Interstage Business Application Server V11.0.0

解説書

Windows/Solaris/Linux

B1X1-0175-01Z0(00)
2012年8月

まえがき

■本書の目的

本書は、“Interstage Business Application Server 解説書”です。
本書は、Interstage Business Application Serverの機能概要および運用する上で必要となる基礎知識について説明しています。

本書は、以下の方を対象にしています。

- Interstage Business Application Serverの導入を検討される方および機能を知りたい方

■前提知識

本書を読む場合、以下の知識が必要です。

- 使用するOSに関する基本的な知識
- Javaに関する基本的な知識
- COBOLに関する基本的な知識
- インターネットに関する基本的な知識
- リレーショナルデータベースに関する基本的な知識

■本書の構成

本書は、以下のように構成されています。

第1部 概要

第1章 リリース情報
以前のバージョン・レベルからの追加機能、非互換情報およびプログラム修正の一覧について説明します。

第2章 Interstage Business Application Serverの概要
Interstage Business Application Serverの概要について説明します。

第2部 同期アプリケーション連携実行基盤編

第3章 同期アプリケーション連携実行基盤の機能
同期アプリケーション連携実行基盤の機能について説明します。

第3部 非同期アプリケーション連携実行基盤編 EE

第4章 非同期アプリケーション連携実行基盤の機能
非同期アプリケーション連携実行基盤の機能について説明します。

第4部 共通機能編

第5章 ログ機能
標準ログおよびユーザログの機能について説明します。

第6章 データベースアクセス管理機能
データベースアクセス管理機能の機能について説明します。

第5部 ユーティリティ編

第7章 エクスポートユーティリティ EE
エクスポートユーティリティ機能について説明します。

第8章 アプリケーション安定稼動機能 EE Solaris Linux64
アプリケーション安定稼働機能について説明します。

第6部 導入から運用まで

第9章 導入から運用までの流れ
導入から運用を行うまでの各シーンにおける作業について説明します。

付録A エディション毎の提供機能一覧
Interstage Business Application Serverのエディション毎の提供機能について説明します。

付録B アプリケーション安定稼動機能（V10.0旧版互換）
アプリケーション安定稼動機能（V10.0旧版互換）について説明します。

■著作権

2012年8月　初版

目　次

第1部 概要

第1章 リリース情報

1.1 追加機能の概要

以前のバージョン・レベルより追加された機能を説明します。

アプリケーションサーバ機能として追加された機能は、“Interstage Application Server リリース情報”の“追加機能の概要”を参照してください。

■表記について

以下の表で修正一覧を示します。

項番	VL	機能名	内容	参照マニュアル	SE	EE

項番

通番です。

VL

追加されるバージョンレベルを示します。

機能名

追加機能名を示します。

内容

追加機能の内容を示します。

参照マニュアル

追加機能の情報が記載されているマニュアルの箇所を示します。

SE, EE,

SE: Interstage Business Application Server Standard Edition
EE: Interstage Business Application Server Enterprise Edition
この追加機能がそれぞれのエディションで該当するかどうかを示します。
○:該当します。
ー:このエディションには関係ありません。

■追加機能一覧

項番	VL	機能名	内容	参照マニュアル	SE	EE
1	11.0.0	アプリケーション連携実行基盤	Windows X64 プラットフォームに対応します。	・ セットアップガイド ・ アプリケーション開発ガイド ・ 運用ガイド(アプリケーション連携実行基盤編)	○	○
2	11.0.0	オープンJavaフレームワーク	Spring Framework3.1を提供します。	・ オープンJavaフレームワークユーザーズガイド「Spring Framework3.1」	○	○

1.2 互換に関する情報

■旧バージョン・レベルからの変更

◆トラブル時の一括情報採取ツールの変更

Interstage Business Application Server V10.0.0以降では、トラブル時に調査用の資料採取に使用するコマンドを変更しています。

Windows32

	バージョン・レベルが9.2以前	バージョン・レベルが10.0以降
一括情報採取ツール	[Interstageのインストールディレクトリ]¥bin ¥apfwcollectinfo.exe	[Interstageのインストールディレクトリ]¥bin ¥iscollectinfo.exe

Solaris Linux32/64

	バージョン・レベルが9.2以前	バージョン・レベルが10.0以降
一括情報採取ツール	/opt/FJSVibs/bin/apfwcollectinfo	/opt/FJSVisco/bin/iscollectinfo

◆旧版マニュアルからの変更

V9.0からチュートリアルガイドが追加されています。

V9.0から以下のメッセージのエラー種別を変更しています。

	バージョン・レベルが8.0.1	バージョン・レベルが9.0.0以降
エラー時のメッセージ番号 FSP_INTS-BAS_AP8004	エラー種別は“情報(INFO)”	エラー種別は“エラー(ERROR)”

V9.1.0からログ機能を使用する場合のIPC資源の見積り式を以下のように変更しています。詳細は、“Interstage Business Application Server チューニングガイド”の“ログ機能を使用する場合のチューニング”を参照してください。

Solaris

Solaris 10の場合

共用メモリ		バージョン・レベルが9.0.0以前	バージョン・レベルが9.1.0以降
パラメタ	種類		
project.max-shm-memory	加算値	起動するログ出力サービス数 * (20 + (maxMsgSize + 33) * maxMsgCount) + 15166264	起動するログ出力サービス数 * (1100 + (maxMsgSize + 33) * maxMsgCount) + 15167344

Linux32/64

共用メモリ		バージョン・レベルが9.0.0以前	バージョン・レベルが9.1.0以降
パラメタ	種類		
kernel.shmmax	設定値	以下の値のうち、最大値を指定 起動するログ出力サービス数 * (20 + (maxMsgSize + 33) * maxMsgCount)	以下の値のうち、最大値を指定 起動するログ出力サービス数 * (4200 + (maxMsgSize + 33) * maxMsgCount)

◆同期アプリケーション連携実行基盤におけるサーバアプリケーション名指定時の動作変更

V9.0以降では、同期アプリケーション連携実行基盤において、業務共通制御の振分け制御でサーバアプリケーション名を指定できるようになりました。そのため、次の場合の動作が変更となります。

- クライアントアプリケーションで指定するサーバアプリケーション名が空文字の場合

	バージョン・レベルが8.0.1	バージョン・レベルが9.0.0以降	
		振分け制御で正しいサーバアプリケーション名を指定した場合	振分け制御でサーバアプリケーション名を指定しない場合、または指定した名前のサーバアプリケーションが存在しない場合
サーバの動作	- (リクエストはサーバへ届きません)	指定したサーバアプリケーションを呼び出します。	メッセージを出力します。
クライアントAPIの動作	IllegalArgumentExceptionをthrowします。	正常復帰します。	ApfwSystemExceptionをthrowします。
エラー時のメッセージ番号	FSP_INTS-BAS_AP20005	-	FSP_INTS-BAS_AP20103

◆同期アプリケーション連携実行基盤のバージョン組み合わせ

旧バージョン・レベルの同期アプリケーション連携実行基盤との組み合わせは以下のようになります。

V9.1以前では、クライアントとサーバは同一筐体のみをサポートします。

クライアント		サーバ	
バージョン	使用法	V9.2.0、V9.2.1	V10.0.0、V10.1.0、V11.0.0
V9.2.0、V9.2.1	Apcoordinator連携 MsyncCall	○(同一筐体のみ)	×
	JCA	○(別筐体可、J2EEのIJServerのみ)	○(J2EEのIJServerのみ)
V10.0、V10.1、V11.0	Apcoordinator連携 MsyncCall	×	○(同一筐体のみ)
	JCA	△(J2EEのIJServerのみ)	○

[○:使用可 △:一部使用不可 ×:使用不可]

◆TERASOLUNAフレームワークのバージョン変更

TERASOLUNAフレームワークのベースとなるバージョンが変更されました。

フレームワーク名	V9.2.0 V9.2.1 (RHEL6以外)	V9.2.1 (RHEL6のみ)	V10.0.0 Windows32	V10.0.0 Solaris Linux32/64	V10.1.0	V11.0.0
TERASOLUNA Server Framework for Java (Web版)	V2.0.2.0	V2.0.3.0	V2.0.3.0	V2.0.3.0	V2.0.3.1	V2.0.4.0
TERASOLUNA Server Framework for Java (Rich版)	V2.0.2.0	V2.0.3.0	V2.0.3.0	V2.0.3.1	V2.0.3.1	V2.0.4.0
TERASOLUNA Batch Framework for Java	V2.0.1.0	V2.0.2.0	V2.0.3.0	V2.0.3.1	V2.0.3.1	V2.0.3.2

◆オープンJavaフレームワークが使用するライブラリのバージョン変更

Interstage Business Application Server V9.2.0で提供されたオープンJavaフレームワークを使用する場合に、クラスパスへ設定するjarのファイル名に一部変更があります。

Strutsで使用するクラスパス

項番	V9.2.0、V9.2.1	V10.0.0、V10.1.0	V11.0.0
1	commons-beanutils-1.8.0.jar	commons-beanutils-1.8.3.jar	commons-beanutils-1.8.3.jar
2	commons-digester-1.8.1.jar	commons-digester-1.8.1.jar	commons-digester-2.1.jar

SpringFramework 2.5で使用するクラスパス

項番	V9.2.0、V9.2.1	V10.0.0、V10.1.0	V11.0.0
1	cglib-nodep2.1_3.jar	cglib-nodep2.2.2.jar	cglib-nodep2.2.2.jar

SpringFramework 3.Xで使用するクラスパス

jarファイル名、および格納パスが変更されました。

	V10.0.0、V10.1.0	V11.0.0
格納パス Windows32/64	[Interstageのインストールディレクトリ]¥BAS¥spring30¥lib¥	[Interstageのインストールディレクトリ]¥BAS¥spring31¥lib¥
格納パス Solaris Linux32/64	/opt/FJSVibs/spring30/lib/	/opt/FJSVibs/spring31/lib/
ファイル名	org.springframework.web.servlet-3.0.5.FUJITSU.jar org.springframework.web-3.0.5.FUJITSU.jar org.springframework.asm-3.0.5.FUJITSU.jar org.springframework.beans-3.0.5.FUJITSU.jar org.springframework.core-3.0.5.FUJITSU.jar org.springframework.context-3.0.5.FUJITSU.jar org.springframework.expression-3.0.5.FUJITSU.jar org.springframework.orm-3.0.5.FUJITSU.jar org.springframework.jdbc-3.0.5.FUJITSU.jar org.springframework.transaction-3.0.5.FUJITSU.jar org.springframework.aop-3.0.5.FUJITSU.jar org.springframework.aspects-3.0.5.FUJITSU.jar org.springframework.context.support-3.0.5.FUJITSU.jar org.springframework.jms-3.0.5.FUJITSU.jar org.springframework.oxm-3.0.5.FUJITSU.jar	org.springframework.web.servlet-3.1.1.FUJITSU.jar org.springframework.web-3.1.1.FUJITSU.jar org.springframework.asm-3.1.1.FUJITSU.jar org.springframework.beans-3.1.1.FUJITSU.jar org.springframework.core-3.1.1.FUJITSU.jar org.springframework.context-3.1.1.FUJITSU.jar org.springframework.expression-3.1.1.FUJITSU.jar org.springframework.orm-3.1.1.FUJITSU.jar org.springframework.jdbc-3.1.1.FUJITSU.jar org.springframework.transaction-3.1.1.FUJITSU.jar org.springframework.aop-3.1.1.FUJITSU.jar org.springframework.aspects-3.1.1.FUJITSU.jar org.springframework.context.support-3.1.1.FUJITSU.jar org.springframework.jms-3.1.1.FUJITSU.jar org.springframework.oxm-3.1.1.FUJITSU.jar

TERASOLUNAで使用するクラスパス

項番	V9.2.0、V9.2.1	V10.0.0、V10.1.0	V11.0.0
1	cglib-nodep2.1_3.jar	cglib-nodep2.2.2.jar	cglib-nodep2.2.2.jar
2	commons-beanutils-1.8.0.jar	commons-beanutils-1.8.3.jar	commons-beanutils-1.8.3.jar
3	commons-dbcp-1.2.2.jar	commons-dbcp-1.4.jar (JDK6) またはcommons-dbcp-1.3.jar (JDK5)	commons-dbcp-1.2.2.patch_DBCP264_DBCP372.jar
4	commons-digester-1.8.1.jar	commons-digester-1.8.1.jar	commons-digester-2.1.jar
5	commons-lang-2.4.jar	commons-lang-2.5.jar	commons-lang-2.6.jar
6	commons-pool-1.4.jar	commons-pool-1.5.4.jar	commons-pool-1.6.jar
7	velocity-1.6.2.jar	velocity-1.6.4.jar	velocity-1.6.4.jar

◆LIKE述語の構文解析時に出力されるJYPメッセージの内容変更

Interstage Business Application Server V9.1.0以降に同梱されるSymfoware/RDBまたはSymfoware Server V9.0以降のバージョン(注)では、LIKE述語における暗黙的な型変換機能の強化として、照合値に指定可能なデータ型の範囲が拡大されることにより、LIKE述語の照合値、パターン、エスケープ文字に指定できない値式のデータ型を指定した場合に出力される、JYP7165Eのメッセージ内容が異なります。

	バージョン・レベルが9.0.0以前	バージョン・レベルが9.1.0以降
JYP7165Eのメッセージ本文	LIKE述語の照合値、パターン、エスケープ文字のデータ型が文字列型、または各国語文字列型ではありません。	LIKE述語の照合値、パターン、エスケープ文字に指定した値式のデータ型に誤りがあります。

注)Symfoware Server V8.0をお使いの場合は、上記の内容変更はありません。

◆各国語文字列型への半角カタカナ格納のエラー通知

Interstage Business Application Server V9.1.0以降に同梱されるSymfoware/RDBまたはSymfoware Server V9.0以降のバージョン(注)では、rdbsloaderコマンドでテキスト形式の入力ファイルを指定した場合、データベースの文字コード系がEUCコードまたはShift_JISコードで、各国語文字列型の列に半角カタカナを格納しようとした場合に、従来正常終了していたものがエラー通知されるようになります。

	バージョン・レベルが9.0.0以前	バージョン・レベルが9.1.0以降
各国語文字列型への半角カタカナ格納のエラー通知	rdbsloaderコマンドでテキスト形式の入力ファイルを指定した場合、データベースの文字コード系がEUCコードまたはShift_JISコードであるにもかかわらず、各国語文字列型の列に半角カタカナを格納していました(P番号PG49764により修正されており、これを含む緊急修正を適用していない場合に該当します)。	rdbsloaderコマンドでテキスト形式の入力ファイルを指定した場合、データベースの文字コード系がEUCコードまたはShift_JISコードで、各国語文字列型の列に半角カタカナを格納しようとした場合にエラー通知します。

注)Symfoware Server V8.0をお使いの場合は、上記の内容変更はありません。

◆データベースリソース定義でOracle使用時の動作変更

V9.0以降では、データベースリソース定義において、Oracle使用時に“File System Service Provider”を使用した接続方法から、“File System Service Provider”を使用しない接続方法に変更となりました。

	バージョン・レベルが8.0.1以前	バージョン・レベルが9.0.0以降
対応接続方法	File System Service Providerを使用した接続方法	File System Service Providerを使用しない接続方法 (注)

注) File System Service Providerを使用しない接続方法を利用した場合、“.bindings”ファイルは作成されません。

apfwmkrscコマンドにより、バージョン・レベルが8.0.1以前のOracleのデータベースリソース定義入力ファイルを登録する場合、“データソース名”、“PROVIDER_URL”、および“.bindings”ファイル作成の有無は無効となります。
詳細は“Interstage Business Application Server リファレンス”の“apfwmkrsc”を参照してください。

◆rdbstopコマンドのmcオプションによるコマンドの強制停止

9.1.0からrdbstopコマンドのmcオプション指定でRDBコマンドを強制終了した場合、処理時間がデータベースの規模や扱うデータ量に依存する、以下のコマンドが処理中断するようになります。

- rdbsloader
- rdbfmt
- rdbunl
- rdbprdic

・ rdbgcdic

このとき、サーバプロセスでコマンド処理を実行中の場合、クライアントプロセスが停止したことを認識した旨のメッセージを、コンソールおよびRDB構成パラメタファイルのRDBREPORTで指定したメッセージログファイルに出力します。

```
qdg14185i： s*コマンドの処理の中断が指示されました 対象資源='t*' u*
```

	バージョン・レベルが9.0.0以前	バージョン・レベルが9.1.0以降
コマンド処理	コマンド処理中にrdbstopコマンドのmcオプションにより強制停止した場合、コマンドの処理が完結するまで動作し続けます。	コマンド処理中にrdbstopコマンドのmcオプションにより強制停止した場合、コマンドの処理を中断します。(注)

注) コマンドの実行結果に以下の変更があります。

・ コマンドの処理中断による対象資源のアクセス禁止状態の設定
以下のコマンドでは、処理の中断により対象の資源に対してアクセス禁止状態が設定される場合があります。その時、コンソールおよびRDBREPORTで指定したメッセージログファイルに、メッセージ“qdg03400u”または“qdg13217u”が出力されます。

　－ rdbsloader

　－ rdbfmt

・ 出力ファイルの途中状態
以下のコマンドでは、処理の中断により出力ファイルが出力途中の状態で残ります。

　－ rdbunl

◆アプリケーション安定稼動機能使用時の留意事項

V10.1以降では、以下の点について留意してください。

・ システム全体でクッキーは使用できません。

・ ログアウトやセションタイムアウト時のログイン画面への遷移時にクライアントにセションIDを返却できません。

◆電子フォームアプリケーション使用時のクライアント部品インストール Windows32 Solaris Linux32

V9.2以降では、クライアント部品の自動インストール機能がありません。このため、以下に留意してください。

・ クライアント印刷を行う場合、事前にクライアント部品(印刷機能)をクライアントコンピュータにインストールしておく必要があります。

・ 署名オプションを追加インストールし、署名送信を行う場合、事前にクライアント部品(署名共通)、およびクライアント部品(署名機能)をクライアントコンピュータにインストールしておく必要があります。

注意

クライアント部品は、接続するすべてのサーバの内で最新の製品に同梱された部品をインストールする必要があります。

■旧バージョン・レベルからの資源の移行方法

◆同期アプリケーション連携実行基盤の資源

定義ファイル(アプリケーション連携実行基盤定義ファイルなど)

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	○
9.0.0、9.1.0、9.2.0、9.2.1	○
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし －：定義が存在しない］

サーバアプリケーション(COBOL)

COBOLで作成したサーバアプリケーションのバイナリを移行する場合は、移行先で運用時に使用するNetCOBOL運用パッケージのバージョンにおいて、移行元でビルドの際に使用したNetCOBOL開発パッケージのバージョンがサポートされているかどうかを確認してください。再ビルドする場合は、本バージョンのソフトウェア条件に従ってください。

ソースレベルの互換性については以下の表のとおりとなります。

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	○
9.0.0、9.1.0、9.2.0、9.2.1	○
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし］

サーバアプリケーション(C言語)

C言語で作成したサーバアプリケーションを移行する場合は、移行元と移行先のOSにより対応が異なります。

プラットフォームに変更がない場合、ソースレベルの互換性については以下の表のとおりとなります。

プラットフォームに変更がない場合で、下記の表で○となっている場合でも、旧バージョン・レベルを実行していた環境のOSと、新バージョン・レベルを実行する環境のOSのバージョンが異なる場合は、再ビルドする必要があります。

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	Solaris ○
9.0.0、9.1.0、9.2.0、9.2.1	Solaris Linux32/64 ○ Windows32 × (注1) (注2)
10.0.0、10.1.0	Solaris Linux32/64 ○ Windows32 × (注1) (注2)

［○：互換あり △：一部互換なし ×：互換なし］

注1) 実行基盤インタフェースを再生成する必要があります。

注2) サーバアプリケーションのパラメタとして受け渡されるメモリ領域をアプリケーションで開放、もしくは再獲得する場合、V11以降で提供されるAPIを使用する必要があります。詳細は、“Interstage Business Application Server リファレンス”の“メモリ獲得・開放API”を参照してください。

プラットフォームに変更がある場合、移行元と移行先のOSにより対応が異なります。

- SolarisからLinux、LinuxからSolaris、またはLinuxから別プラットフォームのLinuxへ移行する場合、基本的にソース互換性があります。
- 32bit版Windowsから64bit版Windowsへ移行する場合、32bit版WindowsのV10以前のバージョン・レベルから32bit版Windowsの本バージョン・レベルに移行する場合と同様の注意点があります。
- SolarisからWindows、およびLinuxからWindowsへ移行する場合、32bit版WindowsのV10以前のバージョン・レベルから32bit版Windowsの本バージョン・レベルに移行する場合と同様の注意点があります。

・ WindowsからSolarisおよびWindowsからLinuxへ移行する場合、実行基盤インタフェースを再生成する必要があります。

プラットフォームに変更がある場合は、上記に加えアプリケーションにおいてポインタ演算など移植時に問題となるコーディングがないか一般的な注意点について確認してください。

クライアントアプリケーション(Java)

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	○
9.0.0、9.1.0、9.2.0、9.2.1	○
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし］

クライアントアプリケーション(C言語)

C言語で作成したクライアントアプリケーションを移行する場合、C言語で作成したサーバアプリケーションを移行する場合と同様の注意点があります。

◆非同期アプリケーション連携実行基盤の資源

定義ファイル(アプリケーション連携実行基盤定義ファイルなど)

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	○
9.0.0、9.1.0、9.2.0、9.2.1	○
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし －：定義が存在しない］

サーバアプリケーション

サーバアプリケーションを移行する場合は、旧バージョン・レベルを実行していた環境のOSと、新バージョン・レベルを実行する環境のOSのバージョンが異なる場合は、再ビルドする必要があります。

ソースレベルの互換性については以下の表のとおりとなります。

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	○
9.0.0、9.1.0、9.2.0、9.2.1	○
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし］

クライアントアプリケーション

旧バージョン・レベル	本バージョン・レベルでの使用可否
8.0.1	○
9.0.0、9.1.0、9.2.0、9.2.1	○
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし］

◆オープンJavaフレームワークの資源

定義ファイル(Bean定義ファイルなど)

旧バージョン・レベル	本バージョン・レベルでの使用可否
9.2.0、9.2.1	△（注）

旧バージョン・レベル	本バージョン・レベルでの使用可否
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし －：定義が存在しない］

注）IJServer連携機能を使用していて、Spring Framework 3.1へ移行する場合

アプリケーション

旧バージョン・レベル	本バージョン・レベルでの使用可否
9.2.0、9.2.1	△（注）
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし］

注）IJServer連携機能を使用していて、Spring Framework 3.1へ移行する場合

Interstage Business Application Server 9.2.0においてSpringFrameworkのIJServer連携機能を使用していた場合は、次のように修正する必要があります。

- SpringFramework 2.5のIJServer連携機能を使用する場合
  APサーバ側の業務アプリケーションは、10.0、10.1または11.0で提供されるapfw-spring-ejb-api.jarを使用してEARを再ビルドする必要があります。
- SpringFramework 3.1を使用する場合
  IJServer連携機能を使用せずに、JMSまたはHTTP invokerとして通信を行うようにBean定義ファイルの設定を変更する必要があります。

◆TERASOLUNAフレームワーク (J2EE)

定義ファイル(Bean定義ファイル)

旧バージョン・レベル	本バージョン・レベルでの使用可否
9.2.0、9.2.1	×
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし －：定義が存在しない］

アプリケーション

旧バージョン・レベル	本バージョン・レベルでの使用可否
9.2.0、9.2.1	×
10.0.0、10.1.0	△

［○：互換あり △：一部互換なし ×：互換なし］

詳細は、“オープンJavaフレームワーク ユーザーズガイド”の“TERASOLUNA (J2EE)” > “旧バージョン・レベルからの移行”を参照してください。

◆TERASOLUNAフレームワーク (Java EE)

定義ファイル(Bean定義ファイル)

旧バージョン・レベル	本バージョン・レベルでの使用可否
10.0.0、10.1.0	○

［○：互換あり △：一部互換なし ×：互換なし －：定義が存在しない］

アプリケーション

旧バージョン・レベル	本バージョン・レベルでの使用可否
10.0.0、10.1.0	△

［○：互換あり △：一部互換なし ×：互換なし］

詳細は、“オープンJavaフレームワーク ユーザーズガイド”の“TERASOLUNA (Java EE)” > “旧バージョン・レベルからの移行”を参照してください。

◆開発環境の資源

サーバパッケージと開発環境パッケージの組み合わせ

サーバパッケージと開発環境パッケージのバージョンの組み合わせは以下のようになります。

		サーバパッケージ						
		V8.0.1	V9.0.0	V9.1.0	V9.2.0	V10.0.0	V10.1.0	V11.0.0
開発環境パッケージ	V8.0.1	○	×	×	×	×	×	×
	V9.0.0	×	○	○	○	×	×	×
	V9.1.0	×	△（注1）	○	○	×	×	×
	V9.2.0	×	△（注1）	○	○	×	×	×
	V10.0.0	×	×	×	×	○	○	×
	V10.1.0	×	×	×	×	△（注2）	○	×
	V11.0.0	×	×	×	×	×	×	○

［○：使用可 △：一部使用不可 ×：使用不可］

注1）C言語で作成したクライアントアプリケーションは、V9.0.0 サーバパッケージでは使用できません。

注2）IJServerクラスタ向けにCOBOLプロジェクトから生成したWebサービスアプリケーションは、V10.0.0 サーバパッケージでは使用できません。

ポイント

COBOLサーバアプリケーションの開発にCOBOL開発支援ツールを使用します。COBOL開発支援ツールは、Interstage Stuido上で動作します。V9.0.0以前のCOBOLプロジェクトマネージャで管理していた資源を使用する場合は、Interstage StudioのCOBOLプロジェクトに移行する必要があります。移行手順の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“COBOLプロジェクトマネージャからの移行”を参照してください。

1.3 修正情報

Interstage Business Application Serverのプログラムの修正情報については、マニュアルCDの“Release”フォルダに格納されている“program.pdf”を参照してください。
Interstage Application Serverのプログラムの修正情報については、マニュアルCDの“Release”フォルダに格納されている“program_isaps.pdf”を参照してください。

第2章 Interstage Business Application Serverの概要

本章では、Interstage Business Application Serverの概要について説明します。

2.1 Interstage Business Application Serverとは

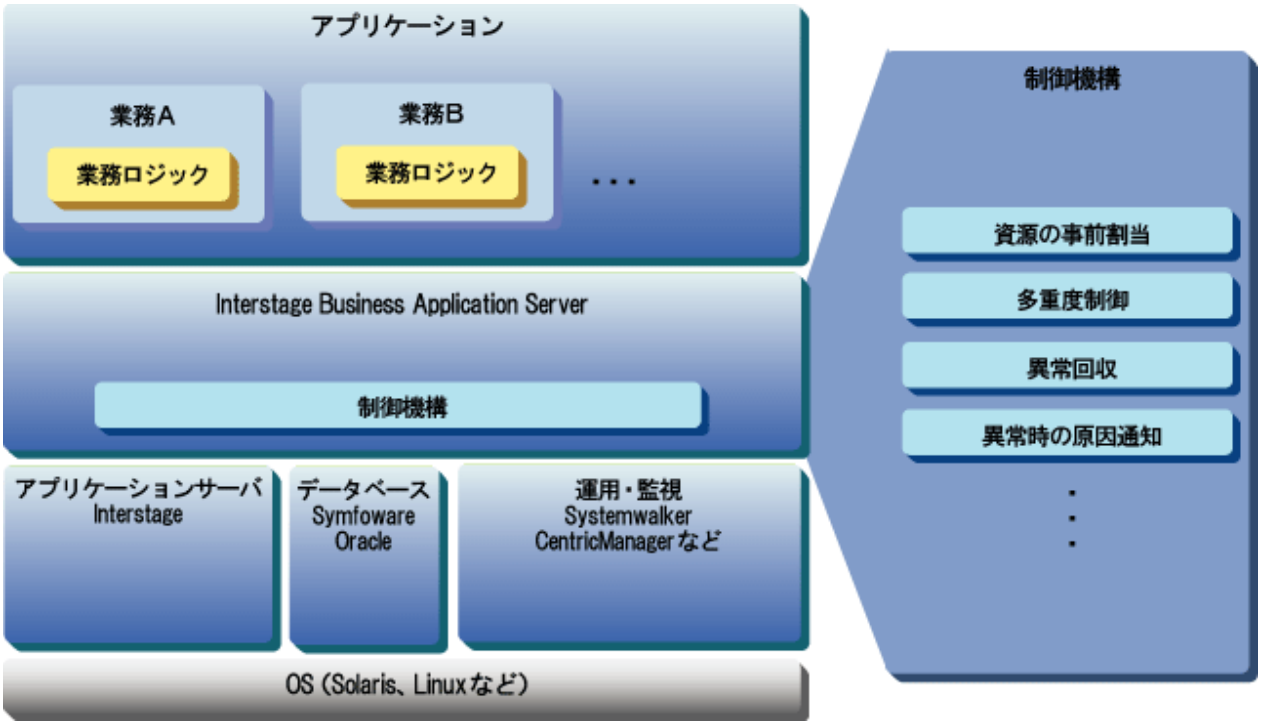
従来メインフレーム中心であった基幹システムもインターネット技術をはじめとするオープン技術の進展により周辺業務だけでなく、コア業務もオープン化、分散化が進んでいます。

また、基幹業務のオープン化の流れは、オンライン処理中心に始まり、それにともないデータベースがオープン化され、分散したシステムの連携、オンライン処理やデータベースに付随する一括処理のオープン化へと進展しています。このような環境において、情報システム部門では、システム構築および運用に関して、次に記す点が最重要課題となっています。

- システムの安定稼動
- システム開発の効率化および開発期間の短縮
- 柔軟なシステム拡張およびシステム構成の変更
- TCO削減

Interstage Business Application Serverは、オープンプラットフォームを活用した基幹系システム構築に必要な共通技術、基幹システムのアプリケーション構造の共通化と連携技術、および業務運用を支援する機能を提供することで、生産性および堅牢性の高い、柔軟性および拡張性に富むシステム構築と運用を支援するソフトウェア製品です。

Interstage Business Application Serverの位置づけを以下に示します。



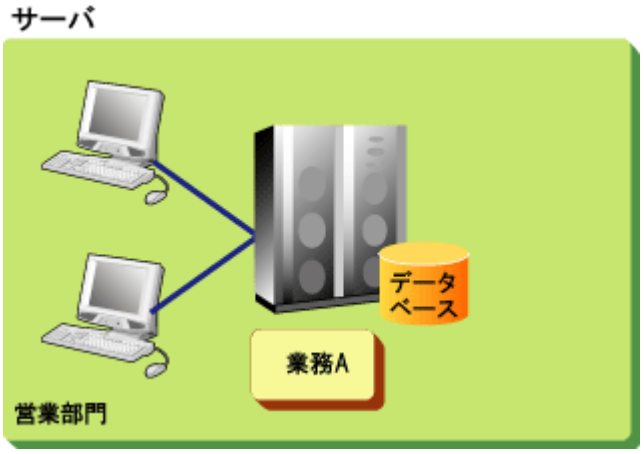
注意

- Interstage Business Application Serverには、アプリケーションサーバ機能としてInterstage Application Server Enterprise Editionが同梱されています。

本製品を適用することで以下に示すように業務の変更に応じて、システムを容易に拡張していくことができます。

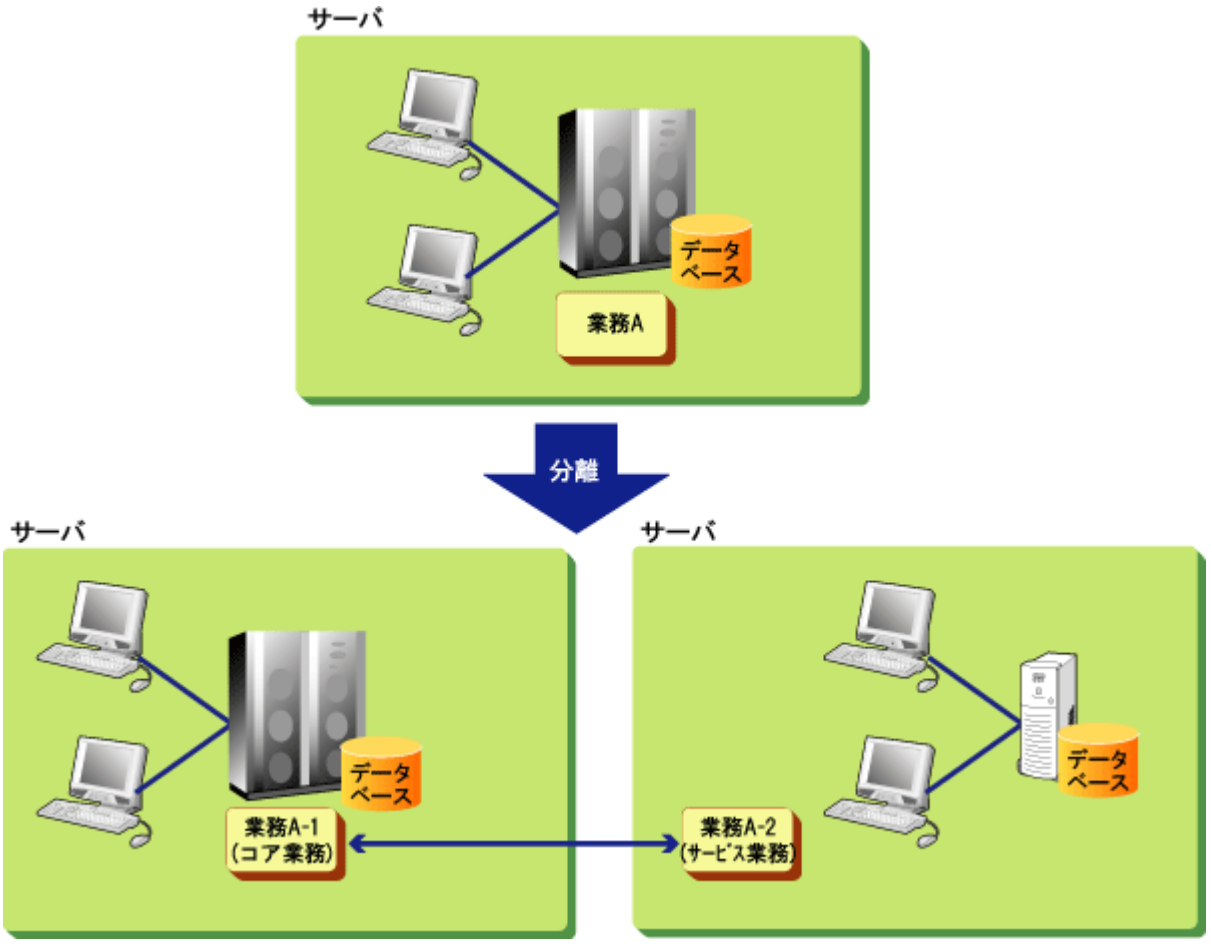
■(1) 1つのシステム内やサービスで稼働する業務

1つのシステムやサービスで業務を構成するケースです。新規に独立した業務を開発する時や部門ごとなどに個別の業務を運用する場合、この形態となります。



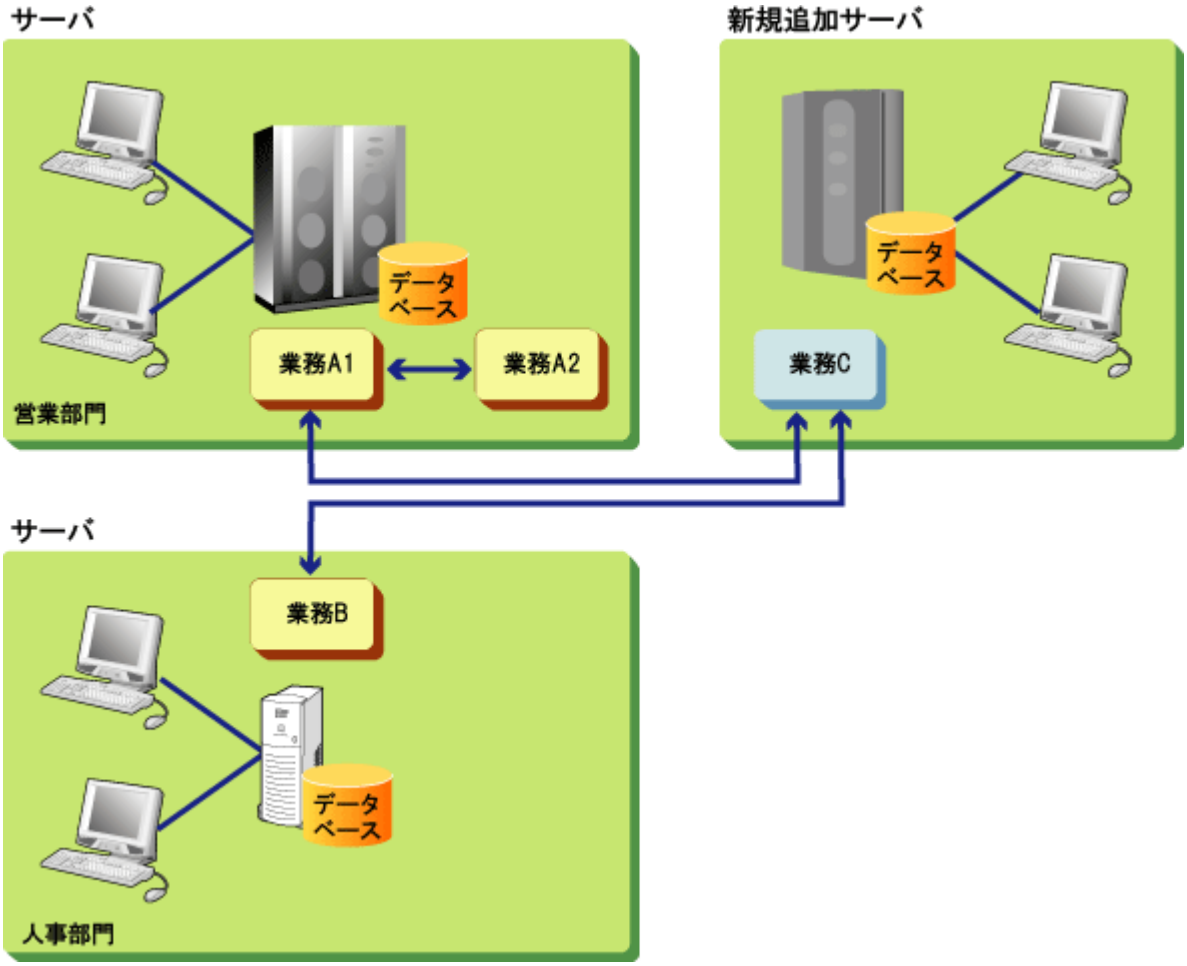
■(2) 1つのシステムを分離し、それらを連携

もともと1つになっていたシステム内の業務をライフサイクルの違いで複数のシステムに分離し、分離した複数システム間を連携するケースです。この状況が生じる原因は、タイムリーな顧客ニーズへの対応や、ライフサイクルの短い業務の頻繁な入替えによる他業務への影響の局所化などが考えられます。なお、ライフサイクルが短い業務とは、顧客の要件や外部要因などで頻繁に変更のある業務であり、ライフサイクルが長い業務とは、顧客管理など変更がほとんどない業務を意味します。



■(3) 分散されたシステムを整理後、それらを連携

部門ごとに構築されているシステムを統廃合した後、それらのシステムを連携するケースです。このアプローチの目的は、ITシステムのTCO削減を行うことにあります。



2.2 特徴

Interstage Business Application Serverは、以下の特徴を持っています。

- 安全なシステムを短期に開発

  アプリケーションは、制御ロジックと業務ロジックに大別できます。制御ロジックとは、アプリケーションの実行制御、トランザクションの管理、データベースのアクセス制御など処理形態に依存しない共通制御、およびオンライン通信におけるクライアントとサーバのプログラムで異なる言語を利用している場合のデータ変換処理や非同期通信におけるキュー制御など処理形態ごとの固有制御に分類され、アプリケーション間の連携や資源管理を行うためのプログラムを意味します。また、業務ロジックとは、顧客情報や注文内容をデータベースに反映するなど業務に特化した制御を行うプログラムを意味します。
  アプリケーション連携実行基盤では、業務アプリケーションを効率よく開発するために業務ロジックの開発作法を規定しています。また、制御ロジックと業務ロジックを分離して開発し、複数アプリケーションを連携するための機能やメッセージの整合性を保証するための制御ロジックを実行環境として提供することで、アプリケーションの生産性を向上し、信頼性のあるシステムを短期に開発することができます。

- 柔軟性、拡張性を持ったシステムの開発

  企業を取り巻く市場や需要の急激な変化や多様な顧客ニーズに即応するためには、柔軟性および拡張性の高いシステムを構築する必要があります。本製品では、制御ロジックと分離した業務ロジックの構造を規定し、プラットフォーム、ミドルウェアおよびオンライン形態(同期処理/ディレイド処理)に依存しない業務部品として汎用性および再利用性を高め、また、異なる開発言語(COBOL、C言語またはJava)でも、同じ設計方法で業務ロジックの開発を可能としています。
  また、オンラインの同期処理に対しては、クライアントとサーバのアプリケーション間の文字コード変換やデータ型変換機能を提供し、異なる言語でのクライアントとサーバのアプリケーションの開発を容易にしています。

ディレイド処理に対しては、業務アプリケーションの呼び出し順序、受け渡すメッセージ、および異常が発生した場合のシーケンスを定義するフロー定義ツールを提供し、業務ロジックの柔軟な組み合わせにより新サービスを簡単に構築することを可能としています。

- システム運用のコスト低減
  オープンシステムは、Webサーバ、アプリケーションサーバおよびデータベースサーバの機能分担が進み、また、これらの機能をサーバに分散してシステムの構築が行われています。これにより、業務量拡大への柔軟な対応や信頼性向上のメリットがある一方で、複数サーバや業務管理の煩雑さから管理者の運用コストが増大しています。
  本製品では、アプリケーションの呼び出しから応答までの時間、パラメタおよびエラー内容などの稼働状況を利用者の識別情報を自動的に付加して各サーバ単位に出力する標準ログ機能を提供することにより、出力したログを、Systemwalker Service Quality Coordinatorを含めた各種のツールを使って容易に分析が可能になりました。また、課金や監査のために、アプリケーションの処理状況(ユーザログ)を記録するためのAPIを提供し、ユーザログを簡単、かつ確実に取得することが可能としています。これらの機能により、業務の可視化や性能のボトルネックの早期検出が可能となり、性能チューニングやキャパシティプランニングなどの運用コストを削減できます。

2.3 主な機能

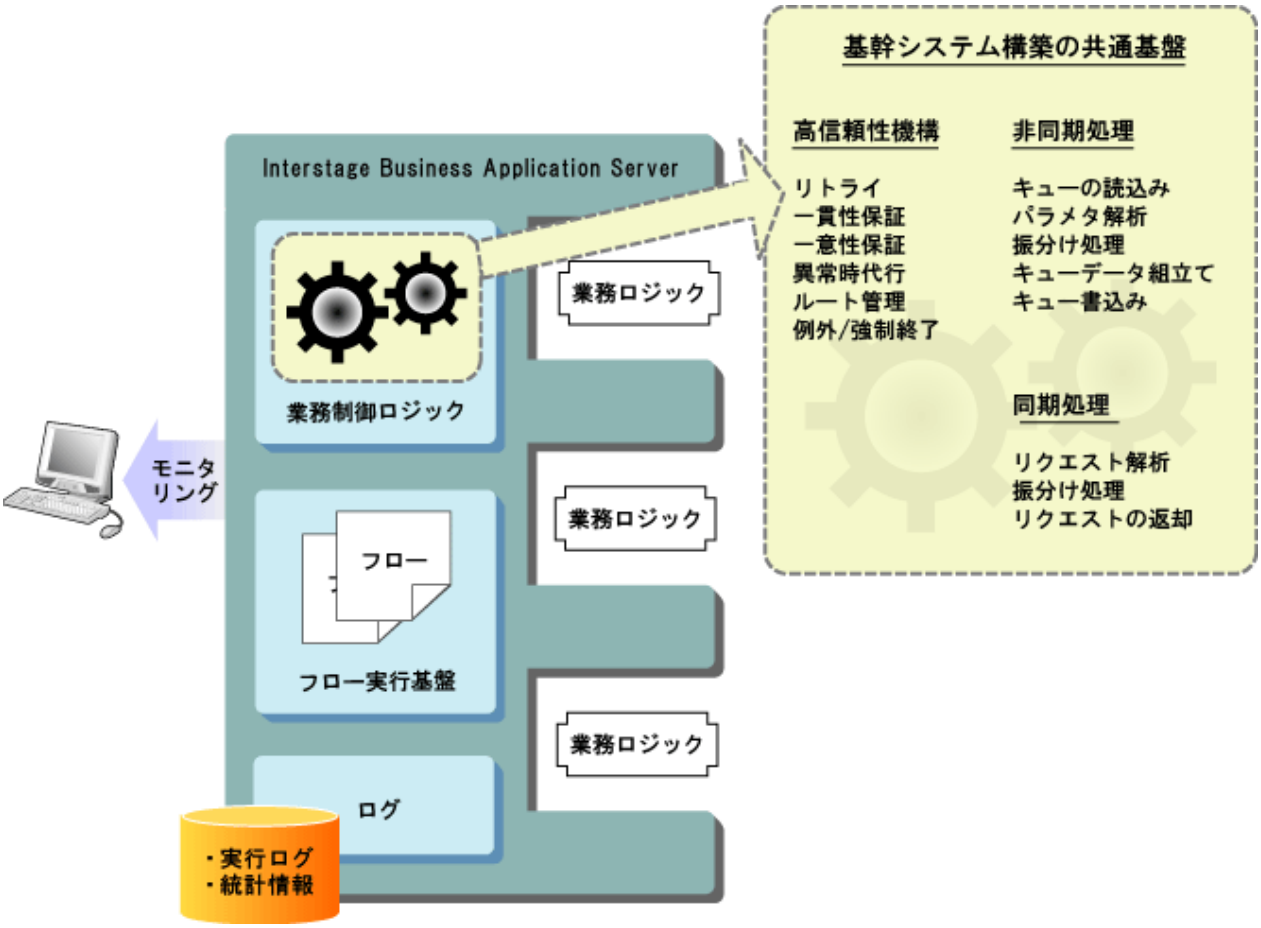
Interstage Business Application Serverでは、オンライン業務(同期型、ディレイド型)を構築するための機能および、オンデマンド型のオンライン・バッチ業務や他システムとの連携を容易にするための機能を提供しています。以下に代表的な機能を説明します。

■アプリケーションフレームワーク

Interstage Business Application Serverでは、COBOLをはじめ各種言語のアプリケーションの構築を支援する“アプリケーション連携実行基盤”、Java(TM) 2 Platform, Enterprise Edition（J2EE）に従ったアプリケーションの構築を支援する“Apcoordinator”、およびSpring Frameworkをはじめとする“オープンJavaフレームワーク”の3つのアプリケーションフレームワークを提供しています。

◆アプリケーション連携実行基盤

高い信頼性および高い品質が要求されるCOBOL、C言語およびJavaの制御ロジックを、アプリケーションの通信形態ごとに、“同期アプリケーション連携実行基盤”および“非同期アプリケーション連携実行基盤”として提供し、これらの総称を“アプリケーション連携実行基盤”と呼びます。
本機能の詳細は、“第2部 同期アプリケーション連携実行基盤編”および“第3部 非同期アプリケーション連携実行基盤編”を参照してください。



◆Apcoordinator

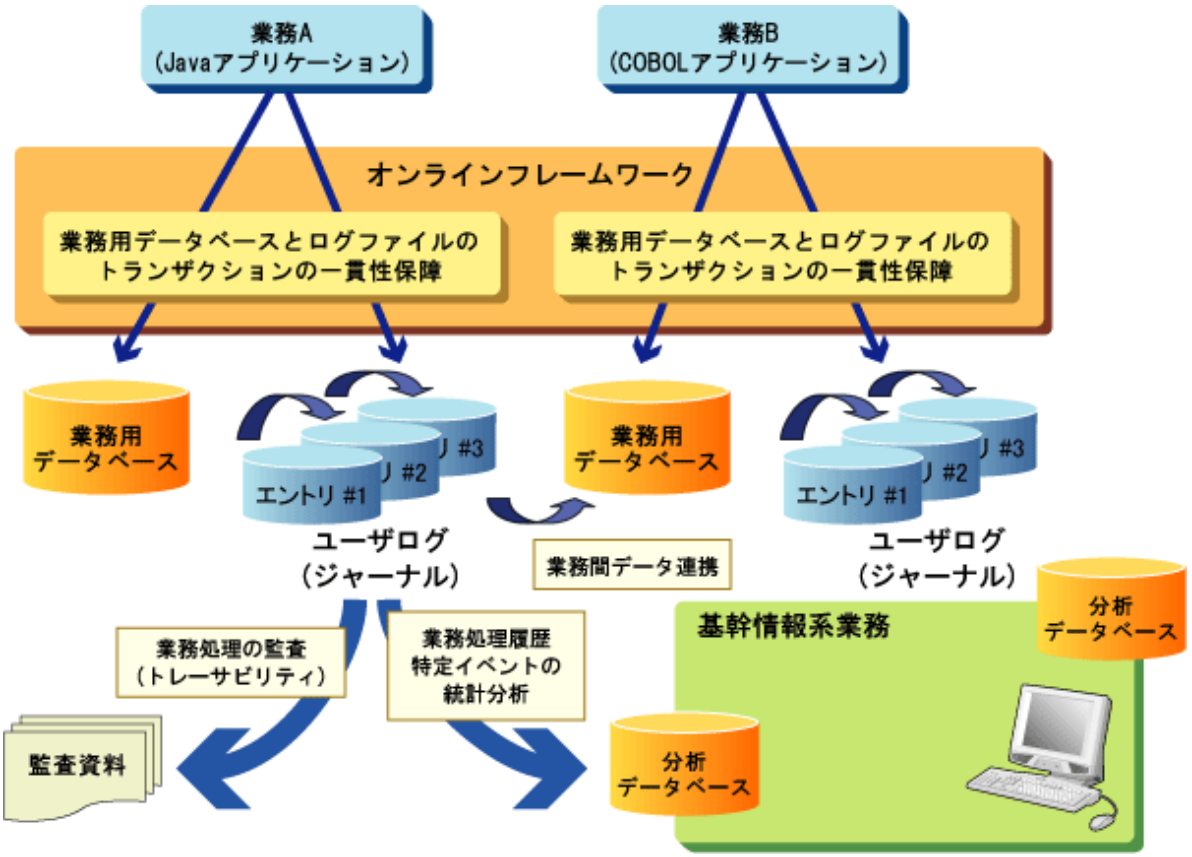
Java(TM) 2 Platform, Enterprise Edition（J2EE）に特化したアプリケーションの構築を支援する機能をApcoordinatorと呼びます。本機能を利用することで、Webアプリケーション、EJB（Enterprise JavaBeans）および電子フォームアプリケーションの開発を行うことができます。
本機能の詳細は、“Apcoordinator ユーザーズガイド”を参照してください。

◆オープンJavaフレームワーク

オープンソースのフレームワークとして人気のあるStruts、Spring Framework、iBATIS、そしてそれらを統合した汎用フレームワークであるTERASOLUNAを提供します。これらの総称を“オープンJavaフレームワーク”と呼びます。
本機能の詳細は、“オープンJavaフレームワーク ユーザーズガイド”を参照してください。

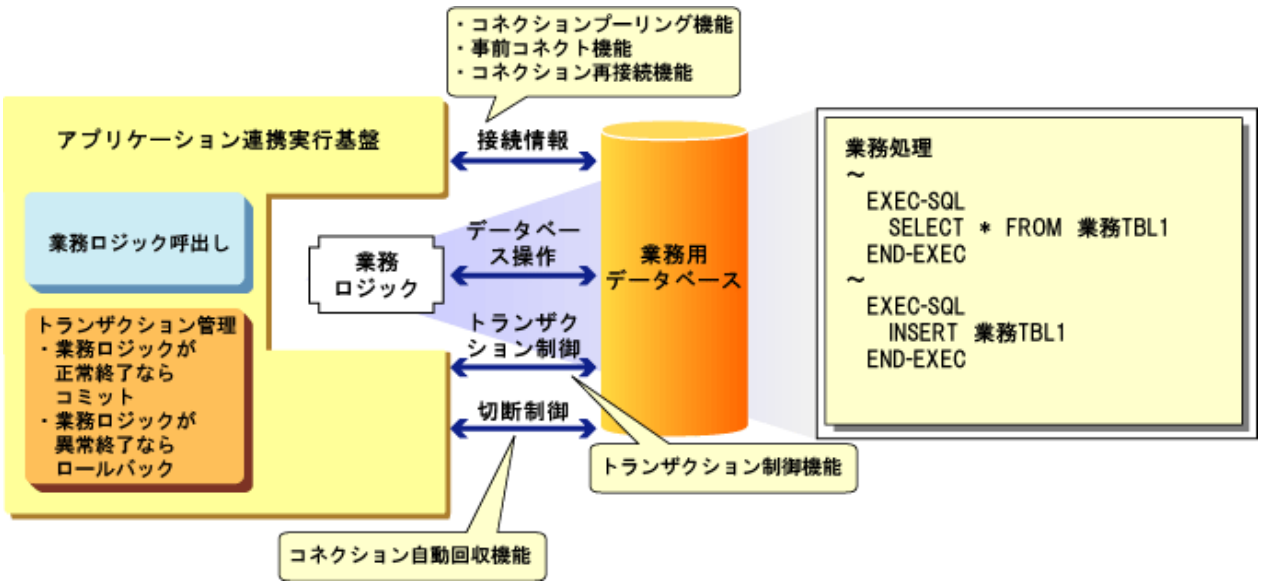
■ログ機能

ログ機能には、アプリケーションの呼び出しから応答までの時間、パラメタおよびエラー内容などの稼働状況を利用者の識別情報を自動的に付加して各サーバ単位に出力する“標準ログ”および課金や監査のために、アプリケーションの処理状況を記録するための“ユーザログ”があります。
ユーザログには、性能測定やデバッグ情報に使用する“汎用ログ”、業務処理の履歴をユーザのトランザクションと連動して取得可能な“高信頼性ログ”の2つが存在し、用途により使い分けができます。
また、出力したログは、表計算ツールなどにより容易に分析が可能です。本機能の詳細は、“第4部 共通機能編”の“第5章 ログ機能”を参照してください。



■データベースアクセス管理機能

Javaアプリケーションが、JDBCやEJBを利用してデータベースのアクセスを簡易化し、生産性を向上できるのと同様に、データベースを利用したCOBOLおよびC言語のアプリケーション向けに、データベースコネクションの事前接続やコネクションが切断された場合に再接続する“データベースアクセス管理機能”を提供します。
本機能を利用することで、データベースへのアクセスの高速化やコネクション異常時の処理が簡易化できることに加え、業務ロジックの処理結果(終了コード)により、データベースに対してコミットやロールバックが自動的に行われるため、アプリケーションで、開始や終了などのトランザクションに対する制御が不要になります。
本機能の詳細は、“第4部 共通機能編”の“第6章 データベースアクセス管理機能”を参照してください。
なお、Javaアプリケーションで、JDBCやEJBを利用してデータベースアクセスを簡易化する場合の設定は、IJServerの環境設定で行います。IJServerの詳細については、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。

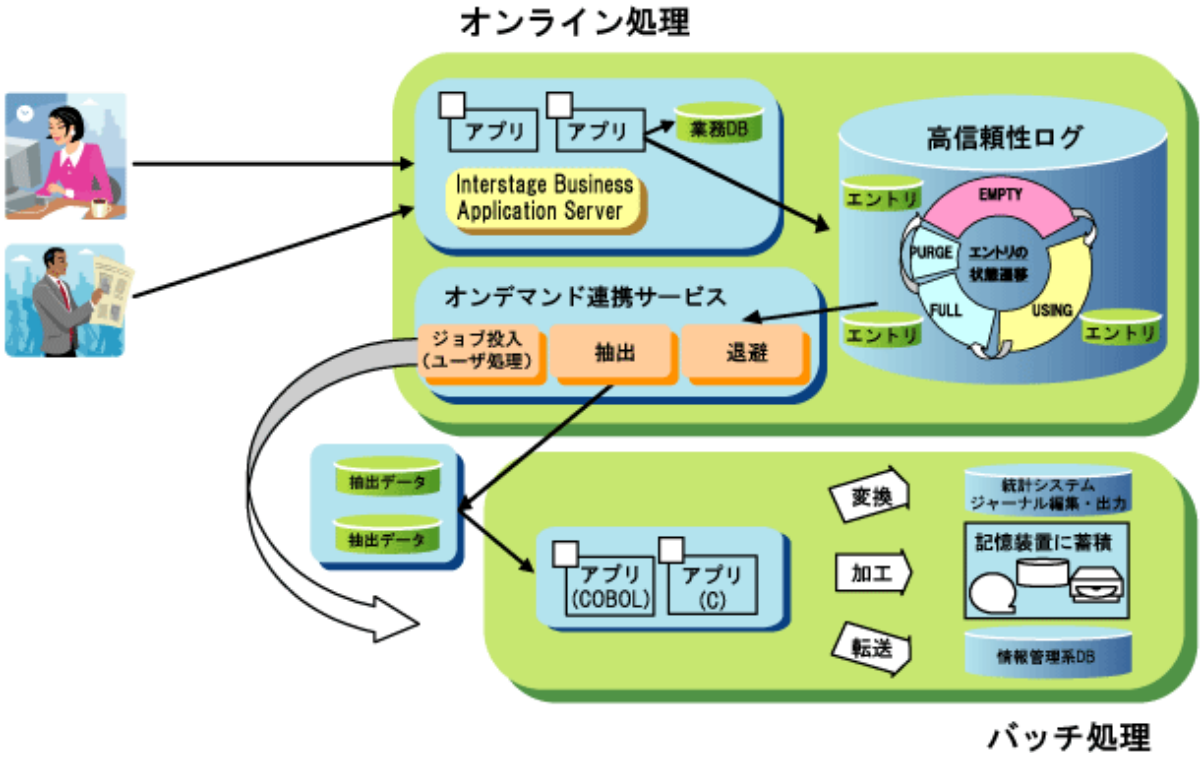


■アプリケーション安定稼働機能

業務アプリケーションが無応答になった場合や、Javaヒープ不足・ガーベジコレクション多発によるレスポンス遅延が発生した場合に正常実行中のアプリケーションの終了を待ち合わせ、新規リクエストを別の処理可能なJavaVMに振り分けることで安全かつ安定した業務継続を実現します。

■エクスポートユーティリティ

オンライン処理から高信頼性ログに蓄積されたログデータの退避処理、退避したログデータから必要なデータの抽出処理、およびバッチ投入などのユーザ処理実行等制御する機能をオンデマンド連携サービスの“エクスポートユーティリティ”として提供します。
本ユーティリティを利用することで、高信頼性ログに蓄積されたログデータを分析処理やバッチ処理に活用することが可能となります。
本機能の詳細は、“第5部 ユーティリティ編”の“第7章 エクスポートユーティリティ”を参照してください。



ポイント

関連製品として、バッチ業務を行うための“Interstage Job Workload Server”を提供しています。詳細は、“Interstage Job Workload Server 解説書”を参照してください。

2.4 製品構成

Interstage Business Application Serverは、InterstageのFoundation体系に属するソフトウェアです。

本製品は、以下に示す2つのエディションを提供しています。利用者の業務形態に合わせ、適切なエディションを利用してください。

- Standard Edition
  Servlet、EJBおよびCOBOLを利用して1つの業務やWebサービス内のアプリケーションを同期呼び出し連携するオンライン業務を構築することができます。J2EEに特化したアプリケーションの構築を支援するApcoordinator、同期アプリケーション連携実行基盤およびログ機能を使用することができます。

- Enterprise Edition
  Standard Editionの機能に加え、オンライン業務として利用されるディレイド通信(突き放し型)をはじめ、複数のアプリケーションを処理フローに従って連携する業務を構築することができます。アプリケーションの言語には、COBOL、C言語およびJavaを利用することができます。

業務システム		エディション	
利用形態	利用できる言語	Standard Edition	Enterprise Edition
同期アプリケーション連携実行基盤	COBOL	○	○
	C言語	○	○
非同期アプリケーション連携実行基盤	COBOL	×	○
	Java	×	○
Apcoordinator	Java	○	○
オープンJavaフレームワーク	Java	○	○

本製品には、アプリケーションサーバとしてInterstage Application Server Enterprise Editionを同梱しており、Interstage Application Serverが提供する豊富なアプリケーション形態と、短時間でセットアップできる高信頼・高性能のアプリケーション実行環境を併用できます。

また、高信頼性ログのユーザログ(ジャーナル)および本製品のフロー定義や運用情報を格納するためのデータベースとしてSymfoware/RDBを同梱していますが、利用者が業務のデータを格納する場合は、別途データベース製品(Symfoware ServerまたはOracle)を購入してください。

その他、関連製品として、バッチ業務を行うための“Interstage Job Workload Server”を提供しています。詳細については、“Interstage Job Workload Server 解説書”を参照してください。

2.5 システム構成

Interstage Business Application Serverのシステム構成について説明します。

2.5.1 業務モデル

本製品がサポートする業務モデルには、以下の3つがあります。

- 同期型のオンライン業務
- ディレイド型のオンライン業務
- オンデマンド型のオンライン・バッチ連携業務

それぞれの業務モデルの選択ポイントを説明します。

業務モデル	選択のポイント
同期型のオンライン業務	1つの業務やWebサービス内のアプリケーションを一問一答で同期的に呼び出し、処理結果をその場で受け取るオンライン業務に使用します。形式によって、同期アプリケーション連携実行基盤またはApcoordinatorを選択します。 同期アプリケーション連携実行基盤を使用する場合 ブラウザの画面から発行したリクエストをServletやEJBのアプリケーションで受け付け、業務処理を行うCOBOLやC言語のアプリケーションの処理結果を待ち合わせて呼出元に復帰します。 このような業務モデルを選択する場合、同期アプリケーション連携実行基盤を使用します。 Apcoordinatorを使用する場合 ブラウザの画面から発行したリクエストの受け付けと業務処理を行うサーバアプリケーションの両方をJ2EEに特化して作成する場合は、Apcoordinatorを使用します。詳細は、“Apcoordinator ユーザーズガイド”を参照してください。
ディレイド型のオンライン業務 (業務フローを使用した複数アプリケーションの連携)	アプリケーションの処理要求を突き放して実行し、業務処理自体は非同期に実行して後から処理結果を受け取る場合や、サーバ上の複数のアプリケーションを業務フローに従って連鎖的に実行する業務に使用します。この形態の場合は、非同期アプリケーション連携実行基盤を選択します。 非同期アプリケーション連携実行基盤 処理要求に沿った業務フローを開始するアプリケーションから、複数のアプリケーション(注1)を連携して業務処理を行い、処理結果を受け取ります。 なお、処理結果を受け取らず処理を突き放して実行することもできます。

