Interstage

FUJITSU

Interstage Application Server

チューニングガイド

Linux(64)

J2UL-1333-02Z0(00)
2011年4月

まえがき

本書の目的

本書は、運用形態を変更したり、システム規模を変更する場合などに必要な環境設定のチューニングについて説明しています。
本書は、Interstage Application Serverの運用を行う方を対象に記述されています。

前提知識

本書を読む場合、以下の知識が必要です。

- C言語に関する基本的な知識
- C++言語に関する基本的な知識
- COBOLに関する基本的な知識
- Java言語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
- オブジェクト指向技術に関する基本的な知識
- 分散オブジェクト技術(CORBA)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
- 使用するOSに関する基本的な知識

本書の構成

本書は以下の構成になっています。

第1章 必要資源
Interstageの運用時に必要な資源について説明します。

第2章 Interstageのチューニング
Interstageのモデルケースから、より詳細なシステム構築を行う場合に必要となるチューニングについて説明します。

第3章 システムのチューニング
システムのチューニングについて説明しています。

第4章 ワークユニットのチューニング
ワークユニットのチューニングについて説明しています。

第5章 J2EEのチューニング
J2EEアプリケーションの動作に必要なチューニングについて説明します。

第6章 業務構成管理機能のチューニング
業務構成管理機能が管理するリポジトリのチューニングについて説明します。

第7章 JDK/JREのチューニング
JDK/JREのチューニングに関して、基本的な知識、Java VM、異常発生時の原因振り分け方法およびチューニング方法について説明します。

付録A CORBAサービスの動作環境ファイル
CORBAサービスの環境定義について説明しています。

付録B コンポーネントトランザクションサービスの環境定義
コンポーネントトランザクションサービスの環境定義について説明しています。

付録C データベース連携サービスの環境定義
データベース連携サービスの環境定義について説明しています。

付録D イベントサービスの環境定義
イベントサービスの環境定義について説明しています。

付録E Interstage HTTP Serverの環境定義
Interstage HTTP Serverの環境定義について説明します。

付録F Interstage シングル・サインオンの環境定義
Interstage シングル・サインオンを運用するための、環境定義のチューニングについて説明します。

付録G マルチサーバ管理の環境定義
マルチサーバ管理の環境定義について説明します。

付録H Portable-ORBの環境設定
Portable-ORBの環境設定について説明します。

製品の表記について

本書での以下の表記については、それぞれの基本ソフトウェアに対応した製品を示しています。

表記	説明
RHEL-AS4(x86)	Red Hat Enterprise Linux AS (v.4 for x86)を前提基本ソフトウェアとしたInterstage Application Server/Interstage Web Server
RHEL-AS4(EM64T)	Red Hat Enterprise Linux AS (v.4 for EM64T)を前提基本ソフトウェアとしたInterstage Application Server/Interstage Web Server
RHEL-AS4(IPF)	Red Hat Enterprise Linux AS (v.4 for Itanium)を前提基本ソフトウェアとしたInterstage Application Server
RHEL5(x86)	Red Hat Enterprise Linux 5 (for x86)を前提基本ソフトウェアとしたInterstage Application Server/Interstage Web Server
RHEL5(Intel64)	Red Hat Enterprise Linux 5 (for Intel64)を前提基本ソフトウェアとしたInterstage Application Server/Interstage Web Server
RHEL5(IPF)	Red Hat Enterprise Linux 5 (for Intel Itanium)を前提基本ソフトウェアとしたInterstage Application Server
RHEL6(x86)	Red Hat Enterprise Linux 6 (for x86)を前提基本ソフトウェアとしたInterstage Application Server/Interstage Web Server
RHEL6(Intel64)	Red Hat Enterprise Linux 6 (for Intel64)を前提基本ソフトウェアとしたInterstage Application Server/Interstage Web Server

輸出許可

本ドキュメントを非居住者に提供する場合には、経済産業大臣の許可が必要となる場合がありますので、ご注意ください。

著作権

2010年10月　初版
2011年 4月　第2版

目　次

第1章 必要資源

1.1 運用時に必要なディスク容量

運用時に必要なディスク容量は次のとおりです。

1.1.1 サーバ機能を使用する場合

以下の各機能を使用する場合のディスク容量を示します。

- Interstage動作環境
- Interstage管理コンソール
- 業務構成管理
- Interstage HTTP Server
- Interstage JMXサービス
- Interstage シングル・サインオン
- Interstage ディレクトリサービス
- J2EE
- IJServerワークユニット
- Interstage JMS
- Servletサービス(OperationManagement)
- ワークユニット
- CORBAサービス
- イベントサービス
- Portable-ORB
- コンポーネントトランザクションサービス
- データベース連携サービス
- 性能監視ツール
- SOAPサービス
- UDDIレジストリサービス
- MessageQueueDirector
- フレームワーク

機能: Interstage動作環境 Windows

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
コンポーネントトランザクションサービスのインストールディレクトリ¥var¥td001 (Interstage動作環境定義ファイルの“TD path for system”で指定)	2 以上	Interstage動作環境作成時

機能: Interstage管理コンソール

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage管理コンソールのインストールディレクトリ¥isAdmin¥var¥download Solaris Linux /var/opt/FJSVisgui/tmp/download	(注)	ログ情報

注)
Interstage管理コンソールの以下の画面において、ログファイルをダウンロードする場合、同時にダウンロードするログファイルのサイズ分のディスク容量が一時的に必要となります。

機能	画面(スタンドアロン)
Interstage HTTP Server	[システム] > [サービス] > [Webサーバ] > [Webサーバ名] > [ログ参照]タブ
	[システム] > [サービス] > [Webサーバ] > [Webサーバ名] > [バーチャルホスト] > [バーチャルホスト名] > [ログ参照]タブ
Webサーバコネクタ	[システム] > [サービス] > [Webサーバ] > [Webサーバ名] > [Webサーバコネクタ] > [ログ参照]タブ
IJServerワークユニット	[システム] > [ワークユニット] > [ワークユニット名] > [ログ参照]タブ

ログファイルのサイズについては、各機能のログ情報のディスク容量を参照し、運用の内容により必要とするサイズを検討してください。
なお、ログファイルのサイズが大きいため、ディスク容量の不足によりログファイルのダウンロードに失敗する場合は、FTPなどを使用してダウンロードしてください。

機能: 業務構成管理

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
業務構成管理のリポジトリ Windows Interstage JMXサービスのインストールディレクトリ¥var¥repository Solaris Linux /var/opt/FJSVisas/repository	運用の内容により、必要とするサイズを検討してください。(注)	デフォルトから変更した場合は、変更先

注)
業務構成管理のリポジトリの格納先のサイズは、“Interstage Application Server 運用ガイド(基本編)”の“業務構成管理機能の操作”を参照してください。

機能: Interstage HTTP Server

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
アクセスログ、エラーログ、トレースログ格納ディレクトリ	運用の内容により、必要とするサイズを検討してください。	アクセスログ、エラーログ、トレースログ
Windows Interstage HTTP Serverのインストールディレクトリ¥logs Solaris Linux /var/opt/FJSVihs/logs	2	オペレーションログ

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage HTTP Serverのインストールディレクトリ¥logs Solaris Linux /var/opt/FJSVihs/logs	10	保守ログ
コンテンツ格納するディレクトリ	運用の内容により、必要とするサイズを検討してください。	コンテンツ(HTML文書など)

機能: Interstage JMXサービス

Windows

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Interstage JMXサービスのインストールディレクトリ	14以上 (注)	

Linux

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
/var/opt	32 以上 (注)	
/etc/opt	0.1 以上	

注)
Interstage JMXサービスのカスタマイズでログインログのファイルサイズの上限値を変更している場合、以下のディスク所要量が必要となります。

ー ログインログ
ログインログのファイルサイズの上限値 × 2 (Mバイト)

上限値を変更していない場合、ログインログのファイルサイズの上限値は1に設定されています。

機能: Interstage シングル・サインオン

Interstage シングル・サインオンの業務サーバ機能

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssoatzag¥log Solaris Linux /var/opt/FJSVssoaz/log	運用の内容により、必要とするサイズを検討してください。(注1)	アクセスログなどのログ情報 (アクセスログファイルの出力先ディレクトリ)
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssocm¥etc Solaris Linux /var/opt/FJSVssocm/etc	2	

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage シングル・サインオンのインストールディレクトリ Solaris Linux /etc/opt	運用の内容により、必要とするサイズを検討してください。	アクセス制御情報

Interstage シングル・サインオンの認証サーバ機能

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssoatcag¥log Solaris Linux /var/opt/FJSVssoac/log	運用の内容により、必要とするサイズを検討してください。(注1)	アクセスログなどのログ情報 (アクセスログファイルの出力先ディレクトリ)
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssofsv¥log Solaris Linux /var/opt/FJSVssofs/log	運用の内容により、必要とするサイズを検討してください。(注1)(注2)	
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssocm¥etc Solaris Linux /var/opt/FJSVssocm/etc	2 (注3)(注4)	

Interstage シングル・サインオンのリポジトリサーバ機能

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssoatcsv¥log Solaris Linux /var/opt/FJSVssosv/log	運用の内容により、必要とするサイズを検討してください。(注1)(注5)	アクセスログなどのログ情報 (アクセスログファイル、およびセション管理ログファイルの出力先ディレクトリ)
Windows Interstage シングル・サインオンのインストールディレクトリ¥ssocm¥etc Solaris Linux /var/opt/FJSVssocm/etc	2	

注1)
デフォルト設定のままでは使用ディスクサイズの上限なしにログが採取されます。ディスク不足発生を防止するために、定期的に不要になったログファイルを削除するか、ログの採取方法を変更してください。

注2)
認証サーバ間連携を行わない場合は、0Mバイトです。

注3)
統合Windows認証を行う場合には、2Mバイトを加算してください。

注4)
認証サーバ間連携を行う場合には、2Mバイトを加算してください。

注5)
セション管理を行うリポジトリサーバをクラスタシステム上で運用する場合には、52Mバイトを加算してください。

機能: Interstage ディレクトリサービス

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage ディレクトリサービスのインストールディレクトリ¥var Linux /var/opt	20 × リポジトリ作成数 ＋ 20	ログ情報
Windows Interstage ディレクトリサービスのインストールディレクトリ¥etc Linux /etc/opt	0.5 × リポジトリ作成数	環境定義
Interstage ディレクトリサービスのアクセスログ作成ディレクトリ	Interstage管理コンソールのアクセスログの設定値に依存 サイズ × 世代管理数	アクセスログ
Windows Interstage ディレクトリサービス SDKのインストールディレクトリ¥var Solaris Linux /var/opt/FJSVirepc	Windows プロセス数 × 8 Solaris Linux 同時接続数 × 8 (注)	Interstage ディレクトリサービス SDKのログ情報

注)
同時接続数には、以下の必要数の総和を指定して見積もってください。

- スレッド多重のアプリケーションを使用してリポジトリにアクセスする場合の、アプリケーションのプロセス数
- プロセス多重のアプリケーションを使用してリポジトリにアクセスする場合の、アプリケーションのプロセス多重度
- Interstage シングル・サインオン機能を使用する場合の、Interstage シングル・サインオンのリポジトリサーバ機能を組み込んだWebサーバのクライアント同時接続数

Interstage ディレクトリサービスは、以下の製品で利用可能です。

・ Interstage Application Server Enterprise Edition

機能: J2EE

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows J2EE共通ディレクトリ	運用の内容により、必要とするサイズを検討してください。	J2EEアプリケーションの資産一式

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Solaris Linux /var/opt/FJSVj2ee/deployment		

機能: IJServerワークユニット

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows J2EE共通ディレクトリ¥ijserver¥IJServer名¥logディレクトリ Solaris Linux J2EE共通ディレクトリ/ijserver/IJServer名/logディレクトリ	24 以上 (注1)	
Windows Interstageのインストールディレクトリ¥F3FMjs5¥logs¥jk2 Solaris Linux /var/opt/FJSVjs5/logs/jk2	2 以上 (注2)	
Windows J2EE共通ディレクトリ¥ijserver¥Session Registry Server(IJServer)名¥apps¥srs.ear¥srs.war¥serializedata¥sessionrecovery Solaris Linux J2EE共通ディレクトリ/ijserver/Session Registry Server(IJServer)名/apps /srs.ear/srs.war/serializedata/sessionrecovery	(注3)	Servletサービスのセッションリカバリ機能使用時 セッションの永続化有効時、Session Registry Serverを運用する環境で必要です。

