコピーして開く(&C)"
          SET TEMP94 TO PROP-TOOLSTRIPMENUITEM6 OF SELF
          SET PROP-TEXT OF TEMP94 TO TEMP93
          SET TEMP95 TO PROP-TOOLSTRIPMENUITEM6 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE
N"toolStripMenuItem6_Click" RETURNING TEMP96
          INVOKE TEMP95 "add_Click" USING BY VALUE TEMP96
      *
      *toolStripMenuItem7
      *
          SET TEMP97 TO N"toolStripMenuItem7"
          SET TEMP98 TO PROP-TOOLSTRIPMENUITEM7 OF SELF
          SET PROP-NAME OF TEMP98 TO TEMP97
          MOVE 234 TO TEMP99
          MOVE 22 TO TEMP100
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP99 BY VALUE TEMP100 RETURNING TEMP101
          SET TEMP102 TO PROP-TOOLSTRIPMENUITEM7 OF SELF
```

```
        SET PROP-SIZE OF TEMP102 TO TEMP101
        SET TEMP103 TO N"選択された記事の削除(&D)"
        SET TEMP104 TO PROP-TOOLSTRIPMENUITEM7 OF SELF
        SET PROP-TEXT OF TEMP104 TO TEMP103
        SET TEMP105 TO PROP-TOOLSTRIPMENUITEM7 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
               BY VALUE N"toolStripMenuItem7_Click" RETURNING TEMP106
        INVOKE TEMP105 "add_Click" USING BY VALUE TEMP106
    *
    *contextMenuStrip1
    *
        MOVE 3 TO TEMP112
        INVOKE ARRAY-TOOLSTRIPITEM "NEW" USING BY VALUE TEMP112 RETURNING TEMP113
        SET TEMP109 TO PROP-TOOLSTRIPMENUITEM8 OF SELF
        INVOKE TEMP113 "Set" USING BY VALUE 0 BY VALUE TEMP109
        SET TEMP110 TO PROP-TOOLSTRIPMENUITEM9 OF SELF
        INVOKE TEMP113 "Set" USING BY VALUE 1 BY VALUE TEMP110
        SET TEMP111 TO PROP-TOOLSTRIPMENUITEM10 OF SELF
        INVOKE TEMP113 "Set" USING BY VALUE 2 BY VALUE TEMP111
        SET TEMP107 TO PROP-CONTEXTMENUSTRIP1 OF SELF
        SET TEMP108 TO PROP-ITEMS OF TEMP107
        INVOKE TEMP108 "AddRange" USING BY VALUE TEMP113
        SET TEMP114 TO N"contextMenuStrip1"
        SET TEMP115 TO PROP-CONTEXTMENUSTRIP1 OF SELF
        SET PROP-NAME OF TEMP115 TO TEMP114
        MOVE 219 TO TEMP116
        MOVE 70 TO TEMP117
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP116 BY VALUE TEMP117 RETURNING TEMP118
        SET TEMP119 TO PROP-CONTEXTMENUSTRIP1 OF SELF
        SET PROP-SIZE OF TEMP119 TO TEMP118
    *
    *toolStripMenuItem8
    *
        SET TEMP120 TO N"toolStripMenuItem8"
        SET TEMP121 TO PROP-TOOLSTRIPMENUITEM8 OF SELF
        SET PROP-NAME OF TEMP121 TO TEMP120
        MOVE 218 TO TEMP122
        MOVE 22 TO TEMP123
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP122 BY VALUE TEMP123 RETURNING TEMP124
        SET TEMP125 TO PROP-TOOLSTRIPMENUITEM8 OF SELF
        SET PROP-SIZE OF TEMP125 TO TEMP124
        SET TEMP126 TO N"新規に登録ウィンドウを開く"
        SET TEMP127 TO PROP-TOOLSTRIPMENUITEM8 OF SELF
        SET PROP-TEXT OF TEMP127 TO TEMP126
        SET TEMP128 TO PROP-TOOLSTRIPMENUITEM8 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                BY VALUE N"toolStripMenuItem5_Click" RETURNING TEMP129
        INVOKE TEMP128 "add_Click" USING BY VALUE TEMP129
    *
    *toolStripMenuItem9
    *
        SET TEMP130 TO N"toolStripMenuItem9"
        SET TEMP131 TO PROP-TOOLSTRIPMENUITEM9 OF SELF
        SET PROP-NAME OF TEMP131 TO TEMP130
        MOVE 218 TO TEMP132
        MOVE 22 TO TEMP133
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP132 BY VALUE TEMP133 RETURNING TEMP134
        SET TEMP135 TO PROP-TOOLSTRIPMENUITEM9 OF SELF
        SET PROP-SIZE OF TEMP135 TO TEMP134
        SET TEMP136 TO N"選択された記事をコピーして開く"
        SET TEMP137 TO PROP-TOOLSTRIPMENUITEM9 OF SELF
        SET PROP-TEXT OF TEMP137 TO TEMP136
        SET TEMP138 TO PROP-TOOLSTRIPMENUITEM9 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                BY VALUE N"toolStripMenuItem6_Click" RETURNING TEMP139
        INVOKE TEMP138 "add_Click" USING BY VALUE TEMP139
    *
    *toolStripMenuItem10
    *
        SET TEMP140 TO N"toolStripMenuItem10"
        SET TEMP141 TO PROP-TOOLSTRIPMENUITEM10 OF SELF
        SET PROP-NAME OF TEMP141 TO TEMP140
        MOVE 218 TO TEMP142
        MOVE 22 TO TEMP143
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP142 BY VALUE TEMP143 RETURNING TEMP144
        SET TEMP145 TO PROP-TOOLSTRIPMENUITEM10 OF SELF
        SET PROP-SIZE OF TEMP145 TO TEMP144
        SET TEMP146 TO N"選択された記事の削除"
        SET TEMP147 TO PROP-TOOLSTRIPMENUITEM10 OF SELF
        SET PROP-TEXT OF TEMP147 TO TEMP146
        SET TEMP148 TO PROP-TOOLSTRIPMENUITEM10 OF SELF
```

```
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                    BY VALUE N"toolStripMenuItem7_Click" RETURNING TEMP149
          INVOKE TEMP148 "add_Click" USING BY VALUE TEMP149
      *
      *listView1
      *
          SET TEMP150 TO PROP-TOP OF ENUM-ANCHORSTYLES
          SET TEMP151 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
          SET TEMP152 TO FUNCTION ENUM-OR(TEMP150 TEMP151)
          SET TEMP153 TO PROP-LEFT OF ENUM-ANCHORSTYLES
          SET TEMP154 TO FUNCTION ENUM-OR(TEMP152 TEMP153)
          SET TEMP155 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
          SET TEMP156 TO FUNCTION ENUM-OR(TEMP154 TEMP155)
          SET TEMP157 TO TEMP156
          SET TEMP158 TO PROP-LISTVIEW1 OF SELF
          SET PROP-ANCHOR OF TEMP158 TO TEMP157
          MOVE 2 TO TEMP163
          INVOKE ARRAY-COLUMNHEADER "NEW" USING BY VALUE TEMP163 RETURNING TEMP164
          SET TEMP161 TO PROP-COLUMNHEADER1 OF SELF
          INVOKE TEMP164 "Set" USING BY VALUE 0 BY VALUE TEMP161
          SET TEMP162 TO PROP-COLUMNHEADER2 OF SELF
          INVOKE TEMP164 "Set" USING BY VALUE 1 BY VALUE TEMP162
          SET TEMP159 TO PROP-LISTVIEW1 OF SELF
          SET TEMP160 TO PROP-COLUMNS OF TEMP159
          INVOKE TEMP160 "AddRange" USING BY VALUE TEMP164
          SET TEMP165 TO PROP-LISTVIEW1 OF SELF
          SET PROP-CONTEXTMENUSTRIP OF TEMP165 TO PROP-CONTEXTMENUSTRIP1 OF SELF
          SET TEMP166 TO PROP-LISTVIEW1 OF SELF
          SET PROP-FULLROWSELECT OF TEMP166 TO B"1"
          SET TEMP167 TO PROP-LISTVIEW1 OF SELF
          SET PROP-HEADERSTYLE OF TEMP167 TO PROP-NONCLICKABLE OF ENUM-COLUMNHEADERSTYLE
          MOVE 12 TO TEMP168
          MOVE 27 TO TEMP169
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP168 BY VALUE TEMP169 RETURNING TEMP170
          SET TEMP171 TO PROP-LISTVIEW1 OF SELF
          SET PROP-LOCATION OF TEMP171 TO TEMP170
          SET TEMP172 TO PROP-LISTVIEW1 OF SELF
          SET PROP-MULTISELECT OF TEMP172 TO B"0"
          SET TEMP173 TO N"listView1"
          SET TEMP174 TO PROP-LISTVIEW1 OF SELF
          SET PROP-NAME OF TEMP174 TO TEMP173
          MOVE 268 TO TEMP175
          MOVE 227 TO TEMP176
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP175 BY VALUE TEMP176 RETURNING TEMP177
          SET TEMP178 TO PROP-LISTVIEW1 OF SELF
          SET PROP-SIZE OF TEMP178 TO TEMP177
          SET TEMP179 TO PROP-LISTVIEW1 OF SELF
          SET PROP-SORTING OF TEMP179 TO PROP-ASCENDING OF ENUM-SORTORDER
          MOVE 1 TO TEMP180
          SET TEMP181 TO PROP-LISTVIEW1 OF SELF
          MOVE TEMP180 TO PROP-TABINDEX OF TEMP181
          SET TEMP182 TO PROP-LISTVIEW1 OF SELF
          SET PROP-USECOMPATIBLESTATEIMAGEBE OF TEMP182 TO B"0"
          SET TEMP183 TO PROP-LISTVIEW1 OF SELF
          SET PROP-VIEW OF TEMP183 TO PROP-DETAILS OF ENUM-VIEW
      *
      *columnHeader1
      *
          SET TEMP184 TO N"日付"
          SET TEMP185 TO PROP-COLUMNHEADER1 OF SELF
          SET PROP-TEXT OF TEMP185 TO TEMP184
          MOVE 80 TO TEMP186
          SET TEMP187 TO PROP-COLUMNHEADER1 OF SELF
          MOVE TEMP186 TO PROP-WIDTH OF TEMP187
      *
      *columnHeader2
      *
          SET TEMP188 TO N"記事"
          SET TEMP189 TO PROP-COLUMNHEADER2 OF SELF
          SET PROP-TEXT OF TEMP189 TO TEMP188
          MOVE 200 TO TEMP190
          SET TEMP191 TO PROP-COLUMNHEADER2 OF SELF
          MOVE TEMP190 TO PROP-WIDTH OF TEMP191
      *
      *MainForm
      *
          MOVE 6.00000000000000E+00 TO TEMP192
          MOVE 1.20000000000000E+01 TO TEMP193
          INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP192 BY VALUE TEMP193 RETURNING TEMP194
          SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP194
          SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
```

```
           MOVE 292 TO TEMP195
           MOVE 266 TO TEMP196
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP195 BY VALUE TEMP196 RETURNING TEMP197
           SET PROP-CLIENTSIZE OF SELF TO TEMP197
           SET TEMP199 TO PROP-MENUSTRIP1 OF SELF
           SET TEMP198 TO PROP-CONTROLS OF SELF
           INVOKE TEMP198 "Add" USING BY VALUE TEMP199
           SET TEMP201 TO PROP-LISTVIEW1 OF SELF
           SET TEMP200 TO PROP-CONTROLS OF SELF
           INVOKE TEMP200 "Add" USING BY VALUE TEMP201
           SET PROP-MAINMENUSTRIP OF SELF TO PROP-MENUSTRIP1 OF SELF
           SET TEMP202 TO N"MainForm"
           SET PROP-NAME OF SELF TO TEMP202
           SET TEMP203 TO N"スケジュール"
           SET PROP-TEXT OF SELF TO TEMP203
           INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"MainForm_Load"
                  RETURNING TEMP204
           INVOKE SELF "add_Load" USING BY VALUE TEMP204
           SET TEMP205 TO PROP-MENUSTRIP1 OF SELF
           INVOKE TEMP205 "ResumeLayout" USING BY VALUE B"0"
           SET TEMP206 TO PROP-MENUSTRIP1 OF SELF
           INVOKE TEMP206 "PerformLayout"
           SET TEMP207 TO PROP-CONTEXTMENUSTRIP1 OF SELF
           INVOKE TEMP207 "ResumeLayout" USING BY VALUE B"0"
           INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
           INVOKE SELF "PerformLayout"
       END METHOD INITIALIZECOMPONENT.

       METHOD-ID. NEW.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
           INVOKE SELF "InitializeComponent".
       END METHOD NEW.

       METHOD-ID. Clear PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 lvItems OBJECT REFERENCE COLL-LISTVIEWITEMS.
       PROCEDURE DIVISION.
      * 初期設定
           SET lvItems TO PROP-ITEMS OF listView1.
           INVOKE lvItems "Clear".
       END METHOD Clear.

       METHOD-ID. Registration_Disposed PRIVATE.
       DATA DIVISION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  閉じられた登録フォームをクリアします
           SET registrationForm1 TO NULL.
       END METHOD Registration_Disposed.

       METHOD-ID. Registration.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 strDate OBJECT REFERENCE CLASS-STRING.
       01 lvItems OBJECT REFERENCE COLL-LISTVIEWITEMS.
       01 lvItem OBJECT REFERENCE CLASS-LISTVIEWITEM.
       01 lvSubItems OBJECT REFERENCE COLL-LISTVIEWSUBITEMS.
       LINKAGE SECTION.
       01 registDate OBJECT REFERENCE CLASS-DATETIME.
       01 registText OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION USING BY VALUE registDate registText.
      *  日付を文字列にフォーマットします。
           SET strDate TO registDate :: "ToString" (N"yyyy/MM/dd").

      *  リストビューのアイテムを作成します。
           SET lvItem TO CLASS-LISTVIEWITEM :: "NEW" (strDate).
           SET lvSubItems TO PROP-SUBITEMS OF lvItem.
           INVOKE lvSubItems "Add" USING registText.

      *  リストビューに登録します。
           SET lvItems TO PROP-ITEMS OF listView1.
           INVOKE lvItems "Add" USING lvItem.

       END METHOD Registration.

       METHOD-ID. MainForm_Load PRIVATE.
```

```
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           INVOKE SELF "Clear".
       END METHOD MainForm_Load.

       METHOD-ID. toolStripMenuItem3_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  このフォームを閉じます
           INVOKE SELF "Close".
       END METHOD toolStripMenuItem3_Click.

       METHOD-ID. toolStripMenuItem5_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 eventDisposed OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 eventRegistration OBJECT REFERENCE DELEGATE-REGISTRATION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  [新規に登録ウィンドウを開く]
      *  登録フォームを開きます。
           IF registrationForm1 = NULL
               INVOKE FORM-REGISTRATION "NEW" RETURNING registrationForm1

      *  登録フォームの Disposed イベントを設定します。
               INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                        BY VALUE N"Registration_Disposed" RETURNING eventDisposed
               INVOKE registrationForm1 "add_Disposed" USING BY VALUE eventDisposed

      *  登録フォームの Registration イベントを登録します。
               INVOKE DELEGATE-REGISTRATION "NEW" USING BY VALUE SELF
                          BY VALUE N"Registration" RETURNING eventRegistration
               INVOKE registrationForm1 "SetRegistrationEvent" USING BY VALUE eventRegistration

      *  登録フォームをモードレスで開きます。
               INVOKE registrationForm1 "Show"
           ELSE
               INVOKE registrationForm1 "Activate"
           END-IF.

      *  登録フォームをモードレスで開きます。
           INVOKE registrationForm1 "Clear".

       END METHOD toolStripMenuItem5_Click.

       METHOD-ID. toolStripMenuItem6_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 lvSelItems OBJECT REFERENCE COLL-SELECTEDLISTVIEWITEMS.
       01 lvItem OBJECT REFERENCE CLASS-LISTVIEWITEM.
       01 lvSubItems OBJECT REFERENCE COLL-LISTVIEWSUBITEMS.
       01 lvSubItem OBJECT REFERENCE CLASS-LISTVIEWSUBITEM.
       01 tmpCount BINARY-LONG.
       01 tmpText OBJECT REFERENCE CLASS-STRING.
       01 tmpDate OBJECT REFERENCE CLASS-DATETIME.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  [選択された記事をコピーして開く]
      *  まず登録フォームを開くため toolStripMenuItem5_Click を呼び出します。
           INVOKE toolStripMenuItem5 "PerformClick"

      *  リストビューから選択されたアイテムを取り出します。
           SET lvSelItems TO PROP-SELECTEDITEMS OF listView1.
           MOVE PROP-COUNT OF lvSelItems TO tmpCount.
           IF tmpCount > 0
               SET lvItem TO lvSelItems :: "get_Item" (0)
               SET lvSubItems TO PROP-SUBITEMS OF lvItem
```

```
            SET lvSubItem TO lvSubItems :: "get_Item" (0)
            SET tmpText TO PROP-TEXT OF lvSubItem

            INVOKE CLASS-DATETIME "Parse" USING tmpText RETURNING tmpDate

            SET lvSubItem TO lvSubItems :: "get_Item" (1)
            SET tmpText TO PROP-TEXT OF lvSubItem

      *  登録フォームに設定します
            INVOKE registrationForm1 "SetData" USING tmpDate tmpText

        END-IF.

     END METHOD toolStripMenuItem6_Click.

     METHOD-ID. toolStripMenuItem7_Click PRIVATE.
     DATA DIVISION.
     WORKING-STORAGE SECTION.
     01 lvSelItems OBJECT REFERENCE COLL-SELECTEDLISTVIEWITEMS.
     01 lvItem OBJECT REFERENCE CLASS-LISTVIEWITEM.
     01 lvItems OBJECT REFERENCE COLL-LISTVIEWITEMS.
     01 tmpCount BINARY-LONG.
     LINKAGE SECTION.
     01 sender OBJECT REFERENCE CLASS-OBJECT.
     01 e OBJECT REFERENCE CLASS-EVENTARGS.
     PROCEDURE DIVISION USING BY VALUE sender e.
    *  [選択された記事を削除]
    *  リストビューから選択されたアイテムを取り出します。
        SET lvSelItems TO PROP-SELECTEDITEMS OF listView1.
        MOVE PROP-COUNT OF lvSelItems TO tmpCount.
        IF tmpCount > 0
            SET lvItem TO lvSelItems :: "get_Item" (0)
    *  アイテムをリストビューから削除します。
            SET lvItems TO PROP-ITEMS OF listView1
            INVOKE lvItems "Remove" USING lvItem
        END-IF.
     END METHOD toolStripMenuItem7_Click.

     END OBJECT.
     END CLASS CLASS-THIS.
```

NCOBU00935

ソースコード：Registration.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "Schedule.RegistrationForm"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS FORM-CALENDAR AS "Schedule.CalendarForm"
   DELEGATE DELEGATE-REGISTRATION AS "Schedule.Registration"
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   INTERFACE INTERFACE-ISUPPORTINITIALIZE AS "System.ComponentModel.ISupportInitialize"
   CLASS CLASS-DATETIME AS "System.DateTime"
   CLASS CLASS-DECIMAL AS "System.Decimal"
   ENUM ENUM-CONTENTALIGNMENT AS "System.Drawing.ContentAlignment"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS ARRAY-INT32 AS "System.Int32[]"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   ENUM ENUM-ANCHORSTYLES AS "System.Windows.Forms.AnchorStyles"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   ENUM ENUM-DIALOGRESULT AS "System.Windows.Forms.DialogResult"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-NUMERICUPDOWN AS "System.Windows.Forms.NumericUpDown"
   ENUM ENUM-SCROLLBARS AS "System.Windows.Forms.ScrollBars"
   CLASS CLASS-TEXTBOX AS "System.Windows.Forms.TextBox"
   PROPERTY PROP-ANCHOR AS "Anchor"
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-BOTTOM AS "Bottom"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-BUTTON2 AS "button2"
   PROPERTY PROP-BUTTON3 AS "button3"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-DAY AS "Day"
   PROPERTY PROP-EMPTY AS "Empty"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-LABEL1 AS "label1"
   PROPERTY PROP-LABEL2 AS "label2"
   PROPERTY PROP-LABEL3 AS "label3"
   PROPERTY PROP-LABEL4 AS "label4"
   PROPERTY PROP-LABEL5 AS "label5"
   PROPERTY PROP-LEFT AS "Left"
   PROPERTY PROP-LOCATION AS "Location"
   PROPERTY PROP-MAXIMUM AS "Maximum"
   PROPERTY PROP-MAXLENGTH AS "MaxLength"
   PROPERTY PROP-MIDDLELEFT AS "MiddleLeft"
   PROPERTY PROP-MINIMUM AS "Minimum"
   PROPERTY PROP-MONTH AS "Month"
   PROPERTY PROP-MULTILINE AS "Multiline"
   PROPERTY PROP-NAME AS "Name"
   PROPERTY PROP-NUMERICUPDOWN1 AS "numericUpDown1"
   PROPERTY PROP-NUMERICUPDOWN2 AS "numericUpDown2"
   PROPERTY PROP-NUMERICUPDOWN3 AS "numericUpDown3"
   PROPERTY PROP-OK AS "OK"
   PROPERTY PROP-RIGHT AS "Right"
   PROPERTY PROP-SCROLLBARS AS "ScrollBars"
   PROPERTY PROP-SELECTIONDATE AS "SelectionDate"
   PROPERTY PROP-SIZE AS "Size"
   PROPERTY PROP-TABINDEX AS "TabIndex"
```

```
           PROPERTY PROP-TEXT AS "Text"
           PROPERTY PROP-TEXTALIGN AS "TextAlign"
           PROPERTY PROP-TEXTBOX1 AS "textBox1"
           PROPERTY PROP-TODAY AS "Today"
           PROPERTY PROP-TOP AS "Top"
           PROPERTY PROP-USEVISUALSTYLEBACKCOLOR AS "UseVisualStyleBackColor"
           PROPERTY PROP-VALUE AS "Value"
           PROPERTY PROP-VERTICAL AS "Vertical"
           PROPERTY PROP-YEAR AS "Year"
           .

       OBJECT.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 label1 OBJECT REFERENCE CLASS-LABEL.
       01 numericUpDown1 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 label2 OBJECT REFERENCE CLASS-LABEL.
       01 numericUpDown2 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 label3 OBJECT REFERENCE CLASS-LABEL.
       01 numericUpDown3 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 label4 OBJECT REFERENCE CLASS-LABEL.
       01 button1 OBJECT REFERENCE CLASS-BUTTON.
       01 label5 OBJECT REFERENCE CLASS-LABEL.
       01 textBox1 OBJECT REFERENCE CLASS-TEXTBOX.
       01 button2 OBJECT REFERENCE CLASS-BUTTON.
       01 button3 OBJECT REFERENCE CLASS-BUTTON.
       01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
       01 registrationEvent OBJECT REFERENCE DELEGATE-REGISTRATION.
       PROCEDURE DIVISION.

       METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING BY VALUE disposing.
           IF disposing NOT = B"0" AND components NOT = NULL THEN
             INVOKE components "Dispose"
           END-IF.
           INVOKE SUPER "Dispose" USING disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP2 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP3 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP4 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP5 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP6 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP7 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP8 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP9 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP10 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP11 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP12 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP13 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP14 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       01 TEMP15 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP16 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       01 TEMP17 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP18 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       01 TEMP19 BINARY-LONG.
       01 TEMP20 BINARY-LONG.
       01 TEMP21 OBJECT REFERENCE CLASS-POINT.
       01 TEMP22 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP23 OBJECT REFERENCE CLASS-STRING.
       01 TEMP24 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP25 BINARY-LONG.
       01 TEMP26 BINARY-LONG.
       01 TEMP27 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP28 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP29 BINARY-LONG.
       01 TEMP30 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP31 OBJECT REFERENCE CLASS-STRING.
       01 TEMP32 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP33 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP34 BINARY-LONG.
       01 TEMP35 BINARY-LONG.
       01 TEMP36 OBJECT REFERENCE CLASS-POINT.
       01 TEMP37 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
```

```
       01 TEMP38 BINARY-LONG.
       01 TEMP39 BINARY-LONG.
       01 TEMP40 BINARY-LONG.
       01 TEMP41 BINARY-LONG.
       01 TEMP42 BINARY-LONG.
       01 TEMP43 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP44 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP45 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP46 BINARY-LONG.
       01 TEMP47 BINARY-LONG.
       01 TEMP48 BINARY-LONG.
       01 TEMP49 BINARY-LONG.
       01 TEMP50 BINARY-LONG.
       01 TEMP51 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP52 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP53 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP54 OBJECT REFERENCE CLASS-STRING.
       01 TEMP55 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP56 BINARY-LONG.
       01 TEMP57 BINARY-LONG.
       01 TEMP58 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP59 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP60 BINARY-LONG.
       01 TEMP61 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP62 BINARY-LONG.
       01 TEMP63 BINARY-LONG.
       01 TEMP64 BINARY-LONG.
       01 TEMP65 BINARY-LONG.
       01 TEMP66 BINARY-LONG.
       01 TEMP67 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP68 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP69 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP70 BINARY-LONG.
       01 TEMP71 BINARY-LONG.
       01 TEMP72 OBJECT REFERENCE CLASS-POINT.
       01 TEMP73 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP74 OBJECT REFERENCE CLASS-STRING.
       01 TEMP75 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP76 BINARY-LONG.
       01 TEMP77 BINARY-LONG.
       01 TEMP78 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP79 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP80 BINARY-LONG.
       01 TEMP81 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP82 OBJECT REFERENCE CLASS-STRING.
       01 TEMP83 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP84 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP85 BINARY-LONG.
       01 TEMP86 BINARY-LONG.
       01 TEMP87 OBJECT REFERENCE CLASS-POINT.
       01 TEMP88 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP89 BINARY-LONG.
       01 TEMP90 BINARY-LONG.
       01 TEMP91 BINARY-LONG.
       01 TEMP92 BINARY-LONG.
       01 TEMP93 BINARY-LONG.
       01 TEMP94 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP95 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP96 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP97 BINARY-LONG.
       01 TEMP98 BINARY-LONG.
       01 TEMP99 BINARY-LONG.
       01 TEMP100 BINARY-LONG.
       01 TEMP101 BINARY-LONG.
       01 TEMP102 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP103 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP104 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP105 OBJECT REFERENCE CLASS-STRING.
       01 TEMP106 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP107 BINARY-LONG.
       01 TEMP108 BINARY-LONG.
       01 TEMP109 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP110 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP111 BINARY-LONG.
       01 TEMP112 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP113 BINARY-LONG.
       01 TEMP114 BINARY-LONG.
       01 TEMP115 BINARY-LONG.
       01 TEMP116 BINARY-LONG.
       01 TEMP117 BINARY-LONG.
       01 TEMP118 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP119 OBJECT REFERENCE CLASS-DECIMAL.
```

```
       01 TEMP120 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP121 BINARY-LONG.
       01 TEMP122 BINARY-LONG.
       01 TEMP123 OBJECT REFERENCE CLASS-POINT.
       01 TEMP124 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP125 OBJECT REFERENCE CLASS-STRING.
       01 TEMP126 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP127 BINARY-LONG.
       01 TEMP128 BINARY-LONG.
       01 TEMP129 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP130 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP131 BINARY-LONG.
       01 TEMP132 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP133 OBJECT REFERENCE CLASS-STRING.
       01 TEMP134 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP135 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP136 BINARY-LONG.
       01 TEMP137 BINARY-LONG.
       01 TEMP138 OBJECT REFERENCE CLASS-POINT.
       01 TEMP139 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP140 BINARY-LONG.
       01 TEMP141 BINARY-LONG.
       01 TEMP142 BINARY-LONG.
       01 TEMP143 BINARY-LONG.
       01 TEMP144 BINARY-LONG.
       01 TEMP145 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP146 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP147 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP148 BINARY-LONG.
       01 TEMP149 BINARY-LONG.
       01 TEMP150 BINARY-LONG.
       01 TEMP151 BINARY-LONG.
       01 TEMP152 BINARY-LONG.
       01 TEMP153 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP154 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP155 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP156 OBJECT REFERENCE CLASS-STRING.
       01 TEMP157 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP158 BINARY-LONG.
       01 TEMP159 BINARY-LONG.
       01 TEMP160 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP161 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP162 BINARY-LONG.
       01 TEMP163 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP164 BINARY-LONG.
       01 TEMP165 BINARY-LONG.
       01 TEMP166 BINARY-LONG.
       01 TEMP167 BINARY-LONG.
       01 TEMP168 BINARY-LONG.
       01 TEMP169 OBJECT REFERENCE ARRAY-INT32.
       01 TEMP170 OBJECT REFERENCE CLASS-DECIMAL.
       01 TEMP171 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP172 BINARY-LONG.
       01 TEMP173 BINARY-LONG.
       01 TEMP174 OBJECT REFERENCE CLASS-POINT.
       01 TEMP175 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP176 OBJECT REFERENCE CLASS-STRING.
       01 TEMP177 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP178 BINARY-LONG.
       01 TEMP179 BINARY-LONG.
       01 TEMP180 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP181 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP182 BINARY-LONG.
       01 TEMP183 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP184 OBJECT REFERENCE CLASS-STRING.
       01 TEMP185 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP186 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP187 BINARY-LONG.
       01 TEMP188 BINARY-LONG.
       01 TEMP189 OBJECT REFERENCE CLASS-POINT.
       01 TEMP190 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP191 OBJECT REFERENCE CLASS-STRING.
       01 TEMP192 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP193 BINARY-LONG.
       01 TEMP194 BINARY-LONG.
       01 TEMP195 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP196 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP197 BINARY-LONG.
       01 TEMP198 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP199 OBJECT REFERENCE CLASS-STRING.
       01 TEMP200 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP201 OBJECT REFERENCE CLASS-BUTTON.
```

```
       01 TEMP202 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP203 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP204 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP205 BINARY-LONG.
       01 TEMP206 BINARY-LONG.
       01 TEMP207 OBJECT REFERENCE CLASS-POINT.
       01 TEMP208 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP209 OBJECT REFERENCE CLASS-STRING.
       01 TEMP210 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP211 BINARY-LONG.
       01 TEMP212 BINARY-LONG.
       01 TEMP213 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP214 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP215 BINARY-LONG.
       01 TEMP216 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP217 OBJECT REFERENCE CLASS-STRING.
       01 TEMP218 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP219 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP220 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP221 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP222 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP223 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP224 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP225 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP226 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP227 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP228 BINARY-LONG.
       01 TEMP229 BINARY-LONG.
       01 TEMP230 OBJECT REFERENCE CLASS-POINT.
       01 TEMP231 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP232 BINARY-LONG.
       01 TEMP233 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP234 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP235 OBJECT REFERENCE CLASS-STRING.
       01 TEMP236 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP237 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP238 BINARY-LONG.
       01 TEMP239 BINARY-LONG.
       01 TEMP240 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP241 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP242 BINARY-LONG.
       01 TEMP243 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP244 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP245 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP246 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP247 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP248 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP249 BINARY-LONG.
       01 TEMP250 BINARY-LONG.
       01 TEMP251 OBJECT REFERENCE CLASS-POINT.
       01 TEMP252 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP253 OBJECT REFERENCE CLASS-STRING.
       01 TEMP254 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP255 BINARY-LONG.
       01 TEMP256 BINARY-LONG.
       01 TEMP257 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP258 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP259 BINARY-LONG.
       01 TEMP260 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP261 OBJECT REFERENCE CLASS-STRING.
       01 TEMP262 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP263 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP264 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP265 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP266 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP267 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP268 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP269 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP270 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP271 BINARY-LONG.
       01 TEMP272 BINARY-LONG.
       01 TEMP273 OBJECT REFERENCE CLASS-POINT.
       01 TEMP274 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP275 OBJECT REFERENCE CLASS-STRING.
       01 TEMP276 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP277 BINARY-LONG.
       01 TEMP278 BINARY-LONG.
       01 TEMP279 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP280 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP281 BINARY-LONG.
       01 TEMP282 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP283 OBJECT REFERENCE CLASS-STRING.
```

```
       01 TEMP284 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP285 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP286 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP287 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP288 COMP-1.
       01 TEMP289 COMP-1.
       01 TEMP290 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP291 BINARY-LONG.
       01 TEMP292 BINARY-LONG.
       01 TEMP293 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP294 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP295 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP296 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP297 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP298 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP299 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP300 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP301 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP302 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP303 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP304 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP305 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP306 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP307 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP308 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP309 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP310 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP311 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP312 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP313 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP314 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP315 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP316 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP317 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP318 OBJECT REFERENCE CLASS-STRING.
       01 TEMP319 OBJECT REFERENCE CLASS-STRING.
       01 TEMP320 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP321 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP322 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       01 TEMP323 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP324 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       01 TEMP325 OBJECT REFERENCE CLASS-NUMERICUPDOWN.
       01 TEMP326 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
```

```
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.NumericUpDown" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Label" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="numericUpDown2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.NumericUpDown" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.NumericUpDown" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="BeginInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
```

```
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="BeginInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="BeginInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="40" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
```

```
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="0" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="日付：" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TextAlign" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.ContentAlignment" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
```

```
      *<string value="MiddleLeft" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="58" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Maximum" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9998" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
```

```
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Minimum" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1753" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
```

```
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="numericUpDown1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="numericUpDown1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="50" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="19" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="numericUpDown1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="1" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Value" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1753" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="114" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="9" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
      *<int32 value="2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="年" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TextAlign" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.ContentAlignment" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="MiddleLeft" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="numericUpDown2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="numericUpDown2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="140" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="9" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="numericUpDown2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Maximum" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Minimum" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
```

```
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
```

```
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="40" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Value" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="186" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="9" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label3" />
*</prop>
*</object>
*</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="月" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TextAlign" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.ContentAlignment" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="MiddleLeft" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="numericUpDown3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
```

```
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="212" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Maximum" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="31" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Minimum" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
```

```
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="40" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Value" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
```

```
      *<string value="System.Decimal" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Int32" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label4" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="258" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="9" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label4" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label4" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label4" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
```

```
      *<prop name="Value">
      *<string value="日" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TextAlign" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.ContentAlignment" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="MiddleLeft" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="284" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="7" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="..." />
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button1_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
```

```
      *<prop name="Text">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="50" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="35" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="8" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="記事：" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Left" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
```

```
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="14" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="65" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MaxLength" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1024" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Multiline" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
```

```
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ScrollBars" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ScrollBars" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Vertical" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
```

```
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="296" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="130" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="188" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="201" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="56" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
      *<string value="登録" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button2_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
```

```
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="254" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="201" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
```

```
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="56" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="11" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="閉じる" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
```

```
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button3_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="RegistrationForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleDimensions" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.SizeF" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="6" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AutoScaleMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Font" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="322" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="243" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="RegistrationForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="登録" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Load" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="RegistrationForm_Load" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="EndInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="EndInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
```

```
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="numericUpDown3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="EndInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP1
          SET PROP-LABEL1 OF SELF TO TEMP1
          INVOKE CLASS-NUMERICUPDOWN "NEW" RETURNING TEMP2
          SET PROP-NUMERICUPDOWN1 OF SELF TO TEMP2
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP3
          SET PROP-LABEL2 OF SELF TO TEMP3
          INVOKE CLASS-NUMERICUPDOWN "NEW" RETURNING TEMP4
          SET PROP-NUMERICUPDOWN2 OF SELF TO TEMP4
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP5
          SET PROP-LABEL3 OF SELF TO TEMP5
          INVOKE CLASS-NUMERICUPDOWN "NEW" RETURNING TEMP6
          SET PROP-NUMERICUPDOWN3 OF SELF TO TEMP6
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP7
```

```
        SET PROP-LABEL4 OF SELF TO TEMP7
        INVOKE CLASS-BUTTON "NEW" RETURNING TEMP8
        SET PROP-BUTTON1 OF SELF TO TEMP8
        INVOKE CLASS-LABEL "NEW" RETURNING TEMP9
        SET PROP-LABEL5 OF SELF TO TEMP9
        INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP10
        SET PROP-TEXTBOX1 OF SELF TO TEMP10
        INVOKE CLASS-BUTTON "NEW" RETURNING TEMP11
        SET PROP-BUTTON2 OF SELF TO TEMP11
        INVOKE CLASS-BUTTON "NEW" RETURNING TEMP12
        SET PROP-BUTTON3 OF SELF TO TEMP12
        SET TEMP13 TO PROP-NUMERICUPDOWN1 OF SELF
        SET TEMP14 TO TEMP13 AS INTERFACE-ISUPPORTINITIALIZE
        INVOKE TEMP14 "BeginInit"
        SET TEMP15 TO PROP-NUMERICUPDOWN2 OF SELF
        SET TEMP16 TO TEMP15 AS INTERFACE-ISUPPORTINITIALIZE
        INVOKE TEMP16 "BeginInit"
        SET TEMP17 TO PROP-NUMERICUPDOWN3 OF SELF
        SET TEMP18 TO TEMP17 AS INTERFACE-ISUPPORTINITIALIZE
        INVOKE TEMP18 "BeginInit"
        INVOKE SELF "SuspendLayout"
    *
    *label1
    *
        MOVE 12 TO TEMP19
        MOVE 9 TO TEMP20
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP19 BY VALUE TEMP20 RETURNING TEMP21
        SET TEMP22 TO PROP-LABEL1 OF SELF
        SET PROP-LOCATION OF TEMP22 TO TEMP21
        SET TEMP23 TO N"label1"
        SET TEMP24 TO PROP-LABEL1 OF SELF
        SET PROP-NAME OF TEMP24 TO TEMP23
        MOVE 40 TO TEMP25
        MOVE 20 TO TEMP26
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP25 BY VALUE TEMP26 RETURNING TEMP27
        SET TEMP28 TO PROP-LABEL1 OF SELF
        SET PROP-SIZE OF TEMP28 TO TEMP27
        MOVE 0 TO TEMP29
        SET TEMP30 TO PROP-LABEL1 OF SELF
        MOVE TEMP29 TO PROP-TABINDEX OF TEMP30
        SET TEMP31 TO N"日付："
        SET TEMP32 TO PROP-LABEL1 OF SELF
        SET PROP-TEXT OF TEMP32 TO TEMP31
        SET TEMP33 TO PROP-LABEL1 OF SELF
        SET PROP-TEXTALIGN OF TEMP33 TO PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
    *
    *numericUpDown1
    *
        MOVE 58 TO TEMP34
        MOVE 9 TO TEMP35
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP34 BY VALUE TEMP35 RETURNING TEMP36
        SET TEMP37 TO PROP-NUMERICUPDOWN1 OF SELF
        SET PROP-LOCATION OF TEMP37 TO TEMP36
        MOVE 4 TO TEMP42
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP42 RETURNING TEMP43
        MOVE 9998 TO TEMP38
        INVOKE TEMP43 "Set" USING BY VALUE 0 BY VALUE TEMP38
        MOVE 0 TO TEMP39
        INVOKE TEMP43 "Set" USING BY VALUE 1 BY VALUE TEMP39
        MOVE 0 TO TEMP40
        INVOKE TEMP43 "Set" USING BY VALUE 2 BY VALUE TEMP40
        MOVE 0 TO TEMP41
        INVOKE TEMP43 "Set" USING BY VALUE 3 BY VALUE TEMP41
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP43 RETURNING TEMP44
        SET TEMP45 TO PROP-NUMERICUPDOWN1 OF SELF
        SET PROP-MAXIMUM OF TEMP45 TO TEMP44
        MOVE 4 TO TEMP50
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP50 RETURNING TEMP51
        MOVE 1753 TO TEMP46
        INVOKE TEMP51 "Set" USING BY VALUE 0 BY VALUE TEMP46
        MOVE 0 TO TEMP47
        INVOKE TEMP51 "Set" USING BY VALUE 1 BY VALUE TEMP47
        MOVE 0 TO TEMP48
        INVOKE TEMP51 "Set" USING BY VALUE 2 BY VALUE TEMP48
        MOVE 0 TO TEMP49
        INVOKE TEMP51 "Set" USING BY VALUE 3 BY VALUE TEMP49
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP51 RETURNING TEMP52
        SET TEMP53 TO PROP-NUMERICUPDOWN1 OF SELF
        SET PROP-MINIMUM OF TEMP53 TO TEMP52
        SET TEMP54 TO N"numericUpDown1"
        SET TEMP55 TO PROP-NUMERICUPDOWN1 OF SELF
```

```
        SET PROP-NAME OF TEMP55 TO TEMP54
        MOVE 50 TO TEMP56
        MOVE 19 TO TEMP57
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP56 BY VALUE TEMP57 RETURNING TEMP58
        SET TEMP59 TO PROP-NUMERICUPDOWN1 OF SELF
        SET PROP-SIZE OF TEMP59 TO TEMP58
        MOVE 1 TO TEMP60
        SET TEMP61 TO PROP-NUMERICUPDOWN1 OF SELF
        MOVE TEMP60 TO PROP-TABINDEX OF TEMP61
        MOVE 4 TO TEMP66
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP66 RETURNING TEMP67
        MOVE 1753 TO TEMP62
        INVOKE TEMP67 "Set" USING BY VALUE 0 BY VALUE TEMP62
        MOVE 0 TO TEMP63
        INVOKE TEMP67 "Set" USING BY VALUE 1 BY VALUE TEMP63
        MOVE 0 TO TEMP64
        INVOKE TEMP67 "Set" USING BY VALUE 2 BY VALUE TEMP64
        MOVE 0 TO TEMP65
        INVOKE TEMP67 "Set" USING BY VALUE 3 BY VALUE TEMP65
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP67 RETURNING TEMP68
        SET TEMP69 TO PROP-NUMERICUPDOWN1 OF SELF
        SET PROP-VALUE OF TEMP69 TO TEMP68
    *
    *label2
    *
        MOVE 114 TO TEMP70
        MOVE 9 TO TEMP71
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP70 BY VALUE TEMP71 RETURNING TEMP72
        SET TEMP73 TO PROP-LABEL2 OF SELF
        SET PROP-LOCATION OF TEMP73 TO TEMP72
        SET TEMP74 TO N"label2"
        SET TEMP75 TO PROP-LABEL2 OF SELF
        SET PROP-NAME OF TEMP75 TO TEMP74
        MOVE 20 TO TEMP76
        MOVE 20 TO TEMP77
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP76 BY VALUE TEMP77 RETURNING TEMP78
        SET TEMP79 TO PROP-LABEL2 OF SELF
        SET PROP-SIZE OF TEMP79 TO TEMP78
        MOVE 2 TO TEMP80
        SET TEMP81 TO PROP-LABEL2 OF SELF
        MOVE TEMP80 TO PROP-TABINDEX OF TEMP81
        SET TEMP82 TO N"年"
        SET TEMP83 TO PROP-LABEL2 OF SELF
        SET PROP-TEXT OF TEMP83 TO TEMP82
        SET TEMP84 TO PROP-LABEL2 OF SELF
        SET PROP-TEXTALIGN OF TEMP84 TO PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
    *
    *numericUpDown2
    *
        MOVE 140 TO TEMP85
        MOVE 9 TO TEMP86
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP85 BY VALUE TEMP86 RETURNING TEMP87
        SET TEMP88 TO PROP-NUMERICUPDOWN2 OF SELF
        SET PROP-LOCATION OF TEMP88 TO TEMP87
        MOVE 4 TO TEMP93
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP93 RETURNING TEMP94
        MOVE 12 TO TEMP89
        INVOKE TEMP94 "Set" USING BY VALUE 0 BY VALUE TEMP89
        MOVE 0 TO TEMP90
        INVOKE TEMP94 "Set" USING BY VALUE 1 BY VALUE TEMP90
        MOVE 0 TO TEMP91
        INVOKE TEMP94 "Set" USING BY VALUE 2 BY VALUE TEMP91
        MOVE 0 TO TEMP92
        INVOKE TEMP94 "Set" USING BY VALUE 3 BY VALUE TEMP92
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP94 RETURNING TEMP95
        SET TEMP96 TO PROP-NUMERICUPDOWN2 OF SELF
        SET PROP-MAXIMUM OF TEMP96 TO TEMP95
        MOVE 4 TO TEMP101
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP101 RETURNING TEMP102
        MOVE 1 TO TEMP97
        INVOKE TEMP102 "Set" USING BY VALUE 0 BY VALUE TEMP97
        MOVE 0 TO TEMP98
        INVOKE TEMP102 "Set" USING BY VALUE 1 BY VALUE TEMP98
        MOVE 0 TO TEMP99
        INVOKE TEMP102 "Set" USING BY VALUE 2 BY VALUE TEMP99
        MOVE 0 TO TEMP100
        INVOKE TEMP102 "Set" USING BY VALUE 3 BY VALUE TEMP100
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP102 RETURNING TEMP103
        SET TEMP104 TO PROP-NUMERICUPDOWN2 OF SELF
        SET PROP-MINIMUM OF TEMP104 TO TEMP103
        SET TEMP105 TO N"numericUpDown2"
```

```
        SET TEMP106 TO PROP-NUMERICUPDOWN2 OF SELF
        SET PROP-NAME OF TEMP106 TO TEMP105
        MOVE 40 TO TEMP107
        MOVE 19 TO TEMP108
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP107 BY VALUE TEMP108 RETURNING TEMP109
        SET TEMP110 TO PROP-NUMERICUPDOWN2 OF SELF
        SET PROP-SIZE OF TEMP110 TO TEMP109
        MOVE 3 TO TEMP111
        SET TEMP112 TO PROP-NUMERICUPDOWN2 OF SELF
        MOVE TEMP111 TO PROP-TABINDEX OF TEMP112
        MOVE 4 TO TEMP117
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP117 RETURNING TEMP118
        MOVE 1 TO TEMP113
        INVOKE TEMP118 "Set" USING BY VALUE 0 BY VALUE TEMP113
        MOVE 0 TO TEMP114
        INVOKE TEMP118 "Set" USING BY VALUE 1 BY VALUE TEMP114
        MOVE 0 TO TEMP115
        INVOKE TEMP118 "Set" USING BY VALUE 2 BY VALUE TEMP115
        MOVE 0 TO TEMP116
        INVOKE TEMP118 "Set" USING BY VALUE 3 BY VALUE TEMP116
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP118 RETURNING TEMP119
        SET TEMP120 TO PROP-NUMERICUPDOWN2 OF SELF
        SET PROP-VALUE OF TEMP120 TO TEMP119
    *
    *label3
    *
        MOVE 186 TO TEMP121
        MOVE 9 TO TEMP122
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP121 BY VALUE TEMP122 RETURNING TEMP123
        SET TEMP124 TO PROP-LABEL3 OF SELF
        SET PROP-LOCATION OF TEMP124 TO TEMP123
        SET TEMP125 TO N"label3"
        SET TEMP126 TO PROP-LABEL3 OF SELF
        SET PROP-NAME OF TEMP126 TO TEMP125
        MOVE 20 TO TEMP127
        MOVE 20 TO TEMP128
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP127 BY VALUE TEMP128 RETURNING TEMP129
        SET TEMP130 TO PROP-LABEL3 OF SELF
        SET PROP-SIZE OF TEMP130 TO TEMP129
        MOVE 4 TO TEMP131
        SET TEMP132 TO PROP-LABEL3 OF SELF
        MOVE TEMP131 TO PROP-TABINDEX OF TEMP132
        SET TEMP133 TO N"月"
        SET TEMP134 TO PROP-LABEL3 OF SELF
        SET PROP-TEXT OF TEMP134 TO TEMP133
        SET TEMP135 TO PROP-LABEL3 OF SELF
        SET PROP-TEXTALIGN OF TEMP135 TO PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
    *
    *numericUpDown3
    *
        MOVE 212 TO TEMP136
        MOVE 9 TO TEMP137
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP136 BY VALUE TEMP137 RETURNING TEMP138
        SET TEMP139 TO PROP-NUMERICUPDOWN3 OF SELF
        SET PROP-LOCATION OF TEMP139 TO TEMP138
        MOVE 4 TO TEMP144
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP144 RETURNING TEMP145
        MOVE 31 TO TEMP140
        INVOKE TEMP145 "Set" USING BY VALUE 0 BY VALUE TEMP140
        MOVE 0 TO TEMP141
        INVOKE TEMP145 "Set" USING BY VALUE 1 BY VALUE TEMP141
        MOVE 0 TO TEMP142
        INVOKE TEMP145 "Set" USING BY VALUE 2 BY VALUE TEMP142
        MOVE 0 TO TEMP143
        INVOKE TEMP145 "Set" USING BY VALUE 3 BY VALUE TEMP143
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP145 RETURNING TEMP146
        SET TEMP147 TO PROP-NUMERICUPDOWN3 OF SELF
        SET PROP-MAXIMUM OF TEMP147 TO TEMP146
        MOVE 4 TO TEMP152
        INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP152 RETURNING TEMP153
        MOVE 1 TO TEMP148
        INVOKE TEMP153 "Set" USING BY VALUE 0 BY VALUE TEMP148
        MOVE 0 TO TEMP149
        INVOKE TEMP153 "Set" USING BY VALUE 1 BY VALUE TEMP149
        MOVE 0 TO TEMP150
        INVOKE TEMP153 "Set" USING BY VALUE 2 BY VALUE TEMP150
        MOVE 0 TO TEMP151
        INVOKE TEMP153 "Set" USING BY VALUE 3 BY VALUE TEMP151
        INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP153 RETURNING TEMP154
        SET TEMP155 TO PROP-NUMERICUPDOWN3 OF SELF
        SET PROP-MINIMUM OF TEMP155 TO TEMP154
```

```
          SET TEMP156 TO N"numericUpDown3"
          SET TEMP157 TO PROP-NUMERICUPDOWN3 OF SELF
          SET PROP-NAME OF TEMP157 TO TEMP156
          MOVE 40 TO TEMP158
          MOVE 19 TO TEMP159
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP158 BY VALUE TEMP159 RETURNING TEMP160
          SET TEMP161 TO PROP-NUMERICUPDOWN3 OF SELF
          SET PROP-SIZE OF TEMP161 TO TEMP160
          MOVE 5 TO TEMP162
          SET TEMP163 TO PROP-NUMERICUPDOWN3 OF SELF
          MOVE TEMP162 TO PROP-TABINDEX OF TEMP163
          MOVE 4 TO TEMP168
          INVOKE ARRAY-INT32 "NEW" USING BY VALUE TEMP168 RETURNING TEMP169
          MOVE 1 TO TEMP164
          INVOKE TEMP169 "Set" USING BY VALUE 0 BY VALUE TEMP164
          MOVE 0 TO TEMP165
          INVOKE TEMP169 "Set" USING BY VALUE 1 BY VALUE TEMP165
          MOVE 0 TO TEMP166
          INVOKE TEMP169 "Set" USING BY VALUE 2 BY VALUE TEMP166
          MOVE 0 TO TEMP167
          INVOKE TEMP169 "Set" USING BY VALUE 3 BY VALUE TEMP167
          INVOKE CLASS-DECIMAL "NEW" USING BY VALUE TEMP169 RETURNING TEMP170
          SET TEMP171 TO PROP-NUMERICUPDOWN3 OF SELF
          SET PROP-VALUE OF TEMP171 TO TEMP170
      *
      *label4
      *
          MOVE 258 TO TEMP172
          MOVE 9 TO TEMP173
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP172 BY VALUE TEMP173 RETURNING TEMP174
          SET TEMP175 TO PROP-LABEL4 OF SELF
          SET PROP-LOCATION OF TEMP175 TO TEMP174
          SET TEMP176 TO N"label4"
          SET TEMP177 TO PROP-LABEL4 OF SELF
          SET PROP-NAME OF TEMP177 TO TEMP176
          MOVE 20 TO TEMP178
          MOVE 20 TO TEMP179
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP178 BY VALUE TEMP179 RETURNING TEMP180
          SET TEMP181 TO PROP-LABEL4 OF SELF
          SET PROP-SIZE OF TEMP181 TO TEMP180
          MOVE 6 TO TEMP182
          SET TEMP183 TO PROP-LABEL4 OF SELF
          MOVE TEMP182 TO PROP-TABINDEX OF TEMP183
          SET TEMP184 TO N"日"
          SET TEMP185 TO PROP-LABEL4 OF SELF
          SET PROP-TEXT OF TEMP185 TO TEMP184
          SET TEMP186 TO PROP-LABEL4 OF SELF
          SET PROP-TEXTALIGN OF TEMP186 TO PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
      *
      *button1
      *
          MOVE 284 TO TEMP187
          MOVE 6 TO TEMP188
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP187 BY VALUE TEMP188 RETURNING TEMP189
          SET TEMP190 TO PROP-BUTTON1 OF SELF
          SET PROP-LOCATION OF TEMP190 TO TEMP189
          SET TEMP191 TO N"button1"
          SET TEMP192 TO PROP-BUTTON1 OF SELF
          SET PROP-NAME OF TEMP192 TO TEMP191
          MOVE 20 TO TEMP193
          MOVE 23 TO TEMP194
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP193 BY VALUE TEMP194 RETURNING TEMP195
          SET TEMP196 TO PROP-BUTTON1 OF SELF
          SET PROP-SIZE OF TEMP196 TO TEMP195
          MOVE 7 TO TEMP197
          SET TEMP198 TO PROP-BUTTON1 OF SELF
          MOVE TEMP197 TO PROP-TABINDEX OF TEMP198
          SET TEMP199 TO N"..."
          SET TEMP200 TO PROP-BUTTON1 OF SELF
          SET PROP-TEXT OF TEMP200 TO TEMP199
          SET TEMP201 TO PROP-BUTTON1 OF SELF
          SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP201 TO B"1"
          SET TEMP202 TO PROP-BUTTON1 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click" RETURNING
TEMP203
          INVOKE TEMP202 "add_Click" USING BY VALUE TEMP203
      *
      *label5
      *
          SET TEMP204 TO PROP-LABEL5 OF SELF
          SET PROP-AUTOSIZE OF TEMP204 TO B"1"
```

```
        MOVE 12 TO TEMP205
        MOVE 50 TO TEMP206
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP205 BY VALUE TEMP206 RETURNING TEMP207
        SET TEMP208 TO PROP-LABEL5 OF SELF
        SET PROP-LOCATION OF TEMP208 TO TEMP207
        SET TEMP209 TO N"label5"
        SET TEMP210 TO PROP-LABEL5 OF SELF
        SET PROP-NAME OF TEMP210 TO TEMP209
        MOVE 35 TO TEMP211
        MOVE 12 TO TEMP212
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP211 BY VALUE TEMP212 RETURNING TEMP213
        SET TEMP214 TO PROP-LABEL5 OF SELF
        SET PROP-SIZE OF TEMP214 TO TEMP213
        MOVE 8 TO TEMP215
        SET TEMP216 TO PROP-LABEL5 OF SELF
        MOVE TEMP215 TO PROP-TABINDEX OF TEMP216
        SET TEMP217 TO N"記事："
        SET TEMP218 TO PROP-LABEL5 OF SELF
        SET PROP-TEXT OF TEMP218 TO TEMP217
    *
    *textBox1
    *
        SET TEMP219 TO PROP-TOP OF ENUM-ANCHORSTYLES
        SET TEMP220 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
        SET TEMP221 TO FUNCTION ENUM-OR(TEMP219 TEMP220)
        SET TEMP222 TO PROP-LEFT OF ENUM-ANCHORSTYLES
        SET TEMP223 TO FUNCTION ENUM-OR(TEMP221 TEMP222)
        SET TEMP224 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP225 TO FUNCTION ENUM-OR(TEMP223 TEMP224)
        SET TEMP226 TO TEMP225
        SET TEMP227 TO PROP-TEXTBOX1 OF SELF
        SET PROP-ANCHOR OF TEMP227 TO TEMP226
        MOVE 14 TO TEMP228
        MOVE 65 TO TEMP229
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP228 BY VALUE TEMP229 RETURNING TEMP230
        SET TEMP231 TO PROP-TEXTBOX1 OF SELF
        SET PROP-LOCATION OF TEMP231 TO TEMP230
        MOVE 1024 TO TEMP232
        SET TEMP233 TO PROP-TEXTBOX1 OF SELF
        MOVE TEMP232 TO PROP-MAXLENGTH OF TEMP233
        SET TEMP234 TO PROP-TEXTBOX1 OF SELF
        SET PROP-MULTILINE OF TEMP234 TO B"1"
        SET TEMP235 TO N"textBox1"
        SET TEMP236 TO PROP-TEXTBOX1 OF SELF
        SET PROP-NAME OF TEMP236 TO TEMP235
        SET TEMP237 TO PROP-TEXTBOX1 OF SELF
        SET PROP-SCROLLBARS OF TEMP237 TO PROP-VERTICAL OF ENUM-SCROLLBARS
        MOVE 296 TO TEMP238
        MOVE 130 TO TEMP239
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP238 BY VALUE TEMP239 RETURNING TEMP240
        SET TEMP241 TO PROP-TEXTBOX1 OF SELF
        SET PROP-SIZE OF TEMP241 TO TEMP240
        MOVE 9 TO TEMP242
        SET TEMP243 TO PROP-TEXTBOX1 OF SELF
        MOVE TEMP242 TO PROP-TABINDEX OF TEMP243
    *
    *button2
    *
        SET TEMP244 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
        SET TEMP245 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP246 TO FUNCTION ENUM-OR(TEMP244 TEMP245)
        SET TEMP247 TO TEMP246
        SET TEMP248 TO PROP-BUTTON2 OF SELF
        SET PROP-ANCHOR OF TEMP248 TO TEMP247
        MOVE 188 TO TEMP249
        MOVE 201 TO TEMP250
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP249 BY VALUE TEMP250 RETURNING TEMP251
        SET TEMP252 TO PROP-BUTTON2 OF SELF
        SET PROP-LOCATION OF TEMP252 TO TEMP251
        SET TEMP253 TO N"button2"
        SET TEMP254 TO PROP-BUTTON2 OF SELF
        SET PROP-NAME OF TEMP254 TO TEMP253
        MOVE 56 TO TEMP255
        MOVE 30 TO TEMP256
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP255 BY VALUE TEMP256 RETURNING TEMP257
        SET TEMP258 TO PROP-BUTTON2 OF SELF
        SET PROP-SIZE OF TEMP258 TO TEMP257
        MOVE 10 TO TEMP259
        SET TEMP260 TO PROP-BUTTON2 OF SELF
        MOVE TEMP259 TO PROP-TABINDEX OF TEMP260
        SET TEMP261 TO N"登録"
```

```
        SET TEMP262 TO PROP-BUTTON2 OF SELF
        SET PROP-TEXT OF TEMP262 TO TEMP261
        SET TEMP263 TO PROP-BUTTON2 OF SELF
        SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP263 TO B"1"
        SET TEMP264 TO PROP-BUTTON2 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button2_Click"
               RETURNING TEMP265
        INVOKE TEMP264 "add_Click" USING BY VALUE TEMP265
    *
    *button3
    *
        SET TEMP266 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
        SET TEMP267 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP268 TO FUNCTION ENUM-OR(TEMP266 TEMP267)
        SET TEMP269 TO TEMP268
        SET TEMP270 TO PROP-BUTTON3 OF SELF
        SET PROP-ANCHOR OF TEMP270 TO TEMP269
        MOVE 254 TO TEMP271
        MOVE 201 TO TEMP272
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP271 BY VALUE TEMP272 RETURNING TEMP273
        SET TEMP274 TO PROP-BUTTON3 OF SELF
        SET PROP-LOCATION OF TEMP274 TO TEMP273
        SET TEMP275 TO N"button3"
        SET TEMP276 TO PROP-BUTTON3 OF SELF
        SET PROP-NAME OF TEMP276 TO TEMP275
        MOVE 56 TO TEMP277
        MOVE 30 TO TEMP278
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP277 BY VALUE TEMP278 RETURNING TEMP279
        SET TEMP280 TO PROP-BUTTON3 OF SELF
        SET PROP-SIZE OF TEMP280 TO TEMP279
        MOVE 11 TO TEMP281
        SET TEMP282 TO PROP-BUTTON3 OF SELF
        MOVE TEMP281 TO PROP-TABINDEX OF TEMP282
        SET TEMP283 TO N"閉じる"
        SET TEMP284 TO PROP-BUTTON3 OF SELF
        SET PROP-TEXT OF TEMP284 TO TEMP283
        SET TEMP285 TO PROP-BUTTON3 OF SELF
        SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP285 TO B"1"
        SET TEMP286 TO PROP-BUTTON3 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button3_Click"
               RETURNING TEMP287
        INVOKE TEMP286 "add_Click" USING BY VALUE TEMP287
    *
    *RegistrationForm
    *
        MOVE 6.00000000000000E+00 TO TEMP288
        MOVE 1.20000000000000E+01 TO TEMP289
        INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP288 BY VALUE TEMP289 RETURNING TEMP290
        SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP290
        SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
        MOVE 322 TO TEMP291
        MOVE 243 TO TEMP292
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP291 BY VALUE TEMP292 RETURNING TEMP293
        SET PROP-CLIENTSIZE OF SELF TO TEMP293
        SET TEMP295 TO PROP-BUTTON3 OF SELF
        SET TEMP294 TO PROP-CONTROLS OF SELF
        INVOKE TEMP294 "Add" USING BY VALUE TEMP295
        SET TEMP297 TO PROP-BUTTON2 OF SELF
        SET TEMP296 TO PROP-CONTROLS OF SELF
        INVOKE TEMP296 "Add" USING BY VALUE TEMP297
        SET TEMP299 TO PROP-TEXTBOX1 OF SELF
        SET TEMP298 TO PROP-CONTROLS OF SELF
        INVOKE TEMP298 "Add" USING BY VALUE TEMP299
        SET TEMP301 TO PROP-LABEL5 OF SELF
        SET TEMP300 TO PROP-CONTROLS OF SELF
        INVOKE TEMP300 "Add" USING BY VALUE TEMP301
        SET TEMP303 TO PROP-BUTTON1 OF SELF
        SET TEMP302 TO PROP-CONTROLS OF SELF
        INVOKE TEMP302 "Add" USING BY VALUE TEMP303
        SET TEMP305 TO PROP-LABEL4 OF SELF
        SET TEMP304 TO PROP-CONTROLS OF SELF
        INVOKE TEMP304 "Add" USING BY VALUE TEMP305
        SET TEMP307 TO PROP-NUMERICUPDOWN3 OF SELF
        SET TEMP306 TO PROP-CONTROLS OF SELF
        INVOKE TEMP306 "Add" USING BY VALUE TEMP307
        SET TEMP309 TO PROP-LABEL3 OF SELF
        SET TEMP308 TO PROP-CONTROLS OF SELF
        INVOKE TEMP308 "Add" USING BY VALUE TEMP309
        SET TEMP311 TO PROP-NUMERICUPDOWN2 OF SELF
        SET TEMP310 TO PROP-CONTROLS OF SELF
        INVOKE TEMP310 "Add" USING BY VALUE TEMP311
```

```
          SET TEMP313 TO PROP-LABEL2 OF SELF
          SET TEMP312 TO PROP-CONTROLS OF SELF
          INVOKE TEMP312 "Add" USING BY VALUE TEMP313
          SET TEMP315 TO PROP-NUMERICUPDOWN1 OF SELF
          SET TEMP314 TO PROP-CONTROLS OF SELF
          INVOKE TEMP314 "Add" USING BY VALUE TEMP315
          SET TEMP317 TO PROP-LABEL1 OF SELF
          SET TEMP316 TO PROP-CONTROLS OF SELF
          INVOKE TEMP316 "Add" USING BY VALUE TEMP317
          SET TEMP318 TO N"RegistrationForm"
          SET PROP-NAME OF SELF TO TEMP318
          SET TEMP319 TO N"登録"
          SET PROP-TEXT OF SELF TO TEMP319
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"RegistrationForm_Load"
                 RETURNING TEMP320
          INVOKE SELF "add_Load" USING BY VALUE TEMP320
          SET TEMP321 TO PROP-NUMERICUPDOWN1 OF SELF
          SET TEMP322 TO TEMP321 AS INTERFACE-ISUPPORTINITIALIZE
          INVOKE TEMP322 "EndInit"
          SET TEMP323 TO PROP-NUMERICUPDOWN2 OF SELF
          SET TEMP324 TO TEMP323 AS INTERFACE-ISUPPORTINITIALIZE
          INVOKE TEMP324 "EndInit"
          SET TEMP325 TO PROP-NUMERICUPDOWN3 OF SELF
          SET TEMP326 TO TEMP325 AS INTERFACE-ISUPPORTINITIALIZE
          INVOKE TEMP326 "EndInit"
          INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
          INVOKE SELF "PerformLayout"
      END METHOD INITIALIZECOMPONENT.

      METHOD-ID. NEW.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      PROCEDURE DIVISION.
          INVOKE SELF "InitializeComponent".
      END METHOD NEW.

      METHOD-ID. Clear.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 today OBJECT REFERENCE CLASS-DATETIME.
      PROCEDURE DIVISION.
      *  初期設定
          SET today TO PROP-TODAY OF CLASS-DATETIME.
          SET PROP-VALUE OF numericUpDown1 TO PROP-YEAR OF today.
          SET PROP-VALUE OF numericUpDown2 TO PROP-MONTH OF today.
          SET PROP-VALUE OF numericUpDown3 TO PROP-DAY OF today.
          SET PROP-TEXT OF textBox1 TO PROP-EMPTY OF CLASS-STRING.
      END METHOD Clear.

      METHOD-ID. SetData.
      DATA DIVISION.
      LINKAGE SECTION.
      01 registDate OBJECT REFERENCE CLASS-DATETIME.
      01 registText OBJECT REFERENCE CLASS-STRING.
      PROCEDURE DIVISION USING BY VALUE registDate registText.
      *  初期設定
          SET PROP-VALUE OF numericUpDown1 TO PROP-YEAR OF registDate.
          SET PROP-VALUE OF numericUpDown2 TO PROP-MONTH OF registDate.
          SET PROP-VALUE OF numericUpDown3 TO PROP-DAY OF registDate.
          SET PROP-TEXT OF textBox1 TO registText.
      END METHOD SetData.

      METHOD-ID. SetRegistrationEvent.
      DATA DIVISION.
      LINKAGE SECTION.
      01 event1 OBJECT REFERENCE DELEGATE-REGISTRATION.
      PROCEDURE DIVISION USING BY VALUE event1.
      *  Registration デリゲートを登録します。
          SET registrationEvent TO event1.
      END METHOD SetRegistrationEvent.

      METHOD-ID. RegistrationForm_Load PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.
      *  初期設定
          INVOKE SELF "Clear".
      END METHOD RegistrationForm_Load.
```

```
METHOD-ID. button1_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 currentDate OBJECT REFERENCE CLASS-DATETIME.
01 calendarForm OBJECT REFERENCE FORM-CALENDAR.
01 result OBJECT REFERENCE ENUM-DIALOGRESULT.
01 workYear BINARY-LONG.
01 workMonth BINARY-LONG.
01 workDay BINARY-LONG.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
* カレンダーフォームから日付を選択します。
    SET workYear  TO PROP-VALUE OF numericUpDown1.
    SET workMonth TO PROP-VALUE OF numericUpDown2.
    SET workDay   TO PROP-VALUE OF numericUpDown3.
    INVOKE CLASS-DATETIME "NEW" USING workYear workMonth workDay RETURNING currentDate.

    SET calendarForm TO FORM-CALENDAR :: "NEW".
    SET PROP-SELECTIONDATE OF calendarForm TO currentDate
* カレンダーフォームをモーダルで開きます。
    INVOKE calendarForm "ShowDialog" RETURNING result.
    IF result = PROP-OK OF ENUM-DIALOGRESULT
       SET currentDate TO PROP-SELECTIONDATE OF calendarForm
       SET PROP-VALUE OF numericUpDown1 TO PROP-YEAR OF currentDate
       SET PROP-VALUE OF numericUpDown2 TO PROP-MONTH OF currentDate
       SET PROP-VALUE OF numericUpDown3 TO PROP-DAY OF currentDate
    END-IF.

END METHOD button1_Click.

METHOD-ID. button2_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 registDate OBJECT REFERENCE CLASS-DATETIME.
01 registText OBJECT REFERENCE CLASS-STRING.
01 workYear BINARY-LONG.
01 workMonth BINARY-LONG.
01 workDay BINARY-LONG.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
*  Registration デリゲートを実行します。
    IF registrationEvent NOT = NULL
*  「日付」のパラメタを作成します。
       SET workYear  TO PROP-VALUE OF numericUpDown1
       SET workMonth TO PROP-VALUE OF numericUpDown2
       SET workDay   TO PROP-VALUE OF numericUpDown3
       INVOKE CLASS-DATETIME "NEW" USING workYear workMonth workDay RETURNING registDate
*  「内容」のパラメタを作成します。
       SET registText TO PROP-TEXT OF textBox1

*  デリゲートを呼び出します。
       INVOKE registrationEvent "Invoke" USING registDate registText
    END-IF.

END METHOD button2_Click.

METHOD-ID. button3_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
*  このフォームを閉じます
    INVOKE SELF "Close".
END METHOD button3_Click.

END OBJECT.
END CLASS CLASS-THIS.
```