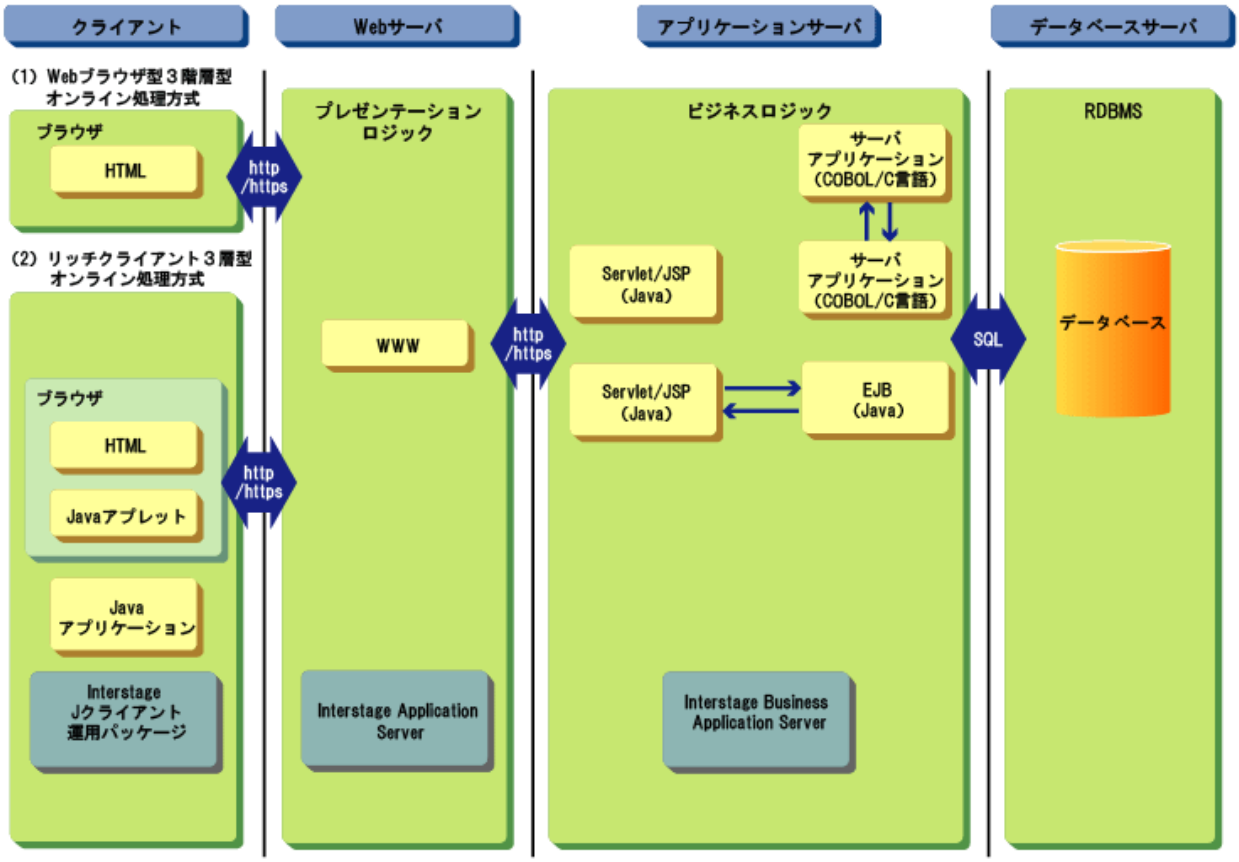
業務モデル	選択のポイント
オンデマンド型のオンライン・バッチ連携業務	オンライン業務を停めずに、オンライン処理の取引履歴や実行履歴を元にタイムリーに行うバッチ業務に使用します。

注1) アプリケーションには、COBOLおよびJavaが使用できます。C言語は使用できません。

2.5.1.1 同期型のオンライン業務

オンライン業務は、Webがベースで構築されることが多く、その構成は、画面表示を行うフロント部分をJSPやServletで作成し、実際の業務処理を行うアプリケーションと画面の入出力をコントロールするEJB(Servlet)に役割を分離するMVCモデルが主流です。
本製品を使用した同期型のオンライン業務のシステムモデルを以下に示します。

[同期型のオンライン業務]



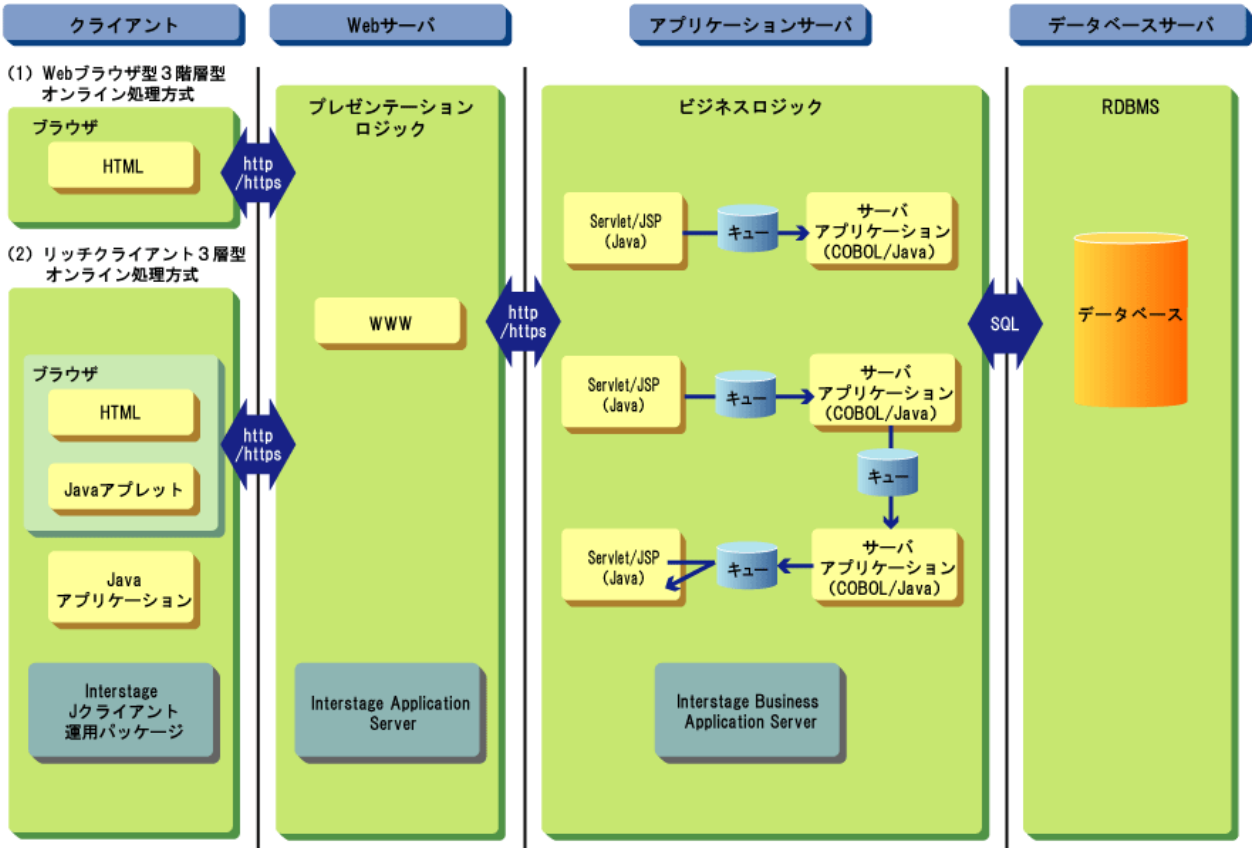
クライアントおよびサーバの両方のアプリケーションをJ2EEに特化して作成する場合は、Apcoordinatorを使用します。詳細は、“Apcoordinator ユーザーズガイド”を参照してください。

2.5.1.2 ディレイド型のオンライン業務(業務フローを使用した複数アプリケーションの連携) EE

ディレイド型のオンライン業務や業務フローを使用した複数アプリケーションの連携を行う場合は、同期型のオンライン業務モデルと基本的な構成は同一です。しかし、同期型のオンライン業務モデルが、特定の業務処理を要求応答形式で実行するのに対し、サーバで複数の業務を実行するアプリケーションを突き放して実行したり、フローに定義した順番で逐次呼び出し、最終的な結果を業務処理の応答として受信する点が異なります。

本製品を使用したディレイド型のオンライン業務(業務フローを使用した複数アプリケーションの連携)のシステムモデルを以下に示します。

[ディレイド型のオンライン業務(業務フローを使用した複数アプリケーションの連携)]

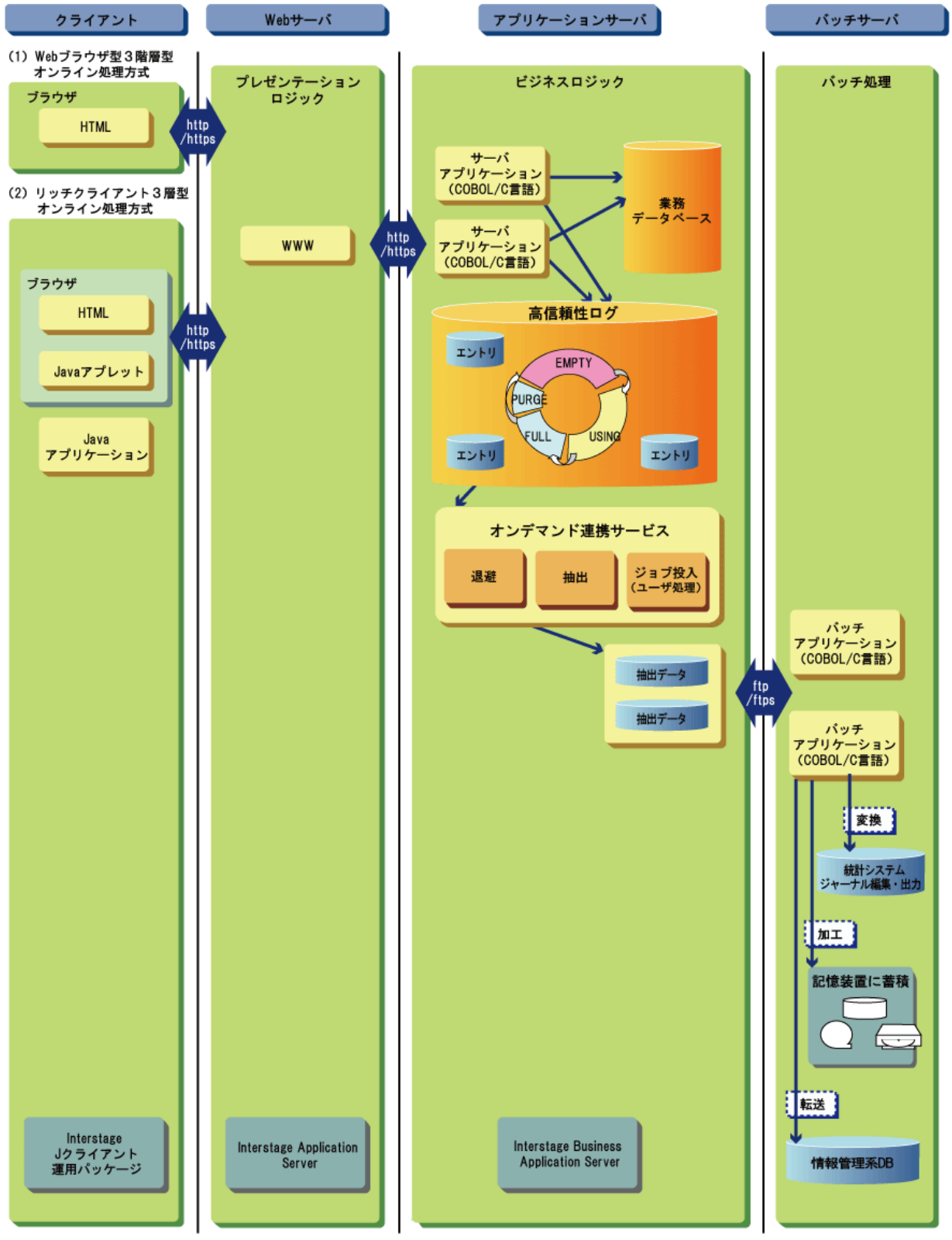


2.5.1.3 オンデマンド型のオンライン・バッチ連携業務 EE

高信頼性ログに蓄積されたユーザログを元に、バッチ処理など他システムとの連携を容易にする機能をオンデマンド連携サービスとして提供します。
オンデマンド連携サービスでは、本製品の提供するエクスポートユーティリティを使用することによりオンライン処理から高信頼性ログへ書き込んだログデータをファイルとして出力して、分析処理やバッチ処理で利用することが可能となります。
オンデマンド連携サービスでは、バイナリデータを扱うことができます。

本製品を使用したオンデマンド型のオンライン・バッチ連携業務のシステムモデルを以下に示します。

[オンデマンド型のオンライン・バッチ連携業務]



注意

全てのプラットホームで文字コードおよびデータ型を統一する必要があります。

ポイント

関連製品に、バッチ業務を行うための“Interstage Job Workload Server”を提供しています。詳細は、“Interstage Job Workload Server 解説書”を参照してください。

2.5.2 システム構成のパターン

業務モデルを決定後、アプリケーションサーバやデータベースサーバの配置など実際のシステム構成を決定します。

本製品で構築できるシステム構成は、最小構成である“シングルサーバ構成”および信頼性や性能を向上した構成である“Web/AP分離構成”および“複数サーバ構成”があります。各構成の関係は、次の図に示す通りです。各構成の詳細については、“セットアップガイド”の“概要編”で説明します。なお、利用可能なシステム構成は、業務モデルに依存せず同一となります。

シングルサーバ構成

本製品を使用する場合の最小のシステム構成です。Webサーバ、アプリケーションサーバおよびデータベースサーバを同一サーバ上に配置します。

Web/AP分離構成

シングルサーバ構成で運用していた業務のWebサーバ機能を別サーバで運用し、スケールアウトすることで機能分散を行う場合のシステム構成です。更にDB層を分離して3階層で運用することも可能です。

複数サーバ構成

シングルサーバ構成やWeb/AP分離構成で運用していた複数の業務を業務単位に別サーバで運用し、スケールアウトすることで負荷分散/危険分散を行う場合のシステム構成です。本構成では、サーバ内の複数の業務間の連携を行うことはできますが、別サーバで動作する業務間の連携を行うことはできません。

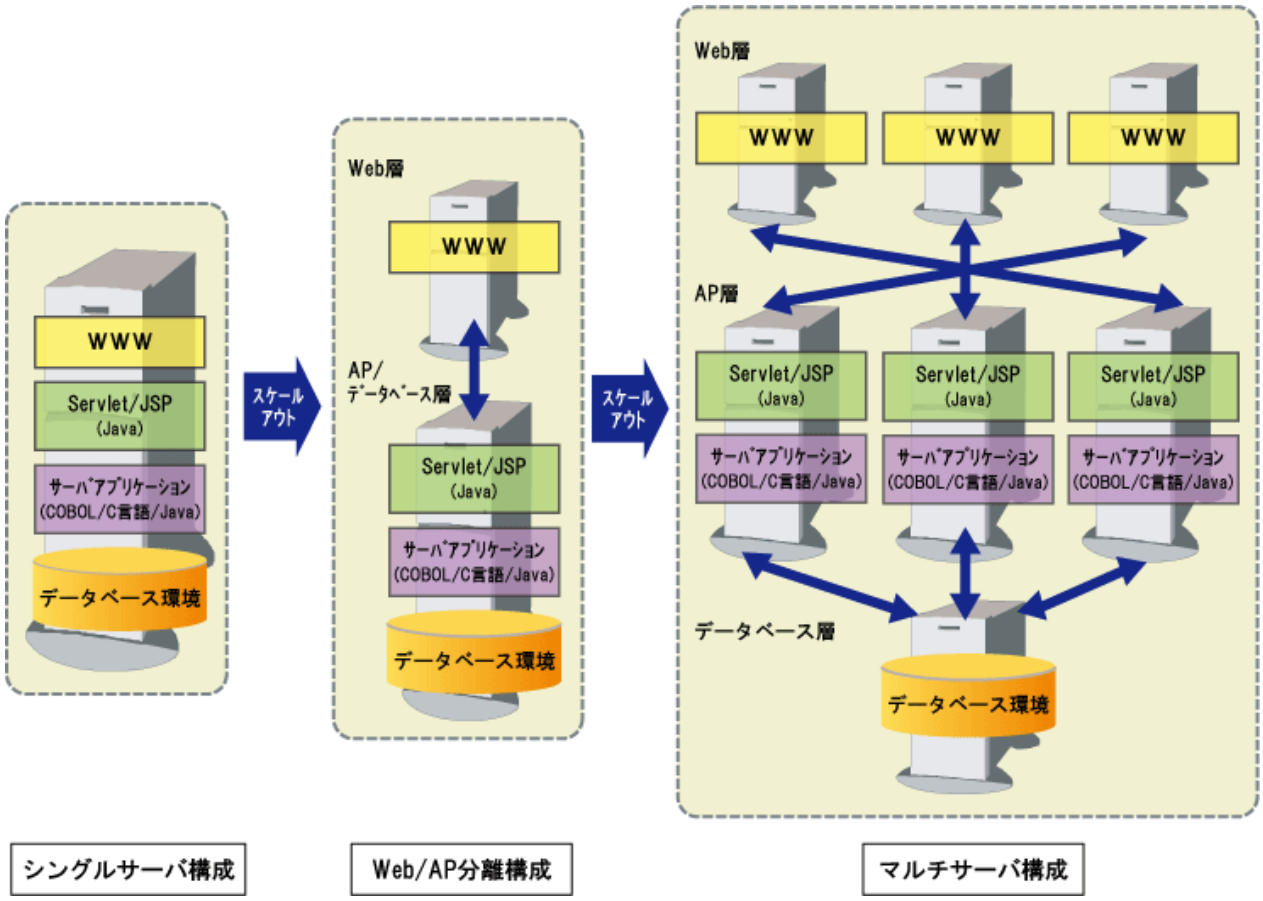
また、上記構成の信頼性を向上するための構成として以下の2つのシステム構成を併用することができます。

クラスタ構成

運用ノードと待機ノードの2つで1つのサーバを構成します。システム構成を冗長化することで異常が発生した場合でも待機ノードに処理を引き継いで業務を継続することが可能です。

負荷分散

Webサーバを負荷分散する構成です。多端末からの処理を複数サーバで並列に実行することで処理能力を向上することが可能です。

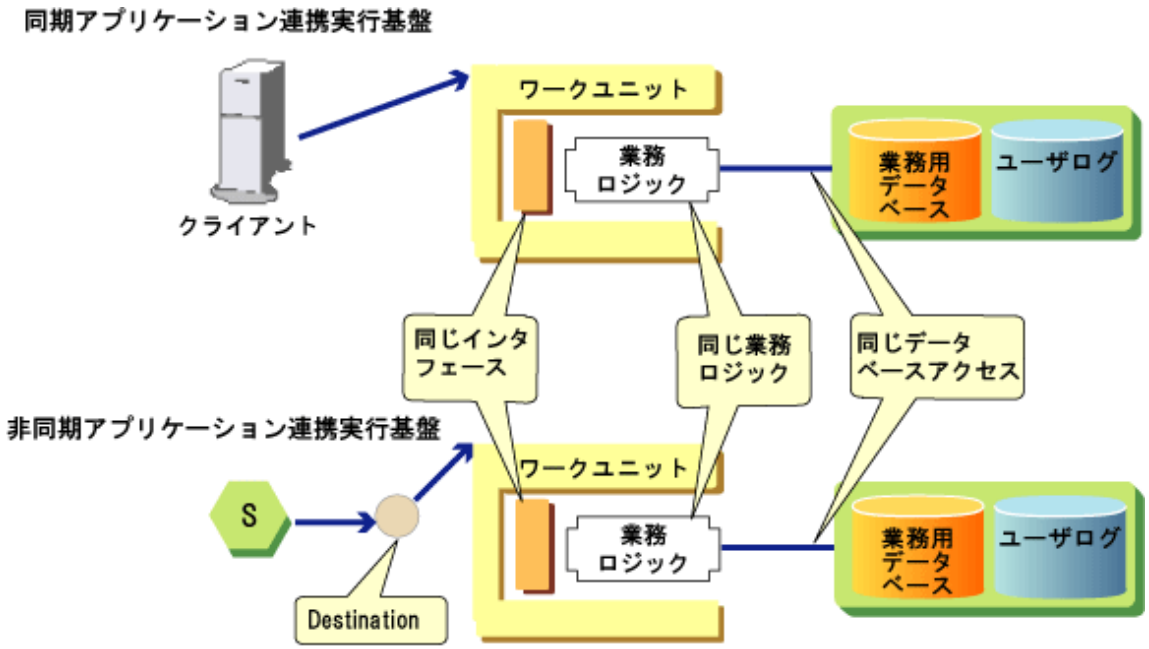


注意

- Web/AP分離構成や複数サーバ構成のシステム形態で業務を構築する場合、必ずシングルサーバ構成のシステム形態(適用条件、インストールする製品、および配置するコンポーネント)を確認したうえで作業を行ってください。
- 各システム構成におけるデータベースに関する推奨形態は以下のとおりです。
  - 業務用データベースで使用するデータベース製品
    Symfoware ServerまたはOracle
  - フロー定義DBおよびメッセージトラッキングDBで使用するデータベース製品(非同期アプリケーション連携実行基盤を使用する場合)
    業務用データベースで使用するデータベース製品と同じデータベース製品
  - 高信頼性ログの配置先データベースサーバ
    アプリケーション連携実行基盤で使用するデータベースサーバ
- C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

2.6 アプリケーション連携実行基盤

業務アプリケーションは、アプリケーション連携実行基盤上で動作します。アプリケーション連携実行基盤には、同期アプリケーション連携実行基盤および非同期アプリケーション連携実行基盤の2つがあり、それぞれについて、アプリケーション構成を説明します。
本製品では、サーバで動作する業務アプリケーションは、同期、非同期といった業務形態に依存せず共通に開発することができます。結果、業務形態が変更になった場合もアプリケーションを変更することなく、アプリケーションの再利用が可能であり、システム変更を柔軟に短期間で行うことができます。



アプリケーション連携実行基盤のクライアントまたはサーバのアプリケーションで利用できる言語は、下表の通りです。

種別	クライアントアプリケーションで利用できる言語	サーバアプリケーションで利用できる言語
同期アプリケーション連携実行基盤	Java	COBOL
		C言語
	C言語	COBOL
		C言語
非同期アプリケーション連携実行基盤	Java	COBOL
		Java

注意

同期アプリケーション連携実行基盤と非同期アプリケーション連携実行基盤とでは、アプリケーションで利用できるデータの型および復帰コードに対する動作が一部異なります。詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の以下を参照してください。

- “サーバアプリケーションの開発(COBOL)”の“業務処理の作成”

2.6.1 同期アプリケーション連携実行基盤

同期アプリケーション連携実行基盤では、オンライン処理(即時応答型)におけるクライアントからの処理要求に対して、リアルタイムに処理結果を応答する処理形態の基幹系業務に必要な共通技術を提供します。オープンシステムにおいて、メインフレームレベルの業務運用性、堅牢性を実現し、短期間で、高い信頼性のシステムを実現します。また、既存資産や既存ノウハウを最大限に活かし、業務ロジックの開発に専念することが可能です。
以下に同期アプリケーション連携実行基盤のアプリケーションについて説明します。

同期アプリケーション連携実行基盤の詳細は、“第2部 同期アプリケーション連携実行基盤編”で説明します。

2.6.1.1 アプリケーションの種類

同期アプリケーション連携実行基盤では、クライアントアプリケーションおよびサーバアプリケーションを使用して業務を構築します。

アプリケーションの種類	アプリケーションの概要	利用可能な言語(アプリケーションの形態)
クライアントアプリケーション	同期アプリケーション連携実行基盤のクライアントAPIを実行してサーバアプリケーションにリクエストを発行するアプリケーションです。	Java (Servlet/EJB) または C言語 (共有ライブラリまたは実行ファイル)
サーバアプリケーション	クライアントアプリケーションの入力情報に応じて業務用のデータベースの更新などの業務処理を行うアプリケーションです。	COBOLまたはC言語 (共有ライブラリ)
Webサービスアプリケーション	Webサービスクライアントから受け取ったデータを同期アプリケーション連携実行基盤にリクエストを発効、および同期アプリケーション連携実行基盤からのレスポンスをWebサービスクライアントに返却するアプリケーションです。	Java (Webサービスインタフェース生成ツールが生成します)

なお、同期アプリケーション連携実行基盤では、Javaのクライアントアプリケーションおよび Webサービスアプリケーションは IJServerクラスタまたはJ2EEのIJServerに配備し、C言語やCOBOLのサーバアプリケーションはCORBAワークユニットに配備して運用します。IJServerクラスタまたはJ2EEのIJServer、およびCORBAワークユニットの詳細は、“Interstage Application Server Java EE 運用ガイド”、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”および“Interstage Application Server OLTPサーバ運用ガイド”を参照してください。C言語のクライアンアプリケーションはサーバアプリケーション上で実行するか、または単体のアプリケーションとして実行するか、のいずれかで運用します。

注意

- Javaのクライアントアプリケーションは、Javaアプリケーションとしても実装可能ですが、テスト時だけ利用してください。実際の業務で利用するクライアントアプリケーションについては、IJServer上で運用する形態(EJBまたはServlet)を推奨します。
- WebサービスアプリケーションはCOBOLのサーバアプリケーションのみを呼出すことができます。
- サーバアプリケーションは、アプリケーション連携実行基盤で動作するアプリケーションです。サーバアプリケーションの詳細は、“Interstage Business Application Server アプリケーション開発ガイド”の“サーバアプリケーション編”を参照してください。

2.6.1.2 アプリケーションの連携パターン

同期アプリケーション連携実行基盤では、要求応答型のアプリケーション連携を行うことができます。クライアントアプリケーションがサーバアプリケーションに要求を発行し、サーバアプリケーションでは、処理結果をクライアントアプリケーションに応答します。
このとき、クライアントアプリケーションは、サーバアプリケーションの処理が完了するまで応答を待ち合わせます。

Webサービスアプリケーションを利用する場合、Webサービスクライアントアプリケーションから Webサービスアプリケーションに要求を発行します。Webサービスアプリケーションは同期アプリケーション連携実行基盤にリクエストを発効してサーバアプリケーションを呼出します。

サーバアプリケーションは、業務処理を行ったあとWebサービスアプリケーションに復帰します。Webサービスアプリケーションは同期アプリケーション連携実行基盤からのレスポンスをWebサービスクライアントアプリケーションに返却します。

このとき、Webサービスクライアントアプリケーションは、サーバアプリケーションの処理が完了するまで応答を待ち合わせます。

サーバアプリケーションからC言語のクライアントアプリケーションを実行して、さらに別のサーバアプリケーションを呼び出すことも可能です。

2.6.1.3 アプリケーションの構造

クライアントアプリケーションおよびサーバアプリケーションについて説明します。

■クライアントアプリケーション

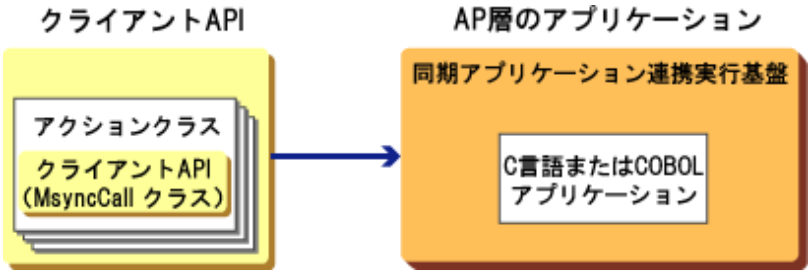
◆Javaのクライアントアプリケーション

Javaのクライアントアプリケーションは、ServletまたはEJBとして実装し、以下のいずれかの方法でクライアントAPIを発行することで、サーバアプリケーションを呼び出します。Java用のクライアントAPIには、以下の3種類があります。

- 一般的なJavaアプリケーションからの呼び出し
  サーバアプリケーションが、COBOLやC言語だけの場合に使用します。
- Apcoordinatorのリモート共通インタフェースによる呼び出し
  サーバアプリケーションが、COBOLまたはC言語に加え、Javaで作成する場合、別途、Apcoordinatorが提供するフレームワークを使用します。Apcoordinatorの詳細は、“Apcoordinator ユーザーズガイド”を参照してください。
- J2EE Connector ArchitectureのCommon Client Interface(CCI)による呼び出し
  サーバアプリケーションが、COBOLやC言語だけの場合に使用します。

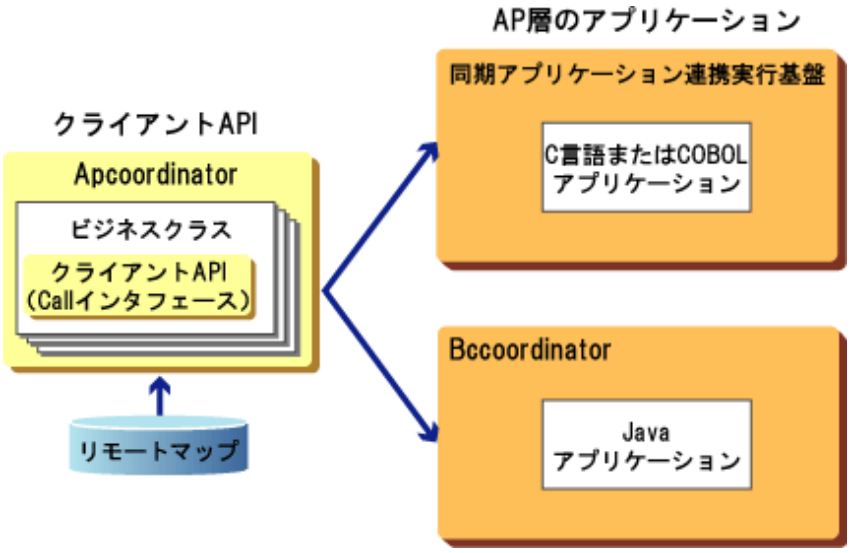
一般的なJavaアプリケーションからの呼び出し

一般的なJavaアプリケーションからサーバアプリケーションを呼び出すための機能を提供します。Servlet、JSP、およびEJBなどの一般アプリケーションからの呼び出しが可能です。



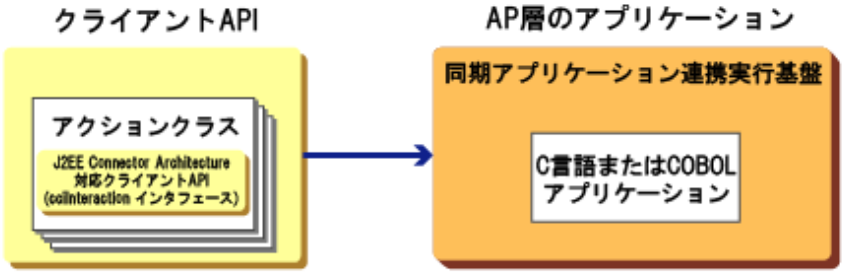
Apcoordinatorのリモート共通インタフェースによる呼び出し

Apcoordinatorのリモート共通インタフェースを使用して、サーバアプリケーションを呼び出すための機能を提供します。これにより、Apcoordinatorによるアプリケーション開発の親和性が高まり、EJBと共通のインタフェースによる呼び出しが可能となります。



J2EE Connector ArchitectureのCommon Client Interface(CCI)による呼び出し

J2EE Connector Architectureで規定されたインタフェースであるCCIを使用してサーバアプリケーションを呼び出すための機能を、同期アプリケーション連携実行基盤のResource Adapterとして提供します。これにより、J2EEの標準的なインタフェースによる呼び出しが可能となります。



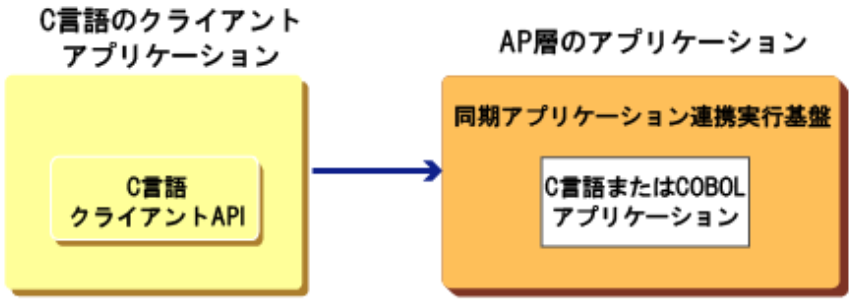
◆C言語のクライアントアプリケーション

C言語のクライアントアプリケーションは、サーバアプリケーション上からC言語用のクライアントAPIを発行することで、サーバアプリケーションを呼び出します。アプリケーションの構造には以下の2種類があります。C言語用のクライアントAPIは、サーバアプリケーションが、COBOLやC言語だけの場合に使用します。

- 単体のアプリケーションからの呼び出し
- 同期アプリケーション連携実行基盤上のサーバアプリケーションからの呼び出し

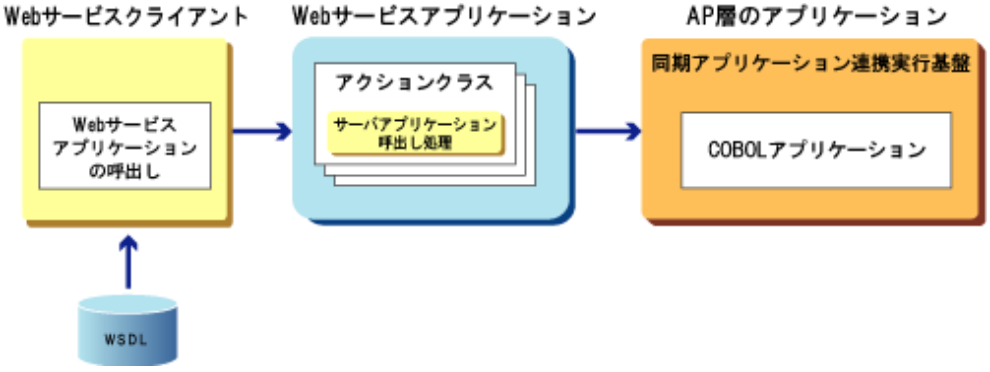
単体のアプリケーションからの呼び出し

C言語のクライアントアプリケーションから、サーバアプリケーションを呼び出すための機能を提供します。



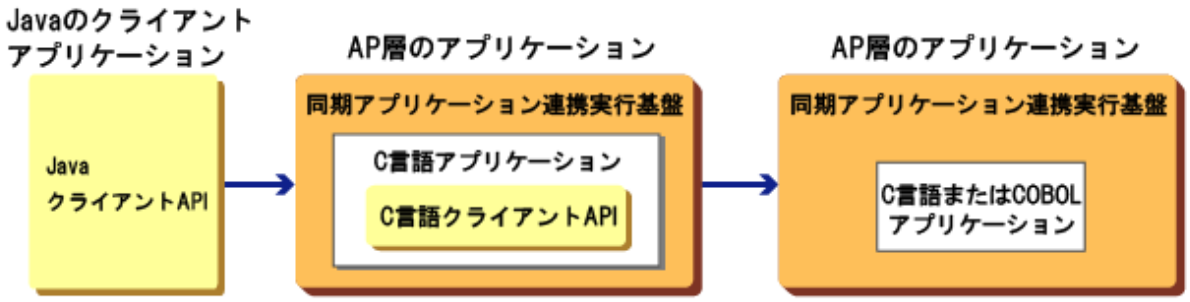
◆Webサービスクライアントアプリケーション

Webサービスクライアントアプリケーションは、Webサービスアプリケーションのインタフェースが定義されたWSDLをインポートしてWebサービスアプリケーションの呼出しを実装します。Webサービスアプリケーションは同期アプリケーション連携実行基盤のクライアントAPIを実行してサーバアプリケーションにリクエストを発行します。



◆同期アプリケーション連携実行基盤上のサーバアプリケーションからの呼び出し

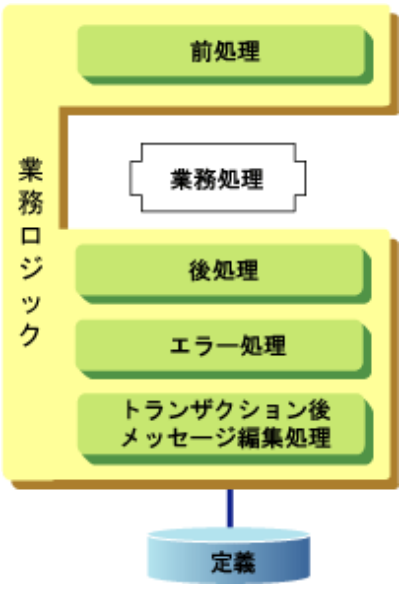
Javaのクライアントアプリケーションから呼び出されたサーバアプリケーションから別のサーバアプリケーションを呼び出します。
また、C言語クライアントから呼び出されたサーバアプリケーションから、さらに別のサーバアプリケーションを呼び出すことも可能です。
これにより、サーバアプリケーション間の連携を行うことができます。



■サーバアプリケーション

同期アプリケーション連携実行基盤は、クライアントアプリケーションから要求を受け付けると対応するサーバアプリケーション(業務ロジック)を定義ファイルに従い実行します。アプリケーションは、前処理、後処理、エラー処理およびトランザクション後メッセージ編集処理の各処理と業務処理から構成されます。

同期アプリケーション連携実行基盤では、各処理が返す処理結果により、トランザクションの完了を制御するなど、アプリケーションの動作を支援するため、利用者がアプリケーションの制御を行うロジックの開発を不要とします。



業務ロジックの処理の一覧を以下に説明します。

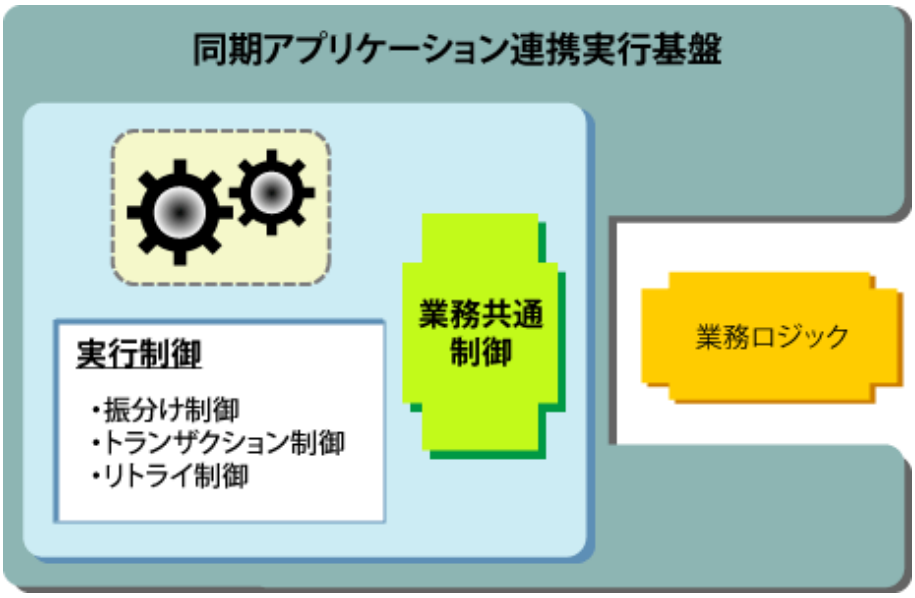
処理名	説明	要否
前処理	クライアントからのリクエスト(メッセージ)ごとに業務処理の直前に呼び出されます。 業務処理を開始するにあたってのリソースの初期化など、業務処理単位の初期処理に用います。	任意
後処理	クライアントからのリクエスト(メッセージ)ごとに前処理、および業務処理が正常終了した場合に、業務処理の直後に呼び出されます。 業務処理を終了するにあたってのリソースの解放など、業務処理単位の終了処理に用います。	任意

処理名	説明	要否
エラー処理	前処理、業務処理、または後処理が異常終了した場合、その直後に呼び出されます。 エラー発生時のリソースの初期化など、業務単位のエラー後処理に用います。	任意
トランザクション後メッセージ編集処理	アプリケーション連携実行基盤で管理しているトランザクション完了後に呼び出されます。 トランザクション完了の結果により、クライアントへ返却するメッセージを編集する場合に用います。トランザクション後メッセージ編集処理はコンテナのトランザクション制御機能が有効の場合のみ呼び出されます。	任意

同期アプリケーション連携実行基盤のアプリケーション制御は、必要に応じて業務共通制御でカスタマイズすることもできます。
以下に業務共通制御について説明します。

◆業務共通制御

業務共通制御とは、同期アプリケーション連携実行基盤が提供する振分け制御、トランザクション制御、リトライ制御といった実行制御をカスタマイズするための制御ロジックです。同期アプリケーション連携実行基盤の振分け制御、業務前制御、業務後制御、および結果編集制御の各種制御動作を利用者が任意に変更することができます。



業務共通制御の処理の一覧を以下に説明します。

処理名	説明	要否
振分け制御	クライアントからの要求ごとに呼び出す処理です。 制御用のデータを受け取り、どの業務処理を呼び出すかを決定するための処理です。 アプリケーション連携実行基盤が提供するAPIを使用して、呼び出す業務処理(ターゲット名)を変更することができます。 1つのアプリケーション連携実行基盤に対して1つだけ定義することができます。	任意
業務前制御	サーバアプリケーションの呼出しの直前に呼び出す処理です。 業務処理に必要なリソースの獲得・初期化や、業務処理の実行状態の管理などの処理を行います。 1つのアプリケーション連携実行基盤に対して1つだけ定義することができます。	任意
業務後制御	サーバアプリケーションの呼び出しの直後に呼び出す処理です。 業務処理で使用したリソースの解放や、業務処理の実行結果の管理などの処理を行います。 処理結果情報を変更することでアプリケーション連携実行基盤に対してトランザクションの完了指示を行うことができます。 1つのアプリケーション連携実行基盤に対して1つだけ定義することができます。	任意
結果編集制御	アプリケーション連携実行基盤で管理しているトランザクションの完了後に呼び出す処理です。	任意

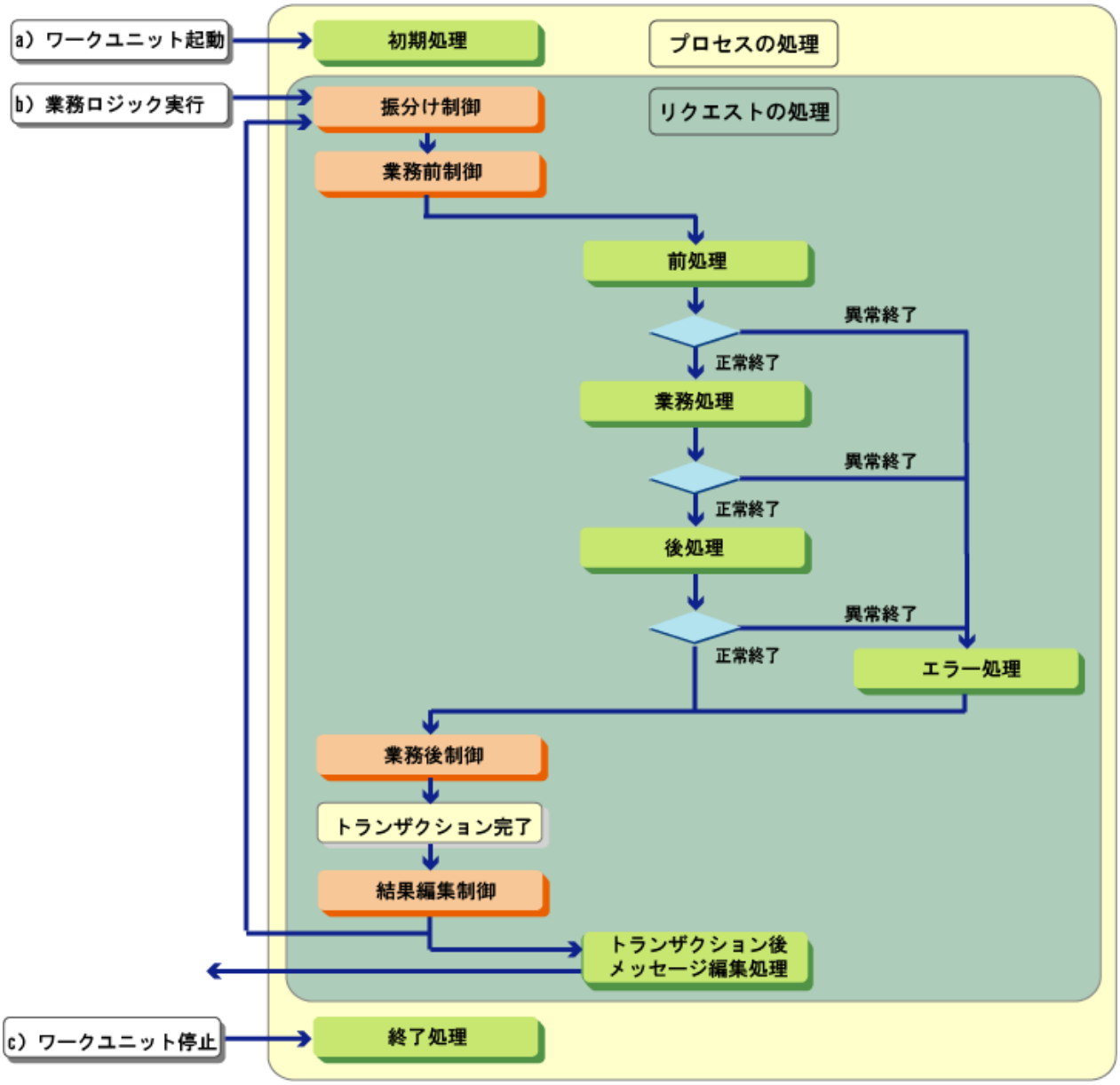
処理名	説明	要否
	処理結果情報やトランザクションの完了状態を判定し、電文を返却するか、例外として通知するか、またはリトライするかの制御を行います。 アプリケーション連携実行基盤のデータベースアクセス管理機能を使用しない場合も呼び出されます。 1つのアプリケーション連携実行基盤に対して1つだけ定義することができます。	

注意

- 業務共通制御は、C言語で開発することができます。

◆サーバアプリケーションと業務共通制御の処理シーケンス

サーバアプリケーション(業務ロジック)と業務共通制御の各種処理の呼び出しシーケンスは下記の通りです。ワークユニットのプロセス起動時、および終了時に、アプリケーション連携実行基盤が呼び出す出口処理に加え、ワークユニット出口を呼び出すことができます。ワークユニット出口の詳細は、“Interstage Application Server OLTPサーバ運用ガイド”を参照してください。



2.6.1.4 アプリケーションの実行環境

同期アプリケーション連携実行基盤のアプリケーションの実行環境について説明します。なお、アプリケーションの実行環境として使用するワークユニットの機能およびInterstage管理コンソールについては“Interstage Application Server 運用ガイド(基本編)”を参照してください。

■クライアントアプリケーションの実行環境

クライアントアプリケーションは、Interstageの開発環境であるInterstage Studioを使用してJSP、ServletまたはEJBとして作成し、IJServerクラスタまたはJ2EEのIJServerに配備して実行します。

■サーバアプリケーションの実行環境

サーバアプリケーションは、実行基盤インタフェースとともに共有ライブラリとして作成し、本製品の配備コマンド(apfwdeploy)を使用してCORBAワークユニットに配備します。
CORBAワークユニットは、本製品の“2.7.1.1 システム構築シート”で出力したワークユニット管理コマンドまたは、“2.7.1.3 環境構築コマンド”を使用して定義し、Interstage管理コンソールを使用して運用します。
なお、サーバアプリケーションの実行制御は、同期アプリケーション連携実行基盤が行います。同期アプリケーション連携実行基盤によるアプリケーションの制御機能について以下に説明します。

動作モード

サーバアプリケーションの動作モードとしてスレッドモードとプロセスモードを提供します。アプリケーションの多重度は、スレッド単位(1プロセスnスレッド)、プロセス単位(nプロセス)および混在(mプロセスnスレッド)で設定できます。
動作モードは、使用するアプリケーションの言語特性(COBOLでの実行環境の引継ぎ)やアプリケーションが異常終了した場合の影響範囲など、業務に応じて選択してください。

トランザクションとリトライ制御

データベース操作を伴う処理では、処理結果により、トランザクションの完了操作を行う必要があります。また、エラー終了した場合には、業務処理の再実行が必要な場合があります。同期アプリケーション連携実行基盤では、これらの操作をアプリケーションの処理結果情報に2(異常終了)または3(強制リトライ)が設定されていた場合にリトライの定義に従ってリトライ処理を行います。リトライ実行時のリクエストデータは初回呼び出し時のデータが渡されます。

プレロード・ダイナミックロード

アプリケーションのライブラリをロードする方式にプレロードとダイナミックロードの2種類の方式を提供します。

- プレロード

  同期アプリケーション連携実行基盤のプロセス起動時にユーザ作成ライブラリをロードします。性能向上のためアプリケーションをプロセス起動時にメモリに常駐する場合に使用し、アプリケーションのアクセスを初回から高速化できます。

- ダイナミックロード

  ユーザ作成ライブラリをリクエスト時に動的にロードを行い、リクエスト終了後にアンロードします。常に動作する必要がないユーザ作成ライブラリやユーザ作成ライブラリの常駐メモリを節約する場合などに用います。

業務共通制御

同期アプリケーション連携実行基盤の提供する振分け制御、トランザクション制御、リトライ制御などの実行制御をカスタマイズするための制御ロジックです。これにより、制御ロジックと業務ロジックを分離して、業務ロジックの独立性が高めることが可能です。

COBOL実行環境の開設・閉鎖

COBOL実行環境の開設・閉鎖処理を同期アプリケーション連携実行基盤が実施します。WORKING-STRAGE SECTION における無駄な領域の獲得や解放を行わない制御が行えます。COBOL実行環境の開設から閉鎖までの間、変数などの実行環境情報がCOBOLランタイムで保持される(前回呼び出し時の状態が保持される)ため、アプリケーションで共通に使用する情報を保持しておくことができます。

データ型変換

サーバアプリケーションの言語に合わせて、データ型の変換処理を行います。これにより、すでに作成されているアプリケーションの移植性を高めます。データ型の対応関係の詳細は、“Interstage Business Application Server アプリケーション開発ガイド”の“サーバアプリケーション開発の概要”の“データ型の変換”を参照してください。

ログ出力

同期アプリケーション連携実行基盤の稼動状況および性能情報の2種類のログを自動的に出力する標準ログ機能を提供します。
標準ログを使用することでユーザのアプリケーションにログの出力処理を記述しなくても、上記ログを自動的に採取することができます。
詳細は、“5.2 標準ログ”を参照してください。

2.6.2 非同期アプリケーション連携実行基盤 EE

非同期アプリケーション連携実行基盤では、ディレイド型のオンライン処理(突き放し型など)および複数業務を相互接続する業務モデル(HUB型)における処理形態において、フロー定義に従ってCOBOLまたはJavaのアプリケーション連携を行い、処理結果を応答する基幹系業務に必要な共通技術を提供します。オープンシステムにおいて、メインフレームレベルの業務運用性、堅牢性を実現し、短期間で、高い信頼性のシステムを実現します。また、既存資産や既存ノウハウを最大限に活かし、業務ロジックの開発に専念することが可能です。

以下に非同期アプリケーション連携実行基盤のアプリケーションについて説明します。
非同期アプリケーション連携実行基盤の詳細は、“第3部 非同期アプリケーション連携実行基盤編”で説明します。

注意

1つのフロー定義では、異なる言語(COBOLまたはJava)のアプリケーションを相互に連携させることはできません。

2.6.2.1 アプリケーションの種類

非同期アプリケーション連携実行基盤では、業務処理開始アプリケーションおよび業務処理実行アプリケーションを使用して業務を構築します。

アプリケーションの種類	アプリケーションの概要	利用可能な言語(アプリケーションの形態)
業務処理開始アプリケーション	非同期アプリケーション連携実行基盤のフロー起動APIまたは結果受信APIを実行して業務処理実行アプリケーションを連携した業務(フロー)の開始、および業務(フロー)の処理結果を受信するアプリケーションです。	Java (Servlet/EJBまたはJavaアプリケーション)
業務処理実行アプリケーション	メッセージの内容に応じて業務用のデータベースの更新などの業務処理を行うアプリケーションです。 COBOLでアプリケーションを作成する場合、同期アプリケーション連携実行基盤のサーバアプリケーションと同一の方法でアプリケーションを作成します。	COBOL (共有ライブラリ)
		Java (クラス)

なお、非同期アプリケーション連携実行基盤では、業務処理開始アプリケーションをJavaで開発する場合は、J2EEのIJServerに配備して運用します。
また、業務処理実行アプリケーションをCOBOLで作成する場合は、非同期ワークユニットに配備し、Javaで作成する場合は、J2EEのIJServerに配備して運用します。IJServerおよび非同期ワークユニットの詳細は、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”および“Interstage Business Application Server 運用ガイド(アプリケーション連携実行基盤編)”を参照してください。

注意

- 非同期アプリケーション連携実行基盤は、JavaEEでは運用できません。また、利用できるJDK/JREはJDK6/JRE6のみです。
- 業務処理開始アプリケーションは、Javaアプリケーションの単体のアプリケーションとしても実装可能ですが、テスト時だけ利用してください。実際の業務で利用するクライアントアプリケーションについては、IJServer上で運用する形態(Servlet、JSPまたはEJB)を推奨します。

非同期アプリケーション連携実行基盤の業務処理開始アプリケーションと業務処理実行アプリケーションの開発言語は、以下の組み合わせをサポートしています。利用業務に応じて最適なパターンを使用してください。

		業務処理実行アプリケーション	
		COBOL	Java
業務処理開始アプリケーション	Java	○(注1)	○

○:連携が可能です。

注1) Javaのユーザ定義クラスは、COBOLアプリケーションで入出力できません。

2.6.2.2 アプリケーションの連携パターン

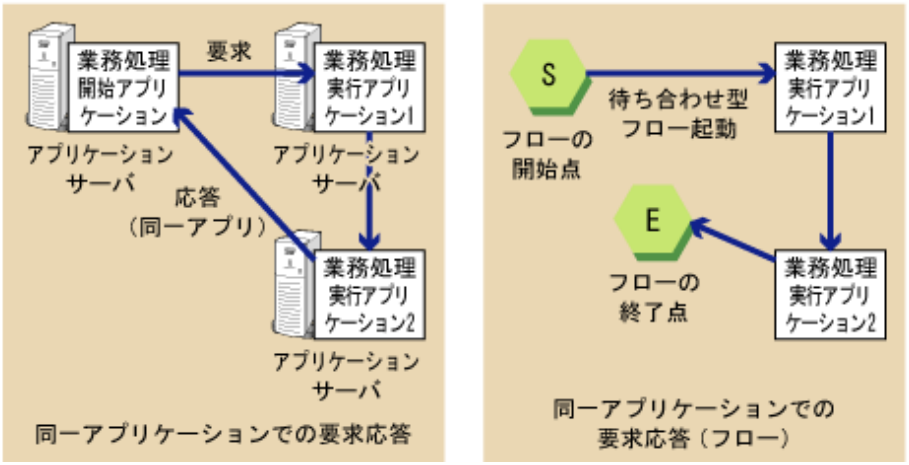
非同期アプリケーション連携実行基盤で利用できるアプリケーションの連携パターンの例を以下に示します。非同期アプリケーション連携実行基盤では、アプリケーションの連携形態は、フロー定義ツールで作成するフロー定義により決定します。
なお、以下に示す例は、代表的なアプリケーション連携パターンです。フロー定義によっては、これらのほかにも構築可能な連携のパターンがあります。

- 要求応答型
- 突き放し型
- 回覧型
- 同報型

■要求応答型

要求を発行したアプリケーションまたは別のアプリケーションに対して、処理結果を応答として返す形態です。他の形態と組み合わせて使用することができます。

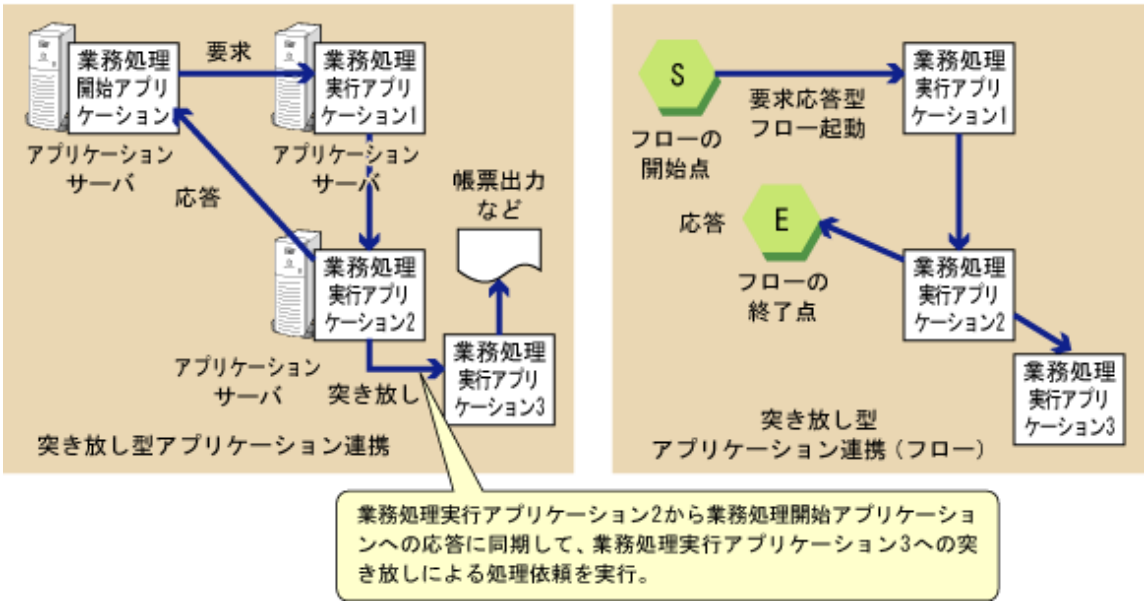
[要求応答型のアプリケーション連携（同一アプリケーションでの応答受信）]



■突き放し型

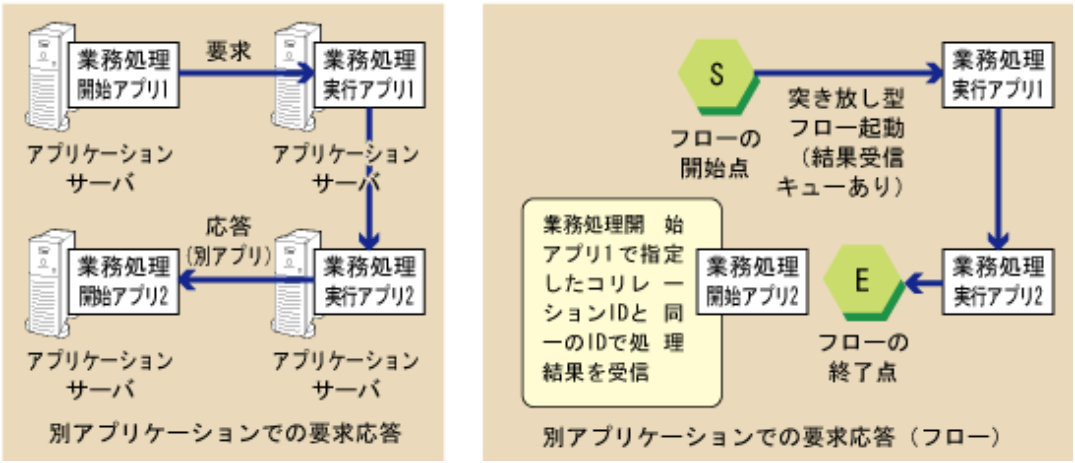
アプリケーションの実行完了と同期して、別のアプリケーションに処理を依頼する形態です。他の形態と組み合わせて使用することができます。
以下の形態では、業務処理開始アプリケーションは、処理結果を待ち合わせるため要求応答型フローを起動しますが、業務処理実行アプリケーション3の処理は突き放しで実行されます。

［突き放し型のアプリケーション連携］



以下の形態では、業務処理実行アプリケーションが、結果を待ち合わせないため、フローの起動は、突き放し型フローとして行われ、処理結果は、別のアプリケーションで取得します。

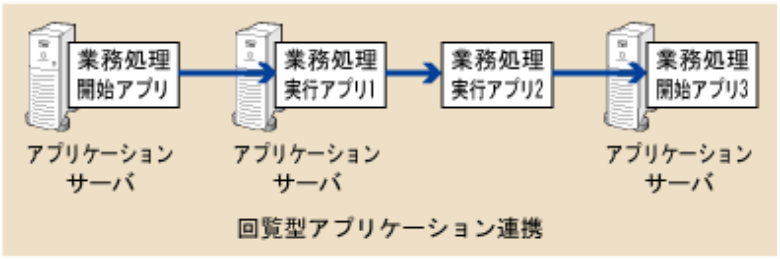
［突き放し型のアプリケーション連携（別アプリケーションでの応答受信）］

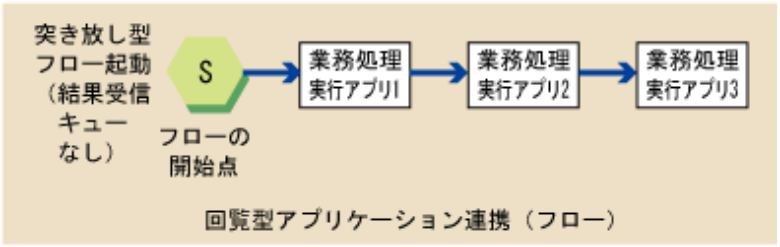


■回覧型

アプリケーション間で回覧型の通信を繰り返す形態です。ワークフロー業務の処理段階に対応するアプリケーションを順次実行する場合に使用します。回覧型で連携するアプリケーションは、同期通信のように処理が完了する度に呼び出し元に復帰(リターン)することなく、次の処理を実行するアプリケーションを呼び出します。