注1)
IJServerワークユニット1つにつき以下を加算してください。

プロセス多重度 ×
4(コンテナログとコンテナ情報ログのデフォルトディスク使用量) ×
6(世代分のバックアップ) 以上

アプリケーションのタイムアウトが多発する場合、アプリケーションで短時間に大量のメッセージを出力する場合、およびデバッグ情報出力を行う場合は、“J2EE共通ディレクトリ/ijserver/IJServer名/log”配下のコンテナ情報ログのディスク使用量が大きくなります。このような操作が想定される場合は、十分なディスク容量をご用意ください。

注2)
Webサーバ1つにつきデフォルトで2Mバイトです。アプリケーションで短時間に大量のメッセージを出力する場合、デバッグ情報出力を行う場合は、ディスク使用量が大きくなります。このような操作が想定される場合は、十分なディスク容量をご用意ください。

注3)
セッションリカバリ機能を使用して、セッションの永続化を有効にした場合は、Session Registry Server環境定義ファイルで指定したセッションの永続化ファイルの保存先にセッションの永続化ファイルが生成されます。配備するWebアプリケーション1つについて、次のディスク容量が必要です。(単位:Mバイト)

Windows
(0.005 ＋ (0.005 ＋ セッションの保持するデータ容量) × セッション数) × 2
Solaris
(0.001 ＋ (0.002 ＋ セッションの保持するデータ容量) × セッション数) × 2
Linux
(0.008 ＋ (0.008 ＋ セッションの保持するデータ容量) × セッション数) × 2

“セッションの保持するデータ容量”は、Webアプリケーションでセッションの属性(Attribute)にセットするオブジェクトおよびキーのサイズの合計値です。

上記の値は、利用しているファイルシステムによっては値が増減する場合があります。
なお、Session Registry Serverは、Interstage Application Server Enterprise Editionで運用可能です。

機能: Interstage JMS

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage JMSのインストールディレクトリ¥etc Solaris Linux /etc/opt	0.01 ＋ (durable Subscriber数 × 0.002)	定義情報
Windows Interstage JMSのインストールディレクトリ¥var Solaris Linux /var/opt	10 以上	コンソールファイル

機能: Servletサービス(OperationManagement)

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Servletサービス(OperationManagement)インストールディレクトリ¥log Solaris Linux /var/opt/FJSVjs2su/log	12	ログ情報

機能: ワークユニット

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Solaris Linux Interstage動作環境定義の定義項目“TD path for system”で指定	1つのワークユニット定義サイズ × ワークユニット定義数 (注)	ワークユニット定義登録時
	1つのワークユニット定義サイズ × ワークユニット起動数 (注)	ワークユニット運用時

注)
1つのワークユニット定義サイズ ＝

1000 ＋
(500 × “[Application Program]セクション定義数”) ＋
(500 × “[Resource Manager]セクション定義数”) ＋
(500 × “[Nonresident Application Process]セクション定義数”) ＋
(500 × “[Multiresident Application Process]セクション定義数”) ＋
ユーザ任意指定文字列データ長

機能: CORBAサービス

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows CORBAサービスのインストールディレクト	0.1 以上	インプリメンテーション情報、ネーミングサービス、イン

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
リ¥etcフォルダ Solaris Linux /etc/opt		タフェースリポジトリのデータサイズに依存します。
	4.1 以上 (注1)	ネーミングサービス情報
Windows CORBAサービスのインストールディレクトリ¥varフォルダ Solaris Linux /var/opt	8 以上	
	Windows 42.0 Solaris Linux 24.0 (デフォルト時の最大サイズ) (注2)	ログ情報
	2.0 以上 (注3)	内部ログ採取時(プレインストール型Javaライブラリ以外の場合)
	4.0 以下 (注1)	ネーミングサービスのユーザ例外ログ情報
	4.0 以下 (注1)	ネーミングサービスの実行トレース情報(サービス動作時のみ)
	32.3 以下	インタフェースリポジトリサービスのログ情報(サービス動作時のみ)
Windows CORBAサービスのインストールディレクトリ¥var¥traceフォルダ Solaris /var/opt/FSUNod/trace Linux /var/opt/FJSVod/trace	10.4 (注4)	トレース情報
Windows コンポーネントトランザクションサービスインストールディレクトリ¥var¥IRDB Solaris Linux コンポーネントトランザクションサービスインストールディレクトリ/var/IRDB (Interstage統合コマンド使用時のデフォルト)	10.3 以上 (注5)	インタフェースリポジトリサービス情報
Solaris Linux /tmp	1.0 以上 IDL定義の量に依存 C/C++コンパイラ動作時には、別途作業用のディスク容量が必要	IDLコンパイラ動作時
Java VMのシステムプロパティのuser.dirで指定	(注6)	内部ログ採取時(プレインストール型Javaライブラリの場合)

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
環境変数OD_HTTPGW_HOMEまたは OD_HOMEで指定されたvarディレクトリ	2.0 以上 (注7)	HTTP-IIOPゲートウェイの 内部ログ採取時

注1)
CORBAサービスのサーバマシンにネーミングサービスを構築する場合に、必要となるディスク容量について以下に示します。

ー ディスク容量

用途		容量
ネーミングサービス情報	オブジェクトリポジトリ	(固定)16Kバイト
	制御ファイル	(固定)2056Kバイト
	データファイル	(可変)2048(Kバイト) × コンテキスト数 ＋ (オブジェクトリファレンス長 × オブジェクト数 × 2)
実行トレース情報		(最大)4096Kバイト
ユーザ例外ログ情報		(最大)4096Kバイト

注2)
CORBAサービスのログ採取機能を使用している場合、最大で以下のディスク容量を使用します。(各パラメタはconfigファイルで定義)

access_log_size × 2 ＋ error_log_size × 2 ＋ process_log_size × 2 ＋ info_log_size × 2

Windowsでは、上記に加えてさらに以下のディスク容量を使用します。

error_log_size × 2 ＋ process_log_size × 2 ＋ info_log_size × 2

ログ採取機能については“トラブルシューティング集”の“CORBAサービスのログ情報の採取”を、上記パラメタについては“A.1 config”(CORBAサービス)を参照してください。

注3)
以下のディスク所要量(単位:バイト)が必要です。

Windows

(max_processes(*) ＋ 2) × log_file_size(*) × 2
*: CORBAサービスのインストールフォルダ¥etc¥configファイルのパラメタ

Solaris Linux

log_file_size(*)×2
*: configファイルで定義

なお、ログファイルは、不要になった時点で、削除してください。

Windows

採取されるログファイルはlog、log.old以外にサーバアプリケーションごとに“appNNNN.log”、“appNNNN.old”(NNNNは英数字)の名前で採取されます。
自ホストでネーミングサービス、インタフェースリポジトリを動作させる場合には、それぞれ、4Mバイト、32Mバイトの領域が必要です。

注4)
CORBAサービスのトレース情報の採取機能を使用している場合、最大で以下のディスク容量を使用します。(各パラメタはconfigファイルで定義)

trace_size_per_process× max_processes × 2 ＋ trace_size_of_daemon × 2

トレース情報の採取機能については“トラブルシューティング集”の“CORBAサービスのトレース情報の採取”を、上記パラメタについては“A.1 config”(CORBAサービス)を参照してください。

注5)
インタフェースリポジトリを使用する場合のディスク容量について以下に示します。インタフェースリポジトリのデータベースのサイ

ズは、以下の計算式に従って見積もり、ディスクを確保してください。
なお、インタフェースリポジトリのデータベースは、初期値(10240Kバイト)から自動拡張します。

― ディスク容量

用途		容量
インタフェースリポジトリサービス情報	管理域	(固定)220Kバイト
	利用者定義領域	(初期値:可変)10240Kバイト
実行トレース情報		(最大)33000Kバイト

― 見積り式

利用者定義領域(オブジェクトに要するディスク容量)の見積り式を以下に示します。

項番	IDL定義	計算式(単位:バイト)
1	モジュール宣言	1708 ＋ ((a－1)÷32＋1)×176
2	インタフェース宣言	1712 ＋ ((a－1)÷32＋1)×176 ＋ ((b－1)÷32＋1)×176 ＋ 512×b
3	オペレーション宣言	2304 ＋ ((e－1)/32＋1)×176 ＋ ((f－1)÷32＋1)×176 ＋ ((g－1)÷32＋1)×176
4	属性宣言	2224
5	定数宣言	2160 ＋ c
6	例外宣言	1712 ＋ ((d－1)÷32＋1)×176 ＋ 836×d
7	データ型宣言	2220
8	文字列型宣言(ワイド文字列を含む)	1716
9	列挙型宣言	1824 ＋ ((j－1)÷32＋1)×176 ＋ 64×j
10	シーケンス型宣言	2228
11	構造体宣言	1712 ＋ ((h－1)÷32＋1)×176 ＋ 836×h
12	共用体宣言	2436 ＋ ((i－1)÷32＋1)×176 ＋ 972×i
13	固定小数点型宣言	1716
14	配列宣言	2228

記号	項目	意味
a	包含数	包含する型宣言数
b	継承数	インタフェース宣言が継承するインタフェース数
c	定数値長	定数宣言の値の長さ
d	例外構造体メンバ数	例外宣言の構造体のメンバ数
e	パラメタ数	オペレーション宣言でのパラメタ数
f	コンテキスト数	オペレーション宣言でのコンテキスト数
g	例外数	オペレーション宣言での例外数
h	構造体メンバ数	構造体宣言でのメンバ数
i	共用体メンバ数	共用体宣言でのメンバ数
j	列挙型メンバ数	列挙型宣言でのメンバ数

注6)
ログファイルのサイズの上限値は、CORBAサービスのconfigファイルのlog_file_sizeで設定することができます。アプリケーションごとにJVxxxxxxxxxx.log/JVxxxxxxxxxx.old(xxxxxxxxxxは一意の数字)の名前で採取されます。なお、ログファイルは、不要になった時点で、削除してください。

注7)
ログファイルのサイズの上限値は、HTTPトンネリングの“gwconfigファイル”の“max_log_file_size”で設定することができます。ディスク容量は、バックアップファイルを1つ残すため“max_log_file_size”で指定した値×2”となります。また、SolarisまたはLinuxで、Webサーバに Interstage HTTP Serverを使用している場合は、Interstage HTTP Serverの通信プロセスごとにログファイルが作成されます。なお、ログファイルは、不要になった時点で、削除してください。

機能: イベントサービス

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows イベントサービスのインストールディレクトリ¥etc Solaris Linux /etc/opt	Windows 0.1 以上 Solaris Linux 1.0 以上	チャネル情報
Windows イベントサービスのインストールディレクトリ¥var Solaris Linux /var/opt	61(Mバイト) ＋ essetcnfコマンドの-s -logsizeオプションの指定値×2(Kバイト)以上	ログ情報
	Windows Solaris traceconfigファイルのtrace_size × トレースファイルの世代数 Linux トレースの採取方法によって異なります。(注1)	トレース情報
Interstage管理コンソールで保存先(新規作成)の格納ディレクトリで指定、またはイベントサービスのユニット定義ファイルの“trandir”、“sysdir”、“userdir”で指定	Windows 38 × イベントサービスで作成したユニット数以上 (注2) Solaris Linux 運用の内容により、必要とするサイズを検討してください。(注2)	不揮発チャネル運用時

注1)

－ プロセス単位で内部トレースを採取する
(traceconfigファイルのtrace_buffer = process)場合

traceconfigファイルのtrace_size × イベントチャネルのプロセス数 ×
トレースファイルの世代数

イベントチャネルのプロセス数 ＝
静的イベントチャネルグループ数 ＋ 動的イベントチャネルのプロセス数

動的イベントチャネルのプロセス数:

イベントサービスのセットアップコマンド(essetup)による-pオプションの設定値。
ノーティフィケーションサービスを使用している場合は、“動的イベントチャネルのプロセス数×2”としてください。

－ イベントサービス単位で内部トレースを採取する
(traceconfigファイルのtrace_buffer = system)場合

traceconfigファイルのtrace_size × トレースファイルの世代数

注2)
Interstage管理コンソールで設定した場合は、保存先(新規作成)の格納ディレクトリには、以下の容量が必要となります。

－ イベントデータ用ファイル容量

－ システム用ファイル容量

－ トランザクション用ファイル容量:
((トランザクション多重度 × 4) ＋ 256 ＋
(1トランザクション内最大メッセージサイズ × 2)) × 16 (KB)