ソースコード：Calendar.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "Schedule.CalendarForm"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-DATETIME AS "System.DateTime"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   ENUM ENUM-DIALOGRESULT AS "System.Windows.Forms.DialogResult"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   ENUM ENUM-FORMBORDERSTYLE AS "System.Windows.Forms.FormBorderStyle"
   ENUM ENUM-FORMSTARTPOSITION AS "System.Windows.Forms.FormStartPosition"
   CLASS CLASS-MONTHCALENDAR AS "System.Windows.Forms.MonthCalendar"
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-BUTTON2 AS "button2"
   PROPERTY PROP-CANCEL AS "Cancel"
   PROPERTY PROP-CANCELBUTTON AS "CancelButton"
   PROPERTY PROP-CENTERPARENT AS "CenterParent"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-DIALOGRESULT AS "DialogResult"
   PROPERTY PROP-FIXEDDIALOG AS "FixedDialog"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-FORMBORDERSTYLE AS "FormBorderStyle"
   PROPERTY PROP-LOCATION AS "Location"
   PROPERTY PROP-MAXIMIZEBOX AS "MaximizeBox"
   PROPERTY PROP-MINIMIZEBOX AS "MinimizeBox"
   PROPERTY PROP-MONTHCALENDAR1 AS "monthCalendar1"
   PROPERTY PROP-NAME AS "Name"
   PROPERTY PROP-OK AS "OK"
   PROPERTY SELECTIONDATE AS "SelectionDate"
   PROPERTY PROP-SELECTIONSTART AS "SelectionStart"
   PROPERTY PROP-SIZE AS "Size"
   PROPERTY PROP-STARTPOSITION AS "StartPosition"
   PROPERTY PROP-TABINDEX AS "TabIndex"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-USEVISUALSTYLEBACKCOLOR AS "UseVisualStyleBackColor"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 monthCalendar1 OBJECT REFERENCE CLASS-MONTHCALENDAR.
01 button1 OBJECT REFERENCE CLASS-BUTTON.
01 button2 OBJECT REFERENCE CLASS-BUTTON.
01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
PROCEDURE DIVISION.

METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
PROCEDURE DIVISION USING BY VALUE disposing.
   IF disposing NOT = B"0" AND components NOT = NULL THEN
```

```
            INVOKE components "Dispose"
           END-IF.
           INVOKE SUPER "Dispose" USING disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-MONTHCALENDAR.
       01 TEMP2 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP3 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP4 BINARY-LONG.
       01 TEMP5 BINARY-LONG.
       01 TEMP6 OBJECT REFERENCE CLASS-POINT.
       01 TEMP7 OBJECT REFERENCE CLASS-MONTHCALENDAR.
       01 TEMP8 OBJECT REFERENCE CLASS-STRING.
       01 TEMP9 OBJECT REFERENCE CLASS-MONTHCALENDAR.
       01 TEMP10 BINARY-LONG.
       01 TEMP11 OBJECT REFERENCE CLASS-MONTHCALENDAR.
       01 TEMP12 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP13 BINARY-LONG.
       01 TEMP14 BINARY-LONG.
       01 TEMP15 OBJECT REFERENCE CLASS-POINT.
       01 TEMP16 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP17 OBJECT REFERENCE CLASS-STRING.
       01 TEMP18 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP19 BINARY-LONG.
       01 TEMP20 BINARY-LONG.
       01 TEMP21 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP22 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP23 BINARY-LONG.
       01 TEMP24 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP25 OBJECT REFERENCE CLASS-STRING.
       01 TEMP26 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP27 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP28 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP29 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP30 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP31 BINARY-LONG.
       01 TEMP32 BINARY-LONG.
       01 TEMP33 OBJECT REFERENCE CLASS-POINT.
       01 TEMP34 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP35 OBJECT REFERENCE CLASS-STRING.
       01 TEMP36 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP37 BINARY-LONG.
       01 TEMP38 BINARY-LONG.
       01 TEMP39 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP40 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP41 BINARY-LONG.
       01 TEMP42 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP43 OBJECT REFERENCE CLASS-STRING.
       01 TEMP44 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP45 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP46 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP47 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP48 COMP-1.
       01 TEMP49 COMP-1.
       01 TEMP50 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP51 BINARY-LONG.
       01 TEMP52 BINARY-LONG.
       01 TEMP53 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP54 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP55 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP56 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP57 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP58 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP59 OBJECT REFERENCE CLASS-MONTHCALENDAR.
       01 TEMP60 OBJECT REFERENCE CLASS-STRING.
       01 TEMP61 OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="monthCalendar1" />
      *</prop>
```

```
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.MonthCalendar" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Button" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Button" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="SuspendLayout" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="monthCalendar1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="monthCalendar1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="18" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="18" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="monthCalendar1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="monthCalendar1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="monthCalendar1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="0" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="DialogResult" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.DialogResult" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="OK" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="18" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="210" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="64" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="OK" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="UseVisualStyleBackColor" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="True" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
```

```
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="button1_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="DialogResult" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
```

```
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.DialogResult" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Cancel" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="88" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="210" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="64" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="キャンセル" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
```

```
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button2_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="CalendarForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleDimensions" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.SizeF" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="6" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
```

```
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AutoScaleMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Font" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="CancelButton" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
```

```
      *<prop name="Value">
      *<int32 value="253" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="245" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="monthCalendar1" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="FormBorderStyle" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.FormBorderStyle" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="FixedDialog" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="MaximizeBox" />
*</prop>
*</object>
*</prop>
```

```
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MinimizeBox" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="CalendarForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="StartPosition" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.FormStartPosition" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="CenterParent" />
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="日付の選択" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-MONTHCALENDAR "NEW" RETURNING TEMP1
          SET PROP-MONTHCALENDAR1 OF SELF TO TEMP1
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP2
          SET PROP-BUTTON1 OF SELF TO TEMP2
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP3
          SET PROP-BUTTON2 OF SELF TO TEMP3
          INVOKE SELF "SuspendLayout"
      *
      *monthCalendar1
      *
          MOVE 18 TO TEMP4
          MOVE 18 TO TEMP5
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP4 BY VALUE TEMP5 RETURNING TEMP6
          SET TEMP7 TO PROP-MONTHCALENDAR1 OF SELF
          SET PROP-LOCATION OF TEMP7 TO TEMP6
          SET TEMP8 TO N"monthCalendar1"
          SET TEMP9 TO PROP-MONTHCALENDAR1 OF SELF
          SET PROP-NAME OF TEMP9 TO TEMP8
          MOVE 0 TO TEMP10
          SET TEMP11 TO PROP-MONTHCALENDAR1 OF SELF
          MOVE TEMP10 TO PROP-TABINDEX OF TEMP11
      *
      *button1
      *
          SET TEMP12 TO PROP-BUTTON1 OF SELF
          SET PROP-DIALOGRESULT OF TEMP12 TO PROP-OK OF ENUM-DIALOGRESULT
          MOVE 18 TO TEMP13
          MOVE 210 TO TEMP14
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP13 BY VALUE TEMP14 RETURNING TEMP15
          SET TEMP16 TO PROP-BUTTON1 OF SELF
          SET PROP-LOCATION OF TEMP16 TO TEMP15
          SET TEMP17 TO N"button1"
          SET TEMP18 TO PROP-BUTTON1 OF SELF
          SET PROP-NAME OF TEMP18 TO TEMP17
          MOVE 64 TO TEMP19
```

```
        MOVE 30 TO TEMP20
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP19 BY VALUE TEMP20 RETURNING TEMP21
        SET TEMP22 TO PROP-BUTTON1 OF SELF
        SET PROP-SIZE OF TEMP22 TO TEMP21
        MOVE 1 TO TEMP23
        SET TEMP24 TO PROP-BUTTON1 OF SELF
        MOVE TEMP23 TO PROP-TABINDEX OF TEMP24
        SET TEMP25 TO N"OK"
        SET TEMP26 TO PROP-BUTTON1 OF SELF
        SET PROP-TEXT OF TEMP26 TO TEMP25
        SET TEMP27 TO PROP-BUTTON1 OF SELF
        SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP27 TO B"1"
        SET TEMP28 TO PROP-BUTTON1 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click"
               RETURNING TEMP29
        INVOKE TEMP28 "add_Click" USING BY VALUE TEMP29
    *
    *button2
    *
        SET TEMP30 TO PROP-BUTTON2 OF SELF
        SET PROP-DIALOGRESULT OF TEMP30 TO PROP-CANCEL OF ENUM-DIALOGRESULT
        MOVE 88 TO TEMP31
        MOVE 210 TO TEMP32
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP31 BY VALUE TEMP32 RETURNING TEMP33
        SET TEMP34 TO PROP-BUTTON2 OF SELF
        SET PROP-LOCATION OF TEMP34 TO TEMP33
        SET TEMP35 TO N"button2"
        SET TEMP36 TO PROP-BUTTON2 OF SELF
        SET PROP-NAME OF TEMP36 TO TEMP35
        MOVE 64 TO TEMP37
        MOVE 30 TO TEMP38
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP37 BY VALUE TEMP38 RETURNING TEMP39
        SET TEMP40 TO PROP-BUTTON2 OF SELF
        SET PROP-SIZE OF TEMP40 TO TEMP39
        MOVE 2 TO TEMP41
        SET TEMP42 TO PROP-BUTTON2 OF SELF
        MOVE TEMP41 TO PROP-TABINDEX OF TEMP42
        SET TEMP43 TO N"キャンセル"
        SET TEMP44 TO PROP-BUTTON2 OF SELF
        SET PROP-TEXT OF TEMP44 TO TEMP43
        SET TEMP45 TO PROP-BUTTON2 OF SELF
        SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP45 TO B"1"
        SET TEMP46 TO PROP-BUTTON2 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button2_Click"
               RETURNING TEMP47
        INVOKE TEMP46 "add_Click" USING BY VALUE TEMP47
    *
    *CalendarForm
    *
        MOVE 6.00000000000000E+00 TO TEMP48
        MOVE 1.20000000000000E+01 TO TEMP49
        INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP48 BY VALUE TEMP49 RETURNING TEMP50
        SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP50
        SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
        SET PROP-CANCELBUTTON OF SELF TO PROP-BUTTON2 OF SELF
        MOVE 253 TO TEMP51
        MOVE 245 TO TEMP52
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP51 BY VALUE TEMP52 RETURNING TEMP53
        SET PROP-CLIENTSIZE OF SELF TO TEMP53
        SET TEMP55 TO PROP-BUTTON2 OF SELF
        SET TEMP54 TO PROP-CONTROLS OF SELF
        INVOKE TEMP54 "Add" USING BY VALUE TEMP55
        SET TEMP57 TO PROP-BUTTON1 OF SELF
        SET TEMP56 TO PROP-CONTROLS OF SELF
        INVOKE TEMP56 "Add" USING BY VALUE TEMP57
        SET TEMP59 TO PROP-MONTHCALENDAR1 OF SELF
        SET TEMP58 TO PROP-CONTROLS OF SELF
        INVOKE TEMP58 "Add" USING BY VALUE TEMP59
        SET PROP-FORMBORDERSTYLE OF SELF TO PROP-FIXEDDIALOG OF ENUM-FORMBORDERSTYLE
        SET PROP-MAXIMIZEBOX OF SELF TO B"0"
        SET PROP-MINIMIZEBOX OF SELF TO B"0"
        SET TEMP60 TO N"CalendarForm"
        SET PROP-NAME OF SELF TO TEMP60
        SET PROP-STARTPOSITION OF SELF TO PROP-CENTERPARENT OF ENUM-FORMSTARTPOSITION
        SET TEMP61 TO N"日付の選択"
        SET PROP-TEXT OF SELF TO TEMP61
        INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
    END METHOD INITIALIZECOMPONENT.

    METHOD-ID. NEW.
    DATA DIVISION.
```

```
      WORKING-STORAGE SECTION.
      PROCEDURE DIVISION.
          INVOKE SELF "InitializeComponent".
      END METHOD NEW.

      METHOD-ID. GET PROPERTY SELECTIONDATE AS "SelectionDate".
      DATA DIVISION.
      LINKAGE SECTION.
      01 RetVal OBJECT REFERENCE CLASS-DATETIME.
      PROCEDURE DIVISION RETURNING RetVal.
     * カレンダーから選択されている日付を取得します
          SET RetVal TO PROP-SELECTIONSTART OF monthCalendar1.
      END METHOD.

      METHOD-ID. SET PROPERTY SELECTIONDATE AS "SelectionDate".
      DATA DIVISION.
      LINKAGE SECTION.
      01 SetVal OBJECT REFERENCE CLASS-DATETIME.
      PROCEDURE DIVISION USING BY VALUE SetVal.
     * カレンダーの日付を選択します
          INVOKE monthCalendar1 "SetDate" USING SetVal.
      END METHOD.

      METHOD-ID. button1_Click PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.
     * OK ボタンでフォームを終了します
          INVOKE SELF "Close".
      END METHOD button1_Click.

      METHOD-ID. button2_Click PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.
     * キャンセルボタンでフォームを終了します
          INVOKE SELF "Close".
      END METHOD button2_Click.

      END OBJECT.
      END CLASS CLASS-THIS.
```

ソースコード：Delegate.cob

```
*  登録ボタンが押されたことを示すデリゲートを定義します。
 IDENTIFICATION  DIVISION.
 DELEGATE-ID. Registration AS "Schedule.Registration".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
    CLASS CLASS-DATETIME AS "System.DateTime"
    CLASS CLASS-STRING AS "System.String"
    .
 DATA DIVISION.
 LINKAGE SECTION.
 01 regDate OBJECT REFERENCE CLASS-DATETIME.
 01 regText OBJECT REFERENCE CLASS-STRING.
 PROCEDURE DIVISION USING BY VALUE regDate regText.

 END DELEGATE Registration.
```

NCOBU00938

ソースコード：Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
   .
REPOSITORY.
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-MAINFORM AS "Schedule.MainForm"
   CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
   .

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-MAINFORM OBJECT REFERENCE CLASS-MAINFORM.
PROCEDURE DIVISION.
   INVOKE CLASS-APPLICATION "EnableVisualStyles".
   INVOKE CLASS-APPLICATION "SetCompatibleTextRenderingDefault" USING BY VALUE B"0".

   INVOKE CLASS-MAINFORM "NEW" RETURNING WK-MAINFORM.
   INVOKE CLASS-APPLICATION "Run" USING BY VALUE WK-MAINFORM.
END PROGRAM MAIN.
```

ソースコード：MainForm.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "CsvToIdx.MainForm"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-INDICATORFORM AS "CsvToIdx.IndicatorForm"
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-PATH AS "System.IO.Path"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-THREAD AS "System.Threading.Thread"
   DELEGATE DELEGATE-THREADSTART AS "System.Threading.ThreadStart"
   ENUM ENUM-ANCHORSTYLES AS "System.Windows.Forms.AnchorStyles"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   ENUM ENUM-DIALOGRESULT AS "System.Windows.Forms.DialogResult"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-MESSAGEBOX AS "System.Windows.Forms.MessageBox"
   ENUM ENUM-MESSAGEBOXBUTTONS AS "System.Windows.Forms.MessageBoxButtons"
   ENUM ENUM-MESSAGEBOXICON AS "System.Windows.Forms.MessageBoxIcon"
   DELEGATE DELEGATE-METHODINVOKER AS "System.Windows.Forms.MethodInvoker"
   CLASS CLASS-OPENFILEDIALOG AS "System.Windows.Forms.OpenFileDialog"
   CLASS CLASS-SAVEFILEDIALOG AS "System.Windows.Forms.SaveFileDialog"
   CLASS CLASS-TEXTBOX AS "System.Windows.Forms.TextBox"
   CLASS CLASS-DIRECTORY AS "System.IO.Directory"
   PROPERTY PROP-ANCHOR AS "Anchor"
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-BUTTON2 AS "button2"
   PROPERTY PROP-BUTTON3 AS "button3"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-EMPTY AS "Empty"
   PROPERTY PROP-EXCLAMATION AS "Exclamation"
   PROPERTY PROP-FILENAME AS "FileName"
   PROPERTY PROP-FILTER AS "Filter"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-ISBACKGROUND AS "IsBackground"
   PROPERTY PROP-LABEL1 AS "label1"
   PROPERTY PROP-LABEL2 AS "label2"
   PROPERTY PROP-LEFT AS "Left"
   PROPERTY PROP-LOCATION AS "Location"
   PROPERTY PROP-NAME AS "Name"
   PROPERTY PROP-OK AS "OK"
   PROPERTY PROP-OPENFILEDIALOG1 AS "openFileDialog1"
   PROPERTY PROP-RIGHT AS "Right"
   PROPERTY PROP-SAVEFILEDIALOG1 AS "saveFileDialog1"
   PROPERTY PROP-SIZE AS "Size"
   PROPERTY PROP-TABINDEX AS "TabIndex"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-TEXTBOX1 AS "textBox1"
   PROPERTY PROP-TEXTBOX2 AS "textBox2"
   PROPERTY PROP-TOP AS "Top"
   PROPERTY PROP-USEVISUALSTYLEBACKCOLOR AS "UseVisualStyleBackColor"
   .

OBJECT.
```

```
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 label1 OBJECT REFERENCE CLASS-LABEL.
       01 textBox1 OBJECT REFERENCE CLASS-TEXTBOX.
       01 button1 OBJECT REFERENCE CLASS-BUTTON.
       01 button2 OBJECT REFERENCE CLASS-BUTTON.
       01 textBox2 OBJECT REFERENCE CLASS-TEXTBOX.
       01 label2 OBJECT REFERENCE CLASS-LABEL.
       01 button3 OBJECT REFERENCE CLASS-BUTTON.
       01 openFileDialog1 OBJECT REFERENCE CLASS-OPENFILEDIALOG.
       01 saveFileDialog1 OBJECT REFERENCE CLASS-SAVEFILEDIALOG.
       01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
       01 indicator OBJECT REFERENCE CLASS-INDICATORFORM.
       01 background OBJECT REFERENCE CLASS-THREAD.
       PROCEDURE DIVISION.

       METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING BY VALUE disposing.
           IF disposing NOT = B"0" AND components NOT = NULL THEN
             INVOKE components "Dispose"
           END-IF.
           INVOKE SUPER "Dispose" USING disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP2 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP3 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP4 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP5 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP6 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP7 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP8 OBJECT REFERENCE CLASS-OPENFILEDIALOG.
       01 TEMP9 OBJECT REFERENCE CLASS-SAVEFILEDIALOG.
       01 TEMP10 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP11 BINARY-LONG.
       01 TEMP12 BINARY-LONG.
       01 TEMP13 OBJECT REFERENCE CLASS-POINT.
       01 TEMP14 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP15 OBJECT REFERENCE CLASS-STRING.
       01 TEMP16 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP17 BINARY-LONG.
       01 TEMP18 BINARY-LONG.
       01 TEMP19 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP20 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP21 BINARY-LONG.
       01 TEMP22 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP23 OBJECT REFERENCE CLASS-STRING.
       01 TEMP24 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP25 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP26 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP27 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP28 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP29 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP30 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP31 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP32 BINARY-LONG.
       01 TEMP33 BINARY-LONG.
       01 TEMP34 OBJECT REFERENCE CLASS-POINT.
       01 TEMP35 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP36 OBJECT REFERENCE CLASS-STRING.
       01 TEMP37 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP38 BINARY-LONG.
       01 TEMP39 BINARY-LONG.
       01 TEMP40 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP41 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP42 BINARY-LONG.
       01 TEMP43 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP44 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP45 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP46 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP47 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP48 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP49 BINARY-LONG.
       01 TEMP50 BINARY-LONG.
       01 TEMP51 OBJECT REFERENCE CLASS-POINT.
```

```
       01 TEMP52 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP53 OBJECT REFERENCE CLASS-STRING.
       01 TEMP54 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP55 BINARY-LONG.
       01 TEMP56 BINARY-LONG.
       01 TEMP57 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP58 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP59 BINARY-LONG.
       01 TEMP60 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP61 OBJECT REFERENCE CLASS-STRING.
       01 TEMP62 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP63 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP64 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP65 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP66 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP67 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP68 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP69 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP70 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP71 BINARY-LONG.
       01 TEMP72 BINARY-LONG.
       01 TEMP73 OBJECT REFERENCE CLASS-POINT.
       01 TEMP74 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP75 OBJECT REFERENCE CLASS-STRING.
       01 TEMP76 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP77 BINARY-LONG.
       01 TEMP78 BINARY-LONG.
       01 TEMP79 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP80 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP81 BINARY-LONG.
       01 TEMP82 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP83 OBJECT REFERENCE CLASS-STRING.
       01 TEMP84 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP85 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP86 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP87 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP88 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP89 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP90 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP91 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP92 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP93 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP94 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP95 BINARY-LONG.
       01 TEMP96 BINARY-LONG.
       01 TEMP97 OBJECT REFERENCE CLASS-POINT.
       01 TEMP98 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP99 OBJECT REFERENCE CLASS-STRING.
       01 TEMP100 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP101 BINARY-LONG.
       01 TEMP102 BINARY-LONG.
       01 TEMP103 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP104 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP105 BINARY-LONG.
       01 TEMP106 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP107 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP108 BINARY-LONG.
       01 TEMP109 BINARY-LONG.
       01 TEMP110 OBJECT REFERENCE CLASS-POINT.
       01 TEMP111 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP112 OBJECT REFERENCE CLASS-STRING.
       01 TEMP113 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP114 BINARY-LONG.
       01 TEMP115 BINARY-LONG.
       01 TEMP116 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP117 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP118 BINARY-LONG.
       01 TEMP119 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP120 OBJECT REFERENCE CLASS-STRING.
       01 TEMP121 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP122 BINARY-LONG.
       01 TEMP123 BINARY-LONG.
       01 TEMP124 OBJECT REFERENCE CLASS-POINT.
       01 TEMP125 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP126 OBJECT REFERENCE CLASS-STRING.
       01 TEMP127 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP128 BINARY-LONG.
       01 TEMP129 BINARY-LONG.
       01 TEMP130 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP131 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP132 BINARY-LONG.
       01 TEMP133 OBJECT REFERENCE CLASS-BUTTON.
```

```
       01 TEMP134 OBJECT REFERENCE CLASS-STRING.
       01 TEMP135 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP136 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP137 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP138 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP139 OBJECT REFERENCE CLASS-STRING.
       01 TEMP140 OBJECT REFERENCE CLASS-OPENFILEDIALOG.
       01 TEMP141 OBJECT REFERENCE CLASS-STRING.
       01 TEMP142 OBJECT REFERENCE CLASS-SAVEFILEDIALOG.
       01 TEMP143 COMP-1.
       01 TEMP144 COMP-1.
       01 TEMP145 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP146 BINARY-LONG.
       01 TEMP147 BINARY-LONG.
       01 TEMP148 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP149 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP150 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP151 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP152 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP153 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP154 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP155 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP156 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP157 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP158 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP159 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP160 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP161 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP162 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP163 OBJECT REFERENCE CLASS-STRING.
       01 TEMP164 OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
```

```
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Button" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Button" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.TextBox" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="openFileDialog1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.OpenFileDialog" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="saveFileDialog1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.SaveFileDialog" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="SuspendLayout" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
```

```
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="96" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="0" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="入力(CSV ファイル):" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
```

```
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Expression">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Right" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Left" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Top" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="14" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="24" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="237" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="19" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="button1" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="Top" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="257" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="22" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
```

```
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="..." />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button1_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="257" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="75" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="..." />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
```

```
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button2_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
```

```
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Left" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="14" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="77" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="237" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
```

```
      *<prop name="Text">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="62" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="97" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="出力(索引ファイル):" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="112" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="121" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="60" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
```

```
*<prop name="Value">
*<string value="変換" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="UseVisualStyleBackColor" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="True" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="button3_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="openFileDialog1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="openFileDialog1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Filter" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="CSV ファイル|*.csv|すべてのファイル|*.*" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="saveFileDialog1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
*</object>
*</prop>
*<prop name="FieldName">
*<string value="saveFileDialog1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="FileName" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="doc1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="MainForm" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="AutoScaleDimensions" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.SizeF" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<float32 value="6" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<float32 value="12" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AutoScaleMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Font" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="292" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="183" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
```

```
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
```

```
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="MainForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="CsvToIdx" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP1
          SET PROP-LABEL1 OF SELF TO TEMP1
          INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP2
          SET PROP-TEXTBOX1 OF SELF TO TEMP2
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP3
```

```
        SET PROP-BUTTON1 OF SELF TO TEMP3
        INVOKE CLASS-BUTTON "NEW" RETURNING TEMP4
        SET PROP-BUTTON2 OF SELF TO TEMP4
        INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP5
        SET PROP-TEXTBOX2 OF SELF TO TEMP5
        INVOKE CLASS-LABEL "NEW" RETURNING TEMP6
        SET PROP-LABEL2 OF SELF TO TEMP6
        INVOKE CLASS-BUTTON "NEW" RETURNING TEMP7
        SET PROP-BUTTON3 OF SELF TO TEMP7
        INVOKE CLASS-OPENFILEDIALOG "NEW" RETURNING TEMP8
        SET PROP-OPENFILEDIALOG1 OF SELF TO TEMP8
        INVOKE CLASS-SAVEFILEDIALOG "NEW" RETURNING TEMP9
        SET PROP-SAVEFILEDIALOG1 OF SELF TO TEMP9
        INVOKE SELF "SuspendLayout"
    *
    *label1
    *
        SET TEMP10 TO PROP-LABEL1 OF SELF
        SET PROP-AUTOSIZE OF TEMP10 TO B"1"
        MOVE 12 TO TEMP11
        MOVE 9 TO TEMP12
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP11 BY VALUE TEMP12 RETURNING TEMP13
        SET TEMP14 TO PROP-LABEL1 OF SELF
        SET PROP-LOCATION OF TEMP14 TO TEMP13
        SET TEMP15 TO N"label1"
        SET TEMP16 TO PROP-LABEL1 OF SELF
        SET PROP-NAME OF TEMP16 TO TEMP15
        MOVE 96 TO TEMP17
        MOVE 12 TO TEMP18
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP17 BY VALUE TEMP18 RETURNING TEMP19
        SET TEMP20 TO PROP-LABEL1 OF SELF
        SET PROP-SIZE OF TEMP20 TO TEMP19
        MOVE 0 TO TEMP21
        SET TEMP22 TO PROP-LABEL1 OF SELF
        MOVE TEMP21 TO PROP-TABINDEX OF TEMP22
        SET TEMP23 TO N"入力(CSV ファイル):"
        SET TEMP24 TO PROP-LABEL1 OF SELF
        SET PROP-TEXT OF TEMP24 TO TEMP23
    *
    *textBox1
    *
        SET TEMP25 TO PROP-TOP OF ENUM-ANCHORSTYLES
        SET TEMP26 TO PROP-LEFT OF ENUM-ANCHORSTYLES
        SET TEMP27 TO FUNCTION ENUM-OR(TEMP25 TEMP26)
        SET TEMP28 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP29 TO FUNCTION ENUM-OR(TEMP27 TEMP28)
        SET TEMP30 TO TEMP29
        SET TEMP31 TO PROP-TEXTBOX1 OF SELF
        SET PROP-ANCHOR OF TEMP31 TO TEMP30
        MOVE 14 TO TEMP32
        MOVE 24 TO TEMP33
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP32 BY VALUE TEMP33 RETURNING TEMP34
        SET TEMP35 TO PROP-TEXTBOX1 OF SELF
        SET PROP-LOCATION OF TEMP35 TO TEMP34
        SET TEMP36 TO N"textBox1"
        SET TEMP37 TO PROP-TEXTBOX1 OF SELF
        SET PROP-NAME OF TEMP37 TO TEMP36
        MOVE 237 TO TEMP38
        MOVE 19 TO TEMP39
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP38 BY VALUE TEMP39 RETURNING TEMP40
        SET TEMP41 TO PROP-TEXTBOX1 OF SELF
        SET PROP-SIZE OF TEMP41 TO TEMP40
        MOVE 1 TO TEMP42
        SET TEMP43 TO PROP-TEXTBOX1 OF SELF
        MOVE TEMP42 TO PROP-TABINDEX OF TEMP43
    *
    *button1
    *
        SET TEMP44 TO PROP-TOP OF ENUM-ANCHORSTYLES
        SET TEMP45 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP46 TO FUNCTION ENUM-OR(TEMP44 TEMP45)
        SET TEMP47 TO TEMP46
        SET TEMP48 TO PROP-BUTTON1 OF SELF
        SET PROP-ANCHOR OF TEMP48 TO TEMP47
        MOVE 257 TO TEMP49
        MOVE 22 TO TEMP50
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP49 BY VALUE TEMP50 RETURNING TEMP51
        SET TEMP52 TO PROP-BUTTON1 OF SELF
        SET PROP-LOCATION OF TEMP52 TO TEMP51
        SET TEMP53 TO N"button1"
        SET TEMP54 TO PROP-BUTTON1 OF SELF
```

```
            SET PROP-NAME OF TEMP54 TO TEMP53
            MOVE 23 TO TEMP55
            MOVE 23 TO TEMP56
            INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP55 BY VALUE TEMP56 RETURNING TEMP57
            SET TEMP58 TO PROP-BUTTON1 OF SELF
            SET PROP-SIZE OF TEMP58 TO TEMP57
            MOVE 2 TO TEMP59
            SET TEMP60 TO PROP-BUTTON1 OF SELF
            MOVE TEMP59 TO PROP-TABINDEX OF TEMP60
            SET TEMP61 TO N"..."
            SET TEMP62 TO PROP-BUTTON1 OF SELF
            SET PROP-TEXT OF TEMP62 TO TEMP61
            SET TEMP63 TO PROP-BUTTON1 OF SELF
            SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP63 TO B"1"
            SET TEMP64 TO PROP-BUTTON1 OF SELF
            INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click"
                  RETURNING TEMP65
            INVOKE TEMP64 "add_Click" USING BY VALUE TEMP65
        *
        *button2
        *
            SET TEMP66 TO PROP-TOP OF ENUM-ANCHORSTYLES
            SET TEMP67 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
            SET TEMP68 TO FUNCTION ENUM-OR(TEMP66 TEMP67)
            SET TEMP69 TO TEMP68
            SET TEMP70 TO PROP-BUTTON2 OF SELF
            SET PROP-ANCHOR OF TEMP70 TO TEMP69
            MOVE 257 TO TEMP71
            MOVE 75 TO TEMP72
            INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP71 BY VALUE TEMP72 RETURNING TEMP73
            SET TEMP74 TO PROP-BUTTON2 OF SELF
            SET PROP-LOCATION OF TEMP74 TO TEMP73
            SET TEMP75 TO N"button2"
            SET TEMP76 TO PROP-BUTTON2 OF SELF
            SET PROP-NAME OF TEMP76 TO TEMP75
            MOVE 23 TO TEMP77
            MOVE 23 TO TEMP78
            INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP77 BY VALUE TEMP78 RETURNING TEMP79
            SET TEMP80 TO PROP-BUTTON2 OF SELF
            SET PROP-SIZE OF TEMP80 TO TEMP79
            MOVE 5 TO TEMP81
            SET TEMP82 TO PROP-BUTTON2 OF SELF
            MOVE TEMP81 TO PROP-TABINDEX OF TEMP82
            SET TEMP83 TO N"..."
            SET TEMP84 TO PROP-BUTTON2 OF SELF
            SET PROP-TEXT OF TEMP84 TO TEMP83
            SET TEMP85 TO PROP-BUTTON2 OF SELF
            SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP85 TO B"1"
            SET TEMP86 TO PROP-BUTTON2 OF SELF
            INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button2_Click"
                  RETURNING TEMP87
            INVOKE TEMP86 "add_Click" USING BY VALUE TEMP87
        *
        *textBox2
        *
            SET TEMP88 TO PROP-TOP OF ENUM-ANCHORSTYLES
            SET TEMP89 TO PROP-LEFT OF ENUM-ANCHORSTYLES
            SET TEMP90 TO FUNCTION ENUM-OR(TEMP88 TEMP89)
            SET TEMP91 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
            SET TEMP92 TO FUNCTION ENUM-OR(TEMP90 TEMP91)
            SET TEMP93 TO TEMP92
            SET TEMP94 TO PROP-TEXTBOX2 OF SELF
            SET PROP-ANCHOR OF TEMP94 TO TEMP93
            MOVE 14 TO TEMP95
            MOVE 77 TO TEMP96
            INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP95 BY VALUE TEMP96 RETURNING TEMP97
            SET TEMP98 TO PROP-TEXTBOX2 OF SELF
            SET PROP-LOCATION OF TEMP98 TO TEMP97
            SET TEMP99 TO N"textBox2"
            SET TEMP100 TO PROP-TEXTBOX2 OF SELF
            SET PROP-NAME OF TEMP100 TO TEMP99
            MOVE 237 TO TEMP101
            MOVE 19 TO TEMP102
            INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP101 BY VALUE TEMP102 RETURNING TEMP103
            SET TEMP104 TO PROP-TEXTBOX2 OF SELF
            SET PROP-SIZE OF TEMP104 TO TEMP103
            MOVE 4 TO TEMP105
            SET TEMP106 TO PROP-TEXTBOX2 OF SELF
            MOVE TEMP105 TO PROP-TABINDEX OF TEMP106
        *
        *label2
        *
```

```
        SET TEMP107 TO PROP-LABEL2 OF SELF
        SET PROP-AUTOSIZE OF TEMP107 TO B"1"
        MOVE 12 TO TEMP108
        MOVE 62 TO TEMP109
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP108 BY VALUE TEMP109 RETURNING TEMP110
        SET TEMP111 TO PROP-LABEL2 OF SELF
        SET PROP-LOCATION OF TEMP111 TO TEMP110
        SET TEMP112 TO N"label2"
        SET TEMP113 TO PROP-LABEL2 OF SELF
        SET PROP-NAME OF TEMP113 TO TEMP112
        MOVE 97 TO TEMP114
        MOVE 12 TO TEMP115
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP114 BY VALUE TEMP115 RETURNING TEMP116
        SET TEMP117 TO PROP-LABEL2 OF SELF
        SET PROP-SIZE OF TEMP117 TO TEMP116
        MOVE 3 TO TEMP118
        SET TEMP119 TO PROP-LABEL2 OF SELF
        MOVE TEMP118 TO PROP-TABINDEX OF TEMP119
        SET TEMP120 TO N"出力(索引ファイル):"
        SET TEMP121 TO PROP-LABEL2 OF SELF
        SET PROP-TEXT OF TEMP121 TO TEMP120
    *
    *button3
    *
        MOVE 112 TO TEMP122
        MOVE 121 TO TEMP123
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP122 BY VALUE TEMP123 RETURNING TEMP124
        SET TEMP125 TO PROP-BUTTON3 OF SELF
        SET PROP-LOCATION OF TEMP125 TO TEMP124
        SET TEMP126 TO N"button3"
        SET TEMP127 TO PROP-BUTTON3 OF SELF
        SET PROP-NAME OF TEMP127 TO TEMP126
        MOVE 60 TO TEMP128
        MOVE 30 TO TEMP129
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP128 BY VALUE TEMP129 RETURNING TEMP130
        SET TEMP131 TO PROP-BUTTON3 OF SELF
        SET PROP-SIZE OF TEMP131 TO TEMP130
        MOVE 6 TO TEMP132
        SET TEMP133 TO PROP-BUTTON3 OF SELF
        MOVE TEMP132 TO PROP-TABINDEX OF TEMP133
        SET TEMP134 TO N"変換"
        SET TEMP135 TO PROP-BUTTON3 OF SELF
        SET PROP-TEXT OF TEMP135 TO TEMP134
        SET TEMP136 TO PROP-BUTTON3 OF SELF
        SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP136 TO B"1"
        SET TEMP137 TO PROP-BUTTON3 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button3_Click"
                RETURNING TEMP138
        INVOKE TEMP137 "add_Click" USING BY VALUE TEMP138
    *
    *openFileDialog1
    *
        SET TEMP139 TO N"CSV ファイル|*.csv|すべてのファイル|*.*"
        SET TEMP140 TO PROP-OPENFILEDIALOG1 OF SELF
        SET PROP-FILTER OF TEMP140 TO TEMP139
    *
    *saveFileDialog1
    *
        SET TEMP141 TO N"doc1"
        SET TEMP142 TO PROP-SAVEFILEDIALOG1 OF SELF
        SET PROP-FILENAME OF TEMP142 TO TEMP141
    *
    *MainForm
    *
        MOVE 6.00000000000000E+00 TO TEMP143
        MOVE 1.20000000000000E+01 TO TEMP144
        INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP143 BY VALUE TEMP144 RETURNING TEMP145
        SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP145
        SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
        MOVE 292 TO TEMP146
        MOVE 183 TO TEMP147
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP146 BY VALUE TEMP147 RETURNING TEMP148
        SET PROP-CLIENTSIZE OF SELF TO TEMP148
        SET TEMP150 TO PROP-BUTTON3 OF SELF
        SET TEMP149 TO PROP-CONTROLS OF SELF
        INVOKE TEMP149 "Add" USING BY VALUE TEMP150
        SET TEMP152 TO PROP-BUTTON2 OF SELF
        SET TEMP151 TO PROP-CONTROLS OF SELF
        INVOKE TEMP151 "Add" USING BY VALUE TEMP152
        SET TEMP154 TO PROP-TEXTBOX2 OF SELF
        SET TEMP153 TO PROP-CONTROLS OF SELF
```

```
           INVOKE TEMP153 "Add" USING BY VALUE TEMP154
           SET TEMP156 TO PROP-LABEL2 OF SELF
           SET TEMP155 TO PROP-CONTROLS OF SELF
           INVOKE TEMP155 "Add" USING BY VALUE TEMP156
           SET TEMP158 TO PROP-BUTTON1 OF SELF
           SET TEMP157 TO PROP-CONTROLS OF SELF
           INVOKE TEMP157 "Add" USING BY VALUE TEMP158
           SET TEMP160 TO PROP-TEXTBOX1 OF SELF
           SET TEMP159 TO PROP-CONTROLS OF SELF
           INVOKE TEMP159 "Add" USING BY VALUE TEMP160
           SET TEMP162 TO PROP-LABEL1 OF SELF
           SET TEMP161 TO PROP-CONTROLS OF SELF
           INVOKE TEMP161 "Add" USING BY VALUE TEMP162
           SET TEMP163 TO N"MainForm"
           SET PROP-NAME OF SELF TO TEMP163
           SET TEMP164 TO N"CsvToIdx"
           SET PROP-TEXT OF SELF TO TEMP164
           INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
           INVOKE SELF "PerformLayout"
       END METHOD INITIALIZECOMPONENT.

       METHOD-ID. NEW.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
           INVOKE SELF "InitializeComponent".
       END METHOD NEW.

       METHOD-ID. button1_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 result OBJECT REFERENCE ENUM-DIALOGRESULT.
       01 inputfile OBJECT REFERENCE CLASS-STRING.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      * OpenFile ダイアログを表示します
           INVOKE openFileDialog1 "ShowDialog" RETURNING result.
           IF result = OK OF ENUM-DIALOGRESULT THEN
      * テキストボックスにファイル名を設定します。
               SET inputfile TO PROP-FILENAME OF openFileDialog1
               SET PROP-TEXT OF textBox1 TO inputfile
               SET inputfile TO CLASS-PATH :: "ChangeExtension" (inputfile ".IDX")
               SET PROP-TEXT OF textBox2 TO inputfile
           END-IF.
       END METHOD button1_Click.

       METHOD-ID. button2_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 result OBJECT REFERENCE ENUM-DIALOGRESULT.
       01 outputfile OBJECT REFERENCE CLASS-STRING.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      * SaveFile ダイアログを表示します
           SET PROP-FILENAME OF saveFileDialog1 TO PROP-TEXT OF textBox2.
           INVOKE saveFileDialog1 "ShowDialog" RETURNING result.
           IF result = OK OF ENUM-DIALOGRESULT THEN
      * テキストボックスにファイル名を設定します。
               SET outputfile TO PROP-FILENAME OF saveFileDialog1
               SET PROP-TEXT OF textBox2 TO outputfile
           END-IF.
       END METHOD button2_Click.

       METHOD-ID. button3_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 proc OBJECT REFERENCE DELEGATE-THREADSTART.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *    バックグラウンドで抽出処理を開始します。
           IF indicator = NULL
               SET indicator TO CLASS-INDICATORFORM :: "NEW" .
           INVOKE SELF "StopThread".
           INVOKE DELEGATE-THREADSTART "NEW" USING SELF "ThreadProc" RETURNING proc.
           SET background TO CLASS-THREAD :: "NEW" (proc).
```

```
           SET PROP-ISBACKGROUND OF background TO B'1'.
           INVOKE background "Start".

       *   インジケータフォーム(モーダルフォーム)に経過を表示します。
           INVOKE indicator "ShowDialog" USING SELF.
       END METHOD button3_Click.

       METHOD-ID. Extract.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 csvpath OBJECT REFERENCE CLASS-STRING.
       01 idxpath OBJECT REFERENCE CLASS-STRING.
       01 csvdir  PIC X(256).
       01 csvfile PIC X(256).
       01 idxfile PIC X(256).
       01 emptysting OBJECT REFERENCE CLASS-STRING.
       01 ret OBJECT REFERENCE CLASS-BOOLEAN.
       01 msgboxicon OBJECT REFERENCE ENUM-MESSAGEBOXICON.
       01 msgboxbuttons OBJECT REFERENCE ENUM-MESSAGEBOXBUTTONS.
       01 msgboxtitle OBJECT REFERENCE CLASS-STRING.
       01 msgboxbody OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION.
       *   CSV ファイルからデータを抽出して索引ファイルを作成するために
       *   Extraction プログラムを呼び出します。
           SET emptysting TO PROP-EMPTY OF CLASS-STRING.
           SET csvpath TO PROP-TEXT OF textBox1.
           INVOKE csvpath "Equals" USING emptysting RETURNING ret.
           IF ret = B'0'
              SET idxpath TO PROP-TEXT OF textBox2
              INVOKE idxpath "Equals" USING emptysting RETURNING ret
              IF ret = B'0'

       *   PATH をフォルダ名とファイル名に分離します。
                 SET csvdir TO CLASS-PATH :: "GetDirectoryName" (csvpath)
                 SET csvfile TO CLASS-PATH :: "GetFileName" (csvpath)
                 SET idxfile TO idxpath
       *   CSV ファイルからデータを抽出し索引ファイルを生成します。
                 CALL "CsvToIdx.Extraction" USING csvdir csvfile idxfile indicator
              ELSE
                 SET msgboxtitle TO PROP-TEXT OF SELF
                 SET msgboxbody TO N"索引ファイル名が指定されていません。"
                 SET msgboxicon TO PROP-EXCLAMATION OF ENUM-MESSAGEBOXICON
                 SET msgboxbuttons TO PROP-OK OF ENUM-MESSAGEBOXBUTTONS
                 GO TO ERRORPROC

              END-IF
           ELSE
                 SET msgboxtitle TO PROP-TEXT OF SELF
                 SET msgboxbody TO N"CSV ファイル名が指定されていません。"
                 SET msgboxicon TO PROP-EXCLAMATION OF ENUM-MESSAGEBOXICON
                 SET msgboxbuttons TO PROP-OK OF ENUM-MESSAGEBOXBUTTONS
                 GO TO ERRORPROC
           END-IF.

           GO TO EXITPROC.

       ERRORPROC.
           INVOKE CLASS-MESSAGEBOX "Show"
                 USING indicator  msgboxbody msgboxtitle msgboxbuttons msgboxicon.

       EXITPROC.
       *   インジケータフォームを閉じます。
           INVOKE indicator "Close".
       END METHOD Extract.

       METHOD-ID. ThreadProc.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 mi OBJECT REFERENCE DELEGATE-METHODINVOKER.
       PROCEDURE DIVISION.
       *   Extract をバックグラウンドで実行するために MethodInvoker デリゲートを
       *   設定し、非同期で BeginInvoke を実行させます。
           INVOKE DELEGATE-METHODINVOKER "NEW" USING SELF "Extract" RETURNING mi.
           INVOKE SELF "BeginInvoke" USING mi.
       END METHOD ThreadProc.

       METHOD-ID. StopThread.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
```

```
      *   バックグラウンドのスレッドを停止します。
          IF background NOT = NULL
             INVOKE background "Interrupt"
             SET background TO NULL
          END-IF.
       END METHOD StopThread.

       END OBJECT.
       END CLASS CLASS-THIS.
```

ソースコード：Extraction.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Extraction AS "CsvToIdx.Extraction".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-INDICATORFORM AS "CsvToIdx.IndicatorForm"
  .
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT CSV-FILE ASSIGN TO CSV-PATH
     ORGANIZATION IS LINE SEQUENTIAL.
   SELECT ZIPCODE-FILE ASSIGN TO IDXFILE
     ORGANIZATION IS INDEXED
     ACCESS MODE IS DYNAMIC
     RECORD KEY IS ZIPCODE OF ZIPCODE-RECORD WITH DUPLICATES.
DATA DIVISION.
FILE SECTION.
FD CSV-FILE RECORD IS VARYING IN SIZE FROM 1 TO 2000
   DEPENDING ON LINE-SIZE.
01 CSV-RECORD.
  02 CSV-LETTER PIC X OCCURS 1 TO 2000
   DEPENDING ON LINE-SIZE.    *> CSV ファイル行レコード
FD ZIPCODE-FILE.
01 ZIPCODE-RECORD.
  02 ZIPCODE PIC 9(7).         *> 郵便番号
  02 PREFECTURE-NAME PIC N(5).  *> 都道府県名
  02 CITY-NAME PIC N(20).      *> 市町村名
  02 AREA-NAME PIC N(40).      *> 町域名
WORKING-STORAGE SECTION.
01 CSVROW.
  02 COLUMN01  PIC X(9).        *> 1.全国地方公共団体コード
  02 COLUMN02  PIC X(5).        *> 2.旧郵便番号(5桁)
  02 ZIPCODE   PIC X(7).        *> 3.郵便番号(7桁)
  02 COLUMN04  PIC X(8).        *> 4.都道府県名(半角カナ)
  02 COLUMN05  PIC X(25).       *> 5.市町村名(半角カナ)
  02 COLUMN06  PIC X(80).       *> 6.町域名(半角カナ)
  02 PREFECTURE-NAME PIC N(5).   *> 7.都道府県名
  02 CITY-NAME  PIC N(20).      *> 8.市町村名
  02 AREA-NAME  PIC N(40).      *> 9.町域名
  02 COLUMN10  PIC X(4).      *> 10.一町域が二以上の郵便番号で表される場合の表示(1:該当,0:該当せず)
  02 COLUMN11  PIC X(4).      *> 11.小字毎に番地が起番されている町域の表示(1:該当,0:該当せず)
  02 COLUMN12  PIC X(4).      *> 12.丁目を有する町域の場合の表示(1:該当,0:該当せず)
  02 COLUMN13  PIC X(4).     *> 13.一つの郵便番号で二以上の町域を表す場合の表示(1:該当,0:該当せず)
  02 COLUMN14  PIC X(4).      *> 14.更新の表示(0:変更なし,1:変更あり,2:廃止)
  02 COLUMN15  PIC X(4).      *> 15.変更理由(0:変更なし,1:市政/区政/町政/分区/政令指定都市施行,
                              *>            2:住居表示の実施,
                              *>            3:区画整理,4:郵便区調整/集配局新設,5:訂正,6:廃止)

01 LINE-SIZE   PIC 9(4).
01 CSV-PATH    PIC X(256) VALUE SPACE.
01 LEN-DIR     PIC S9(9) COMP-5.
01 LEN-FILE     PIC S9(9) COMP-5.
01 NUM-ROWS  PIC S9(9) COMP-5.
01 CSV-LABEL PIC X      VALUE SPACE.
  88 END-CSV           VALUE "E".
01 CSV-LINE  PIC X(2000) VALUE SPACE.
LINKAGE SECTION.
01 CSVDIR  PIC X(256).
01 CSVFILE PIC X(256).
01 IDXFILE PIC X(256).
01 INDICATOR OBJECT REFERENCE CLASS-INDICATORFORM.
PROCEDURE DIVISION USING CSVDIR CSVFILE IDXFILE INDICATOR.
*    初期設定
    INVOKE INDICATOR "InitProgress" USING 0.

*    CSV ファイルパスの生成
    COMPUTE LEN-DIR = FUNCTION STORED-CHAR-LENGTH(CSVDIR).
    COMPUTE LEN-FILE = FUNCTION STORED-CHAR-LENGTH(CSVFILE).
```

```
           STRING CSVDIR (1:LEN-DIR) "¥" CSVFILE (1:LEN-FILE) DELIMITED BY SIZE INTO CSV-PATH.
       *   読み込む CSV ファイルを開きます。
           OPEN INPUT CSV-FILE.

       *   データ件数を求めて、インジケータの初期値とします。
           MOVE 0 TO NUM-ROWS.
           PERFORM UNTIL END-CSV
              READ CSV-FILE AT END SET END-CSV TO TRUE
                  NOT AT END COMPUTE NUM-ROWS = NUM-ROWS + 1
              END-READ
           END-PERFORM.

       *   インジケータを更新します。
           INVOKE INDICATOR "InitProgress" USING NUM-ROWS.

       *   読込み位置初期化のため、一度 CSV ファイルを閉じます。
           CLOSE CSV-FILE.
       *   標識を初期化します。
           MOVE SPACE TO CSV-LABEL.

       *   再度読み込む CSV ファイルを開きます。
           OPEN INPUT CSV-FILE.
       *   書き出す索引ファイルを開きます。
           OPEN OUTPUT ZIPCODE-FILE.

       *   CSV ファイルの最初の行レコードを読み込みます。
           READ CSV-FILE AT END SET END-CSV TO TRUE.
           PERFORM UNTIL END-CSV
       *   CSV 形式の行レコードを要素に分解します。
              MOVE CSV-RECORD TO CSV-LINE
              UNSTRING CSV-LINE (1:LINE-SIZE) INTO CSVROW BY CSV-FORMAT
       *   シフト JIS 形式で書き込みます。
              MOVE ZIPCODE OF CSVROW TO ZIPCODE OF ZIPCODE-RECORD
              MOVE FUNCTION ACP-OF(PREFECTURE-NAME OF CSVROW) TO PREFECTURE-NAME OF ZIPCODE-RECORD
              MOVE FUNCTION ACP-OF(CITY-NAME OF CSVROW) TO CITY-NAME OF ZIPCODE-RECORD
              MOVE FUNCTION ACP-OF(AREA-NAME OF CSVROW) TO AREA-NAME OF ZIPCODE-RECORD
              WRITE ZIPCODE-RECORD
       *   インジケーターを更新します。
              INVOKE INDICATOR "IncrProgress"
       *   次の行を読み込みます。
              READ CSV-FILE AT END SET END-CSV TO TRUE
              END-READ
           END-PERFORM.

       *   ファイルを閉じます。
           CLOSE CSV-FILE.
           CLOSE ZIPCODE-FILE.

        END PROGRAM Extraction.
```

ソースコード：IndecatorForm.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "CsvToIdx.IndicatorForm"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   ENUM ENUM-FORMBORDERSTYLE AS "System.Windows.Forms.FormBorderStyle"
   ENUM ENUM-FORMSTARTPOSITION AS "System.Windows.Forms.FormStartPosition"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-PROGRESSBAR AS "System.Windows.Forms.ProgressBar"
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-CENTERPARENT AS "CenterParent"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-CONTROLBOX AS "ControlBox"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-FIXEDTOOLWINDOW AS "FixedToolWindow"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-FORMBORDERSTYLE AS "FormBorderStyle"
   PROPERTY PROP-LABEL1 AS "label1"
   PROPERTY PROP-LOCATION AS "Location"
   PROPERTY PROP-MAXIMIZEBOX AS "MaximizeBox"
   PROPERTY PROP-MAXIMUM AS "Maximum"
   PROPERTY PROP-MINIMIZEBOX AS "MinimizeBox"
   PROPERTY PROP-MINIMUM AS "Minimum"
   PROPERTY PROP-NAME AS "Name"
   PROPERTY PROP-PROGRESSBAR1 AS "progressBar1"
   PROPERTY PROP-SHOWICON AS "ShowIcon"
   PROPERTY PROP-SHOWINTASKBAR AS "ShowInTaskbar"
   PROPERTY PROP-SIZE AS "Size"
   PROPERTY PROP-STARTPOSITION AS "StartPosition"
   PROPERTY PROP-STEP AS "Step"
   PROPERTY PROP-TABINDEX AS "TabIndex"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-VALUE AS "Value"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 label1 OBJECT REFERENCE CLASS-LABEL.
01 progressBar1 OBJECT REFERENCE CLASS-PROGRESSBAR.
01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
PROCEDURE DIVISION.

METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
PROCEDURE DIVISION USING BY VALUE disposing.
   IF disposing NOT = B"0" AND components NOT = NULL THEN
     INVOKE components "Dispose"
   END-IF.
   INVOKE SUPER "Dispose" USING disposing.
END METHOD DISPOSE.
```

```
       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP2 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP3 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP4 BINARY-LONG.
       01 TEMP5 BINARY-LONG.
       01 TEMP6 OBJECT REFERENCE CLASS-POINT.
       01 TEMP7 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP8 OBJECT REFERENCE CLASS-STRING.
       01 TEMP9 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP10 BINARY-LONG.
       01 TEMP11 BINARY-LONG.
       01 TEMP12 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP13 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP14 BINARY-LONG.
       01 TEMP15 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP16 OBJECT REFERENCE CLASS-STRING.
       01 TEMP17 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP18 BINARY-LONG.
       01 TEMP19 BINARY-LONG.
       01 TEMP20 OBJECT REFERENCE CLASS-POINT.
       01 TEMP21 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP22 OBJECT REFERENCE CLASS-STRING.
       01 TEMP23 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP24 BINARY-LONG.
       01 TEMP25 BINARY-LONG.
       01 TEMP26 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP27 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP28 BINARY-LONG.
       01 TEMP29 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP30 BINARY-LONG.
       01 TEMP31 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP32 COMP-1.
       01 TEMP33 COMP-1.
       01 TEMP34 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP35 BINARY-LONG.
       01 TEMP36 BINARY-LONG.
       01 TEMP37 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP38 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP39 OBJECT REFERENCE CLASS-PROGRESSBAR.
       01 TEMP40 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP41 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP42 OBJECT REFERENCE CLASS-STRING.
       01 TEMP43 OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="progressBar1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ProgressBar" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="SuspendLayout" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="AutoSize" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="82" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="0" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
```

```
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="変換しています..." />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="progressBar1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="progressBar1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="24" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="progressBar1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="progressBar1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="progressBar1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="255" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="23" />
*</prop>
*</object>
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="progressBar1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Step" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="progressBar1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="IndicatorForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleDimensions" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.SizeF" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="6" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AutoScaleMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Font" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="279" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="58" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ControlBox" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="progressBar1" />
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="FormBorderStyle" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.FormBorderStyle" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="FixedToolWindow" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="MaximizeBox" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MinimizeBox" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="IndicatorForm" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ShowIcon" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ShowInTaskbar" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="StartPosition" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.FormStartPosition" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="CenterParent" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="CsvToIdx" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP1
          SET PROP-LABEL1 OF SELF TO TEMP1
          INVOKE CLASS-PROGRESSBAR "NEW" RETURNING TEMP2
          SET PROP-PROGRESSBAR1 OF SELF TO TEMP2
          INVOKE SELF "SuspendLayout"
      *
      *label1
      *
          SET TEMP3 TO PROP-LABEL1 OF SELF
          SET PROP-AUTOSIZE OF TEMP3 TO B"1"
          MOVE 12 TO TEMP4
          MOVE 9 TO TEMP5
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP4 BY VALUE TEMP5 RETURNING TEMP6
          SET TEMP7 TO PROP-LABEL1 OF SELF
          SET PROP-LOCATION OF TEMP7 TO TEMP6
          SET TEMP8 TO N"label1"
          SET TEMP9 TO PROP-LABEL1 OF SELF
          SET PROP-NAME OF TEMP9 TO TEMP8
          MOVE 82 TO TEMP10
          MOVE 12 TO TEMP11
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP10 BY VALUE TEMP11 RETURNING TEMP12
          SET TEMP13 TO PROP-LABEL1 OF SELF
          SET PROP-SIZE OF TEMP13 TO TEMP12
          MOVE 0 TO TEMP14
          SET TEMP15 TO PROP-LABEL1 OF SELF
          MOVE TEMP14 TO PROP-TABINDEX OF TEMP15
          SET TEMP16 TO N"変換しています..."
          SET TEMP17 TO PROP-LABEL1 OF SELF
          SET PROP-TEXT OF TEMP17 TO TEMP16
      *
      *progressBar1
      *
          MOVE 12 TO TEMP18
          MOVE 24 TO TEMP19
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP18 BY VALUE TEMP19 RETURNING TEMP20
          SET TEMP21 TO PROP-PROGRESSBAR1 OF SELF
          SET PROP-LOCATION OF TEMP21 TO TEMP20
          SET TEMP22 TO N"progressBar1"
          SET TEMP23 TO PROP-PROGRESSBAR1 OF SELF
          SET PROP-NAME OF TEMP23 TO TEMP22
          MOVE 255 TO TEMP24
          MOVE 23 TO TEMP25
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP24 BY VALUE TEMP25 RETURNING TEMP26
          SET TEMP27 TO PROP-PROGRESSBAR1 OF SELF
          SET PROP-SIZE OF TEMP27 TO TEMP26
          MOVE 1 TO TEMP28
          SET TEMP29 TO PROP-PROGRESSBAR1 OF SELF
          MOVE TEMP28 TO PROP-STEP OF TEMP29
          MOVE 1 TO TEMP30
          SET TEMP31 TO PROP-PROGRESSBAR1 OF SELF
          MOVE TEMP30 TO PROP-TABINDEX OF TEMP31
      *
      *IndicatorForm
      *
          MOVE 6.00000000000000E+00 TO TEMP32
          MOVE 1.20000000000000E+01 TO TEMP33
          INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP32 BY VALUE TEMP33 RETURNING TEMP34
```

```
          SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP34
          SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
          MOVE 279 TO TEMP35
          MOVE 58 TO TEMP36
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP35 BY VALUE TEMP36 RETURNING TEMP37
          SET PROP-CLIENTSIZE OF SELF TO TEMP37
          SET PROP-CONTROLBOX OF SELF TO B"0"
          SET TEMP39 TO PROP-PROGRESSBAR1 OF SELF
          SET TEMP38 TO PROP-CONTROLS OF SELF
          INVOKE TEMP38 "Add" USING BY VALUE TEMP39
          SET TEMP41 TO PROP-LABEL1 OF SELF
          SET TEMP40 TO PROP-CONTROLS OF SELF
          INVOKE TEMP40 "Add" USING BY VALUE TEMP41
          SET PROP-FORMBORDERSTYLE OF SELF TO PROP-FIXEDTOOLWINDOW OF ENUM-FORMBORDERSTYLE
          SET PROP-MAXIMIZEBOX OF SELF TO B"0"
          SET PROP-MINIMIZEBOX OF SELF TO B"0"
          SET TEMP42 TO N"IndicatorForm"
          SET PROP-NAME OF SELF TO TEMP42
          SET PROP-SHOWICON OF SELF TO B"0"
          SET PROP-SHOWINTASKBAR OF SELF TO B"0"
          SET PROP-STARTPOSITION OF SELF TO PROP-CENTERPARENT OF ENUM-FORMSTARTPOSITION
          SET TEMP43 TO N"CsvToIdx"
          SET PROP-TEXT OF SELF TO TEMP43
          INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
          INVOKE SELF "PerformLayout"
      END METHOD INITIALIZECOMPONENT.

      METHOD-ID. NEW.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      PROCEDURE DIVISION.
          INVOKE SELF "InitializeComponent".
      END METHOD NEW.

      METHOD-ID. InitProgress.
      DATA DIVISION.
      LINKAGE SECTION.
      01 val BINARY-LONG.
      PROCEDURE DIVISION USING BY VALUE val.
     *  ProgressBar の最小値と Value をクリアします
          SET PROP-VALUE OF progressBar1 TO PROP-MINIMUM OF progressBar1.
     *  ProgressBar に最大値を設定します。
          SET PROP-MAXIMUM OF progressBar1 TO val.
          INVOKE CLASS-APPLICATION "DoEvents".
      END METHOD InitProgress.

      METHOD-ID. IncrProgress.
      DATA DIVISION.
      PROCEDURE DIVISION.
     *  ProgressBar に現在値を設定します。
          INVOKE progressBar1 "PerformStep".
          INVOKE CLASS-APPLICATION "DoEvents".
      END METHOD IncrProgress.

      END OBJECT.
      END CLASS CLASS-THIS.
```

NCOBU00941

ソースコード：Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
   .
REPOSITORY.
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-MAINFORM AS "CsvToIdx.MainForm"
   CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
   .

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-MAINFORM OBJECT REFERENCE CLASS-MAINFORM.
PROCEDURE DIVISION.
   INVOKE CLASS-APPLICATION "EnableVisualStyles".
   INVOKE CLASS-APPLICATION "SetCompatibleTextRenderingDefault" USING BY VALUE B"0".

   INVOKE CLASS-MAINFORM "NEW" RETURNING WK-MAINFORM.
   INVOKE CLASS-APPLICATION "Run" USING BY VALUE WK-MAINFORM.
END PROGRAM MAIN.
```

ソースコード：Fib_COBOL.cob

```
CLASS-ID. Fib_COBOL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   INTERFACE FIBINSTANCE AS "Fib.FibInstance"
   CLASS SYS-STRING AS "System.String"
   CLASS INT-ARRAY AS "System.Int32[]".
OBJECT. IMPLEMENTS FIBINSTANCE.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 Lang OBJECT REFERENCE SYS-STRING PROPERTY.
PROCEDURE DIVISION.
METHOD-ID. NEW.
PROCEDURE DIVISION.
   SET Lang TO "COBOL".
END METHOD NEW.
METHOD-ID. NextTerms.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I BINARY-LONG.
77 Term BINARY-LONG.
LINKAGE SECTION.
77 Cnt BINARY-LONG.
77 Term0 BINARY-LONG.
77 Term1 BINARY-LONG.
77 Result OBJECT REFERENCE INT-ARRAY.
PROCEDURE DIVISION USING BY VALUE Cnt Term0 Term1 RETURNING Result.
   INVOKE INT-ARRAY "NEW" USING BY VALUE Cnt RETURNING Result.
   PERFORM VARYING I FROM 0 BY 1 UNTIL I >= Cnt
      COMPUTE Term = Term0 + Term1
      INVOKE Result "Set" USING BY VALUE I Term
      MOVE Term1 TO Term0
      INVOKE Result "Get" USING BY VALUE I RETURNING Term1
   END-PERFORM.
END METHOD NextTerms.
END OBJECT.
END CLASS Fib_COBOL.
```

NCOBU00926

NetCOBOL for .NET データベースアクセスサンプルアプリケーション

データベースアクセスサンプルは、 COBOL によるデータベースアクセスをデモンストレーションします。 サンプルは Examples の Database サブフォルダにあります。 サンプルを試すためには、 Visual Studio からソリューションファイル(.sln)を開いてください。 なお、サンプルで使用するデータベースは Microsoft SQL Server 2016 Express LocalDB です。 Microsoft SQL Server 2016 Express LocalDB のインストールおよび実行環境の設定については、NetCOBOL for .NET サンプルデータベースのセットアップを参照してください。

1. ADO.NET Access

フォルダ	ADONETAccess
ソリューションファイル	ADONETAccess.sln
ソースコード	ADONETAccess.cob

説明

データベース(SQL)機能を使ってデータベースからデータを取り出しホスト変数に格納する例を示します。 データベースはサーバ上に存在し、クライアント側からこれにアクセスします。 このサンプルでのデータベースのアクセスは、ADO.NET データプロバイダーを使用して行います。 ADO.NET データプロバイダーを使用するデータベースアクセスについては、ADO.NET 概要を参照してください。

2. ODBC Access

フォルダ	ODBCAccess
ソリューションファイル	ODBCAccess.sln
ソースコード	ODBCAccess.cob

説明

データベース機能のより進んだ使い方として、 データベースの複数の行を 1 つの埋込み SQL 文で操作する例を示します。 このサンプルでのデータベースのアクセスは、ODBC ドライバを経由して行います。 ODBC ドライバを使用するデータベースアクセスについては、ODBC 概要を参照してください。

3. Data Binding

フォルダ	DataBinding
ソリューションファイル	DataBinding.sln
ソースコード	DataBinding.cob,OrderList.cob,Main.cob

説明

ADO.NET のクラスライブラリを使用してデータベースをアクセスする Windows フォームアプリケーションの例を示します。 データベースとデータグリッドを連結し、 サーバ上のデータベースのテーブルの参照や更新、 レコードの削除および追加を行います。

4. ZipCode Database Setup

フォルダ	ZipCodeDBSetup
ソリューションファイル	ZipCodeDBSetup.sln
ソースコード	DBSetup.cob,Main.cob

説明

ADO.NET のクラスライブラリを使用して郵便番号データ(CSV ファイル)を SQL Server 上のデータベースにテーブルを作成するサンプルアプリケーションです。 このサンプルプログラムでセットアップされたデータベースは WCF サンプルアプリケーションで使用します。

NCOBU00715

NetCOBOL for .NET サンプルデータベースのセットアップ

Microsoft SQL Server 2016 Express LocalDB のインストール

NetCOBOL for .NET のサンプルアプリケーションでは、Microsoft SQL Server 2016 Express LocalDB を使用します。

Microsoft SQL Server 2016 Express LocalDB は、Microsoft 社のダウンロードサイトから入手してください。 また、Visual Studio 2017 にも同梱されています。 Microsoft SQL Server 2016 Express LocalDB がインストールされていないときは、 Visual Studio インストーラーを起動し、 「個別のコンポーネント」から「SQL Server Express 2016 LocalDB」 を追加インストールしてください。

なお、Microsoft SQL Server 2016 Express LocalDB は" (localdb)¥MSSQLLocalDB "という固定のインスタンス名を使用します。

ローカルアカウントでサンプルデータベースの操作を行う場合、SQL Server の必要なリソースにアクセスできないと SQL Server と接続することはできません。 その場合はローカルアカウントに適切なアクセス許可を与える必要があります。

サンプルデータベースの作成

サンプルデータベースを設定するには、 コマンドプロンプトで `Database` のフォルダにカレントフォルダを移動し、 以下のコマンドを実行してください。

```
SetupSampleDB
```

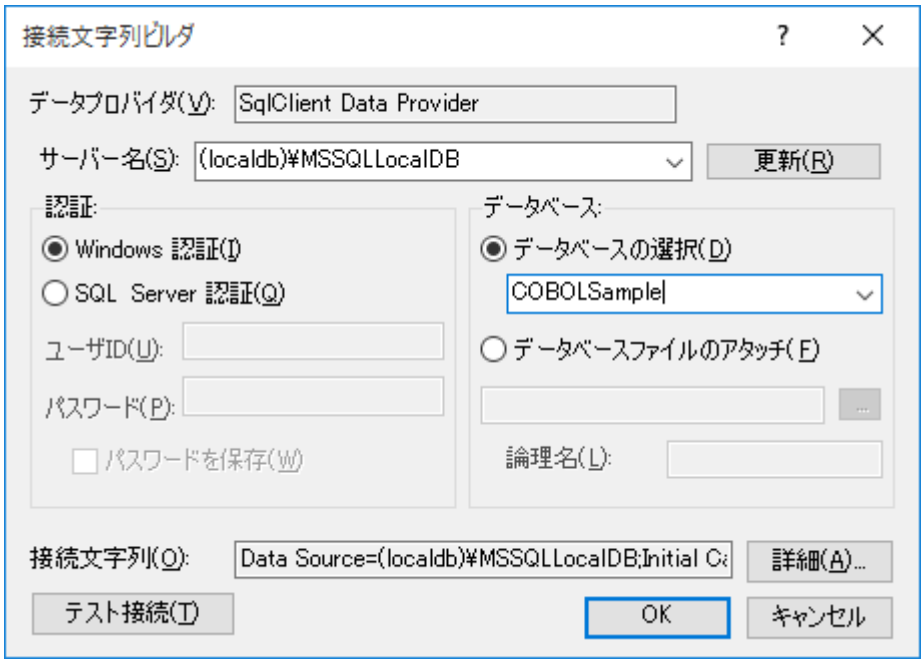
SQL Server 上に、 "COBOLSample"というデータベースが作成され、 "STOCK"、"ORDERS"および"COMPANY"というテーブルが作成されます。

ADO.NET 接続文字列の設定

ADO.NET 接続を行う場合、アプリケーション構成ファイルに指定する接続文字列は以下のように記述します。

```
Data Source=(localdb)¥MSSQLLocalDB;
Initial Catalog=COBOLSample;Integrated Security=True;Persist Security Info=False
```

実行環境設定ユーティリティを使用して、接続文字列の設定を行う場合は以下のように設定します。



ODBC データソースの設定

ODBC データソースはコントロールパネル([管理ツール]-[データソース(ODBC)])から、以下のデータソースの設定を行ってください。

ドライバ	SQL Server Native Client 11.0
データソース名	COBOLSample
サーバ名	(localdb)¥MSSQLLocalDB
既定のデータベース	COBOLSample

NCOBU00722

ソースコード：ADONETAccess.cob

```
000010*================================================================
000020*「データベース機能を使ったプログラム」
000030*    データベースのある表からカーソルを使って全データを取り出して
000040*  表示します。
000050*
000060*================================================================
000070****  注意事項  ****
000080*================================================================
000090*     このプログラムでは、接続データベースとして
000100*     Microsoft(R) SQL Server(TM)を想定しています。
000110*     接続データベースとしてMicrosoft(R) SQL Server(TM)以外を
000120*     使用する場合、接続確認の処理を次のように変更してください。
000130*
000140*      000570 接続確認.
000150*      000580     IF SQLSTATE = "00000" THEN
000160*      000590*       SQLSTATE = "01000" THEN
000170*      000600       DISPLAY "接続に成功しました"
000180*      000610       DISPLAY " "
000190*      000620     ELSE
000200*      000630       GO TO P-END
000210*      000640     END-IF
000220*
000230*     Copyright 2000-2010 FUJITSU LIMITED
000240*================================================================
000250 IDENTIFICATION DIVISION.
000260 PROGRAM-ID.    MAIN AS "ADONETAccess.Main".
000270 ENVIRONMENT    DIVISION.
000280 DATA          DIVISION.
000290 WORKING-STORAGE SECTION.
000300*================================================================
000310*  ホスト変数宣言
000320*================================================================
000330     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000340 01 在庫表.
000350   02 製品番号   PIC S9(4) COMP-5.
000360   02 製品名     PIC X(20).
000370   02 在庫数量   PIC S9(9) COMP-5.
000380   02 倉庫番号   PIC S9(4) COMP-5.
000390 01 SQLSTATE     PIC X(5).
000400 01 SQLMSG       PIC X(128).
000410     EXEC SQL END DECLARE SECTION END-EXEC.
000420 01 データ件数   PIC 9(2).
000430 01 カーソル状態 PIC 1(1) BIT.
000440 88 カーソルオープン VALUE B"1".
000450 PROCEDURE DIVISION.
000460*================================================================
000470*  カーソルを宣言します
000480*================================================================
000490     EXEC SQL
000500       DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
000510     END-EXEC.
000520*================================================================
000530*  データベースに接続 (DEFAULTサーバに接続)します
000540*================================================================
000550 P-START.
000560     EXEC SQL CONNECT TO DEFAULT END-EXEC.
000570* 接続確認
000580     IF SQLSTATE = "00000" OR
000590        SQLSTATE = "01000" THEN
000600       DISPLAY "接続に成功しました"
000610       DISPLAY " "
000620     ELSE
000630       GO TO P-END
000640     END-IF
000650*================================================================
000660*  例外事象が発生した場合の動作を指定します
000670*================================================================
000680     EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
```

```
000690    EXEC SQL WHENEVER SQLERROR GO TO :P-END END-EXEC.
000700*================================================================
000710*  カーソルをＯＰＥＮします
000720*================================================================
000730    EXEC SQL OPEN CUR1 END-EXEC.
000740    SET カーソルオープン TO TRUE.
000750*================================================================
000760*  カーソルを用いて順にデータ取り出します。
000770*================================================================
000780 P-FETCH.
000790    MOVE 0 TO データ件数
000800*
000810    EXEC SQL FETCH CUR1 INTO :在庫表  END-EXEC
000820    PERFORM TEST BEFORE
000830            UNTIL SQLSTATE = "02000"
000840      COMPUTE データ件数 = データ件数 + 1
000850      DISPLAY データ件数 NC"件目のデータ："
000860      DISPLAY NC"     製品番号  ＝  "  製品番号
000870      DISPLAY NC"     製品名    ＝  "  製品名
000880      DISPLAY NC"     在庫数量  ＝  "  在庫数量
000890      DISPLAY NC"     倉庫番号  ＝  "  倉庫番号
000900      EXEC SQL FETCH CUR1 INTO :在庫表  END-EXEC
000910    END-PERFORM
000920*
000930    DISPLAY " "
000940    DISPLAY  データ件数 NC"件のデータの取り出しに成功しました。"
000950    MOVE "00000" TO SQLSTATE.
000960*================================================================
000970*  例外事象が発生した場合の動作を無効化します。
000980*================================================================
000990 P-END.
001000    EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.
001010*================================================================
001020*  カーソルをＣＬＯＳＥします
001030*================================================================
001040    IF カーソルオープン THEN
001050      EXEC SQL CLOSE CUR1 END-EXEC
001060    END-IF.
001070*================================================================
001080*  トランザクションを終了し、接続を解除します。
001090*================================================================
001100    IF SQLSTATE = "00000" THEN
001110      EXEC SQL COMMIT WORK END-EXEC
001120    ELSE
001130      DISPLAY NC"ＳＱＬ文の実行でエラーが発生しました。"
001140      DISPLAY "      SQLSTATE : " SQLSTATE
001150      DISPLAY "      SQLMSG   : " SQLMSG
001160      EXEC SQL ROLLBACK WORK END-EXEC
001170    END-IF.
001180    EXEC SQL DISCONNECT DEFAULT END-EXEC.
001190 END-PROC.
001200    DISPLAY NC"終了します".
001210    STOP RUN.
001220 END PROGRAM MAIN.
```

NCOBU00723

ソースコード：ODBCAccess.cob

```
000010*================================================================
000020*「データベース機能を使ったプログラム」
000030*    集団項目として定義したホスト変数を使用して、データベースの表
000040*  の複数の行を１度に操作します。
000050*
000060*================================================================
000070****  注意事項  ****
000080*================================================================
000090*     このプログラムでは、接続データベースとして
000100*     Microsoft(R) SQL Server(TM)を想定しています。
000110*     接続データベースとして Microsoft(R) SQL Server(TM)以外を
000120*     使用する場合、接続確認の処理を次のように変更してください。
000130*
000140*      000510 接続確認.
000150*      000520     IF SQLSTATE = "00000" THEN
000160*      000530*       SQLSTATE = "01000" THEN
000170*      000540       DISPLAY "接続に成功しました"
000180*      000550       DISPLAY " "
000190*      000560     ELSE
000200*      000570       GO TO P-END
000210*      000580     END-IF
000220*
000230*      Copyright 2000-2010 FUJITSU LIMITED
000240*================================================================
000250 IDENTIFICATION DIVISION.
000260 PROGRAM-ID.     MAIN AS "ODBCAccess.Main".
000270 ENVIRONMENT    DIVISION.
000280 DATA DIVISION.
000290 WORKING-STORAGE SECTION.
000300*================================================================
000310*    ホスト変数宣言
000320*================================================================
000330     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000340 01 在庫表 IS GLOBAL.
000350   02 製品番号   PIC S9(4) COMP-5 OCCURS 10.
000360   02 製品名     PIC X(20)       OCCURS 10.
000370   02 在庫数量   PIC S9(9) COMP-5 OCCURS 10.
000380   02 倉庫番号   PIC S9(4) COMP-5 OCCURS 10.
000390 01 SQLSTATE     PIC X(5)   IS GLOBAL.
000400 01 SQLMSG       PIC X(128) IS GLOBAL.
000410 01 SQLINFOA.
000420   02 SQLERRD    PIC S9(9) COMP-5 OCCURS 6.
000430     EXEC SQL END DECLARE SECTION END-EXEC.
000440 01 処理数       PIC S9(4) COMP-5.
000450 PROCEDURE DIVISION.
000460*================================================================
000470*  データベースに接続 (DEFAULT サーバに接続)します
000480*================================================================
000490 P-START.
000500     EXEC SQL CONNECT TO DEFAULT END-EXEC.
000510* 接続確認
000520     IF SQLSTATE = "00000" OR
000530        SQLSTATE = "01000" THEN
000540       DISPLAY "接続に成功しました" UPON CONSOLE
000550       DISPLAY " "                  UPON CONSOLE
000560     ELSE
000570       GO TO P-END
000580     END-IF.
000590*================================================================
000600*  例外事象が発生した場合の動作を指定します
000610*================================================================
000620     EXEC SQL WHENEVER SQLERROR GO TO :P-END END-EXEC.
000630*================================================================
000640*  操作を行う前の STOCK テーブルの初期状態を表示します。
000650*================================================================
000660     DISPLAY NC"処理前のテーブルの内容".
000670     DISPLAY " ".
000680     CALL "LIST_STOCK".
```

```
000690*=================================================================
000700*  複数行指定のホスト変数を用いて、製品名が TELEVISION の製品番号を
000710*  １度に取得します。取得したデータの個数は SQLERRD(3)に返されます
000720*=================================================================
000730    DISPLAY  "製品名が TELEVISION の行の製品番号を取り出します。"
000740                                              UPON CONSOLE.
000750    EXEC SQL
000760      SELECT GNO FROM STOCK INTO :製品番号
000770         WHERE GOODS = 'TELEVISION'
000780    END-EXEC.
000790*=================================================================
000800*  取得した製品番号を使用して、在庫数の更新を行います。
000810*=================================================================
000820    DISPLAY  "以下の製品の在庫数量を 10 ずつ減少させます。"
000830                                              UPON CONSOLE.
000840    PERFORM TEST BEFORE
000850      VARYING 処理数 FROM 1 BY 1
000860            UNTIL 処理数 > SQLERRD(3)
000870      DISPLAY "  TELEVISION -> " 製品番号(処理数) UPON CONSOLE
000880    END-PERFORM.
000890*
000900    EXEC SQL
000910      UPDATE STOCK SET QOH = QOH - 10
000920         WHERE GNO = :製品番号
000930    END-EXEC.
000940*=================================================================
000950*  製品名が RADIO, SHAVER, DRIER である行のデータを削除します。
000960*=================================================================
000970    DISPLAY  "製品名が'RADIO'、'SHAVER'、'DRIER'の行を削除します。"
000980                                              UPON CONSOLE.
000990    MOVE "RADIO"  TO 製品名(1).
001000    MOVE "SHAVER" TO 製品名(2).
001010    MOVE "DRIER"  TO 製品名(3).
001020    EXEC SQL
001030      FOR 3
001040      DELETE FROM STOCK WHERE GOODS = :製品名
001050    END-EXEC.
001060*=================================================================
001070*  操作後の STOCK テーブルの状態を表示します。
001080*=================================================================
001090    DISPLAY NC"処理後のテーブルの内容".
001100    DISPLAY " ".
001110    CALL "LIST_STOCK".
001120*=================================================================
001130*  例外事象が発生した場合の動作を無効化します。
001140*=================================================================
001150 P-END.
001160    EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.
001170*=================================================================
001180*  トランザクションを終了し、接続を解除します。
001190*=================================================================
001200    IF SQLSTATE = "00000" THEN
001210      EXEC SQL COMMIT WORK END-EXEC
001220    ELSE
001230      DISPLAY NC"ＳＱＬ文の実行でエラーが発生しました。" UPON CONSOLE
001240      DISPLAY "      SQLSTATE : " SQLSTATE           UPON CONSOLE
001250      DISPLAY "      SQLMSG   : " SQLMSG             UPON CONSOLE
001260      EXEC SQL ROLLBACK WORK END-EXEC
001270    END-IF.
001280    EXEC SQL DISCONNECT DEFAULT END-EXEC.
001290 END-PROC.
001300    DISPLAY " "           UPON CONSOLE.
001310    DISPLAY NC"終了します"  UPON CONSOLE.
001320    STOP RUN.
001330
001340*=================================================================
001350*
001360*  テーブル STOCK に含まれる全データを取り出し表示します。
001370*
001380*=================================================================
001390 IDENTIFICATION DIVISION.
001400 PROGRAM-ID.   LIST_STOCK.
001410 DATA         DIVISION.
001420 WORKING-STORAGE SECTION.
001430    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
001440 01 未処理件数   PIC S9(9) COMP-5.
001450 01 処理件数     PIC S9(9) COMP-5.
001460    EXEC SQL END DECLARE SECTION END-EXEC.
```

```
001470 01 行数        PIC S9(4) BINARY.
001480 01 カーソル状態 PIC 1(1) BIT.
001490 88 カーソルオープン VALUE B"1".
001500 01 データ件数   PIC 9(2).
001510 PROCEDURE    DIVISION.
001520*================================================================
001530*  カーソルと例外発生時の処理を宣言します。
001540*================================================================
001550    EXEC SQL
001560      DECLARE CUR1 CURSOR FOR SELECT * FROM STOCK
001570    END-EXEC.
001580    EXEC SQL WHENEVER SQLERROR GO TO :LIST-EXIT END-EXEC.
001590*================================================================
001600*  テーブル中のデータ件数を集合関数 COUNT(*)を用いて取得します。
001610*================================================================
001620    EXEC SQL
001630      SELECT COUNT(*) INTO :未処理件数 FROM STOCK
001640    END-EXEC.
001650    EXEC SQL OPEN CUR1 END-EXEC.
001660    SET カーソルオープン TO TRUE.
001670*================================================================
001680*  表指定ホスト変数と FOR 句を使用して、複数行のデータを一度に取り
001690*  出して表示します。これを未処理の行がなくなるまで繰り返します
001700*================================================================
001710    MOVE 0 TO データ件数.
001720    PERFORM TEST AFTER
001730            UNTIL 未処理件数 <= 0
001740      IF 未処理件数 > 10 THEN
001750        MOVE 10        TO 処理件数
001760      ELSE
001770        MOVE 未処理件数 TO 処理件数
001780      END-IF
001790      EXEC SQL
001800        FOR :処理件数
001810        FETCH CUR1 INTO  :在庫表
001820      END-EXEC
001830      PERFORM  TEST BEFORE
001840         VARYING 行数 FROM 1 BY 1
001850                 UNTIL 行数 > 処理件数
001860        COMPUTE データ件数 = データ件数 + 1
001870        DISPLAY データ件数 NC"件目のデータ："
001880        DISPLAY NC"     製品番号  ＝  "  製品番号(行数)
001890        DISPLAY NC"     製品名    ＝  "  製品名(行数)
001900        DISPLAY NC"     在庫数量  ＝  "  在庫数量(行数)
001910        DISPLAY NC"     倉庫番号  ＝  "  倉庫番号(行数)
001920        DISPLAY  " "
001930      END-PERFORM
001940      COMPUTE 未処理件数 = 未処理件数 - 処理件数
001950    END-PERFORM.
001960    DISPLAY "全データ件数は" データ件数 "件です".
001970    DISPLAY  " ".
001980    MOVE "00000" TO SQLSTATE.
001990 LIST-EXIT.
002000*================================================================
002010*  カーソルをＣＬＯＳＥします
002020*================================================================
002030    IF カーソルオープン THEN
002040      EXEC SQL CLOSE CUR1 END-EXEC
002050    END-IF.
002060    MOVE LOW-VALUE TO 在庫表
002070 END PROGRAM LIST_STOCK.
002080 END PROGRAM MAIN.
```

ソースコード：DataBinding.cob

```
CLASS-ID. CLASS-THIS AS "DataBinding.MainForm"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS FORM-ORDERLIST AS "DataBinding.OrderList"
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ISUPPORTINITIALIZE AS "System.ComponentModel.ISupportInitialize"
   CLASS CLASS-DATAROW AS "System.Data.DataRow"
   ENUM ENUM-DATAROWVERSION AS "System.Data.DataRowVersion"
   CLASS CLASS-DATATABLE AS "System.Data.DataTable"
   CLASS CLASS-SQLCOMMAND AS "System.Data.SqlClient.SqlCommand"
   CLASS CLASS-SQLCOMMANDBUILDER AS "System.Data.SqlClient.SqlCommandBuilder"
   CLASS CLASS-SQLCONNECTION AS "System.Data.SqlClient.SqlConnection"
   CLASS CLASS-SQLDATAADAPTER AS "System.Data.SqlClient.SqlDataAdapter"
   CLASS CLASS-SQLPARAMETER AS "System.Data.SqlClient.SqlParameter"
   CLASS CLASS-SQLPARAMETERS AS "System.Data.SqlClient.SqlParameterCollection"
   ENUM ENUM-SQLDBTYPE AS "System.Data.SqlDbType"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SYSTEMCOLORS AS "System.Drawing.SystemColors"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-EXCEPTION AS "System.Exception"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-STRINGBUILDER AS "System.Text.StringBuilder"
   ENUM ENUM-ANCHORSTYLES AS "System.Windows.Forms.AnchorStyles"
   CLASS APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS ARRAY-CONTROL AS "System.Windows.Forms.Control[]"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   CLASS CLASS-DATAGRIDVIEW AS "System.Windows.Forms.DataGridView"
   CLASS CLASS-DATAGRIDVIEWCELL AS "System.Windows.Forms.DataGridViewCell"
   CLASS CLASS-DATAGRIDVIEWCOLUMN AS "System.Windows.Forms.DataGridViewColumn"
   CLASS CLASS-DATAGRIDVIEWCOLUMNS AS "System.Windows.Forms.DataGridViewColumnCollection"
   ENUM ENUM-DATAGRIDVIEWCOLUMNHEADERS AS
         "System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode"
   ENUM ENUM-CONTENTALIGNMENT AS "System.Windows.Forms.DataGridViewContentAlignment"
   CLASS CLASS-DATAGRIDVIEWROW AS "System.Windows.Forms.DataGridViewRow"
   CLASS CLASS-DATAGRIDVIEWSELECTEDROWS AS
         "System.Windows.Forms.DataGridViewSelectedRowCollection"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-MESSAGEBOX AS "System.Windows.Forms.MessageBox"
   CLASS CLASS-CONFIGURATIONMANAGER AS "System.Configuration.ConfigurationManager"
   CLASS CLASS-CONNECTIONSTRINGSETTINGS AS
         "System.Configuration.ConnectionStringSettings"
   ENUM ENUM-SCROLLBARS AS "System.Windows.Forms.ScrollBars"
   PROPERTY PROP-ALIGNMENT AS "Alignment"
   PROPERTY PROP-ANCHOR AS "Anchor"
   PROPERTY PROP-AUTOSCALEBASESIZE AS "AutoScaleBaseSize"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-BOTTOM AS "Bottom"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-BUTTON2 AS "button2"
   PROPERTY PROP-BUTTON3 AS "button3"
   PROPERTY PROP-BUTTON4 AS "button4"
   PROPERTY PROP-BUTTON5 AS "button5"
   PROPERTY PROP-CAPTIONTEXT AS "CaptionText"
   PROPERTY PROP-CELLS AS "Cells"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-COBOLSAMPLE AS "COBOLSample"
   PROPERTY PROP-COLUMNHEADERSHEIGHTSIZEMO AS "ColumnHeadersHeightSizeMode"
   PROPERTY PROP-COLUMNS AS "Columns"
   PROPERTY PROP-COMMANDTEXT AS "CommandText"
   PROPERTY PROP-CONNECTION AS "Connection"
   PROPERTY PROP-CONNECTIONSTRING AS "ConnectionString"
   PROPERTY PROP-CONTROLS AS "Controls"
```

```
          PROPERTY PROP-CONTROLTEXT AS "ControlText"
          PROPERTY PROP-COUNT AS "Count"
          PROPERTY PROP-CURRENTROW AS "CurrentRow"
          PROPERTY PROP-DATAGRID1 AS "dataGrid1"
          PROPERTY PROP-DATAMEMBER AS "DataMember"
          PROPERTY PROP-DATASOURCE AS "DataSource"
          PROPERTY PROP-DEFAULT AS "Default"
          PROPERTY PROP-DEFAULTCELLSTYLE AS "DefaultCellStyle"
          PROPERTY PROP-DELETECOMMAND AS "DeleteCommand"
          PROPERTY PROP-EMPTY AS "Empty"
          PROPERTY PROP-EMPTYSTRING AS "EmptyString"
          PROPERTY PROP-ENABLED AS "Enabled"
          PROPERTY PROP-FIREINFOMESSAGEEVENTONUSE AS "FireInfoMessageEventOnUserErrors"
          PROPERTY PROP-FORMATTEDVALUE AS "FormattedValue"
          PROPERTY PROP-GRIDCOLUMNSTYLES AS "GridColumnStyles"
          PROPERTY PROP-HEADERFORECOLOR AS "HeaderForeColor"
          PROPERTY PROP-HEADERTEXT AS "HeaderText"
          PROPERTY PROP-HEIGHT AS "Height"
          PROPERTY PROP-INDEX AS "Index"
          PROPERTY PROP-INSERTCOMMAND AS "InsertCommand"
          PROPERTY PROP-INT AS "Int"
          PROPERTY PROP-LABEL1 AS "label1"
          PROPERTY PROP-LABEL2 AS "label2"
          PROPERTY PROP-LABEL3 AS "label3"
          PROPERTY PROP-LABEL4 AS "label4"
          PROPERTY PROP-LABEL5 AS "label5"
          PROPERTY PROP-LEFT AS "Left"
          PROPERTY PROP-LENGTH AS "Length"
          PROPERTY PROP-LOCATION AS "Location"
          PROPERTY PROP-MAPPINGNAME AS "MappingName"
          PROPERTY PROP-MESSAGE AS "Message"
          PROPERTY PROP-MIDDLELEFT AS "MiddleLeft"
          PROPERTY PROP-MIDDLERIGHT AS "MiddleRight"
          PROPERTY PROP-NAME AS "Name"
          PROPERTY PROP-NEWROWINDEX AS "NewRowIndex"
          PROPERTY PROP-ORIGINAL AS "Original"
          PROPERTY PROP-PARAMETERS AS "Parameters"
          PROPERTY PROP-READONLY AS "ReadOnly"
          PROPERTY PROP-RIGHT AS "Right"
          PROPERTY PROP-ROWS AS "Rows"
          PROPERTY PROP-ROWTEMPLATE AS "RowTemplate"
          PROPERTY PROP-SCROLLBARS AS "ScrollBars"
          PROPERTY PROP-SELECTCOMMAND AS "SelectCommand"
          PROPERTY PROP-SELECTEDROWS AS "SelectedRows"
          PROPERTY PROP-SIZE AS "Size"
          PROPERTY PROP-SMALLINT AS "SmallInt"
          PROPERTY PROP-SOURCEVERSION AS "SourceVersion"
          PROPERTY PROP-TABINDEX AS "TabIndex"
          PROPERTY PROP-TABLES AS "Tables"
          PROPERTY PROP-TABLESTYLES AS "TableStyles"
          PROPERTY PROP-TEXT AS "Text"
          PROPERTY PROP-TOP AS "Top"
          PROPERTY PROP-UPDATECOMMAND AS "UpdateCommand"
          PROPERTY PROP-VALUE AS "Value"
          PROPERTY PROP-VARCHAR AS "VarChar"
          PROPERTY PROP-WIDTH AS "Width"
          PROPERTY PROP-CONNECTIONSTRINGS AS "ConnectionStrings".

       OBJECT.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 label1 OBJECT REFERENCE CLASS-LABEL.
       01 button1 OBJECT REFERENCE CLASS-BUTTON.
       01 button2 OBJECT REFERENCE CLASS-BUTTON.
       01 components OBJECT REFERENCE CONTAINER.
       01 button3 OBJECT REFERENCE CLASS-BUTTON.
       01 label2 OBJECT REFERENCE CLASS-LABEL.
       01 label3 OBJECT REFERENCE CLASS-LABEL.
       01 button4 OBJECT REFERENCE CLASS-BUTTON.
       01 label4 OBJECT REFERENCE CLASS-LABEL.
       01 sqlDataAdapter1 OBJECT REFERENCE CLASS-SQLDATAADAPTER.
       01 button5 OBJECT REFERENCE CLASS-BUTTON.
       01 label5 OBJECT REFERENCE CLASS-LABEL.
       01 dataTable1 OBJECT REFERENCE CLASS-DATATABLE.
       01 dataGrid1 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 sqlConnection1 OBJECT REFERENCE CLASS-SQLCONNECTION.

       PROCEDURE DIVISION.

       METHOD-ID. NEW.
       PROCEDURE DIVISION.
          INVOKE SELF "InitializeComponent".
```

```
      END METHOD NEW.

      METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      LINKAGE SECTION.
      01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
      PROCEDURE DIVISION USING BY VALUE disposing.
         IF disposing = B"1" THEN
           IF components NOT = NULL THEN
             INVOKE components "Dispose"
           END-IF
         END-IF.
         INVOKE SUPER "Dispose" USING BY VALUE disposing.
      END METHOD DISPOSE.

      METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 TEMP1 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP2 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP3 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP4 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP5 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP6 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP7 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP8 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP9 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP10 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP11 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
      01 TEMP12 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
      01 TEMP13 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
      01 TEMP14 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP15 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP16 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP17 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP18 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP19 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP20 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP21 OBJECT REFERENCE ENUM-ANCHORSTYLES.
      01 TEMP22 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP23 BINARY-LONG.
      01 TEMP24 BINARY-LONG.
      01 TEMP25 OBJECT REFERENCE CLASS-POINT.
      01 TEMP26 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP27 OBJECT REFERENCE CLASS-STRING.
      01 TEMP28 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP29 BINARY-LONG.
      01 TEMP30 BINARY-LONG.
      01 TEMP31 OBJECT REFERENCE CLASS-SIZE.
      01 TEMP32 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP33 BINARY-LONG.
      01 TEMP34 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP35 OBJECT REFERENCE CLASS-STRING.
      01 TEMP36 OBJECT REFERENCE CLASS-LABEL.
      01 TEMP37 BINARY-LONG.
      01 TEMP38 BINARY-LONG.
      01 TEMP39 OBJECT REFERENCE CLASS-POINT.
      01 TEMP40 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP41 OBJECT REFERENCE CLASS-STRING.
      01 TEMP42 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP43 BINARY-LONG.
      01 TEMP44 BINARY-LONG.
      01 TEMP45 OBJECT REFERENCE CLASS-SIZE.
      01 TEMP46 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP47 BINARY-LONG.
      01 TEMP48 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP49 OBJECT REFERENCE CLASS-STRING.
      01 TEMP50 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP51 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP52 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
      01 TEMP53 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP54 BINARY-LONG.
      01 TEMP55 BINARY-LONG.
      01 TEMP56 OBJECT REFERENCE CLASS-POINT.
      01 TEMP57 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP58 OBJECT REFERENCE CLASS-STRING.
      01 TEMP59 OBJECT REFERENCE CLASS-BUTTON.
      01 TEMP60 BINARY-LONG.
      01 TEMP61 BINARY-LONG.
      01 TEMP62 OBJECT REFERENCE CLASS-SIZE.
      01 TEMP63 OBJECT REFERENCE CLASS-BUTTON.
```

```
       01 TEMP64 BINARY-LONG.
       01 TEMP65 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP66 OBJECT REFERENCE CLASS-STRING.
       01 TEMP67 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP68 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP69 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP70 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP71 BINARY-LONG.
       01 TEMP72 BINARY-LONG.
       01 TEMP73 OBJECT REFERENCE CLASS-POINT.
       01 TEMP74 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP75 OBJECT REFERENCE CLASS-STRING.
       01 TEMP76 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP77 BINARY-LONG.
       01 TEMP78 BINARY-LONG.
       01 TEMP79 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP80 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP81 BINARY-LONG.
       01 TEMP82 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP83 OBJECT REFERENCE CLASS-STRING.
       01 TEMP84 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP85 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP86 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP87 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP88 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP89 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP90 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP91 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP92 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP93 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP94 BINARY-LONG.
       01 TEMP95 BINARY-LONG.
       01 TEMP96 OBJECT REFERENCE CLASS-POINT.
       01 TEMP97 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP98 OBJECT REFERENCE CLASS-STRING.
       01 TEMP99 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP100 BINARY-LONG.
       01 TEMP101 BINARY-LONG.
       01 TEMP102 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP103 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP104 BINARY-LONG.
       01 TEMP105 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP106 OBJECT REFERENCE CLASS-STRING.
       01 TEMP107 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP108 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP109 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP110 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP111 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP112 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP113 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP114 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP115 BINARY-LONG.
       01 TEMP116 BINARY-LONG.
       01 TEMP117 OBJECT REFERENCE CLASS-POINT.
       01 TEMP118 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP119 OBJECT REFERENCE CLASS-STRING.
       01 TEMP120 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP121 BINARY-LONG.
       01 TEMP122 BINARY-LONG.
       01 TEMP123 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP124 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP125 BINARY-LONG.
       01 TEMP126 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP127 OBJECT REFERENCE CLASS-STRING.
       01 TEMP128 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP129 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP130 BINARY-LONG.
       01 TEMP131 BINARY-LONG.
       01 TEMP132 OBJECT REFERENCE CLASS-POINT.
       01 TEMP133 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP134 OBJECT REFERENCE CLASS-STRING.
       01 TEMP135 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP136 BINARY-LONG.
       01 TEMP137 BINARY-LONG.
       01 TEMP138 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP139 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP140 BINARY-LONG.
       01 TEMP141 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP142 OBJECT REFERENCE CLASS-STRING.
       01 TEMP143 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP144 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP145 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
```

```
       01 TEMP146 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP147 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP148 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP149 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP150 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP151 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP152 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP153 BINARY-LONG.
       01 TEMP154 BINARY-LONG.
       01 TEMP155 OBJECT REFERENCE CLASS-POINT.
       01 TEMP156 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP157 OBJECT REFERENCE CLASS-STRING.
       01 TEMP158 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP159 BINARY-LONG.
       01 TEMP160 BINARY-LONG.
       01 TEMP161 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP162 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP163 BINARY-LONG.
       01 TEMP164 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP165 OBJECT REFERENCE CLASS-STRING.
       01 TEMP166 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP167 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP168 BINARY-LONG.
       01 TEMP169 BINARY-LONG.
       01 TEMP170 OBJECT REFERENCE CLASS-POINT.
       01 TEMP171 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP172 OBJECT REFERENCE CLASS-STRING.
       01 TEMP173 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP174 BINARY-LONG.
       01 TEMP175 BINARY-LONG.
       01 TEMP176 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP177 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP178 BINARY-LONG.
       01 TEMP179 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP180 OBJECT REFERENCE CLASS-STRING.
       01 TEMP181 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP182 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP183 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP184 BINARY-LONG.
       01 TEMP185 BINARY-LONG.
       01 TEMP186 OBJECT REFERENCE CLASS-POINT.
       01 TEMP187 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP188 OBJECT REFERENCE CLASS-STRING.
       01 TEMP189 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP190 BINARY-LONG.
       01 TEMP191 BINARY-LONG.
       01 TEMP192 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP193 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP194 BINARY-LONG.
       01 TEMP195 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP196 OBJECT REFERENCE CLASS-STRINGBUILDER.
       01 TEMP197 OBJECT REFERENCE CLASS-STRING.
       01 TEMP198 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP199 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP200 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP201 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP202 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP203 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP204 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP205 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP206 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP207 BINARY-LONG.
       01 TEMP208 BINARY-LONG.
       01 TEMP209 OBJECT REFERENCE CLASS-POINT.
       01 TEMP210 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP211 OBJECT REFERENCE CLASS-STRING.
       01 TEMP212 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP213 BINARY-LONG.
       01 TEMP214 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP215 OBJECT REFERENCE CLASS-DATAGRIDVIEWROW.
       01 TEMP216 BINARY-LONG.
       01 TEMP217 BINARY-LONG.
       01 TEMP218 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP219 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP220 BINARY-LONG.
       01 TEMP221 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP222 BINARY-LONG.
       01 TEMP223 BINARY-LONG.
       01 TEMP224 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP225 BINARY-LONG.
       01 TEMP226 BINARY-LONG.
       01 TEMP227 OBJECT REFERENCE CLASS-SIZE.
```

```
       01 TEMP228 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP229 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP230 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP231 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP232 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP233 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP234 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP235 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP236 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP237 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP238 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP239 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP240 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP241 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP242 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP243 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP244 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP245 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP246 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP247 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP248 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP249 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP250 OBJECT REFERENCE CLASS-STRING.
       01 TEMP251 OBJECT REFERENCE CLASS-STRING.
       01 TEMP252 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP253 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP254 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Button" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Button" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Label" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.DataGridView" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
```

```
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="BeginInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
```

```
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Left" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
```

```
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
```

```
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="120" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="10" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="280" />
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="サンプルデータベースの STOCK テーブルの内容をデータグリッドに表示します。" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button1" />
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="16" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="10" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="button1" />
*</prop>
*</object>
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="88" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="0" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="ロード" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="button1_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
```

```
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Enabled" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="220" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
```

```
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="100" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="更新" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button2_Click" />
      *</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Enabled" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
```

```
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="260" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
*<int32 value="100" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="5" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="削除" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button3" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
```

```
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button3_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Left" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="220" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
```

```
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="280" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
```

```
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="データグリッドで変更された内容をサンプルデータベースに反映します。" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Anchor" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeCastExpression">
*<prop name="TargetType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Expression">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
```

```
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Right" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Left" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Top" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="260" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
```

```
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="280" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="6" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="選択した行を削除します。" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
```

```
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="button4" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button4" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Enabled" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="False" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button4" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
```

```
      *<int32 value="10" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="300" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="100" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="7" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="リセット" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
```

```
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button4_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
```

```
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Right" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Left" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Top" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
*</prop>
*<prop name="Operator">
*<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
```

```
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="300" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label4" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
```

```
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="280" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label4" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="8" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label4" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="STOCK テーブルの内容をリセットします。" />
*</prop>
*</object>
*</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Enabled" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
```

```
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="340" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="100" />
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button5" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="9" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button5" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="オーダー表" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button5" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
```

```
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="button5_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label5" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label5" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="340" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="280" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="30" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label5" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="別フォームで、ORDERS、STOCK および COMPANY テーブルから在庫製品別のオーダー覧を作成します。" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
```

```
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Anchor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Right" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeBinaryOperatorExpression">
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Left" />
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ColumnHeadersHeightSizeMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="46" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="RowTemplate" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Height" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="21" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="390" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="155" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
```

```
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="MainForm" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="AutoScaleBaseSize" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="5" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="412" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="383" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
```

```
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label5" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button5" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
```

```
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="label4" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="MainForm" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="在庫表" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="EventName">
*<string value="Load" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="MainForm_Load" />
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="EndInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP1
          SET PROP-LABEL1 OF SELF TO TEMP1
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP2
          SET PROP-BUTTON1 OF SELF TO TEMP2
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP3
          SET PROP-BUTTON2 OF SELF TO TEMP3
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP4
          SET PROP-BUTTON3 OF SELF TO TEMP4
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP5
          SET PROP-LABEL2 OF SELF TO TEMP5
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP6
          SET PROP-LABEL3 OF SELF TO TEMP6
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP7
          SET PROP-BUTTON4 OF SELF TO TEMP7
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP8
          SET PROP-LABEL4 OF SELF TO TEMP8
```

```
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP9
          SET PROP-BUTTON5 OF SELF TO TEMP9
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP10
          SET PROP-LABEL5 OF SELF TO TEMP10
          INVOKE CLASS-DATAGRIDVIEW "NEW" RETURNING TEMP11
          SET PROP-DATAGRID1 OF SELF TO TEMP11
          SET TEMP12 TO PROP-DATAGRID1 OF SELF
          SET TEMP13 TO TEMP12 AS INTERFACE-ISUPPORTINITIALIZE
          INVOKE TEMP13 "BeginInit"
          INVOKE SELF "SuspendLayout"
      *
      *label1
      *
          SET TEMP14 TO PROP-TOP OF ENUM-ANCHORSTYLES
          SET TEMP15 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
          SET TEMP16 TO FUNCTION ENUM-OR(TEMP14 TEMP15)
          SET TEMP17 TO PROP-LEFT OF ENUM-ANCHORSTYLES
          SET TEMP18 TO FUNCTION ENUM-OR(TEMP16 TEMP17)
          SET TEMP19 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
          SET TEMP20 TO FUNCTION ENUM-OR(TEMP18 TEMP19)
          SET TEMP21 TO TEMP20
          SET TEMP22 TO PROP-LABEL1 OF SELF
          SET PROP-ANCHOR OF TEMP22 TO TEMP21
          MOVE 120 TO TEMP23
          MOVE 10 TO TEMP24
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP23 BY VALUE TEMP24 RETURNING TEMP25
          SET TEMP26 TO PROP-LABEL1 OF SELF
          SET PROP-LOCATION OF TEMP26 TO TEMP25
          SET TEMP27 TO N"label1"
          SET TEMP28 TO PROP-LABEL1 OF SELF
          SET PROP-NAME OF TEMP28 TO TEMP27
          MOVE 280 TO TEMP29
          MOVE 30 TO TEMP30
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP29 BY VALUE TEMP30 RETURNING TEMP31
          SET TEMP32 TO PROP-LABEL1 OF SELF
          SET PROP-SIZE OF TEMP32 TO TEMP31
          MOVE 1 TO TEMP33
          SET TEMP34 TO PROP-LABEL1 OF SELF
          MOVE TEMP33 TO PROP-TABINDEX OF TEMP34
          SET TEMP35 TO N"サンプルデータベースの STOCK テーブルの内容をデータグリッドに表示します。"
          SET TEMP36 TO PROP-LABEL1 OF SELF
          SET PROP-TEXT OF TEMP36 TO TEMP35
      *
      *button1
      *
          MOVE 16 TO TEMP37
          MOVE 10 TO TEMP38
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP37 BY VALUE TEMP38 RETURNING TEMP39
          SET TEMP40 TO PROP-BUTTON1 OF SELF
          SET PROP-LOCATION OF TEMP40 TO TEMP39
          SET TEMP41 TO N"button1"
          SET TEMP42 TO PROP-BUTTON1 OF SELF
          SET PROP-NAME OF TEMP42 TO TEMP41
          MOVE 88 TO TEMP43
          MOVE 30 TO TEMP44
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP43 BY VALUE TEMP44 RETURNING TEMP45
          SET TEMP46 TO PROP-BUTTON1 OF SELF
          SET PROP-SIZE OF TEMP46 TO TEMP45
          MOVE 0 TO TEMP47
          SET TEMP48 TO PROP-BUTTON1 OF SELF
          MOVE TEMP47 TO PROP-TABINDEX OF TEMP48
          SET TEMP49 TO N"ロード"
          SET TEMP50 TO PROP-BUTTON1 OF SELF
          SET PROP-TEXT OF TEMP50 TO TEMP49
          SET TEMP51 TO PROP-BUTTON1 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click"
                 RETURNING TEMP52
          INVOKE TEMP51 "add_Click" USING BY VALUE TEMP52
      *
      *button2
      *
          SET TEMP53 TO PROP-BUTTON2 OF SELF
          SET PROP-ENABLED OF TEMP53 TO B"0"
          MOVE 10 TO TEMP54
          MOVE 220 TO TEMP55
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP54 BY VALUE TEMP55 RETURNING TEMP56
          SET TEMP57 TO PROP-BUTTON2 OF SELF
          SET PROP-LOCATION OF TEMP57 TO TEMP56
          SET TEMP58 TO N"button2"
          SET TEMP59 TO PROP-BUTTON2 OF SELF
          SET PROP-NAME OF TEMP59 TO TEMP58
```

```
          MOVE 100 TO TEMP60
          MOVE 30 TO TEMP61
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP60 BY VALUE TEMP61 RETURNING TEMP62
          SET TEMP63 TO PROP-BUTTON2 OF SELF
          SET PROP-SIZE OF TEMP63 TO TEMP62
          MOVE 3 TO TEMP64
          SET TEMP65 TO PROP-BUTTON2 OF SELF
          MOVE TEMP64 TO PROP-TABINDEX OF TEMP65
          SET TEMP66 TO N"更新"
          SET TEMP67 TO PROP-BUTTON2 OF SELF
          SET PROP-TEXT OF TEMP67 TO TEMP66
          SET TEMP68 TO PROP-BUTTON2 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button2_Click"
                 RETURNING TEMP69
          INVOKE TEMP68 "add_Click" USING BY VALUE TEMP69
      *
      *button3
      *
          SET TEMP70 TO PROP-BUTTON3 OF SELF
          SET PROP-ENABLED OF TEMP70 TO B"0"
          MOVE 10 TO TEMP71
          MOVE 260 TO TEMP72
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP71 BY VALUE TEMP72 RETURNING TEMP73
          SET TEMP74 TO PROP-BUTTON3 OF SELF
          SET PROP-LOCATION OF TEMP74 TO TEMP73
          SET TEMP75 TO N"button3"
          SET TEMP76 TO PROP-BUTTON3 OF SELF
          SET PROP-NAME OF TEMP76 TO TEMP75
          MOVE 100 TO TEMP77
          MOVE 30 TO TEMP78
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP77 BY VALUE TEMP78 RETURNING TEMP79
          SET TEMP80 TO PROP-BUTTON3 OF SELF
          SET PROP-SIZE OF TEMP80 TO TEMP79
          MOVE 5 TO TEMP81
          SET TEMP82 TO PROP-BUTTON3 OF SELF
          MOVE TEMP81 TO PROP-TABINDEX OF TEMP82
          SET TEMP83 TO N"削除"
          SET TEMP84 TO PROP-BUTTON3 OF SELF
          SET PROP-TEXT OF TEMP84 TO TEMP83
          SET TEMP85 TO PROP-BUTTON3 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button3_Click"
                 RETURNING TEMP86
          INVOKE TEMP85 "add_Click" USING BY VALUE TEMP86
      *
      *label2
      *
          SET TEMP87 TO PROP-TOP OF ENUM-ANCHORSTYLES
          SET TEMP88 TO PROP-LEFT OF ENUM-ANCHORSTYLES
          SET TEMP89 TO FUNCTION ENUM-OR(TEMP87 TEMP88)
          SET TEMP90 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
          SET TEMP91 TO FUNCTION ENUM-OR(TEMP89 TEMP90)
          SET TEMP92 TO TEMP91
          SET TEMP93 TO PROP-LABEL2 OF SELF
          SET PROP-ANCHOR OF TEMP93 TO TEMP92
          MOVE 120 TO TEMP94
          MOVE 220 TO TEMP95
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP94 BY VALUE TEMP95 RETURNING TEMP96
          SET TEMP97 TO PROP-LABEL2 OF SELF
          SET PROP-LOCATION OF TEMP97 TO TEMP96
          SET TEMP98 TO N"label2"
          SET TEMP99 TO PROP-LABEL2 OF SELF
          SET PROP-NAME OF TEMP99 TO TEMP98
          MOVE 280 TO TEMP100
          MOVE 30 TO TEMP101
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP100 BY VALUE TEMP101 RETURNING TEMP102
          SET TEMP103 TO PROP-LABEL2 OF SELF
          SET PROP-SIZE OF TEMP103 TO TEMP102
          MOVE 4 TO TEMP104
          SET TEMP105 TO PROP-LABEL2 OF SELF
          MOVE TEMP104 TO PROP-TABINDEX OF TEMP105
          SET TEMP106 TO N"データグリッドで変更された内容をサンプルデータベースに反映します。"
          SET TEMP107 TO PROP-LABEL2 OF SELF
          SET PROP-TEXT OF TEMP107 TO TEMP106
      *
      *label3
      *
          SET TEMP108 TO PROP-TOP OF ENUM-ANCHORSTYLES
          SET TEMP109 TO PROP-LEFT OF ENUM-ANCHORSTYLES
          SET TEMP110 TO FUNCTION ENUM-OR(TEMP108 TEMP109)
          SET TEMP111 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
          SET TEMP112 TO FUNCTION ENUM-OR(TEMP110 TEMP111)
```

```
        SET TEMP113 TO TEMP112
        SET TEMP114 TO PROP-LABEL3 OF SELF
        SET PROP-ANCHOR OF TEMP114 TO TEMP113
        MOVE 120 TO TEMP115
        MOVE 260 TO TEMP116
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP115 BY VALUE TEMP116 RETURNING TEMP117
        SET TEMP118 TO PROP-LABEL3 OF SELF
        SET PROP-LOCATION OF TEMP118 TO TEMP117
        SET TEMP119 TO N"label3"
        SET TEMP120 TO PROP-LABEL3 OF SELF
        SET PROP-NAME OF TEMP120 TO TEMP119
        MOVE 280 TO TEMP121
        MOVE 30 TO TEMP122
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP121 BY VALUE TEMP122 RETURNING TEMP123
        SET TEMP124 TO PROP-LABEL3 OF SELF
        SET PROP-SIZE OF TEMP124 TO TEMP123
        MOVE 6 TO TEMP125
        SET TEMP126 TO PROP-LABEL3 OF SELF
        MOVE TEMP125 TO PROP-TABINDEX OF TEMP126
        SET TEMP127 TO N"選択した行を削除します。"
        SET TEMP128 TO PROP-LABEL3 OF SELF
        SET PROP-TEXT OF TEMP128 TO TEMP127
    *
    *button4
    *
        SET TEMP129 TO PROP-BUTTON4 OF SELF
        SET PROP-ENABLED OF TEMP129 TO B"0"
        MOVE 10 TO TEMP130
        MOVE 300 TO TEMP131
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP130 BY VALUE TEMP131 RETURNING TEMP132
        SET TEMP133 TO PROP-BUTTON4 OF SELF
        SET PROP-LOCATION OF TEMP133 TO TEMP132
        SET TEMP134 TO N"button4"
        SET TEMP135 TO PROP-BUTTON4 OF SELF
        SET PROP-NAME OF TEMP135 TO TEMP134
        MOVE 100 TO TEMP136
        MOVE 30 TO TEMP137
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP136 BY VALUE TEMP137 RETURNING TEMP138
        SET TEMP139 TO PROP-BUTTON4 OF SELF
        SET PROP-SIZE OF TEMP139 TO TEMP138
        MOVE 7 TO TEMP140
        SET TEMP141 TO PROP-BUTTON4 OF SELF
        MOVE TEMP140 TO PROP-TABINDEX OF TEMP141
        SET TEMP142 TO N"リセット"
        SET TEMP143 TO PROP-BUTTON4 OF SELF
        SET PROP-TEXT OF TEMP143 TO TEMP142
        SET TEMP144 TO PROP-BUTTON4 OF SELF
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button4_Click"
               RETURNING TEMP145
        INVOKE TEMP144 "add_Click" USING BY VALUE TEMP145
    *
    *label4
    *
        SET TEMP146 TO PROP-TOP OF ENUM-ANCHORSTYLES
        SET TEMP147 TO PROP-LEFT OF ENUM-ANCHORSTYLES
        SET TEMP148 TO FUNCTION ENUM-OR(TEMP146 TEMP147)
        SET TEMP149 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP150 TO FUNCTION ENUM-OR(TEMP148 TEMP149)
        SET TEMP151 TO TEMP150
        SET TEMP152 TO PROP-LABEL4 OF SELF
        SET PROP-ANCHOR OF TEMP152 TO TEMP151
        MOVE 120 TO TEMP153
        MOVE 300 TO TEMP154
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP153 BY VALUE TEMP154 RETURNING TEMP155
        SET TEMP156 TO PROP-LABEL4 OF SELF
        SET PROP-LOCATION OF TEMP156 TO TEMP155
        SET TEMP157 TO N"label4"
        SET TEMP158 TO PROP-LABEL4 OF SELF
        SET PROP-NAME OF TEMP158 TO TEMP157
        MOVE 280 TO TEMP159
        MOVE 30 TO TEMP160
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP159 BY VALUE TEMP160 RETURNING TEMP161
        SET TEMP162 TO PROP-LABEL4 OF SELF
        SET PROP-SIZE OF TEMP162 TO TEMP161
        MOVE 8 TO TEMP163
        SET TEMP164 TO PROP-LABEL4 OF SELF
        MOVE TEMP163 TO PROP-TABINDEX OF TEMP164
        SET TEMP165 TO N"STOCK テーブルの内容をリセットします。"
        SET TEMP166 TO PROP-LABEL4 OF SELF
        SET PROP-TEXT OF TEMP166 TO TEMP165
    *
```

```
      *button5
      *
          SET TEMP167 TO PROP-BUTTON5 OF SELF
          SET PROP-ENABLED OF TEMP167 TO B"0"
          MOVE 10 TO TEMP168
          MOVE 340 TO TEMP169
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP168 BY VALUE TEMP169 RETURNING TEMP170
          SET TEMP171 TO PROP-BUTTON5 OF SELF
          SET PROP-LOCATION OF TEMP171 TO TEMP170
          SET TEMP172 TO N"button5"
          SET TEMP173 TO PROP-BUTTON5 OF SELF
          SET PROP-NAME OF TEMP173 TO TEMP172
          MOVE 100 TO TEMP174
          MOVE 30 TO TEMP175
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP174 BY VALUE TEMP175 RETURNING TEMP176
          SET TEMP177 TO PROP-BUTTON5 OF SELF
          SET PROP-SIZE OF TEMP177 TO TEMP176
          MOVE 9 TO TEMP178
          SET TEMP179 TO PROP-BUTTON5 OF SELF
          MOVE TEMP178 TO PROP-TABINDEX OF TEMP179
          SET TEMP180 TO N"オーダー表"
          SET TEMP181 TO PROP-BUTTON5 OF SELF
          SET PROP-TEXT OF TEMP181 TO TEMP180
          SET TEMP182 TO PROP-BUTTON5 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button5_Click"
                 RETURNING TEMP183
          INVOKE TEMP182 "add_Click" USING BY VALUE TEMP183
      *
      *label5
      *
          MOVE 120 TO TEMP184
          MOVE 340 TO TEMP185
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP184 BY VALUE TEMP185 RETURNING TEMP186
          SET TEMP187 TO PROP-LABEL5 OF SELF
          SET PROP-LOCATION OF TEMP187 TO TEMP186
          SET TEMP188 TO N"label5"
          SET TEMP189 TO PROP-LABEL5 OF SELF
          SET PROP-NAME OF TEMP189 TO TEMP188
          MOVE 280 TO TEMP190
          MOVE 30 TO TEMP191
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP190 BY VALUE TEMP191 RETURNING TEMP192
          SET TEMP193 TO PROP-LABEL5 OF SELF
          SET PROP-SIZE OF TEMP193 TO TEMP192
          MOVE 10 TO TEMP194
          SET TEMP195 TO PROP-LABEL5 OF SELF
          MOVE TEMP194 TO PROP-TABINDEX OF TEMP195
          INVOKE CLASS-STRINGBUILDER "NEW" RETURNING TEMP196
          INVOKE TEMP196 "Append" USING
          BY VALUE N"別フォームで、ORDERS、STOCK および COMPANY テーブルから在庫製品別のオーダー一覧を作成し"
          INVOKE TEMP196 "Append" USING BY VALUE N"ます。"
          INVOKE TEMP196 "ToString" RETURNING TEMP197
          SET TEMP198 TO PROP-LABEL5 OF SELF
          SET PROP-TEXT OF TEMP198 TO TEMP197
      *
      *dataGrid1
      *
          SET TEMP199 TO PROP-TOP OF ENUM-ANCHORSTYLES
          SET TEMP200 TO PROP-LEFT OF ENUM-ANCHORSTYLES
          SET TEMP201 TO FUNCTION ENUM-OR(TEMP199 TEMP200)
          SET TEMP202 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
          SET TEMP203 TO FUNCTION ENUM-OR(TEMP201 TEMP202)
          SET TEMP204 TO TEMP203
          SET TEMP205 TO PROP-DATAGRID1 OF SELF
          SET PROP-ANCHOR OF TEMP205 TO TEMP204
          SET TEMP206 TO PROP-DATAGRID1 OF SELF
          SET PROP-COLUMNHEADERSHEIGHTSIZEMO OF TEMP206 TO
              PROP-AUTOSIZE OF ENUM-DATAGRIDVIEWCOLUMNHEADERS
          MOVE 10 TO TEMP207
          MOVE 46 TO TEMP208
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP207 BY VALUE TEMP208 RETURNING TEMP209
          SET TEMP210 TO PROP-DATAGRID1 OF SELF
          SET PROP-LOCATION OF TEMP210 TO TEMP209
          SET TEMP211 TO N"dataGrid1"
          SET TEMP212 TO PROP-DATAGRID1 OF SELF
          SET PROP-NAME OF TEMP212 TO TEMP211
          MOVE 21 TO TEMP213
          SET TEMP214 TO PROP-DATAGRID1 OF SELF
          SET TEMP215 TO PROP-ROWTEMPLATE OF TEMP214
          MOVE TEMP213 TO PROP-HEIGHT OF TEMP215
          MOVE 390 TO TEMP216
          MOVE 155 TO TEMP217
```

```
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP216 BY VALUE TEMP217 RETURNING TEMP218
        SET TEMP219 TO PROP-DATAGRID1 OF SELF
        SET PROP-SIZE OF TEMP219 TO TEMP218
        MOVE 2 TO TEMP220
        SET TEMP221 TO PROP-DATAGRID1 OF SELF
        MOVE TEMP220 TO PROP-TABINDEX OF TEMP221
    *
    *MainForm
    *
        MOVE 5 TO TEMP222
        MOVE 12 TO TEMP223
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP222 BY VALUE TEMP223 RETURNING TEMP224
        SET PROP-AUTOSCALEBASESIZE OF SELF TO TEMP224
        MOVE 412 TO TEMP225
        MOVE 383 TO TEMP226
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP225 BY VALUE TEMP226 RETURNING TEMP227
        SET PROP-CLIENTSIZE OF SELF TO TEMP227
        SET TEMP229 TO PROP-DATAGRID1 OF SELF
        SET TEMP228 TO PROP-CONTROLS OF SELF
        INVOKE TEMP228 "Add" USING BY VALUE TEMP229
        SET TEMP231 TO PROP-LABEL5 OF SELF
        SET TEMP230 TO PROP-CONTROLS OF SELF
        INVOKE TEMP230 "Add" USING BY VALUE TEMP231
        SET TEMP233 TO PROP-BUTTON5 OF SELF
        SET TEMP232 TO PROP-CONTROLS OF SELF
        INVOKE TEMP232 "Add" USING BY VALUE TEMP233
        SET TEMP235 TO PROP-BUTTON2 OF SELF
        SET TEMP234 TO PROP-CONTROLS OF SELF
        INVOKE TEMP234 "Add" USING BY VALUE TEMP235
        SET TEMP237 TO PROP-BUTTON1 OF SELF
        SET TEMP236 TO PROP-CONTROLS OF SELF
        INVOKE TEMP236 "Add" USING BY VALUE TEMP237
        SET TEMP239 TO PROP-LABEL1 OF SELF
        SET TEMP238 TO PROP-CONTROLS OF SELF
        INVOKE TEMP238 "Add" USING BY VALUE TEMP239
        SET TEMP241 TO PROP-BUTTON3 OF SELF
        SET TEMP240 TO PROP-CONTROLS OF SELF
        INVOKE TEMP240 "Add" USING BY VALUE TEMP241
        SET TEMP243 TO PROP-LABEL2 OF SELF
        SET TEMP242 TO PROP-CONTROLS OF SELF
        INVOKE TEMP242 "Add" USING BY VALUE TEMP243
        SET TEMP245 TO PROP-LABEL3 OF SELF
        SET TEMP244 TO PROP-CONTROLS OF SELF
        INVOKE TEMP244 "Add" USING BY VALUE TEMP245
        SET TEMP247 TO PROP-BUTTON4 OF SELF
        SET TEMP246 TO PROP-CONTROLS OF SELF
        INVOKE TEMP246 "Add" USING BY VALUE TEMP247
        SET TEMP249 TO PROP-LABEL4 OF SELF
        SET TEMP248 TO PROP-CONTROLS OF SELF
        INVOKE TEMP248 "Add" USING BY VALUE TEMP249
        SET TEMP250 TO N"MainForm"
        SET PROP-NAME OF SELF TO TEMP250
        SET TEMP251 TO N"在庫表"
        SET PROP-TEXT OF SELF TO TEMP251
        INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"MainForm_Load"
                RETURNING TEMP252
        INVOKE SELF "add_Load" USING BY VALUE TEMP252
        SET TEMP253 TO PROP-DATAGRID1 OF SELF
        SET TEMP254 TO TEMP253 AS INTERFACE-ISUPPORTINITIALIZE
        INVOKE TEMP254 "EndInit"
        INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
    END METHOD INITIALIZECOMPONENT.

    METHOD-ID. MainForm_Load PRIVATE.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01 connectionStringSettings OBJECT REFERENCE CLASS-CONNECTIONSTRINGSETTINGS.
    01 connectionString OBJECT REFERENCE CLASS-STRING.
    LINKAGE SECTION.
    01 sender OBJECT REFERENCE CLASS-OBJECT.
    01 e OBJECT REFERENCE CLASS-EVENTARGS.
    PROCEDURE DIVISION USING BY VALUE sender e.
    * アプリケーション構成ファイルから接続文字列を取得する
        INVOKE PROP-CONNECTIONSTRINGS OF CLASS-CONFIGURATIONMANAGER "get_Item"
               USING N"COBOLSample" RETURNING connectionStringSettings.
        SET connectionString TO PROP-CONNECTIONSTRING OF connectionStringSettings.
    * データベースへの接続
        SET sqlConnection1 TO CLASS-SQLCONNECTION::"NEW"(connectionString).
        INVOKE sqlConnection1 "Open".

    END METHOD MainForm_Load.
```

```
      METHOD-ID. button1_Click PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 tmpMessage OBJECT REFERENCE CLASS-STRING.
      01 tmpGridColumns OBJECT REFERENCE CLASS-DATAGRIDVIEWCOLUMNS.
      01 tmpGridColumn OBJECT REFERENCE CLASS-DATAGRIDVIEWCOLUMN.
      01 excep OBJECT REFERENCE CLASS-EXCEPTION.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.
          TRY
      *  DataAdapter を作成します。
            IF sqlDataAdapter1 = NULL
               SET sqlDataAdapter1 TO CLASS-SQLDATAADAPTER :: "NEW"
            END-IF

      *  データテーブルを作成します。
            IF dataTable1 = NULL
               SET dataTable1 TO CLASS-DATATABLE :: "NEW"
            END-IF

      *  表示するデータテーブルを作成します。
            INVOKE SELF "List_Stocks"

      *  データグリッドにバインドします
            SET PROP-DATASOURCE OF dataGrid1 TO dataTable1

      *  データグリッドのヘッダをカスタマイズします
            SET tmpGridColumns TO PROP-COLUMNS OF dataGrid1
            IF tmpGridColumns NOT = NULL

               SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"GNO")
               SET PROP-HEADERTEXT OF tmpGridColumn TO N"製品番号"
               SET PROP-READONLY OF tmpGridColumn TO B"1"
               SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                   PROP-MIDDLERIGHT OF ENUM-CONTENTALIGNMENT
               SET PROP-WIDTH OF tmpGridColumn TO 80

               SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"GOODS")
               SET PROP-HEADERTEXT OF tmpGridColumn TO N"製品名"
               SET PROP-READONLY OF tmpGridColumn TO B"1"
               SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                   PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
               SET PROP-WIDTH OF tmpGridColumn TO 110

               SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"QOH")
               SET PROP-HEADERTEXT OF tmpGridColumn TO N"在庫数量"
               SET PROP-READONLY OF tmpGridColumn TO B"0"
               SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                   PROP-MIDDLERIGHT OF ENUM-CONTENTALIGNMENT
               SET PROP-WIDTH OF tmpGridColumn TO 80

               SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"WHNO")
               SET PROP-HEADERTEXT OF tmpGridColumn TO N"倉庫番号"
               SET PROP-READONLY OF tmpGridColumn TO B"1"
               SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                   PROP-MIDDLERIGHT OF ENUM-CONTENTALIGNMENT
               SET PROP-WIDTH OF tmpGridColumn TO 80
            END-IF

      *  各ボタンを有効にします。
            SET PROP-ENABLED OF button2 TO B"1"
            SET PROP-ENABLED OF button3 TO B"1"
            SET PROP-ENABLED OF button4 TO B"1"
            SET PROP-ENABLED OF button5 TO B"1"

          CATCH excep
      *  例外を捕捉してメッセージボックスを表示します。
            SET tmpMessage TO PROP-MESSAGE OF excep
            INVOKE CLASS-MESSAGEBOX "Show" USING tmpMessage
          END-TRY.

      END METHOD button1_Click.

      METHOD-ID. button2_Click PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 tmpMessage OBJECT REFERENCE CLASS-STRING.
```

```
       01 excep OBJECT REFERENCE CLASS-EXCEPTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           TRY
      *  データグリッドによる変更をデータベースに反映します。
              INVOKE sqlDataAdapter1 "Update" USING dataTable1
           CATCH excep
      *  例外を捕捉してメッセージボックスを表示します。
              SET tmpMessage TO PROP-MESSAGE OF excep
              INVOKE CLASS-MESSAGEBOX "Show" USING tmpMessage
           END-TRY.

       END METHOD button2_Click.

       METHOD-ID. button3_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 selCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 delCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 tmpParameters OBJECT REFERENCE CLASS-SQLPARAMETERS.
       01 tmpParameter OBJECT REFERENCE CLASS-SQLPARAMETER.
       01 tmpDbType OBJECT REFERENCE ENUM-SQLDBTYPE.
       01 tmpRow OBJECT REFERENCE CLASS-DATAROW.
       01 tmpText OBJECT REFERENCE CLASS-STRING.
       01 tmpMessage OBJECT REFERENCE CLASS-STRING.
       01 excep OBJECT REFERENCE CLASS-EXCEPTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           TRY
      *  削除処理のためコマンドを再構成します。
              SET tmpText TO SELF :: "Get_SelectList"
              IF tmpText = NULL OR PROP-LENGTH OF tmpText = 0
                 EXIT METHOD
              END-IF
              INVOKE dataTable1 "Clear"
              SET selCommand TO PROP-SELECTCOMMAND OF sqlDataAdapter1
              SET PROP-COMMANDTEXT OF selCommand TO CLASS-STRING :: "Format"
                 (N"SELECT GNO, GOODS FROM STOCK WHERE GNO IN ({0});", tmpText)

      *  削除用のコマンドを作成します
              SET delCommand TO CLASS-SQLCOMMAND :: "NEW"
                        (N"DELETE FROM STOCK WHERE GNO = @GNO AND GOODS = @GOODS;", sqlConnection1)
              SET tmpParameters TO PROP-PARAMETERS OF delCommand
              SET tmpDbType TO PROP-VARCHAR OF ENUM-SQLDBTYPE
              SET tmpParameter  TO tmpParameters :: "Add" (N"@GOODS", tmpDbType, 20, N"GOODS")
              SET PROP-SOURCEVERSION OF tmpParameter TO PROP-ORIGINAL OF ENUM-DATAROWVERSION

              SET tmpDbType TO PROP-SMALLINT OF ENUM-SQLDBTYPE
              SET tmpParameter  TO tmpParameters :: "Add" (N"@GNO", tmpDbType, 2, N"GNO")
              SET PROP-SOURCEVERSION OF tmpParameter TO PROP-ORIGINAL OF ENUM-DATAROWVERSION

              SET PROP-DELETECOMMAND OF sqlDataAdapter1 TO delCommand
              INVOKE sqlDataAdapter1 "Fill" USING dataTable1

      *  行コレクションを取り出して行を削除します
              PERFORM VARYING tmpRow THRU PROP-ROWS OF dataTable1
                 INVOKE tmpRow "Delete"
              END-PERFORM

      *  データテーブルを更新します。
              INVOKE sqlDataAdapter1 "Update" USING dataTable1

      *  表示するデータテーブルを更新します。
              INVOKE SELF "List_Stocks"

           CATCH excep
      *  例外を捕捉してメッセージボックスを表示します。
              SET tmpMessage TO PROP-MESSAGE OF excep
              INVOKE CLASS-MESSAGEBOX "Show" USING tmpMessage
           END-TRY.

       END METHOD button3_Click.

       METHOD-ID. button4_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
```

```
       01 selCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 delCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 insCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 tmpParameters OBJECT REFERENCE CLASS-SQLPARAMETERS.
       01 tmpParameter OBJECT REFERENCE CLASS-SQLPARAMETER.
       01 tmpDbType OBJECT REFERENCE ENUM-SQLDBTYPE.
       01 tmpRow OBJECT REFERENCE CLASS-DATAROW.
       01 tmpText OBJECT REFERENCE CLASS-STRING.
       01 tmpMessage OBJECT REFERENCE CLASS-STRING.
       01 excep OBJECT REFERENCE CLASS-EXCEPTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           TRY
      *  全削除処理のためコマンドを再構成します。
            INVOKE dataTable1 "Clear"
            SET selCommand TO PROP-SELECTCOMMAND OF sqlDataAdapter1
            SET PROP-COMMANDTEXT OF selCommand TO N"SELECT * FROM STOCK;"

      *  全削除用のコマンドを作成します。
            SET delCommand TO CLASS-SQLCOMMAND :: "NEW"
                       (N"DELETE FROM STOCK WHERE GNO = @GNO;", sqlConnection1)
            SET tmpParameters TO PROP-PARAMETERS OF delCommand
            SET tmpDbType TO PROP-SMALLINT OF ENUM-SQLDBTYPE
            SET tmpParameter  TO tmpParameters :: "Add" (N"@GNO", tmpDbType, 2, N"GNO")
            SET PROP-SOURCEVERSION OF tmpParameter TO PROP-ORIGINAL OF ENUM-DATAROWVERSION

            SET PROP-DELETECOMMAND OF sqlDataAdapter1 TO delCommand
            INVOKE sqlDataAdapter1 "Fill" USING dataTable1

      *  行コレクションを取り出してすべての行を削除します。
            PERFORM VARYING tmpRow THRU PROP-ROWS OF dataTable1
               INVOKE tmpRow "Delete"
            END-PERFORM

      *  追加用のコマンドを作成します。
            SET insCommand TO CLASS-SQLCOMMAND :: "NEW"
                        (N"INSERT INTO STOCK(GNO,GOODS,QOH,WHNO)
                         VALUES (@GNO,@GOODS,@QOH,@WHNO)",
                         sqlConnection1)
            SET tmpParameters TO PROP-PARAMETERS OF insCommand
            SET tmpDbType TO PROP-SMALLINT OF ENUM-SQLDBTYPE
            SET tmpParameter TO tmpParameters :: "Add" (N"@GNO", tmpDbType, 2, N"GNO")

            SET tmpDbType TO PROP-INT OF ENUM-SQLDBTYPE
            SET tmpParameter  TO tmpParameters :: "Add" (N"@QOH", tmpDbType, 4, N"QOH")

            SET tmpDbType TO PROP-VARCHAR OF ENUM-SQLDBTYPE
            SET tmpParameter TO tmpParameters :: "Add" (N"@GOODS", tmpDbType, 20, N"GOODS")

            SET tmpDbType TO PROP-SMALLINT OF ENUM-SQLDBTYPE
            SET tmpParameter TO tmpParameters :: "Add" (N"@WHNO", tmpDbType, 2, N"WHNO")

            SET PROP-INSERTCOMMAND OF sqlDataAdapter1 TO insCommand

      *  各レコードを追加します。
      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (110,"TELEVISION",85,2)
            INVOKE SELF "Insert_Stock" USING 110 N"TELEVISION" 85 2

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (111,"TELEVISION",90,2)
            INVOKE SELF "Insert_Stock" USING 111 N"TELEVISION" 90 2

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (123,"REFRIGERATOR",60,1)
            INVOKE SELF "Insert_Stock" USING 123 N"REFRIGERATOR" 60 1

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (124,"REFRIGERATOR",75,1)
            INVOKE SELF "Insert_Stock" USING 124 N"REFRIGERATOR" 75 1

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (137,"RADIO",150,2)
            INVOKE SELF "Insert_Stock" USING 137 N"RADIO" 150 2

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (138,"RADIO",200,2)
            INVOKE SELF "Insert_Stock" USING 138 N"RADIO" 200 2

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (140,"CASSETTE DECK",120,2)
            INVOKE SELF "Insert_Stock" USING 140 N"CASSETTE DECK" 120 2

      *  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (141,"CASSETTE DECK",80,2)
            INVOKE SELF "Insert_Stock" USING 141 N"CASSETTE DECK" 80 2
```

```
*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (200,"AIR CONDITIONER",04,1)
       INVOKE SELF "Insert_Stock" USING 200 N"AIR CONDITIONER" 4 1

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (201,"AIR CONDITIONER",15,1)
       INVOKE SELF "Insert_Stock" USING 201 N"AIR CONDITIONER" 15 1

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (212,"TELEVISION",10,2)
       INVOKE SELF "Insert_Stock" USING 212 N"TELEVISION" 10 2

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (215,"VIDEO",05,2)
       INVOKE SELF "Insert_Stock" USING 215 N"VIDEO" 5 2

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (226,"REFRIGERATOR",08,1)
       INVOKE SELF "Insert_Stock" USING 226 N"REFRIGERATOR" 8 1

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (227,"REFRIGERATOR",15,1)
       INVOKE SELF "Insert_Stock" USING 227 N"REFRIGERATOR" 15 1

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (240,"CASSETTE DECK",25,2)
       INVOKE SELF "Insert_Stock" USING 240 N"CASSETTE DECK" 25 2

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (243,"CASSETTE DECK",14,2)
       INVOKE SELF "Insert_Stock" USING 243 N"CASSETTE DECK" 14 2

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (351,"CASSETTE TAPE",2500,2)
       INVOKE SELF "Insert_Stock" USING 351 N"CASSETTE TAPE" 2500 2

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (380,"SHAVER",870,3)
       INVOKE SELF "Insert_Stock" USING 380 N"SHAVER" 870 3

*  INSERT INTO STOCK (GNO,GOODS,QOH,WHNO) VALUES (390,"DRIER",540,3)
       INVOKE SELF "Insert_Stock" USING 390 N"DRIER" 540 3

*  データテーブルを更新します。
       INVOKE sqlDataAdapter1 "Update" USING dataTable1

*  表示するデータテーブルを更新します。
       INVOKE SELF "List_Stocks"

    CATCH excep
*  例外を捕捉してメッセージボックスを表示します。
       SET tmpMessage TO PROP-MESSAGE OF excep
       INVOKE CLASS-MESSAGEBOX "Show" USING tmpMessage
    END-TRY.

END METHOD button4_Click.


METHOD-ID. Insert_Stock PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 tmpRow OBJECT REFERENCE CLASS-DATAROW.
LINKAGE SECTION.
01 GNO    BINARY-SHORT.
01 GOODS  OBJECT REFERENCE CLASS-STRING.
01 QOH    BINARY-LONG.
01 WHNO   BINARY-SHORT.
PROCEDURE DIVISION USING BY VALUE GNO GOODS QOH WHNO.

*  行を設定します。
    SET tmpRow TO dataTable1 :: "NewRow".
    INVOKE tmpRow "set_Item" USING N"GNO" GNO.
    INVOKE tmpRow "set_Item" USING N"GOODS" GOODS.
    INVOKE tmpRow "set_Item" USING N"QOH" QOH.
    INVOKE tmpRow "set_Item" USING N"WHNO" WHNO.

* 行を追加します。
    INVOKE PROP-ROWS OF dataTable1 "Add" USING tmpRow.

END METHOD Insert_Stock.


METHOD-ID. Get_SelectList PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 gridRow OBJECT REFERENCE CLASS-DATAGRIDVIEWROW.
01 selCount BINARY-LONG.
01 cell OBJECT REFERENCE CLASS-DATAGRIDVIEWCELL.
01 stringBuilder OBJECT REFERENCE CLASS-STRINGBUILDER.
01 tmpText OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
```

```
       01 retList OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION RETURNING retList.
      *  選択されている項目の製品番号を取得します
          MOVE 0 TO selCount.
          SET stringBuilder TO CLASS-STRINGBUILDER::"NEW".
          PERFORM VARYING gridRow THRU PROP-SELECTEDROWS OF dataGrid1
              IF selCount NOT = 0
                  INVOKE stringBuilder "Append" USING N","
              END-IF

              ADD 1 TO selCount
              SET cell TO PROP-CELLS OF gridRow::"get_Item"(0)
              INVOKE PROP-FORMATTEDVALUE OF cell "ToString" RETURNING tmpText
              INVOKE stringBuilder "Append" USING BY VALUE tmpText
          END-PERFORM.

          SET retList TO stringBuilder::"ToString".

       END METHOD Get_SelectList.


       METHOD-ID. List_Stocks PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 selCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 updCommand OBJECT REFERENCE CLASS-SQLCOMMAND.
       01 tmpParameters OBJECT REFERENCE CLASS-SQLPARAMETERS.
       01 tmpParameter OBJECT REFERENCE CLASS-SQLPARAMETER.
       01 tmpDbType OBJECT REFERENCE ENUM-SQLDBTYPE.
       PROCEDURE DIVISION.
      *  表示するデータテーブルを作成します。
          SET selCommand TO PROP-SELECTCOMMAND OF sqlDataAdapter1.
          IF selCommand = NULL
              INVOKE dataTable1 "Clear"
              SET selCommand TO CLASS-SQLCOMMAND :: "NEW" (N"SELECT * FROM STOCK;", sqlConnection1)
              SET PROP-SELECTCOMMAND OF sqlDataAdapter1 TO selCommand
          ELSE
              INVOKE dataTable1 "Clear"
              SET PROP-COMMANDTEXT OF selCommand TO N"SELECT * FROM STOCK;"
          END-IF.

      *  データグリッドによる更新のためのコマンドを設定します。
          SET updCommand TO CLASS-SQLCOMMAND :: "NEW"
                (N"UPDATE STOCK SET QOH = @QOH WHERE GNO = @GNO AND QOH = @OLDQOH;",
                 sqlConnection1).
          SET tmpParameters TO PROP-PARAMETERS OF updCommand.
          SET tmpDbType TO PROP-INT OF ENUM-SQLDBTYPE.
          SET tmpParameter  TO tmpParameters :: "Add" (N"@QOH", tmpDbType, 4, N"QOH").

          SET tmpDbType TO PROP-SMALLINT OF ENUM-SQLDBTYPE.
          SET tmpParameter TO tmpParameters :: "Add" (N"@GNO", tmpDbType, 2, N"GNO").
          SET PROP-SOURCEVERSION OF tmpParameter TO PROP-ORIGINAL OF ENUM-DATAROWVERSION.

          SET tmpDbType TO PROP-INT OF ENUM-SQLDBTYPE.
          SET tmpParameter TO tmpParameters :: "Add" (N"@OLDQOH", tmpDbType, 4, N"QOH").
          SET PROP-SOURCEVERSION OF tmpParameter TO PROP-ORIGINAL OF ENUM-DATAROWVERSION.

          SET PROP-UPDATECOMMAND OF sqlDataAdapter1 TO updCommand
          SET PROP-INSERTCOMMAND OF sqlDataAdapter1 TO NULL.
          SET PROP-DELETECOMMAND OF sqlDataAdapter1 TO NULL.

          INVOKE sqlDataAdapter1 "Fill" USING dataTable1.
       END METHOD List_Stocks.


       METHOD-ID. button5_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 formOrderList OBJECT REFERENCE FORM-ORDERLIST.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  オーダー覧フォームを開きます。
          SET formOrderList TO FORM-ORDERLIST :: "NEW".
          INVOKE formOrderList "Init" USING sqlConnection1.
          INVOKE formOrderList "ShowDialog" USING SELF.


       END METHOD button5_Click.
```

```
      END OBJECT.
      END CLASS CLASS-THIS.
```