［回覧型アプリケーション連携］

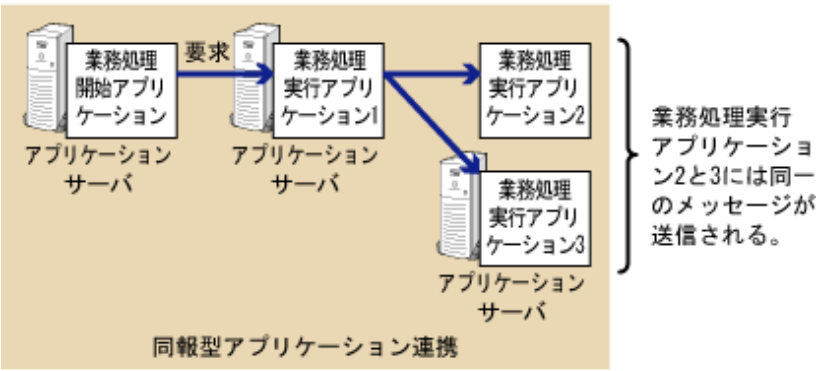


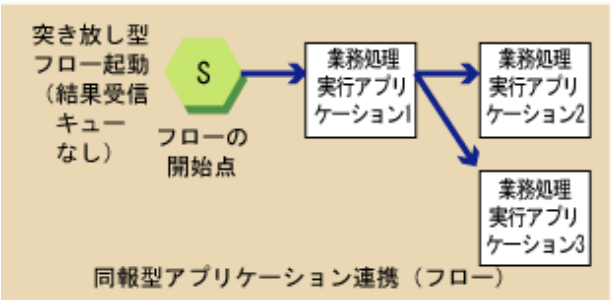


■同報型

アプリケーションの実行完了と同期して、n個の別アプリケーションに同一メッセージを一斉に送信する形態です。

［同報型アプリケーション連携］





2.6.2.3 アプリケーションの構造

業務処理開始アプリケーションおよび業務処理実行アプリケーションの構造について説明します。業務処理実行アプリケーションの構造は、同期アプリケーション連携実行基盤のサーバアプリケーションと同一です。
また、エラーメッセージを処理するアプリケーションについても説明します。

■業務処理開始アプリケーション

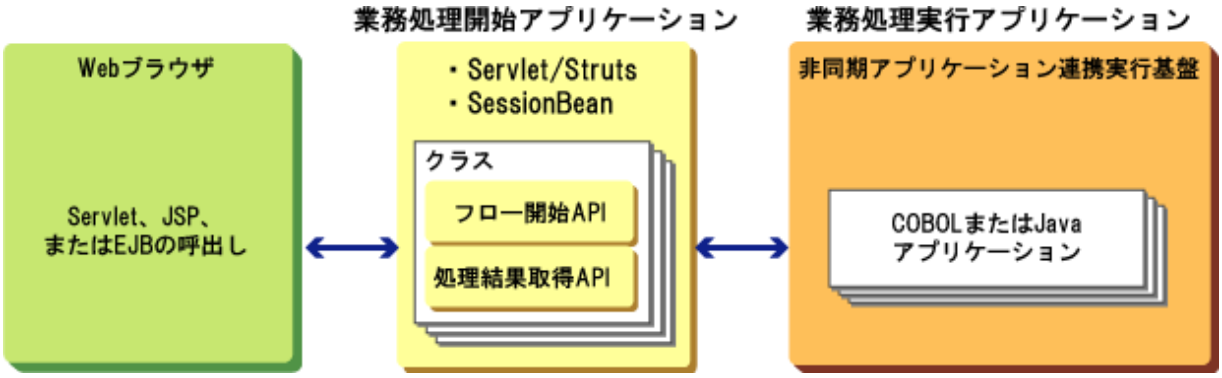
業務処理開始アプリケーションは、Javaアプリケーションであり、フロー起動APIを発行することで、業務処理実行アプリケーションを呼び出します。

Javaのフロー起動APIには、以下の3種類があります。

- SendMessage()
  フローを起動するAPIです。フローを突き放して実行します。
- ReceiveMessage()
  フローの処理結果を受信するAPIです。同期待ち合わせ型と即時応答型の2種類があります。
- SendMessageSync()
  フローを開始し、処理結果を待ち合わせるAPIです。

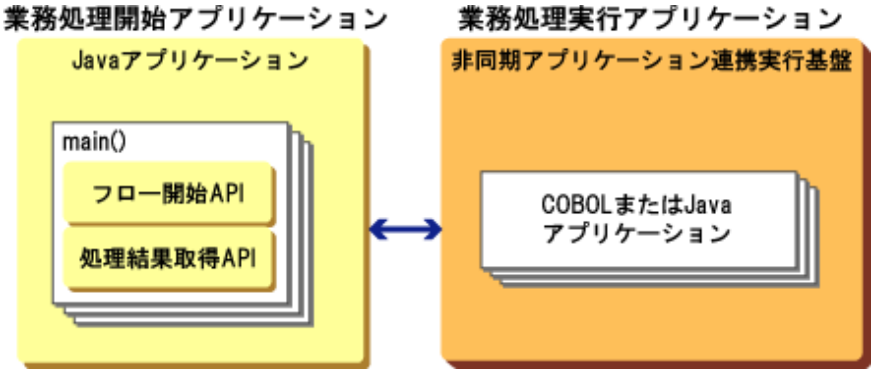
◆ServletまたはEJBアプリケーションからの呼び出し

Servlet、StrutsおよびSession Beanなどのアプリケーションで、フロー起動APIを実行することによりフローを開始し、フロー定義に従って業務処理実行アプリケーションを運用する形態です。業務処理開始アプリケーションは、IJServerに配備して実行します。



◆J2EEアプリケーションクライアントからの呼び出し

Javaアプリケーションで、フロー起動APIを実行することによりフローを開始し、フロー定義に従って業務処理実行アプリケーションを運用する形態です。業務処理開始アプリケーションは、コマンドラインからjavaコマンドで実行します。
本呼び出し形態は単体テスト時だけ利用してください。実際の業務で利用するクライアントアプリケーションについては、IJServer上で運用する形態(Servlet、JSPまたはEJB)を推奨します。



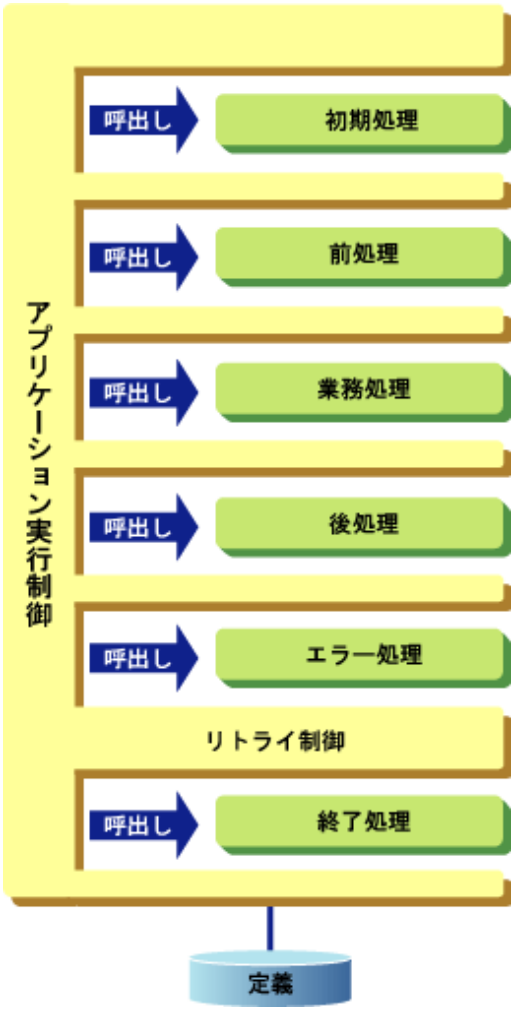
■業務処理実行アプリケーション

非同期アプリケーション連携実行基盤が提供するアプリケーション実行制御では、フロー定義に従い、業務処理実行アプリケーションの呼び出しを行います。また、各処理が返す処理結果により、トランザクションの完了を制御するなど、ユーザアプリケーションの動作を支援する出口処理(初期処理、前処理、後処理、エラー処理、および終了処理)を提供します。
制御ロジックは、非同期アプリケーション連携実行基盤が実行するため、利用者は、業務処理(業務ロジック)だけを記述します。

なお、業務処理実行アプリケーションは、開発する言語により、以下に示す出口の相違があります。

◆COBOLの業務処理実行アプリケーション

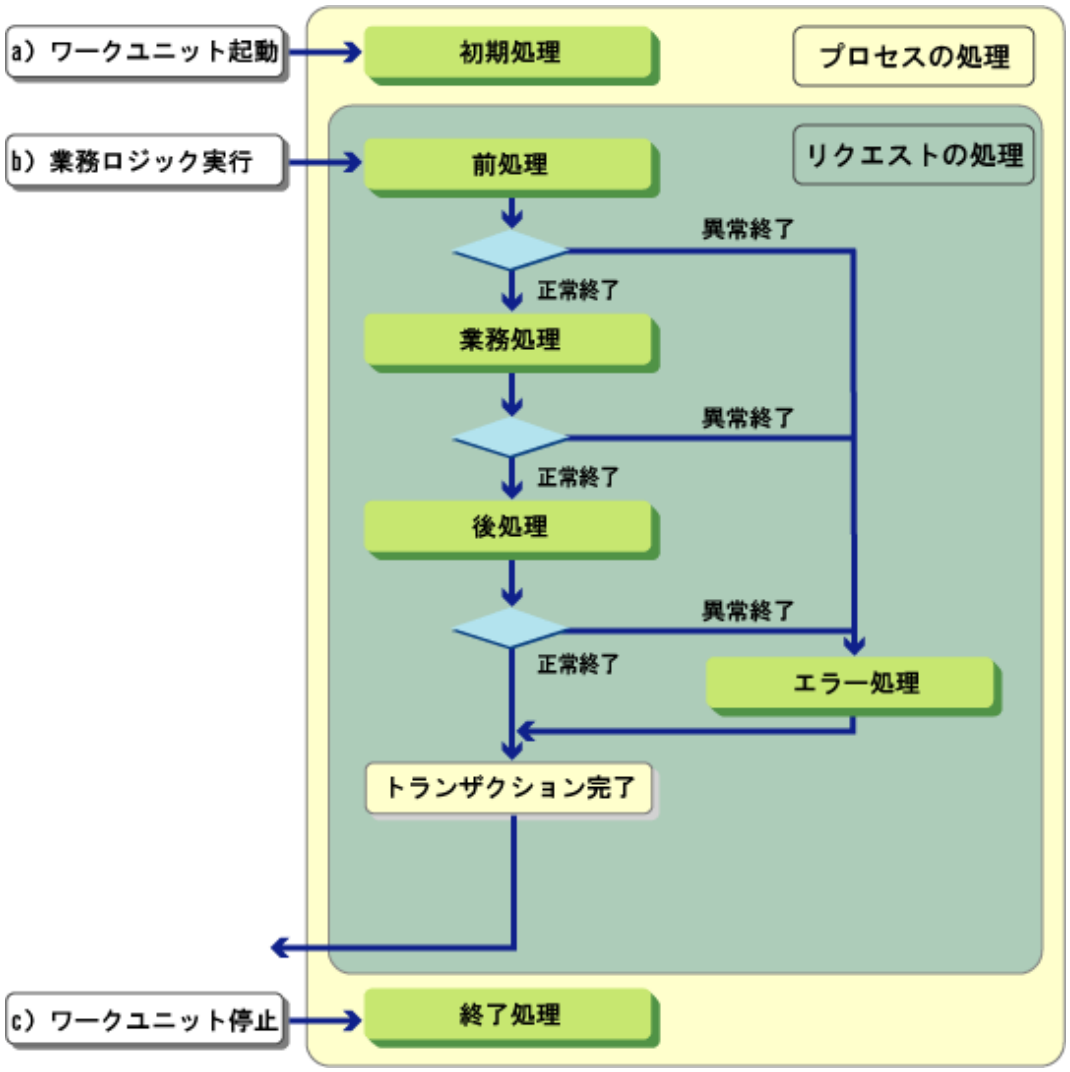
COBOLで業務処理実行アプリケーションを作成した場合、ユーザアプリケーションの動作を支援する出口処理には、初期処理、前処理、後処理、エラー処理および終了処理があります。アプリケーションの構造を以下に示します。
なお、アプリケーションは、非同期ワークユニットに配備して実行します。



アプリケーション連携実行基盤が提供する出口処理の概要を以下に説明します。

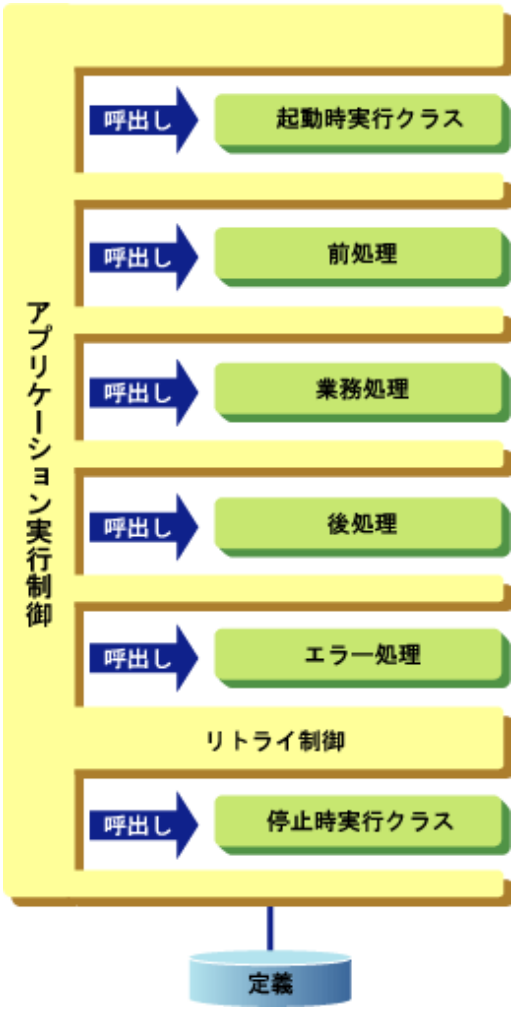
処理名	説明	要否
初期処理	ワークユニット起動時に1プロセスあたり1回だけ呼び出されます。 業務処理に必要なリソースの獲得、初期化等の処理に用います。	任意
前処理	クライアントからのリクエスト(メッセージ)ごとに業務処理の直前に呼び出されます。 業務処理を開始するにあたってのリソースの初期化など、業務処理単位の初期処理に用います。	任意
後処理	クライアントからのリクエスト(メッセージ)ごとに前処理、および業務処理が正常終了した場合に、業務処理の直後に呼び出されます。 業務処理を終了するにあたってのリソースの解放など、業務処理単位の終了処理に用います。	任意
エラー処理	以下の場合、その直後に呼び出されます。エラー発生時のリソースの初期化など、業務単位のエラー後処理に用います。 前処理、業務処理、または後処理が異常終了した場合 前処理、業務処理、または後処理が強制リトライで終了した場合(処理結果情報に3が設定された場合)	任意
終了処理	ワークユニット終了時に1プロセスあたり1回だけ呼び出されます。 各処理で獲得したリソースの解放等に用います。	任意

また、各出口処理の呼び出しシーケンスは下記の通りです。ワークユニットのプロセス起動時、および終了時に、アプリケーション連携実行基盤が呼び出す出口処理に加え、ワークユニット出口を呼び出すことができます。ワークユニット出口の詳細は、“Interstage Business Application Server 運用ガイド(アプリケーション連携実行基盤編)”を参照してください。



◆Javaの業務処理実行アプリケーション

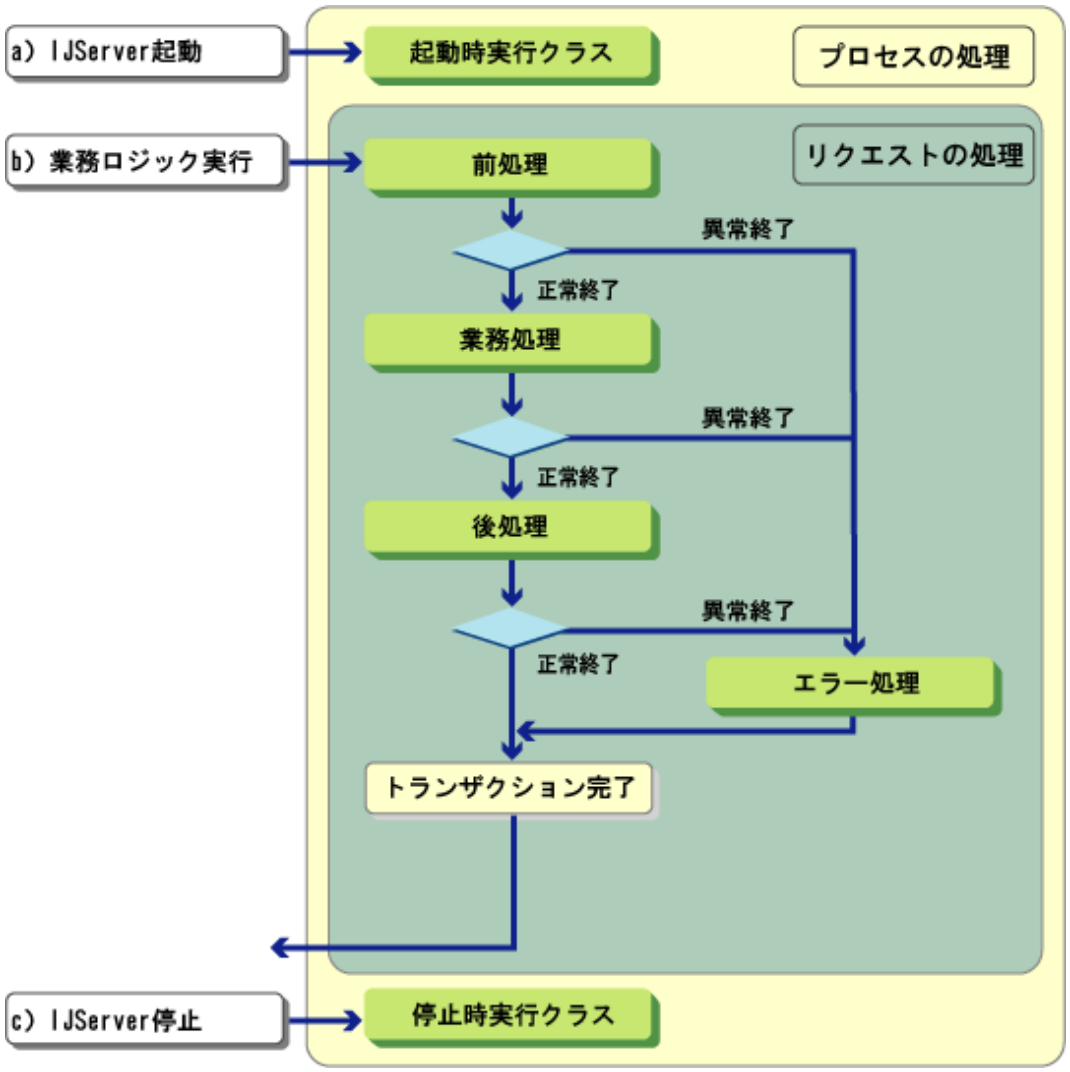
Javaで業務処理実行アプリケーションを作成した場合、ユーザアプリケーションの動作を支援する出口処理には、前処理、後処理およびエラー処理があります。アプリケーションの構造を以下に示します。
なお、アプリケーションは、IJServerに配備して実行します。起動時実行クラスおよび停止時実行クラスは、IJServerが提供する機能です。



アプリケーション連携実行基盤が提供する出口処理の概要を以下に説明します。

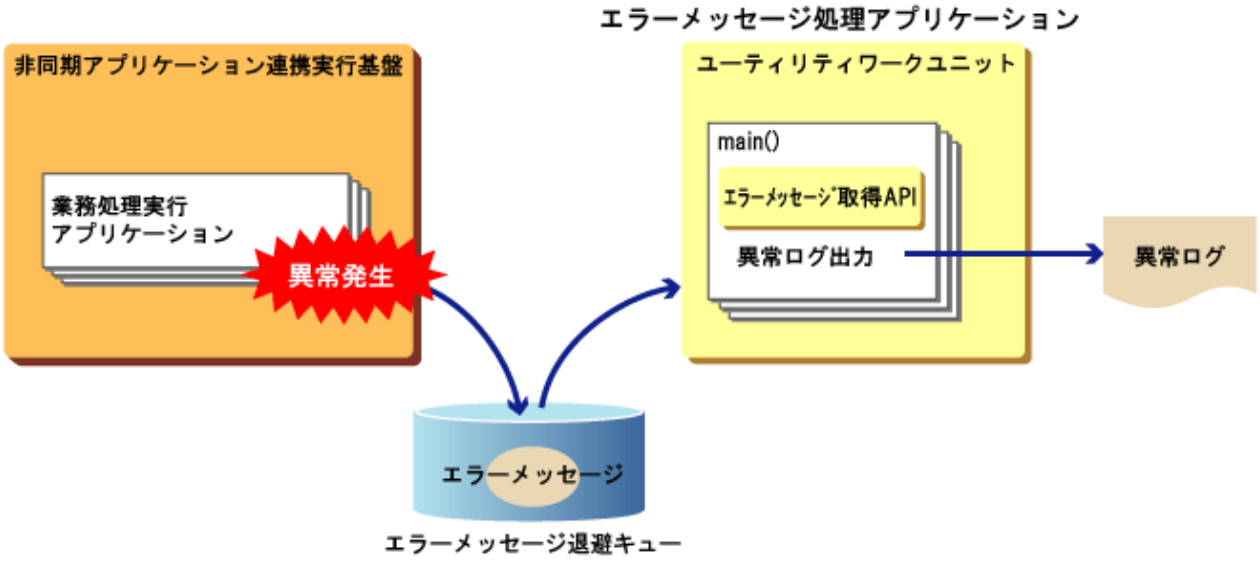
処理名	説明	要否
起動時実行クラス	IJServerの起動時に実行されるクラスです。データベースのコネクションなどアプリケーションで使用する資源の事前獲得などを行う場合に使用します。	任意
前処理	クライアントからのリクエスト(メッセージ)ごとに業務処理の直前に呼び出されます。 業務処理を開始するにあたってのリソースの初期化など、業務処理単位の初期処理に用います。	任意
後処理	クライアントからのリクエスト(メッセージ)ごとに前処理、および業務処理が正常終了した場合に、業務処理の直後に呼び出されます。 業務処理を終了するにあたってのリソースの解放など、業務処理単位の終了処理に用います。	任意
エラー処理	以下の場合、その直後に呼び出されます。エラー発生時のリソースの初期化など、業務単位のエラー後処理に用います。 前処理、業務処理、または後処理が異常終了した場合 前処理、業務処理、または後処理が強制リトライで終了した場合(処理結果情報に3が設定された場合)	任意
停止時実行クラス	IJServerの停止時に実行されるクラスです。アプリケーションで使用した資源を一括して解放するなどの処理に使用します。	任意

また、各出口処理の呼び出しシーケンスは下記の通りです。IJServerおよび起動/停止時の実行クラスの詳細は、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。



■エラーメッセージ処理アプリケーション

エラーメッセージ処理アプリケーションは、業務処理実行アプリケーションで異常が発生した後、エラーメッセージ退避キューに格納されたメッセージをエラーメッセージ取得APIを実行することで受信し、エラー内容を業務ログなどに出力する異常時運用を行うアプリケーションです。
アプリケーションは、単体のアプリケーションとして作成し、ユーティリティワークユニットに配備して実行します。



2.6.2.4 アプリケーションの実行環境

非同期アプリケーション連携実行基盤のアプリケーションの実行環境について説明します。なお、アプリケーションの実行環境として使用するワークユニットの機能およびInterstage管理コンソールについては“Interstage Application Server 運用ガイド(基本編)”を参照してください。

■業務処理開始アプリケーションの実行環境(Javaの場合)

業務処理開始アプリケーションは、Interstageの開発環境であるInterstage Studioを使用してフロー定義ツールのウィザード(本製品のプラグインモジュール)に従ってJSP、ServletまたはEJBとして作成し、IJServerに配備して実行します。IJServerはJ2EEアプリケーションの実行環境であるEJBコンテナとServletコンテナを内包し、これらのコンテナの上位に位置づけられる論理的な概念です。
IJServerはInterstage Application Serverの特徴であるワークユニットと呼ぶアプリケーション運用機能上で動作し、ワークユニットが提供している高度なアプリケーション運用操作/監視機能を利用できます。IJServerは、“2.7.1.1 システム構築シート”の出力結果を使用して作成しInterstage管理コンソールを使用して運用します。

■業務処理実行アプリケーションの実行環境(COBOLの場合)

業務処理実行アプリケーションは、実行基盤インタフェースとともに共有ライブラリとして作成し、本製品の配備コマンド(apfwdeploy)を使用して非同期ワークユニットに配備します。非同期ワークユニットには、ワークユニット運用において、対象となるアプリケーションに業務処理実行アプリケーションを適用することができます。
非同期ワークユニットは、本製品の“2.7.1.1 システム構築シート”で出力したワークユニット管理コマンドを使用して定義し、ワークユニットのコマンドを使用して運用します。
なお、業務処理実行アプリケーションの実行制御は、非同期アプリケーション連携実行基盤が行います。非同期アプリケーション連携実行基盤によるアプリケーションの制御機能について以下に説明します。

動作モード

業務処理実行アプリケーションの動作モードとしてスレッドモードとプロセスモードを提供します。アプリケーションの多重度は、スレッド単位(1プロセスnスレッド)、プロセス単位(nプロセス)および混在(mプロセスnスレッド)で設定できます。
動作モードは、使用するアプリケーションの言語特性(COBOLでの実行環境の引継ぎ)やアプリケーションが異常終了した場合の影響範囲など、業務に応じて選択してください。

トランザクションとリトライ制御

データベース操作を伴う処理では、処理結果により、トランザクションの完了操作を行う必要があります。また、エラー終了した場合には、業務処理の再実行が必要な場合があります。非同期アプリケーション連携実行基盤では、これらの操作をアプリケーションの処理結果情報に2(異常終了)が設定されていた場合にフロー定義に従ってリトライ処理を行います。リトライ実行時のリクエストデータは初回呼び出し時のデータが渡されます。

プレロード、ダイナミックロード

アプリケーションのライブラリをロードする方式にプレロードとダイナミックロードの2種類の方式を提供します。

- プレロード

  非同期アプリケーション連携実行基盤のプロセス起動時にユーザ作成ライブラリをロードします。性能向上のためアプリケーションをプロセス起動時にメモリに常駐する場合に使用し、アプリケーションのアクセスを初回から高速化できます。

- ダイナミックロード

  ユーザ作成ライブラリをリクエスト時に動的にロードを行い、リクエスト終了後にアンロードします。常に動作する必要がないユーザ作成ライブラリやユーザ作成ライブラリの常駐メモリを節約する場合などに用います。

COBOL実行環境の開設・閉鎖

COBOL実行環境の開設・閉鎖処理を非同期アプリケーション連携実行基盤が実施します。WORKING-STRAGE SECTION における無駄な領域の獲得や解放を行わない制御が行えます。COBOL実行環境の開設から閉鎖までの間、変数などの実行環境情報がCOBOLランタイムで保持される(前回呼び出し時の状態が保持される)ため、アプリケーションで共通に使用する情報を保持しておくことができます。

データ型変換

業務処理実行アプリケーションの言語に合わせて、データ型の変換処理を行います。これにより、すでに作成されているアプリケーションの移植性を高めます。データ型の対応関係の詳細は、“Interstage Business Application Server アプリケーション開発ガイド”の“サーバアプリケーション開発の概要”の“データ型の変換”を参照してください。

ログ出力

非同期アプリケーション連携実行基盤の稼動状況および性能情報の2種類のログを自動的に出力する標準ログ機能を提供します。標準ログを使用することでユーザのアプリケーションにログの出力処理を記述しなくても、上記ログを自動的に採取することができます。詳細は、“5.2 標準ログ”を参照してください。

異常処理

アプリケーションの処理結果とフロー定義の内容に応じてアプリケーションで異常が発生した場合、受信したメッセージを“エラーメッセージ退避キューに退避”、“シリアライズファイルへ出力”または“補償ルート”のいずれかにより異常処理することができます。

■業務処理実行アプリケーションの実行環境(Javaの場合)

業務処理実行アプリケーションは、Interstageの開発環境であるInterstage Studioを使用してフロー定義ツールのウィザード(本製品のプラグインモジュール)に従ってJavaクラスとして作成し、Interstage管理コンソールを使用してIJServerにMessage-driven Beanとして配備します。IJServerはJ2EEアプリケーションの実行環境であるEJBコンテナとServletコンテナを内包し、これらのコンテナの上位に位置づけられる論理的な概念です。
IJServerはInterstage Application Serverの特徴であるワークユニットと呼ぶアプリケーション運用機能上で動作し、ワークユニットが提供している高度なアプリケーション運用操作/監視機能を利用できます。IJServerは、“2.7.1.1 システム構築シート”の出力結果を元に作成し、Interstage管理コンソールを使用して作成します。
なお、業務処理実行アプリケーションの実行制御は、EJBコンテナおよび非同期アプリケーション連携実行基盤が行います。非同期アプリケーション連携実行基盤によるアプリケーションの制御機能について以下に説明します。

動作モード

業務処理実行アプリケーションの動作モードとしてスレッドモードを提供します。アプリケーションの多重度は、スレッド単位(1プロセスnスレッド)が設定できます。

トランザクションとリトライ制御

データベース操作を伴う処理では、処理結果により、トランザクションの完了操作を行う必要があります。また、エラー終了した場合には、業務処理の再実行が必要な場合があります。非同期アプリケーション連携実行基盤では、これらの操作をアプリケーションで発生した例外内容を元にフロー定義に従ってリトライ処理を行います。リトライ実行時のリクエストデータは初回呼び出し時のデータが渡されます。

ログ出力

非同期アプリケーション連携実行基盤の稼動状況および性能情報の2種類のログを自動的に出力する標準ログ機能を提供します。標準ログを使用することでユーザのアプリケーションにログの出力処理を記述しなくても、上記ログを自動的に採取することができます。詳細は、“5.2 標準ログ”を参照してください。

異常処理

アプリケーションの処理結果とフロー定義の内容に応じてアプリケーションで異常が発生した場合、受信したメッセージを“エラーメッセージ退避キューに退避”、“シリアライズファイルへ出力”または“補償ルート”のいずれかにより異常処理することができます。

■エラーメッセージ処理アプリケーション

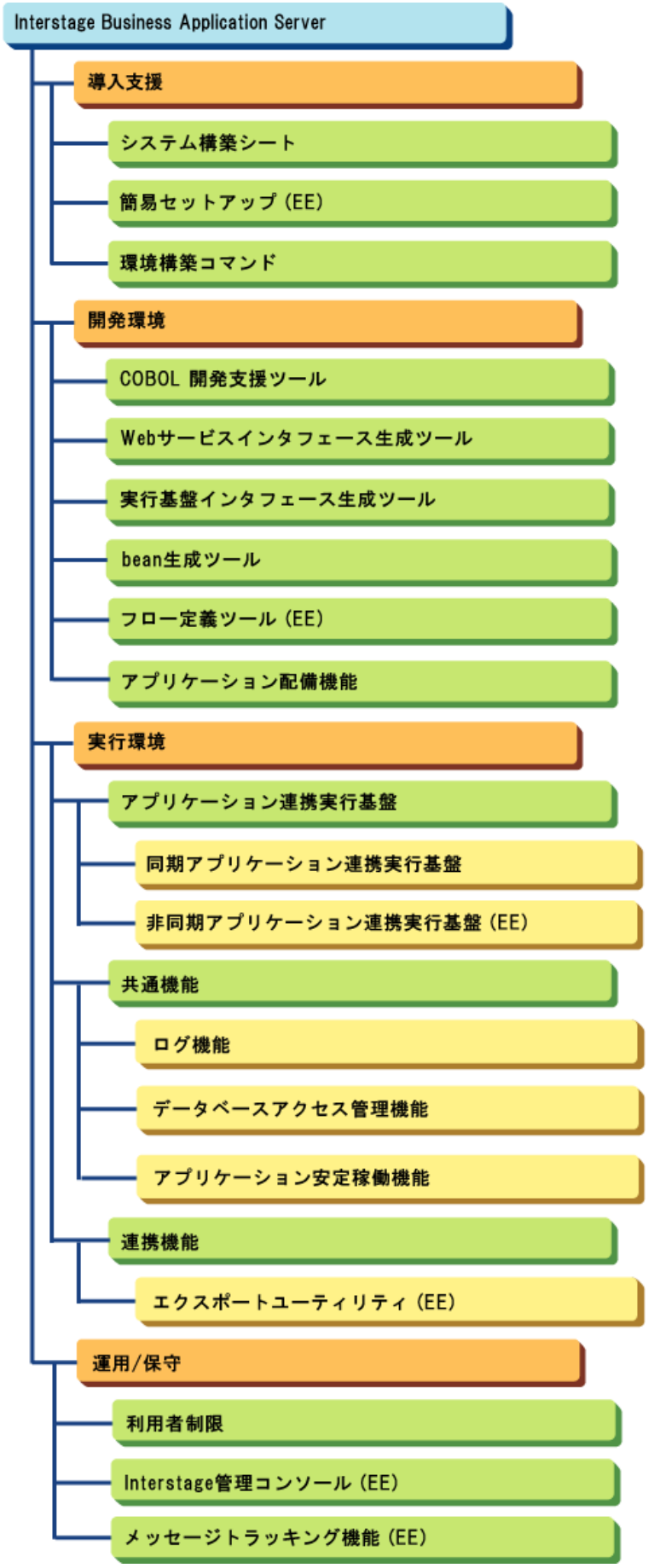
エラーメッセージ処理アプリケーションは、viなどのエディタを使用して単体のJavaのアプリケーションとして作成し、ユーティリティワークユニットに配備して実行します。ユーティリティワークユニットでは、Interstage配下外の一般アプリケーションをトランザクションアプリケーションやEJBアプリケーションと同様に、ワークユニット上で利用することができます。

エラーメッセージ処理アプリケーションは、クライアントからのリクエストにより開始されるのではなく、エラーメッセージ退避キューにメッセージが格納されたのを契機に開始されるため、IJServer、CORBAワークユニットおよびトランザクションアプリケーションとしてではなく、ユーティリティワークユニットで運用します。

ユーティリティワークユニットの作成および運用方法については、“Interstage Application Server OLTPサーバ運用ガイド”を参照してください。

2.7 機能構成

Interstage Business Application Serverの機能構成を説明します。


Interstage Business Application Server
導入支援
システム構築シート
簡易セットアップ (EE)
環境構築コマンド
開発環境
COBOL 開発支援ツール
Webサービスインタフェース生成ツール
実行基盤インタフェース生成ツール
bean生成ツール
フロー定義ツール (EE)
アプリケーション配備機能
実行環境
アプリケーション連携実行基盤
同期アプリケーション連携実行基盤
非同期アプリケーション連携実行基盤 (EE)
共通機能
ログ機能
データベースアクセス管理機能
アプリケーション安定稼働機能
連携機能
エクスポートユーティリティ (EE)
運用/保守
利用者制限
Interstage管理コンソール (EE)
メッセージトラッキング機能 (EE)

注意

図中のInterstage管理コンソールは、非同期アプリケーション連携実行基盤向けに提供しているフロー定義の運用および保守ができる機能について記載しています。
その他のInterstage管理コンソールで提供される機能については、Interstage管理コンソールのヘルプを参照してください。

2.7.1 導入支援

本製品の導入を簡易化し、短時間で環境のセットアップができる機能を提供します。

2.7.1.1 システム構築シート

Interstage Business Application Serverの実行環境作成にあたって発生する各種手順に対し、定義やコマンドの実行パラメタを出力します。本機能により、利用者の手順を明確にし、また各種設定ミスを未然に防止します。
システム構築シートの詳細については、“Interstage Business Application Server セットアップガイド”を参照してください。

2.7.1.2 簡易セットアップ EE

非同期アプリケーション連携実行基盤を使用する場合の環境作成を容易に行うためのセットアップコマンドです。

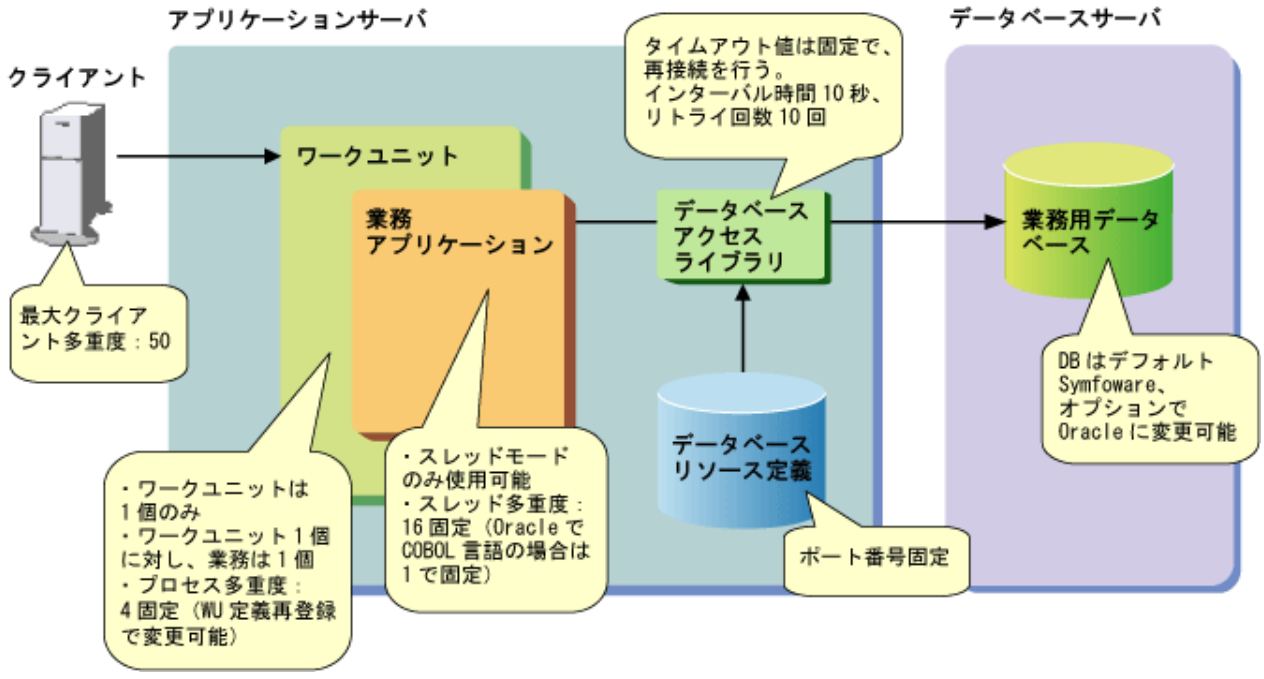
簡易セットアップには、データベースサーバの構築に使用するデータベース環境セットアップコマンドとアプリケーションサーバの構築に使用するアプリケーションサーバ環境セットアップコマンドがあります。

COBOLおよびJavaで共通に利用するデータベース環境の作成、Destination定義の作成、データベースリソース定義の作成およびInterstage Application Serverの設定を行います。

データベースサーバとアプリケーションサーバを別のサーバで運用する場合、データベースサーバにデータベースサーバ機能のインストールを行い、簡易セットアップのデータベース環境セットアップコマンドを実行し、データベースサーバの構築を行います。

2.7.1.3 環境構築コマンド

同期アプリケーション連携実行基盤を使用する場合の基本的な環境作成を行うためのセットアップコマンドです。
C言語およびCOBOLで利用するデータベース環境の作成/登録、ワークユニット定義の作成/登録、データベースリソース定義の作成/登録、CORBAアプリケーション情報定義の作成/登録を行います。
以下に環境構築コマンドを使用した場合に作成される環境を示します。



環境構築コマンドでは、以下の定義値を変更できます。これらの定義値以外を変更する場合は、“2.7.1.1 システム構築シート”を利用してセットアップを行ってください。

- データベース使用の有無
- データベース使用ユーザの設定
- データソース名／データリソース名の変更
- データベースサーバのホスト名の変更
- データベースタイプの“Symfoware”または“Oracle”の選択
- 開発言語の“COBOL”または“C言語”の選択
- 接続先データベース情報の変更（Symfowareの場合、SQLサーバ名。Oracleの場合、ネットサービス名）

上記定義値以外は固定で設定されます。環境構築コマンドで設定される定義値については、“Interstage Business Application Server セットアップガイド”の“環境構築コマンドの設定値”を参照してください。

また、以下の4機能は環境構築コマンドを使用して環境作成できません。これらの機能を使用したい場合は、“2.7.1.1 システム構築シート”を利用してセットアップを行ってください。

- 業務共通制御機能
- 初期処理、終了処理機能
  “Interstage Business Application Server アプリケーション開発ガイド”の“初期処理/終了処理/前処理/後処理/エラー処理/トランザクション後メッセージ編集処理”を参照してください。
- 動作モード
- プレロード・ダイナミックロード

なお、以下の環境設定については、環境構築コマンド実行後チューニング可能です。

- プロセス多重度の変更
- カレントディレクトリの変更
- アプリケーション用の環境変数の変更
- アプリケーション使用ライブラリパスの変更

2.7.2 開発環境

アプリケーション開発を効率化する機能を提供します。

2.7.2.1 COBOL開発支援ツール

アプリケーション連携実行基盤のCOBOLのサーバアプリケーションを動作させるために必要な実行基盤インタフェースおよびインタフェース情報ファイルを生成するためのGUIツールです。
このツールでは、サーバアプリケーションの付加情報、およびサーバアプリケーションで使用するCOBOL登録集を入力することにより、アプリケーション連携実行基盤が利用するファイルを生成します。同期アプリケーション連携実行基盤のC言語クライアントで使用するC言語クライアントソースファイル、およびJavaクライアントアプリケーションインタフェースとして使用するbeanもこのツールで生成します。

2.7.2.2 Webサービスインタフェース生成ツール

同期アプリケーション連携実行基盤のCOBOLのサーバアプリケーションをWebサービスとして呼び出すためのインタフェースを生成するためのGUIツールです。

このツールでは、COBOL開発支援ツールを使用して定義したサーバアプリケーションの付加情報、およびサーバアプリケーションで使用するCOBOL登録集を入力することにより同期アプリケーション連携実行基盤をWebサービスとして呼び出すために必要なファイルを生成します。

2.7.2.3 実行基盤インタフェース生成ツール

アプリケーション連携実行基盤のC言語のサーバアプリケーションを動作させるために必要な実行基盤インタフェースおよびインタフェース情報ファイルを生成するためのツールです。
このツールでは、アプリケーション情報入力ファイルとサーバアプリケーションのインタフェースを定義したIDLファイルの各ファイルを入力することにより、アプリケーション連携実行基盤が利用するファイルを生成します。同期アプリケーション連携実行基盤のC言語クライアントで使用するC言語クライアントソースファイルもこのツールで生成します。
なお、同期アプリケーション連携実行基盤のJavaのクライアントアプリケーションインタフェースとして使用するbeanは、実行基盤インタフェース生成ツールと同じ入力情報を元にbean生成ツールで作成します。

注意

C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

2.7.2.4 bean生成ツール

同期アプリケーション連携実行基盤に配備されたC言語のアプリケーションを呼び出すために、JavaクライアントAPIのためのデータ変換クラスおよび呼び出すサーバアプリケーションのインタフェース情報であるbeanを生成するためのツールです。
このツールでは、アプリケーションインタフェース定義ファイルとアプリケーションのインタフェースを定義したIDLファイルをツールの入力パラメタとして指定することにより、データ変換クラス、bean、アプリケーション呼出し定義ファイルおよびアプリケーションインタフェース定義ファイルを生成します。
C言語のサーバアプリケーションのインタフェースは、実行基盤インタフェース生成ツールで作成します。

ポイント

COBOLのサーバアプリケーションを呼び出すためのJavaクライアントAPIで使用するデータ変換クラス、および呼び出すサーバアプリケーションのインタフェース情報であるbeanはCOBOL開発支援ツールを使用して生成します。

2.7.2.5 フロー定義ツール EE

非同期アプリケーション連携実行基盤を使用する際のアプリケーション連携のパターン(回覧型など業務のシーケンスおよび分岐条件など)や異常処理の方式選択といった設定を定義するツールです。業務データ定義、ルーティング定義、呼出し定義および異常処理定義の4つの定義をGUI上で行います。
また、非同期アプリケーション連携実行基盤のJavaアプリケーション(業務処理開始アプリケーションおよび業務処理実行アプリケーション)は、本ツールのウィザードを利用することで簡単に作成することができます。

2.7.2.6 アプリケーション配備機能

COBOLおよびC言語で作成したサーバアプリケーション(ライブラリ)と各種アプリケーション定義ファイルは、同期アプリケーション連携実行基盤または非同期アプリケーション連携実行基盤に配備して運用します。COBOLおよびC言語のアプリケーションの配備は、本製品のアプリケーションの配備コマンド(apfwdeploy)により行い、配備するときのパラメタは、“2.7.1.1 システム構築シート”で出力されます。
また、Javaアプリケーションの配備は、Interstage Application Serverの配備コマンド(ijsdeployment)で行い、配備するときのパラメタは、“2.7.1.1 システム構築シート”で出力されます。

ポイント

COBOLアプリケーションは、COBOL開発支援ツールの配備機能を利用して配備することもできます。

注意

C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

2.7.3 実行環境

制御ロジックを実行する高信頼かつ高品質な実行環境を提供します。

同期アプリケーション連携実行基盤

詳細は、“第2部 同期アプリケーション連携実行基盤編”を参照してください。

非同期アプリケーション連携実行基盤 EE

詳細は、“第3部 非同期アプリケーション連携実行基盤編”を参照してください。

ログ機能

詳細は、“第4部 共通機能編”の“第5章 ログ機能”を参照してください。

データベースアクセス管理機能

詳細は、“第4部 共通機能編”の“第6章 データベースアクセス管理機能”を参照してください。

アプリケーション安定稼働機能 EE Solaris Linux64

詳細は、“第4部 共通機能編”の“第8章 アプリケーション安定稼動機能”を参照してください。

エクスポートユーティリティ EE

詳細は、“第5部 ユーティリティ編”の“第7章 エクスポートユーティリティ ”を参照してください。

注意

C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

2.7.4 運用保守

運用性を向上し、管理コストを削減する機能を提供します。

2.7.4.1 利用者制限 Solaris Linux32/64

利用者制限機能は、本製品の環境設定や運用を行うユーザを“管理者権限を所有するユーザ”および“Interstage運用グループに所属するユーザ”に限定することで、システムの安全性を向上する機能です。
本機能により、アプリケーション開発に必要な操作を特定のグループに所属するユーザで実行可能となり、管理者権限を所有するユーザとの役割を明確にすることができます。

2.7.4.2 Interstage管理コンソール EE

非同期アプリケーション連携実行基盤向けに、Interstage管理コンソールで運用および保守ができる機能を提供します。非同期アプリケーション連携実行基盤で使用するフローの登録、削除および運用情報の設定を、Interstage管理コンソールから操作することができます。
本製品が使用するInterstage Application Serverの資源も同じWeb画面で確認できるため、各資源の実行状況の把握が容易になり、運用コストを下げることができます。

注意

非同期アプリケーション連携実行基盤を非同期ワークユニットで運用する場合は、Interstege管理コンソールを利用した以下の操作を行うことはできません。Interstage Application Serverが提供するInterstage統合コマンドを利用して運用します。

- 非同期ワークユニットの定義
- 非同期ワークユニットの起動
- 非同期ワークユニットの停止
- 非同期ワークユニットの状態表示

2.7.4.3 メッセージトラッキング機能 EE

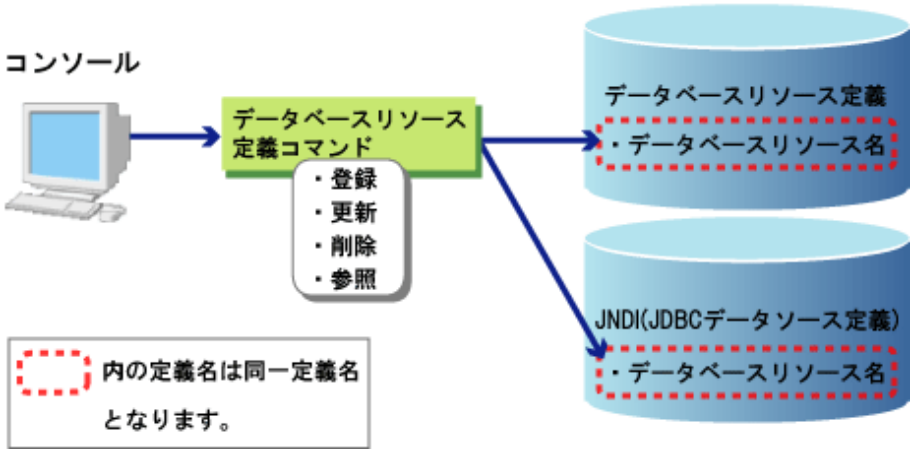
メッセージトラッキング機能は、非同期アプリケーション連携実行基盤において、要求の実行単位であるメッセージ通番(コリレーションID)を元に業務の異常時の状況をリアルタイムに確認する機能です。

2.8 利用するリソース

本製品が提供するリソースについて説明します。

2.8.1 データベースリソース定義

データベースリソース定義は、COBOL、C言語およびJavaで開発したアプリケーションから業務データベースを操作する場合や非同期アプリケーション連携実行基盤がデータベース環境をアクセスする場合に必要となる定義です。
利用者は、開発言語を意識することなく、データベースリソース定義機能を利用して、一貫した定義を行うことができます。
データベースリソース定義機能としてデータベースリソース定義コマンドを提供します。
データベースリソース定義コマンドは、データベースリソース定義およびJDBCデータソース定義に登録、更新、削除および参照するコマンドです。
各コマンドの詳細については、“Interstage Business Application Server リファレンス”を参照してください。



- データベースリソース定義の登録
  データベースリソース定義にデータベースリソース定義情報を登録します。
  また、同時にJDBCデータソース定義の登録も行います。
  なお、すでにデータベースリソース定義またはJDBCデータソース定義に登録済の場合はエラーとなります。
- データベースリソース定義の更新
  データベースリソース定義にデータベースリソース定義情報を上書き登録します。また、同時にJDBCデータソース定義の上書き登録も行います。
  指定された定義がデータベースリソース定義またはJDBCデータソース定義に存在しない場合は、エラーとなります。
- データベースリソース定義の削除
  データベースリソース定義からデータベースリソース定義情報を削除します。
  また、同時にJDBCデータソース定義の削除も行います。
  なお、データベースリソース定義に指定された定義が存在しない場合はエラーとなります。
- データベースリソース定義の参照
  データベースリソース定義の登録状況を参照します。
  なお、データベースリソース定義に存在しない場合は、情報が登録されていない旨を出力します。

注意

- C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。
- データベースリソース定義コマンドで登録したJDBCデータソース定義をInterstage管理コンソールやデータベースリソース定義コマンド以外のコマンドで操作した場合、データベースリソース定義とJDBCデータソース定義との間で不整合が発生します。

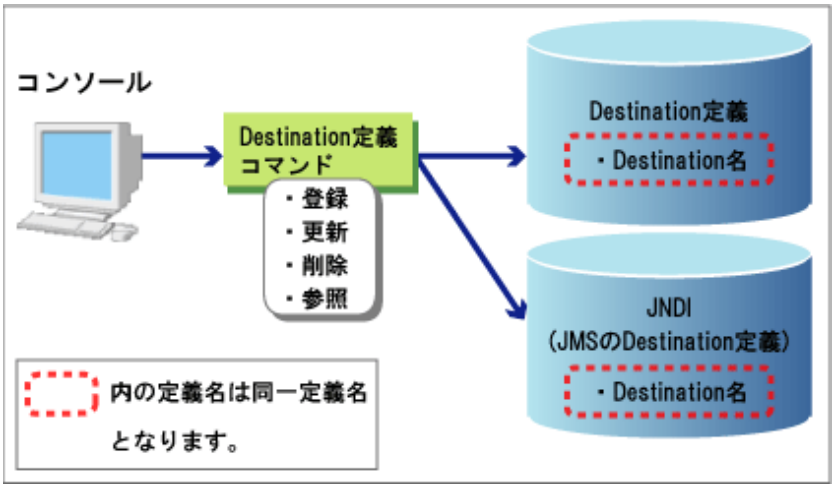
- データベースリソース定義コマンドで登録したJDBCデータソース定義は、データベースリソース定義コマンドで操作してください。

2.8.2 Destination定義 EE

Destination定義機能は、フロー内の各アクティビティに対応する物理キューの定義情報を作成するための機能です。Destination名は、キュー実体を一意に特定するための論理名称です。

キューを特定する場合、COBOLではイベントチャネルグループ名およびイベントチャネル名の組み合わせでキューを特定し、JavaではJMSのDestinationでキューを特定しますが、本製品では、開発言語に依存せず、Destination名という統一的な論理名でアクセスする手段を提供することで利便性を高めます。

本機能により、開発言語に依存せず、1つの定義方法で一貫した定義を行うことができます。
Destination定義機能としてDestination定義コマンドを提供します。Destination定義コマンドは、本製品が管理するDestination定義への登録・更新・削除・参照といった操作を行うコマンドです。各コマンドの詳細については、“Interstage Business Application Server リファレンス”を参照してください。



- Destination定義の登録
  Destination定義にDestination定義情報を登録します。
  また、同時にJMSのDestination定義の登録も行います。
  なお、すでにDestination定義またはJMSのDestination定義に登録済の場合はエラーとなります。

- Destination定義の更新
  Destination定義にDestination定義情報を上書き登録します。
  また、同時にJMSのDestination定義の上書き登録も行います。
  指定された定義がDestination定義またはJMSのDestination定義に存在しない場合は、エラーとなります。

- Destination定義の削除
  Destination定義からDestination定義情報を削除します。
  また、同時にJMSのDestination定義の削除も行います。
  なお、Destination定義に指定された定義が存在しない場合はエラーとなります。

- Destination定義の参照
  Destination定義の登録状況を参照します。
  なお、Destination定義に存在しない場合は、情報が登録されていない旨を出力します。

注意

- Destination定義コマンドで登録したJMS Destination定義をInterstage管理コンソールやDestination定義コマンド以外のコマンドで操作した場合、Destination定義とJMS Destination定義との間で不整合が発生します。

- Destination定義コマンドで登録したJMS Destination定義は、Destination定義コマンドで操作してください。

2.9 文字コード

アプリケーション連携実行基盤では、以下の文字コードのユーザアプリケーションをサポートします。

■クライアントアプリケーションで利用可能な文字コード

種別	クライアントアプリケーションの言語	利用可能な文字コード
同期アプリケーション連携実行基盤	Java	UTF-16
	C言語	Solaris EUC(S90) Shift_JIS UTF-8 Linux32/64 UTF-8 Windows32/64 Shift_JIS UTF-8
非同期アプリケーション連携実行基盤	Java	UTF-16

■サーバアプリケーションで利用可能な文字コード

種別	サーバアプリケーションの言語	利用可能な文字コード
同期アプリケーション連携実行基盤	COBOL C言語	Solaris EUC(S90) Shift_JIS UTF-8 Linux32/64 UTF-8 (推奨) EUC(S90)(RHEL5のみ) Windows32/64 Shift_JIS UTF-8
非同期アプリケーション連携実行基盤	COBOL	Solaris EUC(S90) Shift_JIS UTF-8 Linux64 UTF-8 (推奨) EUC(S90)(RHEL5のみ) Windows64 Shift_JIS UTF-8
	Java	UTF-16

注意

JIS X 0213:2004(JIS2004)で追加された文字を利用する場合、UTF-8を使用してください。

Linux32/64

システムのロケールについて以下の注意事項があります。

注意

ja_JP.eucJPは、システムとして推奨されておらず、動作保証されない可能性があります。新規にシステム構築する場合には、ja_JP.UTF-8で実施することをお勧めします。

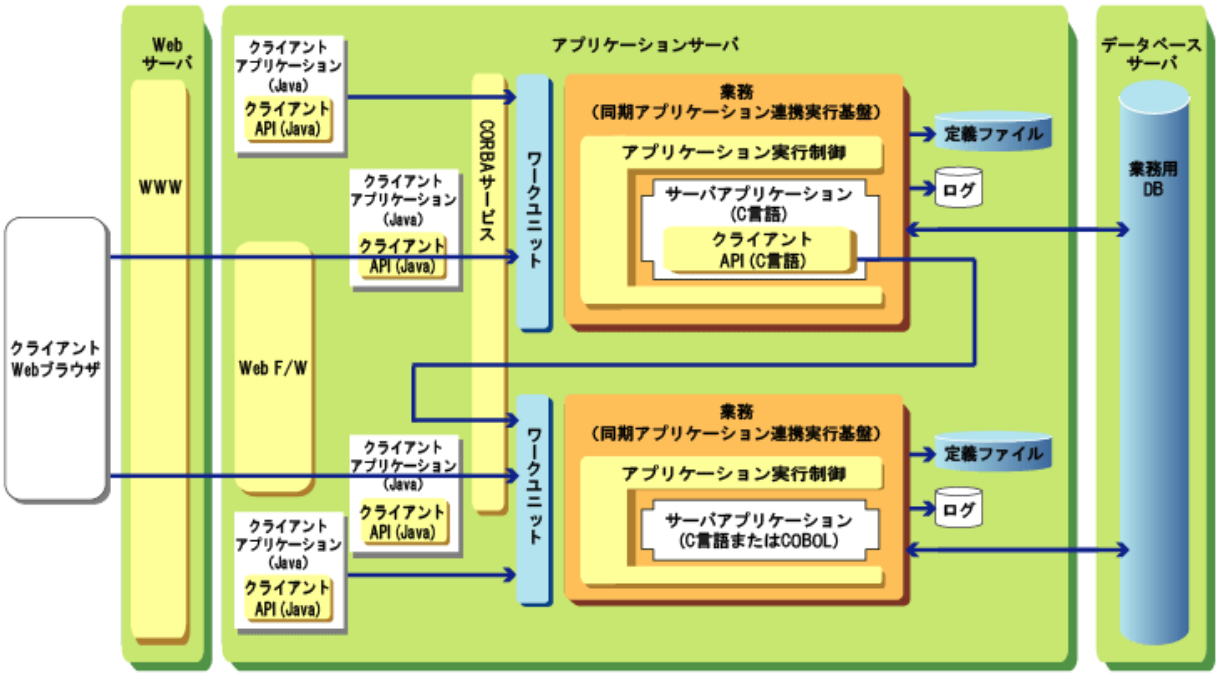
第2部 同期アプリケーション連携実行基盤編

第3章 同期アプリケーション連携実行基盤の機能

同期アプリケーション連携実行基盤の機能について説明します。

3.1 構成要素

本節では、同期アプリケーション連携実行基盤の構成要素について説明します。
同期アプリケーション連携実行基盤で提供するオンライン処理(即時応答型)は、クライアントからの処理要求に対して、リアルタイムに処理結果を応答する処理形態です。
同期アプリケーション連携実行基盤は、以下のように、処理の実行を要求するクライアントアプリケーションと要求された処理を実行する業務から構成します。これらは、同一のサーバ上で動作します。以下に、同期アプリケーション連携実行基盤で提供する機能について説明します。



■同期アプリケーション連携実行基盤の構成

- 同期アプリケーション連携実行基盤

  同期アプリケーション連携実行基盤では、アプリケーションの呼び出しを制御する機能とアプリケーションの開発、運用をサポートする以下の機能を提供します。これにより、アプリケーションのコンポーネント化を実現し、柔軟性、拡張性に富む業務構築を実現します。また、プログラミングの標準化が行え、多数の開発者での、一定品質のアプリケーションの並行開発が可能となります。サーバアプリケーションの開発は、COBOLおよびC言語に対応します。同期アプリケーション連携実行基盤のアプリケーション実行制御では、次の制御を行います。

  ― 初期処理／終了処理／前処理／後処理／エラー処理／トランザクション後メッセージ編集処理

  ― トランザクションとリトライ制御

  ― COBOLまたはC言語のアプリケーション呼出し

  ― 業務共通制御の呼出し

  ― COBOL実行環境の開設・閉鎖

  ― ライブラリのロード

- クライアントAPI

  クライアントをWebアプリケーションとして作成するためのJava用のクライアントAPIと、サーバアプリケーションからさらに別のサーバアプリケーションを呼び出すためのC言語用のクライアントAPIを提供します。
  Javaクライアントアプリケーションの形態は、サーブレット、JavaServer Pages(JSP)ベースのアプリケーションなどです。
  C言語クライアントアプリケーションの形態は、C言語のサーバアプリケーションとの連携、COBOLのサーバアプリケーションとの連携などです。
  各アプリケーションから同期アプリケーション連携実行基盤を呼出すAPIとして、次のAPIを用意します。

  - 一般的なJavaアプリケーションからの呼び出し
  - Apcoordinatorのリモート共通インタフェースによる呼び出し
  - J2EE Connector ArchitectureのCommon Client Interface(CCI)による呼び出し
  - C言語アプリケーションからの呼び出し

- ログ機能

  アプリケーションの処理状況や、万一の異常発生時のトラブル状況を出力します。また、アプリケーション呼び出しから応答までの性能情報を業務単位で出力します。これらアプリケーション連携実行基盤が自動で出力するログを用いることにより、状況を正確に把握し、問題の検証やボトルネックの早期検出および改善に利用することができます。ログの出力内容などログの詳細は、“第5章 ログ機能”を参照してください。

  - システムログ
  - 性能ログ

- ワークユニット

  同期アプリケーション連携実行基盤は、CORBAワークユニット上で動作します。その為、CORBAワークユニットで提供される高信頼な各サービスをそのまま利用することが可能です。

  - 複数クライアントからのプロセス共有機能
  - アプリケーションの非常駐化(アプリケーション連携実行基盤の非常駐化)
  - プロセスの多重化
  - 実行時のスナップショット
  - 性能情報の測定(性能監視ツール)

■同期アプリケーション連携実行基盤におけるサーバ配置

アプリケーション連携実行基盤で利用するサーバの種類を示します。これらのサーバは、スケーラブルに拡張することが可能です。

- アプリケーションサーバ

  アプリケーションサーバは、クライアントアプリケーションおよびサーバアプリケーションといった利用者のアプリケーション、アプリケーション連携実行基盤の実行環境およびワークユニットを運用するサーバです。

- データベースサーバ

  データベースサーバは、ユーザの業務データを格納するサーバです。データベースサーバには、以下のいずれかのデータベースが必要です。

  - Symfoware ServerまたはOracle

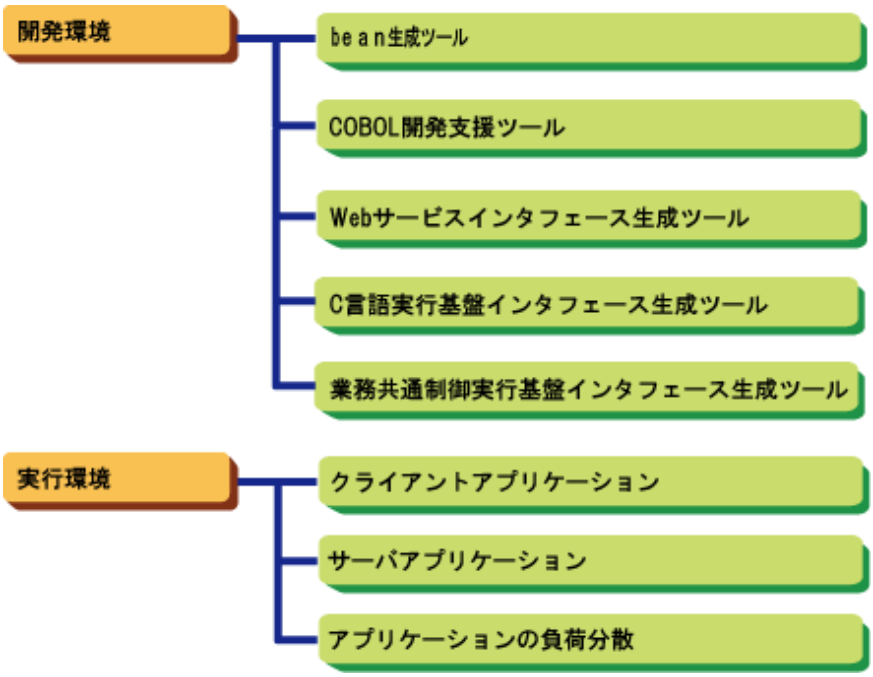
    ユーザの業務データを格納するデータベース製品です。本製品には同梱されていませんので別途購入が必要です。