ユニット定義ファイルで設定した場合は、各ユニット定義ファイルで指定した以下の容量が必要となります。

－ “sysdir”で指定したディレクトリには、“syssize”で指定したサイズ

－ “userdir”で指定したディレクトリには、“usersize”で指定したサイズ

－ “trandir”で指定したディレクトリには、
((tranmax×4) ＋ 256 ＋ (tranunitmax × 2)) × 16 (Kバイト)

機能: Portable-ORB

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Portable-ORBインストールディレクトリ(注1) Solaris Linux /var/opt (注1)	(注2)	ログ情報

注1)
アプレットとして動作する場合は、アプレットが動作するクライアントマシン上のローカルディスクに、porbeditenvコマンドで“ログ格納ディレクトリ”として指定したディレクトリとなります。

注2)
porbeditenvコマンドで“ログ情報を採取”を指定した場合、
設定した“ログファイルサイズ” × 2 × 動作するアプリケーション/アプレット数

機能: コンポーネントトランザクションサービス

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows コンポーネントトランザクションサービスのインストールディレクトリ¥var Solaris /var/opt/FSUNtd/	25以上	ログトレースファイル
Solaris /opt/FSUNtd/etc/isreg(Interstageの動作環境ディレクトリ) Linux /opt/FJSVtd/etc/isreg(Interstageの動作環境ディレクトリ)	15.0 以上	動作環境

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Solaris Linux /var	運用の内容により、必要とするサイズを検討してください。(注)	異常終了した場合に採取されるcoreファイル

注)
ディスク所要量の算出方法は以下のとおりです。

CORBAサービス関連の共有メモリサイズ(*1) × 3 ＋
ワークユニット数 × 0.26 ＋
基本サイズ(*2)

－ *1:CORBAサービス関連の共有メモリサイズ
CORBAサービスのconfigファイル(/opt/FJSVod/config)中の各パラメタから以下の式で算出します。

limit_of_max_IIOP_resp_con × 0.016 ＋
limit_of_max_IIOP_resp_requests × 0.016 ＋
max_impl_rep_entries × 0.006 ＋ 0.01

－ *2: 基本サイズ
コンポーネントトランザクションサービスの環境定義ファイル(/var/opt/FJSVtd/etc/sysdef)のSystem Scale: ステートメントに指定した値に応じて、以下のようになります。

small　: 270
moderate : 350
large　: 860
super　: 1420

機能: データベース連携サービス

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Solaris Linux －	(注)	システムログファイル
Windows ログファイル格納ディレクトリ	トランザクション数 × 0.008 ＋ 0.001	データベース連携サービス運用時
トレースログファイル格納ディレクトリ	運用環境の OTS_TRACE_SIZE × 0.001	
リソース管理トレースログファイル格納ディレクトリ	運用環境の RESOUCE_TRACE_SIZE × 0.001	
リカバリトレースログファイル格納ディレクトリ	運用環境の RECOVERY_TRACE_SIZE × 0.001	
監視プロセストレースログファイル格納ディレクトリ	運用環境の OBSERVE_TRACE_SIZE × 0.001	
Windows リソース定義ファイル格納ディレクトリ	登録したリソース定義ファイル数 × 0.001	
Solaris /opt/FSUNots/var (otsgetdumpコマンドによるダンプファイル格納ディレクトリ)	5.0 以上	

注)
データベース連携サービスのシステムログファイルは、isgendefコマンドで指定したシステム規模により異なりますので、以下のとおり見積もってください。

small　: 1Mバイト以上
moderate : 2Mバイト以上
large　: 8Mバイト以上
super　: 16Mバイト以上

機能: 性能監視ツール Solaris Linux

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
格納ディレクトリ	1.0以上 (注1)	性能ログファイル
Solaris /var	6.4 × 性能監視ツールの共有メモリサイズ(注2) × 6	異常終了した場合に採取されるcoreファイル

注1)
所要量 ＝ ispmakeenvで指定する共用メモリサイズ × (測定時間 ÷ インターバル時間) × 測定日数(日)

注2)
ispmakeenvコマンドの-mオプションで指定する共有メモリサイズです。

機能: SOAPサービス

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows IJServerワークユニットlogディレクトリ Solaris Linux /var/opt (Servletサービスのログ情報格納ディレクトリ)	IJServerで使用する資源と共通です。詳細は、IJServerワークユニットの注1)を参照してください。	IJServerのログ情報

機能: UDDIレジストリサービス Windows Solaris

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage Application Serverのインストールパス Solaris Linux /var/opt (UDDIレジストリサービスのログを格納するディレクトリ)	20(デフォルト設定でのログ出力で利用するディスク容量) 設定は、UDDIレジストリサービス環境設定ファイルで変更可能	詳細は、Interstage V9.1.0の“UDDIサービスユーザーズガイド”を参照してください。
UDDIレジストリサービスで情報を格納するDSAのディレクトリ	80 × 8 ＋ 情報登録ディスク容量	

機能: MessageQueueDirector

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
―	運用の内容により、必要とするサイズを検討してください	詳細は“MessageQueueDirector 説

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
		明書”の“ファイル容量の見積り”を参照してください

機能: フレームワーク

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
Java VMのシステムプロパティのjava.io.tmpdirで指定	Windows クライアント(Webブラウザ)からアップロードされるファイルサイズ Solaris Linux 運用の内容により、必要とするサイズを検討してください。	ファイルアップロード機能の使用時 (注)
フレームワークのログ機能で指定したファイルが格納されるディレクトリ	運用の内容により、必要とするサイズを検討してください。	フレームワークのログ機能の使用時

注)
このディレクトリには、WebブラウザからアップロードされたファイルのサイズがWebアプリケーションの指定したファイル転送用メモリサイズを超えた場合に、アップロードされたファイルが格納されます。

1.1.2 マルチサーバ管理を使用する場合

マルチサーバ管理を使用する場合、以下に示すディスク容量が追加で必要です。

■管理サーバ機能を使用する場合

管理サーバ機能を使用する場合は以下にあげる項目に関するディスク容量が必要です。ディレクトリおよび容量については“1.1.1 サーバ機能を使用する場合”の同一項目名の記述を参照してください。

- Interstage管理コンソール
- Interstage HTTP Server
- Interstage ディレクトリサービス
- Interstage JMXサービス
- 業務構成管理

また、以下のディスク容量も必要です。

ディレクトリ (デフォルト)	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage JMXサービスのインストールディレクトリ¥var¥ssv_ijs Solaris Linux /var/opt/FJSVisjmx/ssv_ijs	配備するアプリケーションのサイズ × 2	Interstage管理コンソールからのアプリケーションの配備

ディレクトリ (デフォルト)	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage管理コンソールのインストールディレクトリ¥isAdmin¥var¥download Solaris Linux /var/opt/FJSVisgui/tmp/download	(注)	Interstage管理コンソールからのログ参照

注)
Interstage管理コンソールの以下の統合管理画面において、ログファイルをダウンロードする場合、同時にダウンロードするログファイルのサイズ分のディスク容量が一時的に必要となります。

機能	画面(スタンドアロン)
Interstage HTTP Server	[統合管理] > [Interstage管理コンソール] > [Interstage Application Server] > [サーバグループ名] > [サーバ名] > [システム] > [サービス] > [Webサーバ] > [FJapache] > [ログ参照]タブ
	[統合管理] > [Interstage管理コンソール] > [Interstage Application Server] > [サーバグループ名] > [サーバ名] > [システム] > [サービス] > [Webサーバ] > [FJapache] > [バーチャルホスト] > [バーチャルホスト名] > [ログ参照]タブ
Webサーバコネクタ	[統合管理] > [Interstage管理コンソール] > [Interstage Application Server] > [サーバグループ名] > [サーバ名] > [システム] > [サービス] > [Webサーバ] > [FJapache] > [Webサーバコネクタ] > [ログ参照]タブ
IJServerワークユニット	[統合管理] > [Interstage管理コンソール] > [Interstage Application Server] > [サーバグループ名] > [サーバ名] > [システム] > [ワークユニット] > [ワークユニット名] > [ログ参照]タブ

ログファイルのサイズについては、各機能のログ情報のディスク容量を参照し、運用の内容により必要とするサイズを検討してください。
なお、ログファイルのサイズが大きいため、ディスク容量の不足によりログファイルのダウンロードに失敗する場合は、FTPなどを使用してダウンロードしてください。

注意
管理サーバ機能は以下の製品で利用可能です。

- Interstage Application Server Enterprise Edition

■管理対象サーバとして運用する場合

管理対象サーバとして運用する場合は、使用する機能に関するディスク容量が必要です。ディレクトリおよび容量については“1.1.1 サーバ機能を使用する場合”の同一項目名の記述を参照してください。

また、以下のディスク容量も必要です。

ディレクトリ (デフォルト)	ディスク容量 (単位:Mバイト)	備考(用途)
Windows Interstage JMXサービスのインストールディレクトリ¥etc Solaris Linux /etc/opt/FJSVisjmx	1	サイトへ追加される場合

■共存サーバとして運用する場合

共存サーバでは管理サーバ機能とInterstageのサーバ機能(管理対象サーバ)が同一マシン上で動作しています。上記の記事を参照して、管理サーバ機能とInterstageのサーバ機能で使用するサービスを列挙してください。各サービスが必要とするディスク容量は“1.1.1 サーバ機能を使用する場合”の同一項目名の記述を参照してください。

管理サーバ機能とInterstageのサーバ機能で同一サービスを使用する場合、そのサービスの必要資源量を2倍する必要はありません。

1.1.3 クライアント機能を使用する場合

本ソフトウェアを以下の運用で動作させるとき、各ディレクトリにはインストールに必要な“静的ディスク資源”に加えて以下のディスク容量が必要です。空き容量が足りない場合は、該当するファイルシステムのサイズを拡張してください。

機能: CORBAサービス

注) Interstage Web Serverでは、本機能を使用できません。

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
/var/opt	(注1)	ログ情報(プレインストール型Javaライブラリ以外の場合)
任意(Java VMのシステムプロパティのuser.dirで指定)	(注2)	ログ情報(プレインストール型Javaライブラリの場合)
/tmp	1.0 以上 IDL定義の量に依存	IDLコンパイラ動作時

注1)

ログファイルのサイズの上限値は、CORBAサービスのconfigファイルのlog_file_sizeで設定することができます。ディスク容量は、バックアップファイルを1つ残すため“ログファイルサイズの上限値×2”となります。configファイルの詳細については、“CORBAサービスの動作環境ファイル”－“A.1 config”を参照してください。なお、ログファイルは、不要になった時点で、削除してください。

注2)

ログファイルのサイズの上限値は、CORBAサービスのconfigファイルのlog_file_sizeで設定することができます。アプリケーションごとにJVxxxxxxxxxx.log/JVxxxxxxxxxx.old(xxxxxxxxxxは一意の数字)の名前で採取されます。なお、ログファイルは、不要になった時点で、削除してください。

機能: Portable-ORB

注) Interstage Web Serverでは、本機能を使用できません。

ディレクトリ	ディスク容量 (単位:Mバイト)	備考(用途)
/var/opt (注)	“ログファイルサイズ”×2×動作するアプリケーション/アプレット数	porbeditenvコマンドで“ログ情報を採取”を指定した場合のログ情報

注)

アプレットとして動作する場合は、アプレットが動作するクライアントマシン上のローカルディスクに、porbeditenvコマンドで“ログ格納ディレクトリ”として指定したディレクトリとなります。

1.2 メモリ容量

本ソフトウェアを動作させるために必要なメモリ容量は次のとおりです。

動作させるために必要なメモリ容量が確保されていない場合、動作に不具合が生じる場合があります。

1.2.1 サーバ機能を使用する場合

以下の各機能を使用する場合のメモリ所要量を示します。

- Interstage管理コンソール
- Interstage HTTP Server
- Interstage JMXサービス
- Interstage シングル・サインオン
- Interstage ディレクトリサービス
- IJServerワークユニット
- Session Registry Server
- CORBAサービス
- イベントサービス／ノーティフィケーションサービス
- Portable-ORB
- コンポーネントトランザクションサービス
- データベース連携サービス
- SOAPサービス
- MessageQueueDirector
- フレームワーク