NCOBU00927

ソースコード：OrderList.cob

```
CLASS-ID. CLASS-THIS AS "DataBinding.OrderList"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ISUPPORTINITIALIZE AS "System.ComponentModel.ISupportInitialize"
   CLASS CLASS-DATACOLUMN AS "System.Data.DataColumn"
   CLASS CLASS-DATACOLUMNS AS "System.Data.DataColumnCollection"
   CLASS CLASS-DATATABLE AS "System.Data.DataTable"
   CLASS CLASS-SQLCOMMAND AS "System.Data.SqlClient.SqlCommand"
   CLASS CLASS-SQLCONNECTION AS "System.Data.SqlClient.SqlConnection"
   CLASS CLASS-SQLDATAADAPTER AS "System.Data.SqlClient.SqlDataAdapter"
   CLASS CLASS-SQLPARAMETER AS "System.Data.SqlClient.SqlParameter"
   CLASS CLASS-SQLPARAMETERS AS "System.Data.SqlClient.SqlParameterCollection"
   ENUM ENUM-SQLDBTYPE AS "System.Data.SqlDbType"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SYSTEMCOLORS AS "System.Drawing.SystemColors"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-INT16 AS "System.Int16"
   CLASS CLASS-INT32 AS "System.Int32"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   ENUM ENUM-ANCHORSTYLES AS "System.Windows.Forms.AnchorStyles"
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS CLASS-COMBOBOX AS "System.Windows.Forms.ComboBox"
   ENUM ENUM-COMBOBOXSTYLE AS "System.Windows.Forms.ComboBoxStyle"
   CLASS ARRAY-CONTROL AS "System.Windows.Forms.Control[]"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   CLASS CLASS-CONTROLBINGINGS AS "System.Windows.Forms.ControlBindingsCollection"
   CLASS CLASS-DATAGRIDVIEW AS "System.Windows.Forms.DataGridView"
   CLASS CLASS-DATAGRIDVIEWCELL AS "System.Windows.Forms.DataGridViewCell"
   CLASS CLASS-DATAGRIDVIEWCOLUMN AS "System.Windows.Forms.DataGridViewColumn"
   CLASS CLASS-DATAGRIDVIEWCOLUMNS AS "System.Windows.Forms.DataGridViewColumnCollection"
   ENUM ENUM-DATAGRIDVIEWCOLUMNHEADERS AS
        "System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode"
   ENUM ENUM-CONTENTALIGNMENT AS "System.Windows.Forms.DataGridViewContentAlignment"
   CLASS CLASS-DATAGRIDVIEWROW AS "System.Windows.Forms.DataGridViewRow"
   CLASS CLASS-DATAGRIDVIEWSELECTEDROWS AS
        "System.Windows.Forms.DataGridViewSelectedRowCollection"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   ENUM ENUM-HORIZONTALALIGNMENT AS "System.Windows.Forms.HorizontalAlignment"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-TEXTBOX AS "System.Windows.Forms.TextBox"
   CLASS CLASS-STRINGBUILDER AS "System.Text.StringBuilder"
   PROPERTY PROP-ALIGNMENT AS "Alignment"
   PROPERTY PROP-ANCHOR AS "Anchor"
   PROPERTY PROP-AUTOSCALEBASESIZE AS "AutoScaleBaseSize"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-BOTTOM AS "Bottom"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-CAPTIONTEXT AS "CaptionText"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-COLUMNHEADERSHEIGHTSIZEMO AS "ColumnHeadersHeightSizeMode"
   PROPERTY PROP-COLUMNS AS "Columns"
   PROPERTY PROP-COMBOBOX1 AS "comboBox1"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-CONTROLTEXT AS "ControlText"
   PROPERTY PROP-DATABINDINGS AS "DataBindings"
   PROPERTY PROP-DATAGRID1 AS "dataGrid1"
   PROPERTY PROP-DATAMEMBER AS "DataMember"
   PROPERTY PROP-DATASOURCE AS "DataSource"
   PROPERTY PROP-DEFAULTCELLSTYLE AS "DefaultCellStyle"
   PROPERTY PROP-DISPLAYMEMBER AS "DisplayMember"
   PROPERTY PROP-DROPDOWNLIST AS "DropDownList"
   PROPERTY PROP-DROPDOWNSTYLE AS "DropDownStyle"
```

```
          PROPERTY PROP-EMPTY AS "Empty"
          PROPERTY PROP-GRIDCOLUMNSTYLES AS "GridColumnStyles"
          PROPERTY PROP-HEADERFORECOLOR AS "HeaderForeColor"
          PROPERTY PROP-HEADERTEXT AS "HeaderText"
          PROPERTY PROP-HEIGHT AS "Height"
          PROPERTY PROP-LABEL1 AS "label1"
          PROPERTY PROP-LABEL2 AS "label2"
          PROPERTY PROP-LABEL3 AS "label3"
          PROPERTY PROP-LEFT AS "Left"
          PROPERTY PROP-LENGTH AS "Length"
          PROPERTY PROP-LOCATION AS "Location"
          PROPERTY PROP-MAPPINGNAME AS "MappingName"
          PROPERTY PROP-MIDDLELEFT AS "MiddleLeft"
          PROPERTY PROP-MIDDLERIGHT AS "MiddleRight"
          PROPERTY PROP-NAME AS "Name"
          PROPERTY PROP-PARAMETERS AS "Parameters"
          PROPERTY PROP-READONLY AS "ReadOnly"
          PROPERTY PROP-RIGHT AS "Right"
          PROPERTY PROP-ROWTEMPLATE AS "RowTemplate"
          PROPERTY PROP-SELECTCOMMAND AS "SelectCommand"
          PROPERTY PROP-SELECTEDINDEX AS "SelectedIndex"
          PROPERTY PROP-SELECTEDVALUE AS "SelectedValue"
          PROPERTY PROP-SIZE AS "Size"
          PROPERTY PROP-SMALLINT AS "SmallInt"
          PROPERTY PROP-TABINDEX AS "TabIndex"
          PROPERTY PROP-TABLES AS "Tables"
          PROPERTY PROP-TABLESTYLES AS "TableStyles"
          PROPERTY PROP-TEXT AS "Text"
          PROPERTY PROP-TEXTBOX1 AS "textBox1"
          PROPERTY PROP-TEXTBOX2 AS "textBox2"
          PROPERTY PROP-TOP AS "Top"
          PROPERTY PROP-VALUE AS "Value"
          PROPERTY PROP-VALUEMEMBER AS "ValueMember"
          PROPERTY PROP-WIDTH AS "Width".

       OBJECT
          .
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 label1 OBJECT REFERENCE CLASS-LABEL.
       01 comboBox1 OBJECT REFERENCE CLASS-COMBOBOX.
       01 button1 OBJECT REFERENCE CLASS-BUTTON.
       01 components OBJECT REFERENCE CLASS-CONTAINER.
       01 label2 OBJECT REFERENCE CLASS-LABEL.
       01 textBox1 OBJECT REFERENCE CLASS-TEXTBOX.
       01 sqlConnection1 OBJECT REFERENCE CLASS-SQLCONNECTION.
       01 label3 OBJECT REFERENCE CLASS-LABEL.
       01 textBox2 OBJECT REFERENCE CLASS-TEXTBOX.
       01 dataGrid1 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 dataTable1 OBJECT REFERENCE CLASS-DATATABLE.
       PROCEDURE DIVISION.

       METHOD-ID. NEW.
       PROCEDURE DIVISION.
          INVOKE SELF "InitializeComponent".
       END METHOD NEW.

       METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING BY VALUE disposing.
          IF disposing = B"1" THEN
            IF components NOT = NULL THEN
              INVOKE components "Dispose"
            END-IF
          END-IF.
          INVOKE SUPER "Dispose" USING BY VALUE disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP2 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP3 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP4 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP5 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP6 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP7 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP8 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
```

```
       01 TEMP9 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP10 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       01 TEMP11 BINARY-LONG.
       01 TEMP12 BINARY-LONG.
       01 TEMP13 OBJECT REFERENCE CLASS-POINT.
       01 TEMP14 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP15 OBJECT REFERENCE CLASS-STRING.
       01 TEMP16 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP17 BINARY-LONG.
       01 TEMP18 BINARY-LONG.
       01 TEMP19 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP20 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP21 BINARY-LONG.
       01 TEMP22 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP23 OBJECT REFERENCE CLASS-STRING.
       01 TEMP24 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP25 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP26 BINARY-LONG.
       01 TEMP27 BINARY-LONG.
       01 TEMP28 OBJECT REFERENCE CLASS-POINT.
       01 TEMP29 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP30 OBJECT REFERENCE CLASS-STRING.
       01 TEMP31 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP32 BINARY-LONG.
       01 TEMP33 BINARY-LONG.
       01 TEMP34 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP35 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP36 BINARY-LONG.
       01 TEMP37 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP38 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP39 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP40 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP41 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP42 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP43 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP44 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP45 BINARY-LONG.
       01 TEMP46 BINARY-LONG.
       01 TEMP47 OBJECT REFERENCE CLASS-POINT.
       01 TEMP48 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP49 OBJECT REFERENCE CLASS-STRING.
       01 TEMP50 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP51 BINARY-LONG.
       01 TEMP52 BINARY-LONG.
       01 TEMP53 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP54 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP55 BINARY-LONG.
       01 TEMP56 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP57 OBJECT REFERENCE CLASS-STRING.
       01 TEMP58 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP59 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP60 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP61 BINARY-LONG.
       01 TEMP62 BINARY-LONG.
       01 TEMP63 OBJECT REFERENCE CLASS-POINT.
       01 TEMP64 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP65 OBJECT REFERENCE CLASS-STRING.
       01 TEMP66 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP67 BINARY-LONG.
       01 TEMP68 BINARY-LONG.
       01 TEMP69 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP70 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP71 BINARY-LONG.
       01 TEMP72 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP73 OBJECT REFERENCE CLASS-STRING.
       01 TEMP74 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP75 BINARY-LONG.
       01 TEMP76 BINARY-LONG.
       01 TEMP77 OBJECT REFERENCE CLASS-POINT.
       01 TEMP78 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP79 OBJECT REFERENCE CLASS-STRING.
       01 TEMP80 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP81 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP82 BINARY-LONG.
       01 TEMP83 BINARY-LONG.
       01 TEMP84 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP85 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP86 BINARY-LONG.
       01 TEMP87 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP88 BINARY-LONG.
       01 TEMP89 BINARY-LONG.
       01 TEMP90 OBJECT REFERENCE CLASS-POINT.
```

```
       01 TEMP91 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP92 OBJECT REFERENCE CLASS-STRING.
       01 TEMP93 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP94 BINARY-LONG.
       01 TEMP95 BINARY-LONG.
       01 TEMP96 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP97 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP98 BINARY-LONG.
       01 TEMP99 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP100 OBJECT REFERENCE CLASS-STRING.
       01 TEMP101 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP102 BINARY-LONG.
       01 TEMP103 BINARY-LONG.
       01 TEMP104 OBJECT REFERENCE CLASS-POINT.
       01 TEMP105 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP106 OBJECT REFERENCE CLASS-STRING.
       01 TEMP107 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP108 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP109 BINARY-LONG.
       01 TEMP110 BINARY-LONG.
       01 TEMP111 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP112 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP113 BINARY-LONG.
       01 TEMP114 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP115 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP116 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP117 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP118 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP119 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP120 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP121 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP122 OBJECT REFERENCE ENUM-ANCHORSTYLES.
       01 TEMP123 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP124 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP125 BINARY-LONG.
       01 TEMP126 BINARY-LONG.
       01 TEMP127 OBJECT REFERENCE CLASS-POINT.
       01 TEMP128 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP129 OBJECT REFERENCE CLASS-STRING.
       01 TEMP130 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP131 BINARY-LONG.
       01 TEMP132 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP133 OBJECT REFERENCE CLASS-DATAGRIDVIEWROW.
       01 TEMP134 BINARY-LONG.
       01 TEMP135 BINARY-LONG.
       01 TEMP136 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP137 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP138 BINARY-LONG.
       01 TEMP139 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP140 BINARY-LONG.
       01 TEMP141 BINARY-LONG.
       01 TEMP142 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP143 BINARY-LONG.
       01 TEMP144 BINARY-LONG.
       01 TEMP145 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP146 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP147 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP148 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP149 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP150 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP151 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP152 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP153 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP154 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP155 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP156 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP157 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP158 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP159 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP160 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP161 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP162 OBJECT REFERENCE CLASS-STRING.
       01 TEMP163 OBJECT REFERENCE CLASS-STRING.
       01 TEMP164 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP165 OBJECT REFERENCE CLASS-DATAGRIDVIEW.
       01 TEMP166 OBJECT REFERENCE INTERFACE-ISUPPORTINITIALIZE.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ComboBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.DataGridView" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="BeginInit" />
      *</prop>
```

```
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="SuspendLayout" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
```

```
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="100" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="在庫製品の選択" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="DropDownStyle" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ComboBoxStyle" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="DropDownList" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="10" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="140" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="SelectedIndexChanged" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="comboBox1_SelectedIndexChanged" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button1" />
```

```
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Anchor" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeCastExpression">
*<prop name="TargetType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Expression">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Right" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
```

```
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="280" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="290" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
*<string value="button1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="80" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="30" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="8" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="閉じる" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button1_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="10" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="40" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label2" />
*</prop>
*</object>
*</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="100" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="20" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="製品番号" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="120" />
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="40" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="ReadOnly" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="True" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
```

```
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="60" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="19" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="4" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
```

```
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="10" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="70" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="100" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="20" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="5" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="在庫数量" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="70" />
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ReadOnly" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
```

```
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="60" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="dataGrid1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Anchor" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeCastExpression">
*<prop name="TargetType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Expression">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Right" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Left" />
*</prop>
*</object>
*</prop>
*<prop name="Left">
*<object type="System.CodeDom.CodeBinaryOperatorExpression">
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AnchorStyles" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Bottom" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AnchorStyles" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Operator">
      *<enum type="System.CodeDom.CodeBinaryOperatorType" value="BitwiseOr" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ColumnHeadersHeightSizeMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="106" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
```

```
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="RowTemplate" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Height" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="21" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="348" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="178" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="OrderList" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleBaseSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="5" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
```

```
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="372" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="323" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
```

```
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
```

```
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="OrderList" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="在庫別オーダー覧" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Load" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="OrderList_Load" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeCastExpression">
      *<prop name="TargetType">
```

```
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.ComponentModel.ISupportInitialize" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="dataGrid1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="EndInit" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP1
          SET PROP-LABEL1 OF SELF TO TEMP1
          INVOKE CLASS-COMBOBOX "NEW" RETURNING TEMP2
          SET PROP-COMBOBOX1 OF SELF TO TEMP2
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP3
          SET PROP-BUTTON1 OF SELF TO TEMP3
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP4
          SET PROP-LABEL2 OF SELF TO TEMP4
          INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP5
```

```
          SET PROP-TEXTBOX1 OF SELF TO TEMP5
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP6
          SET PROP-LABEL3 OF SELF TO TEMP6
          INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP7
          SET PROP-TEXTBOX2 OF SELF TO TEMP7
          INVOKE CLASS-DATAGRIDVIEW "NEW" RETURNING TEMP8
          SET PROP-DATAGRID1 OF SELF TO TEMP8
          SET TEMP9 TO PROP-DATAGRID1 OF SELF
          SET TEMP10 TO TEMP9 AS INTERFACE-ISUPPORTINITIALIZE
          INVOKE TEMP10 "BeginInit"
          INVOKE SELF "SuspendLayout"
      *
      *label1
      *
          MOVE 10 TO TEMP11
          MOVE 10 TO TEMP12
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP11 BY VALUE TEMP12 RETURNING TEMP13
          SET TEMP14 TO PROP-LABEL1 OF SELF
          SET PROP-LOCATION OF TEMP14 TO TEMP13
          SET TEMP15 TO N"label1"
          SET TEMP16 TO PROP-LABEL1 OF SELF
          SET PROP-NAME OF TEMP16 TO TEMP15
          MOVE 100 TO TEMP17
          MOVE 20 TO TEMP18
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP17 BY VALUE TEMP18 RETURNING TEMP19
          SET TEMP20 TO PROP-LABEL1 OF SELF
          SET PROP-SIZE OF TEMP20 TO TEMP19
          MOVE 0 TO TEMP21
          SET TEMP22 TO PROP-LABEL1 OF SELF
          MOVE TEMP21 TO PROP-TABINDEX OF TEMP22
          SET TEMP23 TO N"在庫製品の選択"
          SET TEMP24 TO PROP-LABEL1 OF SELF
          SET PROP-TEXT OF TEMP24 TO TEMP23
      *
      *comboBox1
      *
          SET TEMP25 TO PROP-COMBOBOX1 OF SELF
          SET PROP-DROPDOWNSTYLE OF TEMP25 TO PROP-DROPDOWNLIST OF ENUM-COMBOBOXSTYLE
          MOVE 120 TO TEMP26
          MOVE 10 TO TEMP27
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP26 BY VALUE TEMP27 RETURNING TEMP28
          SET TEMP29 TO PROP-COMBOBOX1 OF SELF
          SET PROP-LOCATION OF TEMP29 TO TEMP28
          SET TEMP30 TO N"comboBox1"
          SET TEMP31 TO PROP-COMBOBOX1 OF SELF
          SET PROP-NAME OF TEMP31 TO TEMP30
          MOVE 140 TO TEMP32
          MOVE 20 TO TEMP33
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP32 BY VALUE TEMP33 RETURNING TEMP34
          SET TEMP35 TO PROP-COMBOBOX1 OF SELF
          SET PROP-SIZE OF TEMP35 TO TEMP34
          MOVE 1 TO TEMP36
          SET TEMP37 TO PROP-COMBOBOX1 OF SELF
          MOVE TEMP36 TO PROP-TABINDEX OF TEMP37
          SET TEMP38 TO PROP-COMBOBOX1 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                 BY VALUE N"comboBox1_SelectedIndexChanged" RETURNING TEMP39
          INVOKE TEMP38 "add_SelectedIndexChanged" USING BY VALUE TEMP39
      *
      *button1
      *
          SET TEMP40 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
          SET TEMP41 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
          SET TEMP42 TO FUNCTION ENUM-OR(TEMP40 TEMP41)
          SET TEMP43 TO TEMP42
          SET TEMP44 TO PROP-BUTTON1 OF SELF
          SET PROP-ANCHOR OF TEMP44 TO TEMP43
          MOVE 280 TO TEMP45
          MOVE 290 TO TEMP46
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP45 BY VALUE TEMP46 RETURNING TEMP47
          SET TEMP48 TO PROP-BUTTON1 OF SELF
          SET PROP-LOCATION OF TEMP48 TO TEMP47
          SET TEMP49 TO N"button1"
          SET TEMP50 TO PROP-BUTTON1 OF SELF
          SET PROP-NAME OF TEMP50 TO TEMP49
          MOVE 80 TO TEMP51
          MOVE 30 TO TEMP52
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP51 BY VALUE TEMP52 RETURNING TEMP53
          SET TEMP54 TO PROP-BUTTON1 OF SELF
          SET PROP-SIZE OF TEMP54 TO TEMP53
          MOVE 8 TO TEMP55
```

```
           SET TEMP56 TO PROP-BUTTON1 OF SELF
           MOVE TEMP55 TO PROP-TABINDEX OF TEMP56
           SET TEMP57 TO N"閉じる"
           SET TEMP58 TO PROP-BUTTON1 OF SELF
           SET PROP-TEXT OF TEMP58 TO TEMP57
           SET TEMP59 TO PROP-BUTTON1 OF SELF
           INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click"
                  RETURNING TEMP60
           INVOKE TEMP59 "add_Click" USING BY VALUE TEMP60
       *
       *label2
       *
           MOVE 10 TO TEMP61
           MOVE 40 TO TEMP62
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP61 BY VALUE TEMP62 RETURNING TEMP63
           SET TEMP64 TO PROP-LABEL2 OF SELF
           SET PROP-LOCATION OF TEMP64 TO TEMP63
           SET TEMP65 TO N"label2"
           SET TEMP66 TO PROP-LABEL2 OF SELF
           SET PROP-NAME OF TEMP66 TO TEMP65
           MOVE 100 TO TEMP67
           MOVE 20 TO TEMP68
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP67 BY VALUE TEMP68 RETURNING TEMP69
           SET TEMP70 TO PROP-LABEL2 OF SELF
           SET PROP-SIZE OF TEMP70 TO TEMP69
           MOVE 3 TO TEMP71
           SET TEMP72 TO PROP-LABEL2 OF SELF
           MOVE TEMP71 TO PROP-TABINDEX OF TEMP72
           SET TEMP73 TO N"製品番号"
           SET TEMP74 TO PROP-LABEL2 OF SELF
           SET PROP-TEXT OF TEMP74 TO TEMP73
       *
       *textBox1
       *
           MOVE 120 TO TEMP75
           MOVE 40 TO TEMP76
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP75 BY VALUE TEMP76 RETURNING TEMP77
           SET TEMP78 TO PROP-TEXTBOX1 OF SELF
           SET PROP-LOCATION OF TEMP78 TO TEMP77
           SET TEMP79 TO N"textBox1"
           SET TEMP80 TO PROP-TEXTBOX1 OF SELF
           SET PROP-NAME OF TEMP80 TO TEMP79
           SET TEMP81 TO PROP-TEXTBOX1 OF SELF
           SET PROP-READONLY OF TEMP81 TO B"1"
           MOVE 60 TO TEMP82
           MOVE 19 TO TEMP83
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP82 BY VALUE TEMP83 RETURNING TEMP84
           SET TEMP85 TO PROP-TEXTBOX1 OF SELF
           SET PROP-SIZE OF TEMP85 TO TEMP84
           MOVE 4 TO TEMP86
           SET TEMP87 TO PROP-TEXTBOX1 OF SELF
           MOVE TEMP86 TO PROP-TABINDEX OF TEMP87
       *
       *label3
       *
           MOVE 10 TO TEMP88
           MOVE 70 TO TEMP89
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP88 BY VALUE TEMP89 RETURNING TEMP90
           SET TEMP91 TO PROP-LABEL3 OF SELF
           SET PROP-LOCATION OF TEMP91 TO TEMP90
           SET TEMP92 TO N"label3"
           SET TEMP93 TO PROP-LABEL3 OF SELF
           SET PROP-NAME OF TEMP93 TO TEMP92
           MOVE 100 TO TEMP94
           MOVE 20 TO TEMP95
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP94 BY VALUE TEMP95 RETURNING TEMP96
           SET TEMP97 TO PROP-LABEL3 OF SELF
           SET PROP-SIZE OF TEMP97 TO TEMP96
           MOVE 5 TO TEMP98
           SET TEMP99 TO PROP-LABEL3 OF SELF
           MOVE TEMP98 TO PROP-TABINDEX OF TEMP99
           SET TEMP100 TO N"在庫数量"
           SET TEMP101 TO PROP-LABEL3 OF SELF
           SET PROP-TEXT OF TEMP101 TO TEMP100
       *
       *textBox2
       *
           MOVE 120 TO TEMP102
           MOVE 70 TO TEMP103
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP102 BY VALUE TEMP103 RETURNING TEMP104
           SET TEMP105 TO PROP-TEXTBOX2 OF SELF
```

```
        SET PROP-LOCATION OF TEMP105 TO TEMP104
        SET TEMP106 TO N"textBox2"
        SET TEMP107 TO PROP-TEXTBOX2 OF SELF
        SET PROP-NAME OF TEMP107 TO TEMP106
        SET TEMP108 TO PROP-TEXTBOX2 OF SELF
        SET PROP-READONLY OF TEMP108 TO B"1"
        MOVE 60 TO TEMP109
        MOVE 19 TO TEMP110
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP109 BY VALUE TEMP110 RETURNING TEMP111
        SET TEMP112 TO PROP-TEXTBOX2 OF SELF
        SET PROP-SIZE OF TEMP112 TO TEMP111
        MOVE 6 TO TEMP113
        SET TEMP114 TO PROP-TEXTBOX2 OF SELF
        MOVE TEMP113 TO PROP-TABINDEX OF TEMP114
    *
    *dataGrid1
    *
        SET TEMP115 TO PROP-TOP OF ENUM-ANCHORSTYLES
        SET TEMP116 TO PROP-BOTTOM OF ENUM-ANCHORSTYLES
        SET TEMP117 TO FUNCTION ENUM-OR(TEMP115 TEMP116)
        SET TEMP118 TO PROP-LEFT OF ENUM-ANCHORSTYLES
        SET TEMP119 TO FUNCTION ENUM-OR(TEMP117 TEMP118)
        SET TEMP120 TO PROP-RIGHT OF ENUM-ANCHORSTYLES
        SET TEMP121 TO FUNCTION ENUM-OR(TEMP119 TEMP120)
        SET TEMP122 TO TEMP121
        SET TEMP123 TO PROP-DATAGRID1 OF SELF
        SET PROP-ANCHOR OF TEMP123 TO TEMP122
        SET TEMP124 TO PROP-DATAGRID1 OF SELF
        SET PROP-COLUMNHEADERSHEIGHTSIZEMO OF TEMP124 TO
            PROP-AUTOSIZE OF ENUM-DATAGRIDVIEWCOLUMNHEADERS
        MOVE 12 TO TEMP125
        MOVE 106 TO TEMP126
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP125 BY VALUE TEMP126 RETURNING TEMP127
        SET TEMP128 TO PROP-DATAGRID1 OF SELF
        SET PROP-LOCATION OF TEMP128 TO TEMP127
        SET TEMP129 TO N"dataGrid1"
        SET TEMP130 TO PROP-DATAGRID1 OF SELF
        SET PROP-NAME OF TEMP130 TO TEMP129
        MOVE 21 TO TEMP131
        SET TEMP132 TO PROP-DATAGRID1 OF SELF
        SET TEMP133 TO PROP-ROWTEMPLATE OF TEMP132
        MOVE TEMP131 TO PROP-HEIGHT OF TEMP133
        MOVE 348 TO TEMP134
        MOVE 178 TO TEMP135
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP134 BY VALUE TEMP135 RETURNING TEMP136
        SET TEMP137 TO PROP-DATAGRID1 OF SELF
        SET PROP-SIZE OF TEMP137 TO TEMP136
        MOVE 9 TO TEMP138
        SET TEMP139 TO PROP-DATAGRID1 OF SELF
        MOVE TEMP138 TO PROP-TABINDEX OF TEMP139
    *
    *OrderList
    *
        MOVE 5 TO TEMP140
        MOVE 12 TO TEMP141
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP140 BY VALUE TEMP141 RETURNING TEMP142
        SET PROP-AUTOSCALEBASESIZE OF SELF TO TEMP142
        MOVE 372 TO TEMP143
        MOVE 323 TO TEMP144
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP143 BY VALUE TEMP144 RETURNING TEMP145
        SET PROP-CLIENTSIZE OF SELF TO TEMP145
        SET TEMP147 TO PROP-DATAGRID1 OF SELF
        SET TEMP146 TO PROP-CONTROLS OF SELF
        INVOKE TEMP146 "Add" USING BY VALUE TEMP147
        SET TEMP149 TO PROP-TEXTBOX2 OF SELF
        SET TEMP148 TO PROP-CONTROLS OF SELF
        INVOKE TEMP148 "Add" USING BY VALUE TEMP149
        SET TEMP151 TO PROP-LABEL3 OF SELF
        SET TEMP150 TO PROP-CONTROLS OF SELF
        INVOKE TEMP150 "Add" USING BY VALUE TEMP151
        SET TEMP153 TO PROP-TEXTBOX1 OF SELF
        SET TEMP152 TO PROP-CONTROLS OF SELF
        INVOKE TEMP152 "Add" USING BY VALUE TEMP153
        SET TEMP155 TO PROP-LABEL2 OF SELF
        SET TEMP154 TO PROP-CONTROLS OF SELF
        INVOKE TEMP154 "Add" USING BY VALUE TEMP155
        SET TEMP157 TO PROP-BUTTON1 OF SELF
        SET TEMP156 TO PROP-CONTROLS OF SELF
        INVOKE TEMP156 "Add" USING BY VALUE TEMP157
        SET TEMP159 TO PROP-COMBOBOX1 OF SELF
        SET TEMP158 TO PROP-CONTROLS OF SELF
        INVOKE TEMP158 "Add" USING BY VALUE TEMP159
```

```
           SET TEMP161 TO PROP-LABEL1 OF SELF
           SET TEMP160 TO PROP-CONTROLS OF SELF
           INVOKE TEMP160 "Add" USING BY VALUE TEMP161
           SET TEMP162 TO N"OrderList"
           SET PROP-NAME OF SELF TO TEMP162
           SET TEMP163 TO N"在庫別オーダー覧"
           SET PROP-TEXT OF SELF TO TEMP163
           INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"OrderList_Load"
                  RETURNING TEMP164
           INVOKE SELF "add_Load" USING BY VALUE TEMP164
           SET TEMP165 TO PROP-DATAGRID1 OF SELF
           SET TEMP166 TO TEMP165 AS INTERFACE-ISUPPORTINITIALIZE
           INVOKE TEMP166 "EndInit"
           INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
           INVOKE SELF "PerformLayout"
       END METHOD INITIALIZECOMPONENT.

       METHOD-ID. Init.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 tmpDataAdapter OBJECT REFERENCE CLASS-SQLDATAADAPTER.
       01 tmpBindings OBJECT REFERENCE CLASS-CONTROLBINGINGS.
       LINKAGE SECTION.
       01 paramConnection OBJECT REFERENCE CLASS-SQLCONNECTION.
       PROCEDURE DIVISION USING BY VALUE paramConnection.
      *   コネクションオブジェクトを保存します。
           SET sqlConnection1 TO paramConnection.

      *  コンボボックスとテキストボックスに STOCK テーブルをバインドします。
           SET tmpDataAdapter TO CLASS-SQLDATAADAPTER :: "NEW"
              (N"SELECT GNO,GOODS,QOH FROM STOCK;", sqlConnection1).
           SET dataTable1 TO CLASS-DATATABLE :: "NEW".
           INVOKE tmpDataAdapter "Fill" USING dataTable1.

           SET PROP-DATASOURCE OF comboBox1 TO dataTable1.
           SET PROP-DISPLAYMEMBER OF comboBox1 TO N"GOODS".
           SET PROP-VALUEMEMBER OF comboBox1 TO N"GNO".

           SET tmpBindings TO PROP-DATABINDINGS OF textBox1.
           INVOKE tmpBindings "Add" USING N"Text" dataTable1 N"GNO".
           SET tmpBindings TO PROP-DATABINDINGS OF textBox2.
           INVOKE tmpBindings "Add" USING N"Text" dataTable1 N"QOH".

           SET PROP-SELECTEDINDEX OF comboBox1 TO -1.

       END METHOD Init.

       METHOD-ID. OrderList_Load PRIVATE.
       DATA DIVISION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           SET PROP-SELECTEDINDEX OF comboBox1 TO 0.
       END METHOD OrderList_Load.

       METHOD-ID. button1_Click PRIVATE.
       DATA DIVISION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  このフォームを終了します。
           INVOKE SELF "Close".

       END METHOD button1_Click.

       METHOD-ID. comboBox1_SelectedIndexChanged PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 adpOrders OBJECT REFERENCE CLASS-SQLDATAADAPTER.
       01 stringBuilder OBJECT REFERENCE CLASS-STRINGBUILDER.
       01 tmpCommandText OBJECT REFERENCE CLASS-STRING.
       01 tmpParameters OBJECT REFERENCE CLASS-SQLPARAMETERS.
       01 tmpParameter  OBJECT REFERENCE CLASS-SQLPARAMETER.
       01 tmpDbType OBJECT REFERENCE ENUM-SQLDBTYPE.
       01 tmpTable OBJECT REFERENCE CLASS-DATATABLE.
       01 tmpColumns OBJECT REFERENCE CLASS-DATACOLUMNS.
       01 tmpColumn1  OBJECT REFERENCE CLASS-DATACOLUMN.
       01 tmpColumn2  OBJECT REFERENCE CLASS-DATACOLUMN.
       01 tmpText OBJECT REFERENCE CLASS-STRING.
```

```
       01 tmpObject OBJECT REFERENCE CLASS-OBJECT.
       01 selValue BINARY-SHORT.
       01 len BINARY-LONG.
       01 tmpGridColumns OBJECT REFERENCE CLASS-DATAGRIDVIEWCOLUMNS.
       01 tmpGridColumn OBJECT REFERENCE CLASS-DATAGRIDVIEWCOLUMN.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      *  データグリッドにオーダー覧を作成します。
           SET tmpText TO PROP-TEXT OF textBox1.
           MOVE PROP-LENGTH OF tmpText TO len.
           IF len <= 0
              EXIT METHOD.

      *  データテーブルを作成します。
           SET tmpTable TO CLASS-DATATABLE :: "NEW".

      *  コマンド文字列を作成します。
           SET stringBuilder TO CLASS-STRINGBUILDER::"NEW".
           INVOKE stringBuilder "Append"
                  USING N"SELECT NAME,ADDRESS,OOH,PHONE FROM STOCK,ORDERS,COMPANY ".
           INVOKE stringBuilder "Append"
                  USING N"WHERE GOODSNO=@GOODSNO AND GOODSNO=GNO AND CNO=COMPANYNO;".
           SET tmpCommandText TO stringBuilder::"ToString".

      *  ORDERS テーブルを取り出します
           SET adpOrders TO CLASS-SQLDATAADAPTER :: "NEW" (tmpCommandText, sqlConnection1).
           SET tmpParameters TO PROP-PARAMETERS OF PROP-SELECTCOMMAND OF adpOrders.
           SET tmpDbType TO PROP-SMALLINT OF ENUM-SQLDBTYPE.
           SET tmpParameter TO tmpParameters :: "Add" (N"@GOODSNO", tmpDbType, 2, N"GOODSNO").
           SET PROP-VALUE OF tmpParameter TO PROP-SELECTEDVALUE OF comboBox1.
           INVOKE adpOrders "Fill" USING tmpTable.

      *  データテーブルをデータグリッドにバインドします。
           SET PROP-DATASOURCE OF dataGrid1 TO tmpTable.

      *  データグリッドのヘッダをカスタマイズします
           SET tmpGridColumns TO PROP-COLUMNS OF dataGrid1.
           IF tmpGridColumns NOT = NULL

              SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"NAME")
              SET PROP-HEADERTEXT OF tmpGridColumn TO N"取引先"
              SET PROP-READONLY OF tmpGridColumn TO B"1"
              SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                  PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
              SET PROP-WIDTH OF tmpGridColumn TO 100

              SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"ADDRESS")
              SET PROP-HEADERTEXT OF tmpGridColumn TO N"所在地"
              SET PROP-READONLY OF tmpGridColumn TO B"1"
              SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                  PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
              SET PROP-WIDTH OF tmpGridColumn TO 120

              SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"OOH")
              SET PROP-HEADERTEXT OF tmpGridColumn TO N"発注数量"
              SET PROP-READONLY OF tmpGridColumn TO B"1"
              SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                  PROP-MIDDLERIGHT OF ENUM-CONTENTALIGNMENT
              SET PROP-WIDTH OF tmpGridColumn TO 80

              SET tmpGridColumn TO tmpGridColumns :: "get_Item" (N"PHONE")
              SET PROP-HEADERTEXT OF tmpGridColumn TO N"連絡先"
              SET PROP-READONLY OF tmpGridColumn TO B"1"
              SET PROP-ALIGNMENT OF PROP-DEFAULTCELLSTYLE OF tmpGridColumn TO
                  PROP-MIDDLELEFT OF ENUM-CONTENTALIGNMENT
              SET PROP-WIDTH OF tmpGridColumn TO 80


           END-IF.

       END METHOD comboBox1_SelectedIndexChanged.

       END OBJECT.
       END CLASS CLASS-THIS.
```

ソースコード：Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
   .
REPOSITORY.
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-MAINFORM AS "DataBinding.MainForm"
   CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
   .