- Webサーバ

  Webブラウザからの要求を契機に業務処理を開始するシステムを構築する場合、Webサーバを利用する必要があります。

3.2 機能構成

同期アプリケーション連携実行基盤は、開発環境および実行環境の2つの機能から構成されています。各機能について以降の節で説明します。



3.3 開発環境

クライアントアプリケーションの開発は、Interstageの統合開発環境であるInterstage Studioを用いてbean生成ツールが出力したソースを元に開発を行います。また、サーバアプリケーションの開発は、COBOL開発支援ツールまたはC言語実行基盤インタフェース生成ツールを使用して行います。
クライアントアプリケーションおよびサーバアプリケーションを開発する場合に必要となる各資源の概要を以下に説明します。
各資源の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”を参照してください。

資源名	説明	クライアント アプリケーション		サーバ アプリケーション		Webサービス インタフェース
		Java	C言語	COBOL	C言語	
COBOL登録集	COBOLのサーバアプリケーションのインタフェースを定義したファイルです。COBOLの実行基盤インタフェースを生成する場合に使用し、利用者が作成します。	○（注1）	○（注1）	○	－	○
IDLファイル	C言語のサーバアプリケーションのインタフェースを定義したファイルです。C言語実行基盤インタフェースを作成するために使用し、利用者が作成します。	○（注2）	○（注2）	－	○	－
アプリケーション情報入力ファイル	サーバアプリケーションの付加情報を定義するファイルです。実行基盤インタフェースおよび同期アプリケーション連携実行基盤のクライアント用Java beanを生成する場合に使用し、利用者が作成します。	○（注2）	○（注2）	－	○	－
COBOL実行基盤インタフェース	アプリケーション連携実行基盤とCOBOLのサーバアプリケーションを繋ぐためのインタフェースです。サーバアプリケーションで利用する言語に合わせたデータ型への変換、アプリケーション連携実行基盤	－	－	○	－	－

資源名	説明	クライアント アプリケーション		サーバ アプリケーション		Webサービス インタフェース
		Java	C言語	COBOL	C言語	
	からサーバアプリケーションへの受け渡しを行うデータ領域の獲得および解放を行います。このインタフェースは、COBOL開発支援ツールで生成します。					
C言語実行基盤インタフェース	アプリケーション連携実行基盤とC言語のサーバアプリケーションを繋ぐためのインタフェースです。サーバアプリケーションで利用する言語に合わせたデータ型への変換、アプリケーション連携実行基盤からサーバアプリケーションへの受け渡しを行うデータ領域の獲得および解放を行います。このインタフェースは、実行基盤インタフェース生成ツールで作成します。	－	－	－	○	－
アプリケーションインタフェース定義ファイル	サーバアプリケーションの業務処理名、パラメタ名および型などのインタフェース情報を定義したファイルです。COBOLまたはC言語のサーバアプリケーションを配備する場合に指定します。指定したファイルは、アプリケーション連携実行基盤の動作時に読み込まれます。このファイルは、COBOL開発支援ツール、または実行基盤インタフェース生成ツールで作成します。	－	－	○	○	－
アプリケーション呼出し定義ファイル	サーバアプリケーション名、業務処理名およびユーザ作成ライブラリ名を定義したファイルです。サーバアプリケーションを配備する場合に指定します。指定したファイルは、アプリケーション連携実行基盤の動作時に読み込まれます。このファイルは、COBOL開発支援ツール、または実行基盤インタフェース生成ツールで作成します。	－	－	○	○	－
Java beanクラスのソース	IDLファイルまたはCOBOL登録集と、アプリケーション情報入力ファイルに定義された内容から生成される以下のJavaソースファイルです。 beanのJavaソースファイル ユーザ定義型クラスのJavaソースファイル データ変換クラスのJavaソースファイル これらのファイルは、bean生成ツール、またはCOBOL開発支援ツールで作成します。	○	－	－	－	○

資源名	説明	クライアント アプリケーション		サーバ アプリケーション		Webサービス インタフェース
		Java	C言語	COBOL	C言語	
C言語クライアントソース	IDLファイルまたはCOBOL登録集と、アプリケーション情報入力ファイルに定義された内容から生成されるデータ変換関数のヘッダファイルおよびソースファイルです。このファイルは、実行基盤インタフェース生成ツールで作成します。	ー	○	ー	ー	ー
サーバアプリケーション(業務ロジック)	サーバで実行する業務用データベースのアクセスなどの業務ロジックです。利用者が、各言語で開発します。	ー	ー	○	○	ー
Javaクライアントアプリケーションソース	サーバを呼び出すためのクライアントの処理です。利用者が、Javaで開発します。	○	ー	ー	ー	ー
C言語クライアントアプリケーションソース	サーバを呼び出すためのクライアントの処理です。利用者が、C言語で開発します。	ー	○	ー	ー	ー
Webサービスアプリケーションソース	サーバをWebサービスとして呼び出す処理です。 このファイルは、COBOL開発支援ツールで作成します。	ー	ー	ー	ー	○

○:必要な資源であることを示します。 ー:必要な資源ではありません。

注1)COBOLのサーバアプリケーションと連携する場合に必要です。
注2)C言語のサーバアプリケーションと連携する場合に必要です。

業務共通制御の開発は、業務共通制御実行基盤インタフェース生成ツールを使用して行います。
業務共通制御を開発する場合に必要となる各資源の概要を以下に説明します。
各資源の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”を参照してください。

資源名	説明	クライアント アプリケーション		業務共通制御
		Java	C言語	
業務共通制御IDLファイル	業務共通制御のインタフェースを定義したファイルです。業務共通制御実行基盤インタフェースを作成するために使用し、利用者が作成します。	○	○	○
業務共通制御情報入力ファイル	業務共通制御の付加情報を定義するファイルです。業務共通制御実行基盤インタフェースおよび業務共通制御用Java beanを生成する場合に使用し、利用者が作成します。	○	○	○
業務共通制御実行基盤インタフェース	アプリケーション連携実行基盤と業務共通制御を繋ぐためのインタフェースです。業務共通制御のインタフェースに合わせたデータ型への変換、アプリケーション連携実行基盤から業務共通制御への受け渡しを行うデータ領域の獲得および解放を行います。このインタフェー	ー	ー	○

資源名	説明	クライアント アプリケーション		業務共通制御
		Java	C言語	
	スは、業務共通制御実行基盤インタフェース生成ツールで作成します。			
業務共通制御アプリケーションインタフェース定義ファイル	業務共通制御のインタフェース情報を定義したファイルです。業務共通制御を配備する場合に指定します。指定したファイルは、アプリケーション連携実行基盤の動作時に読み込まれます。このファイルは、業務共通制御実行基盤インタフェース生成ツールで作成します。	－	－	○
制御アプリケーション定義ファイル	業務共通制御に対応する関数名を定義したファイルです。業務共通制御を配備する場合に指定します。指定したファイルは、アプリケーション連携実行基盤の動作時に読み込まれます。このファイルは、業務共通制御実行基盤インタフェース生成ツールで作成します。	－	－	○
業務共通制御用のJava beanクラスのソース	業務共通制御IDLファイルと業務共通制御情報入力ファイルに定義された内容から生成される以下のJavaソースファイルです。 制御データ用beanのJavaソースファイル 制御データ用ユーザ定義型クラスのJavaソースファイル 制御データ用データ変換クラスのJavaソースファイル これらのファイルは、業務共通制御実行基盤インタフェース生成ツールで作成します。	○	－	－
業務共通制御用のC言語クライアントソース	業務共通制御IDLファイルと業務共通制御情報入力ファイルに定義された内容から生成される制御データ用データ変換関数のC言語ヘッダファイルおよびソースファイルです。これらのファイルは、業務共通制御実行基盤インタフェース生成ツールで作成します。	－	○	－
制御ロジック	サーバで実行する制御ロジックです。利用者が、C言語で開発します。	－	－	○
Javaクライアントアプリケーションソース	サーバで業務共通制御を呼び出すためのクライアントの処理です。利用者が、Javaで開発します。	○	－	－
C言語クライアントアプリケーションソース	サーバで業務共通制御を呼び出すためのクライアントの処理です。利用者が、C言語で開発します。	－	○	－

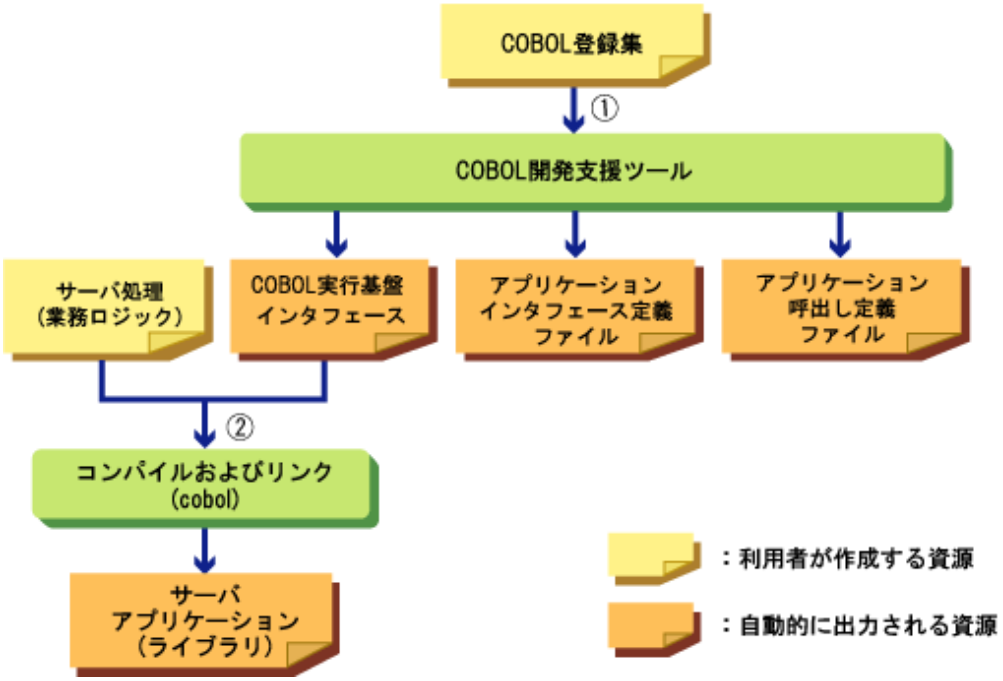
○：必要な資源であることを示します。 －：必要な資源ではありません。

■COBOLのサーバアプリケーションを利用する場合

COBOLのサーバアプリケーションを開発する場合のクライアントアプリケーションとサーバアプリケーションの開発作業の流れと開発する資源を以下に示します。

サーバアプリケーションの開発作業の流れ

COBOLのサーバアプリケーション作成の流れ



注意

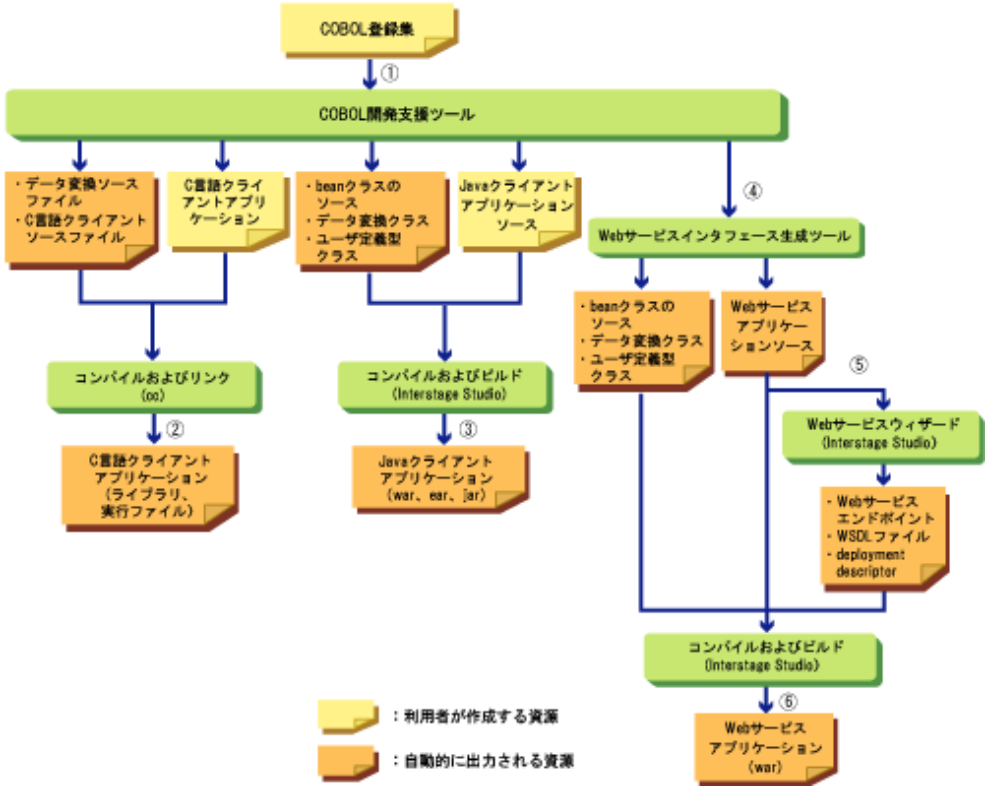
図はクライアントがSolarisおよびLinuxの場合の例です。
クライアントがWindowsの場合は、C言語クライアントのコンパイルおよびリンクのツールとして、Visual Studio 2005またはVisual Studio 2008を使用します。

[図の説明]

1. COBOL登録集を入力にし、COBOL開発支援ツールを使用してCOBOL実行基盤インタフェースを生成します。
2. サーバアプリケーションは、COBOL開発支援ツールによって生成されたCOBOL実行基盤インタフェースと利用者が作成したサーバの業務ロジックを合わせてコンパイルおよびリンクして作成します。

クライアントアプリケーションの開発作業の流れ

クライアントアプリケーション作成の流れ



注意

図はクライアントがSolarisおよびLinuxの場合の例です。
クライアントがWindowsの場合は、C言語クライアントのコンパイルおよびリンクのツールとして、Visual Studio 2005またはVisual Studio 2008を使用します。

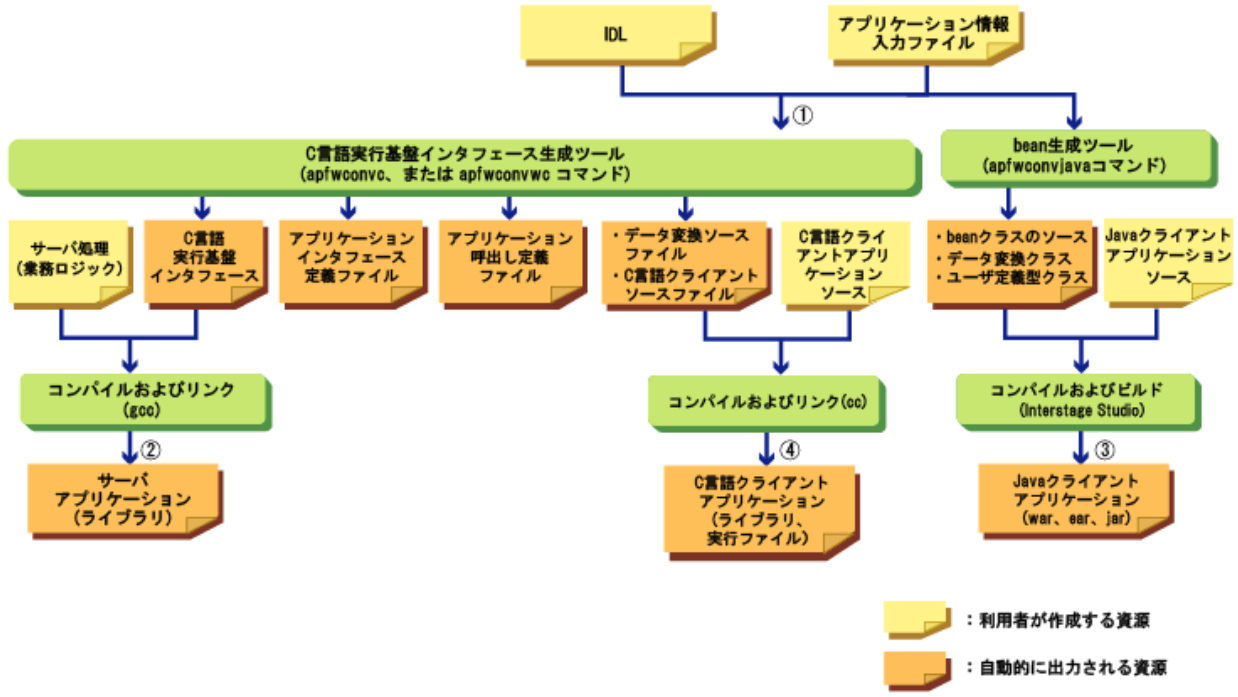
[図の説明]

1. COBOL登録集を入力にし、COBOL開発支援ツールを使用してCOBOL実行基盤インタフェースを生成します。
2. Javaクライアントアプリケーションは、COBOL開発支援ツールによってCOBOL実行基盤インタフェースと同時に生成されたJava beanクラスのソースと利用者が作成したクライアントソースをInterstage Studioでコンパイルおよびビルドして作成します。
3. C言語クライアントアプリケーションは、COBOL開発支援ツールによってCOBOL実行基盤インタフェースと同時に生成されたC言語クライアントソースファイルと利用者が作成したクライアントソースをコンパイルおよびリンクして作成します。
4. COBOL開発支援ツールを使用して登録したサーバアプリケーションの情報を入力にし、Webサービスインタフェース生成ツールを使用してWebサービスアプリケーションソースを生成します。
5. Webサービスインタフェース生成ツールによって生成されたWebアプリケーションソースを入力にし、Webサービスウィザードを使用してWebサービスエンドポイントなどのWebサービスを作成するために必要なファイルを生成します。
6. Webサービスアプリケーションは、Webサービスインタフェース生成ツールによって生成されたWebサービスアプリケーションソース、およびJava beanクラスとWebサービスウィザードによって生成されたファイルをInterstage Studioでコンパイルおよびビルドして作成します。

■C言語のサーバアプリケーションを利用する場合

C言語のサーバアプリケーションを開発する場合のクライアントアプリケーションとサーバアプリケーションの開発作業の流れと開発する資源を以下に示します。

C言語のサーバアプリケーション作成の流れ



注意

図はサーバおよびクライアントがSolarisおよびLinuxの場合の例です。
サーバおよびクライアントがWindowsの場合、C言語実行基盤インタフェース生成ツールとして、“apfwconvwc”を使用します。コンパイルおよびリンクのツールとして、Visual Studio 2005またはVisual Studio 2008を使用します。

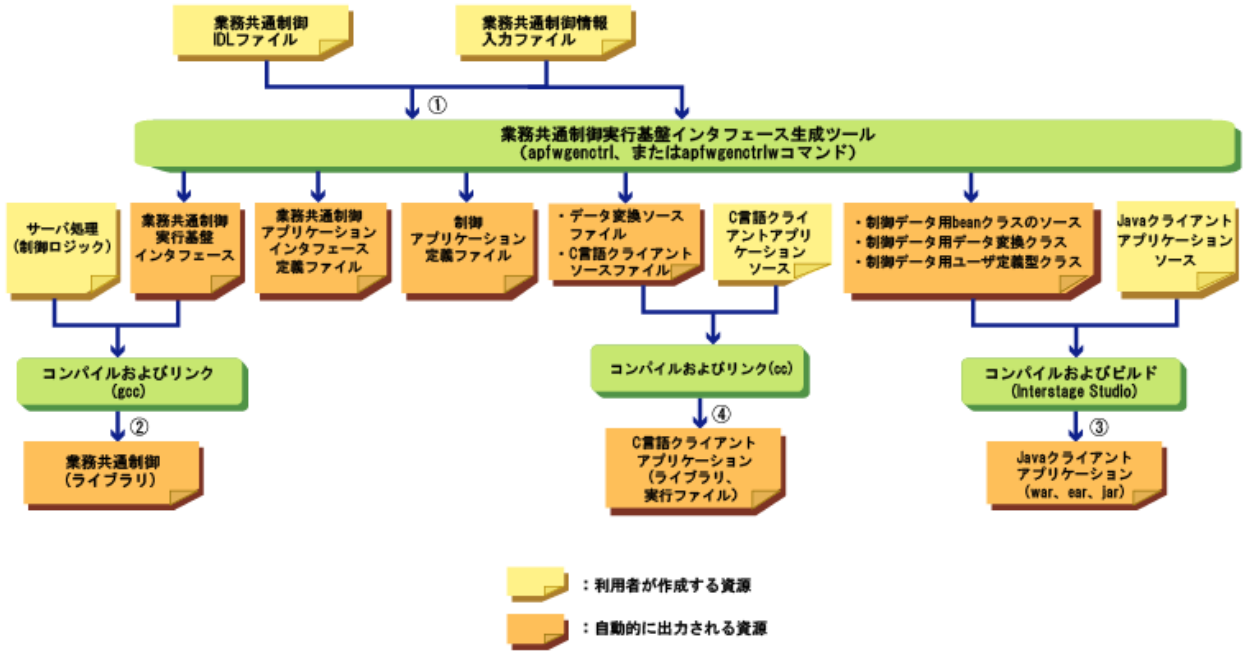
[図の説明]

1. C言語のアプリケーションのインタフェースを記述したIDLおよびアプリケーション情報入力ファイルを入力にし、C言語実行基盤インタフェース生成ツールを実行します。
2. サーバアプリケーションは、C言語実行基盤インタフェース生成ツールが出力したC言語実行基盤インタフェース、アプリケーションインタフェース定義ファイルおよびアプリケーション呼出し定義ファイルに利用者が作成したサーバの業務ロジックを元にコンパイルおよびリンクして作成します。
3. Javaクライアントアプリケーションは、bean生成ツールが出力したJava beanクラスのソースと利用者が作成したクライアントソースをInterstage Studioでコンパイルおよびビルドして作成します。
4. C言語クライアントアプリケーションは、C言語実行基盤インタフェース生成ツールが出力したC言語クライアントソースファイルを元に利用者が作成したクライアントをコンパイルおよびリンクして作成します。

■業務共通制御を利用する場合

業務共通制御を開発する場合のクライアントアプリケーションとサーバアプリケーションの開発作業の流れと開発する資源を以下に示します。

業務共通制御の作成の流れ



注意

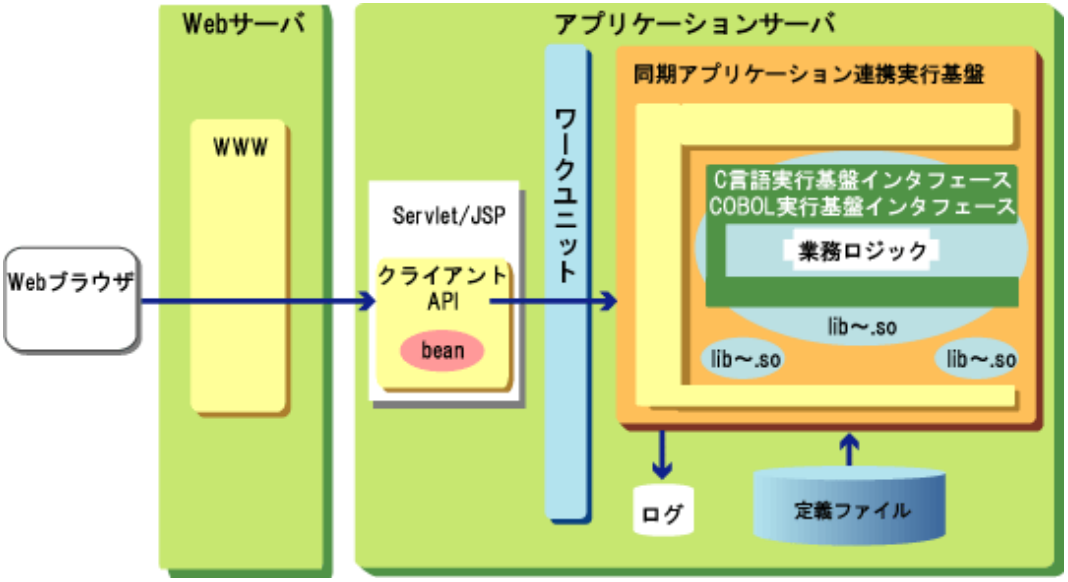
図はサーバおよびクライアントがSolarisおよびLinuxの場合の例です。
サーバおよびクライアントがWindowsの場合、C言語実行基盤インタフェース生成ツールとして、“apfwgenctrlw”を使用します。コンパイルおよびリンクのツールとして、Visual Studio 2005またはVisual Studio 2008を使用します。

[図の説明]

1. 業務共通制御のインタフェースを記述したIDLおよび業務共通制御情報入力ファイルを入力にし、業務共通制御実行基盤インタフェース生成ツールを実行します。
2. 業務共通制御は、業務共通制御実行基盤インタフェース生成ツールが出力した業務共通制御実行基盤インタフェース、業務共通制御アプリケーションインタフェース定義ファイルおよび制御アプリケーション定義ファイルに利用者が作成した制御ロジックを元にコンパイルおよびリンクして作成します。
3. Javaクライアントアプリケーションは、業務共通制御実行基盤インタフェース生成ツールが出力したJava beanクラスのソースと利用者が作成したクライアントソースをInterstage Studioでコンパイルおよびビルドして作成します。
4. C言語クライアントアプリケーションは、業務共通制御実行基盤インタフェース生成ツールが出力したC言語クライアントソースファイルを元に利用者が作成したクライアントをコンパイルおよびリンクして作成します。

3.3.1 bean生成ツール

同期アプリケーション連携実行基盤に配備されたC言語の業務アプリケーションを呼び出すために、Javaクライアントランタイム(以降、クライアントAPI)とサーバアプリケーションを繋ぐためのデータ変換クラス、および呼び出すサーバアプリケーションのインタフェース情報であるbeanを生成するためのツールです。業務アプリケーションがCOBOLの場合は、COBOL開発支援ツールを使用してbeanを生成します。
bean生成ツールでは、業務アプリケーションのインタフェースを定義したIDLファイルと付加情報を定義したアプリケーション情報入力ファイルをツールの入力パラメタとして指定することにより、データ変換クラス、ユーザ定義型クラスおよびbeanを生成します。
作成したJavaロジック、ツールにより生成された、データ変換クラス、ユーザ定義型クラスおよびbeanをJavaコンパイルすることにより、同期アプリケーション連携実行基盤に配備されたCOBOLおよびC言語の業務アプリケーションを呼び出すことが可能になります。
beanの位置づけを以下に示します。



注意

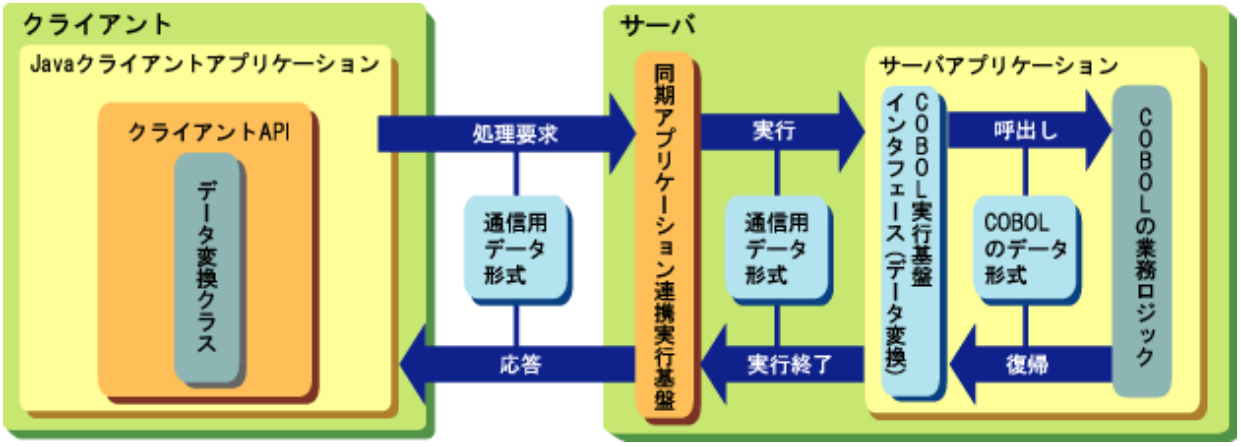
図はサーバがSolarisおよびLinuxの場合の例です。

bean生成ツールを使用したアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“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。

3.3.2 COBOL開発支援ツール

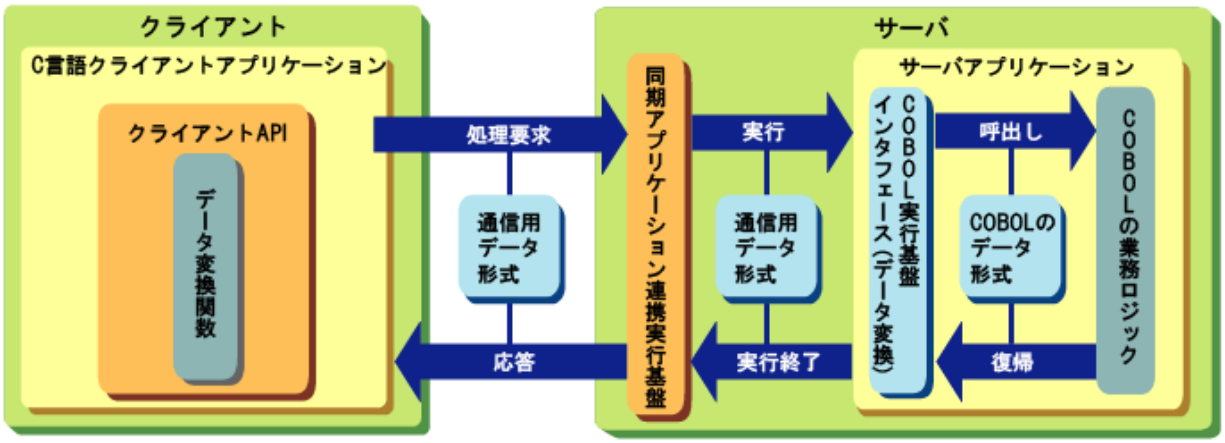
COBOL開発支援ツールは、アプリケーション連携実行基盤と業務ロジックをつなぐために必要なファイルを生成するツールです。

このツールでは、サーバアプリケーションの付加情報とサーバアプリケーションが使用するCOBOL登録集を入力することにより、COBOL実行基盤インタフェース、アプリケーションインタフェース情報ファイルおよびアプリケーション呼出し定義ファイルを生成します。
次に、ユーザ作成の業務ロジックとツールにより生成されたCOBOL実行基盤インタフェースをコンパイルおよびリンクしてライブラリを作成します。アプリケーション呼出し定義ファイルおよびアプリケーションインタフェース定義と共にアプリケーション連携実行基盤上に配備することで、アプリケーション連携実行基盤から業務ロジックを実装したサーバアプリケーションが実行可能になります。
COBOL実行基盤インタフェースの位置づけを以下に示します。



COBOL開発支援ツールを使用したアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の “サーバアプリケーションの開発(COBOL)”を参照してください。

C言語クライアントアプリケーションを開発する場合、COBOL開発支援ツールは、同期アプリケーション連携実行基盤に配備されたCOBOLの業務アプリケーションをC言語クライアントアプリケーションから呼び出すためのデータ変換関数も生成します。



C言語クライアントアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“クライアントアプリケーションの開発(C言語)”を参照してください。

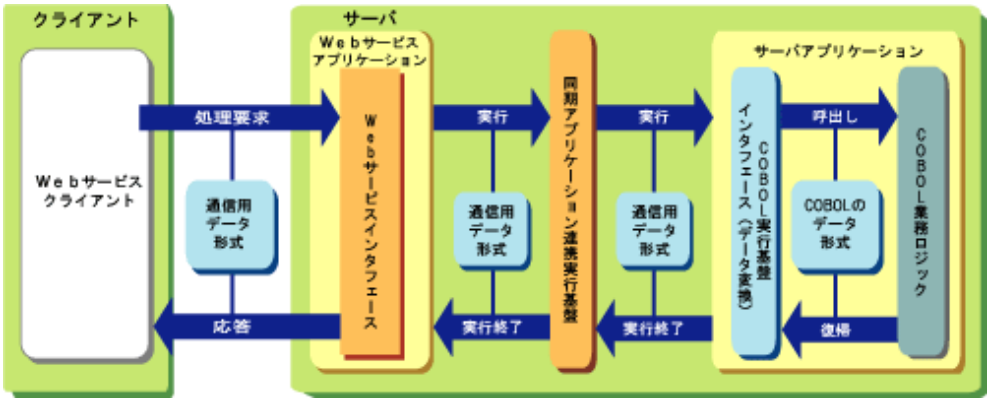
3.3.3 Webサービスインタフェース生成ツール

Webサービスインタフェース生成ツールは、アプリケーション連携実行基盤をWebサービスとして呼び出すために必要なファイルを生成する機能です。

このWebサービスインタフェース生成機能は、COBOL開発支援ツールで定義したサーバアプリケーションの付加情報とサーバアプリケーションが使用するCOBOL登録集を入力することにより、Webサービスインタフェースを生成します。

生成されたWebサービスインタフェースをJavaコンパイルしWebサービスアプリケーションを作成することで、同期アプリケーション連携実行基盤上に配備されたCOBOLの業務アプリケーションをWebサービスとして呼び出すことが可能になります。

Webサービスインタフェースの位置づけを以下に示します。

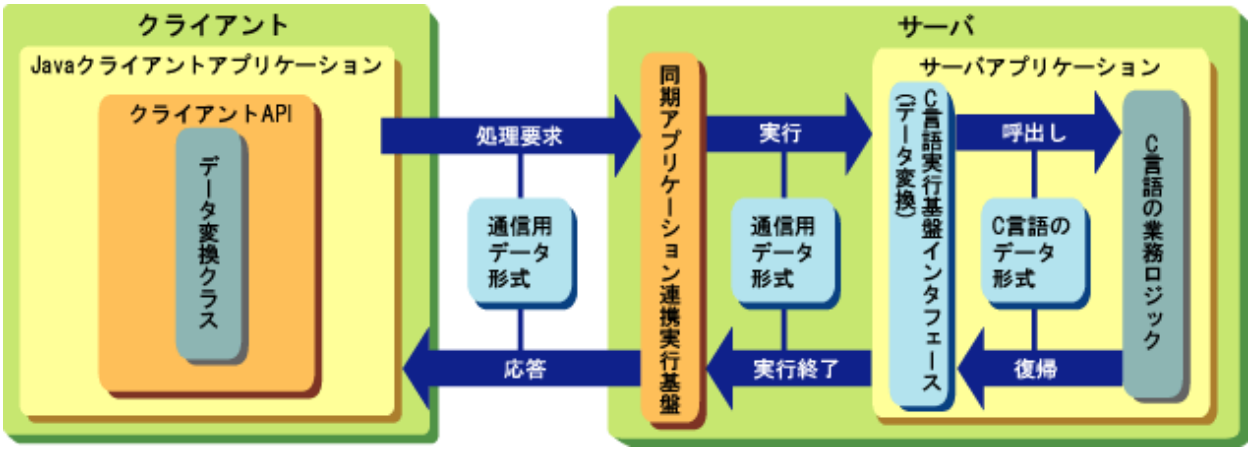


Webサービスインタフェース生成ツールを使用したWebサービスインタフェースの生成の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“Webサービスアプリケーションの開発”を参照してください。

3.3.4 C言語実行基盤インタフェース生成ツール

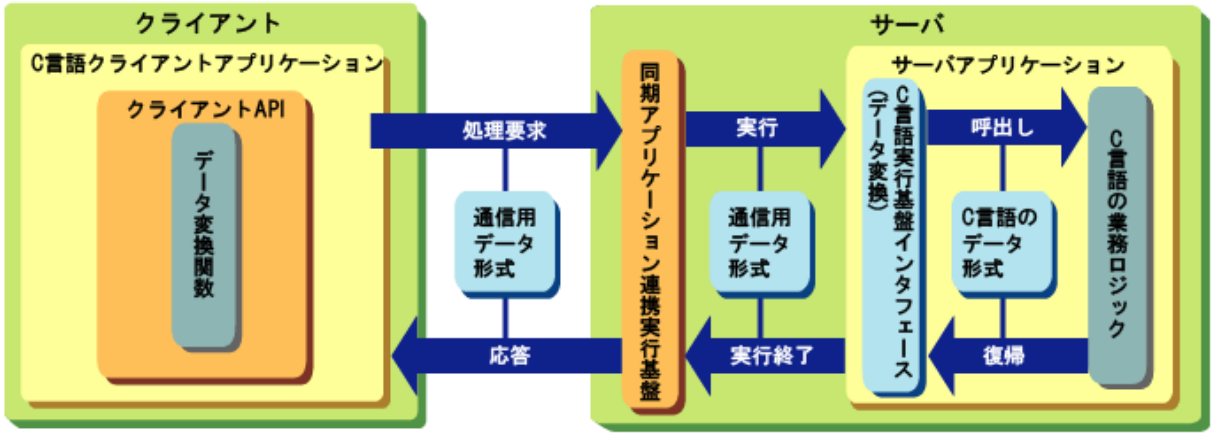
C言語のサーバアプリケーションをアプリケーション連携実行基盤上で動作させるために、アプリケーション連携実行基盤と業務ロジックを繋ぐためのC言語実行基盤インタフェースおよびアプリケーション連携実行基盤上で管理するインタフェース情報ファイルを生成するためのツールです。
このツールでは、サーバアプリケーションのインタフェースを定義したIDLファイルおよび付加情報を定義したアプリケーション情報入力ファイルをツールの入力パラメタとして指定することにより、C言語実行基盤インタフェース、アプリケーション呼出し定義ファイルおよびアプリケーションインタフェース定義ファイルを生成します。
ユーザ作成の業務ロジックとツールにより生成されたC言語実行基盤インタフェースをコンパイル／リンクしてライブラリを作成し、アプリケーション呼出し定義ファイルおよびアプリケーションインタフェース定義と共にアプリケーション連携実行基盤上に配備することで、

アプリケーション連携実行基盤から業務ロジックを実装したサーバアプリケーションが実行可能になります。
C言語実行基盤インタフェースの位置づけを以下に示します。



C言語実行基盤インタフェース生成ツールを使用したアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“サーバアプリケーションの開発(C言語)”を参照してください。

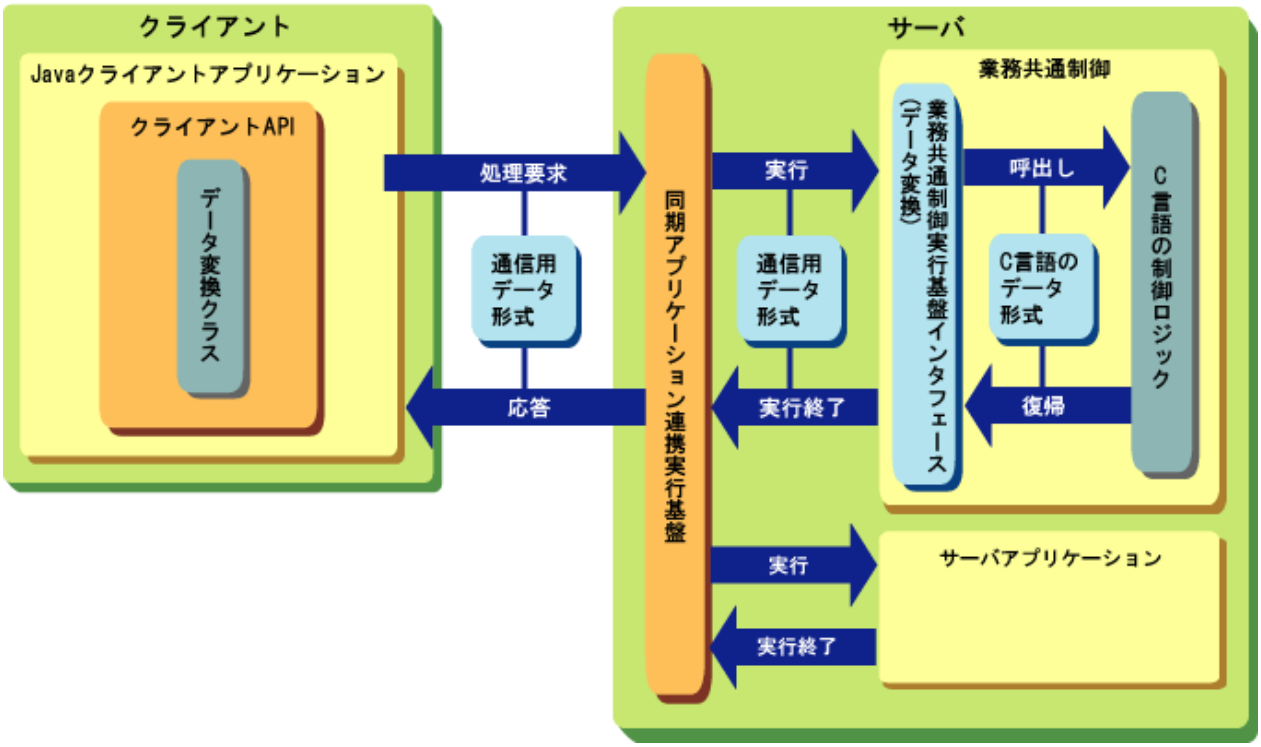
C言語クライアントアプリケーションを開発する場合、C言語実行基盤インタフェース生成ツールは、同期アプリケーション連携実行基盤に配備されたC言語の業務アプリケーションをC言語クライアントアプリケーションから呼び出すためのデータ変換関数も生成します。



C言語クライアントアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“クライアントアプリケーションの開発(C言語)”を参照してください。

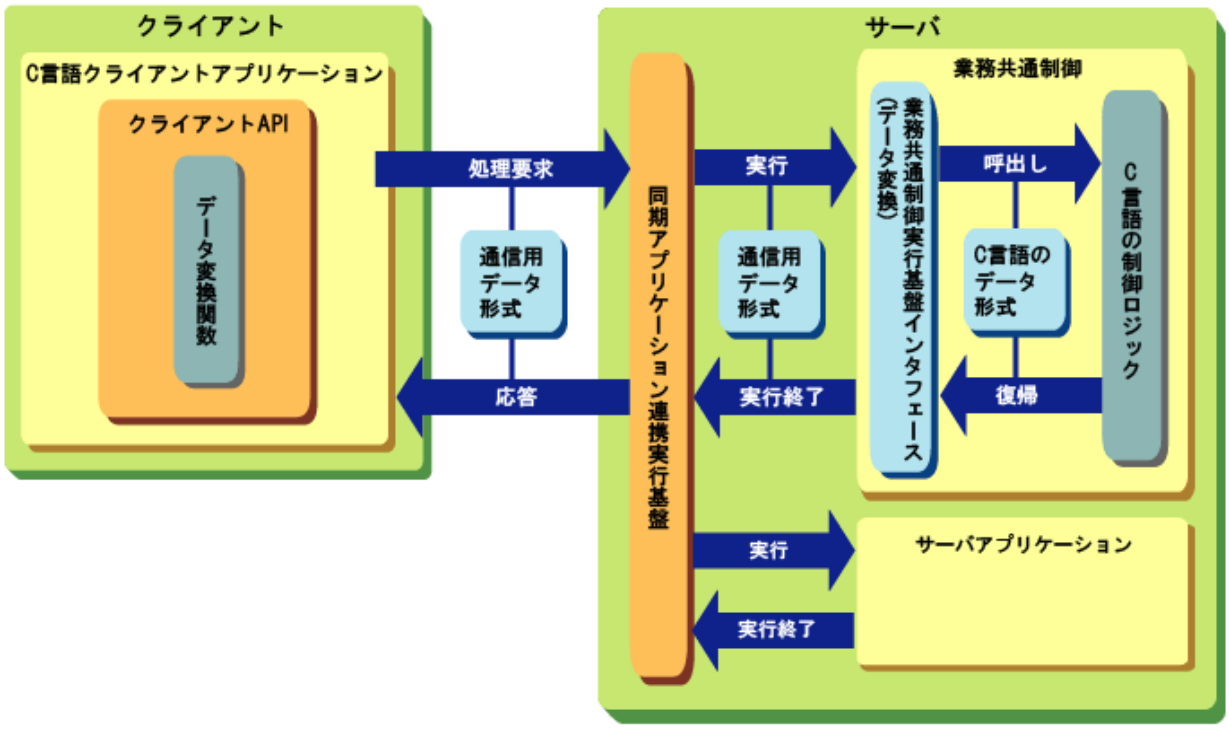
3.3.5 業務共通制御実行基盤インタフェース生成ツール

業務共通制御をアプリケーション連携実行基盤上で動作させるために、アプリケーション連携実行基盤と制御ロジックをつなぐための業務共通制御実行基盤インタフェースおよびアプリケーション連携実行基盤上で管理するインタフェース情報ファイルを生成するためのツールです。
このツールでは、業務共通制御のインタフェースを定義したIDLファイルおよび付加情報を定義した業務共通制御情報入力ファイルをツールの入力パラメタとして指定することにより、業務共通制御実行基盤インタフェース、制御アプリケーション定義ファイルおよび業務共通制御アプリケーションインタフェース定義ファイルを生成します。
ユーザ作成の制御ロジックとツールにより生成された業務共通制御実行基盤インタフェースをコンパイル／リンクしてライブラリを作成し、制御アプリケーション定義ファイルおよび業務共通制御アプリケーションインタフェース定義と共にアプリケーション連携実行基盤上に配備することで、アプリケーション連携実行基盤から制御ロジックを実装した業務共通制御が実行可能になります。
業務共通制御実行基盤インタフェースの位置づけを以下に示します。



業務共通制御実行基盤インタフェース生成ツールを使用したアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“業務共通制御の開発”を参照してください。

C言語クライアントアプリケーションを開発する場合、業務共通制御実行基盤インタフェース生成ツールは、同期アプリケーション連携実行基盤に配備された制御ロジックをC言語クライアントアプリケーションから呼び出すためデータ変換関数も生成します。



C言語クライアントアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“クライアントアプリケーションの開発(C言語)”を参照してください。

3.4 実行環境

アプリケーション連携実行基盤では、ワークユニットによる多重度制御や異常監視機構をベースに、信頼度の高いアプリケーションの実行制御を行います。

同期アプリケーション連携実行基盤は、クライアントアプリケーションのクライアントAPIで呼び出されたサーバアプリケーションを実行します。

同期アプリケーション連携実行基盤は、Interstage Application ServerのCORBAワークユニット上で動作するため、CORBAワークユニットで提供される各機能を利用することで高信頼なシステムを構築することが可能です。

CORBAワークユニットの機能詳細については、“Interstage Application Server OLTPサーバ運用ガイド”を参照してください。

3.4.1 クライアントアプリケーション

従来、CORBAアプリケーションなどのサーバアプリケーションの実行には、実行するサーバアプリケーションのインタフェースに合わせた専用のクライアントAPIをIDLファイルから作成する必要がありました。その為、クライアントアプリケーションでは、サーバインタフェースごとに、異なるクライアントAPIを用いたアプリケーション開発が必要でした。また、クライアントアプリケーションの作成にはCORBAサーバへの接続方法などCORBAアプリケーションの知識を有する必要がありました。
同期アプリケーション連携実行基盤では、クライアントAPIを利用することにより、各サーバアプリケーションで共通のAPIを使用したクライアントアプリケーションの開発が可能となります。また、アプリケーション開発には、CORBAに対する詳しい知識が不要となります。
Javaクライアントアプリケーションにおける操作は、beanまたはHashMapまたはIndexedRecordにデータを設定しJava用のクライアントAPIを呼び出すだけで、サーバアプリケーション固有のAPIを使用する必要はありません。そのため、クライアントのJavaアプリケーションとサーバアプリケーションの依存関係が少なくなり、クライアントとサーバを独立して開発が行えます。
C言語のクライアントアプリケーションにおける操作は、データ変換関数によりデータを変換しC言語用のクライアントAPIへ渡すだけです。
クライアントAPIの詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“クライアントAPI”を参照してください。

3.4.2 サーバアプリケーション

同期アプリケーション連携実行基盤は、定義ファイルに従い、サーバアプリケーション上の業務処理の呼出しなどアプリケーション実行制御を行います。
業務ロジック内の各種出口をアプリケーション実行制御がコントロールするため、ユーザアプリケーションにおいて、アプリケーションの制御を行うロジックの開発を不要とします。また、アプリケーションの振分けやトランザクション完了指示など、システムに応じて個別に制御を行ないたい場合は、業務共通制御を使用することにより同期アプリケーション連携実行基盤の実行制御をカスタマイズすることができます。
アプリケーション実行制御の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“アプリケーション実行制御”を参照してください。また、業務共通制御の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“同期アプリケーション連携実行基盤編”の“業務共通制御の開発”を参照してください。

3.4.3 アプリケーションの負荷分散

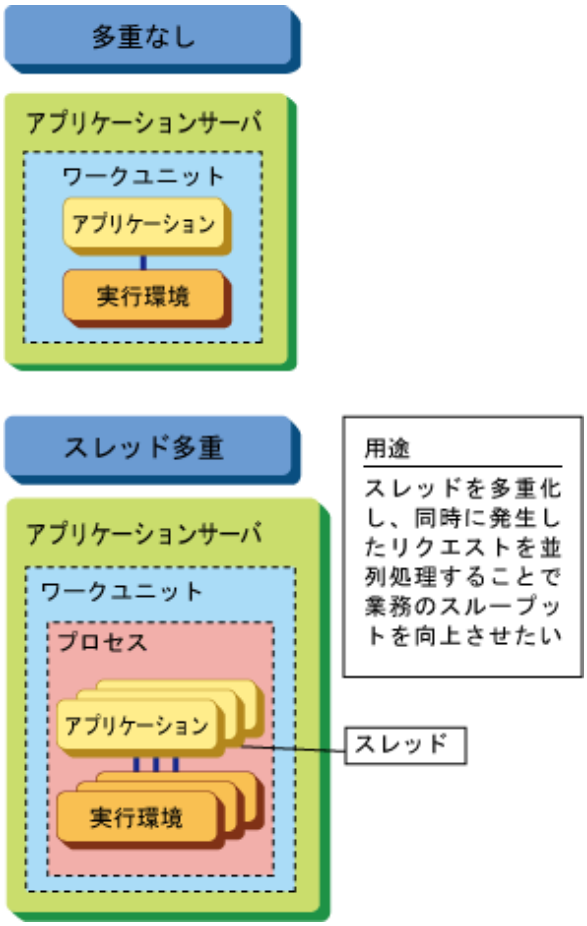
アプリケーション処理性能の不足に対しては、以下の方法で負荷分散することで処理性能の向上を図ることができます。

- アプリケーション処理スレッドの多重化
- アプリケーション処理プロセスの多重化
- アプリケーションごとのサーバ独立

なお、クライアントアプリケーション多重度の設定方法については、“◆Javaでアプリケーションを作成する場合”の業務処理開始アプリケーションをJavaで作成する場合に同じです。また、サーバアプリケーションの多重度の設定方法は、“4.4.3 アプリケーションの負荷分散”の業務処理実行アプリケーションをCOBOLで作成する場合に同じです。

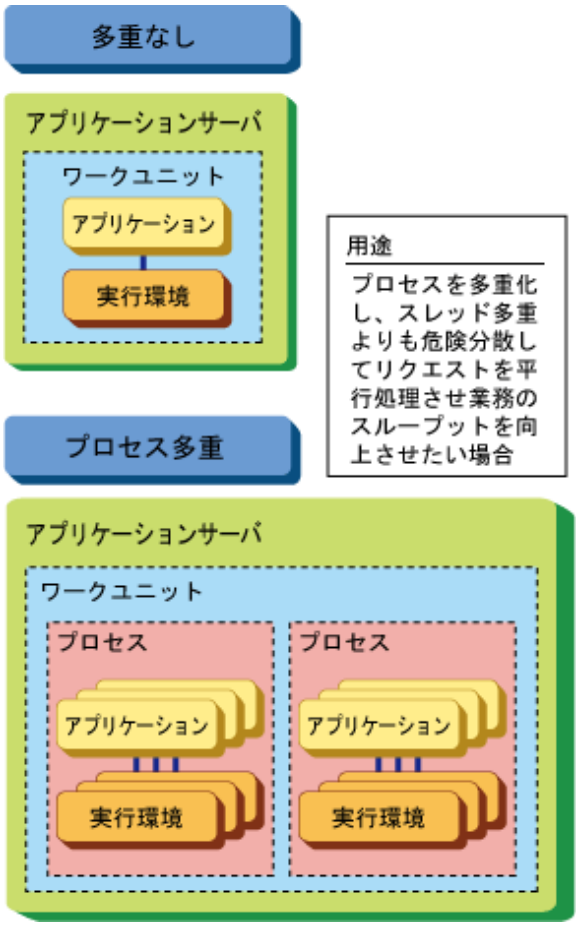
■アプリケーション処理スレッドの多重化

サーバ構成は変更せず、該当のアプリケーションを処理するスレッドの多重度を増加させることにより、アプリケーションの処理性能を高めます。サーバの処理能力に余裕がある場合は、この方法で業務処理のスループットが向上します。



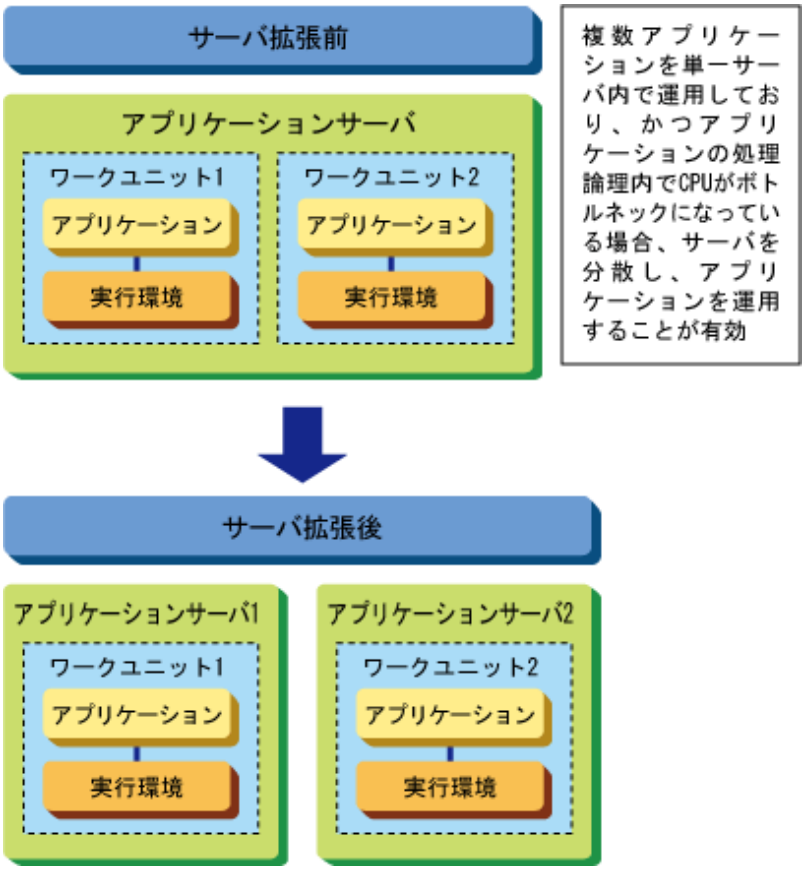
■アプリケーション処理プロセスの多重化

COBOLのアプリケーションなどの理由によりスレッド多重が利用できない場合や特定のスレッドが異常になった場合に他の処理スレッドへ影響を与えたくない場合、該当のアプリケーションを処理するプロセスの多重度を増加させることにより、アプリケーションの処理性能を高めます。なお、プロセス内のスレッドは、スレッドの多重化と併用することができます。



■アプリケーションごとのサーバ独立

単一のアプリケーションサーバ内で複数のアプリケーションを運用している場合、あるアプリケーションの処理コストの影響で、他のアプリケーションの処理能力が低下する可能性があります。そのような場合には、アプリケーションごとにサーバを独立させることにより、それぞれのアプリケーションの処理能力を高めることができます。

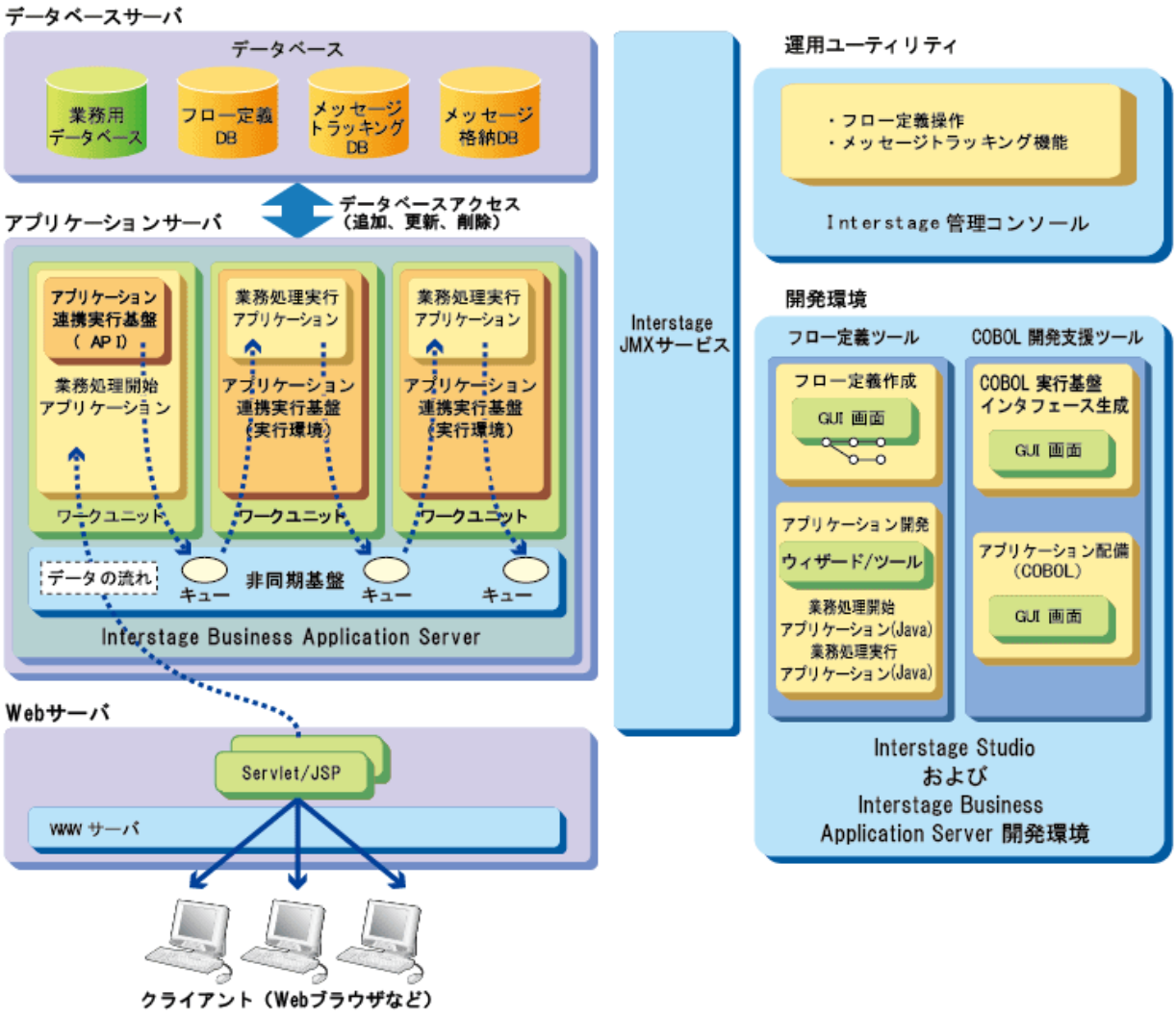

サーバ拡張前
アプリケーションサーバ
ワークユニット1
アプリケーション
実行環境
ワークユニット2
アプリケーション
実行環境
複数アプリケーションを単一サーバ内で運用しており、かつアプリケーションの処理論理内でCPUがボトルネックになっている場合、サーバを分散し、アプリケーションを運用することが有効
サーバ拡張後
アプリケーションサーバ1
ワークユニット1
アプリケーション
実行環境
アプリケーションサーバ2
ワークユニット2
アプリケーション
実行環境

第3部 非同期アプリケーション連携実行基盤編EE

第4章 非同期アプリケーション連携実行基盤の機能

4.1 構成要素

本節では、非同期アプリケーション連携実行基盤の構成要素について説明します。



■アプリケーション連携実行基盤の構成要素

- 開発環境(フロー定義ツール、COBOL開発支援ツール)

  アプリケーション連携実行基盤で用いるCOBOL・Javaアプリケーションの開発、フロー定義の作成および更新をシームレスに行うInterstage Studio上の統合開発環境です。

- 実行環境

  アプリケーション連携実行基盤が実現する信頼性の高いアプリケーション連携環境の基盤となる実行制御プログラムです。この実行環境により、トランザクション制御によるメッセージとデータベースの一貫性保証、多重度制御、異常監視機能を備えたアプリケーション実行機能、およびフロー定義に従ったアプリケーション連携などの処理を行います。

- 運用ユーティリティ

  アプリケーション連携のために使用されるメッセージの経路情報を画面から参照可能にするメッセージトラッキング機能およびフローの環境設定などシステム運用を容易にする機能です。

・ 業務処理開始アプリケーション

アプリケーション連携を開始するアプリケーションです。業務内容の情報をメッセージに設定し、フローを開始します。本アプリケーションは、開発環境を使用して作成します。

・ 業務処理実行アプリケーション

業務処理を行うアプリケーションです。入力したメッセージをもとにデータベースの更新などを行います。本アプリケーションは、開発環境を使用して作成します。

・ 業務用データベース

利用者の業務で使用するデータを格納するデータベースです。

・ フロー定義DB

開発環境で作成したアプリケーション連携を指定したフロー定義を格納するデータベースです。

・ メッセージトラッキングDB

アプリケーション連携の処理中に発生したエラー情報を格納するデータベースです。

・ メッセージ格納DB

メッセージとDBの整合性保証機能を使用する場合にアプリケーションで使用するメッセージを格納するデータベースです。

・ JMX

Sun MicrosystemsのJava Management Extension(JMX)仕様をベースとしたInterstage Application Serverの運用操作を行う機能です。