機能: Interstage管理コンソール

メモリ所要量(単位:Mバイト)	備考
121.0 以上	

機能: Interstage HTTP Server

メモリ所要量(単位:Mバイト)	備考
Windows **Windows Server(R) x64 Editionsの場合** 25.2 ＋ (0.04 × m) ＋ (0.12 × n) **Windows Server(R) for Itanium-based Systemsの場合** 43.8 ＋ (0.05 × m) ＋ (0.1 × n) Linux **Linux for Intel64の場合** 8.0 ＋ (3.0 × n) **Linux for Itaniumの場合** 21.0 ＋ (6.0 × n) m: 環境定義ファイルで指定した最大リクエスト同時処理数(httpd.conf ファイルのThreadsPerChildディレクティブの値) n: クライアントからのHTMLファイル同時アクセス数	HTMLファイルを複数クライアント同時アクセス時

機能: Interstage JMXサービス

メモリ所要量(単位:Mバイト)		備考
Windows	Linux	
90.0 以上	200.0 以上	

機能: Interstage シングル・サインオン

メモリ所要量(単位:Mバイト)	備考
10.0 以上 (注1)	業務サーバ機能
10.0 以上 (注2)	認証サーバ機能
10.0 以上 (注3)	リポジトリサーバ機能

注1)
運用に応じて以下の式で見積もった値を加算してください。(単位:バイト)

－ (2,400 ＋ (ロール数 ＋ ロールセット数 ＋ (ロールセット数 × ロール数)) × 2,048以上) × パス定義数 + 業務サーバ数 × (2,000,000 ＋ キャッシュサイズ × キャッシュ数)

ロール数　　: SSOリポジトリに定義した保護リソースの、チューニングを行う業務サーバのパス定義に設定したロールの総数
ロールセット数 : SSOリポジトリに定義した保護リソースの、チューニングを行う業務サーバのパス定義に設定したロールセットの総数
パス定義数　: SSOリポジトリに定義した保護リソースの、チューニングを行う業務サーバのパス定義の総数
キャッシュサイズ: “F.4 業務サーバを構築する場合のチューニング”を参照してください。
キャッシュ数　: “F.4 業務サーバを構築する場合のチューニング”を参照してください。

注2)
セション管理を行わない場合は、運用に応じて以下の式で見積もった値を加算してください。(単位:バイト)

－ ((サイト定義数 × 1,024) ＋ (パス定義数 × 1,024)) × 2

サイト定義数: SSOリポジトリに定義したサイト定義の総数
パス定義数 : SSOリポジトリのすべてのサイト定義に定義したパス定義の総数

統合Windows認証を行う場合は、256Mバイトを加算してください。

認証サーバ間連携を行う場合は、256Mバイトを加算してください。

注3)
運用に応じて以下の式で見積もった値を加算してください。(単位:バイト)

－ ((ロール数 ＋ ロールセット数 ＋ ロールセット数 × ロール数) × 2,048以上) × 2

ロール数　　: SSOリポジトリに定義したロールの総数
ロールセット数: SSOリポジトリに定義したロールセットの総数

セション管理を行う場合は、上記の算出値に、以下の式から算出される値を加算してください。

－ 23,500,000 ＋ ((同時にシングル・サインオンシステムを使用する利用者数 × (2,560 ＋ α)) × 2)

【α :拡張ユーザ情報】
通知する拡張ユーザ情報の数に応じて、以下の値を加算する。
　通知する拡張ユーザ情報のサイズ × 2

ユーザ情報を登録するディレクトリサービスにActive Directoryを使用し、シングル・サインオンの拡張スキーマを使用しない場合は、上記の算出値に、以下の式から算出される値を加算してください。

－ Active Directoryのロール/ロールセットに使用する属性の総数 × 524 × 2

機能: Interstage ディレクトリサービス

メモリ所要量(単位:Mバイト)			備考
Windows	Solaris	Linux	
340.0 以上 (注1)	150.0 以上 (注1)	217.0以上 (注1)	スタンドアロン、データベース共用、またはスレーブで運用する場合 (注2)
50.0 以上 (前項のスタンドアロンで運用する場合に加えて必要となる値)			マスタで運用する場合 (注2)
2.0 以上			エントリ管理コマンドを使用する場合
22.0 以上	60.0 以上	60.0 以上	エントリ管理ツールを使用する場合
m × n × 3 m: 1エントリの登録に使用したLDIFファイルのサイズ n: 検索により通知されるエントリ数			Interstage ディレクトリサービス SDK 検索時

注1)
リポジトリを複数作成して運用する場合は、リポジトリ数を乗算してください。

注2)
表中の“マスタ”、および“スレーブ”は、リポジトリのデータベースに標準データベースを使用したレプリケーション形態で運用する場合の、マスタ、およびスレーブサーバについて説明しています。
リポジトリのデータベースにRDBを使用したレプリケーション形態で運用する場合の、マスタ、およびスレーブサーバのメモリ所要量は、スタンドアロンで運用する場合と同じです。

Interstage ディレクトリサービスは、以下の製品で利用可能です。

- Interstage Application Server Enterprise Edition

機能: IJServerワークユニット

“WebアプリケーションとEJBアプリケーションを同一JavaVMで運用”時 (注1)(注2)

メモリ所要量(単位:Mバイト)		備考(以下のサンプルアプリケーションを運用した場合)
Windows	Linux	
59.6 以上	114.3 以上	EjbBmp(Web,Session,BMP)
62.1 以上	115.1 以上	EjbCmp11(Web,Session,CMP1.1)
65.8 以上	118.8 以上	EjbCmp20(Web,Session,CMP2.0)
71.6 以上	125.3 以上	EjbMessageDriven(Web,Session,MDB)

“Webアプリケーションのみ運用”時 (注1)

メモリ所要量(単位:Mバイト)		備考(以下のサンプルアプリケーションを運用した場合)
Windows	Linux	
51.6 以上	94.4 以上	HelloServlet(Web)

“EJBアプリケーションのみ運用”時 (注2)

メモリ所要量(単位:Mバイト)		備考(以下のサンプルアプリケーションを運用した場合)
Windows	Linux	
54.3 以上	106.3 以上	EjbBmp(Session,BMP)
55.8 以上	107.6 以上	EjbCmp11(Session,CMP1.1)
57.9 以上	111.8 以上	EjbCmp20(Session,CMP2.0)
59.8 以上	118.3 以上	EjbMessageDriven(Session,MDB)

注1)
詳細は以下の式で見積もってください。(単位:Mバイト)

- Webサーバコネクタ

Windows

0.2 × k ＋ 2.0

Solaris

1.9 × k ＋ 30.0

Linux

1.0 × k ＋ 30.0

k: Servletサービスへの同時アクセス数

- IJServerワークユニット:(プロセス多重度1当り)

― “WebアプリケーションとEJBアプリケーションを運用”の場合

Windows

48.0 ＋ (1.4 × k) ＋ (0.7 × w) ＋ (P1＋P2＋P3＋..＋Pn)

Solaris

121.0 ＋ (2.1 × k) ＋ (0.7 × w) ＋ (P1＋P2＋P3＋..＋Pn)

Linux

28.0 ＋ (1.5 × k) ＋ (0.7 × w) ＋ (P1＋P2＋P3＋..＋Pn)

― “Webアプリケーションのみ運用”の場合

Windows

47.0 ＋ (1.3 × k) ＋ (0.7 × w) ＋ (P1＋P2＋P3＋..＋Pn)

Solaris

84.0 ＋ (2.5 × k) ＋ (0.7 × w) ＋ (P1＋P2＋P3＋..＋Pn)

Linux

27.0 ＋ (1.3 × k) ＋ (0.7 × w) ＋ (P1＋P2＋P3＋..＋Pn)

k: Servletコンテナへの同時アクセス数
w: Webアプリケーションの数
Pn: 各ServletまたはJSPの実行サイズ。上記表では1Mバイトとして計算

セションリカバリ機能(Session Registry Client)を使用する場合は、以下の値を加算してください。

― (0.002 ＋ セションの保持するデータ容量(※)) × 想定されるセション数
※: Webアプリケーションでセションの属性(Attribute)にセットするオブジェクトおよびキーのサイズの合計値

なお、セションリカバリ機能(Session Registry Client)は、Interstage Application Server Enterprise Edition、Interstage Application Server Standard-J Editionで運用可能です。

ServletはJava VM上で動作するため、実際のメモリ使用量(ヒープ領域を含む)は、以下に示す要因により異なります。

― newするクラス型

― newするインスタンスの個数

― インスタンスのライフサイクル

ー GCの動作状況

ー IJServerワークユニットの各種定義

ー 使用するJava VM

そのため正確なメモリ使用量(ヒープ領域、Perm領域)は次のようにして実測することにより見積もることを推奨します。

ー 本番運用のピーク時と同一条件で動作させます。Java VMが使用するメモリが不足すると、イベントログにメッセージが出力されますので、ヒープ領域やPerm領域の最大値を増やして、最適な値としてください。求めたヒープ領域やPerm領域の最大値をそのまま本番運用時の値として利用します。

注2)

以下を参考に、EJBサービス運用時のメモリ所要量を見積もってください。

EJBアプリケーション運用時、Java VMが使用するメモリ量(初期値、最大値)および1プロセスで必要な全メモリ量は、以下に示す要因により異なります。

ー newするクラス型

ー newするインスタンスの個数

ー インスタンスのライフサイクル

ー GCの動作状況

ー EJBアプリケーションの各種定義

いずれのメモリ量も簡単には算出できないので、次のようにして実測することにより見積もってください。

1. Java VMが使用するメモリ量の初期値(javaコマンドの-Xmsオプションで指定する値)
   EJBアプリケーションを、本番運用の通常時(ピーク時ではない)と同一条件で動作させます。Java VMが使用するメモリ量(最大値)が不足すると、IJServer21033またはEJB1033メッセージが出力されますので、試行錯誤によりメモリ量(最大値)を最適な値としてください。このようにして求めたメモリ量(最大値)を本番運用時のメモリ量(初期値)として利用します。メモリ量(初期値)の省略値は2Mバイトです。
2. Java VMが使用するメモリ量の最大値(javaコマンドの-Xmxオプションで指定する値)
   EJBアプリケーションを、本番運用のピーク時と同一条件で動作させます。Java VMが使用するメモリ量(最大値)が不足すると、IJServer21033またはEJB1033メッセージが出力されますので、試行錯誤によりメモリ量(最大値)を最適な値としてください。このようにして求めたメモリ量(最大値)をそのまま本番運用時のメモリ量(最大値)として利用します。メモリ量(最大値)の省略値は64Mバイトです。
3. 1プロセスで必要な全メモリ量
   1)と2)でJava VMが使用するメモリ量を見積り時、同時に1プロセスで必要な全メモリ量も実測して見積もってください。

機能: Session Registry Server

メモリ所要量(単位:Mバイト)			備考
Windows	Solaris	Linux	
	(例)254 (注)	(例)120 (注)	

注)

詳細は以下の式で見積もってください。(単位:Mバイト)

Windows

Solaris

85.7 ＋ (2.5 × k) ＋ (0.01 × a) ＋ ((0.002 ＋ d) × s) × 2

Linux

28.7 ＋ (1.3 × k) ＋ (0.01 × a) ＋ ((0.002 ＋ d) × s) × 2

k: Session Registry Serverの同時処理数
a: (IJServerに配備している)Webアプリケーションの数
d: セションの保持するデータ容量 ＝

Webアプリケーションでセションの属性(Attribute)にセットするオブジェクトおよびキーのサイズの合計値。
s: セション数

例: 対象とするIJServerは同時処理数64、アプリケーション1つ、セションに格納するデータ量が2KB、セション数が1000の場合。

Windows

Solaris

85.7 ＋ (2.5 × 64) ＋ (0.01 × 1) ＋ ((0.002 ＋ 0.002) × 1000) × 2
＝ 85.7 ＋ 160 ＋ 0.01 ＋ 8
≒ 254

Linux

28.7 ＋ (1.3 × 64) ＋ (0.01 × 1) ＋ ((0.002 ＋ 0.002) × 1000) × 2
＝ 28.7 ＋ 83.2 ＋ 0.01 ＋ 8
≒ 120

Session Registry ServerはJava VM上で動作するため、実際のメモリ使用量(ヒープ領域を含む)は、負荷やGCの動作状況により異なります。
そのため正確なメモリ使用量は次のようにして実測することにより見積もることを推奨します。