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-MAINFORM OBJECT REFERENCE CLASS-MAINFORM.
PROCEDURE DIVISION.
   INVOKE CLASS-APPLICATION "EnableVisualStyles".
   INVOKE CLASS-APPLICATION "SetCompatibleTextRenderingDefault" USING BY VALUE B"0".

   INVOKE CLASS-MAINFORM "NEW" RETURNING WK-MAINFORM.
   INVOKE CLASS-APPLICATION "Run" USING BY VALUE WK-MAINFORM.
END PROGRAM MAIN.
```

NCOBU01046

ソースコード：DBSetup.cob

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-SELF AS "ZipCodeDBSetup.DBSetup".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
    CLASS CLASS-OBJECT AS "System.Object"
    CLASS CLASS-CONFIGURATIONMANAGER AS "System.Configuration.ConfigurationManager"
    CLASS CLASS-CONNECTIONSTRINGSETTINGS AS
          "System.Configuration.ConnectionStringSettings"
    CLASS CLASS-SQLCONNECTION AS "System.Data.SqlClient.SqlConnection"
    CLASS CLASS-SQLCOMMAND AS "System.Data.SqlClient.SqlCommand"
    CLASS CLASS-SQLBULKCOPY AS "System.Data.SqlClient.SqlBulkCopy"
    CLASS CLASS-SQLROWSCOPIEDEVENTARGS AS "System.Data.SqlClient.SqlRowsCopiedEventArgs"
    CLASS CLASS-OLEDBCONNECTION AS "System.Data.OleDb.OleDbConnection"
    CLASS CLASS-OLEDBCOMMAND AS "System.Data.OleDb.OleDbCommand"
    CLASS CLASS-OLEDBDATAREADER AS "System.Data.OleDb.OleDbDataReader"
    CLASS CLASS-STRING AS "System.String"
    CLASS CLASS-STRINGBUILDER AS "System.Text.StringBuilder"
    DELEGATE DELEGATE-SQLROWSCOPIEDEVENT AS
          "System.Data.SqlClient.SqlRowsCopiedEventHandler"
    PROPERTY PROP-CONNECTIONSTRING AS "ConnectionString"
    PROPERTY PROP-CONNECTIONSTRINGS AS "ConnectionStrings"
    PROPERTY PROP-COMMANDTEXT AS "CommandText"
    PROPERTY PROP-BATCHSIZE AS "BatchSize"
    PROPERTY PROP-BULKCOPYTIMEOUT AS "BulkCopyTimeout"
    PROPERTY PROP-DESTINATIONTABLENAME AS "DestinationTableName"
    PROPERTY PROP-NOTIFYAFTER AS "NotifyAfter".

STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. SETUP AS "Setup".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 SQL-CONNECTION      OBJECT REFERENCE CLASS-SQLCONNECTION.
01 SQL-BULKCOPY        OBJECT REFERENCE CLASS-SQLBULKCOPY.
01 OLEDB-CONNECTION    OBJECT REFERENCE CLASS-OLEDBCONNECTION.
01 OLEDB-COMMAND       OBJECT REFERENCE CLASS-OLEDBCOMMAND.
01 OLEDB-DATAREADER    OBJECT REFERENCE CLASS-OLEDBDATAREADER.
01 SQL-CONNECTIONSTRING OBJECT REFERENCE CLASS-CONNECTIONSTRINGSETTINGS.
01 CSV-CONNECTIONSTRING OBJECT REFERENCE CLASS-CONNECTIONSTRINGSETTINGS.
01 EVENT-HANDLER       OBJECT REFERENCE DELEGATE-SQLROWSCOPIEDEVENT.
PROCEDURE DIVISION.
    *>  構成ファイルから各接続文字列情報を取得します
    SET SQL-CONNECTIONSTRING TO PROP-CONNECTIONSTRINGS OF
        CLASS-CONFIGURATIONMANAGER::"get_Item"("SQL").
    SET CSV-CONNECTIONSTRING TO PROP-CONNECTIONSTRINGS OF
        CLASS-CONFIGURATIONMANAGER::"get_Item"("CSV").

    *>  各コネクションオブジェクトを生成します
    SET SQL-CONNECTION TO CLASS-SQLCONNECTION::"NEW"
    SET OLEDB-CONNECTION TO CLASS-OLEDBCONNECTION::"NEW"
    TRY
        *>  SQL Server に接続します
        SET PROP-CONNECTIONSTRING OF SQL-CONNECTION TO
            PROP-CONNECTIONSTRING OF SQL-CONNECTIONSTRING
        INVOKE SQL-CONNECTION "Open"

        *>  ZipCode テーブルを作成します
        INVOKE CLASS-SELF "ResetTable" USING SQL-CONNECTION

        *>  CSV ファイルを開きます
        SET PROP-CONNECTIONSTRING OF OLEDB-CONNECTION TO
            PROP-CONNECTIONSTRING OF CSV-CONNECTIONSTRING
        INVOKE OLEDB-CONNECTION "Open"

        *>  CSV データを読み込みます
```

```
            SET OLEDB-COMMAND TO OLEDB-CONNECTION::"CreateCommand"
            SET PROP-COMMANDTEXT OF OLEDB-COMMAND TO
            N"SELECT * FROM [KEN_ALL#CSV] IN """" ""Text;DATABASE=.¥;HDR=NO;FMT=DELIMITED;"""
            SET OLEDB-DATAREADER TO OLEDB-COMMAND::"ExecuteReader"

            *>  SqlBulkCopy で SQL Server にデータを転送するための準備をします
            SET SQL-BULKCOPY TO CLASS-SQLBULKCOPY::"NEW"(SQL-CONNECTION)
            SET PROP-BATCHSIZE OF SQL-BULKCOPY TO 100
            SET PROP-BULKCOPYTIMEOUT OF SQL-BULKCOPY TO 3600
            SET PROP-DESTINATIONTABLENAME OF SQL-BULKCOPY TO N"ZipCode"
            SET PROP-NOTIFYAFTER OF SQL-BULKCOPY TO 1000
            INVOKE DELEGATE-SQLROWSCOPIEDEVENT "NEW" USING CLASS-SELF N"Progress"
                   RETURNING EVENT-HANDLER
            INVOKE SQL-BULKCOPY "add_SqlRowsCopied" USING EVENT-HANDLER
            *>  データの転送を開始します。
            INVOKE SQL-BULKCOPY "WriteToServer" USING OLEDB-DATAREADER

            DISPLAY "."
            DISPLAY "データの転送が完了しました。"
            INVOKE SQL-BULKCOPY "Close"
        FINALLY
            *>  終了処理
            INVOKE SQL-CONNECTION "Dispose"
            INVOKE OLEDB-CONNECTION "Dispose"
        END-TRY.

     END METHOD SETUP.

    *
    * SqlRawsCopied イベント
    *
     METHOD-ID. PREGRESS AS "Progress" PRIVATE.
     DATA DIVISION.
     LINKAGE SECTION.
     01 SENDER OBJECT REFERENCE CLASS-OBJECT.
     01 E     OBJECT REFERENCE CLASS-SQLROWSCOPIEDEVENTARGS.
     PROCEDURE DIVISION USING BY VALUE SENDER E.
        DISPLAY "." NO ADVANCING.
     END METHOD PREGRESS.

    *
    * テーブルを作成します
    *
     METHOD-ID. RESETTABLE AS "ResetTable" PRIVATE.
     DATA DIVISION.
     WORKING-STORAGE SECTION.
     01 DROP-COMMAND   OBJECT REFERENCE CLASS-SQLCOMMAND.
     01 CREATE-COMMAND OBJECT REFERENCE CLASS-SQLCOMMAND.
     01 STRING-BUILDER OBJECT REFERENCE CLASS-STRINGBUILDER.
     01 COMMAND-TEXT   OBJECT REFERENCE CLASS-STRING.
     LINKAGE SECTION.
     01 SQL-CONNECTION OBJECT REFERENCE CLASS-SQLCONNECTION.
     PROCEDURE DIVISION USING BY VALUE SQL-CONNECTION.

        *>  ZipCode テーブルを削除します
        SET DROP-COMMAND TO SQL-CONNECTION::"CreateCommand".
        TRY
            SET PROP-COMMANDTEXT OF DROP-COMMAND TO N"DROP TABLE [dbo].[ZipCode]"
            INVOKE DROP-COMMAND "ExecuteNonQuery"
        CATCH
            CONTINUE
        FINALLY
            INVOKE DROP-COMMAND "Dispose"
        END-TRY.


        *>  ZipCode テーブルを作成します
        SET CREATE-COMMAND TO SQL-CONNECTION::"CreateCommand".
        TRY
            SET STRING-BUILDER TO CLASS-STRINGBUILDER::"NEW"
            INVOKE STRING-BUILDER "Append" USING N"CREATE TABLE [dbo].[ZipCode]("
            INVOKE STRING-BUILDER "Append" USING N" [全国地方公共団体コード] [int] NOT NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [郵便番号5] [varchar](5) COLLATE Japanese_CI_AS NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [郵便番号7] [varchar](7) COLLATE Japanese_CI_AS NOT NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [都道府県名カナ] [varchar](8) COLLATE Japanese_CI_AS NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [市区町村名カナ] [varchar](25) COLLATE Japanese_CI_AS NULL,"
```

```
            INVOKE STRING-BUILDER "Append"
                   USING N" [町域名カナ] [varchar](80) COLLATE Japanese_CI_AS NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [都道府県名] [nvarchar](5) COLLATE Japanese_CI_AS NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [市区町村名] [nvarchar](20) COLLATE Japanese_CI_AS NULL,"
            INVOKE STRING-BUILDER "Append"
                   USING N" [町域名] [nvarchar](40) COLLATE Japanese_CI_AS NULL,"
            INVOKE STRING-BUILDER "Append" USING N" [FLG1] [smallint] NULL,"
            INVOKE STRING-BUILDER "Append" USING N" [FLG2] [smallint] NULL,"
            INVOKE STRING-BUILDER "Append" USING N" [FLG3] [smallint] NULL,"
            INVOKE STRING-BUILDER "Append" USING N" [FLG4] [smallint] NULL,"
            INVOKE STRING-BUILDER "Append" USING N" [FLG5] [smallint] NULL,"
            INVOKE STRING-BUILDER "Append" USING N" [FLG6] [smallint] NULL"
            INVOKE STRING-BUILDER "Append" USING N") ON [PRIMARY]"
            SET COMMAND-TEXT TO STRING-BUILDER::"ToString"
            SET PROP-COMMANDTEXT OF CREATE-COMMAND TO COMMAND-TEXT
            INVOKE CREATE-COMMAND "ExecuteNonQuery"
         CATCH
            CONTINUE
         FINALLY
            INVOKE CREATE-COMMAND "Dispose"
         END-TRY.

      END METHOD RESETTABLE.
      END STATIC.

      END CLASS CLASS-SELF.
```

ソースコード：Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN AS "ZipCodeDBSetup.Main".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-DBSETUP AS "ZipCodeDBSetup.DBSetup"
   .
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
* ZipCode データベースをセットアップします。
   INVOKE CLASS-DBSETUP "Setup".
END PROGRAM MAIN.
```

NCOBU00738

NetCOBOL for .NET Web サンプル アプリケーション（ASP.NET および Web サービス）

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

Web サンプル アプリケーションは、COBOL を使用した ASP.NET および Web サービスをデモンストレーションします。 サンプルは Web サブフォルダにあります。 サンプルを試すには、Visual Studio からソリューションファイル(.sln)を開いてください。

ASP.NET サンプル アプリケーション

ソリューションエクスプローラーでスタートページ(通常は、Default.aspx)を選択し、[デバッグ]-[デバッグなしで開始]メニューを選択することで実行することができます。

1. ページの状態の保存

   ViewState を使ってページ内の変数を保存する例です。

フォルダ	Counter
ソリューションファイル	Counter.sln
スタートページ	`CounterWebSite¥Default.aspx`
ソースコード	CounterWebSite¥Default.aspx.cobx

2. 複数ページのサンプル

   複数 Web ページの簡単な例です。

フォルダ	MultiPage
ソリューションファイル	MultiPage.sln
スタートページ	`MultiPageWebSite¥Default.aspx`
ソースコード	MultiPageWebSite¥Default.aspx.cobx MultiPageWebSite¥SecondPage.aspx.cobx MultiPageWebSite¥ThirdPage.aspx.cobx

3. コントロールの動的生成

   Web コントロールを動的に生成する例です。

フォルダ	CreateControl
ソリューションファイル	CreateControl.sln

スタートページ	CreateControlWebSite¥Default.aspx
ソースコード	CreateControlWebSite¥Default.aspx.cobx

4. Web サービスの呼び出し

郵便番号検索サービスを利用する例です。 また、複数の検索結果を表示するために、HTMLTable コントロールを使用して、 動的に Table の行を追加しています。

フォルダ	ZipCodeWebForms
ソリューションファイル	ZipCodeWebForms.sln
スタートページ	ZipCodeWebFormsWebSite¥Default.aspx
ソースコード	ZipCodeWebFormsWebSite¥Default.aspx.cobx ZipCodeWebFormsWebSite¥ResultForm.aspx.cobx ZipCodeWebFormsWebSite¥SelectAddressForm.aspx.cobx

サンプルを動かす前に、「郵便番号検索サービス」を IIS で公開しておく必要があります。 IIS での公開については、COBOL Web サービス アプリケーションの「郵便番号検索サービス」を参照してください。

5. セッション状態の保存

セッション状態の変数を使って、ページ間で情報を受け渡す例です。 このサンプルではユーザ情報の登録フォームを例にしています。 ページの移動は HttpRequest オブジェクトの Redirect メソッドを利用し、ページの動作をクエリ文字列によって切り分けています。 情報送信ロジックは組み込まれていないため、「送信」ボタンを押しても実際に情報が送信されることはありません。 なお、Web サービスの呼び出しサンプルと同じく、「郵便番号検索サービス」を利用しています。 また、このサンプルでは郵便番号に対応する住所が複数ある場合の表示に Web コントロールの Table クラスを利用しています。

フォルダ	RegistForm
ソリューションファイル	RegistForm.sln
スタートページ	RegistFormWebSite¥Default.aspx
ソースコード	RegistFormWebSite¥ConfirmForm.aspx.cobx RegistFormWebSite¥Default.aspx.cobx RegistFormWebSite¥SearchZipCodeForm.aspx.cobx RegistFormWebSite¥SucceededForm.aspx.cobx

サンプルを動かす前に、「郵便番号検索サービス」を IIS で公開しておく必要があります。 IIS での公開については、COBOL Web サービス アプリケーションの「郵便番号検索サービス」を参照してください。

COBOL Web サービス アプリケーション

1. 郵便番号検索サービス

   日本の 7 桁の郵便番号を入力して、索引ファイルから住所を検索する Web サービスの例です。 既存の郵便番号検索 COBOL ロジックを NetCOBOL for .NET プラットフォームに移行したものを、Web サービス化しています。 既存の COBOL ロジックは、シフト JIS 形式の郵便番号と住所が登録されている索引ファイルを使って住所を検索するため、翻訳オプション RCS(SJIS)を指定して翻訳します。 (この索引ファイルは、NetCOBOL for .NET Visual Studio プロジェクトサンプルアプリケーションの「CSV to Indexed File」サンプルアプリケーションを使って作成することができます。作成した索引ファイルは、ZipCodeServiceWebSite/App_Data フォルダに格納します。) Web サービスには、7 桁の郵便番号と、複数の郵便番号がある場合に何番目の住所を求めるかを指定します。入力されたデータは、既存の COBOL ロジックに渡され住所が検索されます。 既存の COBOL ロジックから返却された検索結果はシフト JIS コード形式であるため、UNICODE-OF 関数で Unicode データに変換します。 また、検索結果には英数字項目データも含まれるため、NATIONAL-OF 関数を使用して日本語項目データに変換し、 STRING 文を使用して次の形式の情報を日本語項目データに格納します。

```
結果コード(1 文字),レコード位置(9 桁の符号付数字列),郵便番号(7 桁の数字列),都道府県名,市区町村名,町域名
```

   Web サービスはこの検索結果を XML 形式で返却します。

フォルダ	ZipCodeService
ソリューションファイル	ZipCodeService.sln
Web サービス	ZipCodeServiceWebSite¥ZipCodeService.asmx
ソースコード	ZipCodeLib¥ZIP2ADDRLINK.CBL ZipCodeLib¥ZipCode.cob ZipCodeLib¥ZIPCODESEARCH.COB ZipCodeServiceWebSite¥App_Code¥ZipCodeService.cob

IIS を使用して郵便番号検索サービスを公開する

他のサンプルアプリケーションで郵便番号検索サービスを使用するには、IIS を使用して郵便番号検索サービスを公開する必要があります。ASP.NET Web アプリケーションの実行のための事前準備を参照し、IIS が使用できる環境にしてください。

サンプルでは、郵便番号検索サービスを以下の URL で公開することを想定しています。 他の URL で公開した場合には、このサービスを使用するサンプル側で Web 参照の設定を変更してください。

```
http://localhost/ZipCodeService/ZipCodeService.asmx
```

以下の手順を行うことで、郵便番号検索サービスを IIS ルート下へ配置し、公開することができます。

1. Visual Studio を管理者権限で起動します。
2. ソリューションエクスプローラーで、郵便番号検索サービスの Web サイト (ZipCodeServiceWebSite)を選択します。
3. [ビルド]-[Web App の発行] メニューを選択し、[Web を発行] ダイアログを表示します。
4. [プロファイル]ページでは、発行先として[カスタム]をクリックし、 [新しいカスタム プロファイル]ダイアログで、プロファイル名を指定します。 ここでは、「Sample」と入力します。

Microsoft Azure への発行はサポートしていません。

5. [OK]ボタンをクリックします。[接続]ページでは、以下を指定します。

発行方法	「Web Deploy」を選択します。
サーバー	「localhost」を入力します。
サイト名	「Default Web Site/ZipCodeService」を入力します。

6. [次へ]をクリックします。
7. [設定]ページでは、以下を指定します。

構成	「Release」を選択します。

8. [次へ]をクリックします。
9. [発行]をクリックします。

[接続]に指定したサイトがすでに存在している場合、そのサイト内のデータはすべて削除されます。 このサイトを別の用途で使用している場合には他の場所を指定してください。

NCOBU00944

ソースコード：CounterWebSite¥Default.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "_Default" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-CONVERT AS "System.Convert"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-INT32 AS "System.Int32"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-VIEWSTATE AS "ViewState"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 MYCOUNTER BINARY-LONG.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   IF PROP-ISPOSTBACK OF SELF NOT = B'0'
      EXIT METHOD.
   MOVE 0 TO MYCOUNTER.
   INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"MyCounter" BY VALUE MYCOUNTER.
END METHOD PAGE-LOAD.

METHOD-ID. Button1_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 MYCOUNTER BINARY-LONG.
01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" (N"MyCounter").
   SET MYCOUNTER TO WORK-OBJ AS CLASS-INT32.
   ADD 1 TO MYCOUNTER.
   SET WORK-STRING TO CLASS-CONVERT::"ToString" (MYCOUNTER).
   SET PROP-TEXT OF Label1 TO WORK-STRING.
   INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"MyCounter" BY VALUE MYCOUNTER.
END METHOD Button1_Click.

END OBJECT.
END CLASS CLASS-THIS.
```

ソースコード：MultiPageWebSite¥Default.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "_Default"
    INHERITS CLASS-PAGE IS PARTIAL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS CLASS-PAGE AS "System.Web.UI.Page"
    CLASS CLASS-OBJECT AS "System.Object"
    CLASS CLASS-EVENTARGS AS "System.EventArgs"
    .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.

END METHOD PAGE-LOAD.

END OBJECT.
END CLASS CLASS-THIS.
```

NCOBU00946

ソースコード：MultiPageWebSite¥SecondPage.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "SecondPage" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY INPUTTEXT AS "InputText"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-LENGTH AS "Length"
   PROPERTY PROP-PARAMS AS "Params"
   PROPERTY PROP-REQUEST AS "Request"
   PROPERTY PROP-SERVER AS "Server"
   PROPERTY PROP-TEXT AS "Text"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. GET PROPERTY INPUTTEXT AS "InputText".
DATA DIVISION.
LINKAGE SECTION.
01 RET-VAL OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION RETURNING RET-VAL.
   SET RET-VAL TO PROP-TEXT OF TextBoxInput.
END METHOD.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   IF PROP-ISPOSTBACK OF SELF = B"0" THEN
      SET PROP-TEXT OF LabelParam TO
          PROP-PARAMS OF PROP-REQUEST OF SELF ::"get_Item" (N"Param").
END METHOD PAGE-LOAD.

METHOD-ID. ButtonNext_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   SET WORK-STRING TO PROP-TEXT OF TextBoxInput.
   IF PROP-LENGTH OF WORK-STRING = 0 THEN
      SET PROP-TEXT OF LabelMessage TO N"入力フィールドに文字列を入力してください。"
      EXIT METHOD
   END-IF.
   INVOKE PROP-SERVER OF SELF "Transfer" USING N"ThirdPage.aspx".
END METHOD ButtonNext_Click.

METHOD-ID. ButtonPrev_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   INVOKE PROP-SERVER OF SELF "Transfer" USING N"Default.aspx".
END METHOD ButtonPrev_Click.

END OBJECT.
END CLASS CLASS-THIS.
```

ソースコード：MultiPageWebSite¥ThirdPage.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "ThirdPage" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-SECONDPAGE AS "SecondPage"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   INTERFACE INTERFACE-IHTTPHANDLER AS "System.Web.IHttpHandler"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY PROP-HANDLER AS "Handler"
   PROPERTY PROP-INPUTTEXT AS "InputText"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-SERVER AS "Server"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 IHTTPHANDLER OBJECT REFERENCE INTERFACE-IHTTPHANDLER.
01 SECONDPAGE OBJECT REFERENCE CLASS-SECONDPAGE.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   IF PROP-ISPOSTBACK OF SELF = B"0" THEN
      SET IHTTPHANDLER TO PROP-HANDLER OF PROP-CONTEXT OF SELF
      IF IHTTPHANDLER = NULL THEN
         EXIT METHOD
      END-IF
      SET SECONDPAGE TO IHTTPHANDLER AS CLASS-SECONDPAGE
      IF SECONDPAGE NOT = NULL THEN
         SET PROP-TEXT OF LabelDisplayProperty TO PROP-INPUTTEXT OF SECONDPAGE
      END-IF
   END-IF.
END METHOD PAGE-LOAD.

METHOD-ID. ButtonPrev_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   INVOKE PROP-SERVER OF SELF "Transfer" USING N"secondpage.aspx".
END METHOD ButtonPrev_Click.

END OBJECT.
END CLASS CLASS-THIS.
```

NCOBU00948

ソースコード：CreateControlWebSite¥Default.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "_Default"
    INHERITS CLASS-PAGE IS PARTIAL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS CLASS-EVENTARGS AS "System.EventArgs"
    DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
    CLASS CLASS-OBJECT AS "System.Object"
    CLASS CLASS-STRING AS "System.String"
    CLASS CLASS-PAGE AS "System.Web.UI.Page"
    CLASS CLASS-BUTTON AS "System.Web.UI.WebControls.Button"
    PROPERTY PROP-CONTROLS AS "Controls"
    PROPERTY PROP-ITEMS AS "Items"
    PROPERTY PROP-TEXT AS "Text"
    .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-B PIC 1.
01 BUTTON OBJECT REFERENCE CLASS-BUTTON.
01 EVENTHANDLER OBJECT REFERENCE DELEGATE-EVENTHANDLER.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
* ページは、ラウンド トリップごとに毎回作成される。
* したがって、動的コントロールは、毎回ページのロード時に作成しなければならない。

* 「項目の追加」ボタンの生成
    SET BUTTON TO CLASS-BUTTON::"NEW" ().
    IF BUTTON = NULL THEN
       EXIT METHOD.
    INVOKE PROP-CONTROLS OF SELF "Add" USING BUTTON.
    INVOKE PROP-CONTROLS OF PlaceHolder1 "Add" USING BUTTON.
    SET PROP-TEXT OF BUTTON TO N"項目の追加".
    INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"Button_Click"
           RETURNING EVENTHANDLER.
    INVOKE BUTTON "add_Click" USING BY VALUE EVENTHANDLER.

* 「項目のクリア」ボタンの生成
    SET BUTTON TO CLASS-BUTTON::"NEW" ().
    IF BUTTON = NULL THEN
       EXIT METHOD.
    INVOKE PROP-CONTROLS OF SELF "Add" USING BUTTON.
    INVOKE PROP-CONTROLS OF PlaceHolder2 "Add" USING BUTTON.
    SET PROP-TEXT OF BUTTON TO N"項目のクリア".
    INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
           BY VALUE N"ButtonClear_Click" RETURNING EVENTHANDLER.
    INVOKE BUTTON "add_Click" USING BY VALUE EVENTHANDLER.
END METHOD PAGE-LOAD.

METHOD-ID. BUTTON_CLICK AS "Button_Click" PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
* 入力フィールド(TextBox1)の内容を、ListBox1 の項目として追加する。
    SET WORK-STRING TO PROP-TEXT OF TextBox1.
    INVOKE PROP-ITEMS OF ListBox1 "Add" USING WORK-STRING.
END METHOD BUTTON_CLICK.

METHOD-ID. BUTTONCLEAR_CLICK AS "ButtonClear_Click" PRIVATE.
```

```
       DATA DIVISION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      * ListBox1の項目を全て削除する。
           INVOKE PROP-ITEMS OF ListBox1 "Clear".
       END METHOD BUTTONCLEAR_CLICK.

       END OBJECT.
       END CLASS CLASS-THIS.
```

NCOBU00949

ソースコード：ZipCodeWebFormsWebSite¥Default.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "_Default" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-CONVERT AS "System.Convert"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY PROP-SERVER AS "Server"
   PROPERTY PROP-TEXT AS "Text"
   .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ZIPCODE BINARY-LONG PROPERTY.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.

END METHOD PAGE-LOAD.

METHOD-ID. Button1_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   SET WORK-STRING TO PROP-TEXT OF TextBox1.
   SET ZIPCODE TO CLASS-CONVERT::"ToInt32" (WORK-STRING).
   INVOKE PROP-SERVER OF SELF "Transfer" USING N"SelectAddressForm.aspx"
   EXIT METHOD.
END METHOD Button1_Click.

END OBJECT.
END CLASS CLASS-THIS.
```

ソースコード：
ZipCodeWebFormsWebSite¥ResultForm.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "ResultForm" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-SELECTADDRESSWEBFORM AS "SelectAddressForm"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   INTERFACE INTERFACE-IHTTPHANDLER AS "System.Web.IHttpHandler"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY PROP-HANDLER AS "Handler"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-SERVER AS "Server"
   PROPERTY PROP-TEXT AS "Text"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 IHTTPHANDLER OBJECT REFERENCE INTERFACE-IHTTPHANDLER.
01 SELECTWEBFORM OBJECT REFERENCE CLASS-SELECTADDRESSWEBFORM.
01 WORK-N PIC N(80).
01 L-ADDRESS-INFO.
  02 L-CODE   PIC N.
  02 L-POS    PIC N(10).
  02 L-ZIPCODE PIC N(7).
  02 L-TODOUFUKEN PIC N(5).
  02 L-SIKUCHOUSON PIC N(20).
  02 L-CHOUIKI    PIC N(40).
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   IF PROP-ISPOSTBACK OF SELF NOT = B"0" THEN
      SET PROP-TEXT OF TextBox1 TO NULL
      EXIT METHOD
   END-IF
   SET IHTTPHANDLER TO PROP-HANDLER OF PROP-CONTEXT OF SELF.
   SET SELECTWEBFORM TO IHTTPHANDLER AS CLASS-SELECTADDRESSWEBFORM.
   SET WORK-N TO ADDRESSINFO OF SELECTWEBFORM.
   UNSTRING WORK-N DELIMITED BY N","
     INTO L-CODE
          L-POS
          L-ZIPCODE
          L-TODOUFUKEN
          L-SIKUCHOUSON
          L-CHOUIKI
   END-UNSTRING.
   MOVE SPACE TO WORK-N.
   STRING L-TODOUFUKEN DELIMITED BY SPACE
          N"  " DELIMITED BY SIZE
          L-SIKUCHOUSON DELIMITED BY SPACE
          N"  " DELIMITED BY SIZE
          L-CHOUIKI DELIMITED BY SPACE
     INTO WORK-N.
   SET PROP-TEXT OF LabelZipCode TO L-ZIPCODE.
   SET PROP-TEXT OF TextBox1 TO WORK-N.
END METHOD PAGE-LOAD.

METHOD-ID. Button1_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
```

```
       PROCEDURE DIVISION USING BY VALUE sender e.
           INVOKE PROP-SERVER OF SELF "Transfer" USING "Default.aspx"
       END METHOD Button1_Click.

       END OBJECT.
       END CLASS CLASS-THIS.
```

ソースコード：
ZipCodeWebFormsWebSite¥SelectAddressForm.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "SelectAddressForm" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-SEARCHWEBFORM AS "_Default"
   CLASS CLASS-CONVERT AS "System.Convert"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-INT32 AS "System.Int32"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   INTERFACE INTERFACE-IHTTPHANDLER AS "System.Web.IHttpHandler"
   CLASS CLASS-HTMLTABLECELL AS "System.Web.UI.HtmlControls.HtmlTableCell"
   CLASS CLASS-HTMLTABLECELLCOLLECTION AS
        "System.Web.UI.HtmlControls.HtmlTableCellCollection"
   CLASS CLASS-HTMLTABLEROW AS "System.Web.UI.HtmlControls.HtmlTableRow"
   CLASS CLASS-HTMLTABLEROWCOLLECTION AS
        "System.Web.UI.HtmlControls.HtmlTableRowCollection"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   CLASS CLASS-FONTUNIT AS "System.Web.UI.WebControls.FontUnit"
   CLASS CLASS-RADIOBUTTON AS "System.Web.UI.WebControls.RadioButton"
   CLASS CLASS-ZIPCODESERVICEPROXY AS "ZipCodeWebService.ZipCodeService"
   PROPERTY PROP-CELLS AS "Cells"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-COUNT AS "Count"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-GROUPNAME AS "GroupName"
   PROPERTY PROP-HANDLER AS "Handler"
   PROPERTY PROP-INNERTEXT AS "InnerText"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-ROWS AS "Rows"
   PROPERTY PROP-SERVER AS "Server"
   PROPERTY PROP-SIZE AS "Size"
   PROPERTY PROP-TABINDEX AS "TabIndex"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-VIEWSTATE AS "ViewState"
   .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ADDRESSINFO OBJECT REFERENCE CLASS-STRING PUBLIC.
01 WORK-N PIC N(80).
01 WORK-STR OBJECT REFERENCE CLASS-STRING.
01 SELECT-RADIO-IDX BINARY-LONG.
01 L-ADDRESS-INFO.
  02 L-CODE   PIC N.
  02 L-POS    PIC N(10).
  02 L-ZIPCODE PIC N(7).
  02 L-TODOUFUKEN PIC N(5).
  02 L-SIKUCHOUSON PIC N(20).
  02 L-CHOUIKI    PIC N(40).
01 ROWCOUNT BINARY-LONG.
01 ROWS BINARY-LONG.
01 COLCOUNT BINARY-LONG.
01 TABLEADDRESSINFO.
  02 TODOUFUKEN PIC N(5).
  02 SIKUCHOUSON PIC N(20).
  02 CHOUIKI    PIC N(40).
01 TABLEVIEWSTATEKEYNAME.
  02 KEYNAME-ROW PIC X(3) VALUE "Row".
  02 KEYNAME-NUM PIC 9(3).
01 TABLEVIEWSTATEKEYNAME-X REDEFINES TABLEVIEWSTATEKEYNAME PIC X(6).
01 TABLEROWCOLLECTION OBJECT REFERENCE CLASS-HTMLTABLEROWCOLLECTION.
01 TABLECELLCOLLECTION OBJECT REFERENCE CLASS-HTMLTABLECELLCOLLECTION.
01 TABLEROW OBJECT REFERENCE CLASS-HTMLTABLEROW.
01 TABLECELL OBJECT REFERENCE CLASS-HTMLTABLECELL.
```

```
       PROCEDURE DIVISION.

       METHOD-ID. PAGE-LOAD AS "Page_Load".
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 SEARCHWEBFORM OBJECT REFERENCE CLASS-SEARCHWEBFORM.
       01 ZIPCODESERVICE OBJECT REFERENCE CLASS-ZIPCODESERVICEPROXY.
       01 ZIPCODE BINARY-LONG.
       01 RECPOS  BINARY-LONG.
       01 RET-ADDRESS OBJECT REFERENCE CLASS-STRING.
       01 IHTTPHANDLER OBJECT REFERENCE INTERFACE-IHTTPHANDLER.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      * ポストバック時には Table の内容を復元する
           IF PROP-ISPOSTBACK OF SELF NOT = B"0" THEN
              INVOKE SELF "RestoreTableRows"
              EXIT METHOD
           END-IF

      * ViewState に行数の初期値を保存しておく
           INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"RowCount" BY VALUE 0.

      * 郵便番号検索 Web サービスの Proxy の準備
           INVOKE CLASS-ZIPCODESERVICEPROXY "NEW" RETURNING ZIPCODESERVICE
           IF ZIPCODESERVICE = NULL THEN
              EXIT METHOD
           END-IF

      * 郵便番号入力フォームに設定された郵便番号を取り出す
           SET IHTTPHANDLER TO PROP-HANDLER OF PROP-CONTEXT OF SELF.
           SET SEARCHWEBFORM TO IHTTPHANDLER AS CLASS-SEARCHWEBFORM.
           MOVE ZIPCODE OF SEARCHWEBFORM TO ZIPCODE.

      * 郵便番号検索 Web サービスを呼び出して Table に検索結果を表示
           MOVE 0 TO RECPOS.
           SET TABLEROWCOLLECTION TO PROP-ROWS OF Table1.
           PERFORM WITH TEST AFTER UNTIL L-CODE NOT = N"C"
              *> Web サービスの呼び出し
              INVOKE ZIPCODESERVICE "ZipCode7ToAddress" USING BY VALUE ZIPCODE
                     BY VALUE RECPOS RETURNING RET-ADDRESS
              *> カンマで区切られた文字列が結果として返ってくるので分割する
              SET WORK-N TO RET-ADDRESS
              UNSTRING WORK-N DELIMITED BY N","
                 INTO L-CODE
                      L-POS
                      L-ZIPCODE
                      L-TODOUFUKEN
                      L-SIKUCHOUSON
                      L-CHOUIKI
              END-UNSTRING
              EVALUATE L-CODE
                WHEN N"0" THRU N"C"
                  *> 検索成功(0 の場合は検索終了、C の場合はまだレコードがある)
                  INVOKE SELF "AddTableRow" USING L-TODOUFUKEN L-SIKUCHOUSON L-CHOUIKI
                  SET WORK-STR TO L-POS
                  SET RECPOS TO CLASS-CONVERT::"ToInt32" (WORK-STR)
                WHEN N"N"
                  SET PROP-TEXT OF Label1 TO N"住所がみつかりませんでした。"
                WHEN N"E"
                  SET PROP-TEXT OF Label1 TO N"検索処理でエラーが発生しました。"
              END-EVALUATE
           END-PERFORM.

      * Table の内容を ViewState に保存
           INVOKE SELF "SaveTableRows".

      * 郵便番号を ViewState に保存
           INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"ZipCode" BY VALUE ZIPCODE.

           EXIT METHOD.
       END METHOD PAGE-LOAD.

       METHOD-ID. Button1_Click PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
```

```
       PROCEDURE DIVISION USING BY VALUE sender e.
           IF SELECT-RADIO-IDX = 0 THEN
              SET PROP-TEXT OF Label1 TO N"住所を選択してから[OK]を押してください。"
              EXIT METHOD
           END-IF.
           INVOKE PROP-SERVER OF SELF "Transfer" USING "ResultForm.aspx".
       END METHOD Button1_Click.

       METHOD-ID. Button2_Click PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           INVOKE PROP-SERVER OF SELF "Transfer" USING "Default.aspx".
       END METHOD Button2_Click.

       METHOD-ID. SaveTableRows PRIVATE. *> Table の内容を ViewState に保存する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
      * 現在の ViewState の内容をクリア
           INVOKE PROP-VIEWSTATE OF SELF "Clear".
      * Tabke の内容を保存
           SET TABLEROWCOLLECTION TO PROP-ROWS OF Table1.
           MOVE PROP-COUNT OF TABLEROWCOLLECTION TO ROWS.
           PERFORM VARYING ROWCOUNT FROM 1 BY 1 UNTIL ROWCOUNT >= ROWS
              MOVE ROWCOUNT TO KEYNAME-NUM
              *> 行の内容をカンマで区切られた一つの文字列にして ViewState に登録
              INVOKE SELF "GetCSVAddress" USING ROWCOUNT RETURNING WORK-N
              INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE TABLEVIEWSTATEKEYNAME-X
                     BY VALUE WORK-N
           END-PERFORM
           INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"RowCount" BY VALUE ROWS.
       END METHOD SaveTableRows.

       METHOD-ID. GetCSVAddress PRIVATE. *> CSV 形式の住所を取得する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 ROWPOS BINARY-LONG.
       01 RET-STR OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION USING ROWPOS RETURNING RET-STR.
           SET TABLEROWCOLLECTION TO PROP-ROWS OF Table1.
           INVOKE TABLEROWCOLLECTION "get_Item" USING ROWPOS RETURNING TABLEROW.
           SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
           INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL.
           SET TODOUFUKEN TO PROP-INNERTEXT OF TABLECELL.
           INVOKE TABLECELLCOLLECTION "get_Item" USING 2 RETURNING TABLECELL.
           SET SIKUCHOUSON TO PROP-INNERTEXT OF TABLECELL.
           INVOKE TABLECELLCOLLECTION "get_Item" USING 3 RETURNING TABLECELL.
           SET CHOUIKI TO PROP-INNERTEXT OF TABLECELL.

           MOVE SPACE TO WORK-N.
           STRING
              TODOUFUKEN
              N","
              SIKUCHOUSON
              N","
              CHOUIKI
              DELIMITED BY SPACE
           INTO WORK-N.

           SET RET-STR TO WORK-N.
       END METHOD GetCSVAddress.

       METHOD-ID. RestoreTableRows PRIVATE. *> Table の内容を ViewState から復元する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
       PROCEDURE DIVISION.
           SET TABLEROWCOLLECTION TO PROP-ROWS OF Table1.
           SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" (N"RowCount").
           SET ROWS TO WORK-OBJ AS CLASS-INT32.
           PERFORM VARYING ROWCOUNT FROM 1 BY 1 UNTIL ROWCOUNT >= ROWS
              MOVE ROWCOUNT TO KEYNAME-NUM
              *> ViewState からカンマで区切られた行の内容を取り出して、Table の行に追加する
              SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" (TABLEVIEWSTATEKEYNAME-X)
              SET WORK-N TO WORK-OBJ AS CLASS-STRING
```

```
            UNSTRING WORK-N DELIMITED BY N","
                INTO TODOUFUKEN
                     SIKUCHOUSON
                     CHOUIKI
            END-UNSTRING
            INVOKE SELF "AddTableRow" USING TODOUFUKEN SIKUCHOUSON CHOUIKI
        END-PERFORM
    END METHOD RestoreTableRows.

    METHOD-ID. AddTableRow PRIVATE. *> Table の末尾に行を追加して住所を設定する。
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01 ROWCOUNT BINARY-LONG.
    01 COLCOUNT BINARY-LONG.
    01 RADIO OBJECT REFERENCE CLASS-RADIOBUTTON.
    01 EVENTHANDLER OBJECT REFERENCE DELEGATE-EVENTHANDLER.
    LINKAGE SECTION.
    01 TODOUFUKEN PIC N(5).
    01 SIKUCHOUSON PIC N(20).
    01 CHOUIKI    PIC N(40).
    PROCEDURE DIVISION USING TODOUFUKEN SIKUCHOUSON CHOUIKI.
    * 現在の Table の行数を求めて、その値を追加する行のインデックスとする
        SET TABLEROWCOLLECTION TO PROP-ROWS OF Table1.
        MOVE PROP-COUNT OF TABLEROWCOLLECTION TO ROWCOUNT.
    * Table に１行追加
        SET TABLEROW TO CLASS-HTMLTABLEROW::"NEW" ()
        INVOKE TABLEROWCOLLECTION "Add" USING TABLEROW
    * 追加された行にセルを四つ追加(インデックスは 0～3)
        SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW
        PERFORM VARYING COLCOUNT FROM 0 BY 1 UNTIL COLCOUNT > 3
            SET TABLECELL TO CLASS-HTMLTABLECELL::"NEW" ()
            INVOKE TABLECELLCOLLECTION "Add" USING TABLECELL
        END-PERFORM
    * 各セルに情報を設定
        SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW

        *> カラム 0 には、行選択用のラジオボタンを追加
        INVOKE TABLECELLCOLLECTION "get_Item" USING 0 RETURNING TABLECELL
        SET RADIO TO CLASS-RADIOBUTTON::"NEW" ()
        IF RADIO NOT = NULL THEN
            SET PROP-TABINDEX OF RADIO TO ROWCOUNT
            SET PROP-GROUPNAME OF RADIO TO "SelectAddress"
            SET PROP-SIZE OF PROP-FONT OF RADIO TO CLASS-FONTUNIT::"Parse" (N"XXLarge")
            INVOKE PROP-CONTROLS OF SELF "Add" USING RADIO
            INVOKE PROP-CONTROLS OF TABLECELL "Add" USING RADIO
            INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                   BY VALUE N"RadioButton1_CheckedChanged" RETURNING EVENTHANDLER
            INVOKE RADIO "add_CheckedChanged" USING BY VALUE EVENTHANDLER
        END-IF

        *> カラム１には、都道府県名を追加
        INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
        SET PROP-INNERTEXT OF TABLECELL TO TODOUFUKEN

        *> カラム２には、市区町村名を追加
        INVOKE TABLECELLCOLLECTION "get_Item" USING 2 RETURNING TABLECELL
        SET PROP-INNERTEXT OF TABLECELL TO SIKUCHOUSON

        *> カラム３には、町域名を追加
        INVOKE TABLECELLCOLLECTION "get_Item" USING 3 RETURNING TABLECELL
        SET PROP-INNERTEXT OF TABLECELL TO CHOUIKI

        EXIT METHOD.
    END METHOD AddTableRow.

    METHOD-ID. RadioButton1_CheckedChanged PRIVATE.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    77 RADIO OBJECT REFERENCE CLASS-RADIOBUTTON.
    77 REC-POS  PIC +9(9).
    77 REC-POS-X REDEFINES REC-POS  PIC X(10).
    77 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
    LINKAGE SECTION.
    01 sender OBJECT REFERENCE CLASS-OBJECT.
    01 e OBJECT REFERENCE CLASS-EVENTARGS.
    PROCEDURE DIVISION USING BY VALUE sender e.
        SET RADIO TO sender AS CLASS-RADIOBUTTON.
        MOVE N"C" TO L-CODE.
        SET SELECT-RADIO-IDX TO PROP-TABINDEX OF RADIO.
        MOVE SELECT-RADIO-IDX TO REC-POS.
```

```
          MOVE FUNCTION UCS2-OF(REC-POS-X) TO L-POS.

          SET TABLEROWCOLLECTION TO PROP-ROWS OF Table1.
          INVOKE TABLEROWCOLLECTION "get_Item" USING SELECT-RADIO-IDX RETURNING TABLEROW.
          SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
          INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL.
          SET L-TODOUFUKEN TO PROP-INNERTEXT OF TABLECELL.
          INVOKE TABLECELLCOLLECTION "get_Item" USING 2 RETURNING TABLECELL.
          SET L-SIKUCHOUSON TO PROP-INNERTEXT OF TABLECELL.
          INVOKE TABLECELLCOLLECTION "get_Item" USING 3 RETURNING TABLECELL.
          SET L-CHOUIKI TO PROP-INNERTEXT OF TABLECELL.
          SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" (N"ZipCode").
          SET L-ZIPCODE TO CLASS-CONVERT:: "ToString" (WORK-OBJ).

          MOVE SPACE TO WORK-N.
          STRING
             L-CODE
             N","
             L-POS
             N","
             L-ZIPCODE
             N","
             L-TODOUFUKEN
             N","
             L-SIKUCHOUSON
             N","
             L-CHOUIKI
             DELIMITED BY SPACE
           INTO WORK-N.

          SET ADDRESSINFO TO WORK-N.
      END METHOD RadioButton1_CheckedChanged.

      END OBJECT.
      END CLASS CLASS-THIS.
```

NCOBU00952

ソースコード：RegistFormWebSite¥ConfirmForm.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "ConfirmForm" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   CLASS CLASS-TABLECELL AS "System.Web.UI.WebControls.TableCell"
   CLASS CLASS-TABLECELLCOLLECTION AS "System.Web.UI.WebControls.TableCellCollection"
   CLASS CLASS-TABLEROW AS "System.Web.UI.WebControls.TableRow"
   CLASS CLASS-TABLEROWCOLLECTION AS "System.Web.UI.WebControls.TableRowCollection"
   PROPERTY PROP-CELLS AS "Cells"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-LENGTH AS "Length"
   PROPERTY PROP-RESPONSE AS "Response"
   PROPERTY PROP-SESSION AS "Session"
   PROPERTY PROP-TEXT AS "Text"
   .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-STR OBJECT REFERENCE CLASS-STRING.
01 WORK2-STR OBJECT REFERENCE CLASS-STRING.
01 WORK3-STR OBJECT REFERENCE CLASS-STRING.
01 WORK4-STR OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 UI-ID OBJECT REFERENCE CLASS-STRING.
01 UI-MAIL OBJECT REFERENCE CLASS-STRING.
01 UI-NAME OBJECT REFERENCE CLASS-STRING.
01 UI-ADRS OBJECT REFERENCE CLASS-STRING.
01 UI-TEL  OBJECT REFERENCE CLASS-STRING.
01 TABLEROWCOLLECTION OBJECT REFERENCE CLASS-TABLEROWCOLLECTION.
01 TABLECELLCOLLECTION OBJECT REFERENCE CLASS-TABLECELLCOLLECTION.
01 TABLEROW OBJECT REFERENCE CLASS-TABLEROW.
01 TABLECELL OBJECT REFERENCE CLASS-TABLECELL.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   IF PROP-ISPOSTBACK OF SELF NOT = B"0" THEN
      EXIT METHOD
   END-IF

   INVOKE SELF "GetUserInfo" USING UI-ID UI-MAIL UI-NAME UI-ADRS UI-TEL.

   SET TABLEROWCOLLECTION TO Rows OF tblUserInfo.

   INVOKE TABLEROWCOLLECTION "get_Item" USING 0 RETURNING TABLEROW.
   SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
   INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
   SET PROP-TEXT OF TABLECELL TO UI-ID.

   INVOKE TABLEROWCOLLECTION "get_Item" USING 1 RETURNING TABLEROW.
   SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
   INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
   SET PROP-TEXT OF TABLECELL TO UI-MAIL.

   INVOKE TABLEROWCOLLECTION "get_Item" USING 2 RETURNING TABLEROW.
   SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
   INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
   SET PROP-TEXT OF TABLECELL TO UI-NAME.

   INVOKE TABLEROWCOLLECTION "get_Item" USING 3 RETURNING TABLEROW.
```

```
        SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
        INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
        SET PROP-TEXT OF TABLECELL TO UI-ADRS.

        INVOKE TABLEROWCOLLECTION "get_Item" USING 4 RETURNING TABLEROW.
        SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
        INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
        SET PROP-TEXT OF TABLECELL TO UI-TEL.

        EXIT METHOD.
    END METHOD PAGE-LOAD.

    METHOD-ID. btnSend_Click PROTECTED.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
    01 LEN BINARY-LONG.
    LINKAGE SECTION.
    01 sender OBJECT REFERENCE CLASS-OBJECT.
    01 e OBJECT REFERENCE CLASS-EVENTARGS.
    PROCEDURE DIVISION USING BY VALUE sender e.
   * ここに、実際にユーザ情報を登録するためのコードを記述します。
   * ここではカスタマ ID が未入力の場合、固定のカスタマ ID をセッション状態に格納して、
   * 次のページに移動します。
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbCustomerID").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        MOVE PROP-LENGTH OF WORK-STR TO LEN.
        IF WORK-STR = NULL OR LEN = 0 THEN
            INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbCustomerID"
                   BY VALUE N"000-000-0000".
        INVOKE PROP-RESPONSE OF SELF "Redirect" USING N"SucceededForm.aspx".
    END METHOD btnSend_Click.

    METHOD-ID. GetUserInfo.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
    LINKAGE SECTION.
    01 UI-ID OBJECT REFERENCE CLASS-STRING.
    01 UI-MAIL OBJECT REFERENCE CLASS-STRING.
    01 UI-NAME OBJECT REFERENCE CLASS-STRING.
    01 UI-ADRS OBJECT REFERENCE CLASS-STRING.
    01 UI-TEL OBJECT REFERENCE CLASS-STRING.
    PROCEDURE DIVISION USING UI-ID UI-MAIL UI-NAME UI-ADRS UI-TEL.

        *> カスタマ ID
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbCustomerID").
        SET UI-ID TO WORK-OBJ AS CLASS-STRING.
        *> メールアドレス
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbMailAddress").
        SET UI-MAIL TO WORK-OBJ AS CLASS-STRING.
        *> 氏名
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbFamilyName").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbFirstName").
        SET WORK2-STR TO WORK-OBJ AS CLASS-STRING.
        SET UI-NAME TO CLASS-STRING::"Concat" (WORK-STR "  " WORK2-STR).
        *> 住所
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbZipCode").
        SET WORK2-STR TO WORK-OBJ AS CLASS-STRING.
        SET WORK-STR TO CLASS-STRING::"Concat" ("〒" WORK2-STR).
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbTodoufuken").
        SET WORK2-STR TO WORK-OBJ AS CLASS-STRING.
        SET WORK3-STR TO CLASS-STRING::"Concat" (WORK-STR "  " WORK2-STR).
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbSikuchouson").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbAdrsOther").
        SET WORK2-STR TO WORK-OBJ AS CLASS-STRING.
        SET WORK4-STR TO CLASS-STRING::"Concat" (WORK-STR "  " WORK2-STR).
        SET UI-ADRS TO CLASS-STRING::"Concat" (WORK3-STR "  " WORK4-STR).
        *> 電話番号
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbTel").
        SET UI-TEL TO WORK-OBJ AS CLASS-STRING.

    END METHOD GetUserInfo.

    END OBJECT.
    END CLASS CLASS-THIS.
```

NCOBU00953

ソースコード：RegistFormWebSite¥Default.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "_Default" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-PARAMS AS "Params"
   PROPERTY PROP-REQUEST AS "Request"
   PROPERTY PROP-RESPONSE AS "Response"
   PROPERTY PROP-SESSION AS "Session"
   PROPERTY PROP-TEXT AS "Text"
   .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-SYSBOOL OBJECT REFERENCE CLASS-BOOLEAN.
01 WORK-STR OBJECT REFERENCE CLASS-STRING.
01 WORK2-STR OBJECT REFERENCE CLASS-STRING.
01 WORK3-STR OBJECT REFERENCE CLASS-STRING.
01 WORK4-STR OBJECT REFERENCE CLASS-STRING.
01 WORK-V BINARY-LONG.
01 WORK-N PIC N(80).
01 WORK-X PIC X(80).
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
*> メッセージの初期化
    SET PROP-TEXT OF lblMessage TO NULL.

*> ポストバック時にはそのまま復帰
    IF PROP-ISPOSTBACK OF SELF NOT = B"0" THEN
       EXIT METHOD.

* ページの最初のロードのとき、郵便番号検索ページからのリダイレクト
* でなければそのまま復帰
    MOVE SPACE TO WORK-N.
    SET WORK-N TO PROP-PARAMS OF PROP-REQUEST OF SELF::"get_Item" (N"Zip2Addr").
    IF WORK-N NOT = N"Succeeded" THEN
       EXIT METHOD
    END-IF

* 郵便番号検索ページからのリダイレクトの場合、セッション状態に格納された入力内容を復元
    *> カスタマ ID
    SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbCustomerID").
    SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
    SET PROP-TEXT OF txbCustomerID TO WORK-STR.
    *> メールアドレス
    SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbMailAddress").
    SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
    SET PROP-TEXT OF txbMailAddress TO WORK-STR.
    *> 姓
    SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbFamilyName").
    SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
    SET PROP-TEXT OF txbFamilyName TO WORK-STR.
    *> 名
```

```
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbFirstName").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET PROP-TEXT OF txbFirstName TO WORK-STR.
        *> 郵便番号
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbZipCode").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET PROP-TEXT OF txbZipCode TO WORK-STR.
        *> 都道府県
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbTodoufuken").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET PROP-TEXT OF txbTodoufuken TO WORK-STR.
        *> 市区町村
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbSikuchouson").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET PROP-TEXT OF txbSikuchouson TO WORK-STR.
        *> その他
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbAdrsOther").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET PROP-TEXT OF txbAdrsOther TO WORK-STR.
        *> 電話番号
        SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbTel").
        SET WORK-STR TO WORK-OBJ AS CLASS-STRING.
        SET PROP-TEXT OF txbTel TO WORK-STR.
    END METHOD PAGE-LOAD.

    METHOD-ID. btnNextPage_Click PROTECTED.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    LINKAGE SECTION.
    01 sender OBJECT REFERENCE CLASS-OBJECT.
    01 e OBJECT REFERENCE CLASS-EVENTARGS.
    PROCEDURE DIVISION USING BY VALUE sender e.
   *> カスタマ ID の確認
        INVOKE SELF "CheckCustomerID" USING WORK-STR RETURNING WORK-SYSBOOL.
        IF WORK-SYSBOOL = B"0" THEN
            SET PROP-TEXT OF lblMessage TO N"カスタマ ID が正しくありません。"
            EXIT METHOD
        END-IF
   * 現在のページの入力内容をセッション状態に格納する
        INVOKE SELF "SaveUserInfo".
   *> 入力確認ページにリダイレクト
        INVOKE PROP-RESPONSE OF SELF "Redirect" USING N"ConfirmForm.aspx".
    END METHOD btnNextPage_Click.

    METHOD-ID. btnZip2Addr_Click PROTECTED.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01 QUERYSTRING.
      02 PARAM1 PIC X(9) VALUE "?ZipCode=".
      02 ZIPCODE PIC 9(7).
      02 ZIPCODE-X REDEFINES ZIPCODE PIC X(7).
      02 PARAM2 PIC X(15) VALUE "&AutoPageBack=1".
    01 PARSEZIPCODE.
      02 ZIPCODE-1 PIC 9(3).
      02 FILLER PIC X.
      02 ZIPCODE-2 PIC 9(4).
    01 PARSEZIPCODE-X REDEFINES PARSEZIPCODE PIC X(8).
    LINKAGE SECTION.
    01 sender OBJECT REFERENCE CLASS-OBJECT.
    01 e OBJECT REFERENCE CLASS-EVENTARGS.
    PROCEDURE DIVISION USING BY VALUE sender e.
   * 郵便番号を取り出す
        SET PARSEZIPCODE-X TO PROP-TEXT OF txbZipCode.
        MOVE ZIPCODE-1 TO ZIPCODE-X(1:3).
        MOVE ZIPCODE-2 TO ZIPCODE-X(4:4).
        IF ZIPCODE-X IS NOT NUMERIC THEN
            SET PROP-TEXT OF lblMessage TO
                N"郵便番号が正しくありません。XXX-YYYY 形式の数字を入力してください。"
            EXIT METHOD
        END-IF
   * 現在のページの入力内容をセッション状態に格納する
        INVOKE SELF "SaveUserInfo".
   * 郵便番号をクエリ文字列に追加
        MOVE SPACE TO WORK-X.
        STRING "SearchZipCodeForm.aspx"
              QUERYSTRING
              DELIMITED BY SIZE
          INTO WORK-X.
   * 郵便番号検索ページにリダイレクト
```

```
           INVOKE PROP-RESPONSE OF SELF "Redirect" USING WORK-X.
       END METHOD btnZip2Addr_Click.