・ 非同期基盤

Interstage Application Serverのイベントサービスおよび JMS機能です。

■アプリケーション連携実行基盤におけるサーバ配置

アプリケーション連携実行基盤で利用するサーバの種類を示します。これらのサーバは、スケーラブルに拡張することが可能です。

・ アプリケーションサーバ

アプリケーションサーバは、ユーザアプリケーションである業務処理開始アプリケーションや業務処理実行アプリケーションをはじめ、アプリケーション連携実行基盤の実行環境、ワークユニットおよびキューなどを運用するサーバです。

・ データベースサーバ

データベースサーバは、メッセージ格納DBを含むユーザの業務データおよびアプリケーション連携実行基盤が使用する資源(フロー定義DBおよびメッセージトラッキングDB)を格納するサーバです。データベースサーバには、以下のいずれかのデータベースが必要です。

ー 同梱のSymfoware/RDB

本製品に同梱されるデータベース機能です。本データベースには、フロー定義DBおよびメッセージトラッキングDBを配置することができます。業務用のデータベースとして使用できませんので、業務用のデータを格納する場合は、製品版のSymfoware ServerまたはOracleを使用してください。

ー 製品版のSymfoware ServerまたはOracle

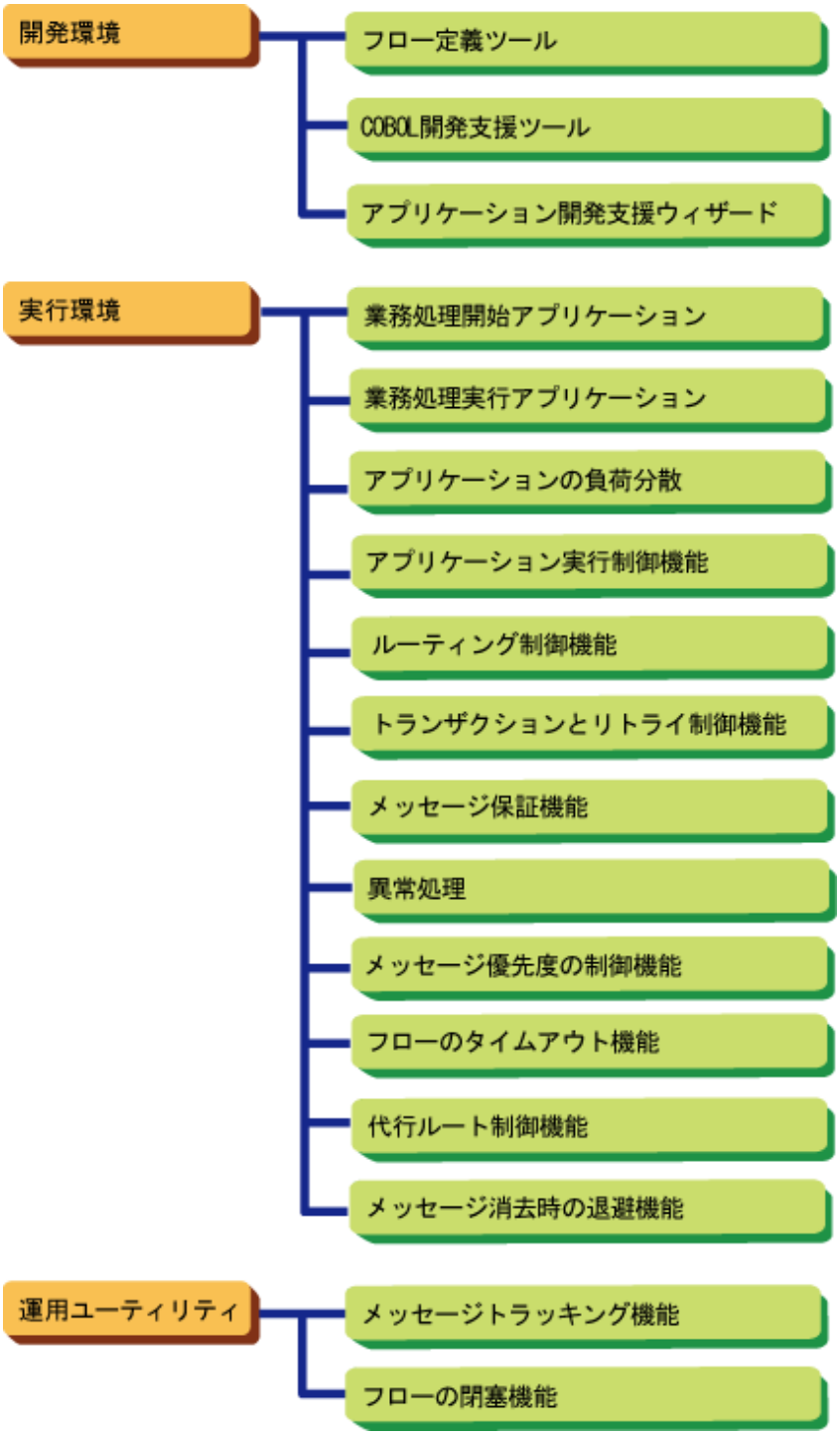
ユーザの業務データを格納するデータベース製品です。本製品には同梱されていませんので別途購入してください。なお、製品版のSymfoware ServerまたはOracleには、フロー定義DB、メッセージトラッキングDBおよびメッセージ格納DBを含む業務用データベースのすべてを配置できます。

・ Webサーバ

Webブラウザからの要求を契機に業務処理を開始するシステムを構築する場合、Webサーバを利用する必要があります。なお、アプリケーション連携実行基盤での使用は必須ではありません。

4.2 機能構成

アプリケーション連携実行基盤は、開発環境、実行環境および運用ユーティリティの3つの機能から構成されています。各機能について以降の節で説明します。



4.3 開発環境

アプリケーション連携実行基盤で使用するアプリケーション連携フロー(フロー定義)およびアプリケーションを開発するための開発環境です。以下に開発環境の機能を説明します。
業務処理開始アプリケーションおよび業務処理実行アプリケーションを開発する場合に必要となる各資源の概要を以下に説明します。

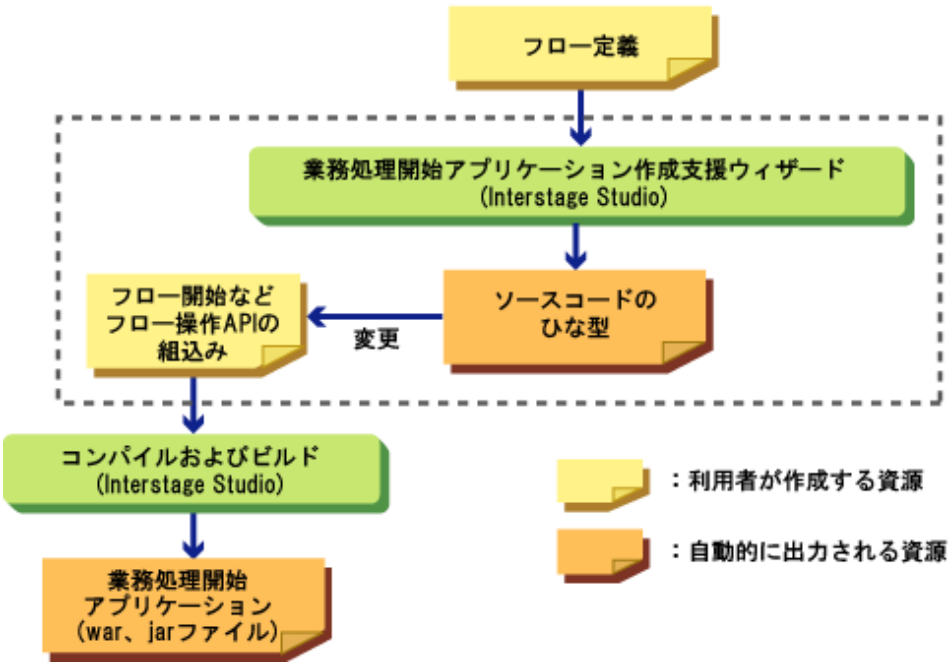
なお、フロー定義の作成方法およびアプリケーションの開発方法の詳細については“Interstage Business Application Server アプリケーション開発ガイド”を参照してください。

■業務処理開始アプリケーション

業務処理開始アプリケーションを開発する場合の開発作業の流れと開発する資源を以下に示します。

資源名	説明
フロー定義	アプリケーション連携のパターン(回覧型など業務のシーケンスおよび分岐条件など)や異常処理の方式選択等の設定を定義します。このフロー定義は、フロー定義生成ツールで作成します。
業務処理開始アプリケーションのソース	業務処理開始アプリケーション作成支援ウィザードが出力したソースコードのひな型にフローを開始するためのAPIなどを組み込んで作成します。

Javaの業務処理開始アプリケーション作成の流れ



[図の説明]

1. フロー定義を入力に業務処理開始アプリケーション作成支援ウィザードを利用して、業務処理開始アプリケーションのソースコードのひな型を出力します。
2. ひな型を編集性、業務処理開始アプリケーションにフロー開始APIなどフローを起動するための処理を記述します。
3. 業務処理開始アプリケーションをInterstage Studioでコンパイルおよびビルドして作成します。

■業務処理実行アプリケーション

業務処理実行アプリケーションを作成する場合、開発するアプリケーションの言語に応じて作成してください。業務処理実行アプリケーションを開発する場合に必要となる各資源の概要を以下に説明します。

資源名	説明	COBOL	Java
COBOL登録集	COBOLの業務処理実行アプリケーションのインタフェースを定義したファイルです。COBOLの実行基盤インタフェースを生成する場合に使用し、利用者が作成します。	○	
COBOL実行基盤インタフェース	アプリケーション連携実行基盤とCOBOLのサーバアプリケーションを繋ぐためのインタフェースです。業務アプリケーションで利用する言語に合わせたデータ型への変換、アプリケーション連携実行基	○	

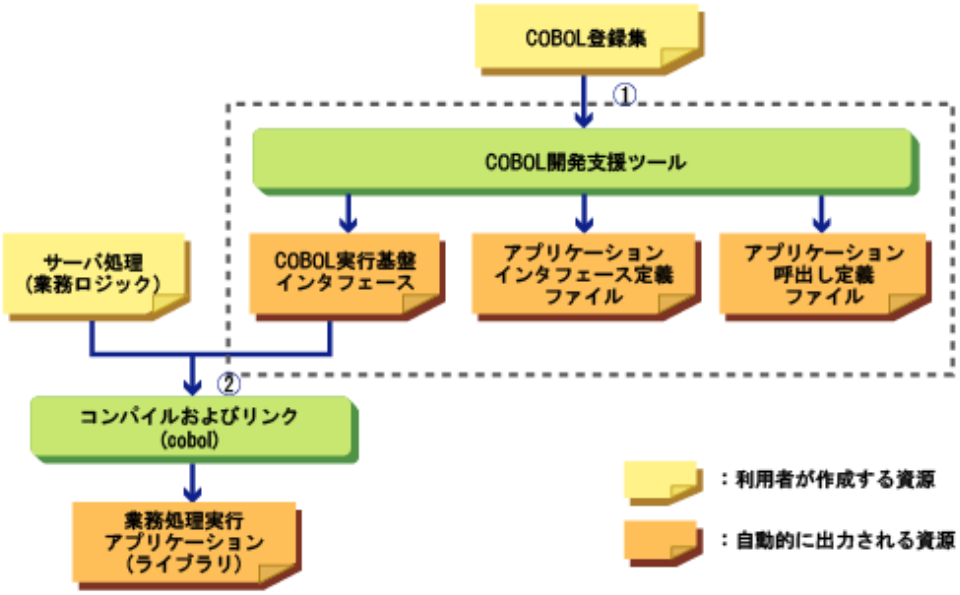
資源名	説明	COBOL	Java
	盤から業務処理実行アプリケーションへの受け渡しを行うデータ領域の獲得および解放を行います。このインタフェースは、COBOL開発支援ツールで生成します。		
アプリケーションインタフェース定義ファイル	業務処理実行アプリケーションの業務処理名、パラメタ名および型などのインタフェース情報を定義したファイルです。COBOLのサーバアプリケーションを配備する場合に指定します。指定したファイルは、アプリケーション連携実行基盤の動作時に読み込まれます。このファイルは、COBOL開発支援ツールで作成します。	○	
アプリケーション呼出し定義ファイル	業務処理実行アプリケーション名、業務処理名およびユーザ作成ライブラリ名を定義したファイルです。サーバアプリケーションを配備する場合に指定します。指定したファイルは、アプリケーション連携実行基盤の動作時に読み込まれます。このファイルは、COBOL開発支援ツールで作成します。	○	
フロー定義	アプリケーション連携のパターン(回覧型など業務のシーケンスおよび分岐条件など)や異常処理の方式選択等の設定を定義します。このフロー定義は、フロー定義生成ツールで作成します。		○
Ejb-jar.xml	Message-driven Beanとして動作する非同期アプリケーション連携実行基盤の実行環境のデプロイメントディスクリプタです。アプリケーション連携フロープロジェクトを新規作成したときに、プロジェクト内のプロジェクトのソースフォルダ > [META-INF]フォルダに自動作成されます。		○
Application.xml	EARに含まれるEJB-JARの情報を記述したデプロイメントディスクリプタです。Interstage Studioの[EARファイル生成]ウィザードを利用してEARファイルを生成する場合に、プロジェクト内のプロジェクトフォルダに自動作成されます。		○
logConf.xml	ログ定義ファイルです。非同期アプリケーション連携実行基盤が利用する汎用ログの出力先やフォーマットなどの情報を設定できます。アプリケーション連携フロープロジェクトを新規作成したときに、プロジェクト内のプロジェクトフォルダに自動作成されます。		○
Logresource.xml	ログメッセージファイルです。アプリケーションで使用する文字列データをログ出力用にカスタマイズして、メッセージの一部に動的な情報を埋め込んで定義します。アプリケーション連携フロープロジェクトを新規作成したときに、プロジェクト内のプロジェクトフォルダに自動作成されます。		○
サーバ処理(業務ロジック)	サーバで実行する業務用データベースのアクセスなどの業務ロジックです。利用者が、各言語で開発します。	○	○

○: 業務処理実行アプリケーションを作成するうえで必要

◆COBOLの業務処理実行アプリケーションを利用する場合

COBOLの業務処理実行アプリケーションを開発する場合の開発作業の流れと開発する資源を以下に示します。

COBOLの業務処理実行アプリケーション作成の流れ



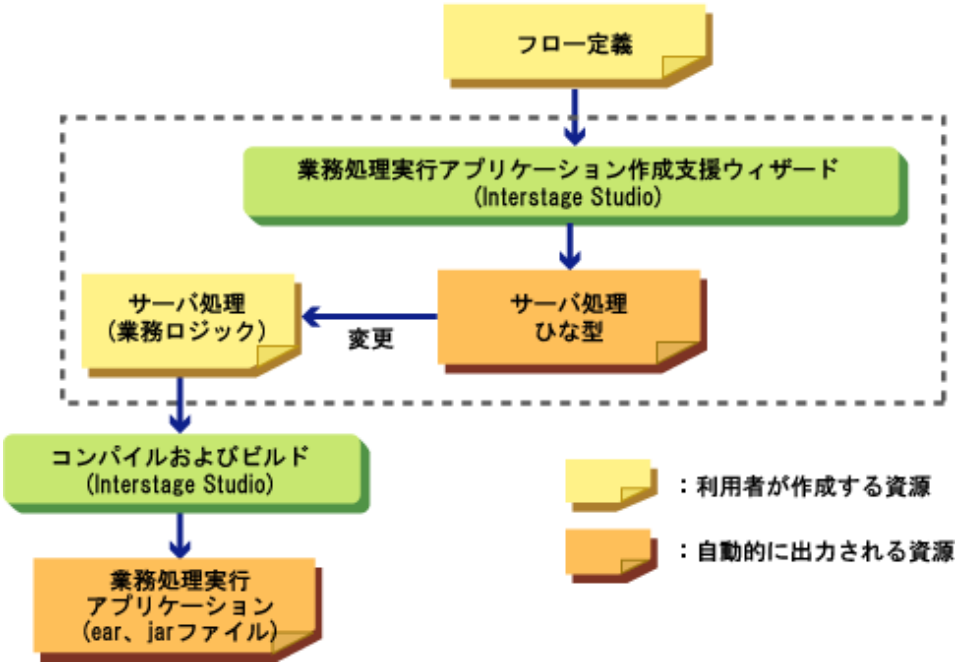
[図の説明]

1. COBOL登録集を入力にし、COBOL開発支援ツールを使用してCOBOL実行基盤インタフェースを生成します。
2. 業務処理実行アプリケーションは、COBOL開発支援ツールによって生成されたCOBOL実行基盤インタフェースと利用者が作成したサーバの業務ロジックを合わせてコンパイルおよびリンクして作成します。

◆Javaの業務処理実行アプリケーションを利用する場合

Javaの業務処理実行アプリケーションを開発する場合の開発作業の流れと開発する資源を以下に示します。

Javaの業務処理実行アプリケーション作成の流れ



[図の説明]

1. フロー定義を入力に業務処理実行アプリケーション作成支援ウィザードを利用して、業務処理実行アプリケーション(サーバ処理)のソースコードのひな型を出力します。
2. ひな型にサーバ処理(業務ロジック)を記述します。
3. 業務処理実行アプリケーションをInterstage Studioでコンパイルおよびビルドして作成します。

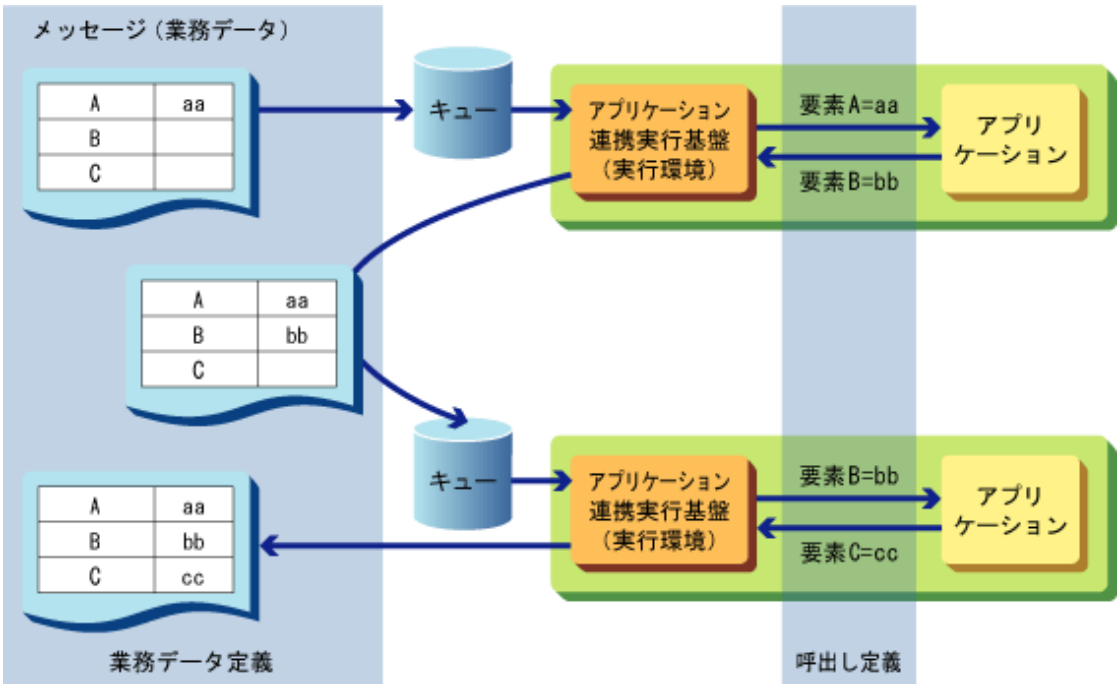
4.3.1 フロー定義ツール

アプリケーション連携実行基盤を使用するためには、アプリケーション連携のパターン(回覧型など業務のシーケンスおよび分岐条件など)や異常処理の方式選択といった設定を、フロー定義ツールにより行います。以下の各項目をGUI画面で設定します。

- 業務データ定義
- ルーティング定義
- 呼出し定義
- 異常処理定義

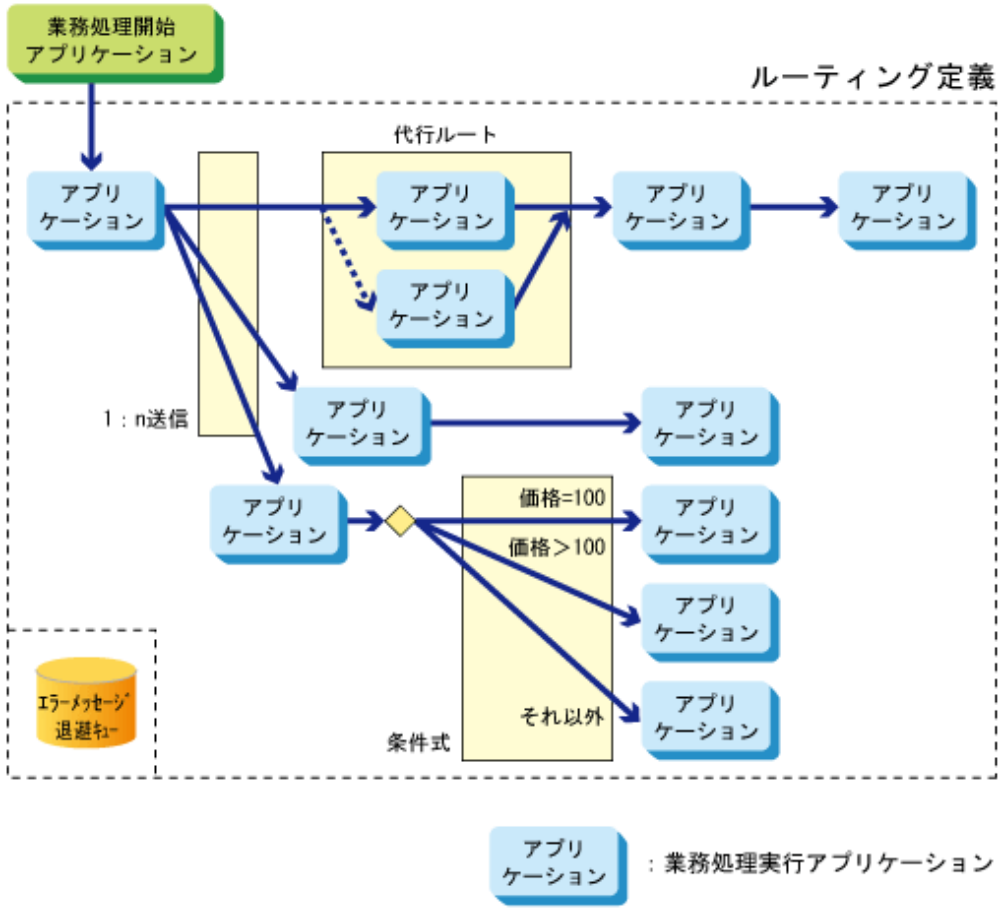
■業務データ定義

業務データ定義では、アプリケーションが使用する業務データの構造(要素名、型)を定義します。
定義した業務データは、業務処理実行アプリケーション定義においてアプリケーションの呼び出し方法を指定する際に利用し、指定した任意の要素をパラメタに関連付けることができます。また、復帰情報に業務データを関連付けることで、アプリケーションの処理結果を業務データに反映することもできます。
以下に業務データとアプリケーションのパラメタとの関係図を示します。



■ルーティング定義

アプリケーション連携実行基盤では、メッセージのルーティングおよびアプリケーションの呼び出し(以降、ルーティング情報と呼びます)を、ルーティング定義に基づいて自動的に処理する機能です。
ルーティング定義は、フロー定義ツールでメッセージの流れを表す図形オブジェクトを並べ替えることで、容易に作成することができます。
アプリケーション連携実行基盤を使用して構築するシステムは、従来のようにメッセージの送受信に関わる処理、およびアプリケーションを呼び出す処理を、アプリケーション自体に記述する必要がないため、システム構築を容易に、かつ早期に実現することを可能にします。
ルーティング定義における概念図を以下に示します。



■呼出し定義

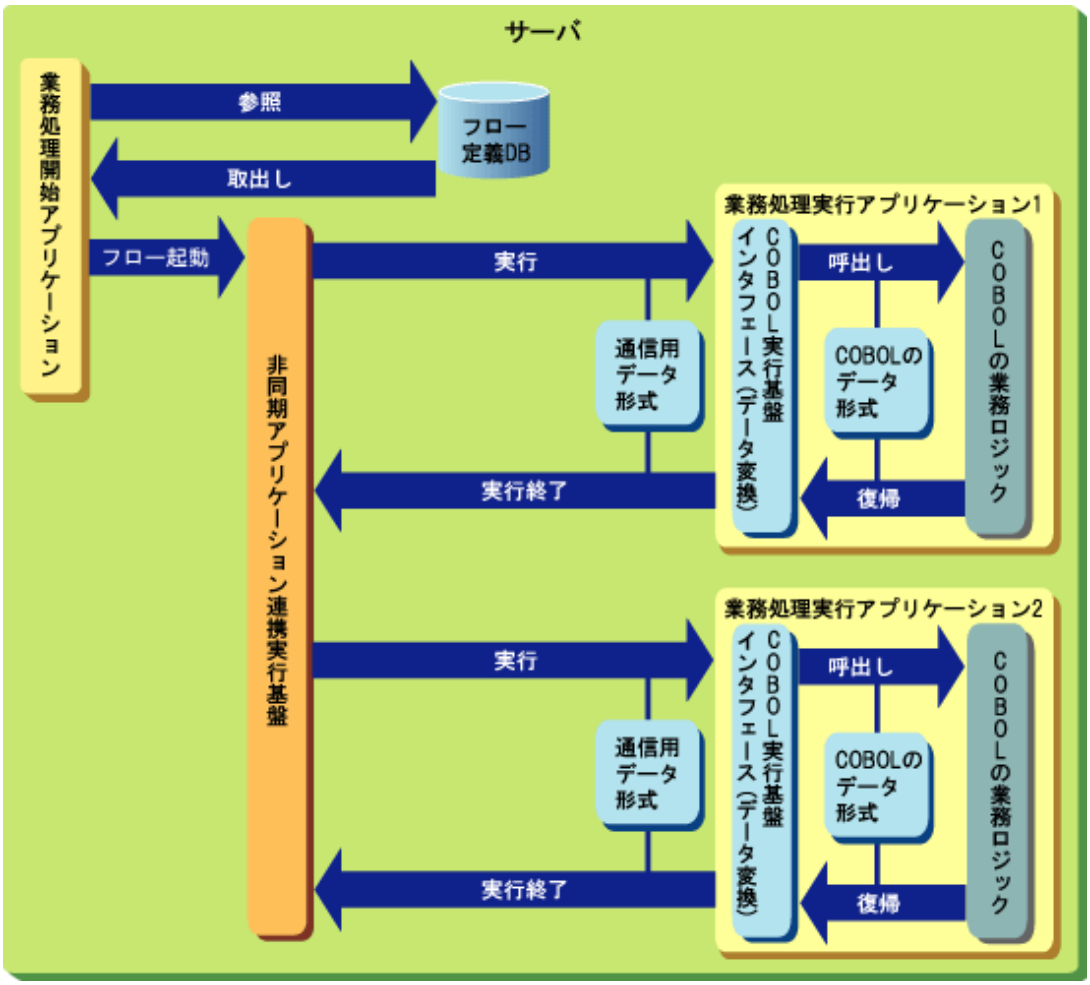
呼出し定義では、各アクティビティで呼び出す業務処理実行アプリケーションの名前と入力パラメタと戻り値を業務データの要素に対応付けます。

■異常処理定義

異常処理定義では、アプリケーション連携実行基盤がルーティング定義にしたがって業務処理実行アプリケーションを呼び出した後、アプリケーションで異常が発生した場合の後処理を定義することができます。異常処理定義では、アプリケーションの例外によりエラーメッセージ退避キューへの退避などの後処理をフロー定義単位に選択することができます。

4.3.2 COBOL開発支援ツール

COBOL開発支援ツールは、アプリケーション連携実行基盤とCOBOLで開発した業務ロジックをつなぐために必要なファイルを生成するツールです。
このツールでは、サーバアプリケーションの付加情報とサーバアプリケーションが使用するCOBOL登録集を入力することにより、COBOL実行基盤インタフェース、アプリケーションインタフェース情報ファイルおよびアプリケーション呼出し定義ファイルを生成します。
次に、ユーザ作成の業務ロジックとツールにより生成されたCOBOL実行基盤インタフェースをコンパイルおよびリンクしてライブラリを作成します。アプリケーション呼出し定義ファイルおよびアプリケーションインタフェース定義と共にアプリケーション連携実行基盤上に配備することで、アプリケーション連携実行基盤から業務ロジックを実装したサーバアプリケーションが実行可能になります。
COBOL実行基盤インタフェースの位置づけを以下に示します。



COBOL開発支援ツールを使用したアプリケーション開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の “サーバアプリケーションの開発(COBOL)”を参照してください。

4.3.3 アプリケーション開発支援ウィザード

Javaの業務処理開始アプリケーションおよび業務処理実行アプリケーションを作成する場合に使用します。アプリケーション連携実行基盤で使用する業務処理開始アプリケーションまたは業務処理実行アプリケーションをJavaで開発する場合、Interstage Studioを使用して開発を行います。この時、アプリケーション連携実行基盤に特化した処理のアプリケーションへの組み込みは、Interstage Studioのウィザードを用いて容易に行うことが可能です。詳細は、“Interstage Business Application Server アプリケーション開発ガイド”を参照してください。

- 業務処理開始アプリケーション

  アプリケーション連携を開始する際に実行するフロー名を指定し、アプリケーション連携の起点(先頭)となるアプリケーションです。

  ー 利用できる言語

  Java

  ー アプリケーションの記述内容

  アプリケーション連携を開始するための情報(フロー定義の種類、メッセージおよび同期応答の有無など)を、アプリケーション連携実行基盤で提供するAPIを用いて記述します。

  ー アプリケーション作成のための作業

  アプリケーションの開発は、Interstage Studioでアプリケーション連携実行基盤のAPI組み込みウィザードにしたがって行います。

- 業務処理実行アプリケーション

  アプリケーション連携の途中(連携の先頭以外)から終点までのそれぞれで業務処理を実行するアプリケーションです。

  ー 利用できる言語

  Java

  ー アプリケーションの記述内容

  データベースの更新などの業務処理を記述します。

  ー アプリケーション作成のための作業

  アプリケーションの開発は、Interstage Studioで業務処理実行アプリケーションウィザードにしたがって行います。

4.4 実行環境

フロー定義に従ってアプリケーションの呼び出しを行うアプリケーション連携実行基盤の実行環境です。Interstage Application Serverのワークユニット上で動作し、複数のアプリケーションを、安全、かつ、短期に開発するための、アプリケーション実行制御機能、ルーティング制御機能、メッセージ保証機能、メッセージ優先度の制御機能、フローのタイムアウト機能、代行ルート制御機能、メッセージ保証時の退避機能等を提供します。

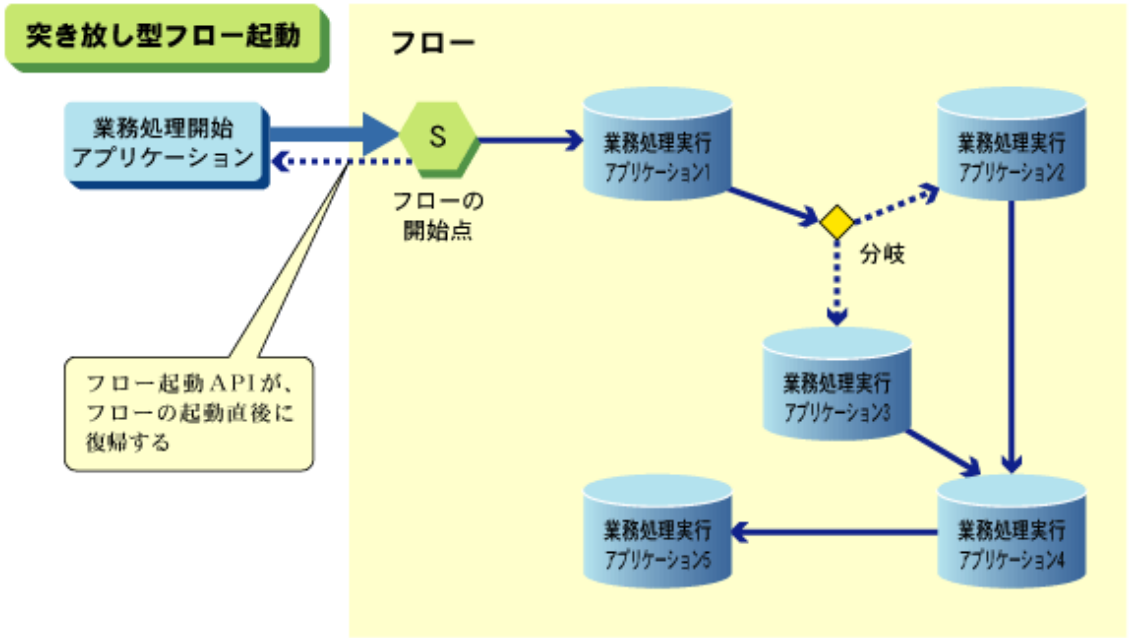
4.4.1 業務処理開始アプリケーション

業務処理開始アプリケーションは、フローとして構築された一連のアプリケーション群の処理を開始させる役割を持ったJavaのアプリケーションです。業務処理開始アプリケーションによるフローの処理開始を、フローの起動と呼びます。フローの起動には、以下の2種類のパターンがあります。

- 突き放し型フロー起動(非同期型)
- 待ち合わせ型フロー起動(同期型)

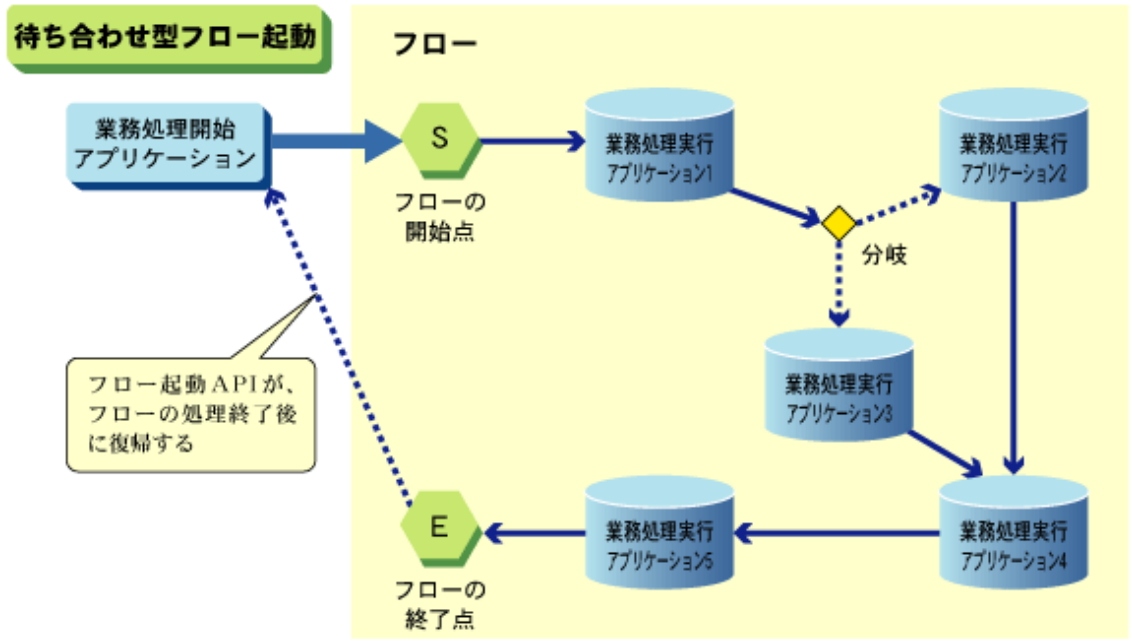
■突き放し型フロー起動(非同期型)

フロー起動を行った後、すぐに処理が復帰する起動方法です。待ち時間が少ないために処理のむだがなく、業務処理開始アプリケーションで多くの処理を受け付けて、連続的にフローを起動する必要がある場合などに適します。



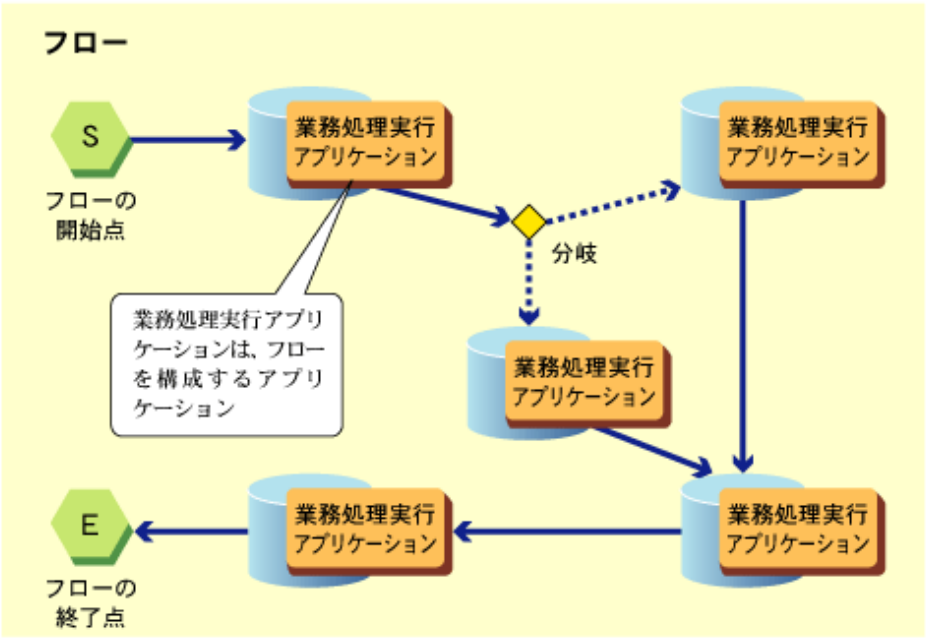
■待ち合わせ型フロー起動(同期型)

フロー起動を行った後、待ち状態になり、フローの処理が完了した時点で処理が復帰する起動方法です。業務処理開始アプリケーションで、フローの処理終了をリアルタイムに認識することが可能なため、業務処理開始アプリケーションがフローの処理終了を待って即時結果に応じた処理を行う場合などに適します。



4.4.2 業務処理実行アプリケーション

業務処理実行アプリケーションは、フローを構成するCOBOLまたはJavaのアプリケーションです。複数の業務処理実行アプリケーションがフロー定義にしたがって連携動作し、全体で業務処理を行います。



個々の業務処理実行アプリケーションは、アプリケーション連携実行基盤の実行環境により、呼出し定義で指定された順序で関数またはメソッドが呼び出されることになります。アプリケーション連携のためにアプリケーションに特別な処理を組み込む必要はありません。

4.4.3 アプリケーションの負荷分散

業務処理実行アプリケーション処理性能の不足に対しては、以下の方法で負荷分散することで処理性能の向上を図ることができます。

- アプリケーション処理スレッドの多重化
- アプリケーション処理プロセスの多重化
- アプリケーションごとのサーバ独立

■アプリケーション処理スレッドの多重化

サーバ構成は変更せず、該当のアプリケーションを処理するスレッドの多重度を増加させることにより、アプリケーションの処理性能を高めます。サーバの処理能力に余裕がある場合は、この方法で業務処理のスループットが向上します。以下にスレッド多重を行う場合の設定箇所を説明します。

◆COBOLでアプリケーションを作成する場合

業務処理開始アプリケーションおよび業務処理実行アプリケーションの多重度の設定は、システム構築シートにおいて、“ワークユニット設定とアプリケーション情報の入力”の“アプリケーション連携実行基盤動作設定”で“動作モード”に“スレッドモード”を指定し、“スレッド最大多重度”にスレッドの多重度を指定して出力されたワークユニット定義を登録または更新することで実施します。なお、多重度は、“ワークユニット定義”の“Application Programセクション”の“Thread Concurrency”でも指定できます。

◆Javaでアプリケーションを作成する場合

業務処理実行アプリケーション、およびMessage-driven Beanとして実装する業務処理開始アプリケーションの多重度の設定は、Interstage管理コンソールの以下の箇所で設定します。

- [Interstage Application Server] > [システム] > [ワークユニット] > [IJServer名] > [EARファイル名] > [業務処理実行アプリケーションのJARファイル名] > [アプリケーション連携実行基盤のEJBアプリケーション名] > [アプリケーション環境定義] > [初期起動インスタンス数]

ただし、業務処理実行アプリケーションとWebアプリケーションとして実装する業務処理開始アプリケーションを同一JavaVMで運用する場合の多重度の設定は、Interstage管理コンソールの以下の箇所で設定します。

- [Interstage Application Server] > [システム] > [ワークユニット] > [IJServer名] > [環境設定]タブ画面の[Servletコンテナ設定]の[同時処理数]

注意

Servletコンテナ設定の同時処理数には、Interstage管理コンソールの[Interstage Application Server] > [システム] > [サービス] > [Webサーバ] > [環境設定]タブ > [詳細設定] > [クライアントの同時接続数]で指定した値以上の数値を設定する必要があります。

また、業務処理開始アプリケーションをMessage-driven Bean以外のEJBとして実装する場合の多重度の設定は、Interstage管理コンソールの以下の箇所で設定します。

- [Interstage Application Server] > [システム] > [ワークユニット] > [IJServer名] > [環境設定]タブ画面の[EJBコンテナ設定]の[同時処理数]

スレッドの多重化は、必要に応じて値を設定してください。

用途

処理要求メッセージ格納キューのFIFO制御により、メッセージの処理順序を保証したい場合

用途

メッセージの処理順序を問われない業務でスループットを向上させたい場合

■アプリケーション処理プロセスの多重化

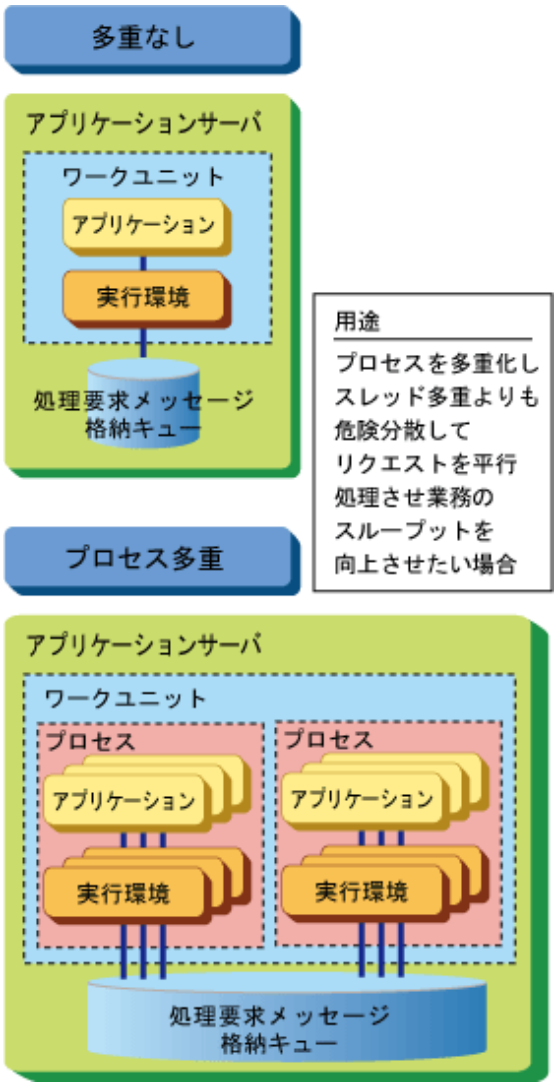
COBOLのアプリケーションなどの理由によりスレッド多重が利用できない場合や特定のスレッドが異常になった場合に他の処理スレッドへ影響を与えたくない場合、COBOLの業務処理実行アプリケーションの場合または業務処理開始アプリケーションでは、該当のアプリケーションを処理するプロセスの多重度を増加させることにより、アプリケーションの処理性能を高めます。なお、プロセス内のスレッドは、スレッドの多重化と併用することができます。

注意

Javaの業務処理開始アプリケーションをプロセス多重で使用する場合、ログ機能を利用することはできません。

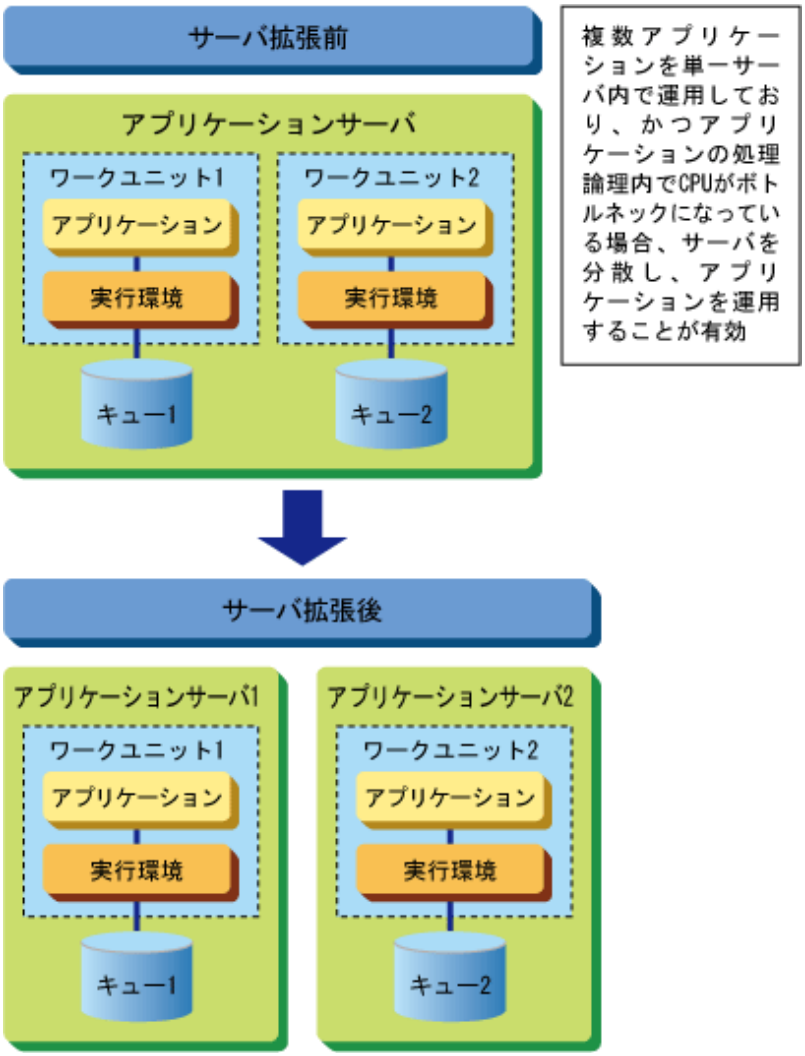
業務処理開始アプリケーションおよび業務処理実行アプリケーションの多重度の設定は、システム構築シートにおいて、“ワークユニット設定とアプリケーション情報の入力”の“アプリケーション連携実行基盤動作設定”で“プロセス最大多重度”に多重度を指定し、出力されたワークユニット定義を登録または更新することで実施します。なお、業務処理実行アプリケーションのプロセスの多重度は、“ワークユニット定義”の“Application Programセクション”の“Concurrency”でも指定できます。
なお、業務処理開始アプリケーションの多重度の設定は、Interstage管理コンソールの以下の箇所でも設定できます。

- [Interstage Application Server] > [システム] > [ワークユニット] > [ワークユニット名] > [環境設定]タブ画面の[ワークユニット設定]の[プロセス多重度]
- スレッド多重を併用する場合は、“■アプリケーション処理スレッドの多重化”を参照してください。



■アプリケーションごとのサーバ独立

単一のアプリケーションサーバ内で複数のアプリケーションを運用している場合、あるアプリケーションの処理コストの影響で、他のアプリケーションの処理能力が低下する可能性があります。そのような場合には、アプリケーションごとにサーバを独立させることにより、それぞれのアプリケーションの処理能力を高めることができます。



注意

アプリケーション連携実行基盤におけるアプリケーション間の連携は、キューへのメッセージ送信を利用して行われます。したがって、アプリケーションが動作するサーバが変わっても、キューが変わらなければ、ルーティング定義などを変更する必要はありません。

4.4.4 アプリケーション実行制御機能

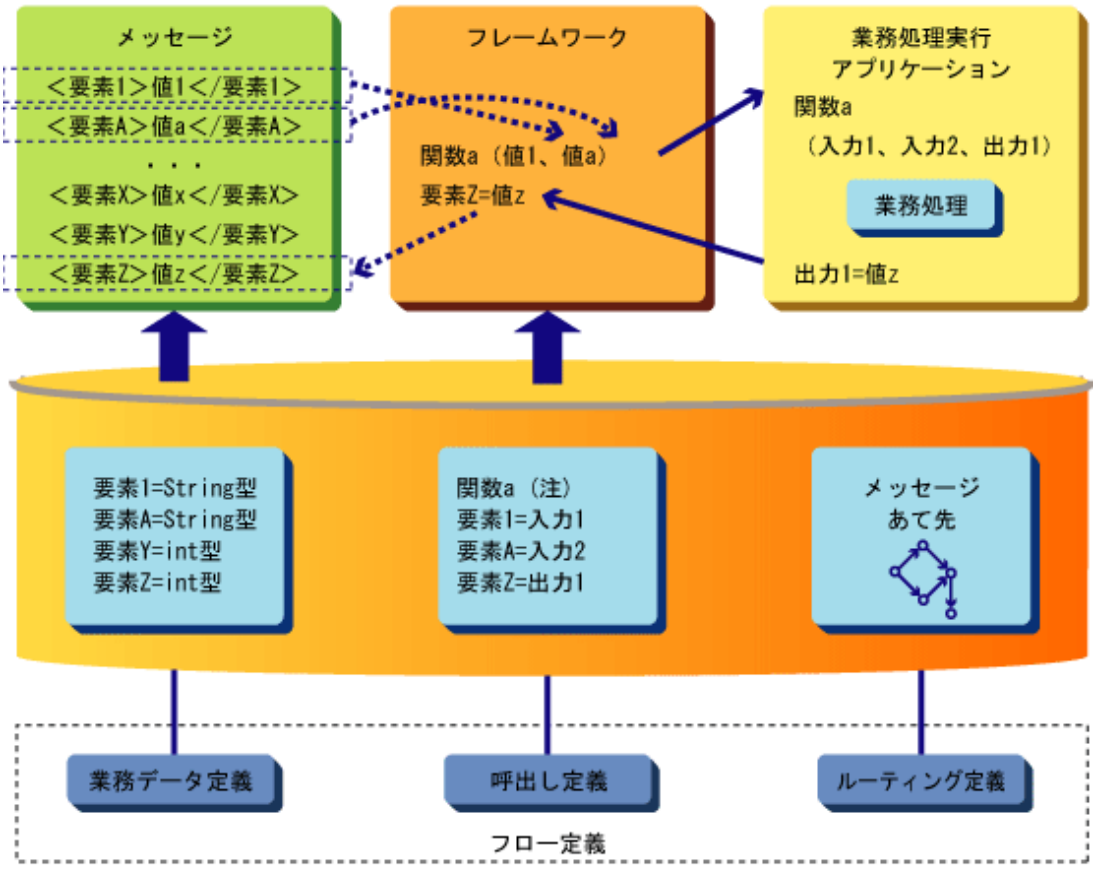
アプリケーション連携実行基盤では、ワークユニットによる多重度制御や異常監視機構をベースに、信頼度の高いアプリケーションの実行制御を行います。アプリケーションの実行は以下のシーケンスで行います。

1. 入力したメッセージを、フロー定義で指定した業務処理実行アプリケーションに受け渡します。メッセージのフォーマットとアプリケーションの入力パラメタの対応関係は、呼出し定義により決定し、アプリケーションの業務処理を行うメソッドを特定します。
2. アプリケーションが業務処理を行います。
3. アプリケーションの出力結果を受け取り、メッセージに格納します。
4. フロー定義にしたがって、次の宛て先となるアプリケーションに対応するキューにメッセージを送信します。
5. 定義ミスやアプリケーションの異常などにより業務処理がエラーとなった場合は、フロー定義で指定した異常処理定義に従った後処理(リトライやエラーメッセージ退避キューへのメッセージの退避など)を実行します。

■フロー定義と業務処理実行アプリケーションの関係

アプリケーションの連携は、メッセージを利用して行います。メッセージは、複数の要素(最大256個)で構成され、業務処理実行アプリケーションの呼び出しパラメタや復帰パラメタは、メッセージの各要素にそれぞれ対応します。メッセージ内の要素と型は、フロー定義の業務データ定義により関係付けます。
メッセージの各要素とアプリケーションの入出力パラメタの対応付けは、フロー定義の呼出し定義で行い、運用時にはアプリケーション連携実行基盤の実行環境が定義にしたがってメッセージと入出力パラメタの変換を行います。
アプリケーション間のメッセージの流れは、フロー定義のルーティング定義で定義します。

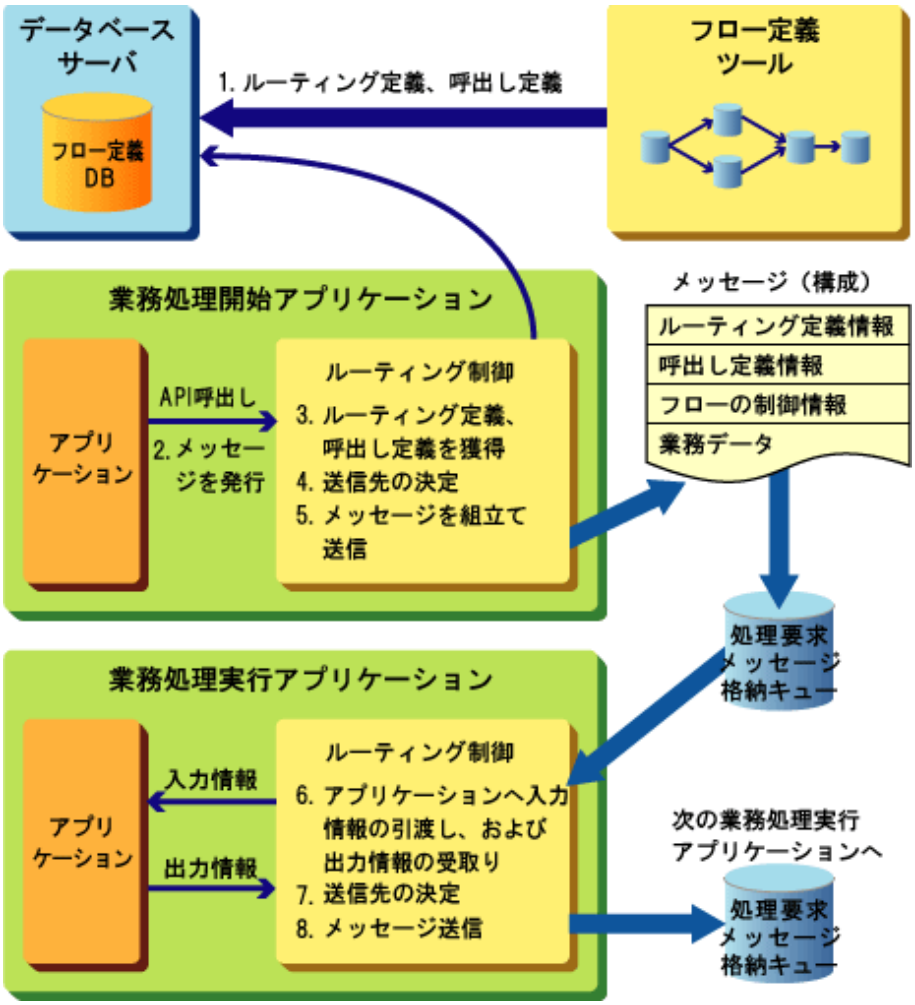
［フロー定義と業務処理実行アプリケーションの関係］



注）関数は、COBOLの場合プログラム名、Javaの場合メソッドに対応します。

4.4.5 ルーティング制御機能

ルーティング制御機能は、フロー定義にしたがってメッセージをルーティングし、アプリケーションを自動的に呼び出す機能です。
従来、アプリケーションの制御ロジックで行っていたアプリケーションの呼び出しやメッセージのルーティング先を判断し、送信する処理をルーティング制御機能が代替して実行します。これにより、アプリケーションを連携するシステム構築を早期に実現することが可能となります。
ルーティング制御機能が行うメッセージルーティングの概念図を以下に示します。



1. フロー定義ツールで定義したルーティング定義および呼出し定義は、データベースサーバのフロー定義DBへ格納されます。
2. メッセージを発行する業務処理開始アプリケーションでは、アプリケーション連携実行基盤が提供するAPIにフロー定義名および業務データの内容などを指定して、メッセージを発行します。
3. ルーティング制御機能では、活性保守の観点から、リアルタイムに定義情報を反映していくため、フロー定義DBに格納されたフロー定義および運用情報の更新確認を行い、定義情報が更新されている場合は、フロー定義名を検索キーとしてルーティング定義および呼出し定義を獲得します。
   フロー定義および運用情報の更新確認時におけるフロー定義DBへの接続は、データベースアクセス管理機能(非同期ワークユニット)またはワークユニットのDBコネクション設定(IJServerの場合)で事前コネクトを行うことにより接続にかかるオーバヘッドはなく、高速に更新確認を行うことが可能になります。そのため、業務処理開始アプリケーションは、単体のアプリケーションからメッセージを発行する運用よりもワークユニット上で動作するアプリケーションから発行する運用とすることで処理を高速化することができます。
4. ルーティング制御機能では、獲得したルーティング情報をもとに最初に送信するキューを決定します。
5. ルーティング制御機能では、ルーティング定義情報、呼出し定義情報、フローの制御情報および業務データを格納したメッセージを生成し、ルーティングされるキューへ送信します。フローの制御情報は、アプリケーション連携実行基盤の各機能が内部管理する制御データです。
6. 業務処理実行アプリケーションでは、受信したメッセージから呼出し定義情報を獲得し、ユーザアプリケーションに業務データの受渡しを行います。
7. ルーティング制御機能では、メッセージから獲得したルーティング情報をもとに次に送信するキューを決定します。送信先が分岐されている場合は、フロー定義で指定された分岐するための条件式でメッセージに格納されている業務データを評価し真となる送信先へメッセージを分岐します。
8. ルーティング制御機能は、決定した次のキューへメッセージを自動的に送信します。

4.4.6 トランザクションとリトライ制御機能

データベース操作を伴う処理では、処理結果により、トランザクションの完了操作を行う必要があります。また、エラー終了した場合には、業務処理の再実行が必要な場合があります。アプリケーション連携実行基盤では、これらの操作を自動的に行う機能を提供します。

■トランザクション制御

アプリケーション連携実行基盤でコネクションの管理を行います。管理するコネクションは、業務処理実行アプリケーションから通知される処理結果情報により、コミットやロールバックといったトランザクションの完了を自動的に行います。アプリケーション連携実行基盤が管理するコネクションを用いて、データベースの操作を行うことで、トランザクションの完了操作を意識することなく業務ロジックの作成に専念することが可能となります。
また、アプリケーション連携実行基盤のトランザクション制御を使用せずに、サーバアプリケーションで独自にトランザクション制御を行うこともできます。

■リトライ制御

業務処理実行アプリケーションが異常終了した場合に、処理を再度実行する機能です。次の2種類のリトライ機能があります。

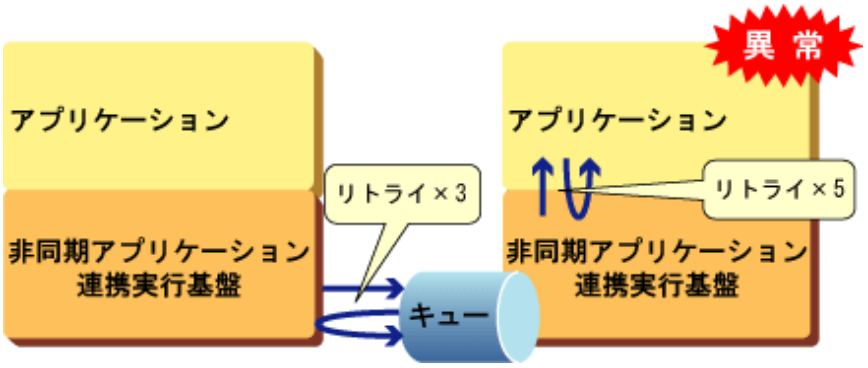
1. 非同期ワークユニットで定義するリトライ
   非同期ワークユニットのトランザクションリトライカウントです。受信したメッセージ、およびデータベースのトランザクションをロールバックした上で、処理を再度実行します。

2. フロー定義で定義するリトライ
   フロー定義の異常処理定義で定義するリトライです。データベースのトランザクションのロールバックを行わずに、異常終了したサーバアプリケーションを前処理より再呼び出しします。
   フロー定義によるリトライは、業務処理実行アプリケーションより通知されたエラーコードがフロー定義の異常処理定義でリトライ条件と一致していた場合に実施されます。

リトライ制御における上記2種類のリトライの関係は次の通りであり、最大「上記1のリトライ回数×上記2のリトライ回数」分、処理の再呼び出しが行われます。
なお、COBOLアプリケーションを実行する非同期アプリケーション連携実行基盤では、上記1のリトライ回数は、リトライ回数ではなく実行回数の意味となります。つまり、リトライ回数に1を設定した場合、リトライは行いません。
また、業務処理実行アプリケーションの前処理、業務処理、または後処理が強制リトライで終了した場合（処理結果情報に3が設定された場合）は、上記2.の定義を無視し、上記1.の定義に従ってリトライが行われます。