ー 本番運用のピーク時と同一条件で動作させます。Java VMが使用するメモリが不足すると、イベントログにメッセージが出力されますので、ヒープ領域の最大値を増やして、最適な値としてください。求めたヒープ領域の最大値をそのまま本番運用時の値として利用します。

なお、Session Registry Serverは、Interstage Application Server Enterprise Editionで運用可能です。

機能: CORBAサービス

メモリ所要量(単位:Mバイト)	備考
16.0 以上 (注1)	
8.0 以上	ネーミングサービス運用時
45.6 以上 (注2)	インタフェースリポジトリ運用時
2.4	COBOL Webサブルーチン使用時

注1)
CORBAサービスの動作環境定義(configファイル)の設定により、16Mバイト ＋ 加算値(下表)が必要です。

運用形態	必要数(加算値)(単位:Kバイト)
CORBAサービス運用時	100.0 ＋ max_IIOP_resp_con × 16.0 ＋ max_IIOP_resp_requests × 16.0 ＋ max_impl_rep_entries × 6.0 (以上)
トレース機能を使用する場合	(CORBAサービス運用時) ＋ 20.0 ＋ max_processes × trace_size_per_process (以上)
スナップショット機能を使用する場合	(CORBAサービス運用時) ＋ 10.0 ＋ snap_size (以上)

また、クライアントパッケージのCORBAアプリケーションを動作させる場合、1プロセスあたり1.5 Mバイトのメモリが必要となります。

注2)
インタフェースリポジトリは、起動時にデータベースに格納されているオブジェクトをメモリ上に展開します。インタフェースリポジトリを使用する場合のメモリ容量について説明します。

ー 固定使用領域
45.6 Mバイト

ー 可変使用領域

インタフェースリポジトリでは、オブジェクトごとにメモリが使用されます。
以下の計算式より、オブジェクトごとの使用メモリを算出することができます。

項番	IDL定義	計算式(単位:バイト)
1	モジュール宣言	3902＋a×(2×b＋2)
2	インタフェース宣言	3902＋a×(2×b＋2)＋a×b×c
3	オペレーション宣言	3934＋a×(3×b＋2＋f)＋a×b×g＋h×(12＋a＋a×b)
4	属性宣言	3910＋a×(3×b＋2)
5	定数宣言	7704＋a×(3×b＋3)＋d
6	例外宣言	3836＋a×(2×b＋e＋1)＋e×(78＋a＋a×b)
7	文字列型宣言(ワイド文字列を含む)	3882＋a×(b＋1)
8	列挙型宣言	3918＋a×(2×b＋k＋2)
9	シーケンス型宣言	3882＋a×(2×b＋1)
10	構造体宣言	3766＋a×(2×b＋i＋1)＋i×(78＋a＋a×b)
11	共用体宣言	3840＋a×(3×b＋j＋1)＋j×(3880＋2×a＋a×b)
12	固定小数点型宣言	3882＋a×(b＋1)
13	配列宣言	3886＋a×(2×b＋1)

記号	項目	意味
a	識別子長	対象オブジェクトの識別子の長さ
b	階層数	対象オブジェクトの存在する階層
c	継承数	インタフェース宣言が継承するインタフェース数
d	定数値長	定数宣言の値の長さ
e	例外構造体メンバ数	例外宣言の構造体のメンバ数
f	コンテキスト数	オペレーション宣言でのコンテキスト数
g	例外数	オペレーション宣言での例外数
h	パラメタ数	オペレーション宣言でのパラメタ数
i	構造体メンバ数	構造体宣言でのメンバ数
j	共用体メンバ数	共用体宣言でのメンバ数
k	列挙型メンバ数	列挙型宣言でのメンバ数

機能: イベントサービス／ノーティフィケーションサービス

メモリ所要量(単位:Mバイト)			備考
Windows	Solaris	Linux	
16.0 以上	8.0 以上	8.0 以上	
ユニット数 × 100 ＋ イベントサービスのユニット定義ファイルのshmmaxの合計			不揮発チャネル運用時
(a＋b)×c (Kバイト) (注1)			essetcnfコマンド実行時に静的生成のイベントチャネルのコンシューマ数・サプライヤ数を拡張する場合

メモリ所要量(単位:Mバイト)			備考
Windows	Solaris	Linux	
(a＋b)×d (Kバイト) (注1)			essetcnfコマンド実行時に動的生成のイベントチャネルのコンシューマ数・サプライヤ数を拡張する場合
(a＋b)×(c－e)＋(f＋g)×e (Kバイト) (注1)			essetcnfおよびessetcnfchnlコマンドを併用して静的生成のイベントチャネルのコンシューマ数・サプライヤ数を拡張する場合
メッセージ本文のサイズ×蓄積メッセージ数			イベントチャネルに蓄積するイベントデータの形式に、any型を使用する場合 (注2)
(メッセージ本文のサイズ＋(QoSプロパティ項目数×4Kバイト))×蓄積メッセージ数			イベントチャネルに蓄積するイベントデータの形式に、StructuredEvent型を使用する場合 (注2)

注1)
a:essetcnfコマンドの-coninitオプションで指定するコンシューマ数の初期値の拡張数(初期設定値からの差分)
b:essetcnfコマンドの-supinitオプションで指定するサプライヤ数の初期値の拡張数(初期設定値からの差分)
c:イベントチャネルのグループ数
d:essetcnfコマンドの-dchmaxオプションで指定するイベントチャネルの最大起動数
e:essetcnfchnlコマンドで設定するイベントチャネルのグループの数
f:essetcnfchnlコマンドの-coninitオプションで指定するコンシューマ数の初期値の拡張数(初期設定値からの差分)
g:essetcnfchnlコマンドの-supinitオプションで指定するサプライヤ数の初期値の拡張数(初期設定値からの差分)
essetcnfコマンドおよびessetcnfchnlコマンドの詳細については、“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。

注2)
イベントサービスの形式については、“アプリケーション作成ガイド(イベントサービス編)”の“イベントデータの形式”を参照してください。

機能: Portable-ORB

メモリ所要量(単位:Mバイト)	備考
3.0 以上	

機能: コンポーネントトランザクションサービス

メモリ所要量(単位:Mバイト)			備考
Windows	Solaris	Linux	
48.0 以上 (注1)	50.0 以上 (注2)		サービスの起動
－	4.0 以上 (注3)		

注1)
この値はCORBAサービスのメモリ容量を含んでいませんので、加算してください。

注2)
ユーザ認証機能を使用する場合は、0.9Mバイト加算してください。
アクセス制御を使用する場合は、0.6Mバイト加算してください。

注3)
1つのワークユニットでプロセス多重度を1とした場合の値です。
詳細は以下の式で見積もってください。

－ 4.0 × ワークユニット配下のプロセス数の総和

機能: データベース連携サービス Windows Solaris

Windows

メモリ所要量(単位:Mバイト)	備考
18.0 ＋ 10.0 × n ＋ 0.008 × m n: リソース管理ごとの多重度＋1の総数 m: 最大トランザクション数	(データベース連携サービス動作マシン上の)サービスの起動
18.0 ＋ 10.0 × n ＋ 0.008 × m n: リソース管理ごとの多重度＋1の総数 m: 最大トランザクション数	(リソース管理プログラムだけの起動するマシン上の)サービスの起動

Solaris

メモリ所要量(単位:Mバイト)	備考
8.0 以上 (注)	(OTSシステムとリソース管理プログラムを起動するマシン上の)サービスの起動 最大トランザクション数512の場合
4.0 以上 (注)	(リソース管理プログラムだけの起動するマシン上の)サービスの起動

注)
詳細は以下の式で見積もってください。

－ 4.0 ＋ 2.0(OTSシステムを起動するマシン上) ＋
0.004 × 最大トランザクション数(OTSシステムを起動するマシン上)

機能: SOAPサービス

メモリ所要量(単位:Mバイト)		備考
Windows	Linux	
(注1)	64.0 以上 (注2)	

注1)
IJServerワークユニットのメモリ容量 ＋ 16.0 以上
さらに、SOAPサーバアプリケーションのメモリ容量を加算してください。

注2)
詳細は以下の式で見積もってください。

－ (64.0 × c) ＋ (s ＋ k) × (P ＋ 0.1) (Mバイト)
c: Servletコンテナの起動数
s: SOAPサービスへのセション数
k: SOAPサービスへの同時アクセス数
P: SOAPサーバアプリケーションひとつあたりの実行サイズ

機能: MessageQueueDirector

注) MQDシステムが複数ある場合には、それぞれのMQDシステムについて見積もった値の合計が所要量になります。

メモリ所要量(単位:Mバイト)	備考
100.0 ＋ m m: MQD環境定義のMQDConfigurationセクションの MessageBufferMaxSize	基本機能使用時
39.0 ＋ sc × 0.3 ＋ rc × 0.3 以上 sc: イベントチャネル連携サービスのCHANNELセクション定義数 rc: イベントチャネル連携サービスのRCHANNELセクション定義全部の総集信数	イベントチャネル連携サービス使用時

機能: フレームワーク

Windows

メモリ所要量(単位:Mバイト)	備考
Application Serverが使用するメモリ使用量 ＋ 32.0	

Solaris Linux

メモリ所要量(単位:Mバイト)	備考
2.9 [参考値] (注)	サンプル“model”を実行した場合

注)
フレームワークを使用して作成したWebアプリケーションを運用する場合、必要となるメモリ容量は、Servletサービスの運用に必要となるメモリ容量に含めて見積もってください。IJServerワークユニットの注1)の計算式のPn(各サーブレットまたはJSPの実行サイズ)の値として、Webアプリケーションのメモリ使用量を適用してください。この値は、フレームワークのサンプル“model”の場合、2.9Mバイトです。なお、Servletサービスの運用に必要となるメモリ容量は、IJServerワークユニットの注1)に記載した方法で実測によって見積もることができます。
フレームワークを使用して作成したEJBアプリケーションを運用する場合、必要となるメモリ容量は、EJBサービスの運用に必要となるメモリ容量に含めて、IJServerワークユニットの注2)に記載した方法で見積もってください。
フレームワークを使用して作成したSOAPサーバアプリケーションを運用する場合、必要となるメモリ容量は、SOAPサービスの運用に必要となるメモリ容量に含めて、SOAPサービスの注2)に記載した方法で見積もってください。

1.2.2 マルチサーバ管理を使用する場合

マルチサーバ管理を使用する場合、以下に示すメモリ容量が追加で必要です。

■管理サーバ機能を使用する場合

管理サーバ機能を使用する場合は以下にあげる項目に関するメモリ容量が必要です。容量については“1.2.1 サーバ機能を使用する場合”の同一項目名の記述を参照してください。

- Interstage管理コンソール
- Interstage HTTP Server
- Interstage ディレクトリサービス
- Interstage JMXサービス

管理サーバ機能は以下の製品で利用可能です。

- Interstage Application Server Enterprise Edition

■管理対象サーバとして運用する場合

管理対象サーバとして運用する場合は、使用する機能に関するメモリ容量が必要です。容量については“1.2.1 サーバ機能を使用する場合”の同一項目名の記述を参照してください。

■共存サーバとして運用する場合

共存サーバでは管理サーバ機能とInterstageのサーバ機能(管理対象サーバ)が同一マシン上で動作しています。上記の記事を参照して、管理サーバ機能とInterstageのサーバ機能で使用するサービスを列挙してください。各サービスが必要とするメモリ容量は“1.2.1 サーバ機能を使用する場合”の同一項目名の記述を参照してください。

管理サーバ機能とInterstageのサーバ機能で同一サービスを使用する場合、そのサービスの必要資源量を2倍する必要はありません。

1.2.3 クライアント機能を使用する場合

本ソフトウェアを以下の運用で動作させるときに使用するメモリ容量を示します。

機能: Interstage ディレクトリサービス

注) Interstage Web Serverでは、本機能を使用できません。

メモリ所要量(単位:Mバイト)	備考
22.0 以上	エントリ管理ツールを使用する場合

第2章 Interstageのチューニング

Interstageはシステム規模の指定だけで、システム運用が可能となるようなモデルケースを設定して各サービスの定義を登録しています。しかし、システムによっては、より詳細なシステム構築が必要となります。

Interstageをチューニングする場合は、isregistdefコマンドで各サービスの定義を登録した後で実施してください。チューニングした内容はInterstageの初期化機能で有効となり、Interstageの起動で反映されます。