       METHOD-ID. SaveUserInfo.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       PROCEDURE DIVISION.
           *> カスタマ ID
           SET WORK-STR TO PROP-TEXT OF txbCustomerID.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbCustomerID" BY VALUE WORK-STR.
           *> メールアドレス
           SET WORK-STR TO PROP-TEXT OF txbMailAddress.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbMailAddress" BY VALUE WORK-STR.
           *> 姓
           SET WORK-STR TO PROP-TEXT OF txbFamilyName.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbFamilyName" BY VALUE WORK-STR.
           *> 名
           SET WORK-STR TO PROP-TEXT OF txbFirstName.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbFirstName" BY VALUE WORK-STR.
           *> 郵便番号
           SET WORK-STR TO PROP-TEXT OF txbZipCode.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbZipCode" BY VALUE WORK-STR.
           *> 都道府県
           SET WORK-STR TO PROP-TEXT OF txbTodoufuken.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbTodoufuken" BY VALUE WORK-STR.
           *> 市区町村
           SET WORK-STR TO PROP-TEXT OF txbSikuchouson.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbSikuchouson" BY VALUE WORK-STR.
           *> その他
           SET WORK-STR TO PROP-TEXT OF txbAdrsOther.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbAdrsOther" BY VALUE WORK-STR.
           *> 電話番号
           SET WORK-STR TO PROP-TEXT OF txbTel.
           INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE N"txbTel" BY VALUE WORK-STR.
       END METHOD SaveUserInfo.

       METHOD-ID. CheckCustomerID.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 CUSTOMERID OBJECT REFERENCE CLASS-STRING.
       01 RESULT OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING CUSTOMERID RETURNING RESULT.
      *> 本来はここでカスタマ ID をチェックするが、本サンプルでは常に True を返す。
           SET RESULT TO B"1".
       END METHOD CheckCustomerID.

       END OBJECT.
       END CLASS CLASS-THIS.
```

ソースコード：RegistFormWebSite¥SearchZipCodeForm.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "SearchZipCodeForm" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-CONVERT AS "System.Convert"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-INT32 AS "System.Int32"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   CLASS CLASS-RADIOBUTTON AS "System.Web.UI.WebControls.RadioButton"
   CLASS CLASS-TABLECELL AS "System.Web.UI.WebControls.TableCell"
   CLASS CLASS-TABLECELLCOLLECTION AS "System.Web.UI.WebControls.TableCellCollection"
   CLASS CLASS-TABLEROW AS "System.Web.UI.WebControls.TableRow"
   CLASS CLASS-TABLEROWCOLLECTION AS "System.Web.UI.WebControls.TableRowCollection"
   CLASS CLASS-ZIPCODESERVICEPROXY AS "ZipCodeWebService.ZipCodeService"
   PROPERTY PROP-ABSOLUTEPATH AS "AbsolutePath"
   PROPERTY PROP-CELLS AS "Cells"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-COUNT AS "Count"
   PROPERTY PROP-GROUPNAME AS "GroupName"
   PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
   PROPERTY PROP-PARAMS AS "Params"
   PROPERTY PROP-REQUEST AS "Request"
   PROPERTY PROP-RESPONSE AS "Response"
   PROPERTY PROP-ROWS AS "Rows"
   PROPERTY PROP-SESSION AS "Session"
   PROPERTY PROP-TABINDEX AS "TabIndex"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-URLREFERRER AS "UrlReferrer"
   PROPERTY PROP-VIEWSTATE AS "ViewState"
   .
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TABLEROWCOLLECTION OBJECT REFERENCE CLASS-TABLEROWCOLLECTION.
01 TABLECELLCOLLECTION OBJECT REFERENCE CLASS-TABLECELLCOLLECTION.
01 TABLEROW OBJECT REFERENCE CLASS-TABLEROW.
01 TABLECELL OBJECT REFERENCE CLASS-TABLECELL.

01 TABLEADDRESSINFO.
  02 TODOUFUKEN PIC N(5).
  02 SIKUCHOUSON PIC N(20).
  02 CHOUIKI    PIC N(40).

01 TABLEVIEWSTATEKEYNAME.
  02 KEYNAME-ROW PIC X(3) VALUE "Row".
  02 KEYNAME-NUM PIC 9(3).
01 TABLEVIEWSTATEKEYNAME-X REDEFINES TABLEVIEWSTATEKEYNAME PIC X(6).
01 ROWS BINARY-LONG.
01 ROWCOUNT BINARY-LONG.
01 SELECT-RADIO-IDX BINARY-LONG.
01 WORK-STR OBJECT REFERENCE CLASS-STRING.
01 WORK-N PIC N(80).

01 L-ADDRESS-INFO.   *> Web サービスで取得する住所情報の形式
  02 L-CODE   PIC N.
  02 L-POS    PIC N(10).
  02 L-ZIPCODE PIC N(7).
  02 L-TODOUFUKEN PIC N(5).
  02 L-SIKUCHOUSON PIC N(20).
  02 L-CHOUIKI    PIC N(40).
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
```

```
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 ZIPCODESERVICE OBJECT REFERENCE CLASS-ZIPCODESERVICEPROXY.
       01 ZIPCODE BINARY-LONG.
       01 ZIPCODE-D PIC 9(7).
       01 ZIPCODE-X REDEFINES ZIPCODE-D PIC X(7).
       01 RECPOS BINARY-LONG.
       01 RET-ADDRESS OBJECT REFERENCE CLASS-STRING.
       01 AUTOPAGEBACK PIC X.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      * ポストバック時には Table の内容を復元する
           IF PROP-ISPOSTBACK OF SELF NOT = B"0" THEN
              INVOKE SELF "RestoreTableRows"
              EXIT METHOD
           END-IF

      * ページの初期化
           *> 前のページの URI を HTTP ヘッダから取得して ViewState に保存
           SET WORK-STR TO PROP-ABSOLUTEPATH OF PROP-URLREFERRER OF PROP-REQUEST OF SELF.
           INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"Referer" BY VALUE WORK-STR.
           *> Table の現在の行数を ViewState に保存
           INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE N"RowCount" BY VALUE 0.
           *> メッセージのクリア
           SET PROP-TEXT OF lblMessage TO NULL.

      * クエリ文字列から郵便番号と検索結果が一つしかなかった場合の動作方法を取得する
           SET ZIPCODE-X TO PROP-PARAMS OF PROP-REQUEST OF SELF::"get_Item" (N"ZipCode").
           MOVE ZIPCODE-D TO ZIPCODE.
           SET WORK-STR TO PROP-PARAMS OF PROP-REQUEST OF SELF::"get_Item" (N"AutoPageBack").
           SET AUTOPAGEBACK TO WORK-STR.

      * 郵便番号検索 Web サービスの Proxy の準備
           INVOKE CLASS-ZIPCODESERVICEPROXY "NEW" RETURNING ZIPCODESERVICE
           IF ZIPCODESERVICE = NULL THEN
              EXIT METHOD
           END-IF

      * 郵便番号検索 Web サービスを呼び出して Table に検索結果を表示
           MOVE 0 TO RECPOS.
           SET TABLEROWCOLLECTION TO PROP-ROWS OF tblAddress.
           PERFORM WITH TEST AFTER UNTIL L-CODE NOT = N"C"
              *> Web サービスの呼び出し
              INVOKE ZIPCODESERVICE "ZipCode7ToAddress" USING BY VALUE ZIPCODE
                     BY VALUE RECPOS RETURNING RET-ADDRESS
              *> カンマで区切られた文字列が結果として返ってくるので分割する
              MOVE SPACE TO WORK-N
              SET WORK-N TO RET-ADDRESS
              UNSTRING WORK-N DELIMITED BY N","
                 INTO L-CODE
                      L-POS
                      L-ZIPCODE
                      L-TODOUFUKEN
                      L-SIKUCHOUSON
                      L-CHOUIKI
              END-UNSTRING
              EVALUATE L-CODE
                WHEN N"0" THRU N"C"
                  *> 検索成功(0 の場合は検索終了、C の場合はまだレコードがある)
                  INVOKE SELF "AddTableRow" USING L-TODOUFUKEN L-SIKUCHOUSON L-CHOUIKI
                  SET WORK-STR TO L-POS
                  SET RECPOS TO CLASS-CONVERT::"ToInt32" (WORK-STR)
                WHEN N"N"
                  SET PROP-TEXT OF lblMessage TO N"住所がみつかりませんでした。"
                WHEN N"E"
                  SET PROP-TEXT OF lblMessage TO N"検索処理でエラーが発生しました。"
              END-EVALUATE
           END-PERFORM.

      * 検索結果が一つしかなければ前のページにリダイレクト
           IF AUTOPAGEBACK = "1" AND RECPOS = 1 THEN
              INVOKE SELF "GetAddressInfo" USING 1 TODOUFUKEN SIKUCHOUSON CHOUIKI
              INVOKE SELF "RedirectPreviousPage"
              EXIT METHOD
           END-IF

      * Table の内容を ViewState に保存
```

```
           INVOKE SELF "SaveTableRows".

           EXIT METHOD.
       END METHOD PAGE-LOAD.

       METHOD-ID. btnOK_Click PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 QUERYSTRING OBJECT REFERENCE CLASS-STRING.
       01 REFERSTRING OBJECT REFERENCE CLASS-STRING.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
      * 住所が選択されていなければメッセージを表示
           IF SELECT-RADIO-IDX = 0 THEN
              SET PROP-TEXT OF lblMessage TO N"住所を選択してから「OK」ボタンを押してください。"
              EXIT METHOD
           END-IF.
      * 前のページにリダイレクト
           INVOKE SELF "RedirectPreviousPage".
       END METHOD btnOK_Click.

       METHOD-ID. SaveTableRows PRIVATE. *> Table の内容を ViewState に保存する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
      * Tabke の内容を保存
           SET TABLEROWCOLLECTION TO PROP-ROWS OF tblAddress.
           MOVE PROP-COUNT OF TABLEROWCOLLECTION TO ROWS.
           PERFORM VARYING ROWCOUNT FROM 1 BY 1 UNTIL ROWCOUNT >= ROWS
              MOVE ROWCOUNT TO KEYNAME-NUM
              *> 行の内容をカンマで区切られた一つの文字列にして ViewState に登録
              INVOKE SELF "GetCSVAddress" USING ROWCOUNT RETURNING WORK-N
              INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE TABLEVIEWSTATEKEYNAME-X
                     BY VALUE WORK-N
           END-PERFORM
           INVOKE PROP-VIEWSTATE OF SELF "Add" USING BY VALUE "RowCount" BY VALUE ROWS.
       END METHOD SaveTableRows.

       METHOD-ID. RestoreTableRows PRIVATE. *> Table の内容を ViewState から復元する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
       PROCEDURE DIVISION.
           SET TABLEROWCOLLECTION TO PROP-ROWS OF tblAddress.
           SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" ("RowCount").
           SET ROWS TO WORK-OBJ AS CLASS-INT32.
           PERFORM VARYING ROWCOUNT FROM 1 BY 1 UNTIL ROWCOUNT >= ROWS
              MOVE ROWCOUNT TO KEYNAME-NUM
              *> ViewState からカンマで区切られた行の内容を取り出して、Table の行に追加する
              MOVE SPACE TO WORK-N
              SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" (TABLEVIEWSTATEKEYNAME-X)
              SET WORK-N TO WORK-OBJ AS CLASS-STRING
              UNSTRING WORK-N DELIMITED BY N","
                 INTO TODOUFUKEN
                      SIKUCHOUSON
                      CHOUIKI
              END-UNSTRING
              INVOKE SELF "AddTableRow" USING TODOUFUKEN SIKUCHOUSON CHOUIKI
           END-PERFORM
       END METHOD RestoreTableRows.

       METHOD-ID. GetCSVAddress PRIVATE. *> CSV 形式の住所を取得する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 ROWPOS BINARY-LONG.
       01 RET-STR OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION USING ROWPOS RETURNING RET-STR.
           INVOKE SELF "GetAddressInfo" USING ROWPOS TODOUFUKEN SIKUCHOUSON CHOUIKI.

           MOVE SPACE TO WORK-N.
           STRING
              TODOUFUKEN
              N","
              SIKUCHOUSON
              N","
              CHOUIKI
              DELIMITED BY SPACE
```

```
           INTO WORK-N.

           SET RET-STR TO WORK-N.
       END METHOD GetCSVAddress.

       METHOD-ID. AddTableRow PRIVATE. *> Table の末尾に行を追加して住所を設定する。
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 ROWCOUNT BINARY-LONG.
       01 COLCOUNT BINARY-LONG.
       01 RADIO OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 EVENTHANDLER OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       LINKAGE SECTION.
       01 TODOUFUKEN PIC N(5).
       01 SIKUCHOUSON PIC N(20).
       01 CHOUIKI    PIC N(40).
       PROCEDURE DIVISION USING TODOUFUKEN SIKUCHOUSON CHOUIKI.
      * 現在の Table の行数を求めて、その値を追加する行のインデックスとする
           SET TABLEROWCOLLECTION TO PROP-ROWS OF tblAddress.
           MOVE PROP-COUNT OF TABLEROWCOLLECTION TO ROWCOUNT.
      * Table に１行追加
           SET TABLEROW TO CLASS-TABLEROW::"NEW" ()
           INVOKE TABLEROWCOLLECTION "Add" USING TABLEROW
      * 追加された行にセルを四つ追加(インデックスは 0～3)
           SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW
           PERFORM VARYING COLCOUNT FROM 0 BY 1 UNTIL COLCOUNT > 3
              SET TABLECELL TO CLASS-TABLECELL::"NEW" ()
              INVOKE TABLECELLCOLLECTION "Add" USING TABLECELL
           END-PERFORM
      * 各セルに情報を設定
           SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW

           *> カラム 0 には、行選択用のラジオボタンを追加
           INVOKE TABLECELLCOLLECTION "get_Item" USING 0 RETURNING TABLECELL
           SET RADIO TO CLASS-RADIOBUTTON::"NEW" ()
           IF RADIO NOT = NULL THEN
              SET PROP-TABINDEX OF RADIO TO ROWCOUNT
              SET PROP-GROUPNAME OF RADIO TO "SelectAddress"
              INVOKE PROP-CONTROLS OF SELF "Add" USING RADIO
              INVOKE PROP-CONTROLS OF TABLECELL "Add" USING RADIO
              INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
                     BY VALUE N"rbtnSelect_CheckedChanged" RETURNING EVENTHANDLER
              INVOKE RADIO "add_CheckedChanged" USING BY VALUE EVENTHANDLER
           END-IF

           *> カラム１には、都道府県名を追加
           INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL
           SET PROP-TEXT OF TABLECELL TO TODOUFUKEN

           *> カラム２には、市区町村名を追加
           INVOKE TABLECELLCOLLECTION "get_Item" USING 2 RETURNING TABLECELL
           SET PROP-TEXT OF TABLECELL TO SIKUCHOUSON

           *> カラム３には、町域名を追加
           INVOKE TABLECELLCOLLECTION "get_Item" USING 3 RETURNING TABLECELL
           SET PROP-TEXT OF TABLECELL TO CHOUIKI

           EXIT METHOD.
       END METHOD AddTableRow.

       METHOD-ID. rbtnSelect_CheckedChanged PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 RADIO OBJECT REFERENCE CLASS-RADIOBUTTON.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           SET RADIO TO sender AS CLASS-RADIOBUTTON.
           SET SELECT-RADIO-IDX TO PROP-TABINDEX OF RADIO.
           INVOKE SELF "GetAddressInfo" USING SELECT-RADIO-IDX TODOUFUKEN SIKUCHOUSON CHOUIKI
       END METHOD rbtnSelect_CheckedChanged.

       METHOD-ID. GetAddressInfo PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 IDX BINARY-LONG.
       01 TODOUFUKEN PIC N(5).
       01 SIKUCHOUSON PIC N(20).
```

```
       01 CHOUIKI    PIC N(40).
       PROCEDURE DIVISION USING IDX TODOUFUKEN SIKUCHOUSON CHOUIKI.
          SET TABLEROWCOLLECTION TO PROP-ROWS OF tblAddress.
          INVOKE TABLEROWCOLLECTION "get_Item" USING IDX RETURNING TABLEROW.
          SET TABLECELLCOLLECTION TO PROP-CELLS OF TABLEROW.
          INVOKE TABLECELLCOLLECTION "get_Item" USING 1 RETURNING TABLECELL.
          SET TODOUFUKEN TO PROP-TEXT OF TABLECELL.
          INVOKE TABLECELLCOLLECTION "get_Item" USING 2 RETURNING TABLECELL.
          SET SIKUCHOUSON TO PROP-TEXT OF TABLECELL.
          INVOKE TABLECELLCOLLECTION "get_Item" USING 3 RETURNING TABLECELL.
          SET CHOUIKI TO PROP-TEXT OF TABLECELL.
       END METHOD GetAddressInfo.

       METHOD-ID. RedirectPreviousPage PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 REFERSTRING OBJECT REFERENCE CLASS-STRING.
       01 QUERYSTRING OBJECT REFERENCE CLASS-STRING.
       01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
       LINKAGE SECTION.
       PROCEDURE DIVISION.
      * セッション状態に住所情報を格納
          *> 都道府県
          INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE "txbTodoufuken" BY VALUE TODOUFUKEN.
          *> 市区町村
          MOVE SPACE TO WORK-N
          STRING SIKUCHOUSON DELIMITED BY SPACE
                 N"  " DELIMITED BY SIZE
                 CHOUIKI DELIMITED BY SPACE
            INTO WORK-N.
          INVOKE PROP-SESSION OF SELF "Add" USING BY VALUE "txbSikuchouson" BY VALUE WORK-N.
      * ViewState に保存されている前のページの URI にリダイレクト
          SET WORK-OBJ TO PROP-VIEWSTATE OF SELF ::"get_Item" (N"Referer").
          SET REFERSTRING TO WORK-OBJ AS CLASS-STRING.
          IF REFERSTRING = NULL
             EXIT METHOD.
          SET QUERYSTRING TO "?Zip2Addr=Succeeded".
          SET WORK-STR TO CLASS-STRING::"Concat" (REFERSTRING QUERYSTRING).
          INVOKE PROP-RESPONSE OF SELF "Redirect" USING WORK-STR.
       END METHOD RedirectPreviousPage.

       END OBJECT.
       END CLASS CLASS-THIS.
```

NCOBU00955

ソースコード：RegistFormWebSite¥SucceededForm.aspx.cobx

```
IDENTIFICATION DIVISION.
*> このクラスの内部名は CLASS-THIS でなければなりません。
CLASS-ID. CLASS-THIS AS "SucceededForm" PARTIAL
   INHERITS CLASS-PAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-PAGE AS "System.Web.UI.Page"
   PROPERTY PROP-RESPONSE AS "Response"
   PROPERTY PROP-SESSION AS "Session"
   PROPERTY PROP-TEXT AS "Text"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WORK-OBJ OBJECT REFERENCE CLASS-OBJECT.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
   SET WORK-OBJ TO PROP-SESSION OF SELF ::"get_Item" (N"txbCustomerID").
   SET PROP-TEXT OF lblCustomerID TO WORK-OBJ AS CLASS-STRING.
END METHOD PAGE-LOAD.

METHOD-ID. btnBack_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
*> 最初のページにリダイレクト
   INVOKE PROP-RESPONSE OF SELF "Redirect" USING N"Default.aspx".
END METHOD btnBack_Click.

END OBJECT.
END CLASS CLASS-THIS.
```

ソースコード：ZipCodeLib¥ZIP2ADDRLINK.CBL

```
000010 77 Ｌ－郵便番号 PIC 9(7).          *> 入力:7桁の郵便番号
000020 77 Ｌ－位置情報 PIC S9(9) COMP-5.  *> 入力:キーが重複している場合、何番目以降から検索を開始するか
000030                                     *>      (１起点で最初からの場合は0)
000040 01 Ｌ－住所レコード.                *> 出力:住所
000050   02 Ｌ－コード   PIC X.            *> E:エラー  N:郵便番号無し  C:郵便番号あり(次あり)
000060                                     *> 0:郵便番号あり(次なし)
000070   02 Ｌ－位置     PIC S9(9).        *> キー(郵便番号)が重複している場合(Ｌ－コードがC)、
000080                                     *>   このレコードが何番目か(１起点)
000090   02 Ｌ－都道府県 PIC N(5).  *> 都道府県名(例:静岡県)
000100   02 Ｌ－市区町村 PIC N(20). *> 市区町村名(例:沼津市)
000110   02 Ｌ－町域     PIC N(40). *> 町域名(例:宮本)
000120
```

NCOBU00957

ソースコード：ZipCodeLib¥ZipCode.cob

```
IDENTIFICATION DIVISION.
CLASS-ID. ZIPCODE AS "ZipCodeLib.ZipCode".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
    CLASS CLASS-EXCEPTION AS "System.Exception"
    CLASS CLASS-STRING AS "System.String"
    PROPERTY PROP-MESSAGE AS "Message"
    .

STATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

METHOD-ID. CANCELPROC AS "CancelProc".
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
    CANCEL "ZIP2ADDR".
END METHOD CANCELPROC.

METHOD-ID. SEARCHADDRESS AS "SearchAddress".
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY "ZIP2ADDRLINK.CBL".
77 W-CODE   PIC X.
77 W-RECPOS PIC +9(9).
77 W-RECPOS-X REDEFINES W-RECPOS PIC X(10).
77 W-ZIPCODE-X PIC X(9).
77 WORK PIC N(80).
77 W-EXCEPTION OBJECT REFERENCE CLASS-EXCEPTION.
77 EXCEPTIONMESSAGE OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
01 ZIPCODE BINARY-LONG.
01 RECPOS  BINARY-LONG.
01 RET-VAL OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION USING BY VALUE ZIPCODE RECPOS RETURNING RET-VAL.
    MOVE ZIPCODE TO Ｌ－郵便番号.
    MOVE RECPOS  TO Ｌ－位置情報.

*   既存の郵便番号検索 COBOL ロジックを呼び出す。
    TRY
        CALL "ZIP2ADDR" USING Ｌ－郵便番号 Ｌ－位置情報 Ｌ－住所レコード END-CALL
    CATCH W-EXCEPTION
        CANCEL "ZIP2ADDR"
        SET EXCEPTIONMESSAGE TO PROP-MESSAGE OF W-EXCEPTION
        SET RET-VAL TO CLASS-STRING::"Concat" (N"E,-99999999," EXCEPTIONMESSAGE N",,")
        EXIT METHOD
    END-TRY.

*   結果を String 型のデータに変換し返却する。
    MOVE FUNCTION UNICODE-OF(Ｌ－コード) TO W-CODE.
    MOVE Ｌ－位置 TO W-RECPOS.
    MOVE FUNCTION UNICODE-OF(W-RECPOS-X) TO W-RECPOS-X.
    MOVE Ｌ－郵便番号 TO W-ZIPCODE-X.
    MOVE FUNCTION UNICODE-OF(W-ZIPCODE-X) TO W-ZIPCODE-X.
    MOVE SPACE TO WORK.
    STRING FUNCTION NATIONAL-OF(W-CODE)  DELIMITED BY SPACE
       N","         DELIMITED BY SIZE
       FUNCTION NATIONAL-OF(W-RECPOS-X)    DELIMITED BY SPACE
       N","         DELIMITED BY SIZE
       FUNCTION NATIONAL-OF(W-ZIPCODE-X)   DELIMITED BY SPACE
       N","         DELIMITED BY SIZE
       FUNCTION UNICODE-OF(Ｌ－都道府県)    DELIMITED BY SPACE
       N","         DELIMITED BY SIZE
       FUNCTION UNICODE-OF(Ｌ－市区町村)    DELIMITED BY SPACE
       N","         DELIMITED BY SIZE
       FUNCTION UNICODE-OF(Ｌ－町域)       DELIMITED BY SPACE
```

```
            INTO WORK
           SET RET-VAL TO WORK.
           EXIT METHOD.
      END METHOD SEARCHADDRESS.

      END STATIC.

      END CLASS ZIPCODE.
```

NCOBU00958

ソースコード：ZipCodeLib¥ZIPCODESEARCH.COB

```
     @OPTIONS RCS(SJIS)
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. ZIP2ADDR.
000030 ENVIRONMENT DIVISION.
000040 INPUT-OUTPUT SECTION.
000050 FILE-CONTROL.
000060    SELECT 郵便番号ファイル ASSIGN TO ZIPCDIDX
000070       FILE STATUS IS ファイルステータス
000080       ORGANIZATION IS INDEXED
000090       ACCESS MODE IS DYNAMIC
000100       RECORD KEY IS Ｒ－郵便番号 OF 郵便番号－レコード WITH DUPLICATES.
000110 DATA DIVISION.
000120 FILE SECTION.
000130 FD  郵便番号ファイル.
000140 01  郵便番号－レコード.
000150   02 Ｒ－郵便番号 PIC 9(7).
000160   02 Ｒ－都道府県 PIC N(5).
000170   02 Ｒ－市区町村 PIC N(20).
000180   02 Ｒ－町域    PIC N(40).
000190 WORKING-STORAGE SECTION.
000200 77 同一郵便番号 PIC 9(7).
000210 77 ファイルオープン状態 PIC S9(9) COMP-5 VALUE 0.
000220 77 IDX PIC S9(9) COMP-5.
000230 77 ファイルステータス PIC XX.
000240 77 Ｗ－町域    PIC N(40).
000250 CONSTANT SECTION.
000260 77 正常   PIC S9(9) COMP-5 VALUE  0.
000270 77 エラー PIC S9(9) COMP-5 VALUE -1.
000280 LINKAGE SECTION.
000290 COPY "ZIP2ADDRLINK.CBL".
000300 PROCEDURE DIVISION USING Ｌ－郵便番号 Ｌ－位置情報 Ｌ－住所レコード.
000310 DECLARATIVES.
000320 ファイルエラー処理 SECTION.
000330     USE AFTER ERROR PROCEDURE ON 郵便番号ファイル.
000340       MOVE  エラー  TO  ファイルオープン状態.
000350 END DECLARATIVES.
000360    PERFORM 初期処理.
000370    PERFORM 郵便番号検索処理.
000380    PERFORM 終了処理.
000390
000400 初期処理.
000410    MOVE SPACE        TO Ｌ－都道府県.
000420    MOVE SPACE        TO Ｌ－市区町村.
000430    MOVE SPACE        TO Ｌ－町域.
000440    MOVE 0           TO Ｌ－位置.
000450    MOVE "N"         TO Ｌ－コード
000460    OPEN INPUT 郵便番号ファイル.
000470    IF ファイルオープン状態 NOT = 正常 THEN
000480       MOVE "E" TO Ｌ－コード
000490       EXIT PROGRAM
000500    END-IF.
000510
000520 終了処理.
000530    CLOSE 郵便番号ファイル.
000540    EXIT PROGRAM.
000550
000560 郵便番号検索処理.
000570    MOVE Ｌ－郵便番号 TO Ｒ－郵便番号
000580                      同一郵便番号.
000590    IF Ｌ－位置情報 = 0 THEN
000600       READ 郵便番号ファイル INVALID KEY PERFORM キー無効処理
000610                          NOT INVALID KEY PERFORM キー有効処理
000620    ELSE
000630       READ 郵便番号ファイル INVALID KEY PERFORM キー無効処理 END-READ
000640       MOVE Ｌ－位置情報 TO IDX
```

```
000650        PERFORM VARYING IDX FROM Ｌ－位置情報 BY -1 UNTIL IDX <= 0 OR Ｌ－コード = "0"
000660           PERFORM 郵便番号次検索処理
000670           ADD 1 TO Ｌ－位置
000680        END-PERFORM
000690        IF Ｒ－郵便番号 NOT = 同一郵便番号 THEN
000700           PERFORM キー無効処理
000710        END-IF
000720        PERFORM キー有効処理
000730     END-IF.
000740     PERFORM 郵便番号次検索処理.
000750
000760 キー無効処理.
000770     MOVE "N" TO Ｌ－コード.
000780     MOVE 0 TO Ｌ－位置
000790     PERFORM 終了処理.
000800
000810 キー有効処理.
000820      MOVE "0" TO Ｌ－コード
000830      ADD 1 TO Ｌ－位置
000840      PERFORM 検索結果出力処理.
000850
000860 郵便番号次検索処理.
000870     READ 郵便番号ファイル NEXT
000880        AT END PERFORM 終了処理
000890     END-READ.
000900     IF Ｒ－郵便番号 = 同一郵便番号 THEN
000910        MOVE "C" TO Ｌ－コード
000920     ELSE
000930        MOVE "0" TO Ｌ－コード
000940     END-IF.
000950
000960 検索結果出力処理.
000970     MOVE Ｒ－都道府県 TO Ｌ－都道府県.
000980     MOVE Ｒ－市区町村 TO Ｌ－市区町村.
000990     MOVE Ｒ－町域 TO Ｗ－町域
001000     UNSTRING Ｗ－町域 DELIMITED BY SPACE INTO Ｗ－町域.
001010     IF Ｗ－町域 = N"以下に掲載がない場合" THEN
001020        MOVE SPACE       TO Ｌ－町域
001030     ELSE
001040        MOVE Ｗ－町域     TO Ｌ－町域
001050     END-IF.
001060
001070 END PROGRAM ZIP2ADDR.
001080
```

NCOBU00959

ソースコード：ZipCodeServiceWebSite¥App_Code¥ZipCodeService.cob

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "ZipCodeService" INHERITS CLASS-WEBSERVICE
   CUSTOM-ATTRIBUTE CA-WEBSERVICE CA-WEBSERVICEBINDING.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-WEBSERVICE
     CLASS CLASS-WEBSERVICEATTRIBUTE
     PROPERTY PROP-NAMESPACE IS N"http://tempuri.org/"
   CUSTOM-ATTRIBUTE CA-WEBSERVICEBINDING
     CLASS CLASS-WEBSERVICEBINDINGATTR
     PROPERTY PROP-CONFORMSTO IS PROP-BASICPROFILE1_1 OF ENUM-WSIPROFILES
   CUSTOM-ATTRIBUTE CA-WEBMETHOD
     CLASS CLASS-WEBMETHODATTRIBUTE
   .
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-WEBMETHODATTRIBUTE AS "System.Web.Services.WebMethodAttribute"
   CLASS CLASS-WEBSERVICE AS "System.Web.Services.WebService"
   CLASS CLASS-WEBSERVICEATTRIBUTE AS "System.Web.Services.WebServiceAttribute"
   CLASS CLASS-WEBSERVICEBINDINGATTR AS "System.Web.Services.WebServiceBindingAttribute"
   ENUM ENUM-WSIPROFILES AS "System.Web.Services.WsiProfiles"
   CLASS CLASS-ZIPCODE AS "ZipCodeLib.ZipCode"
   PROPERTY PROP-BASICPROFILE1_1 AS "BasicProfile1_1"
   PROPERTY PROP-CONFORMSTO AS "ConformsTo"
   PROPERTY PROP-NAMESPACE AS "Namespace"
   .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. NEW.
PROCEDURE DIVISION.
   *> デザインされたコンポーネントを使用する場合、次の行をコメントを解除してください。
   *> INVOKE SELF "InitializeComponent".
END METHOD NEW.

METHOD-ID. DISPOSE AS "Dispose" OVERRIDE IS PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
PROCEDURE DIVISION USING BY VALUE disposing.
* Web サービスの呼び出し中にタイムアウトなどが発生した場合、検索ロジックがロード
* されたままになるため、ここで CANCEL を呼び出す。
   INVOKE CLASS-ZIPCODE "CancelProc".

   INVOKE SUPER "Dispose" USING BY VALUE disposing.
END METHOD DISPOSE.

METHOD-ID. ZIPCODE7TOADDRESS AS "ZipCode7ToAddress" CUSTOM-ATTRIBUTE CA-WEBMETHOD.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 ZIPCODE BINARY-LONG.
01 RECPOS  BINARY-LONG.
01 RET-VAL OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION USING BY VALUE ZIPCODE RECPOS RETURNING RET-VAL.
   INVOKE CLASS-ZIPCODE "SearchAddress" USING BY VALUE ZIPCODE RECPOS RETURNING RET-VAL.
END METHOD ZIPCODE7TOADDRESS.

END OBJECT.
END CLASS CLASS-THIS.
```

NetCOBOL for .NET Windows Communication Foundation サンプル アプリケーション

Windows Communication Foundation(WCF) サンプルアプリケーションは、COBOL による WCF アプリケーションをデモンストレーションします。 サンプルは WCF サブフォルダにあります。 サンプルを試すためには、Visual Studio からソリューションファイルを開いてください。

WCF サービス アプリケーション

1. 郵便番号検索 Windows サービス

フォルダ	ZipCodeWinService
ソリューションファイル	ZipCodeWinService.sln
ソースコード	ZipCodeWinService¥Service1.cob ZipCodeWinService¥Main.cob ZipCodeServiceLib¥IZipCodeService.cob ZipCodeServiceLib¥ZipAddress.cob ZipCodeServiceLib¥ZipCodeService.cob ZipCodeServiceLib¥ZipCodeServiceHost.cob ZipCodeClient¥Form1.cob ZipCodeClient¥Main.cob

説明

WCF を利用した、Windows サービスアプリケーションとクライアントアプリケーションの例です。 このサンプルアプリケーションは都道府県名および市区町村名を指定して郵便番号を検索することができます。詳細については郵便番号検索 Windows サービス サンプルアプリケーションを参照してください。

2. Manage Stock Web Service

フォルダ	WCFManageStock
ソリューションファイル	WCFManageStock.sln
ソースコード	WCFManageStockService¥IManageStockService.cob WCFManageStockService¥ManageStockService.cob WCFManageStockService¥ManageStock.cob WCFManageStockClient¥Form1.cob WCFManageStockClient¥Main.cob

説明

WCF の分散トランザクションを利用した WCF サービスアプリケーションの例です。 この WCF サービスは IIS でホストすることで XML Web サービスとして実行します。 詳細については Manage Stock Web Service サンプルアプリケーションを参照してください。

NCOBU01030

郵便番号検索 Windows サービス サンプルアプリケーション

WCF を利用した、郵便番号検索を行う Windows サービスの例について説明します。

データベースを準備する

郵便番号検索 Windows サービスでは、郵便番号データを使用します。 郵便番号データは SQL Server 上のサンプルデータベース(COBOLSample)の ZipCode 表を使用します。 ZipCode 表は NetCOBOL for .NET データベースアクセスサンプルアプリケーションの ZipCodeDBSetup サンプルをビルド・実行することで作成できます。

Windows サービスを登録して開始する

Windows サービスはシステムに登録する必要があります。 ZipCodeWinService プロジェクトをビルドした後、Visual Studio コマンドプロンプト（管理者権限）で出力フォルダ(例:ZipCodeWinService¥bin¥Debug)に移動し.NET Framework ツールのインストーラー ツール (Installutil.exe)でシステムに登録します。

```
installutil ZipCodeWinService.exe
```

[サービス ログインの設定]ダイアログボックスが表示された場合は、[ユーザ名]ボックスには"コンピュータ名 または ドメイン名¥ユーザ名"の形式で入力します。

サービスが正しく登録された場合は、Visual Studio の[サーバ エクスプローラー]を開き、[サーバ]-[*コンピュータ名*]-[サービス]の下に"ZipCodeWinService"という名前のサービスが表示されます。

サービスが登録されていることを確認したら、"ZipCodeWinService"を選択しコンテキストメニューから[開始]コマンドを選択することでサービスが開始することができます。

コマンドプロンプトからサービスを起動する場合は、以下のコマンドを実行します。

```
net start ZipCodeWinService
```

Windows サービスを停止して登録解除する

Windows サービスを登録解除したい場合は、まずサービスを停止させます。

サービスを停止する場合は登録時と同様に、Visual Studio の[サーバ エクスプローラー]を開き、[サーバ]-[*コンピュータ名*]-[サービス]から"ZipCodeWinService"を選択し、コンテキストメニューから[停止]を選択することによって停止します。

Visual Studio コマンドプロンプト（管理者権限）からサービスを停止する場合は、以下のコマンドを実行します。

```
net stop ZipCodeWinService
```

サービスを停止した後、サービスを登録解除する場合はインストーラツール (Installutil.exe)でサービスの登録を解除します。

```
installutil /u ZipCodeWinService.exe
```

公開されるサービス

サンプルの郵便番号検索サービスは Windows サービスとして実行され、以下のエンドポイントの設定でサービスが公開されることを想定しています。

Address	net.tcp://localhost:8081/ZipCodeService
Binding	NetTcpBinding
Contract	ZipServiceLib.IZipCodeService

サンプルアプリケーションではポート番号として、8081 を想定しています。 環境に応じて適切なポート番号を割り当ててエンドポイントの構成を変更してください。

また、ZipServiceLib.IZipCodeServices サービスコントラクトでは以下のオペレーション(メソッド)を公開します。

オペレーション	説明
GetAddress	7 桁の郵便番号を指定して、住所データ(ZipServiceLib.ZipAddress データコントラクト)オブジェクトを取得します。
GetCities	都道府県名を指定して、市区町村名の一覧を取得します。
GetZipCode	都道府県名および市区町村名を指定して、町域名と対応する郵便番号の一覧を取得します。

ZipServiceLib.IZipCodeServices サービスコントラクトと ZipServiceLib.ZipAddress データコントラクトは、ZipServiceLib というクラスライブラリに定義しています。

クライアントアプリケーションの解説

クライアントアプリケーション(ZipCodeClient)は ZipCodeWinService が起動している場合に正しく実行することができます。

ZipCodeClient では、「都道府県名」を選択すると WCF サービス(IZipCodeSerive)の"GetCities"オペレーションが呼び出され、「市区町村名」のコンボボックスのリストが更新されます。

更に、「市区町村名」を選択し、[検索]ボタンをクリックすると WCF サービス(IZipCodeSerive)の"GetZipCode"オペレーションが呼び出されリストビューに「町域名」と対応する郵便番号の一覧が表示されます。

NCOBU01033

ソースコード：ZipCodeWinService¥Service1.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "ZipCodeWinService.Service1"
   INHERITS CLASS-SERVICEBASE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-SERVICEBASE AS "System.ServiceProcess.ServiceBase"
   CLASS CLASS-STRING AS "System.String"
   CLASS ARRAY-STRING AS "System.String[]"
   CLASS CLASS-SERVICEHOST AS "ZipCodeServiceLib.ZipCodeServiceHost"
   PROPERTY PROP-SERVICENAME AS "ServiceName"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
01 servicehost OBJECT REFERENCE CLASS-SERVICEHOST.
PROCEDURE DIVISION.

METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
PROCEDURE DIVISION USING BY VALUE disposing.
   IF disposing NOT = B"0" AND components NOT = NULL THEN
     INVOKE components "Dispose"
   END-IF.
   INVOKE SUPER "Dispose" USING disposing.
END METHOD DISPOSE.

METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TEMP1 OBJECT REFERENCE CLASS-STRING.
PROCEDURE DIVISION.
*>>IMP BEGIN-EMBEDDED-CODEDOM
*<embedded-codedom>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="Service1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
```

```
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ServiceName" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="ZipCodeService" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
      *
      *Service1
      *
          SET TEMP1 TO N"ZipCodeService"
          SET PROP-SERVICENAME OF SELF TO TEMP1
       END METHOD INITIALIZECOMPONENT.

       METHOD-ID. NEW.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
          INVOKE SELF "InitializeComponent".
       END METHOD NEW.

       METHOD-ID. ONSTART AS "OnStart" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 ARGS OBJECT REFERENCE ARRAY-STRING.
       PROCEDURE DIVISION USING BY VALUE ARGS.
      *  WCF サービスホストを開始します。
          SET servicehost TO CLASS-SERVICEHOST::"NEW".
          INVOKE servicehost "Open".
       END METHOD ONSTART.

       METHOD-ID. ONSTOP AS "OnStop" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
      *  WCF サービスホストを停止します。
          INVOKE servicehost "Close".
       END METHOD ONSTOP.

       END OBJECT.
       END CLASS CLASS-THIS.
```

NCOBU01034

ソースコード：ZipCodeWinService¥Main.cob

```
 IDENTIFICATION DIVISION.
*Windows サービスメインプログラム
 PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
    .
 REPOSITORY.
    CLASS ARRAY-SERVICEBASE AS "System.ServiceProcess.ServiceBase[]"
    CLASS CLASS-SERVICEBASE AS "System.ServiceProcess.ServiceBase"
    CLASS CLASS-SERVICE1 AS "ZipCodeWinService.Service1"
    CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
    .

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 WK-SERVICETORUN OBJECT REFERENCE ARRAY-SERVICEBASE.
 01 WK-SERVICE1 OBJECT REFERENCE CLASS-SERVICE1.
 PROCEDURE DIVISION.
    INVOKE ARRAY-SERVICEBASE "NEW" USING 1 RETURNING WK-SERVICETORUN.
    INVOKE CLASS-SERVICE1 "NEW" RETURNING WK-SERVICE1.
    INVOKE WK-SERVICETORUN "Set" USING 0 WK-SERVICE1.
    INVOKE CLASS-SERVICEBASE "Run" USING WK-SERVICETORUN.
 END PROGRAM MAIN.
```

ソースコード：ZipCodeServiceLib¥IZipCodeService.cob

```
 IDENTIFICATION  DIVISION.
* IZipCodeService サービスコントラクト
 INTERFACE-ID. INTERFACE-SERVICECONTRACT AS "ZipCodeServiceLib.IZipCodeService"
               CUSTOM-ATTRIBUTE IS CA-SERVICECONTRACT.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-SERVICECONTRACT CLASS CLASS-SERVICECONTRACT
    CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT CLASS CLASS-OPERATIONCONTRACT
    .
 REPOSITORY.
    CLASS CLASS-STRING AS "System.String"
    CLASS CLASS-SERVICECONTRACT AS "System.ServiceModel.ServiceContractAttribute"
    CLASS CLASS-OPERATIONCONTRACT AS "System.ServiceModel.OperationContractAttribute"
    CLASS CLASS-ZIPADDRESS AS "ZipCodeServiceLib.ZipAddress"
    CLASS ARRAY-STRING AS "System.String[]"
    .

 PROCEDURE DIVISION.

* 郵便番号を指定して住所情報を取得します
 METHOD-ID. GETADDRESS AS "GetAddress" CUSTOM-ATTRIBUTE IS CA-OPERATIONCONTRACT.
 DATA DIVISION.
 LINKAGE SECTION.
 01 ZIPCODE OBJECT REFERENCE CLASS-STRING.
 01 RETVAL  OBJECT REFERENCE CLASS-ZIPADDRESS.
 PROCEDURE DIVISION USING BY VALUE ZIPCODE RETURNING RETVAL.

 END METHOD GETADDRESS.

* 都道府県名を指定して住所の一覧を取得します。
 METHOD-ID. GETCITIES AS "GetCities" CUSTOM-ATTRIBUTE IS CA-OPERATIONCONTRACT.
 DATA DIVISION.
 LINKAGE SECTION.
 01 都道府県名   PIC N(5).
 01 市区町村一覧.
   02 市区町村数 PIC S9(9) COMP-5.
   02 市区町村名 PIC N(20) OCCURS 200.
 PROCEDURE DIVISION USING 都道府県名 RETURNING 市区町村一覧.

 END METHOD GETCITIES.

* 都道府県名と市区町村名を指定して住所の一覧を取得します。
 METHOD-ID. GETZIPCODE AS "GetZipCode" CUSTOM-ATTRIBUTE IS CA-OPERATIONCONTRACT.
 DATA DIVISION.
 LINKAGE SECTION.
 01 都道府県名 PIC N(5).
 01 市区町村名 PIC N(20).
 01 郵便番号一覧.
   02 町域数   PIC S9(9) COMP-5.
   02 町域名   PIC N(40) OCCURS 100.
   02 郵便番号 PIC X(7)  OCCURS 100.
 PROCEDURE DIVISION USING 都道府県名 市区町村名 RETURNING 郵便番号一覧.

 END METHOD GETZIPCODE.

 END INTERFACE INTERFACE-SERVICECONTRACT.
```

NCOBU01036

ソースコード：ZipCodeServiceLib¥ZipAddress.cob

```
IDENTIFICATION DIVISION.
* ZipAddress データコントラクト
CLASS-ID. CLASS-DATACONTRACT AS "ZipCodeServiceLib.ZipAddress"
          CUSTOM-ATTRIBUTE IS CA-DATACONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-DATACONTRACT CLASS CLASS-DATACONTRACTATTRIBUTE
   CUSTOM-ATTRIBUTE CA-DATAMEMBER0  CLASS CLASS-DATAMEMBERATTRIBUTE
                                PROPERTY PROP-ORDER IS 0
                                PROPERTY PROP-NAME  IS N"ZipCide"
   CUSTOM-ATTRIBUTE CA-DATAMEMBER1  CLASS CLASS-DATAMEMBERATTRIBUTE
                                PROPERTY PROP-ORDER IS 1
                                PROPERTY PROP-NAME  IS N"PrefectureName"
   CUSTOM-ATTRIBUTE CA-DATAMEMBER2  CLASS CLASS-DATAMEMBERATTRIBUTE
                                PROPERTY PROP-ORDER IS 2
                                PROPERTY PROP-NAME  IS N"CityName"
   CUSTOM-ATTRIBUTE CA-DATAMEMBER3  CLASS CLASS-DATAMEMBERATTRIBUTE
                                PROPERTY PROP-ORDER IS 3
                                PROPERTY PROP-NAME  IS N"AreaName"
   .
REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-SERIALIZABLE AS "System.SerializableAttribute"
   CLASS CLASS-DATACONTRACTATTRIBUTE AS
         "System.Runtime.Serialization.DataContractAttribute"
   CLASS CLASS-DATAMEMBERATTRIBUTE AS "System.Runtime.Serialization.DataMemberAttribute"
   PROPERTY PROP-ORDER AS "Order"
   PROPERTY PROP-NAME  AS "Name"
   .

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ZipCode       OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER0 PUBLIC.
01 PrefectureName OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER1 PUBLIC.
01 CityName      OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER2 PUBLIC.
01 AreaName      OBJECT REFERENCE CLASS-STRING CUSTOM-ATTRIBUTE IS CA-DATAMEMBER3 PUBLIC.
END OBJECT.

END CLASS CLASS-DATACONTRACT.
```

ソースコード：ZipCodeServiceLib¥ZipCodeService.cob

```
 IDENTIFICATION DIVISION.
*ZipCodeServiceサービスクラス
 CLASS-ID. CLASS-SERVICE AS "ZipCodeServiceLib.ZipCodeService"
           CUSTOM-ATTRIBUTE IS CA-SERVICEBEHAVIOR.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    CUSTOM-ATTRIBUTE CA-SERVICEBEHAVIOR CLASS CLASS-SERVICEBEHAVIORATTRIBUTE
    PROPERTY PROP-INSTANCECONTEXTMODE IS PROP-PERSESSION OF ENUM-INSTANCECONTEXTMODE
    PROPERTY PROP-CONCURRENCYMODE IS PROP-SINGLE OF ENUM-CONCURRENCYMODE
    .
 REPOSITORY.
    CLASS CLASS-STRING AS "System.String"
    CLASS CLASS-SERVICEBEHAVIORATTRIBUTE AS "System.ServiceModel.ServiceBehaviorAttribute"
    INTERFACE INTERFACE-IZIPCODESERVICE AS "ZipCodeServiceLib.IZipCodeService"
    INTERFACE INTERFACE-IDISPOZABLE AS "System.IDisposable"
    CLASS CLASS-ZIPADDRESS AS "ZipCodeServiceLib.ZipAddress"
    ENUM ENUM-INSTANCECONTEXTMODE AS "System.ServiceModel.InstanceContextMode"
    ENUM ENUM-CONCURRENCYMODE AS "System.ServiceModel.ConcurrencyMode"
    PROPERTY PROP-INSTANCECONTEXTMODE AS "InstanceContextMode"
    PROPERTY PROP-CONCURRENCYMODE AS "ConcurrencyMode"
    PROPERTY PROP-SINGLE AS "Single"
    PROPERTY PROP-PERSESSION AS "PerSession"
    PROPERTY PROP-ZIPCODE AS "ZipCode"
    PROPERTY PROP-PREFECTURENAME AS "PrefectureName"
    PROPERTY PROP-CITYNAME AS "CityName"
    PROPERTY PROP-AREANAME AS "AreaName".

 OBJECT. IMPLEMENTS INTERFACE-IZIPCODESERVICE INTERFACE-IDISPOZABLE.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE           PIC X(5).
 01 SQLMSG             PIC X(256).
 01 データベース状態    PIC 1 VALUE B"0".
   88 データベースオープン    VALUE B"1".
   88 データベースクローズ    VALUE B"0".
 01 市区町村カーソル状態 PIC 1 VALUE B"0".
   88 市区町村カーソルオープン VALUE B"1".
   88 市区町村カーソルクローズ VALUE B"0".
 01 町域カーソル状態    PIC 1 VALUE B"0".
   88 町域カーソルオープン    VALUE B"1".
   88 町域カーソルクローズ    VALUE B"0".
    EXEC SQL END DECLARE SECTION END-EXEC.
 PROCEDURE DIVISION.

*郵便番号を指定して住所情報を取得します。
 METHOD-ID. GETADDRESS AS "GetAddress".
 DATA DIVISION.
 WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 問い合わせ郵便番号 PIC X(7).
 01 郵便番号データ.
   02 郵便番号   PIC X(7).
   02 都道府県名 PIC N(5).
   02 市区町村名 PIC N(20).
   02 町域名     PIC N(40).
    EXEC SQL END DECLARE SECTION END-EXEC.
 LINKAGE SECTION.
 01 ZIPCODE OBJECT REFERENCE CLASS-STRING.
 01 RETVAL OBJECT REFERENCE CLASS-ZIPADDRESS.
 PROCEDURE DIVISION USING BY VALUE ZIPCODE RETURNING RETVAL.
*  データベースに接続します。
    INVOKE SELF "DBOpen".

*  クエリを実行します。
    SET 問い合わせ郵便番号 TO ZIPCODE.
```

```
        EXEC SQL
            SELECT 郵便番号7, 都道府県名, 市区町村名, 町域名 INTO :郵便番号データ
               FROM ZipCode WHERE 郵便番号7 = :問い合わせ郵便番号
        END-EXEC.

    * データコントラクト(ZipAddressクラス)を生成して検索結果を格納します。
        SET RETVAL TO CLASS-ZipAddress::"NEW"
        SET PROP-ZIPCODE OF RETVAL       TO 郵便番号.
        SET PROP-PREFECTURENAME OF RETVAL TO 都道府県名.
        SET PROP-CITYNAME OF RETVAL      TO 市区町村名.
        SET PROP-AREANAME OF RETVAL      TO 町域名.

    *  データベースを切断します。
        INVOKE SELF "DBClose".

     END METHOD GETADDRESS.


    * 都道府県名を指定して住所の一覧を取得します。
     METHOD-ID. GETCITIES AS "GetCities".
     DATA DIVISION.
     WORKING-STORAGE SECTION.
        EXEC SQL BEGIN DECLARE SECTION END-EXEC.
     01 問い合わせ都道府県名 PIC N(5).
     01 検索結果.
       02 市区町村名     PIC N(20).
       02 市区町村名カナ PIC X(25).
        EXEC SQL END DECLARE SECTION END-EXEC.
     LINKAGE SECTION.
     01 都道府県名       PIC N(5).
     01 市区町村一覧.
       02 市区町村数     PIC S9(9) COMP-5.
       02 市区町村名     PIC N(20) OCCURS 200.
     PROCEDURE DIVISION USING 都道府県名 RETURNING 市区町村一覧.
     P-START.
    *  データベースに接続します。
        INVOKE SELF "DBOpen".

        MOVE 都道府県名 TO 問い合わせ都道府県名.

    *  例外事象が発生した場合の動作を指定します。
        EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
        EXEC SQL WHENEVER SQLERROR GO TO :P-END END-EXEC.
    *  カーソルを宣言します。
        EXEC SQL
            DECLARE 市区町村カーソル CURSOR FOR
            SELECT DISTINCT 市区町村名, 市区町村名カナ FROM ZipCode
                           WHERE 都道府県名 = :問い合わせ都道府県名
                           ORDER BY 市区町村名カナ
        END-EXEC.

    *  カーソルをオープンします。
        IF 市区町村カーソルクローズ THEN
            EXEC SQL OPEN 市区町村カーソル END-EXEC
            SET 市区町村カーソルオープン TO TRUE
        END-IF

    *  カーソルを使用してデータを取り出します。
        MOVE 0 TO 市区町村数.
        EXEC SQL FETCH 市区町村カーソル INTO :検索結果  END-EXEC
        PERFORM TEST BEFORE UNTIL SQLSTATE = "02000"
          ADD 1 TO 市区町村数
          MOVE 市区町村名 OF 検索結果 TO 市区町村名 OF 市区町村一覧(市区町村数)
          IF 市区町村数 = 200
    *  取得したデータ件数が200件を超えた場合はメソッドから抜けます。
             EXIT METHOD
          END-IF
          EXEC SQL FETCH 市区町村カーソル INTO :検索結果  END-EXEC
        END-PERFORM

        MOVE "00000" TO SQLSTATE.

    *  例外事象が発生した場合の動作を無効化します。
     P-END.
        EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.
```

```
*  カーソルをクローズします。
    IF 市区町村カーソルオープン THEN
      EXEC SQL CLOSE 市区町村カーソル END-EXEC
      SET 市区町村カーソルクローズ TO TRUE
    END-IF.

 END METHOD GETCITIES.


* 都道府県名と市区町村名を指定して住所の一覧を取得します。
 METHOD-ID. GETZIPCODE AS "GetZipCode".
 DATA DIVISION.
 WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 問い合わせ都道府県名 PIC N(5).
 01 問い合わせ市区町村名 PIC N(20).
 01 検索結果.
   02 町域名    PIC N(40).
   02 郵便番号   PIC X(7).
    EXEC SQL END DECLARE SECTION END-EXEC.
 LINKAGE SECTION.
 01 都道府県名   PIC N(5).
 01 市区町村名   PIC N(20).
 01 郵便番号一覧.
   02 町域数    PIC S9(9) COMP-5.
   02 町域名    PIC N(40) OCCURS 100.
   02 郵便番号   PIC X(7)  OCCURS 100.
 PROCEDURE DIVISION USING 都道府県名 市区町村名 RETURNING 郵便番号一覧.
 P-START.
*  データベースに接続します。
    INVOKE SELF "DBOpen".

    MOVE 都道府県名 TO 問い合わせ都道府県名.
    MOVE 市区町村名 TO 問い合わせ市区町村名.

*  例外事象が発生した場合の動作を指定します。
    EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
    EXEC SQL WHENEVER SQLERROR GO TO :P-END END-EXEC.
*  カーソルを宣言します。
    EXEC SQL
       DECLARE 町域カーソル CURSOR FOR
       SELECT 町域名, 郵便番号 7 FROM ZipCode
                    WHERE 都道府県名 = :問い合わせ都道府県名
                      AND 市区町村名 = :問い合わせ市区町村名
    END-EXEC.

*  カーソルをオープンします。
    IF 町域カーソルクローズ THEN
       EXEC SQL OPEN 町域カーソル END-EXEC
       SET 町域カーソルオープン TO TRUE
    END-IF.

*  カーソルを使用してデータを取り出します。
    MOVE 0 TO 町域数.
    EXEC SQL FETCH 町域カーソル INTO :検索結果  END-EXEC
    PERFORM TEST BEFORE UNTIL SQLSTATE = "02000"
      ADD 1 TO 町域数
      MOVE 町域名 OF 検索結果 TO 町域名 OF 郵便番号一覧(町域数)
      MOVE 郵便番号 OF 検索結果 TO 郵便番号 OF 郵便番号一覧(町域数)
      IF 町域数 = 100
*  取得したデータ件数が 100 件を超えた場合はメソッドから抜けます。
        EXIT METHOD
      END-IF
      EXEC SQL FETCH 町域カーソル INTO :検索結果  END-EXEC
    END-PERFORM
    MOVE "00000" TO SQLSTATE.

*  例外事象が発生した場合の動作を無効化します。
 P-END.
    EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.

*  カーソルをクローズします。
    IF 町域カーソルオープン THEN
      EXEC SQL CLOSE 町域カーソル END-EXEC
      SET 町域カーソルクローズ TO TRUE
```

```
           END-IF.

       END METHOD GETZIPCODE.

      *  オブジェクトが閉じられるときにデータベースを切断します。
       METHOD-ID. DISPOSE AS "Dispose".
       PROCEDURE DIVISION.
           INVOKE SELF "DBClose".
       END METHOD DISPOSE.

      *  データベースをオープンします。
       METHOD-ID. DBOPEN AS "DBOpen" PRIVATE.
       PROCEDURE DIVISION.
           IF データベースクローズ
              EXEC SQL CONNECT TO DEFAULT END-EXEC
              IF SQLSTATE = '00000' OR SQLSTATE = '01000'
                 SET データベースオープン TO TRUE
              END-IF
           END-IF.
       END METHOD DBOPEN.

      *  データベースをクローズします。
       METHOD-ID. DBOPEN AS "DBClose" PRIVATE.
       PROCEDURE DIVISION.
           IF データベースオープン
              EXEC SQL DISCONNECT DEFAULT END-EXEC
              SET データベースクローズ TO TRUE
           END-IF.
       END METHOD DBOPEN.

       END OBJECT.

       END CLASS CLASS-SERVICE.
```

ソースコード：ZipCodeServiceLib¥ZipCodeServiceHost.cob

```
IDENTIFICATION DIVISION.
* ZipCodeServiceHost サービスホスト
CLASS-ID. CLASS-HOST AS "ZipCodeServiceLib.ZipCodeServiceHost" INHERITS CLASS-SERVICEHOST.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
    CLASS CLASS-SERVICEHOST AS "System.ServiceModel.ServiceHost"
    CLASS CLASS-SERVICE AS "ZipCodeServiceLib.ZipCodeService"
    CLASS CLASS-URI AS "System.Uri"
    CLASS CLASS-TYPE AS "System.Type"
    .
OBJECT.
PROCEDURE DIVISION.

METHOD-ID. NEW.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 URI OBJECT REFERENCE CLASS-URI.
01 SERVICE-TYPE OBJECT REFERENCE CLASS-TYPE.
01 SVC OBJECT REFERENCE CLASS-SERVICE.
PROCEDURE DIVISION.
    SET URI TO CLASS-URI::"NEW"("http://localhost:8080/ZipCodeService").
    SET SERVICE-TYPE TO TYPE OF CLASS-SERVICE.
    INVOKE SUPER "NEW" USING SERVICE-TYPE URI
END METHOD NEW.

END OBJECT.
END CLASS CLASS-HOST.
```

NCOBU01037

ソースコード：ZipCodeClient¥Form1.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "ZipCodeClient.Form1"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-BYTE AS "System.Byte"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-FONT AS "System.Drawing.Font"
   ENUM ENUM-FONTSTYLE AS "System.Drawing.FontStyle"
   ENUM ENUM-GRAPHICSUNIT AS "System.Drawing.GraphicsUnit"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-EXCEPTION AS "System.Exception"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-CHANNELFACTORY AS "System.ServiceModel.ChannelFactory<>"
   INTERFACE INTERFACE-ICHANNEL AS "System.ServiceModel.Channels.IChannel"
   ENUM ENUM-COMMUNICATIONSTATE AS "System.ServiceModel.CommunicationState"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-STRINGBUILDER AS "System.Text.StringBuilder"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS CLASS-COLUMNHEADER AS "System.Windows.Forms.ColumnHeader"
   CLASS ARRAY-COLUMNHEADER AS "System.Windows.Forms.ColumnHeader[]"
   CLASS CLASS-COMBOBOX AS "System.Windows.Forms.ComboBox"
   ENUM ENUM-COMBOBOXSTYLE AS "System.Windows.Forms.ComboBoxStyle"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   ENUM ENUM-DOCKSTYLE AS "System.Windows.Forms.DockStyle"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   ENUM ENUM-HORIZONTALALIGNMENT AS "System.Windows.Forms.HorizontalAlignment"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-LISTBOX AS "System.Windows.Forms.ListBox"
   CLASS CLASS-LISTVIEW AS "System.Windows.Forms.ListView"
   CLASS CLASS-COLUMNHEADERCOLLECTION AS
         "System.Windows.Forms.ListView+ColumnHeaderCollection"
   CLASS CLASS-LISTVIEWITEM AS "System.Windows.Forms.ListViewItem"
   CLASS CLASS-LISTVIEWSUBITEM AS "System.Windows.Forms.ListViewItem+ListViewSubItem"
   CLASS CLASS-MASKEDTEXTBOX AS "System.Windows.Forms.MaskedTextBox"
   CLASS CLASS-PANEL AS "System.Windows.Forms.Panel"
   CLASS CLASS-STATUSSTRIP AS "System.Windows.Forms.StatusStrip"
   CLASS CLASS-TEXTBOX AS "System.Windows.Forms.TextBox"
   CLASS ARRAY-TOOLSTRIPITEM AS "System.Windows.Forms.ToolStripItem[]"
   CLASS CLASS-TOOLSTRIPITEMCOLLECTION AS "System.Windows.Forms.ToolStripItemCollection"
   CLASS CLASS-TOOLSTRIPSTATUSLABEL AS "System.Windows.Forms.ToolStripStatusLabel"
   ENUM ENUM-VIEW AS "System.Windows.Forms.View"
   INTERFACE INTERFACE-IZIPCODESERVICE AS "ZipCodeServiceLib.IZipCodeService"
   CLASS CLASS-FACTORY EXPANDS CLASS-CHANNELFACTORY USING INTERFACE-IZIPCODESERVICE
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-BUTTON2 AS "button2"
   PROPERTY PROP-CENTER AS "Center"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-COLUMNHEADER1 AS "columnHeader1"
   PROPERTY PROP-COLUMNHEADER2 AS "columnHeader2"
   PROPERTY PROP-COLUMNS AS "Columns"
   PROPERTY PROP-COMBOBOX1 AS "comboBox1"
   PROPERTY PROP-COMBOBOX2 AS "comboBox2"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-DETAILS AS "Details"
   PROPERTY PROP-DOCK AS "Dock"
   PROPERTY PROP-DROPDOWNLIST AS "DropDownList"
   PROPERTY PROP-DROPDOWNSTYLE AS "DropDownStyle"
```

```
          PROPERTY PROP-FILL AS "Fill"
          PROPERTY PROP-FONT AS "Font"
          PROPERTY PROP-FORMATTINGENABLED AS "FormattingEnabled"
          PROPERTY PROP-FULLROWSELECT AS "FullRowSelect"
          PROPERTY PROP-GRIDLINES AS "GridLines"
          PROPERTY PROP-ITEMHEIGHT AS "ItemHeight"
          PROPERTY PROP-ITEMS AS "Items"
          PROPERTY PROP-LABEL1 AS "label1"
          PROPERTY PROP-LABEL2 AS "label2"
          PROPERTY PROP-LISTBOX1 AS "listBox1"
          PROPERTY PROP-LISTVIEW1 AS "listView1"
          PROPERTY PROP-LOCATION AS "Location"
          PROPERTY PROP-MASK AS "Mask"
          PROPERTY PROP-MASKEDTEXTBOX1 AS "maskedTextBox1"
          PROPERTY PROP-MESSAGE AS "Message"
          PROPERTY PROP-NAME AS "Name"
          PROPERTY PROP-OPENED AS "Opened"
          PROPERTY PROP-PANEL1 AS "panel1"
          PROPERTY PROP-POINT AS "Point"
          PROPERTY PROP-REGULAR AS "Regular"
          PROPERTY PROP-SELECTEDITEM AS "SelectedItem"
          PROPERTY PROP-SELECTEDTEXT AS "SelectedText"
          PROPERTY PROP-SIZE AS "Size"
          PROPERTY PROP-STATE AS "State"
          PROPERTY PROP-STATUSSTRIP1 AS "statusStrip1"
          PROPERTY PROP-SUBITEMS AS "SubItems"
          PROPERTY PROP-TABINDEX AS "TabIndex"
          PROPERTY PROP-TEXT AS "Text"
          PROPERTY PROP-TEXTALIGN AS "TextAlign"
          PROPERTY PROP-TEXTBOX1 AS "textBox1"
          PROPERTY PROP-TOOLSTRIPSTATUSLABEL1 AS "toolStripStatusLabel1"
          PROPERTY PROP-TOP AS "Top"
          PROPERTY PROP-USECOMPATIBLESTATEIMAGEBE AS "UseCompatibleStateImageBehavior"
          PROPERTY PROP-USEVISUALSTYLEBACKCOLOR AS "UseVisualStyleBackColor"
          PROPERTY PROP-VIEW AS "View"
          PROPERTY PROP-WIDTH AS "Width"
          PROPERTY PROP-SELECTEDINDEX AS "SelectedIndex".