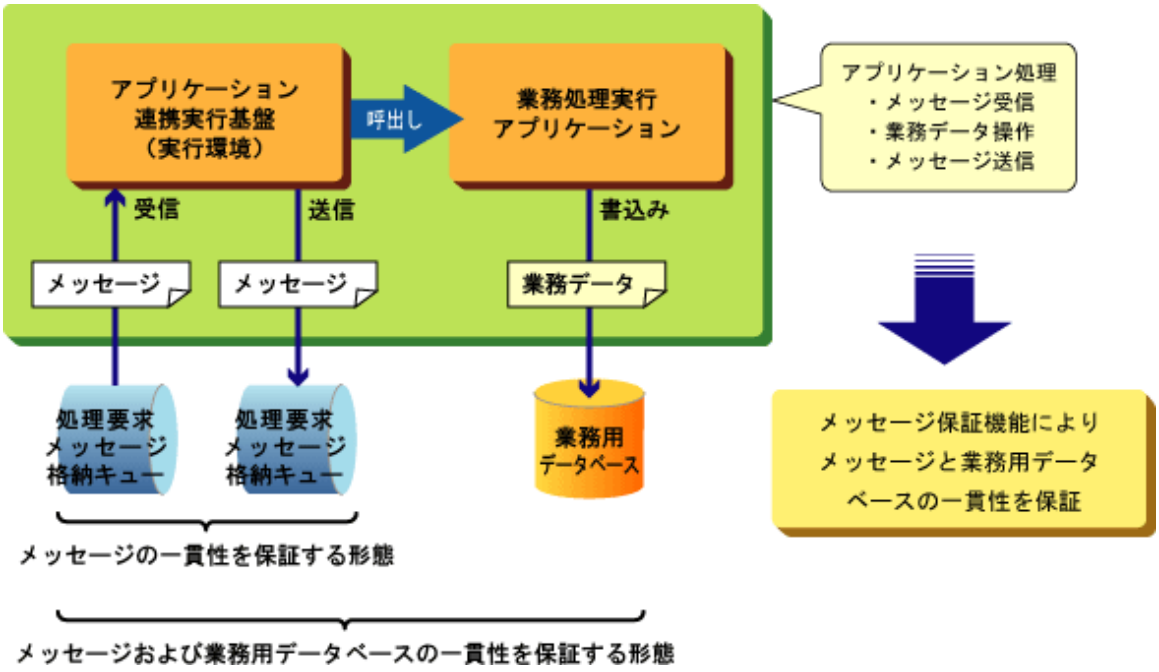
例として、Javaアプリケーションを実行する非同期アプリケーション連携実行基盤において、非同期ワークユニット定義でリトライ回数を3と指定し、フロー定義ツールの異常処理定義のリトライ回数を5と指定した場合、以下のように最大で5×(1+3)＝20回のリトライが実施されることになります。

1. フロー定義のリトライを5回実施
2. 非同期ワークユニット定義のリトライを1回実施（通算1回目）
3. フロー定義のリトライを5回実施
4. 非同期ワークユニット定義のリトライを1回実施（通算2回目）
5. フロー定義のリトライを5回実施
6. 非同期ワークユニット定義のリトライを1回実施（通算3回目）
7. フロー定義ツールのリトライを5回実施



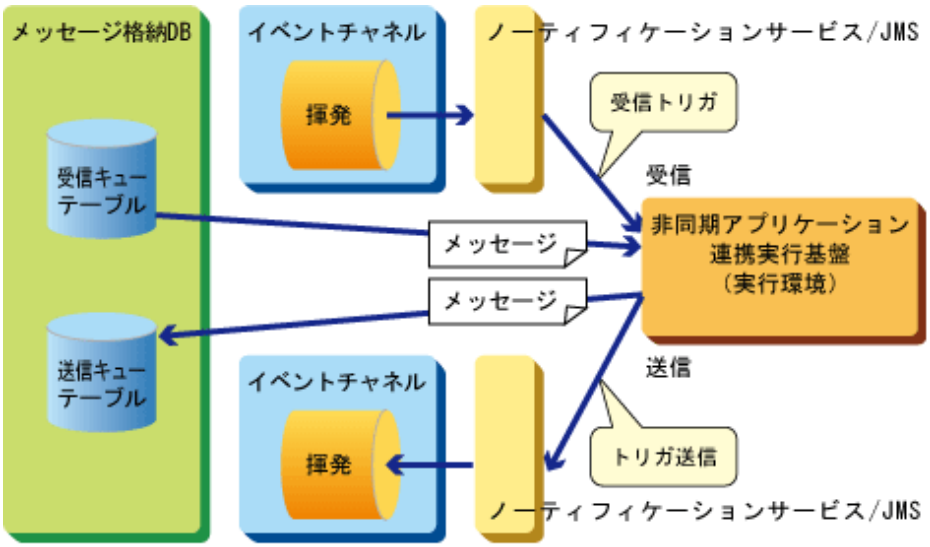
4.4.7 メッセージ保証機能

アプリケーション連携実行基盤では、アプリケーションの連携を行う手段としてメッセージを使用します。
メッセージ保証機能では、アプリケーション連携に使用するメッセージの送受信と業務処理で使用する業務用データベースの更新を一意な処理単位として、システムダウンやアプリケーション異常が発生しても、一貫性を保証する機能を提供します。
なお、メッセージ保証機能では、メッセージの順序性は保証されません。データベースの更新順番を厳守するなどの理由により、メッセージの順序性保証が必要な業務に適用する場合は、アプリケーションや運用により順序性を保証してください。
メッセージ保証機能の概要図を以下に示します。



■メッセージおよび業務用データベースの一貫性の保証

メッセージ保証機能は、トランザクションを使用してメッセージの送受信および業務用データベースの一貫性を保証します。なお、システムダウンなどの異常発生時にもメッセージを消失することなく一貫性を保証するため、メッセージの格納先は、メッセージ格納DBを使用します。
メッセージ格納DBにメッセージを格納し、データベース製品のローカルトランザクションを使用してメッセージの一貫性を保証します。本方式では、メッセージの受信トリガとしてDB連携用のイベントチャネルを使用し、メッセージ本体は業務用データベース内に作成するメッセージ格納DBに格納します。本方式を“メッセージとDBの整合性保証機能”と呼びます。メッセージとDBの整合性保証機能を利用する場合、業務用データベースに対する操作とメッセージ格納DBに対する操作を同一のコネクションで実行する必要があるため、メッセージ格納DBのデータベースリソース定義は、業務用データベースのデータベースリソース定義と同一の定義を使用します。



メッセージの格納先にデータベースを使用し、メッセージの一貫性をデータベースのローカルトランザクションにて保証します。イベントサービスやJMSに関する処理は、揮発運用のイベントチャネルをメッセージ送受信のトリガとして使用するため、トランザクションには含みません。

メッセージの格納先種別とその組み合わせによるメッセージ保証機能の一貫性保証範囲を、以下に示します。

メッセージ格納先種別			送信		
			キュー		メッセージ格納DB
			揮発運用のイベントチャネル	不揮発運用のイベントチャネル	
受信	キュー	揮発運用のイベントチャネル	×	×	×
		不揮発運用のイベントチャネル	×	×	×
	メッセージ格納DB		×	×	○

○: 一貫性保証できます。
×: 一貫性保証できません。

注意

- メッセージの格納先にメッセージ格納DBを使用し、ローカルトランザクションでメッセージと業務用データベースの一貫性を保証するためには、アプリケーション連携実行基盤の実行環境と同一のワークユニット上でアプリケーションを動作させる必要があります。業務処理実行アプリケーションから他のワークユニット上で動作するEJBアプリケーション、Webアプリケーションおよび CORBAアプリケーションなどを呼び出す運用の場合、一貫性の保証はアプリケーションで行います。
- 運用中にエラーが発生した場合や補償ルートのメッセージが発行された場合は、該当するメッセージ処理はロールバックされ、再びメッセージ受信からトランザクションが開始された後に、異常処理定義で指定した後処理や補償ルートのメッセージが発行されます。そのため、エラー発生時や補償ルートのメッセージの発行時でもメッセージの一貫性は保証されます。
- 補償ルートでは、メッセージの一貫性は保証されますが、メッセージの順序性は保証されません。

■メッセージ保証機能とトランザクション

メッセージ保証機能は、メッセージおよび業務用データベースの一貫性をデータベースのローカルトランザクションにより保証します。
業務処理実行アプリケーションの言語により以下の設定を行ってください。

◆(1)COBOLの業務処理実行アプリケーション

メッセージ保証機能を使用する場合、データベースアクセス管理機能を使用する必要があります。アプリケーションが、業務用データベースをアクセスするデータベースリソース定義をメッセージ格納DBのデータベースリソース定義として使用することで、同一のトランザクションでメッセージと業務用データベースの更新を同期することができます。設定方法の詳細については、“Interstage Business Application Server セットアップガイド”を参照してください。

◆(2)Javaの業務処理実行アプリケーション

アプリケーション連携実行基盤の実行環境のベースとなる、EJB(Message-driven Bean)のトランザクション管理およびトランザクション属性によるメッセージ保証を以下に示す通りに設定してください。

アプリケーション連携実行基盤の実行環境のベースとなるEJB(Message-driven Bean)			
トランザクション		一貫性保証	説明
管理種別	属性		
CMT	Mandatory	指定不可	Message-driven Beanでは指定できません。
	Required	○	一貫性を保証します。
	Supports	指定不可	Message-driven Beanでは指定できません。
	RequiresNew	指定不可	Message-driven Beanでは指定できません。
	NotSupported	×	トランザクションにメッセージの受信が含まれないため一貫性を保証できません。
	Never	指定不可	Message-driven Beanでは指定できません。
BMT	－	×	トランザクションにメッセージの受信が含まれないため一貫性を保証できません。

○：保証します
×：保証できません
CMT：Container-Managed Transaction
BMT：Bean-Managed Transaction

アプリケーション連携実行基盤の実行環境のベースとなる、EJB(Message-driven Bean)より呼び出す業務処理実行アプリケーションから、EJBアプリケーションを呼び出す場合のトランザクション管理種別、およびトランザクション属性によるメッセージ保証を、以下に示します。

業務処理実行アプリケーションから呼び出すEJBアプリケーション			
トランザクション		一貫性保証	説明
管理種別	属性		
CMT	Mandatory	○	呼び出し元のトランザクションに業務処理も含まれて一貫性が保証されます。 トランザクションが開始されていない場合はExceptionが返ります。
	Required	×	呼び出し元のトランザクションが開始されていない場合は、別トランザクションとして開始されるため保証できません。
	Supports	×	呼び出し元のトランザクションが開始されていない場合は、トランザクションが開始されていない状態になるため保証できません。
	RequiresNew	×	呼び出し元のトランザクションを中断するため保証できません。
	NotSupported	×	呼び出し元のトランザクションを中断するため保証できません。

業務処理実行アプリケーションから呼び出すEJBアプリケーション			
トランザクション		一貫性保証	説明
管理種別	属性		
	Never	×	例外が返るため保証できません。
BMT	－	×	呼び出し元のトランザクションと別トランザクションになるため保証できません。

○：保証します
×：保証できません
CMT：Container-Managed Transaction
BMT：Bean-Managed Transaction

メッセージ保証機能では、業務処理実行アプリケーションから呼び出すEJBアプリケーション(Entity Bean)の種類であるCMP(Container-managed persistence)や、BMP(Bean-managed persistence)での一貫性を保証します。

注意

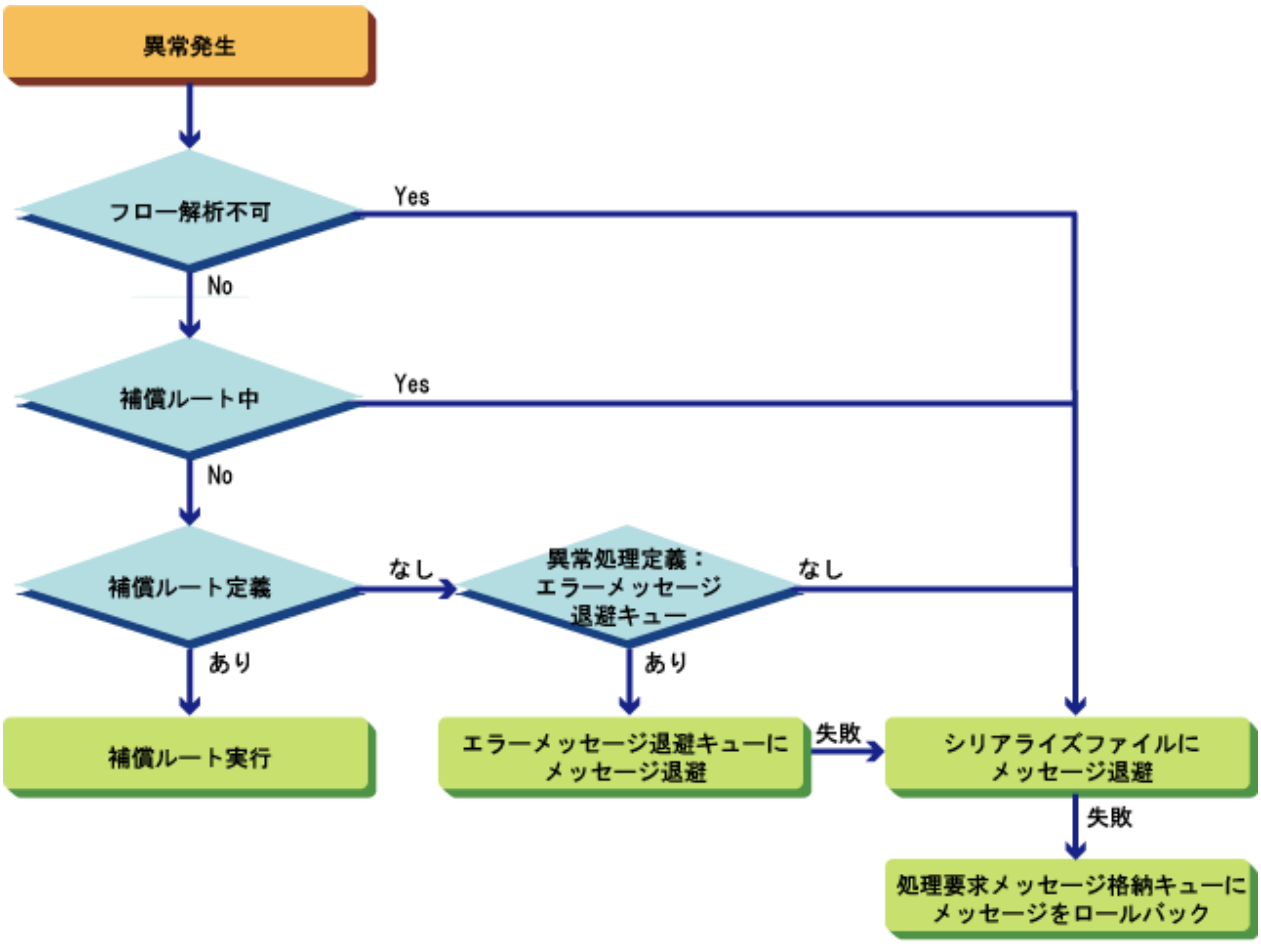
- アプリケーション連携実行基盤の実行環境のベースとなるEJB(Message-driven Bean)では、EJBのトランザクション管理種別は“Container”を、トランザクション属性は“Required”を指定してください。
- 業務処理実行アプリケーションから呼び出すEJBアプリケーション(Entity BeanやSession Bean)では、EJBのトランザクション管理種別は“Container”を、トランザクション属性は“Mandatory”を指定してください。

4.4.8 異常処理

アプリケーション連携実行基盤では、データベース処理の一時的なエラーなどの原因でアプリケーションに異常が発生した場合、後続のメッセージが処理できずにキューに滞り、業務全体が停止することを防止するための機能を提供します。
アプリケーションでのメッセージ処理に異常が発生した場合、フロー定義に設定したリトライ回数のリトライを行います。リトライを行っても問題が解決できない場合、業務処理がロールバックされメッセージ受信からワークユニットに設定したリトライ回数分、処理がリトライされます。しかし、リトライ後も異常が回復できない場合は、異常処理が実行されます。
異常処理では、以下の異常時の後処理を定義することができます。

- メッセージ退避機能
  異常となったメッセージをエラーメッセージ退避キューに送信する機能
- 補償ルート制御機能
  通過してきたルートを逆戻りして取り消しメッセージを送信する機能
- 異常処理の監視
  エラーメッセージ退避キューに格納されたメッセージをAPIを使用してメッセージをキューから取り出し、内容を確認する機能

異常処理では、上記の定義がされていない場合、または上記の処理を開始できない場合、メッセージをシリアライズファイルに退避します。更にシリアライズファイルへの退避にも失敗した場合は、ワークユニットを停止します。
なお、異常処理で行った処理結果はsyslogへ出力されます。
以下に異常処理の動作シーケンスを説明します。



注意

- Javaの業務処理実行アプリケーションを使用する場合、異常処理を正しく動作させるためには、ワークユニットで、EJBのトランザクション管理種別を“Container”トランザクション属性を“Required”で指定してください。EJBのトランザクション管理種別を“Container”以外、またはトランザクション属性を“Required”以外で指定した場合、異常が発生したメッセージは異常時の後処理が行われずに削除されます。
- 異常処理では、フロー定義に指定された情報をもとにエラー時の後処理を行います。そのため、メッセージに格納されたフロー定義情報が壊れているなどによりフロー定義情報の内容を解析できない場合は、異常処理定義で指定された後処理を行わずにメッセージをシリアライズファイルへ退避します。

また、“Interstage Business Application Server 運用ガイド(アプリケーション連携実行基盤編)”の“異常処理出口を使用したメッセージの復旧”を参考に、apfwconvfileコマンドを使用して異常となったメッセージ内容を修正し、apfwrecovmsgコマンドにより異常となったキューへメッセージを再送信することで、処理を復旧することもできます。
異常処理の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”の“非同期アプリケーション連携実行基盤編”の“異常処理”を参照してください。

4.4.9 メッセージ優先度の制御機能

メッセージに優先度を設定する機能です。業務処理を優先して行いたい要件が発生した場合や業務処理中に一定の条件を満たした場合などに、特定のメッセージの優先度を変更することができます。
優先度によるメッセージ処理順序の変更は、業務に関連付けられたキュー単位で行います。キューからメッセージを取り出す際、複数のメッセージが格納されている場合、優先度が高いメッセージから優先的に取り出されます。これにより、優先度が高いメッセージは優先度が低いメッセージよりも優先的に処理されます。優先度はメッセージごとに指定可能なため、同一フロー定義であっても異なった

優先度のメッセージが作成できます。
優先度は以下のレベルが指定できます。

- 高い
- 普通
- 低い

特に指定しない場合は、“普通”が指定されたものとして動作します。

優先度の指定方法を以下に示します。

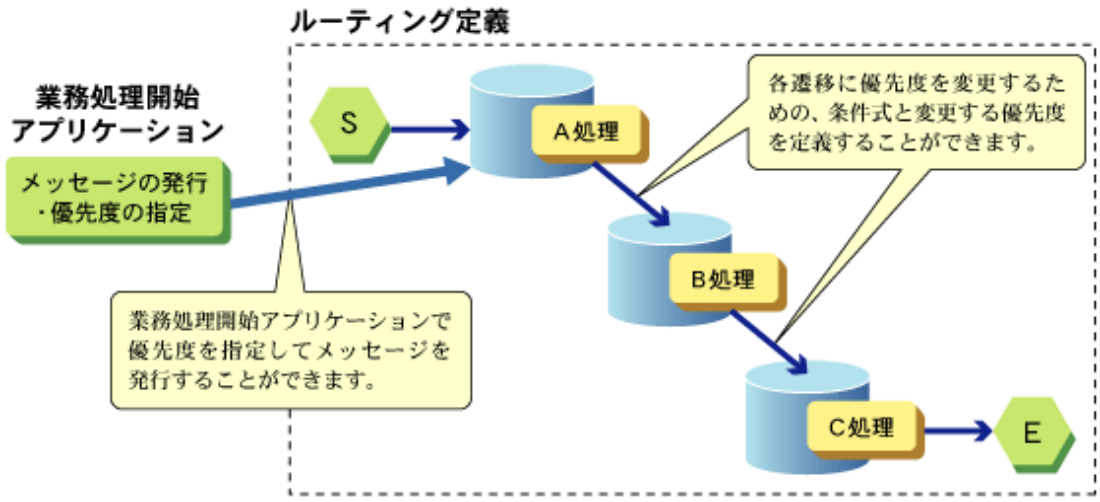
- 業務処理開始アプリケーションで優先度を指定

  メッセージを発行するAPIに優先度のレベルを指定してメッセージを発行します。
  業務処理開始アプリケーションからメッセージを発行する際、送信するメッセージの内容により優先度を選択することができます。
  優先度は、業務処理実行アプリケーションの処理結果により変更されるまで発行時に設定された優先度で動作します。

- 業務処理実行アプリケーションの出力結果により優先度を変更

  フロー定義ツールで設計するルーティングの各遷移(矢印オブジェクト)に優先度を変更するための条件式と変更する優先度のレベルを定義することができます。メッセージを各遷移から送信する際、メッセージの内容から条件式を評価し、真となるメッセージの優先度を指定された優先度に変更します。優先度は、一端変更されると次に再変更する条件に合致するまで変更された優先度で動作します。

優先度の指定方法における概念図を以下に示します。



なお、業務処理開始アプリケーションからメッセージを発行した順序で業務処理を行うことを完全に保証することはできません。
以下にメッセージの順序性について考慮すべき事項を説明します。

- 通常のメッセージ処理におけるメッセージの順序性

  アプリケーション連携実行基盤では、キュー内のデータをFIFO(First In First Out)として扱うため、メッセージ処理の多重度を1に指定し、並列実行しないことで業務処理開始アプリケーションからメッセージを発行した順序でメッセージを処理します。ただし、ルーティングされるメッセージの優先度を変更した場合、優先度の高いメッセージが低いメッセージを追い抜く可能性があります。

- 異常処理に退避されたメッセージ

  メッセージが異常処理に退避された場合、メッセージの順序性は保証されません。syslogに“FSP_INTS-BAS_AP1000”、“FSP_INTS-BAS_AP1001”、“FSP_INTS-BAS_AP1004”、“FSP_INTS-BAS_AP1007”、“FSP_INTS-BAS_AP1008”または“FSP_INTS-BAS_AP1019”のエラーメッセージが出力された場合、後から発行されたメッセージが、異常処理に退避されたメッセージを追い抜いて処理されます。

注意

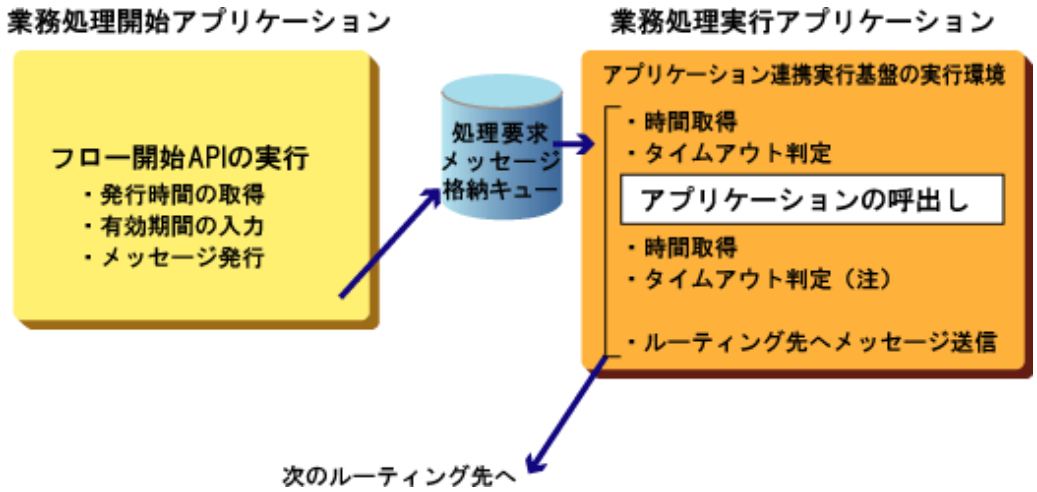
異常処理定義で指定した“総アクティビティ通過数”を超過した場合、当該メッセージに対する業務処理は、アプリケーション連携実行基盤では対処不能なレベルと判断し、メッセージが異常処理に退避されるため、後から発行されたメッセージが、“総アクティビティ通過数”を超えたメッセージを追い抜いて処理されます。

4.4.10 フローのタイムアウト機能

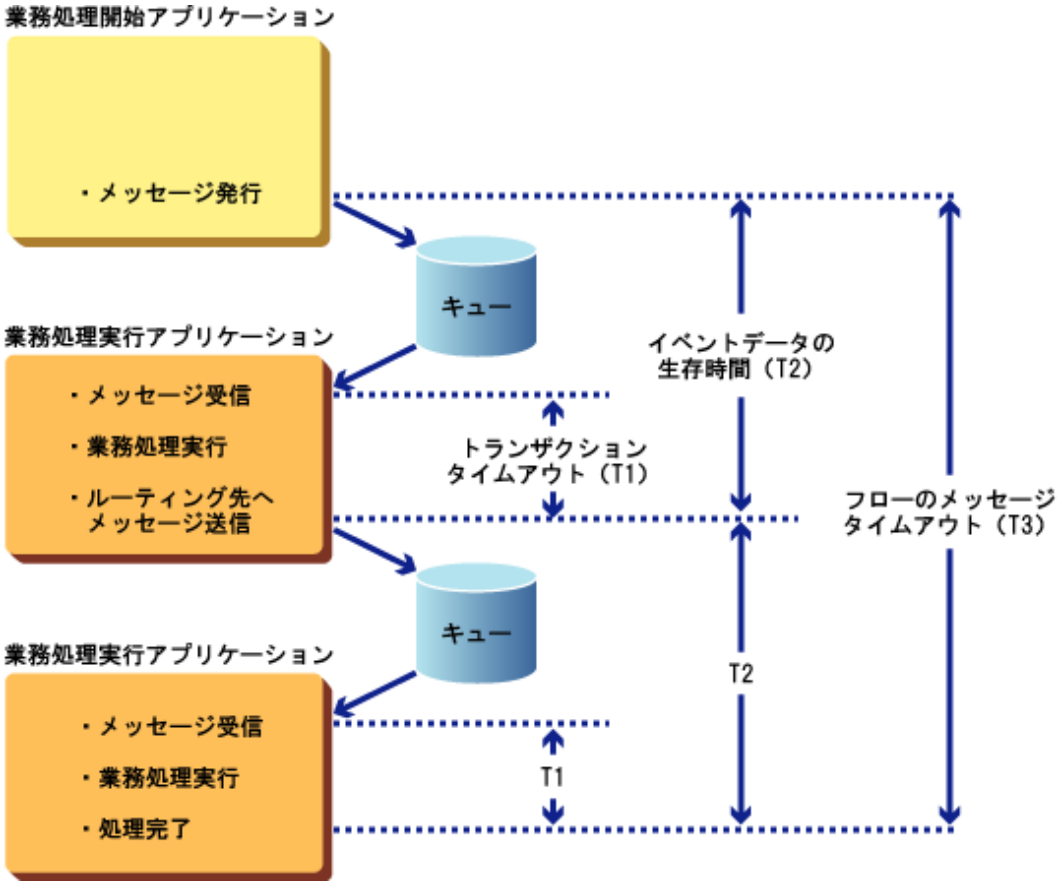
フローのタイムアウト機能は、メッセージ発行ごとに有効期間を設定し、有効期間を過ぎたメッセージ処理を自動的に停止することができる機能です。
タイムアウトしたメッセージは、異常処理定義で指定された後処理が実施され、エラーメッセージ退避キューなどへの退避が行われます。
フローのタイムアウト機能の方式を以下に示します。

- タイムアウト判定方式

  メッセージの発行時間からタイムアウト時間の経過を絶対時間で判定する方式です。例えば、有効期間を30秒で指定し、10時00分00秒でメッセージが発行された場合、10時00分31秒以降で行われるタイムアウト判定処理でタイムアウトとなります。そのため、ルーティング中に閉塞状態のキューなどが存在した場合など、ほとんどルーティングされていないメッセージに対してタイムアウトとなる可能性があります。
  有効期間の範囲は、メッセージの発行からフローの終点へメッセージが到達するまでの時間になります。メッセージ処理中にエラーが発生した場合の後処理や補償ルートのメッセージ処理中にタイムアウトの判定は行われません。
  また、タイムアウトの判定は、アプリケーション連携実行基盤の実行環境が行いますので、アプリケーションに処理論理を記載する必要はありません。



上図で示すように、アプリケーション連携実行基盤がメッセージを受信時、および次のルーティング先へメッセージを送信する直前にタイムアウト判定を行います。そのため、キューにメッセージが滞留している間、およびアプリケーションの処理中などでタイムアウト判定は行われません。タイムアウト発生時は、syslogへタイムアウトが発生した旨のメッセージを出力します。
なお、アプリケーション連携実行基盤で発生するタイムアウトは上記のメッセージタイムアウト以外に、トランザクションタイムアウトとイベントデータの生存時間タイムアウトがあります。



- トランザクションタイムアウト(T1)
  キューからデータを受信し、業務処理の結果によりCommit、またはRollbackするまでの時間です。
- イベントデータの生存時間(T2)
  イベントデータが生存する時間です。キューへデータを送信し、該当のデータがキューから受信され、Commitするまでの時間です。この値は、T1より大きい必要があります。
- フローのメッセージタイムアウト(T3)
  フロー開始から完了までの時間です。この値は、T2の総合計よりも大きい必要があります。
  T1の総和 ＜ T2の総和 ＜ T3

4.4.11 代行ルート制御機能

代行ルート制御機能は、送信先のキューへのメッセージ送信処理で異常が発生時、代行先として指定されたキューへ自動的にメッセージ送信を行います。これにより、従来、利用者がシェルやコンソールメッセージをもとにコマンド操作などで行っていた異常操作に関する運用負荷を軽減します。また、代行ルート制御で業務処理が停止することなくメッセージ処理が行われることで、後続メッセージでキューが満杯になることはなく、フロントのアプリケーションまで影響が波及することを防止できます。

- メッセージ送信処理の異常による代行ルートへのメッセージ送信

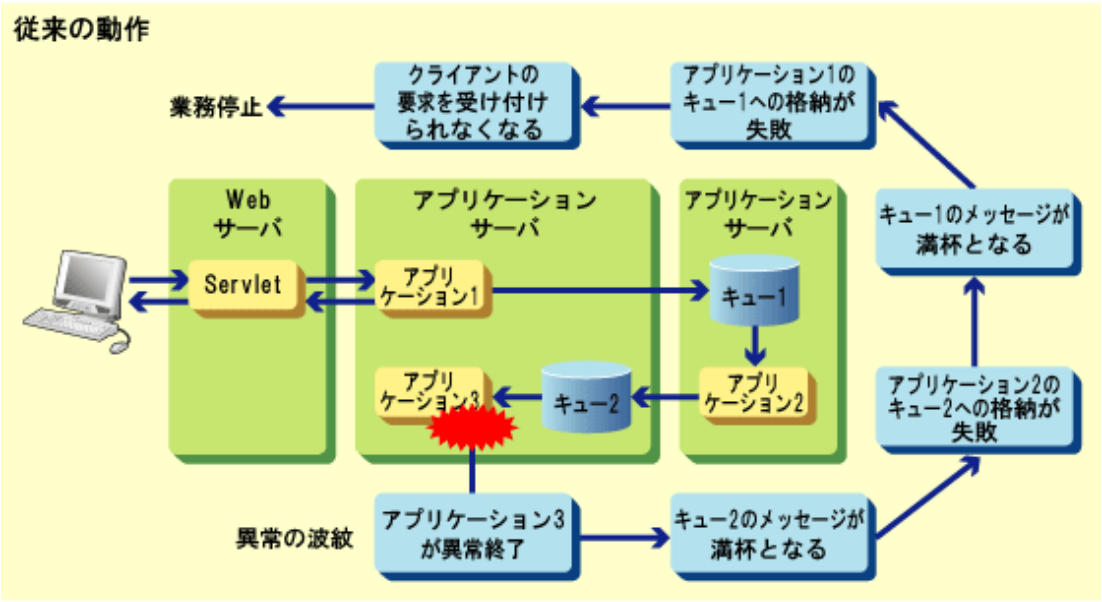
  ルーティング定義で指定された送信先のキューへのメッセージ送信時に送信先のキューに対応するイベントチャネルが停止している場合およびネットワーク異常などで送信処理が失敗した場合、送信先を代行ルートへ変更します。

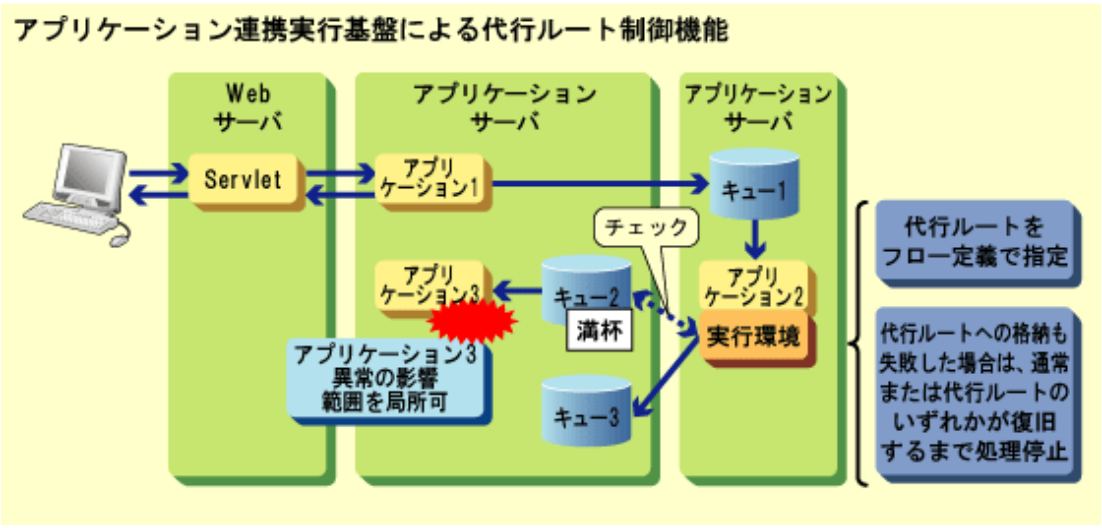
- メッセージ送信先のキューが閉塞状態(満杯)の場合に代行ルートへメッセージ送信

  メッセージを送信するキューの滞留メッセージが満杯になると、送信先を代行ルートへ変更します。これにより、キューに格納されているメッセージの状況をもとに流量制御や負荷分散を行うことが可能です。

以下に、代行ルートが設定されていない場合と設定されている場合での、異常発生時の影響範囲の違いを示します。

送信先キューの稼働状態を元にメッセージの振分けを自動で実行





注意

- Javaの業務処理実行アプリケーションの場合、代行ルートのキューも閉塞や満杯となり送信ができなくなった場合、代行ルート制御機能は、代行ルートのキューまたは通常ルートのキューのどちらかが利用可能となるまで待ち続け、どちらかのキューが利用可能となった契機で自動的に業務が再開されます。
- COBOLの業務処理実行アプリケーションの場合、代行ルートのキューも閉塞や満杯となり送信ができなくなった場合、異常処理に定義した対処が実行されます。
- 通常ルートへのメッセージ送信が成功し、その後、通常ルートの業務処理実行アプリケーションでの異常またはフローのタイムアウトにより、ルーティング処理を継続できない場合は、フロー定義ツールの異常処理定義で指定された後処理が実施され、エラーメッセージ退避キューへの退避などが行われます。apfwrecovmsgコマンドまたはapfwrecovfileコマンドで、メッセージをリカバリする場合には、再び通常ルートからルーティング処理が行われ、フロー定義ツールで代行ルートが定義されていても、代行ルートにメッセージが遷移することはありません。
  代行ルートへのメッセージ送信が成功し、その後、代行ルートの業務処理実行アプリケーションでの異常またはフローのタイムアウトにより、ルーティング処理を継続できない場合のリカバリについても、通常ルートの場合と同様に再び代行ルートからルーティング処理が行われます。

apfwrecovmsgコマンドまたはapfwrecovfileコマンドの詳細については、“Interstage Business Application Server リファレンス”を参照してください。

4.4.12 メッセージ消去時の退避機能 Linux64

揮発運用のイベントチャネルに対し、イベントチャネルの停止操作が行われた場合、メッセージをシリアライズファイルに退避します。

本機能により、揮発運用のイベントチャネルを用いたシステムにおいて不用意なイベントチャネル操作によるメッセージ消失を防止できます。また、緊急時にイベントチャネルの停止を行ってもシリアライズファイルより業務の復旧を行うことができることから、システムの信頼性が向上します。

4.5 運用ユーティリティ

エラーとなったメッセージの進行状態確認や業務処理開始アプリケーションからフローの開始を抑止する運用機能です。アプリケーション連携実行基盤を使用した業務の運用性を向上するために、以下の機能を提供します。

- メッセージトラッキング機能
  - メッセージトラッキング情報照会機能

    Interstage管理コンソール上で、複数サーバ間を流れるメッセージ処理のエラー状況を参照する機能を提供します。
  - メッセージトラッキング情報削除退避機能

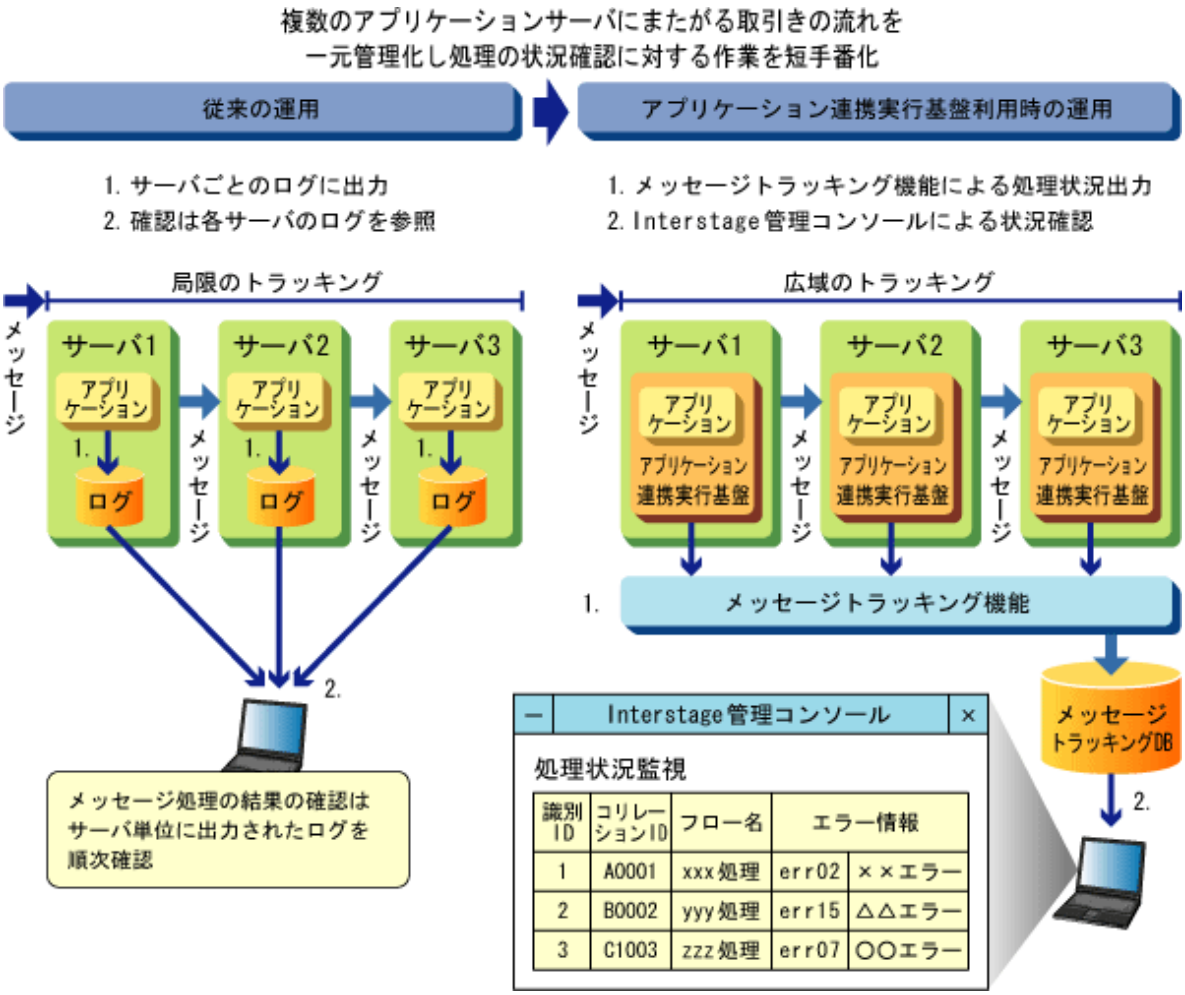
    メッセージトラッキング機能によって採取された情報は、メッセージトラッキングDBへ記録、蓄積されます。不要になったメッセージトラッキング情報を削除する機能を提供します。

    また、削除する際に、削除対象とするメッセージトラッキング情報をCSV形式のファイルとして退避する機能を提供します。
- フローの閉塞機能

  特定の業務(フロー単位)を閉塞する機能を提供します。閉塞している間は、業務処理開始アプリケーションで該当するフローを開始することができません。

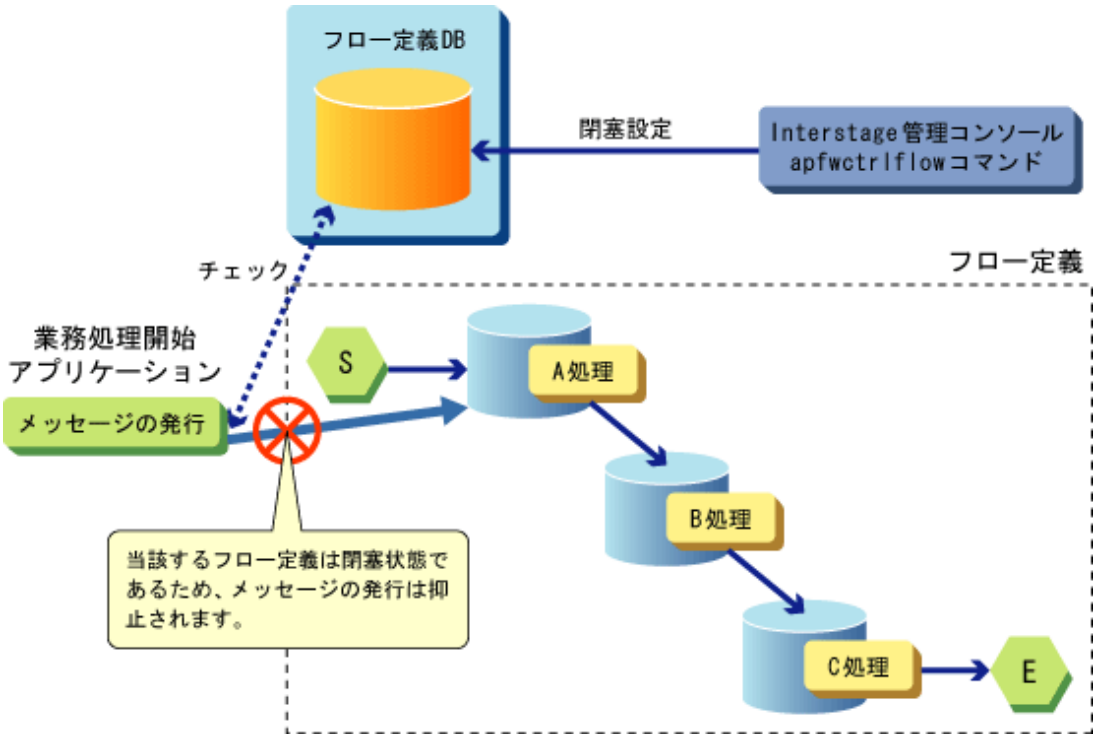
4.5.1 メッセージトラッキング機能

同一メッセージが複数サーバ間を流れるようなシステムを構築した際、メッセージの流れを確認するためには、各サーバ単位に出力されるログを参照しなければなりませんでした。ユーザは各サーバの処理状況を順に参照し、処理が正常に行われているかを確認する必要があり、状況確認を行うための負荷が高いという問題があります。異常発生時における状況確認においても、ユーザは同様の作業が必要となります。
メッセージトラッキング機能を使用することにより、ユーザはInterstage管理コンソールの画面上から、複数サーバ間を流れるメッセージ処理のエラー状況を一度に参照することができるため、ユーザの負荷が軽減されます。
以下に、メッセージトラッキングにおける従来の運用とアプリケーション連携実行基盤での運用を比較した概要図を示します。



4.5.2 フローの閉塞機能

フローの閉塞機能は、業務の計画的な運用をサポートします。フロー定義単位に業務の開始を抑止できるため、業務処理開始アプリケーションで複数の業務を開始している場合、特定の業務の開始だけを抑止することが可能になります。
フローの閉塞の設定および設定解除は、Interstage管理コンソールまたはapfwctrlflowコマンドを用いて行います。
Interstage管理コンソールを使用した閉塞の設定の詳細は、Interstage管理コンソールのヘルプを参照してください。apfwctrlflowコマンドの詳細は、“Interstage Business Application Server リファレンス”を参照してください。
以下にフローの閉塞の概念図を示します。



フローの閉塞中にメッセージを発行するAPIを呼び出した場合、フローの閉塞中を示すエラーによりAPIは異常復帰します。フローの閉塞中の場合に発行されるエラーは、以下の通り通知されます。

- Javaのアプリケーションの場合
  FlowExceptionで返却されます。業務処理開始アプリケーションでは、FlowExceptionに対し、getMessageメソッドまたはtoStringメソッドで取得できるメッセージ文字列からフローが閉塞中であることを識別できます。

第4部 共通機能編

第5章 ログ機能

アプリケーション連携実行基盤が出力するログについて説明します。

5.1 概要

システムの開発、運用および保守の各工程において、有用なログ機能を提供しています。ログ機能を利用することで、アプリケーション連携実行基盤の稼働状況および性能情報といったアプリケーション連携実行基盤を使用する上でのさまざまなログを採取することができます。
また、課金処理や障害時のデータ追跡などに利用するために、アプリケーションの処理状況(ユーザログ)を記録できるAPIを提供しています。アプリケーションでは、このAPIを使用することで、ユーザログを容易に、かつ、確実に書き込むことができます。業務テーブルとログテーブルが同じDBシステムである場合、トランザクションに連動し、一貫したデータ書込みが可能です。また、ユーザログはCSV形式での出力も可能ですので、表計算ソフトなどの各種ツールで分析することができます。

■ログの分類

ログには、アプリケーション連携実行基盤が出力を制御している標準ログとユーザがアプリケーションに記述することで出力されるユーザログの2種類があります。

- 標準ログ

  アプリケーション連携実行基盤が出力を制御しているログです。
  以下の2種類があります。

  ー システムログ

    アプリケーション連携実行基盤の稼働状況や、エラー状況をファイルに出力します。

  ー 性能ログ

    アプリケーションおよびアプリケーション連携実行基盤の処理に要した時間をファイルに出力します。

- ユーザログ

  ユーザが個別に定義したログです。標準ログ以外のログはすべてユーザログとなります。
  アプリケーションに記述することで使用できます。
  以下の2種類があります。

  ー 高信頼性ログ

    データベースの更新やアプリケーションの呼出しで指定するパラメタなど、データ操作の運用履歴を取得します。ログは、ユーザログテーブルと呼ばれる高信頼性ログを格納するための専用のテーブルへ出力します。

  ー 汎用ログ

    アプリケーション実行時に発生したエラー、動作の過程を表すインフォメーションおよび開発過程やトラブル調査などで用いるデバッグ情報のログです。汎用ログは、標準出力やファイルに出力します。

5.2 標準ログ

システムの運用状況をログとして出力するための機能です。システムログと性能ログの2つのログがあり、アプリケーション連携実行基盤が出力します。

■ログ出力の機能

ログ出力機能には、以下の機能があります。

- アプリケーションに記述することなく、アプリケーション連携実行基盤の動作ログを出力することができます。
- 出力レベルを設定することで、出力対象のログを絞り込めます。
- ログをキャッシュすることで、アプリケーションの実行に影響を与えずにログを出力できます。
- 出力先ファイルの自動分割が可能です。また、syslogまたはイベントログが利用できます。

- Webアプリケーションとサーバで動作するCOBOLのアプリケーション、C言語のアプリケーションおよびEJBアプリケーションを対応付けるID(コンテキストID)を出力することで、アプリケーション全体の処理の流れを把握することができます。
- 出力先やフォーマットなどの情報は、定義ファイル(ログ定義ファイル)に記述するため、プログラムを修正することなく設定を変更することができます。

ポイント

C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

■システムログ出力

システムログ出力機能は、アプリケーション連携実行基盤の稼働状況やエラー状況を出力する機能です。システムの起動、停止、および異常検出時にアプリケーション連携実行基盤よりメッセージが出力されます。
また、クライアントやサーバアプリケーションで使用するログとシステムログは、コンテキストIDにより対応づけを行うことにより一元化して管理することができます。

■性能ログ出力

性能ログ出力機能は、アプリケーション連携実行基盤内部での実行時間を出力する機能です。サーバアプリケーションの呼出しから復帰、非同期アプリケーション連携実行基盤におけるフロー開始APIや結果取得APIの発行から復帰、サーバアプリケーションの処理時間および結果受信キューにメッセージが格納された時刻など、アプリケーション連携実行基盤の処理開始から終点までの経過時間を出力します。

5.3 ユーザログ

利用者が、出力する情報やフォーマットをログ定義ファイルに定義し、ログを出力できます。
ユーザログには、高信頼性ログと汎用ログがあり、それぞれ使用するAPIが異なります。

■高信頼性ログと汎用ログの違い

高信頼性ログは、以下のような目的から取得します。取得する情報は、フォーマットの文字列のほかにバイナリによりログ取得ができます。高信頼性ログを使用することで、ユーザログと業務データ(Symfowareのデータベース)のトランザクションの一貫性を取ることも可能です。

- 業務処理の履歴として保管
- トラブル解析情報として使用
- 基礎データ、引継ぎデータとしての活用

汎用ログは以下のような目的から取得します。取得する情報は、文字列です。

- 性能測定
- デバッグ情報として利用
- インフォメーション(動作状況の確認)

5.3.1 高信頼性ログ

ここでは、高信頼性ログ機能の機能と運用パターンについて説明します。

■機能

高信頼性ログ機能には、以下の特長があります。

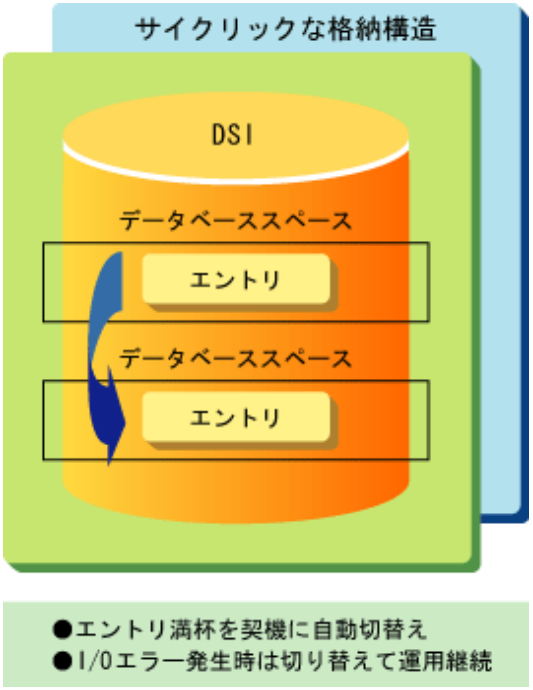
- エンドレスなファイル管理
- ユーザログのトランザクションの一貫性の保証
- データベースと同等の堅牢な書込み保証

- ユーザログの書込み順番の保証
- パーティショニングによるデータアクセスの高速化

◆エンドレスなファイル管理

サイクリックな格納構造によって、エンドレスにファイル管理ができます。この格納構造には、以下の特長があります。

- 高速挿入、循環使用に特化した格納構造
- 容量満杯や障害発生の場合はエントリを自動切替え
- エントリごとにI/O分散・危険分散が可能



◆ユーザログのトランザクションの一貫性の保証

高信頼性ログ機能は、以下の機能によって、トランザクションに一貫性のあるログ取得を実現します。

ユーザログの取得

ユーザログは、ロールバックした場合を含めてすべて取得します。これによってすべての業務処理結果の分析が可能になります。

ユーザログの書込み確定

ユーザログは、トランザクションのコミット、ロールバックなどのトランザクション終了時、またはログの確定APIの使用によって書込みが確定します。

トランザクション結果の自動出力

トランザクション結果を示す制御ログを自動出力します。これによってトランザクション結果の分析が可能になります。

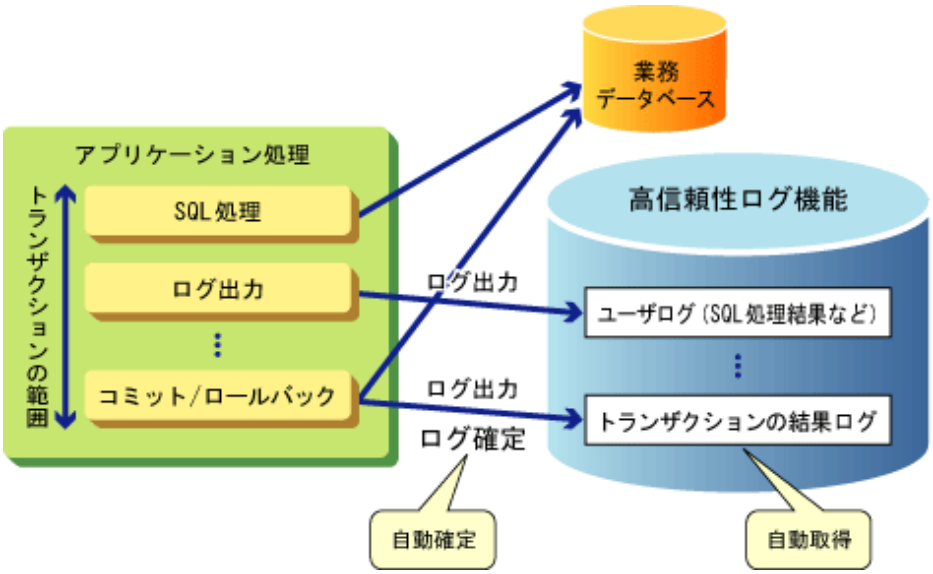
なお、システム構成(接続方法)によって、業務処理の結果情報を示すユーザログの取得方法には、以下の違いがあります。

	ユーザログの取得	ユーザログの書込み確定	トランザクション結果の自動出力
パターン1	○	○ コミットまたはロールバックが完了したタイミング	○
パターン2			
パターン3	○	○ Syncメソッドなどを利用してログを確定したタイミング	×

上記のパターン1～3は、“■運用操作のパターン”で説明しているパターン1～3を示しています。

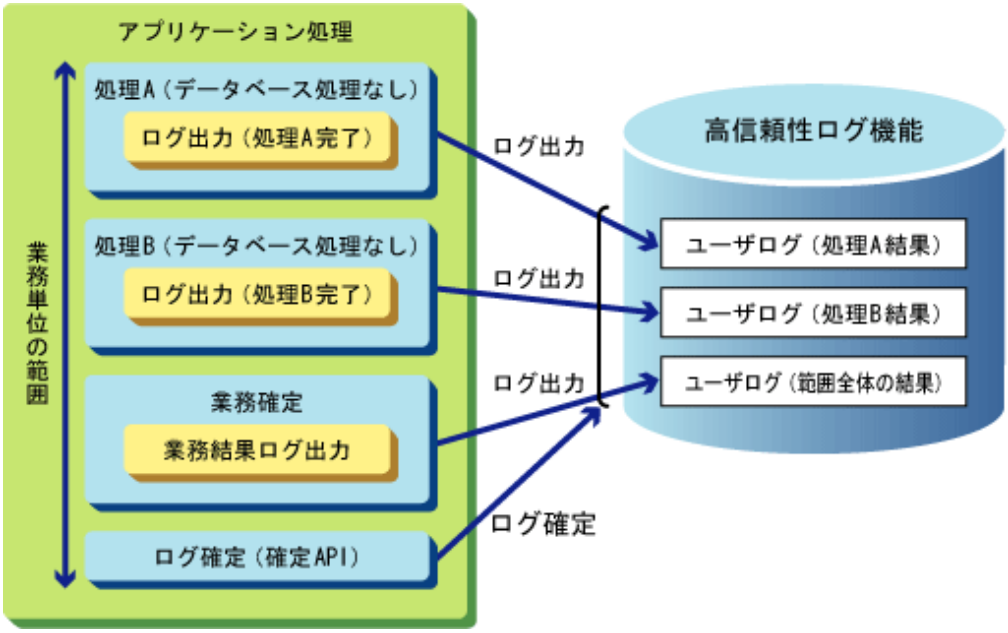
業務データベース処理があるデータベースサーバでのユーザログ取得

データベースサーバでのユーザログの取得は、パターン1およびパターン2の形態です。
パターン1では、業務データベースのトランザクション処理結果に連動して、トランザクション情報レコードログをシステムが自動的に出力します。
パターン2の場合、XA制御を利用して、トランザクション情報レコードログの自動取得やログの書込みを確定します(本バージョンでは、XA制御を利用した整合性保証機能はサポートしていません)。なお、出力したユーザログは、業務トランザクションの結果に関わらず、ユーザログテーブルへの書込みが保証されます。



業務データベース処理がないアプリケーションサーバでのユーザログ取得

アプリケーションサーバでのユーザログの取得は、パターン3の形態です。
この形態では、アプリケーションのログをユーザログとして採取していくもので、ログ確定によって採取したユーザログを保証していくものです。



◆データベースと同等の堅牢な書込み保証

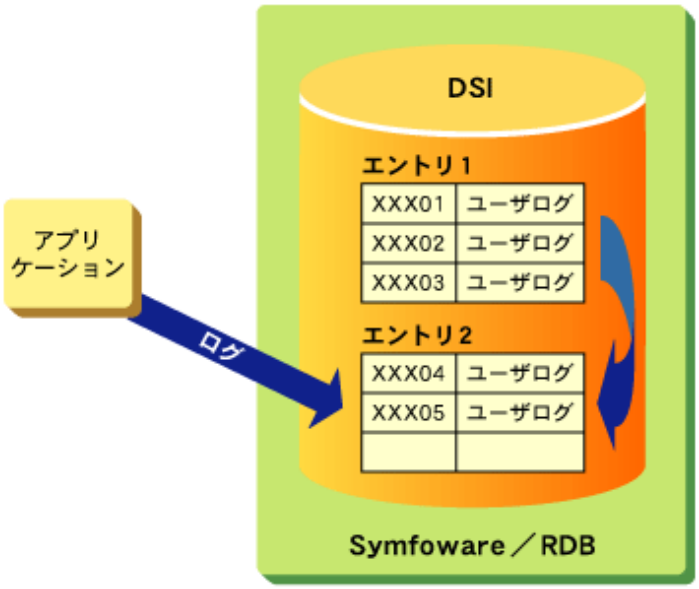
高信頼性ログの書込みは、データベース更新で培われた高い信頼性をベースとした技術で実装されています。
以下の場合も、トランザクションの完了またはログの書込み確定が行われたユーザログの書込みは保証されます。

- システムダウン、コネクション切断時は、テンポラリログより自動リカバリ

・ メディア障害時は、アーカイブログよりrdbrcvコマンドによってリカバリ

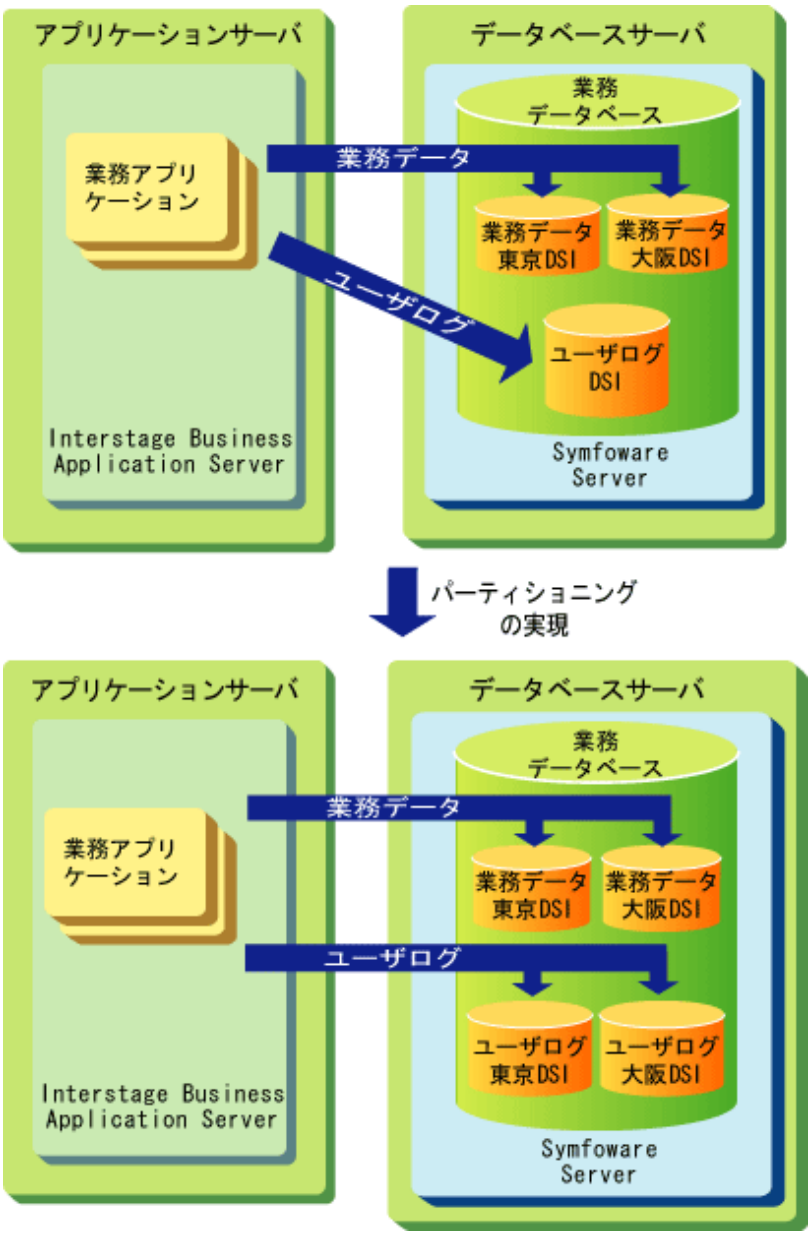
◆ユーザログの書込み順番の保証

ユーザログテーブルの論理定義で格納順番号を指定することによって、格納順番号をユーザログレコードに自動付加します。これをもとにソート処理を行うことで、書込み順にログを参照することが可能となります。ソート処理はユーザが行ってください。



◆パーティショニングによるデータアクセスの高速化

ユーザログテーブルは、格納構造定義により、規則に基づいてDSIを複数に分割することで、それぞれを独立させて運用することができます。これを、パーティショニングといいます。
パーティショニングを行うことで、ユーザログテーブルを分割した単位で独立かつ並行して運用することができるため、運用単位を小さく、かつ並行に処理することができます。細分化されたユーザログテーブルは内部的に独立して処理されるため、複数のトランザクションの同時実行性が高まり、ディスク入出力が分散されます。これにより、データアクセスの高速化を実現することができます。
業務データベースがSymfoware Serverのデータベースサーバで、ユーザログテーブルをパーティショニングした例を以下に示します。