Interstageのチューニングは、以下の定義ファイルに対して行います。

- Interstage動作環境定義
- CORBAサービスの動作環境ファイル
- コンポーネントトランザクションサービスの環境定義 (注)
- データベース連携サービスのシステム環境定義

注) 本定義はInterstage Application Server Enterprise Editionのみです。

2.1 定義ファイルの設定値

Interstageシステム定義により各定義ファイルに以下の値を設定しています。Interstageシステム定義のステートメントごとに説明します。

- System Scaleステートメント(システム規模)
  - small
  - moderate
  - large
  - super

■システム規模ごとの設定値

定義名	ステートメント	値(System Scaleごとの)			
		small	moderate	large	super
Interstage動作環境定義	Corba Host Name	ありません。			
	Corba Port Number	ありません。			
	IR path for DB file	TD_HOME/var/IRDB (注1)			
	IR USE	ありません。(注2)			
	IR Host Name	ありません。(注2)			
	IR Port Number	8002			
	NS USE	ありません。(注2)			
	NS Host Name	ありません。(注2)			
	NS Port Number	8002			
	NS JP	no			
	NS Locale	EUC			
	TD path for system	/var/opt/FJSVisas/system/default/FJSVextp			
	OTS Multiple degree	5			
	OTS Recovery	2			
	OTS path for system log	ありません。(注3)			
	OTS maximum Transaction	50	100	500	1000
	OTS Setup mode	sys			

定義名	ステートメント	値(System Scaleごとの)			
		small	moderate	large	super
	OTS JTS's RMP Multiple degree of Process	5			
	OTS JTS's RMP Multiple degree of Thread	16			
	OTS Participate	4			
	OTS Host	ありません。			
	OTS Port	ありません。			
	OTS Locale	ありません。			
	Event Service	no			
	Event maximum Process	2			
	Event maximum Connection	50	100	500	1000
	Event Locale	EUC			
	Event Auto Disconnect	no			
	Event SSL	no			
	SSL USE	no			
	SSL Port Number	4433			
	SOAP Client GW	no			
	IS Monitor Mode	mode2			
	FJapache	no			
CORBAサービスの動作環境ファイル(注4)	max_IIOP_resp_con	80	135	575	1024
	max_IIOP_resp_requests	1920	2944	10112	20352
	max_processes	31	36	76	126
	max_exec_instance	448	448	1046	2046
コンポーネントトランザクションサービスの環境定義	[SYSTEM ENVIRONMENT] System Scale	small	moderate	large	super
データベース連携サービスの環境定義	ありません。	ありません。			

注1)

TD_HOME:コンポーネントトランザクションサービスのインストールディレクトリ

注2)

運用形態がTYPE3の場合は必ず指定してください。

注3)

運用形態がTYPE2の場合は必ず指定してください。

注4)

CORBAサービスの動作環境ファイル内の値に表中の値がisregistdefコマンド実行時に**加算**されます。また、isregistdefコマンドの投入が初回でない場合は、前回のコマンド投入時に加算した値を、現在設定されている値から減算し、新たに指定したSystem Scaleの値が加算されます。詳細は、“■CORBAサービスの動作環境ファイルの設定について”を参照してください。

■CORBAサービスの動作環境ファイルの設定について

CORBAサービスの動作環境定義ファイルの定義値は、isregistdefコマンドによるセットアップの実行時に、以下のように設定されます。

- セットアップ実行時に、CORBAサービスの動作環境ファイルの定義値に対し、必要な値が加算されます。
- すでにセットアップ済みの環境に対し、スケールを変更した場合には、加算値の差分の値が反映されます。スケールを大きくした場合には、加算値の差分の値が加算され、スケールを小さくした場合には、加算値の差分の値が減算されます。
- 加算値は、スケールに対して一定です。

以下に定義値max_IIOP_resp_conに対する設定例を示します。

- Interstageがセットアップされていない環境に対し、isregistdefコマンドを実行した場合(システム規模:small)。
  - max_IIOP_resp_conの値8に対して、33を加算した値41が設定される。
  - max_IIOP_resp_requestsの値128に対して、772を加算した値900が設定される。
  - max_processの値20に対して、29を加算した値49が設定される。
  - max_exec_instanceの値512に対して、448を加算した値960が設定される。
- システム規模がsmallの環境に対し、システム規模をlargeに変更した場合
  - max_IIOP_resp_conの値41に対して、スケール間の加算値の差分+67(100-33)が反映された値108が設定される。
  - max_IIOP_resp_requestsの値900に対して、スケール間の加算値の差分+1148(1920-772)が反映された値2048が設定される。
  - max_processの値49に対して、スケール間の加算値の差分+22(51-29)が反映された値71が設定される。
  - max_exec_instanceの値960に対して、スケール間の加算値の差分+0(448-448)が反映された値960が設定される。
- システム規模がlargeの環境に対し、システム規模をsmallに変更した場合
  - max_IIOP_resp_conの値108に対して、スケール間の加算値の差分-67(33-100)が反映された値41が設定される。
  - max_IIOP_resp_requestsの値2048に対して、スケール間の加算値の差分-1148(772-1920)が反映された値900が設定される。
  - max_processの値71に対して、スケール間の加算値の差分-22(29-51)が反映された値49が設定される。
  - max_exec_instanceの値960に対して、スケール間の加算値の差分-0(448-448)が反映された値960が設定される。

なお、isregistdefコマンドを使用せずに、CORBAサービスの動作環境定義ファイルの値を変更した場合、それ以降に本セットアップを行っても、その変更時の差分の値は有効です。

システムスケールを小さくする場合には、CORBAサービスの動作環境ファイルの定義値が減算されます。CORBAサービスの動作環境ファイルの定義値が、必要量を下回らないように注意してください(セットアップ後、CORBAサービスの動作環境定義ファイルの値を小さくカストマイズしなおしている場合に注意が必要です)。

2.2 チューニング方法(アプリケーション追加によるチューニング)

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を追加する場合に修正する各サービスの定義のステートメントと加算する値を説明します。

以下に示すアプリケーションを追加した場合のチューニング方法について説明します。

- クライアントアプリケーションを追加した場合
- サーバアプリケーションを追加した場合 (注)
- クライアント、サーバ兼用アプリケーションを追加した場合 (注)

注) 本項目は、Interstage Application Server Enterprise Edition、およびStandard-J Editionにおいてチューニングを行う必要があります。

2.2.1 クライアントアプリケーションを追加した場合

CORBAアプリケーション、CORBA/SOAPサーバゲートウェイ

定義名	ステートメント	加算値
CORBAサービスの動作環境ファイル	max_processes (注1)	プロセス数の合計
	max_IIOP_resp_con (注1)(注2)	

注1) Solaris Linux

max_processes、max_IIOP_resp_conを変更した場合は、システムパラメタの設定が必要です。

注2)

SSL接続のコネクションとSSL接続でないコネクションは別コネクションとして数える必要があります。そのため、SSL連携機能を使用する場合、加算値は“プロセス数の合計 × 2”となります。

EJBクライアントアプリケーション

定義名	ステートメント	加算値
CORBAサービスの動作環境ファイル	max_processes (注)	追加するクライアントアプリケーションのプロセス数

注) Solaris Linux

max_processesを変更した場合は、システムパラメタの設定が必要です。

2.2.2 サーバアプリケーションを追加した場合

CORBAアプリケーション、CORBA/SOAPクライアントゲートウェイ

定義名	ステートメント	加算値
CORBAサービスの動作環境ファイル	max_processes (注)	プロセス数の合計
	max_exec_instance	リクエスト実行用スレッドの合計

注) Solaris Linux

max_processesを変更した場合は、システムパラメタの設定が必要です。

EJBアプリケーション

定義名	ステートメント	加算値
CORBAサービスの動作環境ファイル	max_processes (注)	追加するEJBアプリケーションのプロセス数
	max_exec_instance	Windows ・ EJBアプリケーション作成の際に、スレッド動作可としている場合 追加するEJBアプリケーションのプロセス数 × 16 ・ EJBアプリケーション作成の際に、スレッド動作不可としている場合 追加するEJBアプリケーションのプロセス数 Solaris Linux 追加するEJBアプリケーションのプロセス数 × 16

注) Solaris Linux

max_processesを変更した場合は、システムパラメタの設定が必要です。

2.2.3 クライアント、サーバ兼用アプリケーションを追加した場合

サーバアプリケーションから他のオブジェクトを呼び出したり、オブジェクトリファレンスを獲得したり、セション管理機能、XA連携などを使用する場合など、CORBAクライアントとしても動作するアプリケーションを示します。

CORBAアプリケーション

定義名	ステートメント	加算値
CORBAサービスの動作環境ファイル	max_processes (注1)	プロセス数の合計
	max_IIOP_resp_con (注1)(注2)	
	max_exec_instance	リクエスト実行用スレッドの合計

注1) Solaris Linux

max_processes、max_IIOP_resp_conを変更した場合は、システムパラメタの設定が必要です。

注2)

SSL接続のコネクションとSSL接続でないコネクションは別コネクションとして数える必要があります。そのため、SSL連携機能を使用する場合、加算値は“プロセス数の合計 × 2”となります。

2.3 チューニング方法(Interstageの機能を使用するためのチューニング)

Interstageの機能を使用する場合のチューニング方法について説明します。

2.3.1 データベース連携サービス

データベース連携サービスの多重度

データベース連携サービスの多重度を変更する場合は、以下の値を設定または加算します。

定義名	ステートメント	加算、設定値
Interstage動作環境定義	OTS Multiple degree (注1)	データベース連携サービスの多重度
CORBAサービスの動作環境ファイル	max_IIOP_resp_requests (注2)(注3)	
	max_exec_instance (注2)	

注1)

値を設定します。

注2)

値を加算します。

注3)

データベース連携サービスの多重度がmax_IIOP_resp_requestsより大きい場合は、データベース連携サービスの多重度の値を設定します。

リカバリプログラムの多重度のチューニング

リカバリプログラムの多重度をチューニングする場合は、以下の値を設定または加算します。

定義名	ステートメント	加算、設定値
Interstage動作環境定義	OTS Recovery (注1)	リカバリプログラムの多重度
CORBAサービスの動作環境ファイル	max_IIOP_resp_requests (注2)	
	max_exec_instance (注2)	

注1)

値を設定します。

注2)

値を加算します。

リソース管理プログラムのチューニング

リソース管理プログラムを複数起動する場合または、リソース管理プログラムの多重度を変更する場合は以下の値を加算します。

定義名	ステートメント	加算値
CORBAサービスの動作環境ファイル	max_processes (注)	(リソース管理プログラムの多重度 ＋ 1)の合計
	max_IIOP_resp_con (注)	
	max_exec_instance	

注) Solaris Linux

max_processes、max_IIOP_resp_conを変更した場合は、システムパラメタの設定が必要です。

2.3.2 イベントサービス

イベントサービスを使用する場合は、以下の値を設定または加算します。

定義名	ステートメント	加算、設定値
CORBAサービスの動作環境ファイル	max_exec_instance	(注2)
	max_IIOP_local_init_con	以下のいずれかの最大値 ・ max_IIOP_local_init_con ・ 起動するコンシューマ／サプライヤのプロセス数の最大値 ＋ 3 (注3)
	max_IIOP_local_init_requests	以下のいずれかの最大値 ・ max_IIOP_local_init_requests ・ 起動するコンシューマ／サプライヤのプロセス数の最大値 ＋ 3 (注3) × mixモデルのコンシューマ／サプライヤが1コネクションで同時に接続(送信)できるリクエスト数 ・ 起動するコンシューマ／サプライヤのプロセス数の最大値 ＋ 3 (注3) × pushモデルのコンシューマ／pullモデルのサプライヤが1コネクションで同時に接続(受信)できるリクエスト数
	max_IIOP_resp_con (注1)	すべてのイベントチャネルに接続するコンシューマ・サプライヤの合計値 ＋ 1 (注4)
	max_IIOP_resp_requests	以下のいずれかの最大値 ・ max_IIOP_resp_conの加算値 × (mixモデルのコンシューマ／サプライヤが1コネクションで同時に接続(送信)できるリクエスト数 ＋ 1) ・ max_IIOP_resp_conの加算値 × (pushモデルのコンシューマ／pullモデルのサプ

定義名	ステートメント	加算、設定値
		ライヤが1コネクションで同時に接続(受信)できるリクエスト数 ＋ 1)
	max_processes (注1)	起動するイベントチャネル・コンシューマ・サプライヤのプロセス数の合計値 ＋ 2 (注4)
	max_impl_rep_entries	(作成する静的生成イベントチャネルのプロセス数・ 動的生成イベントチャネルのプロセス数 × 2) の合計 (注5)
	period_receive_timeout	異常が発生した場合にコネクションを回収するまでのタイムアウト時間 (注6)