       OBJECT.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 panel1 OBJECT REFERENCE CLASS-PANEL.
       01 button1 OBJECT REFERENCE CLASS-BUTTON.
       01 comboBox2 OBJECT REFERENCE CLASS-COMBOBOX.
       01 label2 OBJECT REFERENCE CLASS-LABEL.
       01 comboBox1 OBJECT REFERENCE CLASS-COMBOBOX.
       01 label1 OBJECT REFERENCE CLASS-LABEL.
       01 statusStrip1 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 toolStripStatusLabel1 OBJECT REFERENCE CLASS-TOOLSTRIPSTATUSLABEL.
       01 button2 OBJECT REFERENCE CLASS-BUTTON.
       01 listView1 OBJECT REFERENCE CLASS-LISTVIEW.
       01 columnHeader1 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 columnHeader2 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
       PROCEDURE DIVISION.

       METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING BY VALUE disposing.
          IF disposing NOT = B"0" AND components NOT = NULL THEN
            INVOKE components "Dispose"
          END-IF.
          INVOKE SUPER "Dispose" USING disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP2 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP3 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP4 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP5 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP6 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP7 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP8 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP9 OBJECT REFERENCE CLASS-TOOLSTRIPSTATUSLABEL.
       01 TEMP10 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP11 OBJECT REFERENCE CLASS-COLUMNHEADER.
```

```
       01 TEMP12 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP13 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP14 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP15 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP16 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP17 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP18 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP19 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP20 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP21 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP22 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP23 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP24 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP25 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP26 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP27 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP28 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP29 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP30 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP31 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP32 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP33 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP34 BINARY-LONG.
       01 TEMP35 BINARY-LONG.
       01 TEMP36 OBJECT REFERENCE CLASS-POINT.
       01 TEMP37 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP38 OBJECT REFERENCE CLASS-STRING.
       01 TEMP39 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP40 BINARY-LONG.
       01 TEMP41 BINARY-LONG.
       01 TEMP42 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP43 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP44 BINARY-LONG.
       01 TEMP45 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP46 BINARY-LONG.
       01 TEMP47 BINARY-LONG.
       01 TEMP48 OBJECT REFERENCE CLASS-POINT.
       01 TEMP49 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP50 OBJECT REFERENCE CLASS-STRING.
       01 TEMP51 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP52 BINARY-LONG.
       01 TEMP53 BINARY-LONG.
       01 TEMP54 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP55 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP56 BINARY-LONG.
       01 TEMP57 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP58 OBJECT REFERENCE CLASS-STRING.
       01 TEMP59 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP60 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP61 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP62 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP63 BINARY-LONG.
       01 TEMP64 BINARY-LONG.
       01 TEMP65 OBJECT REFERENCE CLASS-POINT.
       01 TEMP66 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP67 OBJECT REFERENCE CLASS-STRING.
       01 TEMP68 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP69 BINARY-LONG.
       01 TEMP70 BINARY-LONG.
       01 TEMP71 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP72 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP73 BINARY-LONG.
       01 TEMP74 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP75 OBJECT REFERENCE CLASS-STRING.
       01 TEMP76 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP77 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP78 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP79 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP80 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP81 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP82 BINARY-LONG.
       01 TEMP83 BINARY-LONG.
       01 TEMP84 OBJECT REFERENCE CLASS-POINT.
       01 TEMP85 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP86 OBJECT REFERENCE CLASS-STRING.
       01 TEMP87 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP88 BINARY-LONG.
       01 TEMP89 BINARY-LONG.
       01 TEMP90 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP91 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP92 BINARY-LONG.
       01 TEMP93 OBJECT REFERENCE CLASS-COMBOBOX.
```

```
       01 TEMP94 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP95 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP96 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP97 BINARY-LONG.
       01 TEMP98 BINARY-LONG.
       01 TEMP99 OBJECT REFERENCE CLASS-POINT.
       01 TEMP100 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP101 OBJECT REFERENCE CLASS-STRING.
       01 TEMP102 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP103 BINARY-LONG.
       01 TEMP104 BINARY-LONG.
       01 TEMP105 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP106 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP107 BINARY-LONG.
       01 TEMP108 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP109 OBJECT REFERENCE CLASS-STRING.
       01 TEMP110 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP111 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP112 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP113 BINARY-LONG.
       01 TEMP114 BINARY-LONG.
       01 TEMP115 OBJECT REFERENCE CLASS-POINT.
       01 TEMP116 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP117 OBJECT REFERENCE CLASS-STRING.
       01 TEMP118 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP119 BINARY-LONG.
       01 TEMP120 BINARY-LONG.
       01 TEMP121 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP122 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP123 BINARY-LONG.
       01 TEMP124 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP125 OBJECT REFERENCE CLASS-COMBOBOX.
       01 TEMP126 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP127 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP128 BINARY-LONG.
       01 TEMP129 BINARY-LONG.
       01 TEMP130 OBJECT REFERENCE CLASS-POINT.
       01 TEMP131 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP132 OBJECT REFERENCE CLASS-STRING.
       01 TEMP133 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP134 BINARY-LONG.
       01 TEMP135 BINARY-LONG.
       01 TEMP136 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP137 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP138 BINARY-LONG.
       01 TEMP139 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP140 OBJECT REFERENCE CLASS-STRING.
       01 TEMP141 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP142 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP143 OBJECT REFERENCE CLASS-TOOLSTRIPITEMCOLLECTION.
       01 TEMP144 OBJECT REFERENCE CLASS-TOOLSTRIPSTATUSLABEL.
       01 TEMP145 BINARY-LONG.
       01 TEMP146 OBJECT REFERENCE ARRAY-TOOLSTRIPITEM.
       01 TEMP147 BINARY-LONG.
       01 TEMP148 BINARY-LONG.
       01 TEMP149 OBJECT REFERENCE CLASS-POINT.
       01 TEMP150 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP151 OBJECT REFERENCE CLASS-STRING.
       01 TEMP152 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP153 BINARY-LONG.
       01 TEMP154 BINARY-LONG.
       01 TEMP155 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP156 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP157 BINARY-LONG.
       01 TEMP158 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP159 OBJECT REFERENCE CLASS-STRING.
       01 TEMP160 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP161 OBJECT REFERENCE CLASS-STRING.
       01 TEMP162 OBJECT REFERENCE CLASS-TOOLSTRIPSTATUSLABEL.
       01 TEMP163 BINARY-LONG.
       01 TEMP164 BINARY-LONG.
       01 TEMP165 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP166 OBJECT REFERENCE CLASS-TOOLSTRIPSTATUSLABEL.
       01 TEMP167 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP168 OBJECT REFERENCE CLASS-COLUMNHEADERCOLLECTION.
       01 TEMP169 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP170 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP171 BINARY-LONG.
       01 TEMP172 OBJECT REFERENCE ARRAY-COLUMNHEADER.
       01 TEMP173 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP174 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP175 OBJECT REFERENCE CLASS-LISTVIEW.
```

```
       01 TEMP176 BINARY-LONG.
       01 TEMP177 BINARY-LONG.
       01 TEMP178 OBJECT REFERENCE CLASS-POINT.
       01 TEMP179 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP180 OBJECT REFERENCE CLASS-STRING.
       01 TEMP181 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP182 BINARY-LONG.
       01 TEMP183 BINARY-LONG.
       01 TEMP184 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP185 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP186 BINARY-LONG.
       01 TEMP187 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP188 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP189 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP190 OBJECT REFERENCE CLASS-STRING.
       01 TEMP191 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP192 BINARY-LONG.
       01 TEMP193 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP194 OBJECT REFERENCE CLASS-STRING.
       01 TEMP195 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP196 BINARY-LONG.
       01 TEMP197 OBJECT REFERENCE CLASS-COLUMNHEADER.
       01 TEMP198 COMP-1.
       01 TEMP199 COMP-1.
       01 TEMP200 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP201 BINARY-LONG.
       01 TEMP202 BINARY-LONG.
       01 TEMP203 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP204 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP205 OBJECT REFERENCE CLASS-LISTVIEW.
       01 TEMP206 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP207 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP208 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP209 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP210 OBJECT REFERENCE CLASS-STRING.
       01 TEMP211 OBJECT REFERENCE CLASS-STRING.
       01 TEMP212 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP213 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP214 OBJECT REFERENCE CLASS-PANEL.
       01 TEMP215 OBJECT REFERENCE CLASS-STATUSSTRIP.
       01 TEMP216 OBJECT REFERENCE CLASS-STATUSSTRIP.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Panel" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ComboBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ComboBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
```

```
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.StatusStrip" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripStatusLabel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripStatusLabel" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ListView" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
```

```
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ColumnHeader" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ColumnHeader" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
```

```
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="SuspendLayout" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="panel1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="panel1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Dock" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
```

```
      *<string value="System.Windows.Forms.DockStyle" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Top" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
```

```
      *<prop name="Value">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="292" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="95" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
```

```
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="205" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="64" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="75" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
```

```
*<prop name="Value">
*<int32 value="5" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="クリア(&amp;amp;L)" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="UseVisualStyleBackColor" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="True" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
```

```
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button2_Click" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="124" />
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="64" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="75" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="23" />
*</prop>
*</object>
*</prop>
*</object>
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="検索(&amp;amp;S)" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
```

```
*</prop>
*</object>
*<object type="System.CodeDom.CodeAttachEventStatement">
*<prop name="Event">
*<object type="System.CodeDom.CodeEventReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button1" />
*</prop>
*</object>
*</prop>
*<prop name="EventName">
*<string value="Click" />
*</prop>
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="button1_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="comboBox2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="comboBox2" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="DropDownStyle" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ComboBoxStyle" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="DropDownList" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="FormattingEnabled" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
```

```
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="134" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="38" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="comboBox2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="comboBox2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="comboBox2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="145" />
```

```
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="SelectedIndexChanged" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="comboBox2_SelectedIndexChanged" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
```

```
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="AutoSize" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="True" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="41" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="117" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
```

```
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Text" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="市区町村名の選択(&amp;amp;C):" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="comboBox1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
```

```
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="DropDownStyle" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ComboBoxStyle" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="DropDownList" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="FormattingEnabled" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="134" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="8" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="145" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="20" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="comboBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="SelectedIndexChanged" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
```

```
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="comboBox1_SelectedIndexChanged" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="11" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
```

```
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="116" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="都道府県名の選択(&amp;amp;P):" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
```

```
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Items" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="AddRange" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeArrayCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.ToolStripItem" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Initializers" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripStatusLabel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Size">
      *<int32 value="0" />
      *</prop>
      *<prop name="SizeExpression">
      *<null />
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="244" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="292" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="22" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="toolStripStatusLabel1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripStatusLabel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="toolStripStatusLabel1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="toolStripStatusLabel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="17" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Columns" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
```

```
*<string value="AddRange" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeArrayCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.ColumnHeader" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Initializers" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="columnHeader1" />
*</prop>
*</object>
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="columnHeader2" />
*</prop>
*</object>
*</prop>
*<prop name="Size">
*<int32 value="0" />
*</prop>
*<prop name="SizeExpression">
*<null />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="listView1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Dock" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.DockStyle" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
```

```
      *<prop name="FieldName">
      *<string value="Fill" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="FullRowSelect" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="GridLines" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
```

```
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="0" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="95" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="292" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="149" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseCompatibleStateImageBehavior" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="listView1" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="View" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.View" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Details" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="町域" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Width" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="132" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="郵便番号" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="columnHeader2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Width" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="139" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="Form1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleDimensions" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.SizeF" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
*<float32 value="6" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<float32 value="12" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="AutoScaleMode" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeTypeReferenceExpression">
*<prop name="Type">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.AutoScaleMode" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*<prop name="FieldName">
*<string value="Font" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="ClientSize" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="292" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="266" />
*</prop>
*</object>
*</prop>
```

```
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="listView1" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Controls" />
*</prop>
*</object>
*</prop>
*<prop name="MethodName">
*<string value="Add" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="statusStrip1" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeExpressionStatement">
*<prop name="Expression">
*<object type="System.CodeDom.CodeMethodInvokeExpression">
*<prop name="Method">
*<object type="System.CodeDom.CodeMethodReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
```

```
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="Form1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="郵便番号の検索" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Load" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
```

```
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Form1_Load" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="panel1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="statusStrip1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-PANEL "NEW" RETURNING TEMP1
          SET PROP-PANEL1 OF SELF TO TEMP1
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP2
          SET PROP-BUTTON2 OF SELF TO TEMP2
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP3
          SET PROP-BUTTON1 OF SELF TO TEMP3
          INVOKE CLASS-COMBOBOX "NEW" RETURNING TEMP4
          SET PROP-COMBOBOX2 OF SELF TO TEMP4
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP5
          SET PROP-LABEL2 OF SELF TO TEMP5
          INVOKE CLASS-COMBOBOX "NEW" RETURNING TEMP6
          SET PROP-COMBOBOX1 OF SELF TO TEMP6
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP7
          SET PROP-LABEL1 OF SELF TO TEMP7
          INVOKE CLASS-STATUSSTRIP "NEW" RETURNING TEMP8
          SET PROP-STATUSSTRIP1 OF SELF TO TEMP8
          INVOKE CLASS-TOOLSTRIPSTATUSLABEL "NEW" RETURNING TEMP9
          SET PROP-TOOLSTRIPSTATUSLABEL1 OF SELF TO TEMP9
          INVOKE CLASS-LISTVIEW "NEW" RETURNING TEMP10
          SET PROP-LISTVIEW1 OF SELF TO TEMP10
          INVOKE CLASS-COLUMNHEADER "NEW" RETURNING TEMP11
          SET PROP-COLUMNHEADER1 OF SELF TO TEMP11
          INVOKE CLASS-COLUMNHEADER "NEW" RETURNING TEMP12
          SET PROP-COLUMNHEADER2 OF SELF TO TEMP12
          SET TEMP13 TO PROP-PANEL1 OF SELF
          INVOKE TEMP13 "SuspendLayout"
          SET TEMP14 TO PROP-STATUSSTRIP1 OF SELF
          INVOKE TEMP14 "SuspendLayout"
          INVOKE SELF "SuspendLayout"
      *
      *panel1
      *
          SET TEMP17 TO PROP-BUTTON2 OF SELF
          SET TEMP15 TO PROP-PANEL1 OF SELF
          SET TEMP16 TO PROP-CONTROLS OF TEMP15
          INVOKE TEMP16 "Add" USING BY VALUE TEMP17
          SET TEMP20 TO PROP-BUTTON1 OF SELF
          SET TEMP18 TO PROP-PANEL1 OF SELF
          SET TEMP19 TO PROP-CONTROLS OF TEMP18
          INVOKE TEMP19 "Add" USING BY VALUE TEMP20
          SET TEMP23 TO PROP-COMBOBOX2 OF SELF
          SET TEMP21 TO PROP-PANEL1 OF SELF
          SET TEMP22 TO PROP-CONTROLS OF TEMP21
          INVOKE TEMP22 "Add" USING BY VALUE TEMP23
          SET TEMP26 TO PROP-LABEL2 OF SELF
          SET TEMP24 TO PROP-PANEL1 OF SELF
          SET TEMP25 TO PROP-CONTROLS OF TEMP24
          INVOKE TEMP25 "Add" USING BY VALUE TEMP26
          SET TEMP29 TO PROP-COMBOBOX1 OF SELF
          SET TEMP27 TO PROP-PANEL1 OF SELF
          SET TEMP28 TO PROP-CONTROLS OF TEMP27
          INVOKE TEMP28 "Add" USING BY VALUE TEMP29
          SET TEMP32 TO PROP-LABEL1 OF SELF
          SET TEMP30 TO PROP-PANEL1 OF SELF
          SET TEMP31 TO PROP-CONTROLS OF TEMP30
          INVOKE TEMP31 "Add" USING BY VALUE TEMP32
          SET TEMP33 TO PROP-PANEL1 OF SELF
          SET PROP-DOCK OF TEMP33 TO PROP-TOP OF ENUM-DOCKSTYLE
          MOVE 0 TO TEMP34
          MOVE 0 TO TEMP35
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP34 BY VALUE TEMP35 RETURNING TEMP36
          SET TEMP37 TO PROP-PANEL1 OF SELF
          SET PROP-LOCATION OF TEMP37 TO TEMP36
          SET TEMP38 TO N"panel1"
          SET TEMP39 TO PROP-PANEL1 OF SELF
          SET PROP-NAME OF TEMP39 TO TEMP38
          MOVE 292 TO TEMP40
          MOVE 95 TO TEMP41
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP40 BY VALUE TEMP41 RETURNING TEMP42
          SET TEMP43 TO PROP-PANEL1 OF SELF
          SET PROP-SIZE OF TEMP43 TO TEMP42
          MOVE 0 TO TEMP44
          SET TEMP45 TO PROP-PANEL1 OF SELF
          MOVE TEMP44 TO PROP-TABINDEX OF TEMP45
```

```
      *
      *button2
      *
          MOVE 205 TO TEMP46
          MOVE 64 TO TEMP47
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP46 BY VALUE TEMP47 RETURNING TEMP48
          SET TEMP49 TO PROP-BUTTON2 OF SELF
          SET PROP-LOCATION OF TEMP49 TO TEMP48
          SET TEMP50 TO N"button2"
          SET TEMP51 TO PROP-BUTTON2 OF SELF
          SET PROP-NAME OF TEMP51 TO TEMP50
          MOVE 75 TO TEMP52
          MOVE 23 TO TEMP53
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP52 BY VALUE TEMP53 RETURNING TEMP54
          SET TEMP55 TO PROP-BUTTON2 OF SELF
          SET PROP-SIZE OF TEMP55 TO TEMP54
          MOVE 5 TO TEMP56
          SET TEMP57 TO PROP-BUTTON2 OF SELF
          MOVE TEMP56 TO PROP-TABINDEX OF TEMP57
          SET TEMP58 TO N"クリア(&L)"
          SET TEMP59 TO PROP-BUTTON2 OF SELF
          SET PROP-TEXT OF TEMP59 TO TEMP58
          SET TEMP60 TO PROP-BUTTON2 OF SELF
          SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP60 TO B"1"
          SET TEMP61 TO PROP-BUTTON2 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button2_Click"
                 RETURNING TEMP62
          INVOKE TEMP61 "add_Click" USING BY VALUE TEMP62
      *
      *button1
      *
          MOVE 124 TO TEMP63
          MOVE 64 TO TEMP64
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP63 BY VALUE TEMP64 RETURNING TEMP65
          SET TEMP66 TO PROP-BUTTON1 OF SELF
          SET PROP-LOCATION OF TEMP66 TO TEMP65
          SET TEMP67 TO N"button1"
          SET TEMP68 TO PROP-BUTTON1 OF SELF
          SET PROP-NAME OF TEMP68 TO TEMP67
          MOVE 75 TO TEMP69
          MOVE 23 TO TEMP70
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP69 BY VALUE TEMP70 RETURNING TEMP71
          SET TEMP72 TO PROP-BUTTON1 OF SELF
          SET PROP-SIZE OF TEMP72 TO TEMP71
          MOVE 4 TO TEMP73
          SET TEMP74 TO PROP-BUTTON1 OF SELF
          MOVE TEMP73 TO PROP-TABINDEX OF TEMP74
          SET TEMP75 TO N"検索(&S)"
          SET TEMP76 TO PROP-BUTTON1 OF SELF
          SET PROP-TEXT OF TEMP76 TO TEMP75
          SET TEMP77 TO PROP-BUTTON1 OF SELF
          SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP77 TO B"1"
          SET TEMP78 TO PROP-BUTTON1 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click"
                 RETURNING TEMP79
          INVOKE TEMP78 "add_Click" USING BY VALUE TEMP79
      *
      *comboBox2
      *
          SET TEMP80 TO PROP-COMBOBOX2 OF SELF
          SET PROP-DROPDOWNSTYLE OF TEMP80 TO PROP-DROPDOWNLIST OF ENUM-COMBOBOXSTYLE
          SET TEMP81 TO PROP-COMBOBOX2 OF SELF
          SET PROP-FORMATTINGENABLED OF TEMP81 TO B"1"
          MOVE 134 TO TEMP82
          MOVE 38 TO TEMP83
          INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP82 BY VALUE TEMP83 RETURNING TEMP84
          SET TEMP85 TO PROP-COMBOBOX2 OF SELF
          SET PROP-LOCATION OF TEMP85 TO TEMP84
          SET TEMP86 TO N"comboBox2"
          SET TEMP87 TO PROP-COMBOBOX2 OF SELF
          SET PROP-NAME OF TEMP87 TO TEMP86
          MOVE 145 TO TEMP88
          MOVE 20 TO TEMP89
          INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP88 BY VALUE TEMP89 RETURNING TEMP90
          SET TEMP91 TO PROP-COMBOBOX2 OF SELF
          SET PROP-SIZE OF TEMP91 TO TEMP90
          MOVE 3 TO TEMP92
          SET TEMP93 TO PROP-COMBOBOX2 OF SELF
          MOVE TEMP92 TO PROP-TABINDEX OF TEMP93
          SET TEMP94 TO PROP-COMBOBOX2 OF SELF
          INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF
```

```
                BY VALUE N"comboBox2_SelectedIndexChanged" RETURNING TEMP95
            INVOKE TEMP94 "add_SelectedIndexChanged" USING BY VALUE TEMP95
        *
        *label2
        *
            SET TEMP96 TO PROP-LABEL2 OF SELF
            SET PROP-AUTOSIZE OF TEMP96 TO B"1"
            MOVE 12 TO TEMP97
            MOVE 41 TO TEMP98
            INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP97 BY VALUE TEMP98 RETURNING TEMP99
            SET TEMP100 TO PROP-LABEL2 OF SELF
            SET PROP-LOCATION OF TEMP100 TO TEMP99
            SET TEMP101 TO N"label2"
            SET TEMP102 TO PROP-LABEL2 OF SELF
            SET PROP-NAME OF TEMP102 TO TEMP101
            MOVE 117 TO TEMP103
            MOVE 12 TO TEMP104
            INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP103 BY VALUE TEMP104 RETURNING TEMP105
            SET TEMP106 TO PROP-LABEL2 OF SELF
            SET PROP-SIZE OF TEMP106 TO TEMP105
            MOVE 2 TO TEMP107
            SET TEMP108 TO PROP-LABEL2 OF SELF
            MOVE TEMP107 TO PROP-TABINDEX OF TEMP108
            SET TEMP109 TO N"市区町村名の選択(&C):"
            SET TEMP110 TO PROP-LABEL2 OF SELF
            SET PROP-TEXT OF TEMP110 TO TEMP109
        *
        *comboBox1
        *
            SET TEMP111 TO PROP-COMBOBOX1 OF SELF
            SET PROP-DROPDOWNSTYLE OF TEMP111 TO PROP-DROPDOWNLIST OF ENUM-COMBOBOXSTYLE
            SET TEMP112 TO PROP-COMBOBOX1 OF SELF
            SET PROP-FORMATTINGENABLED OF TEMP112 TO B"1"
            MOVE 134 TO TEMP113
            MOVE 8 TO TEMP114
            INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP113 BY VALUE TEMP114 RETURNING TEMP115
            SET TEMP116 TO PROP-COMBOBOX1 OF SELF
            SET PROP-LOCATION OF TEMP116 TO TEMP115
            SET TEMP117 TO N"comboBox1"
            SET TEMP118 TO PROP-COMBOBOX1 OF SELF
            SET PROP-NAME OF TEMP118 TO TEMP117
            MOVE 145 TO TEMP119
            MOVE 20 TO TEMP120
            INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP119 BY VALUE TEMP120 RETURNING TEMP121
            SET TEMP122 TO PROP-COMBOBOX1 OF SELF
            SET PROP-SIZE OF TEMP122 TO TEMP121
            MOVE 1 TO TEMP123
            SET TEMP124 TO PROP-COMBOBOX1 OF SELF
            MOVE TEMP123 TO PROP-TABINDEX OF TEMP124
            SET TEMP125 TO PROP-COMBOBOX1 OF SELF
            INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE
N"comboBox1_SelectedIndexChanged" RETURNING TEMP126
            INVOKE TEMP125 "add_SelectedIndexChanged" USING BY VALUE TEMP126
        *
        *label1
        *
            SET TEMP127 TO PROP-LABEL1 OF SELF
            SET PROP-AUTOSIZE OF TEMP127 TO B"1"
            MOVE 12 TO TEMP128
            MOVE 11 TO TEMP129
            INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP128 BY VALUE TEMP129 RETURNING TEMP130
            SET TEMP131 TO PROP-LABEL1 OF SELF
            SET PROP-LOCATION OF TEMP131 TO TEMP130
            SET TEMP132 TO N"label1"
            SET TEMP133 TO PROP-LABEL1 OF SELF
            SET PROP-NAME OF TEMP133 TO TEMP132
            MOVE 116 TO TEMP134
            MOVE 12 TO TEMP135
            INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP134 BY VALUE TEMP135 RETURNING TEMP136
            SET TEMP137 TO PROP-LABEL1 OF SELF
            SET PROP-SIZE OF TEMP137 TO TEMP136
            MOVE 0 TO TEMP138
            SET TEMP139 TO PROP-LABEL1 OF SELF
            MOVE TEMP138 TO PROP-TABINDEX OF TEMP139
            SET TEMP140 TO N"都道府県名の選択(&P):"
            SET TEMP141 TO PROP-LABEL1 OF SELF
            SET PROP-TEXT OF TEMP141 TO TEMP140
        *
        *statusStrip1
        *
            MOVE 1 TO TEMP145
```

```
        INVOKE ARRAY-TOOLSTRIPITEM "NEW" USING BY VALUE TEMP145 RETURNING TEMP146
        SET TEMP144 TO PROP-TOOLSTRIPSTATUSLABEL1 OF SELF
        INVOKE TEMP146 "Set" USING BY VALUE 0 BY VALUE TEMP144
        SET TEMP142 TO PROP-STATUSSTRIP1 OF SELF
        SET TEMP143 TO PROP-ITEMS OF TEMP142
        INVOKE TEMP143 "AddRange" USING BY VALUE TEMP146
        MOVE 0 TO TEMP147
        MOVE 244 TO TEMP148
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP147 BY VALUE TEMP148 RETURNING TEMP149
        SET TEMP150 TO PROP-STATUSSTRIP1 OF SELF
        SET PROP-LOCATION OF TEMP150 TO TEMP149
        SET TEMP151 TO N"statusStrip1"
        SET TEMP152 TO PROP-STATUSSTRIP1 OF SELF
        SET PROP-NAME OF TEMP152 TO TEMP151
        MOVE 292 TO TEMP153
        MOVE 22 TO TEMP154
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP153 BY VALUE TEMP154 RETURNING TEMP155
        SET TEMP156 TO PROP-STATUSSTRIP1 OF SELF
        SET PROP-SIZE OF TEMP156 TO TEMP155
        MOVE 1 TO TEMP157
        SET TEMP158 TO PROP-STATUSSTRIP1 OF SELF
        MOVE TEMP157 TO PROP-TABINDEX OF TEMP158
        SET TEMP159 TO N"statusStrip1"
        SET TEMP160 TO PROP-STATUSSTRIP1 OF SELF
        SET PROP-TEXT OF TEMP160 TO TEMP159
    *
    *toolStripStatusLabel1
    *
        SET TEMP161 TO N"toolStripStatusLabel1"
        SET TEMP162 TO PROP-TOOLSTRIPSTATUSLABEL1 OF SELF
        SET PROP-NAME OF TEMP162 TO TEMP161
        MOVE 0 TO TEMP163
        MOVE 17 TO TEMP164
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP163 BY VALUE TEMP164 RETURNING TEMP165
        SET TEMP166 TO PROP-TOOLSTRIPSTATUSLABEL1 OF SELF
        SET PROP-SIZE OF TEMP166 TO TEMP165
    *
    *listView1
    *
        MOVE 2 TO TEMP171
        INVOKE ARRAY-COLUMNHEADER "NEW" USING BY VALUE TEMP171 RETURNING TEMP172
        SET TEMP169 TO PROP-COLUMNHEADER1 OF SELF
        INVOKE TEMP172 "Set" USING BY VALUE 0 BY VALUE TEMP169
        SET TEMP170 TO PROP-COLUMNHEADER2 OF SELF
        INVOKE TEMP172 "Set" USING BY VALUE 1 BY VALUE TEMP170
        SET TEMP167 TO PROP-LISTVIEW1 OF SELF
        SET TEMP168 TO PROP-COLUMNS OF TEMP167
        INVOKE TEMP168 "AddRange" USING BY VALUE TEMP172
        SET TEMP173 TO PROP-LISTVIEW1 OF SELF
        SET PROP-DOCK OF TEMP173 TO PROP-FILL OF ENUM-DOCKSTYLE
        SET TEMP174 TO PROP-LISTVIEW1 OF SELF
        SET PROP-FULLROWSELECT OF TEMP174 TO B"1"
        SET TEMP175 TO PROP-LISTVIEW1 OF SELF
        SET PROP-GRIDLINES OF TEMP175 TO B"1"
        MOVE 0 TO TEMP176
        MOVE 95 TO TEMP177
        INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP176 BY VALUE TEMP177 RETURNING TEMP178
        SET TEMP179 TO PROP-LISTVIEW1 OF SELF
        SET PROP-LOCATION OF TEMP179 TO TEMP178
        SET TEMP180 TO N"listView1"
        SET TEMP181 TO PROP-LISTVIEW1 OF SELF
        SET PROP-NAME OF TEMP181 TO TEMP180
        MOVE 292 TO TEMP182
        MOVE 149 TO TEMP183
        INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP182 BY VALUE TEMP183 RETURNING TEMP184
        SET TEMP185 TO PROP-LISTVIEW1 OF SELF
        SET PROP-SIZE OF TEMP185 TO TEMP184
        MOVE 2 TO TEMP186
        SET TEMP187 TO PROP-LISTVIEW1 OF SELF
        MOVE TEMP186 TO PROP-TABINDEX OF TEMP187
        SET TEMP188 TO PROP-LISTVIEW1 OF SELF
        SET PROP-USECOMPATIBLESTATEIMAGEBE OF TEMP188 TO B"0"
        SET TEMP189 TO PROP-LISTVIEW1 OF SELF
        SET PROP-VIEW OF TEMP189 TO PROP-DETAILS OF ENUM-VIEW
    *
    *columnHeader1
    *
        SET TEMP190 TO N"町域"
        SET TEMP191 TO PROP-COLUMNHEADER1 OF SELF
        SET PROP-TEXT OF TEMP191 TO TEMP190
        MOVE 132 TO TEMP192
```

```
           SET TEMP193 TO PROP-COLUMNHEADER1 OF SELF
           MOVE TEMP192 TO PROP-WIDTH OF TEMP193
      *
      *columnHeader2
      *
           SET TEMP194 TO N"郵便番号"
           SET TEMP195 TO PROP-COLUMNHEADER2 OF SELF
           SET PROP-TEXT OF TEMP195 TO TEMP194
           MOVE 139 TO TEMP196
           SET TEMP197 TO PROP-COLUMNHEADER2 OF SELF
           MOVE TEMP196 TO PROP-WIDTH OF TEMP197
      *
      *Form1
      *
           MOVE 6.00000000000000E+00 TO TEMP198
           MOVE 1.20000000000000E+01 TO TEMP199
           INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP198 BY VALUE TEMP199 RETURNING TEMP200
           SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP200
           SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
           MOVE 292 TO TEMP201
           MOVE 266 TO TEMP202
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP201 BY VALUE TEMP202 RETURNING TEMP203
           SET PROP-CLIENTSIZE OF SELF TO TEMP203
           SET TEMP205 TO PROP-LISTVIEW1 OF SELF
           SET TEMP204 TO PROP-CONTROLS OF SELF
           INVOKE TEMP204 "Add" USING BY VALUE TEMP205
           SET TEMP207 TO PROP-STATUSSTRIP1 OF SELF
           SET TEMP206 TO PROP-CONTROLS OF SELF
           INVOKE TEMP206 "Add" USING BY VALUE TEMP207
           SET TEMP209 TO PROP-PANEL1 OF SELF
           SET TEMP208 TO PROP-CONTROLS OF SELF
           INVOKE TEMP208 "Add" USING BY VALUE TEMP209
           SET TEMP210 TO N"Form1"
           SET PROP-NAME OF SELF TO TEMP210
           SET TEMP211 TO N"郵便番号の検索"
           SET PROP-TEXT OF SELF TO TEMP211
           INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"Form1_Load"
                  RETURNING TEMP212
           INVOKE SELF "add_Load" USING BY VALUE TEMP212
           SET TEMP213 TO PROP-PANEL1 OF SELF
           INVOKE TEMP213 "ResumeLayout" USING BY VALUE B"0"
           SET TEMP214 TO PROP-PANEL1 OF SELF
           INVOKE TEMP214 "PerformLayout"
           SET TEMP215 TO PROP-STATUSSTRIP1 OF SELF
           INVOKE TEMP215 "ResumeLayout" USING BY VALUE B"0"
           SET TEMP216 TO PROP-STATUSSTRIP1 OF SELF
           INVOKE TEMP216 "PerformLayout"
           INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
           INVOKE SELF "PerformLayout"
       END METHOD INITIALIZECOMPONENT.

       METHOD-ID. NEW.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
           INVOKE SELF "InitializeComponent".
       END METHOD NEW.

       METHOD-ID. Form1_Load PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 都道府県一覧.
          02 FILLER PIC N(5) VALUE N"(選択)".
          02 FILLER PIC N(5) VALUE N"北海道".
          02 FILLER PIC N(5) VALUE N"青森県".
          02 FILLER PIC N(5) VALUE N"岩手県".
          02 FILLER PIC N(5) VALUE N"宮城県".
          02 FILLER PIC N(5) VALUE N"秋田県".
          02 FILLER PIC N(5) VALUE N"山形県".
          02 FILLER PIC N(5) VALUE N"福島県".
          02 FILLER PIC N(5) VALUE N"茨城県".
          02 FILLER PIC N(5) VALUE N"栃木県".
          02 FILLER PIC N(5) VALUE N"群馬県".
          02 FILLER PIC N(5) VALUE N"埼玉県".
          02 FILLER PIC N(5) VALUE N"千葉県".
          02 FILLER PIC N(5) VALUE N"東京都".
          02 FILLER PIC N(5) VALUE N"神奈川県".
          02 FILLER PIC N(5) VALUE N"新潟県".
```

```
          02 FILLER PIC N(5) VALUE N"富山県".
          02 FILLER PIC N(5) VALUE N"石川県".
          02 FILLER PIC N(5) VALUE N"福井県".
          02 FILLER PIC N(5) VALUE N"山梨県".
          02 FILLER PIC N(5) VALUE N"長野県".
          02 FILLER PIC N(5) VALUE N"岐阜県".
          02 FILLER PIC N(5) VALUE N"静岡県".
          02 FILLER PIC N(5) VALUE N"愛知県".
          02 FILLER PIC N(5) VALUE N"三重県".
          02 FILLER PIC N(5) VALUE N"滋賀県".
          02 FILLER PIC N(5) VALUE N"京都府".
          02 FILLER PIC N(5) VALUE N"大阪府".
          02 FILLER PIC N(5) VALUE N"兵庫県".
          02 FILLER PIC N(5) VALUE N"奈良県".
          02 FILLER PIC N(5) VALUE N"和歌山県".
          02 FILLER PIC N(5) VALUE N"鳥取県".
          02 FILLER PIC N(5) VALUE N"島根県".
          02 FILLER PIC N(5) VALUE N"岡山県".
          02 FILLER PIC N(5) VALUE N"広島県".
          02 FILLER PIC N(5) VALUE N"山口県".
          02 FILLER PIC N(5) VALUE N"徳島県".
          02 FILLER PIC N(5) VALUE N"香川県".
          02 FILLER PIC N(5) VALUE N"愛媛県".
          02 FILLER PIC N(5) VALUE N"高知県".
          02 FILLER PIC N(5) VALUE N"福岡県".
          02 FILLER PIC N(5) VALUE N"佐賀県".
          02 FILLER PIC N(5) VALUE N"長崎県".
          02 FILLER PIC N(5) VALUE N"熊本県".
          02 FILLER PIC N(5) VALUE N"大分県".
          02 FILLER PIC N(5) VALUE N"宮崎県".
          02 FILLER PIC N(5) VALUE N"鹿児島県".
          02 FILLER PIC N(5) VALUE N"沖縄県".
       01 都道府県表 REDEFINES 都道府県一覧.
          02 都道府県名 PIC N(5) OCCURS 48 TIMES INDEXED BY IDX.
       01 WK-STRING OBJECT REFERENCE CLASS-STRING.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           *>  都道府県名のリストを作成します。
           PERFORM VARYING IDX FROM 1 BY 1 UNTIL IDX > 48
              SET WK-STRING TO 都道府県名(IDX)
              INVOKE PROP-ITEMS OF comboBox1 "Add" USING WK-STRING
           END-PERFORM.
           SET PROP-SELECTEDINDEX OF comboBox1 TO 0.
       END METHOD Form1_Load.

       METHOD-ID. comboBox1_SelectedIndexChanged PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 CHANNEL-FACTORY OBJECT REFERENCE CLASS-FACTORY.
       01 ZIPCODE-SERVICE OBJECT REFERENCE INTERFACE-IZIPCODESERVICE.
       01 CHANNEL OBJECT REFERENCE INTERFACE-ICHANNEL.
       01 EXCEP   OBJECT REFERENCE CLASS-EXCEPTION.
       01 IDX BINARY-LONG.
       01 都道府県名 PIC N(5).
       01 市区町村一覧.
         02 市区町村数 PIC S9(9) COMP-5.
         02 市区町村名 PIC N(20) OCCURS 200.
       01 WK-SELECTED OBJECT REFERENCE CLASS-OBJECT.
       01 WK-STRING  OBJECT REFERENCE CLASS-STRING.
       01 WK-FAULTED PIC 1.
         88 IS-FAULTED VALUE B"1".
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           *>  選択されている都道府県名を取り出します。
           SET WK-SELECTED TO PROP-SELECTEDITEM OF comboBox1
           SET 都道府県名 TO WK-SELECTED AS CLASS-STRING.
           *>  市区町村リストをクリアします。
           INVOKE PROP-ITEMS OF comboBox2 "Clear"
           SET PROP-TEXT OF toolStripStatusLabel1 TO N" "

           IF 都道府県名 NOT = N"(選択)" THEN
```

```
            TRY
               *> チャネルファクトリを作成します。
               SET CHANNEL-FACTORY TO CLASS-FACTORY::"NEW"("ZipCodeClientConfig")

               TRY
                  *> チャネル(サービスコントラクト)を取得します。
                  SET ZIPCODE-SERVICE TO CHANNEL-FACTORY::"CreateChannel"

                  *> サービスから"GetCties"メソッドを呼び出します。
                  MOVE 200 TO 市区町村数
                  PERFORM TEST BEFORE UNTIL 市区町村数 < 200
                     INVOKE ZIPCODE-SERVICE "GetCities" USING 都道府県名 RETURNING 市区町村一覧
                     PERFORM VARYING IDX FROM 1 BY 1 UNTIL IDX > 市区町村数
                        *> 市区町村名リストを作成します。
                        SET WK-STRING TO 市区町村名(IDX)
                        INVOKE PROP-ITEMS OF comboBox2 "Add" USING WK-STRING
                     END-PERFORM
                  END-PERFORM

                  *> チャネルを閉じます。
                  SET CHANNEL TO ZIPCODE-SERVICE AS INTERFACE-ICHANNEL
                  INVOKE CHANNEL "Close"
               CATCH EXCEP
                  SET PROP-TEXT OF toolStripStatusLabel1 TO N"検索処理で例外が発生しました。"
                  SET IS-FAULTED TO TRUE
               END-TRY
               *> リストビューをクリアします。
               INVOKE PROP-ITEMS OF listView1 "Clear"
            FINALLY
               *> チャネルファクトリを閉じます。
               IF NOT IS-FAULTED
                  INVOKE CHANNEL-FACTORY "Close"
               END-IF
            END-TRY

         END-IF.

      END METHOD comboBox1_SelectedIndexChanged.

      METHOD-ID. comboBox2_SelectedIndexChanged PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.
         *> リストビューをクリアします。
         INVOKE PROP-ITEMS OF listView1 "Clear".
      END METHOD comboBox2_SelectedIndexChanged.

      METHOD-ID. button1_Click PRIVATE.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 CHANNEL-FACTORY OBJECT REFERENCE CLASS-FACTORY.
      01 ZIPCODE-SERVICE OBJECT REFERENCE INTERFACE-IZIPCODESERVICE.
      01 CHANNEL OBJECT REFERENCE INTERFACE-ICHANNEL.
      01 EXCEP   OBJECT REFERENCE CLASS-EXCEPTION.
      01 IDX BINARY-LONG.
      01 都道府県名 PIC N(5).
      01 市区町村名 PIC N(20).
      01 郵便番号一覧.
        02 町域数   PIC S9(9) COMP-5.
        02 町域名   PIC N(40) OCCURS 100.
        02 郵便番号 PIC X(7)  OCCURS 100.
      01 WK-SELECTED OBJECT REFERENCE CLASS-OBJECT.
      01 WK-ITEM   OBJECT REFERENCE CLASS-LISTVIEWITEM.
      01 WK-SUBITEM OBJECT REFERENCE CLASS-LISTVIEWSUBITEM.
      01 WK-STRING  OBJECT REFERENCE CLASS-STRING.
      01 WK-BUILDER OBJECT REFERENCE CLASS-STRINGBUILDER.
      01 WK-FAULTED PIC 1.
        88 IS-FAULTED VALUE B"1".
      LINKAGE SECTION.
      01 sender OBJECT REFERENCE CLASS-OBJECT.
      01 e OBJECT REFERENCE CLASS-EVENTARGS.
      PROCEDURE DIVISION USING BY VALUE sender e.
         *> 選択されている都道府県名および市区町村名を取り出します。
         SET WK-SELECTED TO PROP-SELECTEDITEM OF comboBox1
         SET 都道府県名 TO WK-SELECTED AS CLASS-STRING.
```

```
        SET WK-SELECTED TO PROP-SELECTEDITEM OF comboBox2
        SET 市区町村名 TO WK-SELECTED AS CLASS-STRING.
        *> リストビューをクリアします。
        INVOKE PROP-ITEMS OF listView1 "Clear".
        SET PROP-TEXT OF toolStripStatusLabel1 TO N" "

        TRY
            *> チャネルファクトリを作成します。
            SET CHANNEL-FACTORY TO CLASS-FACTORY::"NEW"("ZipCodeClientConfig")

            TRY
                *> チャネル(サービスコントラクト)を取得します。
                SET ZIPCODE-SERVICE TO CHANNEL-FACTORY::"CreateChannel"

                *> サービスから"GetZipCode"メソッドを呼び出します。
                MOVE 100 TO 町域数
                PERFORM TEST BEFORE UNTIL 町域数 < 100
                    INVOKE ZIPCODE-SERVICE "GetZipCode" USING 都道府県名 市区町村名
                            RETURNING 郵便番号一覧
                    PERFORM VARYING IDX FROM 1 BY 1 UNTIL IDX > 町域数
                        SET WK-ITEM TO CLASS-LISTVIEWITEM::"NEW" ( 町域名(IDX))
                        INVOKE PROP-SUBITEMS OF WK-ITEM "Add" USING 郵便番号(IDX)
                        INVOKE PROP-ITEMS OF listView1 "Add" USING WK-ITEM
                    END-PERFORM
                END-PERFORM

                *> チャネルを閉じます。
                SET CHANNEL TO ZIPCODE-SERVICE AS INTERFACE-ICHANNEL
                INVOKE CHANNEL "Close"
            CATCH EXCEP
                SET PROP-TEXT OF toolStripStatusLabel1 TO N"検索処理で例外が発生しました。"
                SET IS-FAULTED TO TRUE
            END-TRY
        FINALLY
            *> チャネルファクトリを閉じます。
            IF NOT IS-FAULTED
                INVOKE CHANNEL-FACTORY "Close"
            END-IF
        END-TRY.

    END METHOD button1_Click.

    METHOD-ID. button2_Click PRIVATE.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    LINKAGE SECTION.
    01 sender OBJECT REFERENCE CLASS-OBJECT.
    01 e OBJECT REFERENCE CLASS-EVENTARGS.
    PROCEDURE DIVISION USING BY VALUE sender e.
        *> 各フィールドをクリアします。
        SET PROP-SELECTEDITEM OF comboBox1 TO N"(選択)".
        INVOKE PROP-ITEMS OF listView1 "Clear".
        SET PROP-TEXT OF toolStripStatusLabel1 TO N" ".
    END METHOD button2_Click.

    END OBJECT.
    END CLASS CLASS-THIS.
```

ソースコード：ZipCodeClient¥Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
   .
REPOSITORY.
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-MAINFORM AS "ZipCodeClient.Form1"
   CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
   .

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-MAINFORM OBJECT REFERENCE CLASS-MAINFORM.
PROCEDURE DIVISION.
   INVOKE CLASS-APPLICATION "EnableVisualStyles".
   INVOKE CLASS-APPLICATION "SetCompatibleTextRenderingDefault" USING BY VALUE B"0".

   INVOKE CLASS-MAINFORM "NEW" RETURNING WK-MAINFORM.
   INVOKE CLASS-APPLICATION "Run" USING BY VALUE WK-MAINFORM.
END PROGRAM MAIN.
```

NCOBU01039

Manage Stock Web Service サンプルアプリケーション

Web サイトの説明は互換のために残されています。 Visual Studio では、Web サイトの作成を推奨していません。

分散トランザクションを利用した WCF による XML Web サービスアプリケーションの例について説明します。

データベースを準備する

Manage Stock Web Service では SQL Server 上のサンプルデータベース(COBOLSample)の STOCK 表と、ManageStock ストアドプロシージャを使用します。このため、以下の準備が必要です。

- サンプルデータベースの作成

IIS Express を使用する

本のサンプルアプリケーションでは、IIS Express を使用します。

本サンプルでは、Manage Stock Web Service は以下の URL で公開されます。 クライアントアプリケーションの構成ファイルのエンドポイントの設定を変更してください。

```
http://localhost:2317/Service.svc
```

ポート番号の「2317」は環境によって異なる場合があります。 ポート番号は以下の手順で確認することができます。

1. ソリューションエクスプローラーで、Manage Stock Web Service の Web サイト (WCFManageStockService)を選択します。
2. コンテキストメニューから[プロパティ]を選択し、プロパティウィンドウを表示します。
3. [URL]項目にサイトの URL アドレスが表示されます。"http://localhost:"の後ろの数字がポート番号です。

公開されるサービス

サンプルの Manage Stock Web Service は XML Web サービスとして実行され、以下のエンドポイントの設定でサービスが公開されることを想定しています。

Address	http://localhost:2317/Service.svc
Binding	WsHttpBinding
Contract	WCFManageStockService.IManageStockService

WCFManageStockService.IManageStockService サービスコントラクトは、以下のオペレーションを公開します。

オペレーション	説明
Stock	製品番号と入庫量を指定して入庫処理を行います。 正常に実行されると現在の在庫量が返却されます。失敗した場合は-1 が返却されます。
Delivery	製品番号と出庫量を指定して出庫処理を行います。 正常に実行されると現在の在庫量が返却されます。失敗した場合は-1 が返却されます。

クライアントアプリケーションの解説

クライアントアプリケーション(WCFManageStockClient)は Manage Stock Web Service が起動している場合に正しく実行することができます。

Manage Stock Web Service のポート番号(または URL アドレス)が変わっている場合は、以下の手順で更新します。

1. ソリューションエクスプローラで、Manage Stock Web Service のクライアントアプリケーション(WCFManageStockClient)配下の、[Connected Services]-[WCFManageStock]ノードを選択します。
2. コンテキストメニューから[サービス参照の構成]を選択し、[サービス参照設定ダイアログ]を開きます。
3. [アドレス]のフィールドに Manage Stock Web Service の URL を指定し、[OK]ボタンをクリックします。

WCFManageStockClient では、[入庫]または[出庫]の何れかのオプションボタンを選択し、製品番号と数量を入力して[更新]ボタンをクリックするとサービスのオペレーションが呼び出されます。

Manage Stock Web Service はトランザクションフローへの参加を義務づけているため、クライアントアプリケーションでは System.Transactions.TransactionScope クラスを使用してトランザクションを開始し、トランザクションスコープ内からオペレーションを呼び出しています。オペレーションが正常に実行された場合は、TransactionScope.Complete メソッドを呼び出してコミットを完了しています。

NCOBU01040

ソースコード：WCFManageStockService¥IManageStockService.cob

```
IDENTIFICATION DIVISION.
INTERFACE-ID. INTERFACE-SERVICE AS "WCFManageStock.IManageStockService"
   CUSTOM-ATTRIBUTE CA-SERVICECONTRACT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-SERVICECONTRACT CLASS CLASS-SERVICECONTRACT
   CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT CLASS CLASS-OPERATIONCONTRACT
   CUSTOM-ATTRIBUTE CA-TRANSACTIONFLOW CLASS CLASS-TRANSACTIONFLOW
                                USING PROP-MANDATORY OF ENUM-TRANSACTIONFLOWOPTION
   .
REPOSITORY.
   CLASS CLASS-SERVICECONTRACT AS "System.ServiceModel.ServiceContractAttribute"
   CLASS CLASS-OPERATIONCONTRACT AS "System.ServiceModel.OperationContractAttribute"
   CLASS CLASS-TRANSACTIONFLOW AS "System.ServiceModel.TransactionFlowAttribute"
   ENUM  ENUM-TRANSACTIONFLOWOPTION AS "System.ServiceModel.TransactionFlowOption"
   PROPERTY PROP-MANDATORY AS "Mandatory".

PROCEDURE DIVISION.

* 入庫処理オペレーション
METHOD-ID. OPERATION-1 AS "Stock" CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT
                                          CA-TRANSACTIONFLOW.
DATA DIVISION.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
END METHOD OPERATION-1.

* 出庫処理オペレーション
METHOD-ID. OPERATION-2 AS "Delivery" CUSTOM-ATTRIBUTE CA-OPERATIONCONTRACT
                                            CA-TRANSACTIONFLOW.
DATA DIVISION.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
END METHOD OPERATION-2.

END INTERFACE INTERFACE-SERVICE.
```

ソースコード：WCFManageStockService¥ManageStockService.cob

```
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "WCFManageStock.ManageStockService".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-OPERATIONBEHAVIOR CLASS CLASS-OPERATIONBEHAVIOR
                       PROPERTY PROP-TRANSACTIONSCOPEREQUIRED IS B"1"
                       PROPERTY PROP-TRANSACTIONAUTOCOMPLETE IS B"1"
   .

REPOSITORY.
   CLASS CLASS-STRING AS "System.String"
   INTERFACE INTERFACE-SERVICECONTRACT AS "WCFManageStock.IManageStockService"
   CLASS CLASS-OPERATIONBEHAVIOR AS "System.ServiceModel.OperationBehaviorAttribute"
   PROPERTY PROP-TRANSACTIONSCOPEREQUIRED AS "TransactionScopeRequired"
   PROPERTY PROP-TRANSACTIONAUTOCOMPLETE AS "TransactionAutoComplete"
   .
OBJECT. IMPLEMENTS INTERFACE-SERVICECONTRACT.
DATA DIVISION.
PROCEDURE DIVISION.

METHOD-ID. OPERATION-1 AS "Stock" CUSTOM-ATTRIBUTE CA-OPERATIONBEHAVIOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 KIND    PIC S9(4) COMP-5.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
   *> 入庫処理
   MOVE 1 TO KIND.
   CALL "WCFManageStock.ManageStock" USING KIND GOODSNO QUANTITY UPDATED.
END METHOD OPERATION-1.

METHOD-ID. OPERATION-2 AS "Delivery" CUSTOM-ATTRIBUTE CA-OPERATIONBEHAVIOR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 KIND    PIC S9(4) COMP-5.
LINKAGE SECTION.
01 GOODSNO  PIC S9(4) COMP-5.
01 QUANTITY PIC S9(9) COMP-5.
01 UPDATED  PIC S9(9) COMP-5.
PROCEDURE DIVISION USING BY VALUE GOODSNO QUANTITY RETURNING UPDATED.
   *> 出庫処理
   MOVE 2 TO KIND.
   CALL "WCFManageStock.ManageStock" USING KIND GOODSNO QUANTITY UPDATED.
END METHOD OPERATION-2.

END OBJECT.
END CLASS CLASS-THIS.
```

NCOBU01042

ソースコード：WCFManageStockService¥ManageStock.cob

```
****
**** データベースアクセスルーチン
****
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PROGRAM-1 AS "WCFManageStock.ManageStock".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
    CLASS CLASS-TRANSACTION AS "System.Transactions.Transaction"
    PROPERTY PROP-CURRENT AS "Current"
    .
 DATA DIVISION.
 WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE        PIC X(5).
 01 SQLINFOA.
   02 SQLERRD       PIC S9(9) COMP-5 OCCURS 6 TIMES.
 01 PRODUCT_NUMBER   PIC S9(4) COMP-5.
 01 INOUT_KIND       PIC S9(4) COMP-5.
 01 INOUT_QUANTITY   PIC S9(9) COMP-5.
 01 UPDATED_QUANTITY PIC S9(9) COMP-5.
 01 ERROR_MESSAGE    PIC X(256).
    EXEC SQL END   DECLARE SECTION END-EXEC.
 01 CURRENT-TRANSACTION OBJECT REFERENCE CLASS-TRANSACTION.
 LINKAGE SECTION.
 01 KIND    PIC S9(4) COMP-5.
 01 GOODSNO  PIC S9(4) COMP-5.
 01 QUANTITY PIC S9(9) COMP-5.
 01 UPDATED  PIC S9(9) COMP-5.
 PROCEDURE DIVISION USING KIND GOODSNO QUANTITY UPDATED.

    EXEC SQL CONNECT TO DEFAULT END-EXEC.
    IF SQLSTATE NOT = "00000" AND SQLSTATE NOT = "01000"
       MOVE -1 TO UPDATED
       EXIT PROGRAM
    END-IF

    MOVE GOODSNO  TO PRODUCT_NUMBER.
    MOVE KIND    TO INOUT_KIND.
    MOVE QUANTITY TO INOUT_QUANTITY.

    *> ストアドプロシージャの呼び出し
    EXEC SQL
       CALL ManageStock (:PRODUCT_NUMBER, :INOUT_KIND, :INOUT_QUANTITY,
                         :UPDATED_QUANTITY, :ERROR_MESSAGE)
    END-EXEC.

    *> ストアドプロシージャでエラーが発生した場合や
    *> 在庫がマイナスになった場合は-1 を返す
    IF SQLSTATE NOT = "00000" OR SQLERRD(1) NOT = 0 OR UPDATED_QUANTITY < 0
       MOVE -1 TO UPDATED
    ELSE
       MOVE UPDATED_QUANTITY TO UPDATED
    END-IF.

    EXEC SQL DISCONNECT DEFAULT END-EXEC
 END PROGRAM PROGRAM-1.
```