■運用操作のパターン

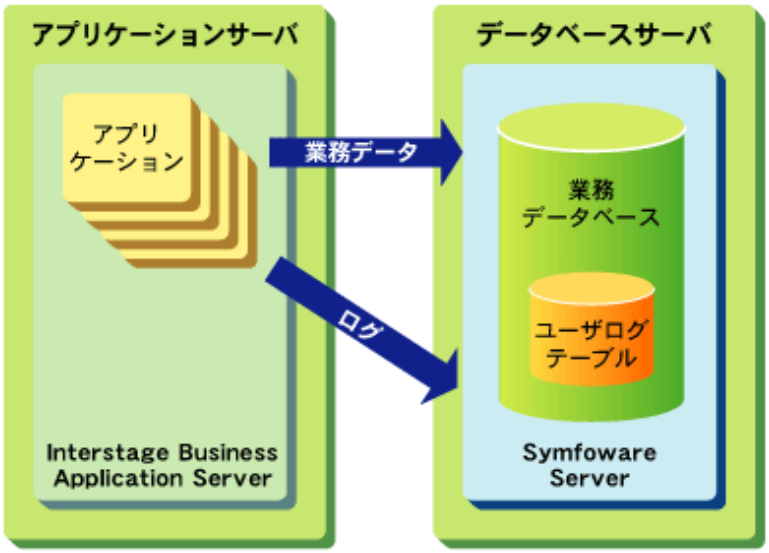
高信頼性ログ機能の運用には、システム構成によって、以下の3つのパターンが考えられます。

- パターン1:業務データベースがSymfoware Serverのデータベースサーバでのユーザログの取得
- パターン2:業務データベースが他社データベースのデータベースサーバでのユーザログの取得(Oracleのデータベースサーバなど)
- パターン3:業務データベース処理がないアプリケーションサーバでのユーザログの取得(HUBサーバ(注)など)

注) HUBサーバとは、業務システムの統合や拡張性・柔軟性・高可用性を実現するために、送信送受信業務代行、複数HUB連携、各種フォーマット変換などを行うためのサーバです。本章ではデータベースを配置しないサーバで、ユーザログの取得要件が高い代表的なサーバという位置付けで例として記述しています。

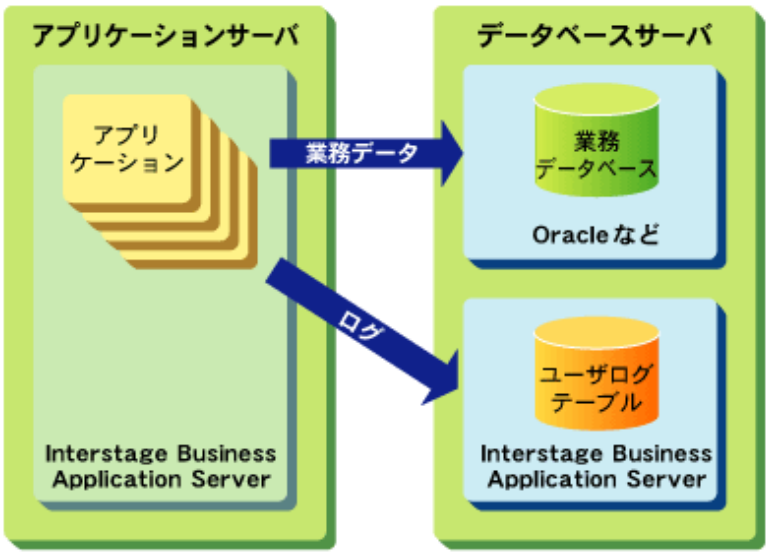
以下に、それぞれの運用パターンのシステム構成図を示します。

図5.1 パターン1：業務データベースがSymfoware Serverのデータベースサーバでのユーザログ取得



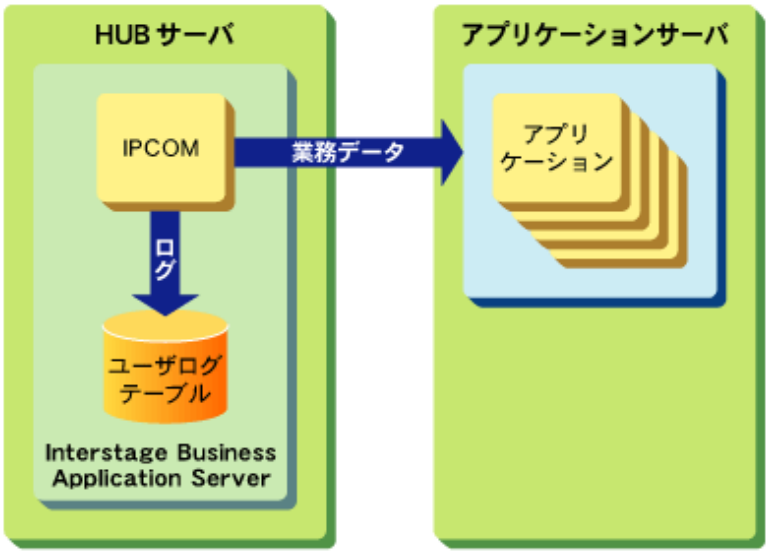
業務データベースがSymfoware Serverの場合は、Symfoware Serverの機能を利用して信頼性の高い高信頼性ログ機能の運用を行うことができます。

図5.2 パターン2：業務データベースが他社データベースのデータベースサーバでのユーザログ取得



なお、本バージョンでは、XA制御を利用した整合性保証機能はサポートしていません。

図5.3 パターン3：業務データベース処理がないHUBサーバなどでのユーザログ取得



5.3.2 汎用ログ

汎用ログでは、定義ファイル(ログ定義ファイル)により、以下の機能を実現します。

■出力レベルの設定

出力レベルを設定することで、重要度の高いログだけを出力し、不要なログの出力を抑制できます。

■ログのフィルタ

アプリケーション名、関数名、メソッド名または時刻などの特定の文字列を含むログだけを出力できます。

■ログのキュー

ログのキューを指定することで、アプリケーション処理の完了後に適切なタイミングでログを出力し、ログの出力によるレスポンス低下を軽減します。

■出力フォーマットの設定

ログ出力が要求された時刻や、ログ出力を要求したスレッド名など、出力するログの項目を出力フォーマットで設定できます。

■汎用ログの出力先

標準出力、標準エラー出力、ファイルおよびsyslogまたはイベントログが、汎用ログの出力先として指定できます。

注意

イベントログに出力できる文字コードは、Shift_JISコードのみです。

第6章 データベースアクセス管理機能

本章では、アプリケーション連携実行基盤上で構築したCOBOLまたはC言語のアプリケーションから業務データベースを利用する場合の処理を簡易化するデータベースアクセス管理機能について説明します。

注意

C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

6.1 概要

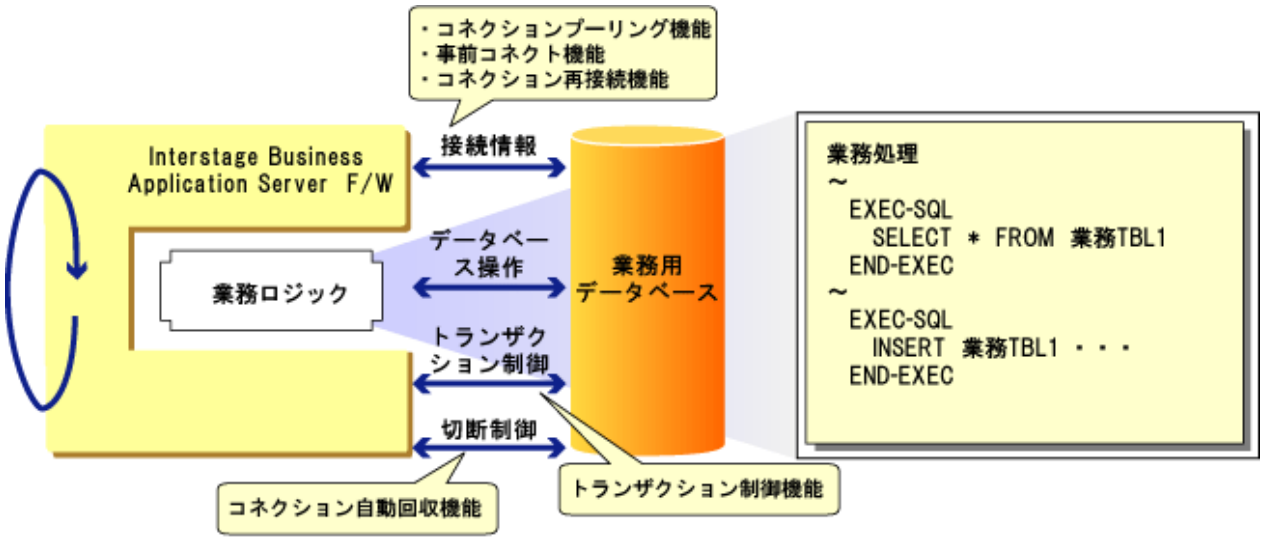
データベースアクセス管理機能では、以下に示すデータベース操作を行うためのコネクション管理機能やトランザクション制御機能を実行環境として提供します。

- コネクション管理機能
  - 事前コネクト機能
  - コネクションプーリング機能
  - コネクション再接続機能
  - コネクション自動回収機能
- トランザクション制御機能

また、データベースアクセス管理機能を使用するため、以下の定義やインタフェースが存在します。

- データベースリソース定義
- データベースアクセス定義
- アプリケーション開発言語とデータベースアクセスインタフェース
- アプリケーションの動作モード(プロセスモード/スレッドモード)

これにより、業務ロジックがデータベース操作を行う論理的な資源と、実際の操作の対象となる物理的なデータベース資源との関係を疎にすることが可能となり、システム形態の変更などに柔軟に対応することが可能となります。
業務アプリケーション開発者は、データベースアクセス管理機能を利用することによりデータベースアクセスのためのコネクション制御およびトランザクション制御を意識することなく、業務ロジックに専念して開発することができます。
以下にデータベースアクセス管理機能の位置づけとアプリケーションの開発者が作成する範囲を示します。



注意

- データベースアクセス管理機能を利用する場合は、アプリケーションでコネクションやトランザクションに関する操作を行わないでください。アプリケーションで、コネクションの操作やトランザクションの操作を行った場合の動作については保証されません。

また、アプリケーション連携実行基盤には、同期アプリケーション連携実行基盤と非同期アプリケーション連携実行基盤がありますが、業務用データベースを利用する方法は同一です。
したがって、業務ロジックは業務システムの変更に伴い、相互に柔軟に適用することができます。
なお、Javaのアプリケーションの場合、本機能は、EJBおよびJDBCの機能として実装しています。Javaアプリケーションからデータベースをアクセスする場合は、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。

ポイント

データベースアクセス管理機能を使用する場合と使用しない場合のアプリケーションでのアクセス方法の相違は以下の通りです。

- データベースアクセス管理機能を使用し業務データベースを利用する方式
  Interstage Business Application Serverが提供するデータベースアクセス管理機能を使用することにより、アプリケーション開発者は業務データベースを操作する場合に必要となる、データベースへの接続や切断などのデータベース制御を意識することなく、データベース操作のための埋め込みSQL文を記述することで容易にアプリケーションの開発を行うことができます。
- 利用者が独自に業務データベースを利用する方式
  アプリケーション開発者は、様々なデータベースシステムに対して、任意のインタフェースで業務データベースを利用するアプリケーションを開発することができますが、データベース制御のための論理を実装する必要があります。

業務用データベース操作の利用方式による差異を以下に示します。

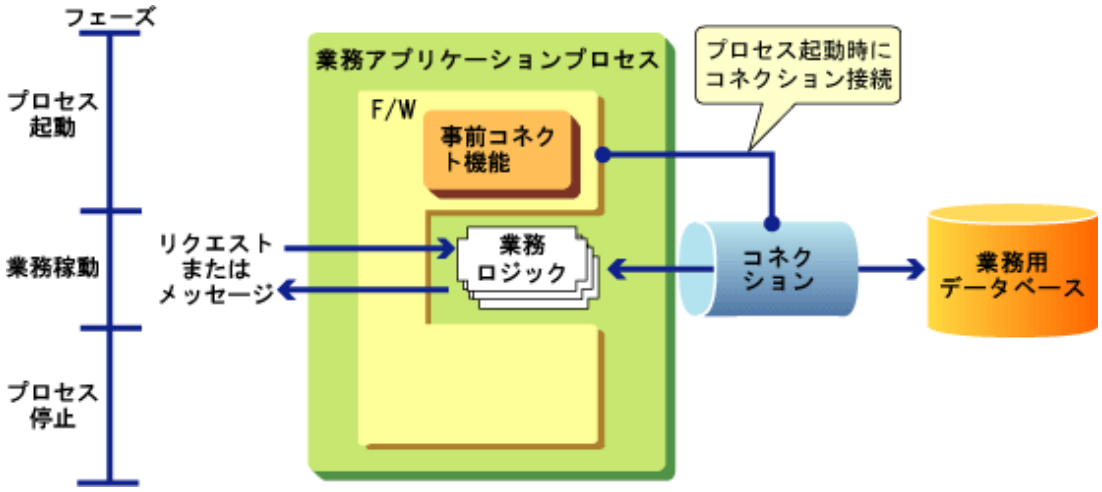
		データベースアクセス管理機能を利用し実現する場合	利用者が独自の方式で実現する場合
業務用データベースに対するサービス	データベースシステム	Symfoware Server Oracle	任意のデータベースシステム
	インタフェース	埋め込みSQL文インタフェース Symfoware Serverの場合 - Esql-c - Esql-COBOL Oracleの場合 - Pro*C - Pro*COBOL	任意のインタフェース
	1トランザクションでアクセス可能な業務データベース数	1	任意(データベースシステムの仕様に従います)
	データベース制御のための論理の実装	不要	必要
メッセージ保証機能 (非同期アプリケーション連携実行基盤の場合のみ有効です)		利用可	利用不可

6.2 コネクション管理機能

コネクション管理機能が提供する機能について説明します。

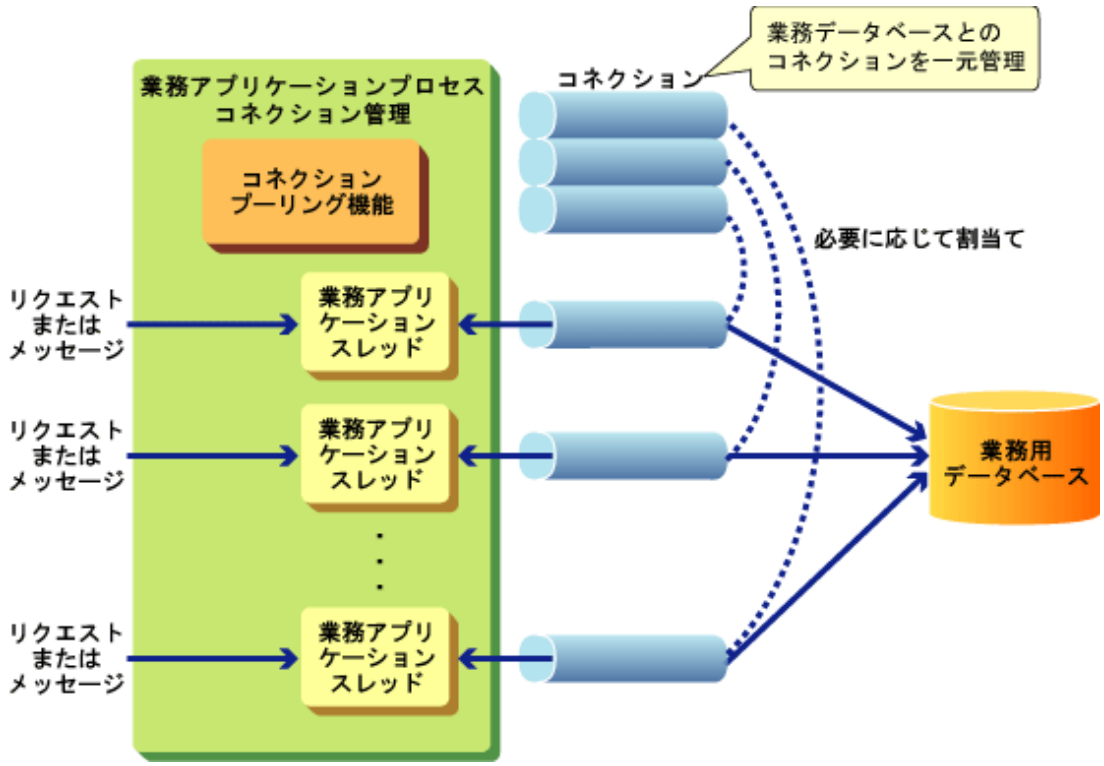
6.2.1 事前コネクト機能

アプリケーションが登録されたプロセスの起動時に、事前に必要とするコネクション接続の確保および必要に応じた最適な割当てを行うことにより、業務ロジック動作時にはデータベースに対するコネクション接続処理を行うことなく、初回アクセスを含め安定したデータベース接続レスポンスを保証します。



6.2.2 コネクションプーリング機能

アプリケーションが利用したコネクション接続をプーリングし管理する機能を提供します。本機能により常に有効なコネクション接続を監視し無駄なリソースの獲得を防止できます。



注意

- コネクションプーリング機能では、同時にアクティブな状態となるトランザクションで必要なコネクションを管理します。
- コネクションプーリング機能では、高負荷などの理由により、管理しているコネクション接続数を超えて必要となった場合には、“6.5 データベースアクセス定義”で指定した最大コネクション数まで新規に業務データベースとのコネクション接続を開設します。

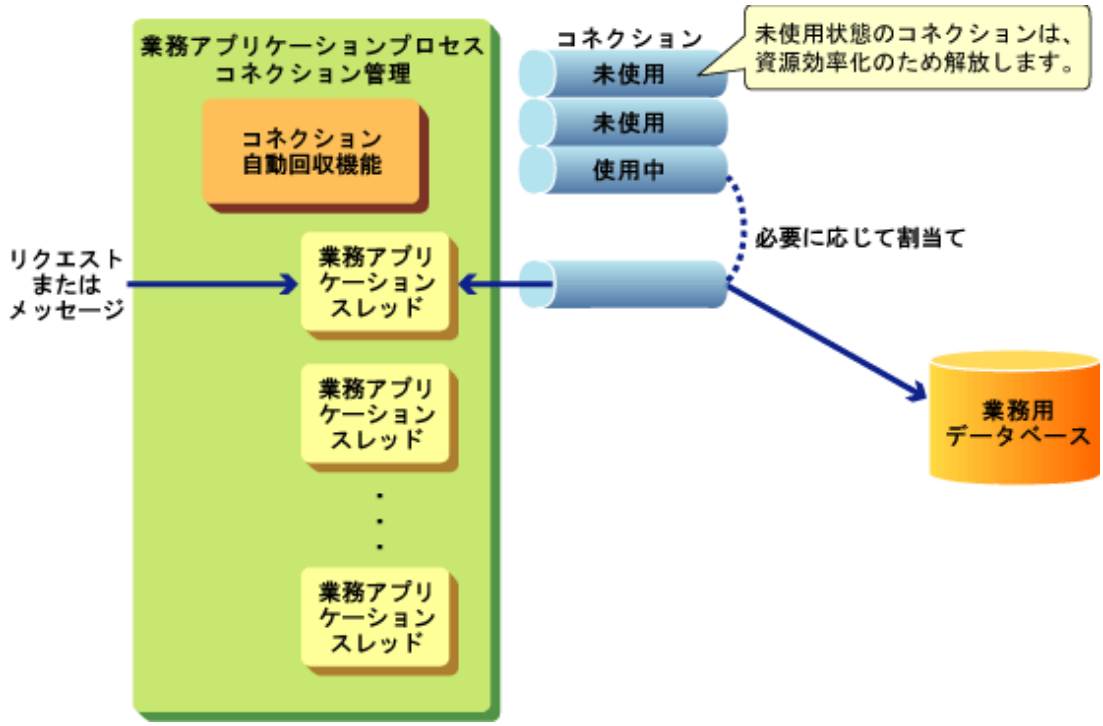
- コネクションの最小値は、事前コネクト機能の事前コネクト数に相当します。“データベースアクセス定義”の事前コネクト数にコネクションの最小値を指定します。

6.2.3 コネクション再接続機能

利用者のトランザクションが開始する際に、業務ロジックが使用するコネクション接続が有効であるかを自動的に判別し、通信異常やデータベースサーバ異常などの理由で無効となっている場合には、コネクションの再接続を自動的に行い、常に有効なコネクションを割当てます。
ただし、利用者のトランザクションが開始した以降にデータベース異常などが発生した場合は、利用者のトランザクションが失敗します。この場合には、トランザクションをリトライすることで、コネクション再接続を行い再開することが可能です。

6.2.4 コネクション自動回収機能

接続したコネクションの利用状況を監視し、利用者が設定した一定時間未使用な状態のコネクション資源の回収を自動的に行うことで、アプリケーションプロセスおよびデータベースの資源の効率化を実現します。



注意

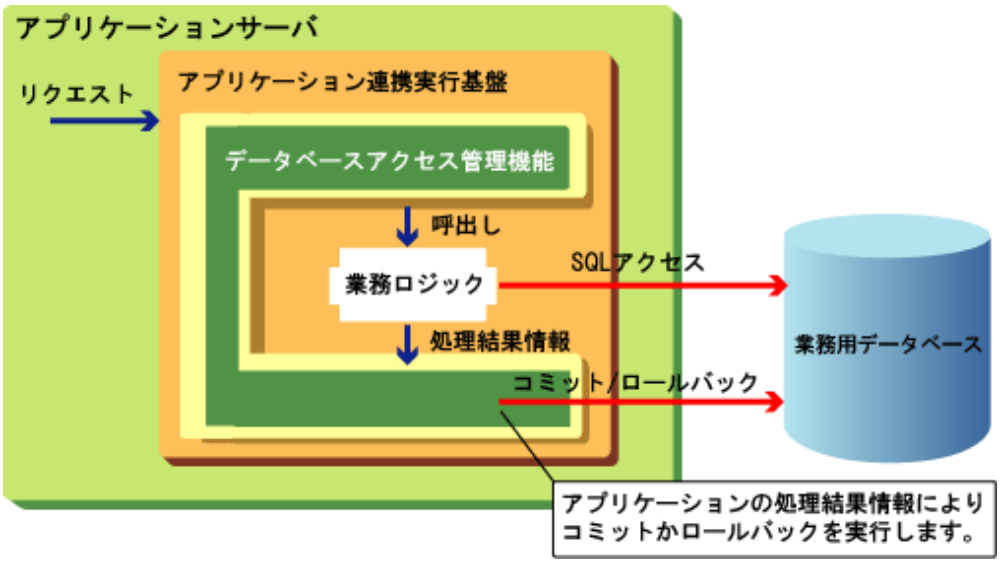
事前コネクト機能で接続したコネクションについては、コネクション自動回収機能の対象とはなりません。

6.3 トランザクション制御機能

利用者はアプリケーション連携実行基盤で提供された正常または異常といった復帰インタフェースを利用することにより、業務ロジック内で利用した業務用データベースへの操作についても同期してトランザクション制御を行うことができます。
アプリケーション連携実行基盤は、アプリケーションからの復帰インタフェースを元に以下の処理を行います。

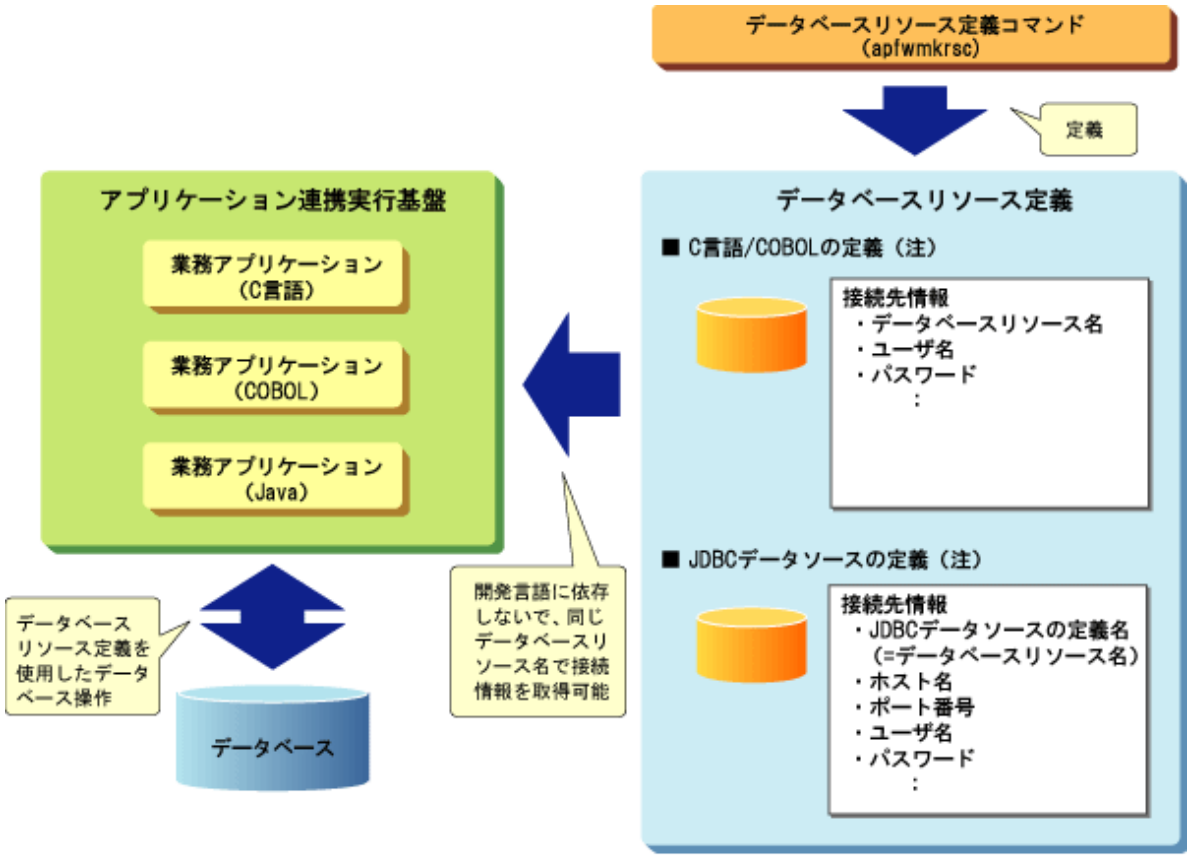
- 処理結果情報:0 業務用データベースへのコミット処理を行います。
- 処理結果情報:1 業務用データベースへのロールバック処理を行います。
  同期アプリケーション連携実行基盤のみ有効であり、非同期アプリケーション連携実行基盤では処理結果情報が、“0”の場合に同じです。
- 処理結果情報:2 業務用データベースへのロールバック処理を行います。

- 処理結果情報:3 同期アプリケーション連携実行の場合は、業務用データベースへのロールバック処理を行い、トランザクションをリトライします。非同期アプリケーション連携実行基盤の場合は、業務用データベースへのコミットおよびロールバック処理を行わずに、アプリケーションの再呼出しを行います。
- その他の処理結果情報:処理結果情報=2として扱います。



6.4 データベースリソース定義

データベースリソース定義は、アプリケーション連携実行基盤で動作する業務アプリケーションが、データベースアクセス管理機能またはJDBCインターフェースを利用してデータベースを操作する場合の接続先情報を管理します。業務アプリケーションが操作するデータベースの接続先情報をデータベースリソース定義として登録することにより、業務アプリケーションの開発言語(COBOL、C言語およびJava)に依存せず、同じデータベースを操作する場合には同一のデータベースリソース定義を利用することが可能です。
データベースリソース定義の登録は、データベースリソース定義コマンド(apfwmkrsc)で行います。登録方法の詳細については、“Interstage Business Application Server セットアップガイド”の“データベースリソース定義の登録”を参照してください。



注）データベースリソース定義コマンドは、C言語/COBOL用とJava用（JDBCデータソース）の両方に定義を行います。

以下にデータベースリソース定義に指定する主なパラメタを示します。詳細については、“Interstage Business Application Server リファレンス”の“apfwmkrsc”を参照してください。

- データベースリソース名
- 接続先のデータベースシステムの種別(SymfowareまたはOracle)
- データベースに接続するユーザ名
- データベースに接続するユーザのパスワード
- データベースサーバのホスト名
- データベースサーバのポート番号(注1)
- 接続先のデータベース名(注2)
- データソース名(注3)

注1) Symfoware Serverの場合は、Symfoware JDBCドライバのネーミングサービスのポート番号です。
Oracleの場合は、リスナのポート番号です。
注2) データベースシステムの種類や接続方法によりデータベース名ではない場合があります。
注3) Symfoware Serverの場合は、JNDIサービスプロバイダのネーミングサービスに登録したデータソース名です。
Oracleの場合は、データソース名は登録されません。

注意

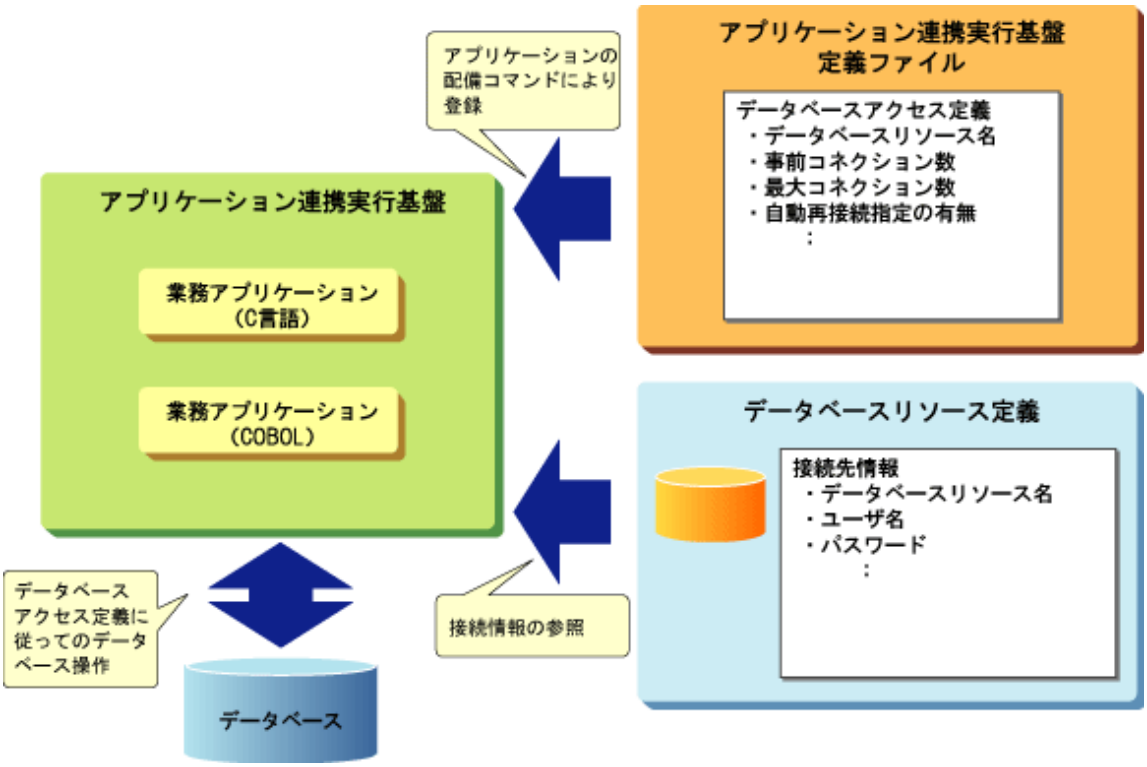
データベースリソース定義コマンドは、必ずJDBCデータソースの定義を行います。このため、COBOLまたはC言語だけを利用する場合においても、JDBCデータソースに関連するパラメタに値を指定してデータベースリソース定義を行う必要があります。指定値の詳細については、“Interstage Business Application Server リファレンス”の“apfwmkrsc”を参照してください。

6.5 データベースアクセス定義

データベースアクセス定義は、業務アプリケーションがデータベースアクセス管理機能を利用してデータベースを操作する場合に必要となる定義です。業務アプリケーションが操作するデータベースのデータベースリソース名や、データベースアクセス管理機能が提供する事前コネクト機能やコネクションプーリング機能の使用の有無などを、アプリケーションの配備単位に登録することが可能です。データベースアクセス定義の登録は、アプリケーションの配備コマンド(apfwdeploy)にデータベースアクセス定義を記載したアプリケーション連携実行基盤定義ファイルを指定して行います。登録方法の詳細については、“Interstage Business Application Server セットアップガイド”の“業務アプリケーションの配備”を参照してください。

注意

データベースアクセス定義は、COBOLまたはC言語を利用した業務アプリケーションからデータベースを操作する場合の定義です。Javaを利用した業務アプリケーションでデータベースを操作する場合は、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してデータベースをアクセスするための環境設定を行います。



6.6 アプリケーション開発言語とデータベースアクセスインタフェース

データベースアクセス管理機能を利用する場合、アプリケーション開発者は、使用する業務用データベースのデータベース種別に従い、以下のいずれかを選択し構築します。

	Symfoware	Oracle
COBOLアプリケーション	Esql-COBOL	Pro*COBOL
C言語アプリケーション	Esql-c	Pro*C

6.7 アプリケーションの動作モード(プロセスモード/スレッドモード)

アプリケーションの動作モードにはプロセスモードとスレッドモードがあります。アプリケーションの動作モードの違いにより、コンパイルおよびリンク時に指定するオプションやライブラリが異なります。コンパイルおよびリンク方法の詳細については、各データベースのマニュアルを参照してください。各動作モードとアプリケーション作成方法で留意する点について以下に示します。

動作モード		プロセスモード	スレッドモード
Symfoware	COBOL	ー	プロセスモードと同じです。 マルチスレッドの操作に関するインタフェースを意識する必要はありません。
	C言語	ー	プロセスモードと同じです。 マルチスレッドの操作に関するインタフェースを意識する必要はありません。
Oracle	COBOL	ー	プロセスモードと同じです。 マルチスレッドの操作に関するインタフェースを意識する必要はありません。 ただし、スレッド多重度は“1”となります。
	C言語	ー	プロセスモードと異なります。 マルチスレッドの操作に関するインタフェース（実行時コンテキスト）の使用を宣言する必要があります。なお、実行時コンテキストはアプリケーション連携実行基盤より取得します。

ー:留意点はありません。

注意

データベースアクセス管理機能を利用しない場合でも、非同期アプリケーション連携実行基盤の業務処理実行アプリケーションでは、以下の注意が必要です。

- フロー定義DBおよびメッセージトラッキングDBをOracleで構築している場合、かつ、COBOLで開発したアプリケーションでOracleデータベースを操作する場合は、実行時コンテキストを使用した業務アプリケーションを作成することはできません。
- データベースにSymfowareを使用している場合、業務処理開始アプリケーションで、フロー開始APIや結果取得APIを実行するとトランザクションはコミットされます。フロー開始APIや結果取得APIは、業務用データベースのトランザクションが開始されていない状態で実行してください。

注意

実行環境のOSがWindowsかつアプリケーション開発言語がC言語の場合、プロセスモードは使用できません。

第5部 ユーティリティ編

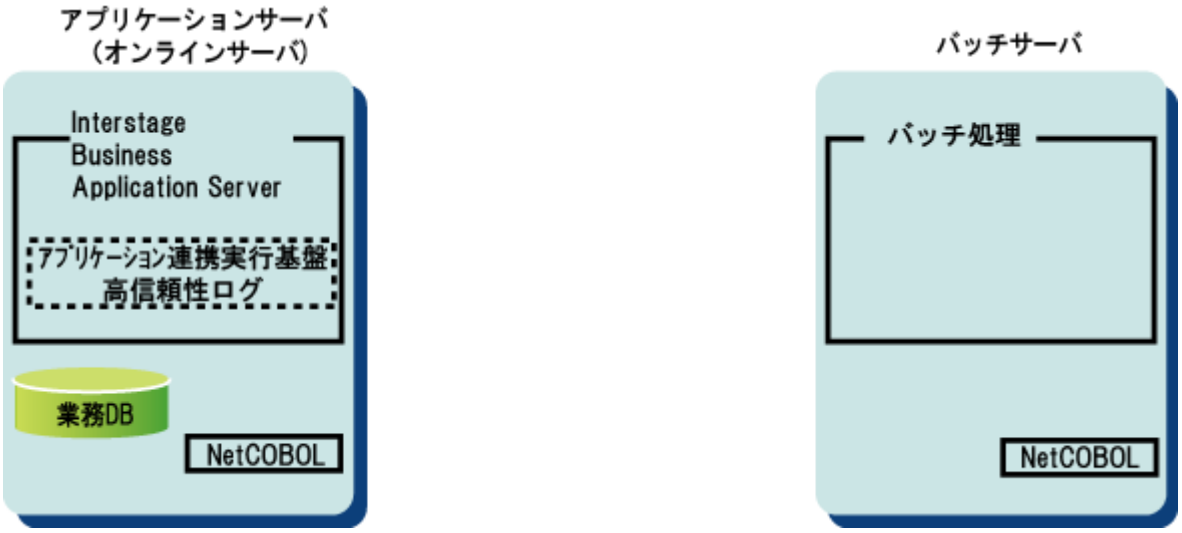
第7章 エクスポートユーティリティ EE

本章では、オンライン処理から高信頼性ログに蓄積されたログデータの退避処理、退避したログデータから必要なデータの抽出処理、およびバッチ投入などのユーザ処理実行等の一連の処理を制御する機能である、エクスポートユーティリティ機能について説明します。

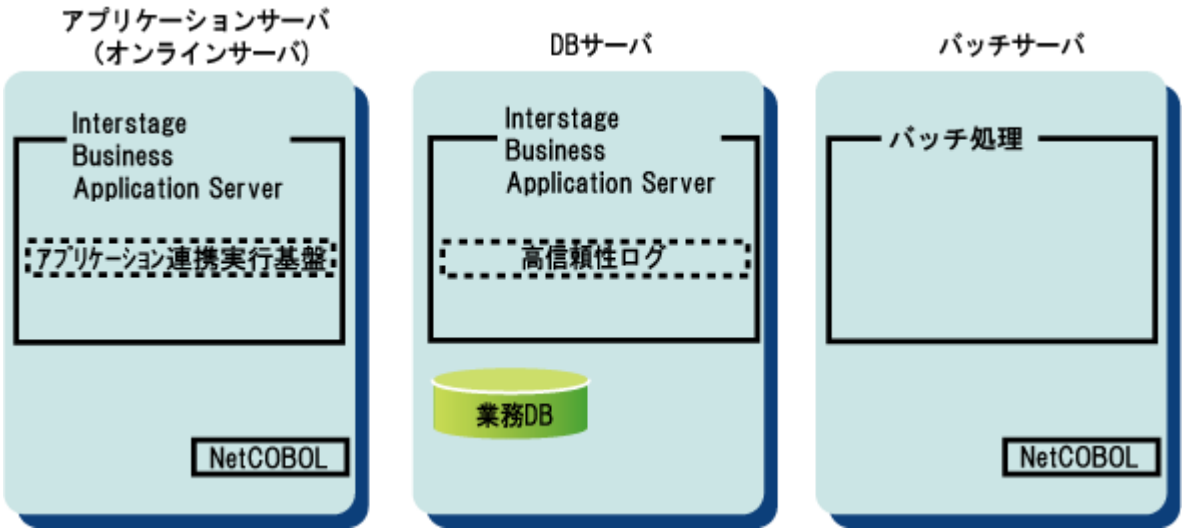
■運用形態

Interstage Business Application Serverは、以下の2つの運用形態をサポートしています。エクスポートユーティリティは、以下の運用形態で高信頼性ログがあるサーバ上で実行できます。

[パターン1] アプリケーションサーバとDBサーバを同一サーバで運用し、バッチサーバを別サーバで運用



[パターン2] アプリケーションサーバ、DBサーバおよびバッチサーバを全て別サーバで運用



7.1 概要

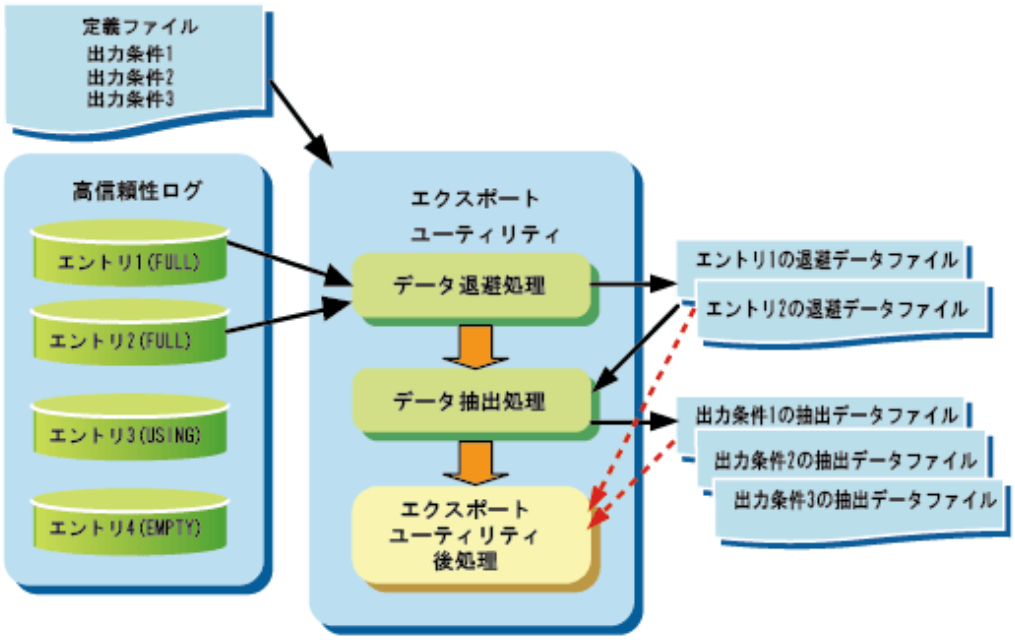
オンデマンド型のオンライン・バッチ連携業務を実現するために、以下の一連の処理を制御する機能をオンデマンド連携サービスのエクスポートユーティリティとして提供します。

- 高信頼性ログに蓄積されたユーザログの退避
- 退避したユーザログから必要なデータの抽出
- バッチ処理などのユーザ処理の実行

エクスポートユーティリティは、ユーザが作成する定義ファイルに設定された内容に従ってデータ退避およびデータ抽出を実行します。
定義ファイルはXML形式で記述します。詳細は“Interstage Business Application Server リファレンス”の“エクスポートユーティリティ定義ファイルリファレンス”を参照してください。
ユーザログを蓄積する高信頼性ログについては、“5.3.1 高信頼性ログ”を参照してください。

エクスポートユーティリティは、以下で構成されています。

- データ退避処理
  データ退避処理では、高信頼性ログに蓄積されたユーザログを退避します。退避したユーザログは、退避データファイルに出力されます。
- データ抽出処理
  データ抽出処理では、データ退避処理で退避したユーザログを抽出条件に応じて必要なデータ(ユーザログ本文)を抽出します。抽出データは、抽出データファイルに出力されます。抽出データファイルは、レコード順ファイルで出力することも可能です。
- エクスポートユーティリティ後処理
  エクスポートユーティリティ後処理では、退避データファイルおよび抽出データファイルをバックアップやバッチ処理への入力といったユーザ任意の処理が実行できます。



詳細は、“Interstage Business Application Server セットアップガイド”の“エクスポートユーティリティの設計と環境作成”を参照してください。

ポイント

バイナリデータをレコード順ファイルとして扱うことにより、COBOLアプリケーションにおいてCOBOL集団項目のデータを扱うことが可能です。

第8章 アプリケーション安定稼動機能 EE Solaris Linux64

Interstage Business Application Serverのアプリケーション安定稼働機能では、メモリ不足、ガーベジコレクションの発生を事前に検知、回避して安全かつ安定したJavaアプリケーションの運用を実現します。

8.1 機能概要

アプリケーション安定稼働機能はJ2EEのIJServerワークユニットにおいてプロセス多重で実行されるJavaVMを運用JavaVMと待機JavaVMに分けて扱い、それらを自動的に切り替えることでJavaアプリケーションの安全かつ安定した業務継続を実現します。例えば、Javaアプリケーションが無応答になった場合、Javaヒープ領域のメモリ不足の予兆を検出した場合、またはガーベジコレクションの予兆を検出した場合に、予兆を検出した運用中のJavaVM(運用JavaVM)の代わりに業務継続するために待機しているJavaVM(待機JavaVM)に自動的に切り替えることで、トラブルとなりうる事象を回避し安定的な業務運用を実現することができます。

以下にアプリケーション安定稼働機能の概要を示します。

1. JavaVMヒープ領域の管理によるスループットの保証
   Javaヒープ領域の不足やガーベジコレクションが発生した場合に、待機しているJavaVMにリクエストを振り分けることで、Javaアプリケーション実行中のガーベジコレクションを防止し、Javaアプリケーションのスループットを保証します。
2. アプリケーションタイムアウト監視による異常の局所化
   アプリケーション処理時間の長期化やデッドロックにより、Javaアプリケーションのタイムアウトの事象を素早く検知して、待機しているJavaVMにリクエストを振り分けたあと、アプリケーションタイムアウトが発生したJavaVMを再起動することで、部分的な不具合がシステム全体に影響しないよう異常を局所化します。
3. 運用コマンド
   IJServerワークユニットを停止せずに、ガーベジコレクションを実施する運用コマンドを提供します。例えば、夜間などの業務量が少ない時間帯に運用コマンドを実行し、事前にガーベジコレクションを実施することで業務量が多い時間帯に備えます。また、アプリケーション安定稼働機能の稼働状況を確認する機能も提供します。

ポイント

アプリケーション安定稼働機能を使用する場合、以下の仕様となります。

- システム全体でクッキーは使用できません。
- ログアウトやセションタイムアウト時のログイン画面への遷移時にクライアントにセションIDを返却できません。

8.2 運用形態

アプリケーション安定稼働機能はJ2EEのIJServerワークユニットでのみ運用可能です。また、Webサーバとアプリケーションサーバを同じサーバで運用する形態でのみ運用可能です。

以下にアプリケーション安定稼働機能が適用できるJavaアプリケーションの形態とInterstage Business Application Serverが提供する機能のアプリケーション形態を示します。

表8.1 アプリケーション安定稼働機能のJ2EEアプリケーション形態

アプリケーション種別	アプリケーションの形態	適用の可否 ○:適用可 ×:適用不可
Webアプリケーション	Servlet	○
	JSP	○
EJBアプリケーション	Session Bean	○
	Entity Bean	○
	Message-driven Bean	×
Webサービス	JAX-RPC	○

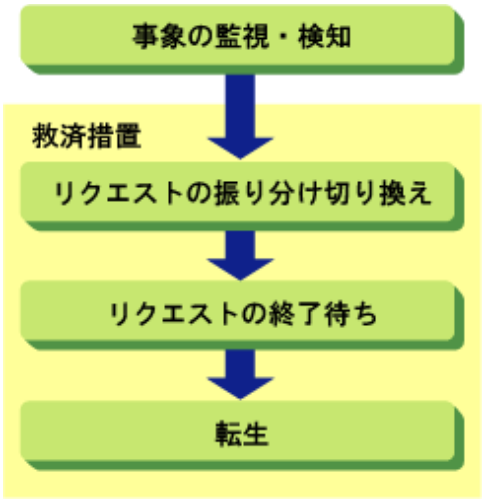
表8.2 Interstage Business Application Serverが提供する機能でのアプリケーション形態

アプリケーションフレームワーク	アプリケーションの形態	適用の可否 ○:適用可 ×:適用不可
アプリケーション連携実行基盤	同期アプリケーション連携実行基盤	×
	非同期アプリケーション連携実行基盤	×
Apcoordinator	Apcoordinator	○
オープンJavaフレームワーク	Struts	○
	Spring Framework	○
	iBATIS	○
	TERASOLUNA Server Framework for Java (Web版)	○
	TERASOLUNA Server Framework for Java (Rich版)	○
	TERASOLUNA Batch Framework for Java (IJServerワークユニット上での非同期ジョブ)	○
	TERASOLUNA Batch Framework for Java (JDK5.0またはJDK6上での動作する同期型ジョブ、および非同期ジョブ)	×

8.3 機能説明

アプリケーション安定稼働機能は、JavaVMのJavaヒープ領域が不足する予兆およびアプリケーションタイムアウトを監視し、どちらかの事象を検知した場合は、新たなリクエストを待機JavaVMに振り分けます。新たなリクエストを振り分けられた待機JavaVMは、以降運用JavaVMとして扱います。もとの運用JavaVMは処理中リクエストの終了を待ち合わせます。処理中リクエストが終了またはアプリケーションタイムアウトとなった運用JavaVMは、再起動またはガーベジコレクションを実施したあと、待機JavaVMとして扱います(転生)。これらの救済措置によりJavaヒープ領域の不足によるJavaアプリケーションのスループット低下の防止、およびJavaアプリケーションの異常の局所化を実現します。

以下にアプリケーション安定稼働機能の動作フローを示します。



また、アプリケーション安定稼働機能ではJavaVM動作中にガーベジコレクションの実行、待機JavaVM個数の動的変更、ユーザクラスの実行、アプリケーション安定稼働機能の稼働状況を確認できる運用コマンドを提供しています。

8.3.1 事象の監視・検知

アプリケーション安定稼働機能が監視する対象について下記に示します。

- Javaヒープ領域のメモリ予兆監視(メモリ予兆監視)
- アプリケーションタイムアウト監視

■Javaヒープ領域の予兆監視(メモリ予兆監視)

Javaヒープ領域の使用量を監視します。

Javaヒープ領域の使用量がしきい値を超過した場合に、リクエストを待機JavaVMに振り分けし、救済措置（JavaVM再起動またはガーベジコレクション）を実行します。

Javaヒープ領域のメモリ予兆監視が警告を出し始めた時点で、JavaVMの性能が著しく低下してアプリケーションのレスポンスが劣化し、JavaVMが致命的な不具合を起こす可能性があります。Javaヒープ領域のメモリ予兆監視によりメモリ不足を早期に検出し、待機JavaVMに新規リクエストを振り分けたあと、JavaVM再起動またはガーベジコレクションを行うため、リクエスト処理中にガーベジコレクションが発生しない運用を実現します。

しきい値については、メモリ予兆監視のしきい値を参照してください。

■アプリケーションタイムアウト監視

Interstage Application Serverのアプリケーション最大処理時間で設定された値をもとに監視します。

アプリケーション最大処理時間を超過した場合、待機JavaVMが存在する場合はアプリケーション最大処理時間を超過したJavaVMに対するリクエストを待機JavaVMへ振り分け、救済措置を実行します。

アプリケーション最大処理時間に“0”を設定した場合は、アプリケーションタイムアウト監視を行いません。アプリケーション最大処理時間の詳細は、“Interstage Application Server Interstage管理コンソールヘルプ”、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”を参照してください。

8.3.2 救済措置

救済措置とは、メモリ予兆監視やアプリケーションタイムアウト監視により、Javaアプリケーションおよび JavaVMの異常状態を検知した場合に、JavaVMを切り替えることで安全かつ安定した業務継続を実現し、異常が発生したJavaVMに対してJavaVMの再起動またはガーベジコレクション実行を行うことで新たな待機JavaVMとして転生する機能です。

下表に救済措置を示します。

救済措置の動作として、以下の4種類の動作の中から選択します。救済措置は定義ファイルに設定します。なお、アプリケーションタイムアウト監視により事象を検出した場合には、JavaVMの再起動が実行されます。

救済措置の名前	処理内容	監視する対象
JavaVM再起動	・待機JavaVMが存在する場合は、待機JavaVMへのリクエスト振り分け開始後に当該JavaVMを再起動。 ・待機JavaVMが存在しない場合は、直ちにJavaVMを再起動。	・アプリケーションタイムアウト監視 ・メモリ予兆監視
転生JavaVM再起動	・待機JavaVMが設定数存在する場合は、待機JavaVMへのリクエスト振り分け開始後に当該JavaVMを再起動。 ・待機JavaVMが存在しない場合は、待機JavaVMが出現するまで待機し、待機JavaVMへのリクエスト振り分け開始後に当該JavaVMを再起動。 ・待機JavaVMが設定数より少ない場合は、直ちに当該JavaVMを再起動。	・メモリ予兆監視

救済措置の名前	処理内容	監視する対象
リセット	・ 待機JavaVMが存在する場合は、待機JavaVMへのリクエスト振り分け開始後に当該JavaVMでガーベジコレクションを実行。 ・ 待機JavaVMが存在しない場合は、直ちに当該JavaVMでガーベジコレクションを実行。	・ メモリ予兆監視
転生リセット	・ 待機JavaVMが設定数存在する場合は、待機JavaVMへのリクエスト振り分け開始後に当該JavaVMでガーベジコレクションを実行。 ・ 待機JavaVMが存在しない場合は、待機JavaVMが出現するまで待機し、待機JavaVMへのリクエスト振り分け開始後に当該JavaVMでガーベジコレクションを実行。 ・ 待機JavaVMが設定数より少ない場合は、直ちに当該JavaVMでガーベジコレクションを実行。	・ メモリ予兆監視

定義の詳細については、“Interstage Business Application Server リファレンス”の“アプリケーション安定稼動機能定義リファレンス”を参照してください。

■メモリ予兆監視のしきい値

救済措置を始める契機として、事象を検知するためのJavaヒープ領域の使用量のしきい値を決めます。しきい値と救済措置は、Javaヒープ領域のヒープ領域およびPermanent世代領域それぞれ1個または2個設定します。救済措置はしきい値ごとに設定します。しきい値と救済措置の種類は、“Interstage Business Application Server リファレンス”の“アプリケーション安定稼動機能定義リファレンス”を参照してください。

しきい値と救済措置を2段階で設定する考え方

救済措置を設定する場合、“しきい値と救済措置の設定例”を元にしきい値と救済措置を調整します。

表8.3 しきい値と救済措置の設定例

	しきい値	救済措置
第1しきい値	待機JavaVMを活性化することができるように、低めのしきい値を設定します。(例:50%)	転生リセット
第2しきい値	新規リクエスト振り分け停止後、処理中アプリケーションがすべて完了するまでにガーベジコレクションが発生しないよう、しきい値を設定します。(例:80%)	リセット

上表の考え方は、次のとおりです。

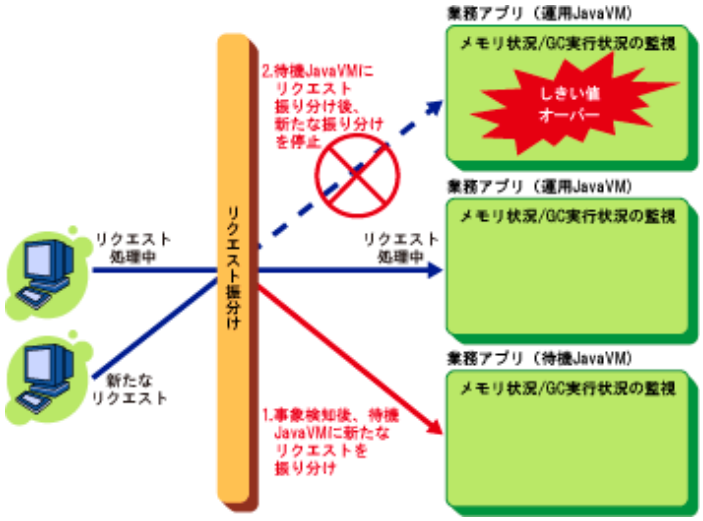
1. 第1しきい値を超過したら待機JavaVMが出現するまで待ち合わせてガーベジコレクションを実行する
2. 第1しきい値を超過しても待機JavaVMが出現せず、リクエスト切り換えができないまま第2しきい値を超過した場合、待機JavaVMの出現を待たずにガーベジコレクションを実行する

設定例の第1しきい値の救済措置では、待機JavaVMが出現するまで待ち続けます。待機JavaVMが出現しない場合には、第1しきい値を超えたJavaVMは何時までも転生せずに待機JavaVMとして利用することができません。このような状態を防ぐために、第2しきい値を設定し、第2しきい値の救済措置としては待機JavaVMの出現を待たないように設定することにより、待機JavaVMを出現させ、新たな事象に備えるように考えます。

第1しきい値の救済措置として“転生JavaVM再起動”、第2しきい値の救済措置として“JavaVM再起動”を設定することもできます。

8.3.2.1 リクエストの振り分け切り換え

運用JavaVMにおいてリクエスト処理中に、メモリ予兆監視およびアプリケーションタイムアウト監視により事象を検知した場合、待機JavaVMにリクエストの振り分けを行い、事象を検知した運用JavaVMへの新規リクエストの振り分けを停止します。



8.3.2.2 リクエストの終了待ち

事象を検知した運用JavaVMにおいて、新規リクエストの振り分け停止後にアプリケーションの終了を待ち合わせます。セション管理していない場合は、アプリケーション終了後、JavaVMの再起動またはガーベジコレクションを実行します。セション管理している場合は、セションが存在しない状態になるまで待ち合わせたあと、JavaVMの再起動またはガーベジコレクションを実行します。アプリケーション最大処理時間を経過してもアプリケーションが終了しない場合はJavaVMを再起動します。

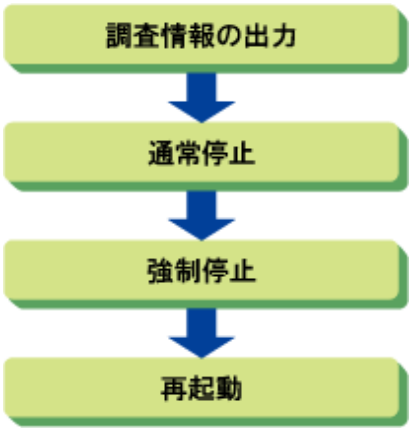
ServletアプリケーションまたはSTATEFUL Session Beanが配備されている場合、すべてのセションが無効になるまで待ち合わせを行います。STATEFUL Session Beanではremoveクラスを呼び出してSession Beanのインスタンスを消去していない場合、無通信監視時間（デフォルト30分）を超過するまで待ち合わせを行います。無通信監視時間の詳細は、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”を参照してください。

8.3.2.3 転生

リクエストの振り分け切り換え後、メモリ予兆監視およびアプリケーションタイムアウト監視により事象を検知した運用JavaVMは、救済措置の設定に従って待機JavaVMに転生します。運用JavaVMから待機JavaVMに転生することにより新たな事象発生に備えます。

8.3.2.3.1 JavaVMの再起動

救済措置が“JavaVM再起動”または“転生JavaVM再起動”の場合、事象を検知した運用JavaVMは次の流れで再起動して待機JavaVMへ転生します。



1. 調査情報の出力
   アプリケーションや動作環境の問題を確認するため以下の情報を出力します。
   - デッドロック情報
   - スレッドダンプ情報

2. 通常停止
   実行中のJavaアプリケーションの終了を待ってJavaVMを通常停止します。

3. 強制停止
   JavaVMの通常停止ができない場合に限り、JavaVMの強制停止を行います。

4. 再起動
   JavaVMの通常停止および強制停止がリトライカウントに達するまで、再起動を行います。リトライカウントに達した場合は、正常運用中のJavaVMも含めて強制停止します。
   JavaVMの再起動の回数は、[Interstage管理コンソール] > [Interstage Application Server] > [システム] > [ワークユニット] > (ワークユニット名) の [環境設定] > [ワークユニット設定] > [リトライカウント]で設定します。

8.3.2.3.2 ガーベジコレクション実行

救済措置が“リセット”または“転生リセット”の場合、事象を検知した運用JavaVMは待機JavaVMとなるためにガーベジコレクションを実行します。ガーベジコレクションを実行して待機JavaVMに転生することにより、リクエスト処理中にガーベジコレクションが発生しない運用を実現します。

■ガーベジコレクション実行後もメモリ使用量が改善しない場合

ガーベジコレクションを実行してもメモリ使用量がしきい値を超過する場合は、メモリリークの可能性があると判断しJavaVMを再起動します。JavaVM再起動時には次のエラーメッセージを出力します。

```
FSP_INTS-BAS_AP: WARNING: 12080: Because the memory usage exceeds the threshold after executing FullGC, the JavaVM is reactivated. WU=s*, pid=t*, instanceNo=u*, generationType=v*, max=w*, used=x*, thresholdIndex=y*
```

ガーベジコレクション実行後もメモリ使用量がしきい値を超過する場合の動作は、定義ファイルで変更することができます。詳細は、“Interstage Business Application Server リファレンス”の“アプリケーション安定稼働機能定義リファレンス”を参照してください。

8.3.3 運用コマンド(apfwctrlapl)

アプリケーション安定稼働機能の環境設定後、アプリケーションの運用中にアプリケーション安定稼働機能が提供する運用コマンド(apfwctrlapl)を使用することができます。

apfwctrlaplでは以下の機能を提供します。

- ガーベジコレクションの実行
  ガーベジコレクションを実行します。夜間などの業務量が少ない時間帯に本コマンドでガーベジコレクションを実行し、以後の業務量が多い時間帯に備えることができます。

- 待機JavaVM個数の設定
  JavaVMを停止せずに待機JavaVM個数を動的に変更することが可能です。IJServerワークユニットを停止した場合は、変更した待機JavaVM個数はリセットされます。
  待機JavaVM数を0でJavaVMを起動し、ダミーリクエストを処理させることでアプリケーション実行時に行われるコンパイルを事前に実施したあと、運用JavaVMの一部を待機JavaVMにする(待機JavaVM数を変更する)ことで、待機JavaVMにリクエストが割り振られた時にアプリケーションを安定して動作させることができます。

- ユーザクラス実行
  アプリケーション運用中にユーザクラスを実行する機能を提供します。ユーザクラスには、JavaVM再起動で初期化されてしまうメモリ情報を書きだす(DBキャッシュの更新など)アプリケーションを作成して実行することで、JavaVM再起動によるデータ消失を防止することができます。

- JavaVM状態の一覧出力
  JavaVMの状態一覧を出力する機能を提供します。アプリケーション安定稼働機能の設定時およびアプリケーションの運用中に運用JavaVMの数、待機JavaVMの数を確認する時に使用します。