注1) Solaris Linux

max_IIOP_resp_con、およびmax_processesを変更した場合は、システムパラメタを設定してください。

注2)

イベントチャネル側のシステムと、コンシューマ・サプライヤ側のシステムで加算値が異なります。システムにより以下の値を加算してください。

Windows

- イベントチャネル側(イベントチャネルを静的起動した場合)
  「イベントチャネルグループの接続数(esmkchnlコマンドの-mオプションの設定値) (*1)」の総和
  *1)“「イベントチャネルグループの接続数」× 2”の値が“256”よりも小さい場合は、“256”として計算してください。

- イベントチャネル側(イベントファクトリを使用する場合)
  「イベントチャネルのプロセス数(essetupコマンドの-pオプションの設定値)」×「接続数(essetupコマンドの-mオプションの設定値) (*2)」＋ 17
  *2)“「接続数」× 2”の値が“256”よりも小さい場合は、“256”として計算してください。

- コンシューマおよびサプライヤ側
  「サーバアプリケーション数(Pushモデルのコンシューマ数、Pullモデルのサプライヤ数)」×「スレッド最大多重度(OD_impl_instコマンドの-axオプションで指定するthr_conc_maximumの設定値)」

Solaris Linux

- イベントチャネル側(イベントチャネルを静的起動した場合)
  「イベントチャネルグループの接続数(esmkchnlコマンドの-mオプションの設定値) (*3)」の総和
  *3)“「イベントチャネルグループの接続数」＋ 16”の値が“256”よりも小さい場合は、“256”として計算してください。

- イベントチャネル側(イベントファクトリを使用する場合)
  「イベントチャネルのプロセス数(essetupコマンドの-pオプションの設定値)」×「接続数(essetupコマンドの-mオプションの設定値) (*4)」＋ 17
  *4)“「接続数」＋ 16”の値が“256”よりも小さい場合は、“256”として計算してください。

- コンシューマおよびサプライヤ側
  「サーバアプリケーション数(Pushモデルのコンシューマ数、Pullモデルのサプライヤ数)」×「スレッド初期多重度(OD_impl_instコマンドの-axオプションで指定するthr_conc_initの設定値)」

注3)

イベントチャネルを動作させる場合は、さらに“3”を加算してください。

注4)

イベントチャネル通信中にイベントサービス運用コマンドを実行する場合は、1を加算してください。

注5)

静的生成イベントチャネルのプロセス数は、esmkchnlコマンドまたはInterstage管理コンソールで作成した静的生成イベントチャネルグループ数です。
動的生成イベントチャネル(イベントファクトリを使用する場合)のプロセス数は、essetupコマンドの-pオプション、またはInterstageの

初期化コマンド(isinit)実行時にInterstage動作環境定義の“Event maximum Process”で指定したイベントチャネルの最大プロセス数です。

注6)

以下の見積もり式を参考にして見積もった値を加算してください。

```
period_receive_timeout × 5 ＞ イベントデータの待ち合わせ時間(essetcnf／essetcnfchnlコマンドの “-wtime” の設定値) ＋ 20
```

イベントデータの待ち合わせ時間より先にperiod_receive_timeoutによるタイムアウトが発生した場合は、以下の現象が発生する可能性があります。

- イベントデータがロストします。
- エラーメッセージ“od10605”が出力されて、応答の送信が失敗します。
- エラーメッセージ“es10033”(CODE=138)が出力されて、イベントチャネルが異常終了します。

なお、イベントデータの待ち合わせ時間には、“0”を指定しないでください。“0”を指定すると、イベントデータの待ち合わせ時間は無限となり、period_receive_timeoutによるタイムアウトが発生します。

2.4 環境変数について

以下にInterstageで使用する環境変数を示します。

環境変数名	用途
OD_HOME	CORBAサービスのインストールパスを指定します。
	例) OD_HOME=/opt/FJSVod
OD_CODE_SET	コード変換を行う際のクライアントコード系を指定します。 サポートコードは以下のとおりです。 ・ SJIS(ShiftJIS) ・ EUC ・ UNICODE ・ SJISMS(ShiftJIS MS) ・ U90 ・ JEF_LOWER(JEF英小文字) ・ JEF_KANA(JEFカナ文字) ・ JEF_ASCII(JEFASCII) ・ UTF8
	例) OD_CODE_SET=EUC
TD_HOME	コンポーネントトランザクションサービスのインストールパスを指定します。
	例) TD_HOME=/opt/FJSVtd
OTS_HOME	データベース連携サービスのインストールパスを指定します。 OTSが動作するためにOTS_HOMEの設定は必須ではありません。
	例) OTS_HOME=/opt/FJSVots
OTS_ SCROLL_SIZE	otstranlist、otspendlistコマンドを使用して、1画面でトランザクションリストを表示する行数を指定します。
	例) OTS_SCROLL_SIZE=30
	デフォルト:20

環境変数名	用途
OTS_INVOKE_MODE	旧バージョンレベルのINTERSTAGE上のOTSシステム、あるいはリソース管理プログラムと分散トランザクション連携する場合に、OTSシステム、あるいはリソース管理プログラムが動作するマシン上で指定します。値として2以外指定できません。
	例) OTS_INVOKE_MODE=2
ES_HOME	イベントサービスのインストールパスを指定します。
	例) ES_HOME=/opt/FJSVes
PORB_HOME	Portable-ORBのインストールパスを指定します。 (Portable-ORBの動作環境ファイル検索時にも使用)
	例) PORB_HOME=/opt/FJSVporb
IS_ISSTOP_MONITOR_TIMER	Interstage 停止やワークユニット停止が無応答になる現象が発生した場合、自動的にトラブル調査資料を採取する内部機構において、調査資料を採取するまでの監視時間を変更する場合に秒単位で設定します。環境変数を指定しない場合はデフォルト5分の監視時間となります。Interstage 停止やワークユニット停止をコマンドで行う場合には、監視時間はisstopおよびisstopwuコマンドの-t オプションで指定可能です。環境変数の指定より、コマンドのオプション指定が優先されます。本環境変数は必須ではありません。無応答になるトラブルが発生して資料の採取が必要な際に、デフォルトの監視時間を変更したい場合のみ設定してください。 isstartコマンドでInterstageを起動する場合には、コマンドを入力する前に環境変数を設定してください。rcプロシジャからInterstageを起動する場合には、rcプロシジャ内で設定してください。管理コンソールからInterstageを起動する場合には、システムの環境変数として設定してください。

2.5 IPv6環境での運用について

Interstageでは、IPv6環境での運用が可能です。IPv6環境での運用方法を説明します。(注)

注) InterstageはIPv6/IPv4デュアルスタックのみをサポートしています。InterstageはIPv6/IPv4デュアルスタックで利用してください。IPv4を無効にした場合の運用はサポートしておりません。

■運用可能なプラットフォーム

以下の機能は、Windows(R)、Solaris、およびLinuxでIPv6環境での運用が可能です。その他の機能は、Solarisだけで運用可能です。(注)

- CORBAサービス
- データベース連携サービス
- イベントサービス
- Interstage HTTP Server
- Interstage シングル・サインオン
- Interstageディレクトリサービス

注) OSがIPv6に対応している必要があります。Interstageで利用可能なIPv6に対応しているOSは、以下のとおりです。下記以外のOSにIPv6対応用のパッチなどを適用してもInterstageではIPv6をサポートしません。

- Microsoft(R) Windows(R) XP Professional
- Microsoft(R) Windows(R) XP Home Edition
- Microsoft(R) Windows(R) XP Professional x64 Edition
- Microsoft(R) Windows Server(R) 2003, Standard Edition

- Microsoft(R) Windows Server(R) 2003, Enterprise Edition
- Microsoft(R) Windows Server(R) 2003, Standard x64 Edition
- Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition
- Microsoft(R) Windows Server(R) 2003, Enterprise Edition for Itanium-based Systems
- Microsoft(R) Windows Server(R) 2003, Datacenter Edition for Itanium-based Systems
- Microsoft(R) Windows Server(R) 2003 R2, Standard Edition
- Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition
- Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition
- Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition
- Windows Vista(R) Ultimate
- Windows Vista(R) Business
- Windows Vista(R) Home Premium
- Windows Vista(R) Home Basic
- Windows Vista(R) Enterprise
- Microsoft(R) Windows Server(R) 2008 Standard
- Microsoft(R) Windows Server(R) 2008 Enterprise
- Microsoft(R) Windows Server(R) 2008 Datacenter
- Microsoft(R) Windows Server(R) 2008 Standard without Hyper-V
- Microsoft(R) Windows Server(R) 2008 Enterprise without Hyper-V
- Microsoft(R) Windows Server(R) 2008 Datacenter without Hyper-V
- Microsoft(R) Windows Server(R) 2008 for Itanium-based Systems
- Microsoft(R) Windows Server(R) 2008 Foundation
- Microsoft(R) Windows Server(R) 2008 R2 Datacenter
- Microsoft(R) Windows Server(R) 2008 R2 Enterprise
- Microsoft(R) Windows Server(R) 2008 R2 Foundation
- Microsoft(R) Windows Server(R) 2008 R2 Standard
- Windows(R) 7 Ultimate
- Windows(R) 7 Enterprise
- Windows(R) 7 Professional
- Windows(R) 7 Home Premium
- Solaris 9
- Solaris 10
- Red Hat Enterprise Linux AS (v.4 for x86)
- Red Hat Enterprise Linux AS (v.4 for EM64T)
- Red Hat Enterprise Linux AS (v.4 for Itanium)
- Red Hat Enterprise Linux 5 (for x86)
- Red Hat Enterprise Linux 5 (for Intel64)
- Red Hat Enterprise Linux 5 (for Intel Itanium)

- Red Hat Enterprise Linux 6 (for x86)
- Red Hat Enterprise Linux 6 (for Intel64)

■運用可能なサービス

IPv6環境において、Interstageの以下の機能が使用できます。

- CORBAサービス(SSL連携、Proxy連携機能を除く)
  IPv6環境でのCORBAアプリケーション連携(IIOP通信)ができます。
- データベース連携サービス(Interstage Web Serverは除く)
  IPv6環境でデータベース連携サービスを利用することができます。
- イベントサービス(Interstage Web Serverは除く)
  IPv6環境でイベントサービスを利用することができます。
- Interstage HTTP Server
  IPv6環境でのHTTP/HTTPS通信を行うことが可能です。
- Interstage シングル・サインオン
  IPv6環境でInterstage シングル・サインオンを利用することが可能です。
- Interstageディレクトリサービス

■運用方法

InterstageをIPv6環境で運用するには、以下のサービスの環境設定が必要です。その他のサービスでは、特別な設定は不要です。

- pingコマンドなどを実行して対象ホストとの通信が可能であるかの確認をしてください。通信ができない場合は、OSのルーティングの設定の確認をお願いします。設定方法の詳細については、OSのマニュアルおよびヘルプを参照してください。今後、サイトローカルアドレスはOSの機能としてサポートされなくなる可能性があります。そのため、サイトローカルアドレスは使用しないことを推奨します。

Windows

- IPv6環境でリンクローカルアドレスまたはサイトローカルアドレスを用いて通信を行う際には、scope-idを意識する必要があります。

Linux

- Interstage Application Serverでは、リンクローカルアドレスを用いた通信をサポートしません。

CORBAサービスの環境設定

IPv6環境でCORBAアプリケーション連携を行う場合には、config(CORBAサービス)に以下を設定し、CORBAサービスを再起動してください。

IP-version=v4-dual または v6 (デフォルト:v4-dual)

データベース連携サービス

IPv6環境でデータベース連携サービスを利用する場合は、CORBAサービスのIPv6環境を設定する必要があります。CORBAサービスのIPv6環境の設定については、“CORBAサービスの環境設定”を参照してください。

イベントサービス

IPv6環境でイベントサービスを利用する場合は、CORBAサービスのIPv6環境を設定する必要があります。CORBAサービスのIPv6環境の設定については、“CORBAサービスの環境設定”を参照してください。