ソースコード：WCFManageStockClient¥Form1.cob

```
@OPTIONS NOALPHAL
IDENTIFICATION DIVISION.
CLASS-ID. CLASS-THIS AS "WCFManageStockClient.Form1"
   INHERITS CLASS-FORM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
   CLASS CLASS-BOOLEAN AS "System.Boolean"
   CLASS CLASS-CONTAINER AS "System.ComponentModel.Container"
   INTERFACE INTERFACE-ICONTAINER AS "System.ComponentModel.IContainer"
   CLASS CLASS-POINT AS "System.Drawing.Point"
   CLASS CLASS-SIZE AS "System.Drawing.Size"
   CLASS CLASS-SIZEF AS "System.Drawing.SizeF"
   CLASS CLASS-EVENTARGS AS "System.EventArgs"
   DELEGATE DELEGATE-EVENTHANDLER AS "System.EventHandler"
   CLASS CLASS-EXCEPTION AS "System.Exception"
   CLASS CLASS-INT16 AS "System.Int16"
   CLASS CLASS-INT32 AS "System.Int32"
   CLASS CLASS-OBJECT AS "System.Object"
   CLASS CLASS-STRING AS "System.String"
   CLASS CLASS-TRANSACTIONSCOPE AS "System.Transactions.TransactionScope"
   CLASS CLASS-TYPE AS "System.Type"
   ENUM ENUM-AUTOSCALEMODE AS "System.Windows.Forms.AutoScaleMode"
   CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
   CLASS CLASS-CONTROLCOLLECTION AS "System.Windows.Forms.Control+ControlCollection"
   CLASS CLASS-FORM AS "System.Windows.Forms.Form"
   ENUM ENUM-FORMBORDERSTYLE AS "System.Windows.Forms.FormBorderStyle"
   CLASS CLASS-LABEL AS "System.Windows.Forms.Label"
   CLASS CLASS-MASKEDTEXTBOX AS "System.Windows.Forms.MaskedTextBox"
   CLASS CLASS-MESSAGEBOX AS "System.Windows.Forms.MessageBox"
   ENUM ENUM-MESSAGEBOXBUTTONS AS "System.Windows.Forms.MessageBoxButtons"
   ENUM ENUM-MESSAGEBOXICON AS "System.Windows.Forms.MessageBoxIcon"
   CLASS CLASS-RADIOBUTTON AS "System.Windows.Forms.RadioButton"
   CLASS CLASS-TEXTBOX AS "System.Windows.Forms.TextBox"
   CLASS CLASS-MANAGESTOCKPROXY AS
         "WCFManageStockClient.WCFManageStock.ManageStockServiceClient"
   PROPERTY PROP-AUTOSCALEDIMENSIONS AS "AutoScaleDimensions"
   PROPERTY PROP-AUTOSCALEMODE AS "AutoScaleMode"
   PROPERTY PROP-AUTOSIZE AS "AutoSize"
   PROPERTY PROP-BUTTON1 AS "button1"
   PROPERTY PROP-BUTTON2 AS "button2"
   PROPERTY PROP-CHECKED AS "Checked"
   PROPERTY PROP-CLIENTSIZE AS "ClientSize"
   PROPERTY PROP-CONTROLS AS "Controls"
   PROPERTY PROP-ERROR AS "Error"
   PROPERTY PROP-FIXEDDIALOG AS "FixedDialog"
   PROPERTY PROP-FONT AS "Font"
   PROPERTY PROP-FORMBORDERSTYLE AS "FormBorderStyle"
   PROPERTY PROP-INFORMATION AS "Information"
   PROPERTY PROP-LABEL1 AS "label1"
   PROPERTY PROP-LABEL2 AS "label2"
   PROPERTY PROP-LABEL3 AS "label3"
   PROPERTY PROP-LOCATION AS "Location"
   PROPERTY PROP-MASK AS "Mask"
   PROPERTY PROP-MASKEDTEXTBOX1 AS "maskedTextBox1"
   PROPERTY PROP-MASKEDTEXTBOX2 AS "maskedTextBox2"
   PROPERTY PROP-MAXIMIZEBOX AS "MaximizeBox"
   PROPERTY PROP-MESSAGE AS "Message"
   PROPERTY PROP-MINIMIZEBOX AS "MinimizeBox"
   PROPERTY PROP-NAME AS "Name"
   PROPERTY PROP-OK AS "OK"
   PROPERTY PROP-RADIOBUTTON1 AS "radioButton1"
   PROPERTY PROP-RADIOBUTTON2 AS "radioButton2"
   PROPERTY PROP-READONLY AS "ReadOnly"
   PROPERTY PROP-SHOWICON AS "ShowIcon"
   PROPERTY PROP-SIZE AS "Size"
   PROPERTY PROP-TABINDEX AS "TabIndex"
   PROPERTY PROP-TABSTOP AS "TabStop"
   PROPERTY PROP-TEXT AS "Text"
   PROPERTY PROP-TEXTBOX1 AS "textBox1"
```

```
          PROPERTY PROP-TEXTBOX2 AS "textBox2"
          PROPERTY PROP-TEXTBOX3 AS "textBox3"
          PROPERTY PROP-USEVISUALSTYLEBACKCOLOR AS "UseVisualStyleBackColor"
          PROPERTY PROP-VALIDATINGTYPE AS "ValidatingType"
          PROPERTY PROP-WARNING AS "Warning".

       OBJECT.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 label1 OBJECT REFERENCE CLASS-LABEL.
       01 radioButton1 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 radioButton2 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 label2 OBJECT REFERENCE CLASS-LABEL.
       01 button1 OBJECT REFERENCE CLASS-BUTTON.
       01 button2 OBJECT REFERENCE CLASS-BUTTON.
       01 textBox1 OBJECT REFERENCE CLASS-TEXTBOX.
       01 textBox2 OBJECT REFERENCE CLASS-TEXTBOX.
       01 textBox3 OBJECT REFERENCE CLASS-TEXTBOX.
       01 label3 OBJECT REFERENCE CLASS-LABEL.
       01 components OBJECT REFERENCE INTERFACE-ICONTAINER.
       PROCEDURE DIVISION.

       METHOD-ID. DISPOSE AS "Dispose" OVERRIDE PROTECTED.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 disposing OBJECT REFERENCE CLASS-BOOLEAN.
       PROCEDURE DIVISION USING BY VALUE disposing.
          IF disposing NOT = B"0" AND components NOT = NULL THEN
            INVOKE components "Dispose"
          END-IF.
          INVOKE SUPER "Dispose" USING disposing.
       END METHOD DISPOSE.

       METHOD-ID. INITIALIZECOMPONENT AS "InitializeComponent" PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 TEMP1 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP2 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP3 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP4 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP5 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP6 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP7 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP8 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP9 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP10 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP11 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP12 BINARY-LONG.
       01 TEMP13 BINARY-LONG.
       01 TEMP14 OBJECT REFERENCE CLASS-POINT.
       01 TEMP15 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP16 OBJECT REFERENCE CLASS-STRING.
       01 TEMP17 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP18 BINARY-LONG.
       01 TEMP19 BINARY-LONG.
       01 TEMP20 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP21 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP22 BINARY-LONG.
       01 TEMP23 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP24 OBJECT REFERENCE CLASS-STRING.
       01 TEMP25 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP26 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP27 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP28 BINARY-LONG.
       01 TEMP29 BINARY-LONG.
       01 TEMP30 OBJECT REFERENCE CLASS-POINT.
       01 TEMP31 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP32 OBJECT REFERENCE CLASS-STRING.
       01 TEMP33 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP34 BINARY-LONG.
       01 TEMP35 BINARY-LONG.
       01 TEMP36 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP37 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP38 BINARY-LONG.
       01 TEMP39 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP40 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP41 OBJECT REFERENCE CLASS-STRING.
       01 TEMP42 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP43 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP44 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP45 BINARY-LONG.
```

```
       01 TEMP46 BINARY-LONG.
       01 TEMP47 OBJECT REFERENCE CLASS-POINT.
       01 TEMP48 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP49 OBJECT REFERENCE CLASS-STRING.
       01 TEMP50 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP51 BINARY-LONG.
       01 TEMP52 BINARY-LONG.
       01 TEMP53 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP54 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP55 BINARY-LONG.
       01 TEMP56 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP57 OBJECT REFERENCE CLASS-STRING.
       01 TEMP58 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP59 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP60 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP61 BINARY-LONG.
       01 TEMP62 BINARY-LONG.
       01 TEMP63 OBJECT REFERENCE CLASS-POINT.
       01 TEMP64 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP65 OBJECT REFERENCE CLASS-STRING.
       01 TEMP66 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP67 BINARY-LONG.
       01 TEMP68 BINARY-LONG.
       01 TEMP69 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP70 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP71 BINARY-LONG.
       01 TEMP72 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP73 OBJECT REFERENCE CLASS-STRING.
       01 TEMP74 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP75 BINARY-LONG.
       01 TEMP76 BINARY-LONG.
       01 TEMP77 OBJECT REFERENCE CLASS-POINT.
       01 TEMP78 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP79 OBJECT REFERENCE CLASS-STRING.
       01 TEMP80 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP81 BINARY-LONG.
       01 TEMP82 BINARY-LONG.
       01 TEMP83 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP84 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP85 BINARY-LONG.
       01 TEMP86 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP87 OBJECT REFERENCE CLASS-STRING.
       01 TEMP88 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP89 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP90 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP91 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP92 BINARY-LONG.
       01 TEMP93 BINARY-LONG.
       01 TEMP94 OBJECT REFERENCE CLASS-POINT.
       01 TEMP95 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP96 OBJECT REFERENCE CLASS-STRING.
       01 TEMP97 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP98 BINARY-LONG.
       01 TEMP99 BINARY-LONG.
       01 TEMP100 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP101 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP102 BINARY-LONG.
       01 TEMP103 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP104 OBJECT REFERENCE CLASS-STRING.
       01 TEMP105 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP106 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP107 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP108 OBJECT REFERENCE DELEGATE-EVENTHANDLER.
       01 TEMP109 BINARY-LONG.
       01 TEMP110 BINARY-LONG.
       01 TEMP111 OBJECT REFERENCE CLASS-POINT.
       01 TEMP112 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP113 OBJECT REFERENCE CLASS-STRING.
       01 TEMP114 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP115 BINARY-LONG.
       01 TEMP116 BINARY-LONG.
       01 TEMP117 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP118 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP119 BINARY-LONG.
       01 TEMP120 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP121 BINARY-LONG.
       01 TEMP122 BINARY-LONG.
       01 TEMP123 OBJECT REFERENCE CLASS-POINT.
       01 TEMP124 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP125 OBJECT REFERENCE CLASS-STRING.
       01 TEMP126 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP127 BINARY-LONG.
```

```
       01 TEMP128 BINARY-LONG.
       01 TEMP129 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP130 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP131 BINARY-LONG.
       01 TEMP132 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP133 BINARY-LONG.
       01 TEMP134 BINARY-LONG.
       01 TEMP135 OBJECT REFERENCE CLASS-POINT.
       01 TEMP136 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP137 OBJECT REFERENCE CLASS-STRING.
       01 TEMP138 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP139 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP140 BINARY-LONG.
       01 TEMP141 BINARY-LONG.
       01 TEMP142 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP143 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP144 BINARY-LONG.
       01 TEMP145 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP146 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP147 BINARY-LONG.
       01 TEMP148 BINARY-LONG.
       01 TEMP149 OBJECT REFERENCE CLASS-POINT.
       01 TEMP150 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP151 OBJECT REFERENCE CLASS-STRING.
       01 TEMP152 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP153 BINARY-LONG.
       01 TEMP154 BINARY-LONG.
       01 TEMP155 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP156 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP157 BINARY-LONG.
       01 TEMP158 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP159 OBJECT REFERENCE CLASS-STRING.
       01 TEMP160 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP161 COMP-1.
       01 TEMP162 COMP-1.
       01 TEMP163 OBJECT REFERENCE CLASS-SIZEF.
       01 TEMP164 BINARY-LONG.
       01 TEMP165 BINARY-LONG.
       01 TEMP166 OBJECT REFERENCE CLASS-SIZE.
       01 TEMP167 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP168 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP169 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP170 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP171 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP172 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP173 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP174 OBJECT REFERENCE CLASS-TEXTBOX.
       01 TEMP175 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP176 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP177 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP178 OBJECT REFERENCE CLASS-BUTTON.
       01 TEMP179 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP180 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP181 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP182 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP183 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP184 OBJECT REFERENCE CLASS-LABEL.
       01 TEMP185 OBJECT REFERENCE CLASS-CONTROLCOLLECTION.
       01 TEMP186 OBJECT REFERENCE CLASS-RADIOBUTTON.
       01 TEMP187 OBJECT REFERENCE CLASS-STRING.
       01 TEMP188 OBJECT REFERENCE CLASS-STRING.
       PROCEDURE DIVISION.
      *>>IMP BEGIN-EMBEDDED-CODEDOM
      *<embedded-codedom>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
```

```
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton1" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.RadioButton" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.RadioButton" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Windows.Forms.Label" />
*</prop>
```

```
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Button" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
```

```
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.TextBox" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.Label" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
```

```
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="SuspendLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
```

```
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="40" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label1" />
*</prop>
*</object>
*</prop>
```

```
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="53" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
      *<string value="製品番号" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Checked" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
```

```
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="47" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="16" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="0" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton1" />
*</prop>
*</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabStop" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="入庫" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
*<string value="radioButton2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="AutoSize" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<bool value="True" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="radioButton2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="71" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="47" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="16" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="出庫" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
```

```
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="121" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="40" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="29" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="4" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="数量" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="43" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="85" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="75" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="23" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="8" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="更新" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Listener">
      *<object type="System.CodeDom.CodeDelegateCreateExpression">
      *<prop name="DelegateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.EventHandler" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="button1_Click" />
      *</prop>
      *</object>
      *</prop>
```

```
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="124" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="85" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
```

```
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="75" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="23" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="button2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
```

```
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="9" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="閉じる" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="UseVisualStyleBackColor" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAttachEventStatement">
      *<prop name="Event">
      *<object type="System.CodeDom.CodeEventReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="EventName">
      *<string value="Click" />
      *</prop>
```

```
*</object>
*</prop>
*<prop name="Listener">
*<object type="System.CodeDom.CodeDelegateCreateExpression">
*<prop name="DelegateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.EventHandler" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="MethodName">
*<string value="button2_Click" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Point" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="71" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="37" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="34" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
```

```
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox1" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="TabIndex" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="textBox2" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeCommentStatement">
*<prop name="Comment">
*<object type="System.CodeDom.CodeComment">
*<prop name="Text">
*<string value="" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="textBox2" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Location" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
```

```
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="156" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="37" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
```

```
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="34" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="5" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
```

```
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="156" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="60" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ReadOnly" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
```

```
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Size" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="34" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="19" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="7" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="True" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Location" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Point" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
```

```
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="115" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="63" />
*</prop>
*</object>
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Name" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*</object>
*<object type="System.CodeDom.CodeAssignStatement">
*<prop name="Left">
*<object type="System.CodeDom.CodePropertyReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeFieldReferenceExpression">
*<prop name="TargetObject">
*<object type="System.CodeDom.CodeThisReferenceExpression">
*</object>
*</prop>
*<prop name="FieldName">
*<string value="label3" />
*</prop>
*</object>
*</prop>
*<prop name="PropertyName">
*<string value="Size" />
*</prop>
*</object>
*</prop>
*<prop name="Right">
*<object type="System.CodeDom.CodeObjectCreateExpression">
*<prop name="CreateType">
*<object type="System.CodeDom.CodeTypeReference">
*<prop name="BaseType">
*<string value="System.Drawing.Size" />
*</prop>
*<prop name="Options">
*<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
*</prop>
*</object>
*</prop>
*<prop name="Parameters" method="add">
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="41" />
*</prop>
*</object>
*<object type="System.CodeDom.CodePrimitiveExpression">
*<prop name="Value">
*<int32 value="12" />
*</prop>
```

```
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="TabIndex" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="6" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="在庫量" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
      *<string value="Form1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeCommentStatement">
      *<prop name="Comment">
      *<object type="System.CodeDom.CodeComment">
      *<prop name="Text">
```

```
      *<string value="" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleDimensions" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.SizeF" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="6" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<float32 value="12" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="AutoScaleMode" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.AutoScaleMode" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="Font" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ClientSize" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeObjectCreateExpression">
      *<prop name="CreateType">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Drawing.Size" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="204" />
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<int32 value="120" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
```

```
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox3" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
```

```
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="textBox1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="button1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
```

```
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="label1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Controls" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="Add" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="radioButton2" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="FormBorderStyle" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodeFieldReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeTypeReferenceExpression">
      *<prop name="Type">
      *<object type="System.CodeDom.CodeTypeReference">
      *<prop name="BaseType">
      *<string value="System.Windows.Forms.FormBorderStyle" />
      *</prop>
      *<prop name="Options">
      *<enum type="System.CodeDom.CodeTypeReferenceOptions" value="0" />
      *</prop>
      *</object>
```

```
      *</prop>
      *</object>
      *</prop>
      *<prop name="FieldName">
      *<string value="FixedDialog" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MaximizeBox" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="MinimizeBox" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Name" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="Form1" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="ShowIcon" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
```

```
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeAssignStatement">
      *<prop name="Left">
      *<object type="System.CodeDom.CodePropertyReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="PropertyName">
      *<string value="Text" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Right">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<string value="入出庫入力" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="ResumeLayout" />
      *</prop>
      *</object>
      *</prop>
      *<prop name="Parameters" method="add">
      *<object type="System.CodeDom.CodePrimitiveExpression">
      *<prop name="Value">
      *<bool value="False" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *<object type="System.CodeDom.CodeExpressionStatement">
      *<prop name="Expression">
      *<object type="System.CodeDom.CodeMethodInvokeExpression">
      *<prop name="Method">
      *<object type="System.CodeDom.CodeMethodReferenceExpression">
      *<prop name="TargetObject">
      *<object type="System.CodeDom.CodeThisReferenceExpression">
      *</object>
      *</prop>
      *<prop name="MethodName">
      *<string value="PerformLayout" />
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</prop>
      *</object>
      *</embedded-codedom>
      *>>IMP END-EMBEDDED-CODEDOM
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP1
          SET PROP-LABEL1 OF SELF TO TEMP1
          INVOKE CLASS-RADIOBUTTON "NEW" RETURNING TEMP2
          SET PROP-RADIOBUTTON1 OF SELF TO TEMP2
          INVOKE CLASS-RADIOBUTTON "NEW" RETURNING TEMP3
          SET PROP-RADIOBUTTON2 OF SELF TO TEMP3
          INVOKE CLASS-LABEL "NEW" RETURNING TEMP4
          SET PROP-LABEL2 OF SELF TO TEMP4
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP5
          SET PROP-BUTTON1 OF SELF TO TEMP5
          INVOKE CLASS-BUTTON "NEW" RETURNING TEMP6
```

```
           SET PROP-BUTTON2 OF SELF TO TEMP6
           INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP7
           SET PROP-TEXTBOX1 OF SELF TO TEMP7
           INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP8
           SET PROP-TEXTBOX2 OF SELF TO TEMP8
           INVOKE CLASS-TEXTBOX "NEW" RETURNING TEMP9
           SET PROP-TEXTBOX3 OF SELF TO TEMP9
           INVOKE CLASS-LABEL "NEW" RETURNING TEMP10
           SET PROP-LABEL3 OF SELF TO TEMP10
           INVOKE SELF "SuspendLayout"
       *
       *label1
       *
           SET TEMP11 TO PROP-LABEL1 OF SELF
           SET PROP-AUTOSIZE OF TEMP11 TO B"1"
           MOVE 12 TO TEMP12
           MOVE 40 TO TEMP13
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP12 BY VALUE TEMP13 RETURNING TEMP14
           SET TEMP15 TO PROP-LABEL1 OF SELF
           SET PROP-LOCATION OF TEMP15 TO TEMP14
           SET TEMP16 TO N"label1"
           SET TEMP17 TO PROP-LABEL1 OF SELF
           SET PROP-NAME OF TEMP17 TO TEMP16
           MOVE 53 TO TEMP18
           MOVE 12 TO TEMP19
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP18 BY VALUE TEMP19 RETURNING TEMP20
           SET TEMP21 TO PROP-LABEL1 OF SELF
           SET PROP-SIZE OF TEMP21 TO TEMP20
           MOVE 2 TO TEMP22
           SET TEMP23 TO PROP-LABEL1 OF SELF
           MOVE TEMP22 TO PROP-TABINDEX OF TEMP23
           SET TEMP24 TO N"製品番号"
           SET TEMP25 TO PROP-LABEL1 OF SELF
           SET PROP-TEXT OF TEMP25 TO TEMP24
       *
       *radioButton1
       *
           SET TEMP26 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-AUTOSIZE OF TEMP26 TO B"1"
           SET TEMP27 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-CHECKED OF TEMP27 TO B"1"
           MOVE 12 TO TEMP28
           MOVE 12 TO TEMP29
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP28 BY VALUE TEMP29 RETURNING TEMP30
           SET TEMP31 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-LOCATION OF TEMP31 TO TEMP30
           SET TEMP32 TO N"radioButton1"
           SET TEMP33 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-NAME OF TEMP33 TO TEMP32
           MOVE 47 TO TEMP34
           MOVE 16 TO TEMP35
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP34 BY VALUE TEMP35 RETURNING TEMP36
           SET TEMP37 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-SIZE OF TEMP37 TO TEMP36
           MOVE 0 TO TEMP38
           SET TEMP39 TO PROP-RADIOBUTTON1 OF SELF
           MOVE TEMP38 TO PROP-TABINDEX OF TEMP39
           SET TEMP40 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-TABSTOP OF TEMP40 TO B"1"
           SET TEMP41 TO N"入庫"
           SET TEMP42 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-TEXT OF TEMP42 TO TEMP41
           SET TEMP43 TO PROP-RADIOBUTTON1 OF SELF
           SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP43 TO B"1"
       *
       *radioButton2
       *
           SET TEMP44 TO PROP-RADIOBUTTON2 OF SELF
           SET PROP-AUTOSIZE OF TEMP44 TO B"1"
           MOVE 71 TO TEMP45
           MOVE 12 TO TEMP46
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP45 BY VALUE TEMP46 RETURNING TEMP47
           SET TEMP48 TO PROP-RADIOBUTTON2 OF SELF
           SET PROP-LOCATION OF TEMP48 TO TEMP47
           SET TEMP49 TO N"radioButton2"
           SET TEMP50 TO PROP-RADIOBUTTON2 OF SELF
           SET PROP-NAME OF TEMP50 TO TEMP49
           MOVE 47 TO TEMP51
           MOVE 16 TO TEMP52
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP51 BY VALUE TEMP52 RETURNING TEMP53
           SET TEMP54 TO PROP-RADIOBUTTON2 OF SELF
```

```
           SET PROP-SIZE OF TEMP54 TO TEMP53
           MOVE 1 TO TEMP55
           SET TEMP56 TO PROP-RADIOBUTTON2 OF SELF
           MOVE TEMP55 TO PROP-TABINDEX OF TEMP56
           SET TEMP57 TO N"出庫"
           SET TEMP58 TO PROP-RADIOBUTTON2 OF SELF
           SET PROP-TEXT OF TEMP58 TO TEMP57
           SET TEMP59 TO PROP-RADIOBUTTON2 OF SELF
           SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP59 TO B"1"
       *
       *label2
       *
           SET TEMP60 TO PROP-LABEL2 OF SELF
           SET PROP-AUTOSIZE OF TEMP60 TO B"1"
           MOVE 121 TO TEMP61
           MOVE 40 TO TEMP62
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP61 BY VALUE TEMP62 RETURNING TEMP63
           SET TEMP64 TO PROP-LABEL2 OF SELF
           SET PROP-LOCATION OF TEMP64 TO TEMP63
           SET TEMP65 TO N"label2"
           SET TEMP66 TO PROP-LABEL2 OF SELF
           SET PROP-NAME OF TEMP66 TO TEMP65
           MOVE 29 TO TEMP67
           MOVE 12 TO TEMP68
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP67 BY VALUE TEMP68 RETURNING TEMP69
           SET TEMP70 TO PROP-LABEL2 OF SELF
           SET PROP-SIZE OF TEMP70 TO TEMP69
           MOVE 4 TO TEMP71
           SET TEMP72 TO PROP-LABEL2 OF SELF
           MOVE TEMP71 TO PROP-TABINDEX OF TEMP72
           SET TEMP73 TO N"数量"
           SET TEMP74 TO PROP-LABEL2 OF SELF
           SET PROP-TEXT OF TEMP74 TO TEMP73
       *
       *button1
       *
           MOVE 43 TO TEMP75
           MOVE 85 TO TEMP76
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP75 BY VALUE TEMP76 RETURNING TEMP77
           SET TEMP78 TO PROP-BUTTON1 OF SELF
           SET PROP-LOCATION OF TEMP78 TO TEMP77
           SET TEMP79 TO N"button1"
           SET TEMP80 TO PROP-BUTTON1 OF SELF
           SET PROP-NAME OF TEMP80 TO TEMP79
           MOVE 75 TO TEMP81
           MOVE 23 TO TEMP82
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP81 BY VALUE TEMP82 RETURNING TEMP83
           SET TEMP84 TO PROP-BUTTON1 OF SELF
           SET PROP-SIZE OF TEMP84 TO TEMP83
           MOVE 8 TO TEMP85
           SET TEMP86 TO PROP-BUTTON1 OF SELF
           MOVE TEMP85 TO PROP-TABINDEX OF TEMP86
           SET TEMP87 TO N"更新"
           SET TEMP88 TO PROP-BUTTON1 OF SELF
           SET PROP-TEXT OF TEMP88 TO TEMP87
           SET TEMP89 TO PROP-BUTTON1 OF SELF
           SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP89 TO B"1"
           SET TEMP90 TO PROP-BUTTON1 OF SELF
           INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button1_Click"
                  RETURNING TEMP91
           INVOKE TEMP90 "add_Click" USING BY VALUE TEMP91
       *
       *button2
       *
           MOVE 124 TO TEMP92
           MOVE 85 TO TEMP93
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP92 BY VALUE TEMP93 RETURNING TEMP94
           SET TEMP95 TO PROP-BUTTON2 OF SELF
           SET PROP-LOCATION OF TEMP95 TO TEMP94
           SET TEMP96 TO N"button2"
           SET TEMP97 TO PROP-BUTTON2 OF SELF
           SET PROP-NAME OF TEMP97 TO TEMP96
           MOVE 75 TO TEMP98
           MOVE 23 TO TEMP99
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP98 BY VALUE TEMP99 RETURNING TEMP100
           SET TEMP101 TO PROP-BUTTON2 OF SELF
           SET PROP-SIZE OF TEMP101 TO TEMP100
           MOVE 9 TO TEMP102
           SET TEMP103 TO PROP-BUTTON2 OF SELF
           MOVE TEMP102 TO PROP-TABINDEX OF TEMP103
           SET TEMP104 TO N"閉じる"
```

```
           SET TEMP105 TO PROP-BUTTON2 OF SELF
           SET PROP-TEXT OF TEMP105 TO TEMP104
           SET TEMP106 TO PROP-BUTTON2 OF SELF
           SET PROP-USEVISUALSTYLEBACKCOLOR OF TEMP106 TO B"1"
           SET TEMP107 TO PROP-BUTTON2 OF SELF
           INVOKE DELEGATE-EVENTHANDLER "NEW" USING BY VALUE SELF BY VALUE N"button2_Click"
                  RETURNING TEMP108
           INVOKE TEMP107 "add_Click" USING BY VALUE TEMP108
       *
       *textBox1
       *
           MOVE 71 TO TEMP109
           MOVE 37 TO TEMP110
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP109 BY VALUE TEMP110 RETURNING TEMP111
           SET TEMP112 TO PROP-TEXTBOX1 OF SELF
           SET PROP-LOCATION OF TEMP112 TO TEMP111
           SET TEMP113 TO N"textBox1"
           SET TEMP114 TO PROP-TEXTBOX1 OF SELF
           SET PROP-NAME OF TEMP114 TO TEMP113
           MOVE 34 TO TEMP115
           MOVE 19 TO TEMP116
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP115 BY VALUE TEMP116 RETURNING TEMP117
           SET TEMP118 TO PROP-TEXTBOX1 OF SELF
           SET PROP-SIZE OF TEMP118 TO TEMP117
           MOVE 3 TO TEMP119
           SET TEMP120 TO PROP-TEXTBOX1 OF SELF
           MOVE TEMP119 TO PROP-TABINDEX OF TEMP120
       *
       *textBox2
       *
           MOVE 156 TO TEMP121
           MOVE 37 TO TEMP122
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP121 BY VALUE TEMP122 RETURNING TEMP123
           SET TEMP124 TO PROP-TEXTBOX2 OF SELF
           SET PROP-LOCATION OF TEMP124 TO TEMP123
           SET TEMP125 TO N"textBox2"
           SET TEMP126 TO PROP-TEXTBOX2 OF SELF
           SET PROP-NAME OF TEMP126 TO TEMP125
           MOVE 34 TO TEMP127
           MOVE 19 TO TEMP128
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP127 BY VALUE TEMP128 RETURNING TEMP129
           SET TEMP130 TO PROP-TEXTBOX2 OF SELF
           SET PROP-SIZE OF TEMP130 TO TEMP129
           MOVE 5 TO TEMP131
           SET TEMP132 TO PROP-TEXTBOX2 OF SELF
           MOVE TEMP131 TO PROP-TABINDEX OF TEMP132
       *
       *textBox3
       *
           MOVE 156 TO TEMP133
           MOVE 60 TO TEMP134
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP133 BY VALUE TEMP134 RETURNING TEMP135
           SET TEMP136 TO PROP-TEXTBOX3 OF SELF
           SET PROP-LOCATION OF TEMP136 TO TEMP135
           SET TEMP137 TO N"textBox3"
           SET TEMP138 TO PROP-TEXTBOX3 OF SELF
           SET PROP-NAME OF TEMP138 TO TEMP137
           SET TEMP139 TO PROP-TEXTBOX3 OF SELF
           SET PROP-READONLY OF TEMP139 TO B"1"
           MOVE 34 TO TEMP140
           MOVE 19 TO TEMP141
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP140 BY VALUE TEMP141 RETURNING TEMP142
           SET TEMP143 TO PROP-TEXTBOX3 OF SELF
           SET PROP-SIZE OF TEMP143 TO TEMP142
           MOVE 7 TO TEMP144
           SET TEMP145 TO PROP-TEXTBOX3 OF SELF
           MOVE TEMP144 TO PROP-TABINDEX OF TEMP145
       *
       *label3
       *
           SET TEMP146 TO PROP-LABEL3 OF SELF
           SET PROP-AUTOSIZE OF TEMP146 TO B"1"
           MOVE 115 TO TEMP147
           MOVE 63 TO TEMP148
           INVOKE CLASS-POINT "NEW" USING BY VALUE TEMP147 BY VALUE TEMP148 RETURNING TEMP149
           SET TEMP150 TO PROP-LABEL3 OF SELF
           SET PROP-LOCATION OF TEMP150 TO TEMP149
           SET TEMP151 TO N"label3"
           SET TEMP152 TO PROP-LABEL3 OF SELF
           SET PROP-NAME OF TEMP152 TO TEMP151
           MOVE 41 TO TEMP153
           MOVE 12 TO TEMP154
```

```
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP153 BY VALUE TEMP154 RETURNING TEMP155
           SET TEMP156 TO PROP-LABEL3 OF SELF
           SET PROP-SIZE OF TEMP156 TO TEMP155
           MOVE 6 TO TEMP157
           SET TEMP158 TO PROP-LABEL3 OF SELF
           MOVE TEMP157 TO PROP-TABINDEX OF TEMP158
           SET TEMP159 TO N"在庫量"
           SET TEMP160 TO PROP-LABEL3 OF SELF
           SET PROP-TEXT OF TEMP160 TO TEMP159
       *
       *Form1
       *
           MOVE 6.00000000000000E+00 TO TEMP161
           MOVE 1.20000000000000E+01 TO TEMP162
           INVOKE CLASS-SIZEF "NEW" USING BY VALUE TEMP161 BY VALUE TEMP162 RETURNING TEMP163
           SET PROP-AUTOSCALEDIMENSIONS OF SELF TO TEMP163
           SET PROP-AUTOSCALEMODE OF SELF TO PROP-FONT OF ENUM-AUTOSCALEMODE
           MOVE 204 TO TEMP164
           MOVE 120 TO TEMP165
           INVOKE CLASS-SIZE "NEW" USING BY VALUE TEMP164 BY VALUE TEMP165 RETURNING TEMP166
           SET PROP-CLIENTSIZE OF SELF TO TEMP166
           SET TEMP168 TO PROP-LABEL3 OF SELF
           SET TEMP167 TO PROP-CONTROLS OF SELF
           INVOKE TEMP167 "Add" USING BY VALUE TEMP168
           SET TEMP170 TO PROP-TEXTBOX3 OF SELF
           SET TEMP169 TO PROP-CONTROLS OF SELF
           INVOKE TEMP169 "Add" USING BY VALUE TEMP170
           SET TEMP172 TO PROP-TEXTBOX2 OF SELF
           SET TEMP171 TO PROP-CONTROLS OF SELF
           INVOKE TEMP171 "Add" USING BY VALUE TEMP172
           SET TEMP174 TO PROP-TEXTBOX1 OF SELF
           SET TEMP173 TO PROP-CONTROLS OF SELF
           INVOKE TEMP173 "Add" USING BY VALUE TEMP174
           SET TEMP176 TO PROP-BUTTON2 OF SELF
           SET TEMP175 TO PROP-CONTROLS OF SELF
           INVOKE TEMP175 "Add" USING BY VALUE TEMP176
           SET TEMP178 TO PROP-BUTTON1 OF SELF
           SET TEMP177 TO PROP-CONTROLS OF SELF
           INVOKE TEMP177 "Add" USING BY VALUE TEMP178
           SET TEMP180 TO PROP-RADIOBUTTON1 OF SELF
           SET TEMP179 TO PROP-CONTROLS OF SELF
           INVOKE TEMP179 "Add" USING BY VALUE TEMP180
           SET TEMP182 TO PROP-LABEL2 OF SELF
           SET TEMP181 TO PROP-CONTROLS OF SELF
           INVOKE TEMP181 "Add" USING BY VALUE TEMP182
           SET TEMP184 TO PROP-LABEL1 OF SELF
           SET TEMP183 TO PROP-CONTROLS OF SELF
           INVOKE TEMP183 "Add" USING BY VALUE TEMP184
           SET TEMP186 TO PROP-RADIOBUTTON2 OF SELF
           SET TEMP185 TO PROP-CONTROLS OF SELF
           INVOKE TEMP185 "Add" USING BY VALUE TEMP186
           SET PROP-FORMBORDERSTYLE OF SELF TO PROP-FIXEDDIALOG OF ENUM-FORMBORDERSTYLE
           SET PROP-MAXIMIZEBOX OF SELF TO B"0"
           SET PROP-MINIMIZEBOX OF SELF TO B"0"
           SET TEMP187 TO N"Form1"
           SET PROP-NAME OF SELF TO TEMP187
           SET PROP-SHOWICON OF SELF TO B"0"
           SET TEMP188 TO N"入出庫入力"
           SET PROP-TEXT OF SELF TO TEMP188
           INVOKE SELF "ResumeLayout" USING BY VALUE B"0"
           INVOKE SELF "PerformLayout"
       END METHOD INITIALIZECOMPONENT.

       METHOD-ID. NEW.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
           INVOKE SELF "InitializeComponent".
       END METHOD NEW.

       METHOD-ID. button2_Click PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.
           INVOKE SELF "Close".
       END METHOD button2_Click.

       METHOD-ID. button1_Click PRIVATE.
```

```
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 PROXY OBJECT REFERENCE CLASS-MANAGESTOCKPROXY.
       01 SCOPE  OBJECT REFERENCE CLASS-TRANSACTIONSCOPE.
       01 KIND    PIC S9(4) COMP-5.
       01 GOODSNO  PIC S9(4) COMP-5.
       01 QUANTITY PIC S9(9) COMP-5.
       01 UPDATED  PIC S9(9) COMP-5.
       01 WK-VAL OBJECT REFERENCE CLASS-STRING.
       01 WK-EXP OBJECT REFERENCE CLASS-EXCEPTION.
       LINKAGE SECTION.
       01 sender OBJECT REFERENCE CLASS-OBJECT.
       01 e OBJECT REFERENCE CLASS-EVENTARGS.
       PROCEDURE DIVISION USING BY VALUE sender e.

           *> チャネルを開きます
           SET PROXY TO CLASS-MANAGESTOCKPROXY::"NEW"(N"WSHttpBinding_IManageStockService")
           TRY
               TRY
                   *> トランザクションスコープを開始します
                   SET SCOPE TO CLASS-TRANSACTIONSCOPE::"NEW"

                   SET WK-VAL TO PROP-TEXT OF textBox1
                   SET GOODSNO TO CLASS-INT16::"Parse"(WK-VAL)
                   SET WK-VAL TO PROP-TEXT OF textBox2
                   SET QUANTITY TO CLASS-INT32::"Parse"(WK-VAL)

                   *> オペレーションを呼び出します
                   IF PROP-CHECKED OF radioButton1 = B"1" THEN
                       INVOKE PROXY "Stock" USING GOODSNO QUANTITY RETURNING UPDATED
                   ELSE
                       INVOKE PROXY "Delivery" USING GOODSNO QUANTITY RETURNING UPDATED
                   END-IF

                   IF UPDATED < 0 THEN
                       SET PROP-TEXT OF textBox3 TO N"---"
                       INVOKE SELF "ShowMessage"
                              USING N"エラーが発生したため更新処理はロールバックされました" "W"
                   ELSE
                       SET WK-VAL TO CLASS-STRING::"Format"(N"{0}" UPDATED)
                       SET PROP-TEXT OF textBox3 TO WK-VAL
                       *> 正常に終了した場合はトランザクションを完了します
                       INVOKE SCOPE "Complete"
                       INVOKE SELF "ShowMessage" USING N"更新処理はコミットされました" "I"
                   END-IF
               FINALLY
                  *> トランザクションスコープを終了します
                  INVOKE SCOPE "Dispose"
               END-TRY
           CATCH WK-EXP
              INVOKE SELF "ShowMessage" USING  N"更新処理中に例外が発生しました" "E"
           END-TRY.

           *> チャネルを閉じます
           INVOKE PROXY "Close".

       END METHOD button1_Click.


       METHOD-ID. ShowMessage PRIVATE.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 CAP  OBJECT REFERENCE CLASS-STRING.
       01 BTN OBJECT REFERENCE ENUM-MESSAGEBOXBUTTONS.
       01 ICO OBJECT REFERENCE ENUM-MESSAGEBOXICON.
       LINKAGE SECTION.
       01 MSG OBJECT REFERENCE CLASS-STRING.
       01 FLG PIC X.
       PROCEDURE DIVISION USING MSG FLG.
           *> メッセージを表示します
           SET CAP TO PROP-TEXT OF SELF.
           SET BTN TO PROP-OK OF ENUM-MESSAGEBOXBUTTONS
           EVALUATE FLG
           WHEN "E"
               SET ICO TO PROP-ERROR OF ENUM-MESSAGEBOXICON
           WHEN "W"
               SET ICO TO PROP-WARNING OF ENUM-MESSAGEBOXICON
           WHEN "I"
               SET ICO TO PROP-INFORMATION OF ENUM-MESSAGEBOXICON
           END-EVALUATE
```

```
           INVOKE CLASS-MESSAGEBOX "Show" USING SELF MSG CAP BTN ICO
       END METHOD ShowMessage.

       END OBJECT.
       END CLASS CLASS-THIS.
```

ソースコード：WCFManageStockClient¥Main.cob

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN CUSTOM-ATTRIBUTE CA-STATHREAD.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CUSTOM-ATTRIBUTE CA-STATHREAD CLASS CLASS-STATHREADATTRIBUTE
   .
REPOSITORY.
   CLASS CLASS-APPLICATION AS "System.Windows.Forms.Application"
   CLASS CLASS-MAINFORM AS "WCFManageStockClient.Form1"
   CLASS CLASS-STATHREADATTRIBUTE AS "System.STAThreadAttribute"
   .

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-MAINFORM OBJECT REFERENCE CLASS-MAINFORM.
PROCEDURE DIVISION.
   INVOKE CLASS-APPLICATION "EnableVisualStyles".
   INVOKE CLASS-APPLICATION "SetCompatibleTextRenderingDefault" USING BY VALUE B"0".

   INVOKE CLASS-MAINFORM "NEW" RETURNING WK-MAINFORM.
   INVOKE CLASS-APPLICATION "Run" USING BY VALUE WK-MAINFORM.
END PROGRAM MAIN.
```

NCOBU01200

例題集

例題集として、Samples¥Cobol フォルダにサンプルプログラムを提供します。

1. 標準入出力を使ったデータ処理(Sample01)
2. 行順ファイルと索引ファイルの操作(Sample02)
3. COBOL プログラム間の呼出し(Sample05)
4. コマンド行引数の受取り方(Sample06)
5. 環境変数の操作(Sample07)
6. 印刷ファイルを使ったプログラム(Sample08)
7. 印刷ファイルを使ったプログラム（応用編）(Sample09)
8. FORMAT 句付き印刷ファイルを使ったプログラム(Sample10)
9. データベース機能を使ったプログラム(Sample11)
10. データベース機能を使ったプログラム（応用編）(Sample12)
11. Visual Basic からの呼出し(Sample13)
12. Visual Basic を使った簡易 ATM 端末処理機能(Sample14)
13. オブジェクト指向プログラム（初級編）(Sample15)
14. コレクションクラス（クラスライブラリ）(Sample16)
15. メッセージボックスの出力(Sample31)
16. 他のプログラムの起動(Sample32)

以下のいずれかの方法によってサンプルプログラムをビルドおよび実行します。

1 つ目の方法は各サンプルのフォルダ直下に含まれるバッチファイル（SampleXX.bat）を実行します。 バッチファイルの実行によって、プログラムがビルドされ、実行可能ファイルが作成されます。 この場合、MSBuild が利用されるため、 バッチファイルは NetCOBOL for .NET コマンドプロンプト上で実行されなければなりません。 [参照] "コマンドプロンプトの起動"

2 つ目の方法は、Visual Studio でサンプルプログラムのソリューションファイル(.sln)をオープンして、 サンプルプルグラムをビルドおよび実行します。

すべてのサンプルプログラムをビルドしたい場合は、1 つ目の方法と同様に、 Samples¥Cobol フォルダにある BUILD_ALL.bat を実行してください。

1. 標準入出力を使ったデータ処理(Sample01)

Sample01 は、COBOL の小入出力機能を使って、コンソールウィンドウからデータを入力したり、 コンソールウィンドウにデータを出力したりするプログラムの例を示します。

概要

コンソールウィンドウからアルファベット 1 文字(小文字)を入力し、入力したアルファベットで始まる単語をコンソールウィンドウに出力します。

使用している COBOL の機能

- 小入出力機能(コンソールウィンドウ)

使用している COBOL の文

- ACCEPT 文
- DISPLAY 文
- EXIT 文
- IF 文
- PERFORM 文

2. 行順ファイルと索引ファイルの操作(Sample02)

エディターを使って作成したデータファイル(行順ファイル)を読み込み、マスタファイル(索引ファイル)を作成するプログラムの例を示します。 行順ファイルおよび索引ファイルの使い方の詳細は、ファイルの処理を参照してください。

概要

商品コード、商品名および単価が入力されているデータファイル(行順ファイル)を読み込み、 商品コードを主レコードキー、商品名を副レコードキーとする索引ファイルを作成します。

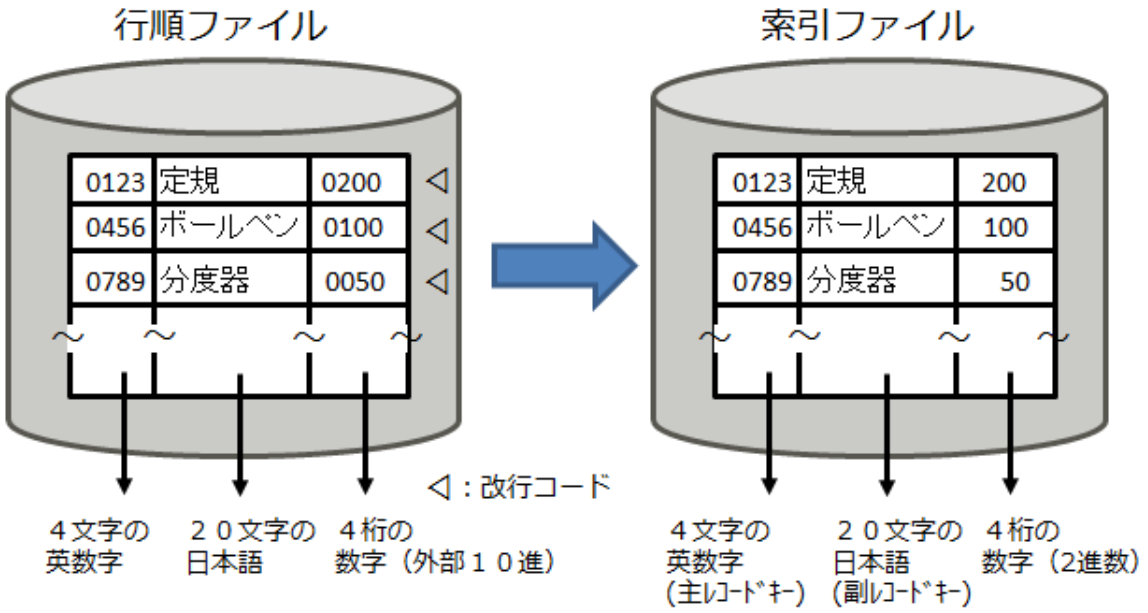
使用している COBOL の機能

- 行順ファイル(参照)
- 索引ファイル(創成)

使用している COBOL の文

- CLOSE 文
- EXIT 文
- GO TO 文
- MOVE 文

- OPEN 文
- READ 文
- WRITE 文
- COPY 文



3. COBOL プログラム間の呼出し(Sample05)

Sample05 では、主プログラムから、副プログラムを呼び出すプログラムの例を示します。 なお、Sample05 では、正書法の自由形式を使用して記述しています。

概要

商品コード、商品名および単価が格納されているマスタファイルの内容を印刷可能文字に変換して作業用のテキストファイル(*.TMP)に格納し、 印刷します。なお、作業用のファイルの名前は、プログラム実行時にパラメタで指定します。

使用している COBOL の機能

- プログラム間連絡機能
- 登録集の取込み
- 小入出力機能(メッセージ出力)
- 印刷ファイル
- 索引ファイル(参照)
- 行順ファイル(創成)
- 実行時パラメタの受渡し
- 正書法の自由形式

使用している COBOL の文

- CALL 文
- DISPLAY 文
- EXIT 文
- GO TO 文
- MOVE 文
- OPEN 文
- READ 文
- SET 文
- WRITE 文

4. コマンド行引数の受取り方(Sample06)

Sample06 では、コマンド行引数の操作機能を使って、COBOL プログラムの実行時に指定された引数を受け取るプログラムの例を示します。 コマンド行引数の操作機能の使い方は、コマンド行引数の取り出しを参照してください。 また、Sample06 では、内部プログラムの呼出しも行います。

概要

開始年月日から終了年月日までの日数を求めます。 開始年月日および終了年月日は、コマンドの引数として次の形式で指定されます。

```
コマンド名　開始年月日　終了年月日
```

開始年月日および終了年月日は、1900 年 1 月 1 日～2172 年 12 月 31 日までの日付を YYYYMMDD で指定します。 西暦については 4 桁で指定します。

使用している COBOL の機能

- コマンド行引数の取出し
- 内部プログラム

使用している COBOL の文

- ACCEPT 文
- CALL 文
- COMPUTE 文
- COPY 文
- DISPLAY 文
- DIVIDE 文
- EXIT 文

- GO TO 文
- IF 文
- MOVE 文
- PERFORM 文

5. 環境変数の操作(Sample07)

Sample07 では、環境変数の操作機能を使って、COBOL プログラム実行中に環境変数の値を変更するプログラムの例を示します。 環境変数の操作機能の使い方は、 の環境変数の操作機能を参照してください。

概要

商品コード、商品名および単価が格納されているマスタファイル中のデータを、 商品コードの値によって 2 つのマスタファイルに分割します。分割方法および新規に作成する 2 つのマスタファイルのファイル名を以下に示します。

商品コードの値	ファイル名
先頭が"0"	マスタファイル名.a
先頭が"0"以外	マスタファイル名.b

使用している COBOL の機能

- 環境変数の操作機能
- 索引ファイル

使用している COBOL の文

- ACCEPT 文
- CLOSE 文
- DISPLAY 文
- EXIT 文
- GO TO 文
- IF 文
- OPEN 文
- READ 文
- STRING 文
- WRITE 文

6. 印刷ファイルを使ったプログラム(Sample08)

Sample08 では、印刷ファイルを使って、コンソールウィンドウから入力したデータを印刷装置に出力するプログラムの例を示します。 印刷ファイルの使い方の詳細は、行単位のデータを印刷する方法を参照してください。

概要

コンソールウィンドウから英数字のデータ 40 文字を入力し、印刷装置に出力します。

使用している COBOL の機能

- 印刷ファイル
- 小入出力機能(コンソールウィンドウ)

使用している COBOL の文

- ACCEPT 文
- CLOSE 文
- EXIT 文
- GO TO 文
- IF 文
- OPEN 文
- WRITE 文

7. 印刷ファイルを使ったプログラム（応用編）(Sample09)

Sample09 では、FORMAT 句なし印刷ファイルを使って、I 制御レコードを使用したページ形式の設定/変更と、CHARACTER TYPE 句や PRINTING POSITION 句を使用して印字したい文字の修飾および配置(行/桁)を意識して印刷装置に出力するプログラムの例を示します。 FORMAT 句なし印刷ファイルの使い方の詳細は、行単位のデータを印刷する方法およびフォームオーバレイおよび FCB を使う方法を参照してください。

概要

FORMAT 句なし印刷ファイルを使用して帳票印刷を行う場合、主に利用される機能を想定し、 以下の項目について印刷デモを行います。

FCB を使用した 6LPI、8LPI での帳票印刷

FCB を利用した任意の行間隔(6/8LPI)で帳票印刷を行うことを想定し、I 制御レコードによる FCB(LPI)の切り替えを行います。 ソースプログラムには、CHARACTER TYPE 句や PRINTING POSITION 句を記述して、 行間隔(LPI)や文字間隔(CPI)などの行・桁を意識して帳票の体裁を整えます。 以下の帳票印刷を行います。

- A4 用紙を横向きに使用し、1 ページすべての行間隔を 6LPI とした場合の帳票をイメージし、6LPI/10CPI フォーマットのスペーシングチャート形式のフォームオーバレイと重畳印刷します。
- A4 用紙を横向きに使用し、1 ページすべての行間隔を 8LPI とした場合の帳票をイメージし、8LPI/10CPI フォーマットのスペーシングチャート形式のフォームオーバレイと重畳印刷します。

CHARACTER TYPE 句で指定する各種文字属性での印刷

I 制御レコードを使用し、用紙サイズを A4/横向きから B4/横向きに変更し、 これにあわせて FCB も A4/横向き用から B4/横向き用に変更します。 以下の各種文字属性の印字サンプルを印刷装置に出力します。

1. 文字サイズ 1 文字ずつ 3 ポ、7.2 ポ、9 ポ、12 ポ、18 ポ、24 ポ、36 ポ、50 ポ、72 ポ、100 ポ、200 ポ、300 ポの文字サイズを印字します。

ここでは、文字ピッチ指定を省略することにより、 文字サイズに合わせた最適な文字ピッチを COBOL ランタイムシステムに自動算出させます。

2. 文字ピッチ 文字ピッチ 1CPI で 1 文字、2CPI で 2 文字、3CPI で 3 文字、5CPI で 5 文字、6CPI で 6 文字、7.5CPI で 15 文字、20CPI で 20 文字、 24CPI で 24 文字指定します。

ここでは、文字サイズ指定を省略することにより、 文字ピッチに合わせた最適な文字サイズを COBOL ランタイムシステムに自動算出させます。

3. 文字書体 ゴシック、ゴシック半角(文字形態半角)、明朝、明朝半角(文字形態半角)を 10 文字ずつ 2 回繰り返し印字します。
4. 文字回転 縦書き(反時計回りに 90 度回転)、横書きを 10 文字ずつ繰り返し印字します。
5. 文字形態 全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、半各倍角の文字形態指定を 9 文字ずつ印刷します。
6. 上記 5 つの文字属性を組み合わせた印刷を行います。

使用している COBOL の機能

- 印刷ファイル
- 小入出力機能(コンソールウィンドウ)

使用している COBOL の文

- ADD 文
- CLOSE 文
- DISPLAY 文
- IF 文
- MOVE 文
- OPEN 文

- PERFORM 文
- STOP 文
- WRITE 文

8. FORMAT 句付き印刷ファイルを使ったプログラム(Sample10)

Sample10 では、FORMAT 句付き印刷ファイルを使って、集計表を印刷装置に出力するプログラムの例を示します。FORMAT 句付き印刷ファイルの使い方は、帳票定義体を使う印刷ファイルの使い方を参照してください。なお、このプログラムを実行するには、MeFt が必要です。

概要

商品コード、商品名および単価が格納されているマスタファイルと受注日、数量および売上げ金額が格納されている売上げファイルを入力して、 売上集計表を印刷装置に出力します。

使用している COBOL の機能

- FORMAT 句付き印刷ファイル
- 索引ファイル(参照)
- 登録集の取込み
- 小入出力機能(メッセージボックス)

使用している COBOL の文

- OPEN 文
- READ 文
- WRITE 文
- START 文
- CLOSE 文
- PERFORM 文
- DISPLAY 文
- IF 文
- MOVE 文
- SET 文
- GO TO 文
- EXIT 文
- COPY 文
- ADD 文

9. データベース機能を使ったプログラム(Sample11)

Sample11 では、データベース(SQL)機能を使ってデータベースからデータを取り出しホスト変数に格納する例を示します。

データベースはサーバ上に存在し、クライアント側からこれにアクセスします。 データベースのアクセスは、ADO.NET データプロバイダーを使用して行います。

ADO.NET データプロバイダーを使用するデータベースアクセスについては、ADO.NET 概要を参照してください。 また、データベースのセットアップについては、NetCOBOL for .NET サンプルデータベースのセットアップを参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側	サーバ側
・ ODBC ドライバマネージャー ・ ODBC ドライバ ・ ODBC ドライバの必要とする製品	・ データベース ・ データベースに ODBC でアクセスするために必要な製品

概要

サーバのデータベースにアクセスし、データベース上のテーブル“STOCK”に格納されている全データをディスプレイ装置に表示します。 データをすべて参照し終わると、データベースとの接続を切断します。

使用している COBOL の機能

- リモートデータベースアクセス

使用している COBOL の文

- DISPLAY 文
- GO TO 文
- IF 文
- PERFORM 文
- 埋込み SQL 文 (埋込み例外宣言、CONNECT 文、カーソル宣言、OPEN 文、FETCH 文、CLOSE 文、ROLLBACK 文、DISCONNECT 文)

10. データベース機能を使ったプログラム（応用編）(Sample12)

Sample12 では、データベース(SQL)機能のより進んだ使い方として、データベースの複数の行を 1 つの埋込み SQL 文で操作する例を示します。データベースはサーバ上に存在し、クライアント側からこれにアクセスします。データベースのアクセスは、ODBC ドライバを経由して行います。 ODBC ドライバを使用するデータベースアクセスについては、ODBC 概要を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側	サーバ側
・ ODBC ドライバマネージャー ・ ODBC ドライバ ・ ODBC ドライバの必要とする製品	・ データベース ・ データベースに ODBC でアクセスするために必要な製品

概要

Sample11 と同じデータベースにアクセスし、次の操作を行ってからデータベースとの接続を切断します。

- データベース全データの表示
- GOODS の値が“TELEVISION”である行の GNO の値の取り出し
- 取り出した GNO を持つ行の QOH の更新
- GOODS の値が“RADIO”、“SHAVER”、“DRIER”の行の削除
- データベースの全データの再表示

なお、出力結果は翻訳オプション SSOUT を使用して、一部をファイルに出力します。

使用している COBOL の機能

- リモートデータベースアクセス

使用している COBOL の文

- CALL 文
- DISPLAY 文
- GO TO 文
- IF 文
- PERFORM 文
- 埋込み SQL 文 (複数行指定ホスト変数、表指定ホスト変数、埋込み例外宣言、CONNECT 文、カーソル宣言、 OPEN 文、FETCH 文、SELECT 文、DELETE 文、UPDATE 文、CLOSE 文、COMMIT 文、ROLLBACK 文、DISCONNECT 文)

11. Visual Basic からの呼出し(Sample13)

Sample13 では、Visual Basic アプリケーションから、 NetCOBOL for .NET で作成した DLL を呼び出すプログラムの例を示します。

概要

2 つの数値を Visual Basic アプリケーションのテキストボックスから入力し、 [=](イコール)ボタンを押すと、その 2 つの数値が COBOL アプリケーションに渡されます。COBOL アプリケーションは、2 つの数値を乗算し、 結果を文字に編集して Visual Basic アプリケーションに返却します。 Visual Basic アプリケーションでは、返却された編集結果の文字をテキストボックスに出力します。 Visual Basic アプリケーションの引数の型を COBOL データ型に変換し、 COBOL アプリケーションを呼び出す COBOL クラスを作成することにより、 Visual Basic のアプリケーションから COBOL アプリケーションを

呼び出すことができます。

使用している COBOL の機能

- 他のプログラミング言語から COBOL プログラム定義の呼出し

12. Visual Basic を使った簡易 ATM 端末処理機能(Sample14)

Sample14 では、Visual Basic アプリケーションから、 COBOL で作成した DLL の呼出し方法を簡易 ATM 端末処理機能のプログラム例で示します。

概要

サンプルプログラムは、以下の ATM 端末の機能を実現しています。

- 新規口座の作成
- 入金処理
- 出金処理

口座のデータ(口座番号、暗証番号、氏名および貯金額)は、索引ファイルに保存します。 索引ファイルの構造は、以下に示すとおりです。

```
口座番号    9(5) 主レコードキー
暗証番号    9(4)
氏名        N(6)
貯金額      9(6)
```

ATM 端末から要求された機能により、索引ファイル内の任意の口座のレコードのデータを更新します。 Visual Basic アプリケーションの引数の型を COBOL データ型に変換し、COBOL アプリケーションを呼び出す COBOL クラスを作成すると、 Visual Basic のアプリケーションから COBOL アプリケーションを呼び出すことが可能になります。

使用している COBOL の機能

- 他のプログラミング言語から COBOL プログラム定義の呼出し

使用している COBOL の文

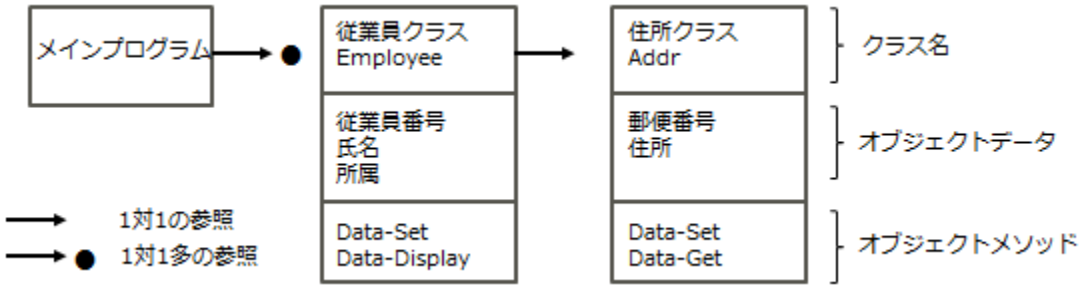
- MOVE 文
- IF 文
- PERFORM 文
- COMPUTE 文
- OPEN 文
- READ 文
- WRITE 文
- REWRITE 文
- CLOSE 文
- EXIT 文

13. オブジェクト指向プログラム（初級編）(Sample15)

Sample15 では、オブジェクト指向プログラミング機能を使ったプログラムの例を示します。 このプログラムでは、カプセル化、オブジェクトの生成、メソッド呼出しといった、 オブジェクト指向の基本的な機能だけを使用しています。

概要

最初に従業員オブジェクトを 3 つ生成しています。 “NEW”メソッドでオブジェクトを生成した後、“Data-Set”を呼び出してデータを設定しています。 それぞれの従業員オブジェクトはすべて同じ形をしていますが、 持っているデータ(従業員番号、氏名、所属、住所情報)は異なります。 また、住所情報もオブジェクトであり、郵便番号と住所を持っています。 画面から従業員番号を入力すると、該当する従業員オブジェクトに対して“Data-Display”メソッドを呼び出し、 そのオブジェクトが持っている従業員情報を画面に表示します。 このとき、従業員オブジェクトは、住所の情報を得るために、従業員オブジェクトが指している住所オブジェクトに対し、“Data-Get”メソッドを呼び出します。 従業員オブジェクトは、3 つのデータと住所オブジェクトから構成されています。 しかし、これを使う側(この場合はメインプログラム)はオブジェクトの構造を知っている必要はありません。 “Data-Set”と“Data-Display”の 2 つのメソッドだけを知っていれば十分です。 つまり、データとアクセス手段を 1 つにまとめる(カプセル化)ことにより、オブジェクト内部の情報を完全に隠蔽しているわけです。



使用している COBOL の機能

オブジェクト指向プログラミング機能

- クラスの定義(カプセル化)
- オブジェクトの生成
- メソッド呼出し

使用しているオブジェクト指向の文/段落/定義

- INVOKE 文、SET 文
- リポジトリ段落
- クラス定義、オブジェクト定義、 メソッド定義

14. コレクションクラス（クラスライブラリ）(Sample16)

Sample16 では、クラスライブラリの作成方法の例として、コレクションクラスの例を示します。 このサンプルは、クラスライブラリの作成方法の例として使用するだけでなく、 実際にプログラムに組み込んで使用することができます。 また、このサンプルには基本的な操作しか含まれていませんが、改造・変更することにより、 さらに使いやすいクラスライブラリにすることができます。

概要

“コレクションクラス”とは、集合を扱うクラスの総称です。 つまり、たくさんのオブジェクトをまとめて扱ったり、格納したりするためのクラスです。 サンプルには、以下のクラスが含まれています。

- Collect(Collection:収集)
- Dict(Dictionary:辞書)
- List(リスト)

15. メッセージボックスの出力(Sample31)

Sample31 では、.NET フレームワークのライブラリクラス“MessageBox”を利用し、メッセージボックスを出力するプログラムの例を示します。 なお、この Sample では COBOL プログラムの復帰値をバッチファイル“CheckSample31.bat”で参照します。

概要

1. CALL 文で Windows システム関数の“MessageBox”を呼び出します。[はい]/[いいえ]/[キャンセル]ボタンを持つメッセージボックスが表示されます。
2. メッセージボックスのボタンを選択します。 メッセージボックスからの復帰値が CALL 文の RETURNING 指定によって返却されます。
3. COBOL プログラムの復帰値をバッチファイルで参照します。 COBOL プログラムの復帰値は、バッチファイルでは ERRORLEVEL という名前で参照することができます。

使用している COBOL の機能

COBOL プログラムから C プログラムを呼び出す方法

- STDCALL 呼出し規約
- BY VALUE でのパラメタの受渡し
- CALL 文の RETURNING 指定
- 特殊レジスタ PROGRAM-STATUS

16. 他のプログラムの起動(Sample32)

Sample32 では、他のプログラムあるいはバッチファイルを起動して、その終了コードを受け取るプログラムの例を示します。

概要

起動するプログラムまたはバッチファイルのパス名と、必要ならコマンド文字列を入力します。この文字列は引数として渡され.NET フレームワークのクラスライブラリが呼び出されます。 渡されたプログラムまたはバッチファイルが起動されます。

使用している COBOL の機能

- BY VALUE でのパラメタの受渡し
- CALL 文の RETURNING 指定
- STORED-CHAR-LENGTH 関数