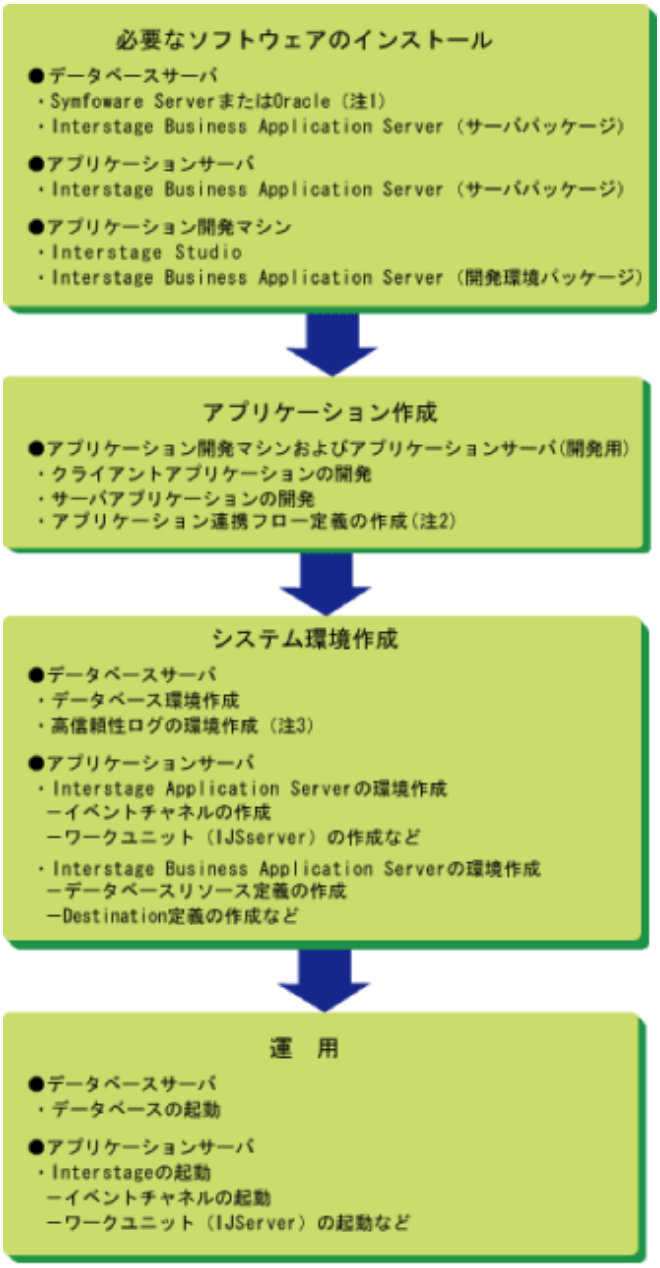
- リクエスト振り分け状況の一覧出力
  リクエスト振り分け状況の一覧を出力する機能を提供します。
  新規リクエストの振り分けを停止しているJavaVMを確認する時など、リクエストの振り分け状況を確認する時に使用します。

詳細は、“Interstage Business Application Server リファレンス”の“apfwctrlapl”コマンドを参照してください。

第6部 導入から運用まで

第9章 導入から運用までの流れ

本章では、アプリケーション連携実行基盤の導入から運用までの概要について説明します。
アプリケーション連携実行基盤を利用したシステムの環境作成は以下の手順で行います。



注1）業務用データベースを使用する場合
注2）非同期アプリケーション連携実行基盤を使用する場合
注3）高信頼性ログを使用する場合

■必要なソフトウェアのインストール

データベースサーバ、アプリケーションサーバ、およびアプリケーション開発コンピュータごとに、必要な製品をインストールしてください。

- データベースサーバ

  ー データベース製品のインストール

    以下の場合、Symfoware ServerまたはOracleをインストールします。インストール方法については、各製品の手順に従ってください。なお、データベース製品は本製品に同梱されていません。

    - 業務用データベースを使用する場合
    - 非同期アプリケーション連携実行基盤を使用する場合（Symfoware Serverのみ使用可能）EE

    なお、フロー定義DBおよびメッセージトラッキングDBは、高信頼性ログServer機能をインストールすることで、本製品内に同梱されているSymfoware/RDBに格納することができます。

  ー Interstage Business Application Serverのインストール

    以下の機能を使用する場合、Interstage Business Application Serverをインストールします。インストール方法の詳細は、“Interstage Business Application Server インストールガイド”を参照してください。

    - 高信頼性ログのデータ
    - フロー定義DBまたはメッセージトラッキングDBを本製品内に同梱されているSymfoware/RDBに格納する場合EE

- アプリケーションサーバ

  ー Interstage Business Application Serverのインストール

    “Interstage Business Application Server インストールガイド”を参照してください。

  ー NetCOBOLのインストール

    サーバアプリケーションをCOBOLで運用する場合にインストールします。インストール方法については、各製品の手順に従ってください。なお、データベース製品は本製品に同梱されていません。

- アプリケーション開発コンピュータ

  ー Interstage Studioをインストールする場合

    Interstage Studioのインストールガイドを参照してください。

  ー Interstage Business Application Serverの開発環境パッケージをインストールする場合

    “Interstage Business Application Server 開発環境パッケージインストールガイド”を参照してください。

  ー NetCOBOLのインストール

    サーバアプリケーションをCOBOLで運用する場合にインストールします。インストール方法については、各製品の手順に従ってください。なお、データベース製品は本製品に同梱されていません。

◆アプリケーション作成

注意

C言語アプリケーションを使用できるのは、同期アプリケーション連携実行基盤だけです。

- アプリケーション開発コンピュータおよびアプリケーションサーバ(開発用)

  業務で使用するクライアントアプリケーションおよびサーバアプリケーションを作成します。アプリケーションの開発の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”を参照してください。

  ー クライアントアプリケーション
  同期アプリケーション連携実行基盤では、COBOL開発支援ツール、または実行基盤インタフェース生成ツールを使用して生成されたbeanおよびデータ変換クラス（Javaの場合）、またはC言語クライアントソースファイル（C言語の場合）と、利用者が作成したクライアント処理を合わせて、アプリケーションを作成します。
  非同期アプリケーション連携実行基盤では、Javaのアプリケーションにフロー開始APIや結果取得APIを組み込むことでアプリケーションを作成します。

ー サーバアプリケーション
COBOLまたはC言語のサーバアプリケーションは、COBOL開発支援ツール、または実行基盤インタフェース生成ツールを使用して生成された実行基盤インタフェースと、利用者が作成した業務ロジックを結合してサーバアプリケーションを作成します。

EE
また、非同期アプリケーション連携実行基盤のJavaのアプリケーションは、フロー定義を作成後、Interstage Studioのウィザードで作成します。

ー アプリケーション連携フロー定義
非同期アプリケーション連携実行基盤を使用して複数業務やサービスの連携を行う場合、アプリケーション連携の流れなど業務を実現するためのアプリケーション連携フロー定義をフロー定義ツールで作成します。

■システム環境作成

システム環境の作成は、システム構築シート、環境構築コマンド、および簡易セットアップを使用して行います。

- データベースサーバ

  アプリケーション連携実行基盤を使用するためのデータベースを作成します。作成する環境には、以下の2つがあります。

  ー データベース環境作成 EE

  非同期アプリケーション連携実行基盤を使用する場合に必要なフロー定義DB、メッセージトラッキングDBおよびメッセージ格納DBといったアプリケーション連携実行基盤を動作させるための環境を作成します。
  詳細は、“Interstage Business Application Server セットアップガイド”を参照してください。

  注意

  メッセージ格納DBは、製品版Symfoware Serverだけで作成することができます。

  ー 高信頼性ログの環境作成

  高信頼性ログのデータを格納する環境を作成します。
  詳細は、“Interstage Business Application Server 運用ガイド(高信頼性ログ編)”を参照してください。

  ポイント

  オンデマンド型のオンライン・バッチ連携業務を実現するために、高信頼性ログのデータを活用したエクスポートユーティリティを提供しています。詳細は、“第7章 エクスポートユーティリティ”を参照してください。

- アプリケーションサーバ

  アプリケーション連携実行基盤が使用する以下に示すInterstage Application Serverの環境を作成します。
  詳細は、“Interstage Business Application Server セットアップガイド”を参照してください。

  ー イベントチャネル作成 EE

  ー ワークユニット(IJServer)の作成など

また、アプリケーション連携実行基盤が使用する以下に示すInterstage Business Application Serverの環境についても作成します。
詳細は、“Interstage Business Application Server セットアップガイド”を参照してください。

- データベースリソース定義
- Destination定義の登録など EE

■運用

- データベースサーバ

  アプリケーション連携実行基盤が使用するデータベースを起動し、アプリケーションサーバからアクセスできる状態にします。
  詳細は、“Interstage Business Application Server 運用ガイド(アプリケーション連携実行基盤編)”を参照してください。

・ アプリケーションサーバ

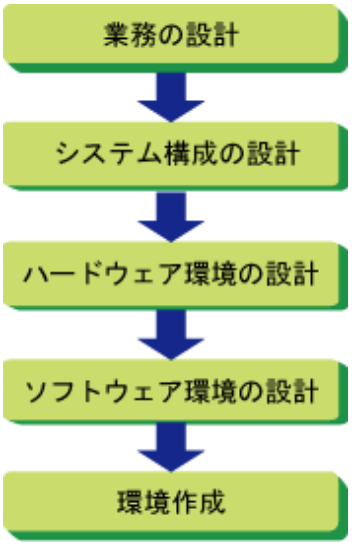
アプリケーション連携実行基盤が使用するInterstage Application Serverの以下の資源を起動します。
詳細は、“Interstage Business Application Server 運用ガイド(アプリケーション連携実行基盤編)”を参照してください。

－ イベントチャネル EE

－ ワークユニット(IJServer)など

9.1 導入作業

アプリケーション連携実行基盤を使用するためには、以下に示す導入作業を行います。それぞれの作業概要について説明します。



9.1.1 業務の設計

システムが実行する業務の内容を明確化します。業務要件、利用者の規模、および期待するスループットなど、実際のシステム構築やアプリケーションの設計に関連する要素を抽出します。

9.1.2 システム構成の設計

業務設計した内容を元に実際に動作するシステム構成を設計します。例えば、1つの業務やサービス内のアプリケーションによる処理であれば、同期アプリケーション連携実行基盤を使用し、複数の業務やサービスの連携を行う場合、非同期アプリケーション連携実行基盤を使用したシステム構成を設計します。

9.1.3 ハードウェア環境の設計

業務の信頼性や性能といったシステム要件を元にサーバの分散やクラスタ化などにより構築する実際の動作環境をマッピングします。
サーバの構成は、以下の3パターンに大別されますので、信頼性や性能といった要件をもとに適切な構成を選択してください。
ハードウェア環境の設計の詳細については、“Interstage Business Application Server セットアップガイド”および“Interstage Business Application Server 運用ガイド(高信頼性ログ編)”を参照してください。

・ 基本構成

単一または複数のアプリケーションサーバとデータベースサーバの構成で、アプリケーション連携実行基盤のシステムを作成します。すべての環境作成パターンの基礎となる構成です。

・ クラスタ構成

アプリケーション連携実行基盤のシステムの可用性を向上させるため、サーバに運用システムと待機システムを用意して、クラスタ運用する場合の構成です。

・ 負荷分散構成

業務処理を行うアプリケーション連携フロー単位にサーバ群を多重化し、業務処理の負荷を分散する場合の構成です。

9.1.4 ソフトウェア環境の設計

アプリケーション連携実行基盤で使用するアプリケーションを運用するワークユニット(IJServer)の決定やイベントチャネルの設計などアプリケーションサーバとして使用するInterstage Application Serverの環境とフロー定義のマッピングを行います。
ソフトウェア環境の設計に詳細については、“Interstage Business Application Server セットアップガイド”および“Interstage Business Application Server 運用ガイド(高信頼性ログ編)”を参照してください。

9.1.5 環境作成

設計したハードウェアおよびソフトウェア環境をデータベースサーバとアプリケーションサーバに作成します。環境作成は、システム構築シート、環境構築コマンド、および簡易セットアップを使用して行います。詳細については、“Interstage Business Application Server セットアップガイド”および“Interstage Business Application Server 運用ガイド(高信頼性ログ編)”を参照してください。

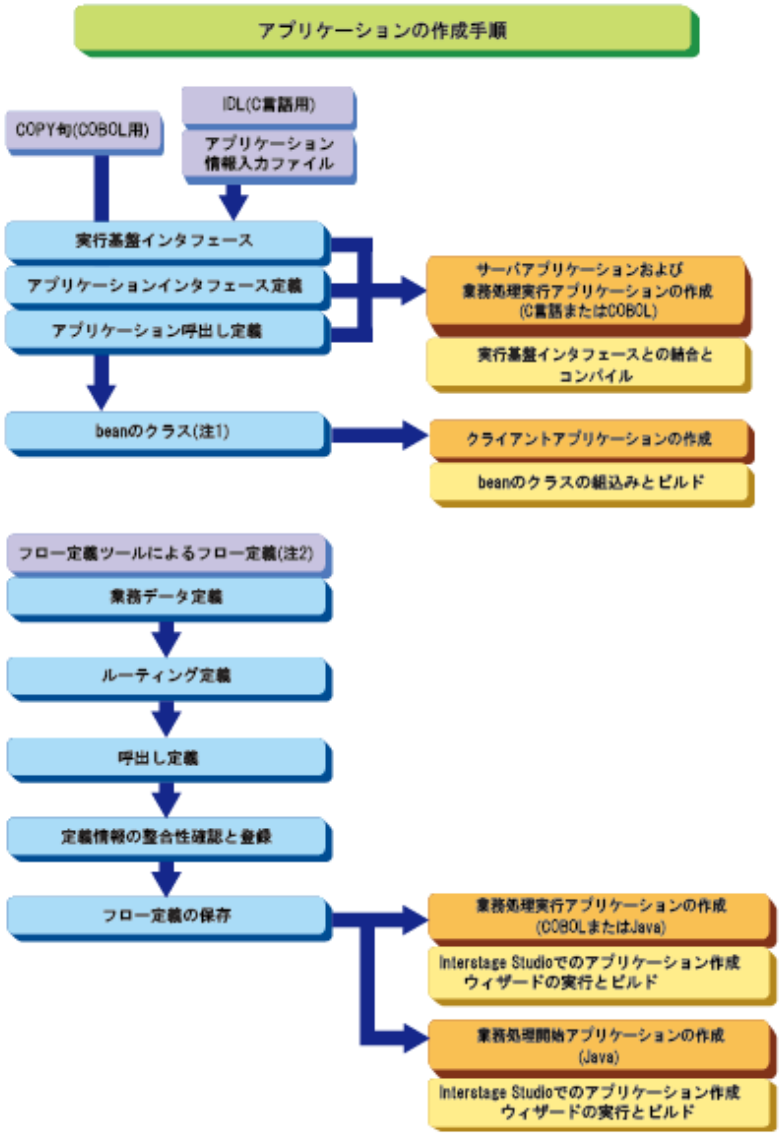
9.2 アプリケーション開発

アプリケーション連携実行基盤を利用したシステムの開発では、以下の2つの作業を行います。本節では、アプリケーションおよびフロー定義の作成について作業の概要を説明します。

・ アプリケーションの作成

・ フロー定義の作成(非同期アプリケーション連携実行基盤を使用する場合) EE

アプリケーションの作成手順の概要を以下の図に示します。



注1) 同期アプリケーション連携実行基盤の場合に使用する。
注2) 非同期アプリケーション連携実行基盤の場合に使用する。

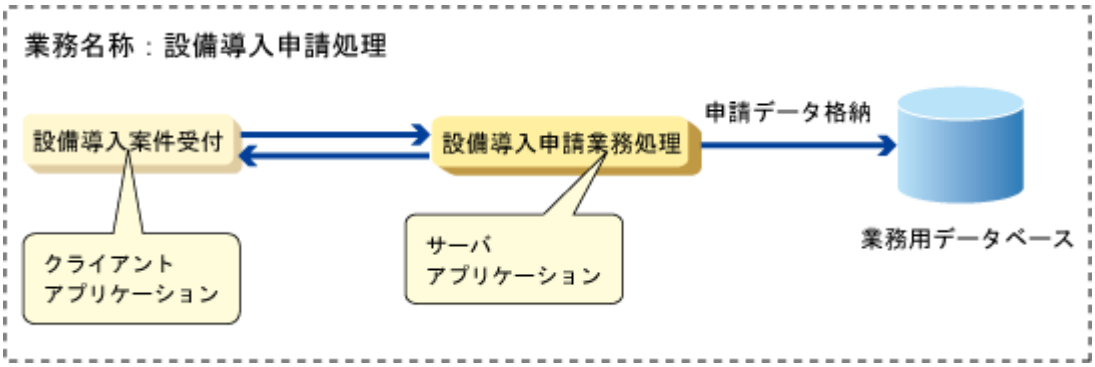
9.2.1 アプリケーションの作成

アプリケーション連携実行基盤では、業務を開始するためのクライアント処理を行うアプリケーションおよびサーバで業務処理を実行するアプリケーションを作成します。
アプリケーションの作成方法の詳細については、“Interstage Business Application Server アプリケーション開発ガイド”を参照してください。

■同期アプリケーション連携実行基盤を使用する場合

- 要求を発行するアプリケーション(クライアントアプリケーション)

- サーバで業務処理を行うアプリケーション(サーバアプリケーション)



これらの2種類のアプリケーションは、以下の点で異なっています。

- 動作の契機が異なる

  クライアントアプリケーションの動作契機は、そのアプリケーションがブラウザのリクエストに対応するServletとして呼び出された時点になります。
  一方、サーバアプリケーションの動作契機は、クライアントアプリケーションから要求が発行された時点となります。

- アプリケーション連携実行基盤のAPIの利用方法が異なる

  クライアントアプリケーションでは、サーバアプリケーションを呼び出すためのAPIを用いる必要があります。このAPIにより呼び出されたサーバアプリケーションが動作します。
  一方、サーバアプリケーションでは、アプリケーション連携実行基盤のAPIを用いる必要はありません。アプリケーションは、クライアントから要求が発行された時点で開始され、終了(復帰)するとクライアントアプリケーションに応答が戻ります。

推奨されるアプリケーションの仕様は、以下のとおりです。

- クライアントアプリケーション

  初期状態の業務データの作成とサーバアプリケーションを呼び出すAPIだけを実行するようにします。例えば、ブラウザからの入力情報をもとに初期状態の業務データを作成し、サーバを呼び出すAPIを実行するWebアプリケーション(Servlet)などが想定されます。

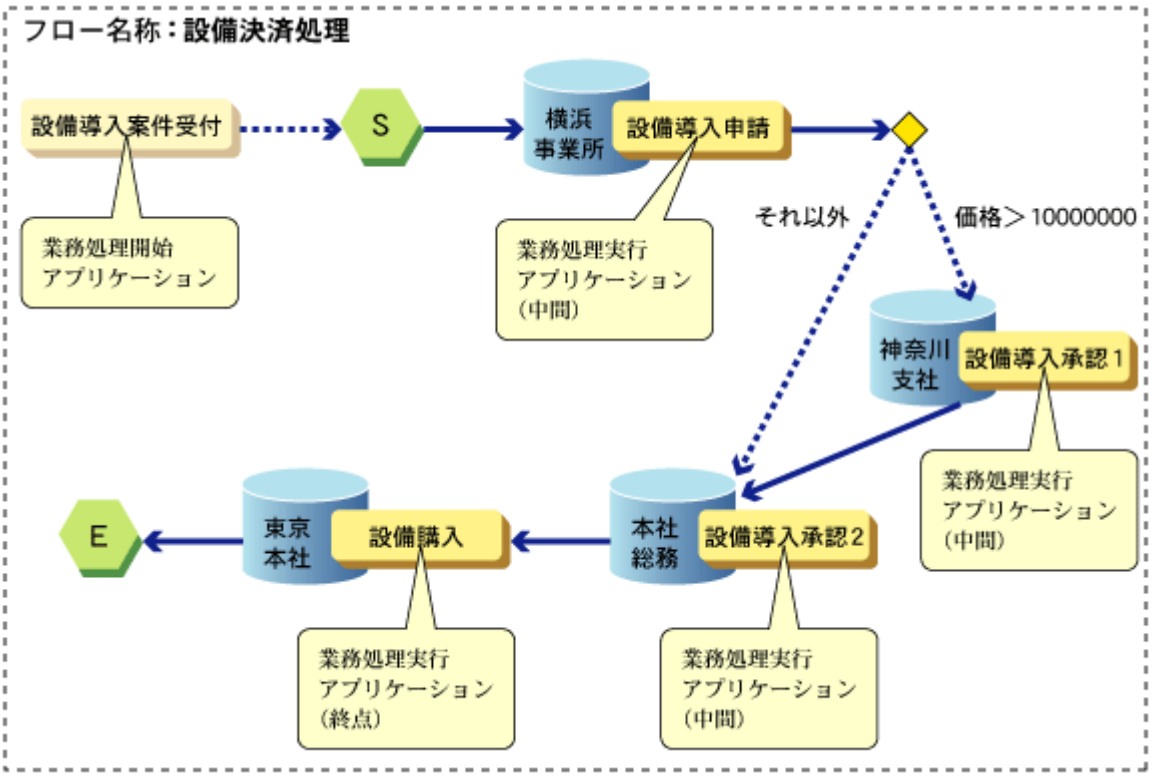
- サーバアプリケーション

  データを受信してデータベース更新するなどの業務ロジックを実行し、その処理結果を出力するCOBOLまたはC言語のアプリケーションになります。

■非同期アプリケーション連携実行基盤を使用する場合 EE

- フローの起点となるアプリケーション(業務処理開始アプリケーション)

- フローの起点以外のアプリケーション(業務処理実行アプリケーション)



これらの2種類のアプリケーションは、以下の点で異なっています。

- 動作の契機が異なる

  業務処理開始アプリケーションには、関連付けられたキューが存在しないため、動作契機はそのアプリケーションがクライアントなどから直接呼び出された時点になります。
  一方、業務処理実行アプリケーションは、フロー定義によってキューに関連付けられます。動作契機は、関連付けられたキューにメッセージが到着した時点になります。

- アプリケーション連携実行基盤のAPIの利用方法が異なる

  業務処理開始アプリケーションでは、フローを開始するためのAPIを用いる必要があります。このAPIにより、アプリケーション連携のためのメッセージが送信され、業務処理実行アプリケーションが順次動作します。
  一方、業務処理実行アプリケーションでは、アプリケーション連携実行基盤のAPIを用いる必要はありません。アプリケーションの動作およびその後のメッセージ送信は、すべてキューへのメッセージの到着に連動する形でアプリケーション連携実行基盤の実行環境が行います。

推奨されるアプリケーションの仕様は、以下のとおりです。

- 業務処理開始アプリケーション

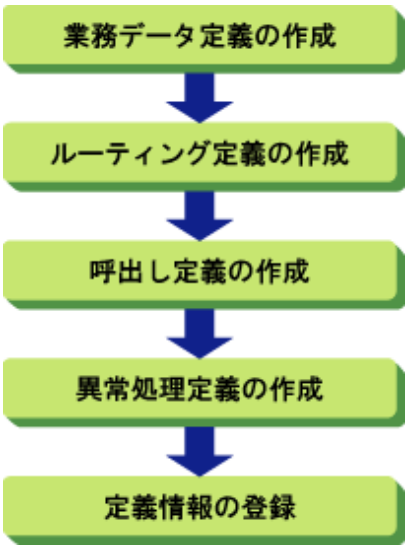
  初期状態の業務データの作成とフロー起動のAPIだけを実行するようにします。例えば、ブラウザからの入力情報をもとに初期状態の業務データを作成し、フロー起動APIを実行するWebアプリケーション(Servlet)などが想定されます。

- 業務処理実行アプリケーション

  データを受信してデータベース更新するなどの業務ロジックを実行し、その処理結果を出力するCOBOLまたはJavaアプリケーションになります。

9.2.2 フロー定義の作成EE

非同期アプリケーション連携実行基盤を使用する場合、フロー定義は、以下の手順に従って作成します。



9.2.2.1 業務データ定義の作成

メッセージとして持ち回る業務データの構造を設計します。

9.2.2.2 ルーティング定義の作成

メッセージのルーティングをDestinationに関連付け、メッセージの起点から終点までのフローをフロー定義ツールのGUI画面上で設計します。

9.2.2.3 呼出し定義の作成

フローの各処理(アクティビティ)に関連付けられる業務処理実行アプリケーションの呼び出し情報を設定します。業務処理実行アプリケーションを呼び出すための関数やメソッド、入出力パラメタおよび補償ルート制御機能を使用する場合は、補償ルート用の関数やメソッドの定義などを行います。

9.2.2.4 異常処理定義の作成

メッセージのルーティング中にエラーが発生した場合の後処理を定義します。

9.2.2.5 定義情報の登録

業務データ定義、ルーティング定義、呼出し定義および異常処理定義の各種定義情報をデータベースサーバへ登録します。フロー定義の登録方法は、以下の3つの方法があります。

- Interstage管理コンソールによるフロー定義の登録

  Interstage管理コンソールによるフロー定義の登録の詳細は、Interstage管理コンソールのヘルプを参照してください。

- apfwaddflowコマンドによるフロー定義の登録

  apfwaddflowコマンドよるフロー定義の登録の詳細は、“Interstage Business Application Server リファレンス”を参照してください。

9.3 日常の運用

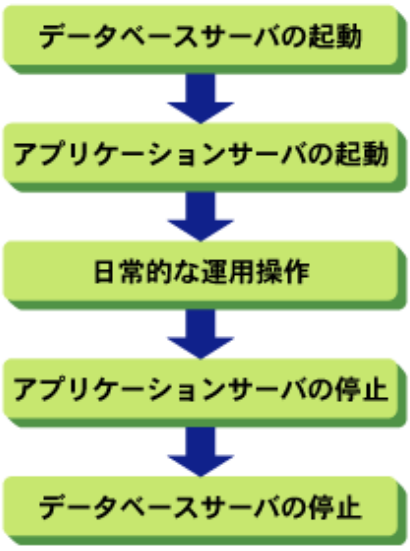
Interstage Business Application Serverを利用したシステムの運用の概要について説明します。

運用の詳細については、“Interstage Business Application Server 運用ガイド(アプリケーション連携実行基盤編)”および“Interstage Business Application Server 運用ガイド(高信頼性ログ編)”を参照してください。

9.3.1 サーバの起動と停止

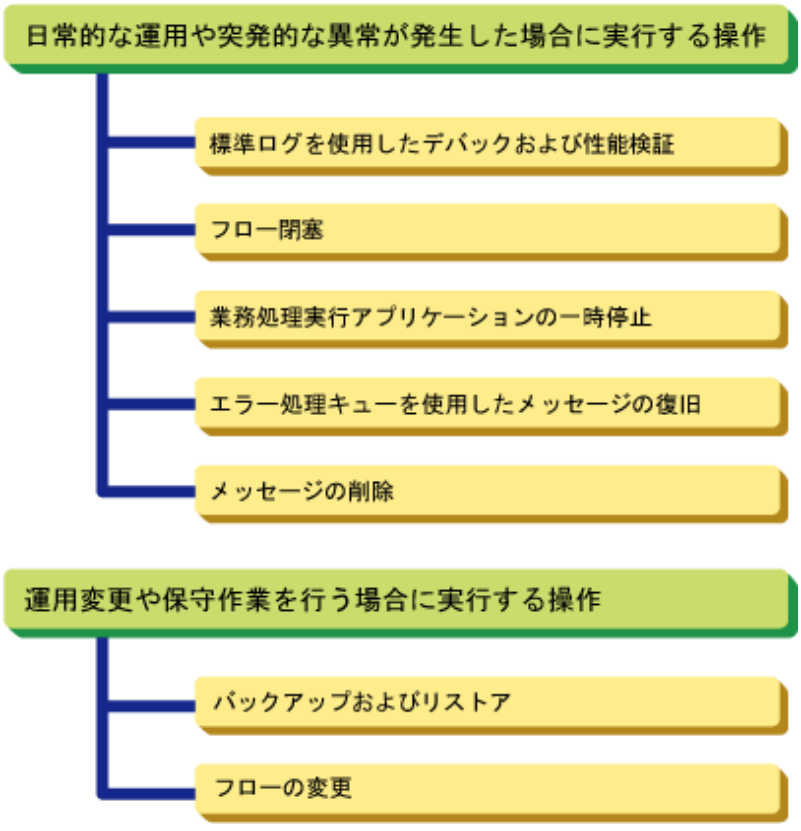
アプリケーション連携実行基盤を利用したシステムにおけるアプリケーションサーバとデータベースサーバの起動および停止順序を以下に示します。
システムが複数のアプリケーションサーバから構成されている場合、それぞれのサーバについて下記の順番となるように運用してください。

［アプリケーションサーバおよびデータベースサーバの起動と停止順序］



9.3.2 運用と保守

アプリケーション連携実行基盤を利用するための運用および保守として、以下に示す操作が存在します。業務処理の利用シーンに合わせて各操作を実行してください。



■標準ログを使用したデバッグおよび性能検証

運用前のデバッグや性能検証および運用開始後にパフォーマンス改善を行う場合、標準ログを用いることでデバッグや性能検証を行います。

■フローの閉塞EE

運用中に特定のフローに閉じた業務異常が発生した場合、特定のフローを閉塞し、フローの開始を抑止することでクライアントに業務処理ができないことを通知します。異常が復旧した時点で閉塞を解除し、業務処理を再開します。

■業務処理実行アプリケーションの一時停止EE

運用中に特定の業務処理実行アプリケーションで異常が発生した場合、任意の業務処理実行アプリケーションの処理を一時的に停止します。メッセージは、処理要求メッセージ格納キューで受け付けておき、異常が復旧した時点で停止解除することで業務を再開します。

■エラーメッセージ退避キューを使用したメッセージの復旧EE

運用中に業務処理実行アプリケーションで異常が発生した場合、異常となったメッセージの修復などの処置を行い、運用を再開します。

■メッセージの削除EE

運用中に規定時間内に業務処理が完了しなかったなどの理由により不要となったメッセージを、業務終了時に削除します。

■バックアップおよびリストア

システムの入れ替えやデータベースの異常などの理由によりアプリケーション連携実行基盤を使用するシステムの修復を行う場合、データベースサーバおよびアプリケーションサーバに作成したアプリケーション連携実行基盤の環境をバックアップおよびリストアします。

■フローの変更EE

業務の追加や削除などの理由により業務処理の流れを変更する場合、フロー定義の変更やアプリケーションの追加や削除などを行います。

付録A　エディション毎の提供機能一覧

Interstage Business Application Serverのエディション毎の提供機能を説明します。

■表記について

以下の表でエディション毎の提供機能一覧を示します。

機能概要	SE	EE

機能概要

提供機能名の概要を示します。

SE, EE,

SE: Interstage Business Application Server Standard Edition
EE: Interstage Business Application Server Enterprise Edition

それぞれのエディションでどの機能が提供されるかを示します。

○:提供します。
－:このエディションでは提供されません。

表A.1 提供機能一覧

機能概要			SE	EE
システム構成	業務モデル	同期型のオンライン業務	○	○
		ディレイド型のオンライン業務	－	○
	システム構成のパターン	シングルサーバ構成	○	○
		Web/AP分離構成	○	○
		複数サーバ構成	○	○
		クラスタ構成	○	○
		負荷分散	○	○
機能構成	導入支援	システム構築シート	○	○
		簡易セットアップ	－	○
		環境構築コマンド	○	○
	開発環境	COBOL開発支援ツール	○	○
		Webサービスインタフェース生成ツール	○	○
		実行基盤インタフェース生成ツール	○	○
		bean生成ツール	○	○
		フロー定義ツール	－	○
		アプリケーション配備機能	○	○
	実行環境	同期アプリケーション連携実行基盤	○	○
		非同期アプリケーション連携実行基盤	－	○
		ログ機能	○	○
		データベースアクセス管理機能	○	○
		アプリケーション安定稼働機能	○	○
		エクスポートユーティリティ	－	○
	運用保守	利用者制限	○	○
		Interstage管理コンソール	－	○

機能概要			SE	EE
		メッセージトラッキング機能	―	○

フレームワークの各エディションで提供される機能の一覧は、“フレームワーク”の“Apcoordinatorユーザーズガイド”の“各エディションで提供される機能”を参照してください。

注意

表中のInterstage管理コンソールは、非同期アプリケーション連携実行基盤向けに提供しているフロー定義の運用および保守ができる機能について記載しています。
その他のInterstage管理コンソールで提供される機能については、Interstage管理コンソールのヘルプを参照してください。

付録B　アプリケーション安定稼動機能（V10.0旧版互換）

アプリケーション安定稼動機能について説明します。

B.1　概要

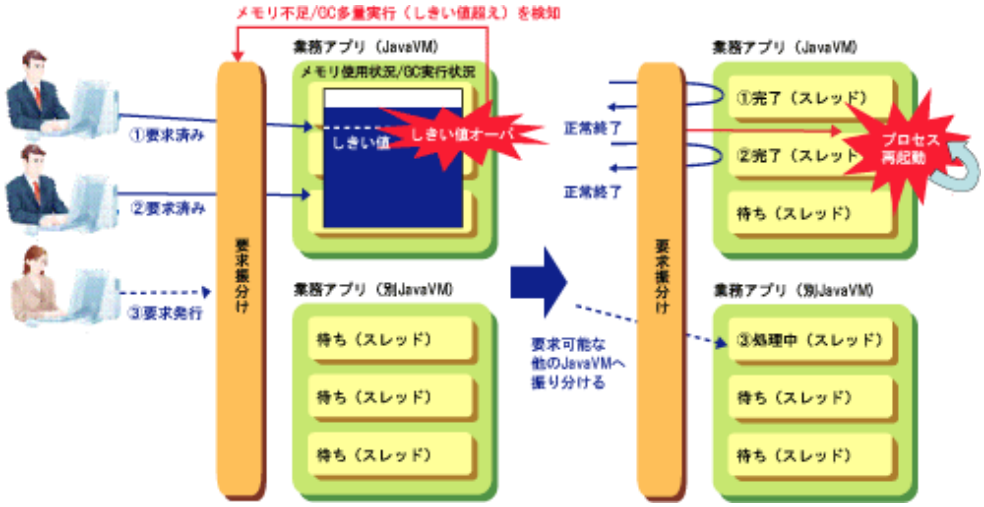
アプリケーション安定稼動機能の概要について説明します。

B.1.1　特徴

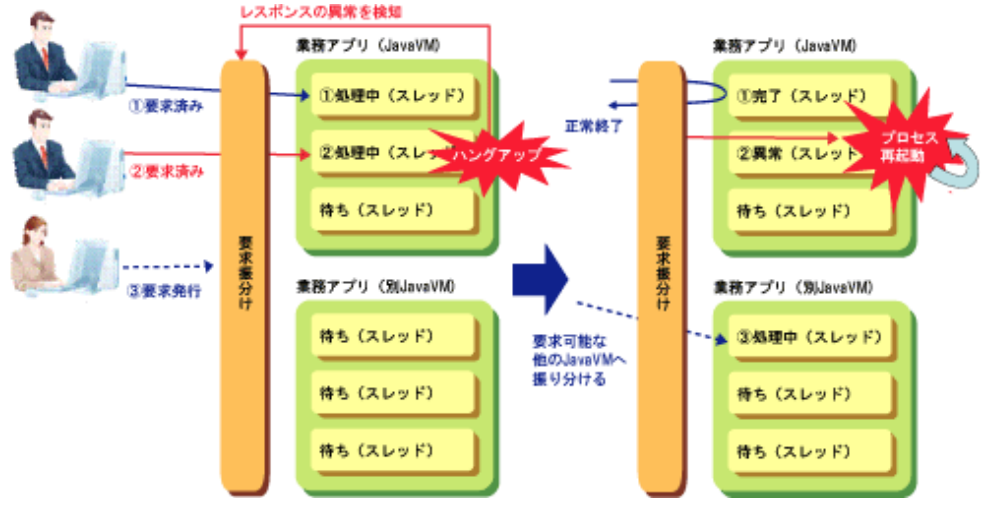
マルチスレッドで動作するJavaアプリケーションの一部が無応答になった場合、およびヒープ不足やガーベジコレクションの多発によるレスポンス遅延が発生した場合に正常実行中のアプリケーションの終了を待ち合わせ、新規リクエストを別の処理可能なJavaVMに振り分けることで安全かつ安定した業務継続を実現します。

本機能による安定性保証および異常の局所化方法を以下に説明します。

(1) 予兆監視によるスループットの安定性保証



(2) レスポンス監視による異常局所化



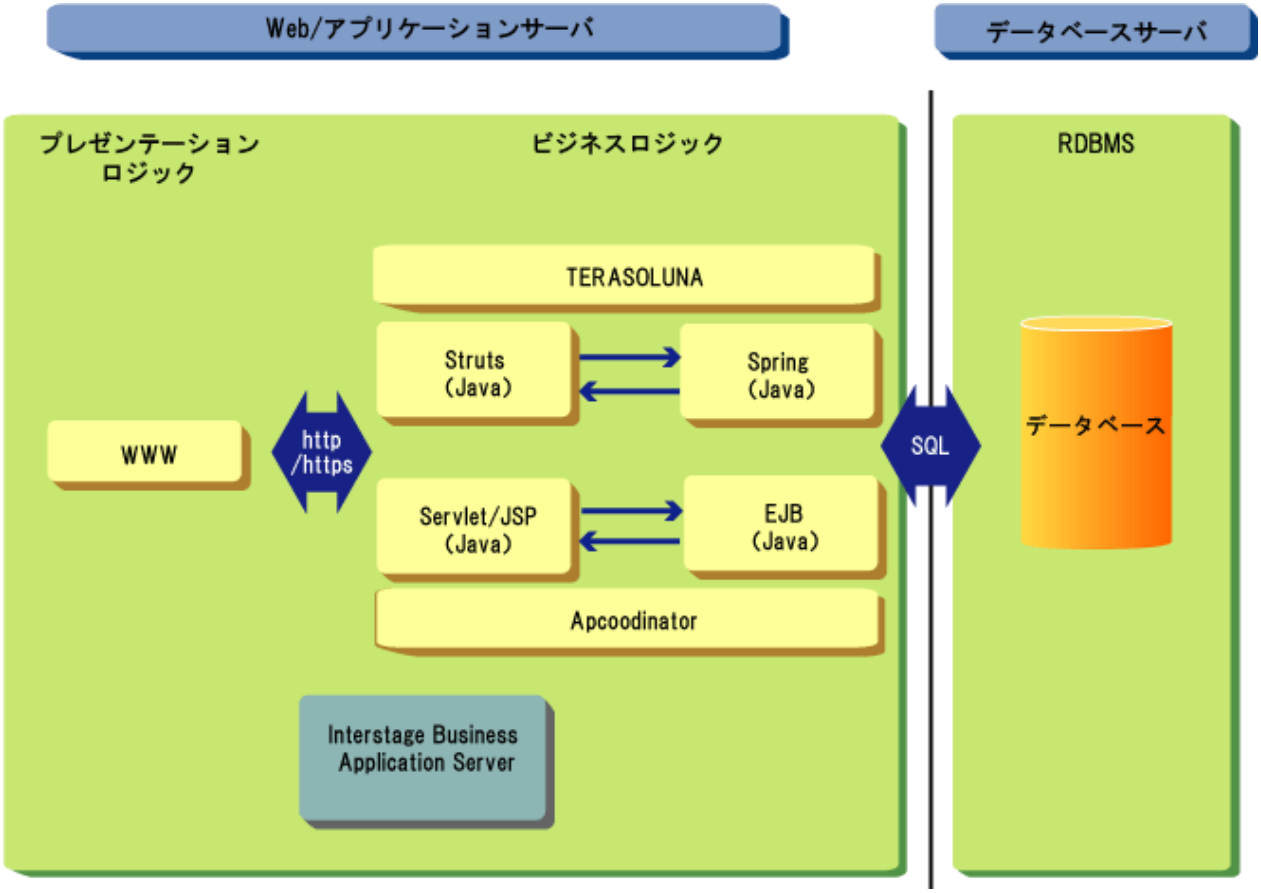
B.1.2　適用パターン

本機能は、Interstage Business Application Serverが提供する“同期型のオンライン業務”、“ディレイド型のオンライン業務”および“オープンJavaフレームワーク”のアプリケーションで利用できます。

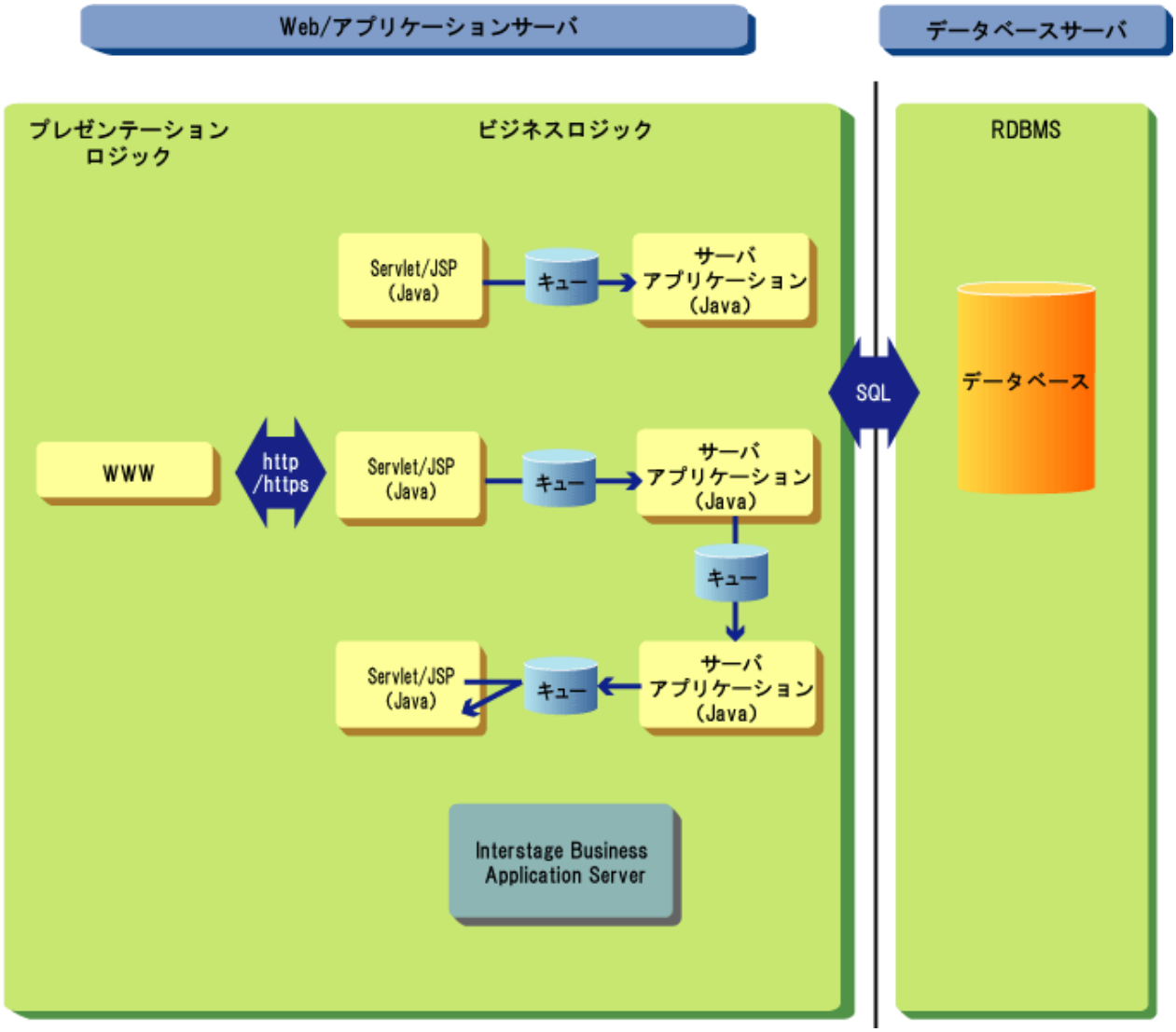
アプリケーションの利用パターンを以下に示します。

[利用パターン]

(1) 同期型のオンライン業務またはオープンJavaフレームワークで利用する場合(Web/AP同居型)

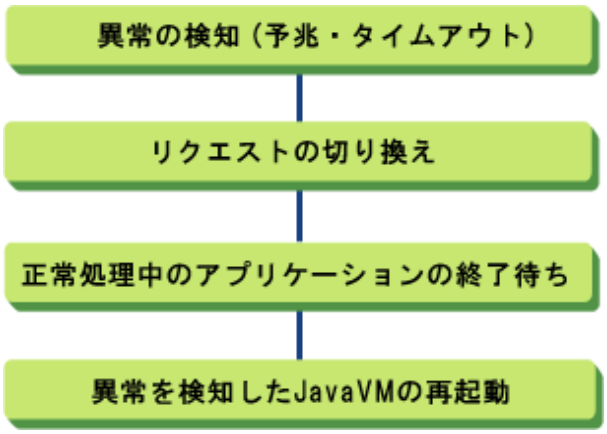


(2) ディレイド型のオンライン業務で利用する場合



B.2 機能

本機能は、異常状態の検知後、アプリケーションを安全に停止するために次の4段階で動作します。



B.2.1 異常の検知(予兆・タイムアウト)

アプリケーションおよびJavaVMの実行状態をモニタリングすることで異常を検知します。本機能が検知する異常項目および条件を以下に示します。異常項目はいずれか片方または両方を検知対象とすることができます。

- JavaVMの予兆監視警告

  Interstage Application Serverの予兆監視機能において予兆監視警告となる条件が本機能の“■定義ファイルの作成”で指定する一定時間内に発生した回数を監視します。Interstage Application Serverの予兆監視警告の対象となるJavaVMの状態を以下に示します。予兆監視警告の詳細は、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”の“予兆監視”を参照してください。

  ー Javaヒープ領域

    Javaヒープ領域不足の危険性を予知

  ー ガーベジコレクション

    Java VMのガーベジコレクション多発による業務処理への影響を予知

- アプリケーションタイムアウト

  Interstage Application Serverのサーバアプリケーションタイマー機能で指定したアプリケーション最大処理時間の超過を監視します。最大処理時間の超過の監視条件は、Interstage Application Serverと同一です。詳細は、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”の“EJBサービスで使用できる時間監視機能”および“Servletコンテナのチューニング”を参照してください。

B.2.2 リクエストの切り換え

異常を検知すると当該JavaVMに対する新規リクエストは、同一のIJServerのプロセス多重で動作している別の正常なJavaVMに自動的に切り換えられます。

B.2.3 正常処理中のアプリケーションの終了待ち

異常を検知したJavaVMには処理中のリクエストが存在します。本機能ではJavaVMを安全に停止するため、アプリケーションタイムアウトとなったリクエストを除いたリクエストが正常に終了するのを待ち合わせます。これにより実行中の正常アプリケーションに影響を与えることなく異常の局所化が可能となります。

B.2.4 異常を検知したJavaVMの再起動

異常を検知したJavaVMを再起動します。再起動後に発生したリクエストは、当該JavaVMに対しても自動的に割り振られ業務を継続することができます。

なお、本機能を適用すると、IJServerの[ワークユニット:環境設定]>[ワークユニット設定]>「アプリケーション最大処理時間超過時の制御」で設定した値は無効化され、必ずプロセスの再起動を行うようになります。

再起動を行った回数が、[ワークユニット:環境設定]>[ワークユニット設定]>[リトライカウント]で設定した値に達した場合は、それ以上は再起動を行いません。

JavaVMを再起動するタイミングには以下の2つの契機が存在します。

- アプリケーションタイムアウトまたは予兆監視警告による異常検知後、正常処理中のアプリケーションがすべて終了した契機
- 予兆監視警告による異常検知後、本機能の定義ファイルで指定した強制停止までの時間が経過した契機

■JavaVMの再起動時に出力するデバッグ情報

JavaVMの再起動時には異常の検知となった原因を利用者が調査しアプリケーションや動作環境の問題を究明するためのデバッグ情報を出力します。デバッグ情報は以下であり、詳細は“B.3.5 採取情報”を参照してください。

- デッドロック情報
- スレッドダンプ情報

B.3 導入・運用

アプリケーション安定稼動機能の導入と運用について説明します。

B.3.1 設計

異常の検知条件となる予兆監視条件およびアプリケーションタイムアウト時間の設計を行います。設計は以下を参考に実施してください。

■予兆監視条件

Interstage Application Serverの予兆監視条件をもとに検知を行います。

予期しない急激なJavaヒープの枯渇や実メモリの不足は、アプリケーションのレスポンスが劣化するだけでなく、OSからJavaVMが強制的に停止されるなど本機能が動作できない可能性があります。このような状態を避けるため、予兆監視条件を設計することを推奨します。

設定値については、Interstage 管理コンソールモニタ機能や、J2EEモニタロギング機能などを使用し、アプリケーション安定稼動を行うための許容条件を割り出して設定してください。

予兆監視条件についての詳細は、Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。

J2EEモニタロギング機能の詳細は、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。

■アプリケーションタイムアウト時間

Interstage Application Serverのアプリケーション最大処理時間で設定された値をもとに状態監視を行います。

設定値については、業務処理として許容可能なアプリケーションの処理時間を基に設計してください。

アプリケーション最大処理時間についての詳細は、“Interstage Application Server Interstage管理コンソールヘルプ”、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。

B.3.2 導入

以下の手順でアプリケーション安定稼動機能の設定を行います。これらすべての設定を行うことにより、アプリケーション安定稼動機能が有効となります。

1. 定義ファイルの作成
2. 起動時実行クラスの登録
3. ワークユニットの設定
4. Webサーバコネクタ(コネクタ)設定

■定義ファイルの作成

アプリケーション安定稼動機能で使用する定義ファイルの作成を行います。

定義ファイルは、ワークユニット毎に異なる監視条件を設定する場合はワークユニット毎に作成します。

また、複数のワークユニットで監視条件が同一の場合は、同じ定義ファイルを利用することができます。

ワークユニットに定義ファイルを指定しない場合には、以下に格納された定義ファイルが読み込まれます。

このファイルを編集するとデフォルトの動作を変更することができます。

[本機能のデフォルト動作となる定義ファイルの格納場所]

Windows32

[Interstageインストールディレクトリ]BAS¥etc¥po¥po.xml

Solaris Linux32/64

/opt/FJSVibs/etc/po/po.xml

定義ファイルの設定項目を以下に示します。

- 予兆監視

  予兆監視の名前を記載した場合は、予兆監視によるアプリケーション安定稼動を行います。

- タイムアウト監視

  タイムアウト監視の名前を記載した場合は、タイムアウト監視を行います。ただし、Interstage Application Serverのワークユニットの設定において、アプリケーション最大処理時間超過の監視を行わない(アプリケーション最大処理時間:0)設定とした場合は、監視を行いません。

定義ファイルの詳細については、“Interstage Business Application Server リファレンス”の“アプリケーション安定稼動定義リファレンス(V10.0旧版互換)”を参照してください。

■起動時実行クラスの登録

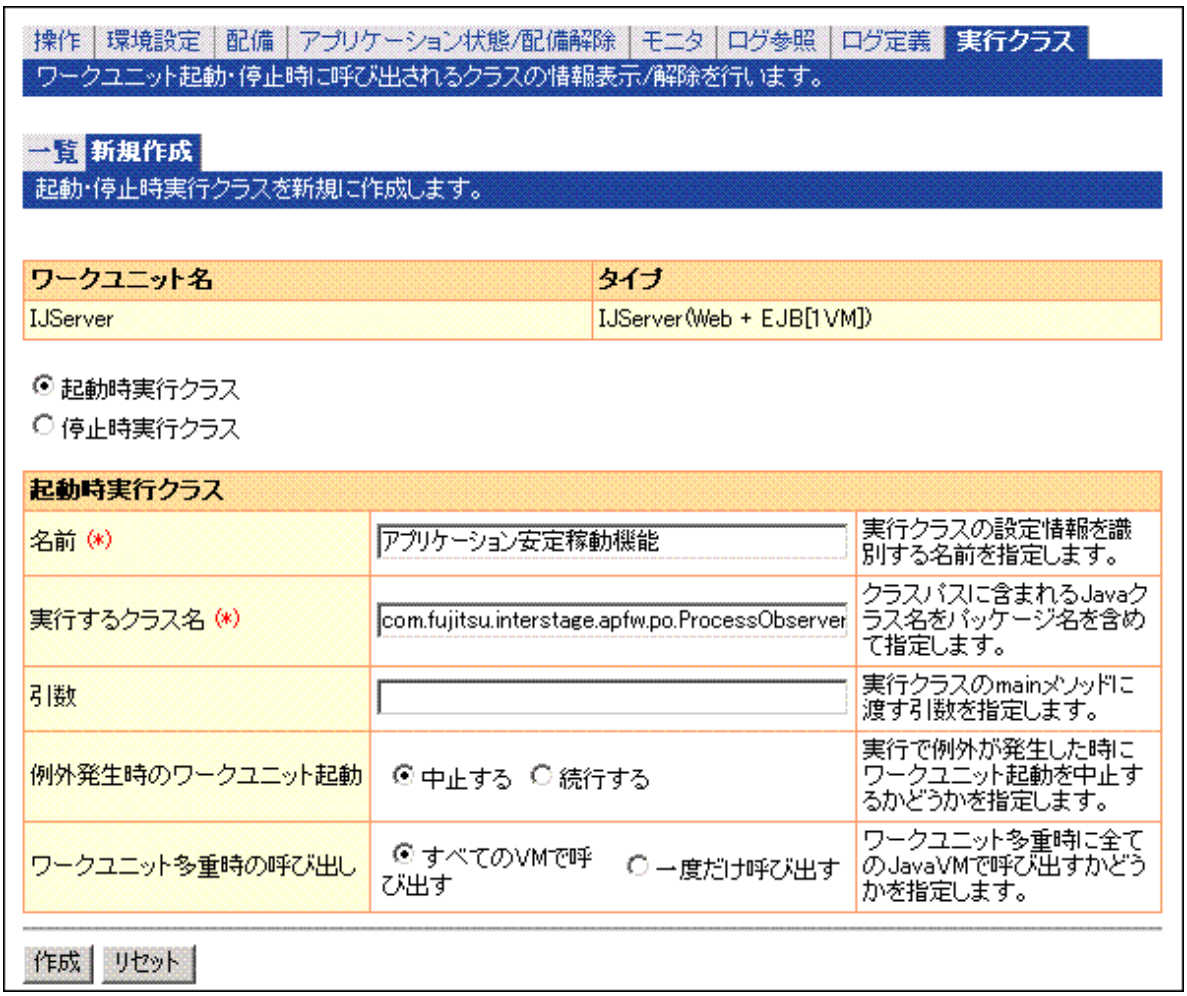
本機能を有効にするために、Interstage Application Serverのワークユニットの設定において、起動時実行クラスの設定を行います。同一ワークユニットに一度だけ設定してください。起動時実行クラスは、Interstage管理コンソールで行うか、isj2eeadminコマンドを用いて設定を行います。

以下に、Interstage管理コンソールでの設定を記述します。

isj2eeadminコマンドの詳細については、“Interstage Application Server リファレンスマニュアル(コマンド編)”を参照してください。

以下にInterstage管理コンソールでの、起動時実行クラス設定例を示します。

Interstage管理コンソールでの設定は、[Interstage管理コンソール] > [システム] > [ワークユニット] > (ワークユニット名) > [実行クラス]で行います。



表B.1 起動時実行クラスの指定内容

項目	設定内容	デフォルト値	意味
名前	アプリケーション安定稼働機能(任意名)	-	実行クラスの設定情報を識別する名前を指定します。
実行するクラス名	com.fujitsu.interstage.apfw.po.ProcessObserverStarter	-	クラスパスに含まれるJavaクラス名をパッケージ名を含めて指定します。
実行順	最初に呼び出す	先に登録した実行クラスの後に呼び出す	登録する実行クラスの実行順番を指定します。
引数	定義ファイル名	-	定義ファイルを指定する場合には、定義ファイルへの絶対パスを記述します。
例外発生時のワークユニット起動	中止する	続行する	実行で例外が発生した時にワークユニット起動を中止するかどうかを指定します。
ワークユニット多重時の呼び出し	すべてのVMで呼び出す	一度だけ呼び出す	ワークユニット多重時に全てのJavaVMで呼び出すかどうかを指定します。
クラスを実行するコンテナ	(任意) アプリケーション安定稼動機能を有効にしたいコンテナを選択してください。	Web-EJB両方	WebアプリケーションとEJBアプリケーションを別JavaVMで運用時にクラスを実行するコンテナを指定します。

引数に定義ファイルを指定しなかった場合には、“本機能のデフォルト動作となる定義ファイル”を読み込みます。詳細につきましては、■定義ファイルの作成を参照してください。

設定に以下の誤りがあった場合は、IJServerのコンテナログにエラーメッセージを出力しワークユニットの起動が失敗します。

- 引数に定義ファイル名を複数指定した場合
- 定義ファイル名に指定されたファイルが存在しなかった場合

注意

- 実行順は、起動時実行クラスを2つ以上指定した場合にのみ表示されます。この場合には、本機能を一番初めに呼び出すように設定してください。
- クラスを実行するコンテナは、ワークユニット作成時に、“WebアプリケーションとEJBアプリケーションを別JavaVMで運用”を選択し、作成している場合にのみ表示されます。

■ワークユニットの設定

ワークユニットの設定を行います。

以下にInterstage管理コンソールでの、設定例を示します。

Interstage管理コンソールでの設定は、[システム] > [ワークユニット] > “ワークユニット名” > [環境設定]タブ > [ワークユニット設定]で行います。

- プロセス多重度

  2以上の値を指定します。

- クラスパスの設定

  ワークユニットにクラスパスを設定します。

  Windows32

[Interstageインストールディレクトリ]¥APC¥lib¥uji.jar
[Interstageインストールディレクトリ]¥APC¥lib¥ujief.jar
[Interstageインストールディレクトリ]¥BAS¥lib¥apfwutils50.jar

Solaris Linux32/64

/opt/FJSVwebc/lib/uji.jar
/opt/FJSVapcef/lib/ujief.jar
/opt/FJSVibs/lib/apfwutils50.jar

- ライブラリパスの設定 Solaris Linux32/64

  ワークユニットに以下のライブラリパスを設定します。

  /opt/FJSVibs/lib

■Webサーバコネクタ(コネクタ)設定

Webサーバコネクタの設定を行います。

以下にInterstage管理コンソールでの、設定例を示します。

Interstage管理コンソールでの設定は、[システム] > [ワークユニット] > “ワークユニット名” > [環境設定]タブ > [Webサーバコネクタ(コネクタ)設定]で行います。

注意

ワークユニットのタイプが、IJServer(EJB Only)の場合には本設定は必要ありません。

- Webサーバ／Webサーバのバーチャルホスト

利用するWebサーバ名にチェックをいれます。

Webサーバコネクタ(コネクタ)設定 [非表示]		
Webサーバ／Webサーバのバーチャルホスト	☑ FJapache	連携するWebサーバと、要求を受け付けるWebサーバのバーチャルホストを指定します。
送受信タイムアウト	[ ] 秒	コネクタがServletコンテナとの間でデータパケットを送受信するときに待機する最長の時間を指定します。
Servletコンテナへの最大接続数	[ ]	Servletコンテナへの接続数の上限値を指定します。
コネクタとServletコンテナ間のSSLの使用	○ 使用する ◉ 使用しない	コネクタとServletコンテナ間の通信にSSLを使用するかどうかを指定します。
コネクタとServletコンテナ間のSSL定義	[なし ▾]	SSLを使用する場合のSSL定義を指定します。
コネクタとServletコンテナ間のKeepAlive	☐ 使用しない	コネクタとServletコンテナ間の通信でKeepAliveを使用しない場合に指定します。

注意

- 本機能では、Webサーバのバーチャルホスト機能は利用できません。

B.3.3 運用操作

■出力メッセージによるアプリケーションおよび運用環境の確認

本機能が動作した場合、アプリケーション処理時間の遅延、使用メモリ量の増加および継続したサーバの高負荷状態など運用設計の想定を超えた異常が発生しています。下表に示すシスログ、もしくはIJServerのコンテナログに出力される情報を元にアプリケーションおよび運用環境の見直しを行ってください。

表B.2 メッセージの出力先

メッセージの出力先	出力契機	出力情報
シスログ(Windowsの場合は、イベントログ)	・ ワークユニット起動時 ・ 予兆条件の検知時 ・ アプリケーション最大処理時間超過の検知時	・ FSP_INTS-BAS_AP12000 番台のメッセージ
IJServerのコンテナログ	・ ワークユニット起動時 ・ 予兆条件の検知時 ・ アプリケーション最大処理時間超過の検知時	・ FSP_INTS-BAS_AP12000 番台のメッセージ ・ デッドロック情報 ・ スレッドダンプ情報

■監視すべき情報

アプリケーション安定稼働機能によってプロセスの再起動が行われた場合、以下のメッセージが出力される場合があります。

表B.3 出力メッセージ

出力メッセージ情報
extp: エラー: EXTP4365: アプリケーションの処理時間が監視時間を超過しました: WU=%s1 PSN=%d1 PID=%d2 SYSTEM=%s2 ［可変情報］ %s1：ワークユニット名 %d1：プロセス通番 %d2：プロセスID %s2：業務システム名
OD: エラー: od10605:%s1: 応答の送信に失敗しました。(from = %s2, intf = %s3, op = %s4) errno = %s5 ［可変情報］ %s1：時刻 %s2：IPアドレス %s3：インタフェース %s4：オペレーション名 %s5：OSから通知されたエラー番号

この場合には、アプリケーションの設計を見直すか、アプリケーション最大処理時間、または、アプリケーション安定稼働機能の強制停止までの時間を見直してください。

B.3.4 チューニング

アプリケーション安定稼動機能で、予兆監視（ヒープ/GC）を使用する場合は、十分考慮してチューニングを行ってください。

チューニングの詳細については、“Interstage Application Server J2EEユーザーズガイド(旧版互換)”を参照してください。

予兆監視での検知が発生しやすいケースの例を、以下に示します。

・ -Xms値と -Xmx値が異なる場合（GC多発)

- -Xmx値が不十分な場合 (GC多発、Javaヒープ不足)
- ロードされるクラスが多い場合 (GC多発、Permanent世代領域の不足)
- セッションが保持しているオブジェクトが多い場合 (GC多発、Javaヒープ不足)
- アプリケーション内でSystem.gc()、Runtime.gc()を必要以上に呼び出している場合 (GC多発)

B.3.5 採取情報

異常検知後、JavaVMを再起動する際に以下の詳細情報をIJServerのコンテナログに出力します。

- デッドロック情報

  異常発生時にデッドロックの発生有無をチェックし、デッドロックが発生している場合には、デッドロック情報をIJServerのコンテナログに出力します。

  出力形式を以下に示します。

  ```
  "スレッド名" tid=スレッドのID in BLOCKED on lock=ロックのオーナー owned by ロックしたスレッド名 tid=ロックしたスレッドのID
  以下、スレッドトレース情報
  ```

  出力例を以下に示します。

  ```
  "MyRunnable0" tid=7 in BLOCKED on lock=java.lang.Object@f72617 owned by MyRunnable1 tid=8
  at app1.Method01$MyRunnable.run(App001.java:113)
  at java.lang.Thread.run(Thread.java:595)

  "MyRunnable1" tid=8 in BLOCKED on lock=java.lang.Object@1e5e2c3 owned by MyRunnable0 tid=7
  at app1.mMethod01$MyRunnable.run(App001.java:113)
  at java.lang.Thread.run(Thread.java:595)
  ```

- スレッドダンプ情報

  異常発生後、プロセスを停止する際にスレッドダンプ情報をIJServerのコンテナログに出力します。

  スレッドダンプの出力形式は、JDK/JREが出力する形式と同じです。