2.6 ホスト情報(IPアドレス/ホスト名)の変更方法について

Interstageを運用しているサーバのホスト情報(IPアドレスやホスト名)を変更する場合には、1台のサーバで、Interstageの資源移出と資源移入を行うことで実施できます。資源移入時に、変更したいホスト情報を指定して資源移入の操作を行ってください。詳細については、“運用ガイド(基本編)”の“メンテナンス(資源のバックアップ)”を参照してください。

第3章 システムのチューニング

この章では、システムチューニングについて説明します。

3.1 サーバ機能運用時に必要なシステム資源

Interstageの各サービスの運用に必要となるシステム資源について説明します。
以下の表を参照し、使用する製品に応じて、以降に示すサービスのチューニングを行ってください。各機能の説明を参照する前に“■システムパラメタについて”を参照してください。

なお、システムパラメタを算出するためのExcelファイルがマニュアルCDの“ApplicationServer¥tuning”フォルダに“ISAS-IPCtuning.xls”として格納されています。Microsoft(R) Excel 2000もしくは以降のバージョンのMicrosoft(R) Excelをお持ちの場合は“ISAS-IPCtuning.xls”を使用してシステムパラメタを算出することが可能です。使用方法などの詳細については、当該Excelファイル内の説明記事を参照してください。

	EE	SJE	WS
CORBAサービスのシステム環境の設定	○	○	○
コンポーネントトランザクションサービスのシステム環境の設定	○	○	○
データベース連携サービスのシステム環境の設定	○	○	×
イベントサービスのシステム環境の設定 (注1)	○	○	×
IJServerまたはEJBサービスのシステム資源の設定	○	○	×
Interstage HTTP Serverのシステム資源の設定	○	○	○
MessageQueueDirectorのシステム資源の設定	○	×	×
Interstage シングル・サインオンのシステム資源の設定	○	○(注2)	○(注3)
Interstage ディレクトリサービスのシステム資源の設定	○	○(注4)	×
Interstage管理コンソールのシステム資源の設定	○	○	○
Webサーバコネクタのシステム資源の設定	○	○	○(注5)

EE : Interstage Application Server Enterprise Edition
SJE: Interstage Application Server Standard-J Edition
WS: Interstage Web Server

○:チューニングを行う必要があります。
×:チューニングを行う必要はありません(該当製品ではサービスが使用できないため)。

注1) Interstage JMSを使用する場合、イベントサービスのシステム環境を設定する必要があります。
注2) RHEL-AS4(IPF)、およびRHEL5(IPF)の場合、Interstage シングル・サインオンの認証サーバ、および業務サーバのシステム資源だけ設定します。
注3) Interstage シングル・サインオンの業務サーバのシステム資源だけ設定します。
注4) Interstage ディレクトリサービスは、RHEL-AS4(IPF)、およびRHEL5(IPF)では提供されていません。
注5) Webサーバコネクタの故障監視機能のシステム資源を設定する必要はありません。

■システムパラメタについて

◆システムパラメタの変更方法 Solaris

IPC資源のパラメタに値を設定するために、以下のいずれかの方法を選択してください。

- **/etc/systemファイルの修正**

  /etc/systemファイルを編集し、必要なパラメタ値を変更します。変更後は、変更した値を反映するためにシステムをリブートしてください。なお、変更方法の詳細については、Solarisのドキュメントを参照してください。

注) Solaris 10において、この方法でシステムチューニングする場合、shmmaxおよびshmmniは以下の関係が成り立つような値を設定してください。

```
project.max-shm-memory ＝ shmmax × shmmni
```

- **資源制御(Solaris 10のみ)**

  以下の手順で、パラメタ値を変更してください。

  1. Interstageの停止
     Interstageを停止してください。もしInterstage管理コンソールを使用するためのサービスを起動している場合はそれらのサービスも停止してください。

  2. user.rootプロジェクトとsystemプロジェクトのパラメタの変更
     projmodコマンドでuser.rootプロジェクトとsystemプロジェクトのパラメタの値を変更してください。以下に例を示します。

     ```
     projmod -s -K 'project.max-sem-ids=(privileged,155,deny)' user.root
     projmod -s -K 'project.max-sem-ids=(privileged,155,deny)' system
     ```

     変更の特権レベルをprivileged、設定したしきい値を超える要求があった場合のアクションをdenyとします。

  3. 値の反映
     newtaskコマンドで変更した値をシステム反映します。

     ```
     newtask -p user.root -c $$
     ```

  4. Interstageの起動
     Interstageを起動してください。必要に応じて、Interstage管理コンソールを使用するためのサービスを起動してください。

  資源制御の詳細については、Solarisのドキュメントを参照してください。

「種類」の意味

システムパラメタの表中の「種類」の意味は以下のとおりです。

- 設定値
  必要数の条件に応じた値に変更してください。
- 加算値
  すでに設定されている値に、必要数を加算してください。

各パラメタの意味

システムパラメタの各パラメタ名と意味は以下のとおりです。
なお、資源制御によるIPC資源のパラメタの設定は、Solaris 10の場合のみ可能です。

共用メモリ Solaris

パラメタ	資源制御	意味
－	project.max-shm-memory	共用メモリの最大サイズ
shmmax	－	共用メモリの最大セグメントサイズ
shmmni	project.max-shm-ids	共用メモリIDの最大数

セマフォ Solaris

パラメタ	資源制御	意味
semmni	project.max-sem-ids	セマフォIDの最大数
semmns	－	システム全体のセマフォの最大数

パラメタ	資源制御	意味
semvmx	－	セマフォに設定できる最大数
semmsl	process.max-sem-nsems	セマフォIDあたりのセマフォの最大数
semopm	process.max-sem-ops	セマフォコールあたりのセマフォ操作の最大数
semmnu	－	システム全体のセマフォ操作の取消記録グループ数
semume	－	プロセスあたりのセマフォ操作の取消記録の最大数

メッセージキュー Solaris

パラメタ	資源制御	意味
msgmax		メッセージの最大サイズ
msgmnb	process.max-msg-qbytes	メッセージキュー上のメッセージの最大バイト数
msgmni	project.max-msg-ids	メッセージキューIDの最大数
msgtql	process.max-msg-messages	メッセージキューにあるメッセージの最大数

ファイルディスクリプタ Solaris

パラメタ	資源制御	意味
rlim_fd_max	process.max-file-descriptor	最大ファイルディスクリプタ数

◆システムパラメタの変更方法 Linux

/etc/sysctl.confを編集し、パラメタ値を変更します。変更後は“sysctl -p /etc/sysctl.conf”を実行するか、システムをリブートしてください。
変更方法の詳細については、OSのドキュメントを参照してください。

「種類」の意味

システムパラメタの表中の「種類」の意味は以下のとおりです。

- 設定値
  必要数の条件に応じた値に変更してください。
- 加算値
  すでに設定されている値に、必要数を加算してください。

各パラメタの意味

システムパラメタの各パラメタ名と意味は以下のとおりです。
なお、資源制御によるIPC資源のパラメタの設定は、Solaris 10の場合のみ可能です。

共用メモリ Linux

パラメタ	意味
kernel.shmall	システム全体の共用メモリの最大サイズ
kernel.shmmax	共用メモリの最大サイズ
kernel.shmmni	共用メモリセグメントの最大数

セマフォ Linux

セマフォの設定値は、各パラメタ値を以下の形式で指定します。

— kernel.sem = *para1 para2 para3 para4*

パラメタ	意味
para1	セマフォIDあたりのセマフォの最大数
para2	システム全体のセマフォの最大数
para3	セマフォコールあたりのセマフォ操作の最大数
para4	セマフォIDの最大数

メッセージキュー Linux

パラメタ	意味
kernel.msgmax	メッセージの最大サイズ
kernel.msgmnb	メッセージキュー上のメッセージの最大バイト数
kernel.msgmni	メッセージキューIDの最大数

◆リソース制限 Linux

システムのリソースを制限するには、/etc/security/limits.confファイルを編集し、必要なパラメタ値を変更します。変更後は、変更した値を反映するためにシステムをリブートしてください。
変更方法の詳細については、OSのドキュメントを参照してください。

「種類」の意味

リソース制限の表中の「種類」の意味は以下のとおりです。

- soft
  ソフトウェアリミット値を変更してください。
- hard
  ハードウェアリミット値を変更してください。

各パラメタの意味

リソース制限の各パラメタ名と意味は以下のとおりです。

ファイルディスクリプタ Linux

パラメタ	意味
nofile	最大ファイルディスクリプタ数

3.1.1 CORBAサービスのシステム環境の設定

CORBAサービスを用いたシステムの運用時には、接続するクライアント/サーバ数、オブジェクト数などによりシステム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ
  - CORBAサービス
  - インタフェースリポジトリ
  - ネーミングサービス
- プロセス・スレッド

・ ファイルディスクリプタ

■システムパラメタ

一般的なCORBAサービスが使用する共用メモリ、セマフォ、メッセージキューのシステムパラメタのチューニングについて説明します。
CORBAサービスの他に共用メモリ、セマフォ、メッセージキューを使用するアプリケーションが存在する場合、そのアプリケーションが使用する資源にCORBAサービスの資源量を加算してください。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

CORBAサービス

CORBAサービスで必要となるシステム資源について、以下に示します。

各表に記述されているパラメタ名(max_IIOP_resp_conなど)は、CORBAサービスのconfigファイルで指定します。詳細については、“A.1 config”を参照してください。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	以下の値のうち、最大値を指定します。 ・ max_IIOP_resp_con × 16K ＋ (max_IIOP_resp_con_extend_number(注1) ＋ 1) × 0.2K ＋ max_IIOP_resp_requests × 16K ＋ (max_IIOP_resp_requests_extend_number(注1) ＋ 1) × 0.2K ＋ max_impl_rep_entries × 6K ＋ max_bind_instances(注5) × 0.1K ＋ 100K 以上 ― [trace_use=yesの場合] 上記値 ＋ max_processes × trace_size_per_process ＋ trace_size_of_daemon(注2) ＋ 20K 以上 ― [snap_use=yesの場合] 上記値 ＋ snap_size ＋ 10K 以上 ・ number_of_common_buffer(注3) × 4K 以上 ＋ (number_of_common_buffer_extend_number(注1) ＋ 1) × 0.2K ・ [Buffer Size、Buffer Number(ワークユニット定義)を指定したCORBAワークユニット、IJServer起動時] (Buffer Size ＋ 0.2K) × Buffer Number 以上 (注4)
kernel.shmmni	加算値	max_IIOP_resp_con_extend_number(注1) ＋ max_IIOP_resp_requests_extend_number(注1) ＋ number_of_common_buffer_extend_number(注1) ＋ “Buffer Size、Buffer Number(ワークユニット定義)を指定したアプリケーション数” ＋ 14

注1)

[パラメタ名]_extend_numberのデフォルト値は以下となります。0が指定された場合も、以下と同様になります。limit_of_[パラメタ名]は、0が指定された場合は自動計算されます。計算式の詳細については、“config”を参照してください。
(limit_of_[パラメタ名] ― [パラメタ名]) ÷ [パラメタ名] (小数部分切り上げ)
isconfig.xmlファイルの定義項目AutoConfigurationModeにMANUALを指定し、自動拡張を行わない設定にした場合は、0となります。

注2)

デフォルトは以下です。0が指定された場合も、以下と同様になります。
trace_size_per_process × 32

注3)

デフォルトは以下です。0が指定された場合も、以下と同様になります。
max_IIOP_resp_requests × 0.2

注4)

Buffer Size、Buffer Numberを指定したワークユニット定義の中で、“(Buffer Size ＋ 0.2KB) × Buffer Number”の最大値が該当します。なお、“(Buffer Size ＋ 0.2KB) × Buffer Number”の最大値が 2,147,483,647より小さい値になるようにBuffer Size、Buffer Numberの値を設定してください。

注5)

デフォルトは以下です。0が指定された場合も、以下と同様になります。
max_processes × 1024 (計算結果が65,535を超えた場合は65,535)

セマフォ

パラメタ	種類	必要数
para1	設定値	以下の値のうちの最大値以上の値を指定します。 ・ max_IIOP_resp_con ・ max_processes
para2	加算値	limit_of_max_IIOP_resp_con(注1) × 4 ＋ max_IIOP_resp_con_extend_number(注2) ＋ max_IIOP_resp_requests_extend_number(注2) ＋ max_impl_rep_entries ＋ max_processes × 4 ＋ プロセスモードのCORBAサーバアプリケーションの起動プロセス数(注3) ＋ “Buffer Size、Buffer Number(ワークユニット定義)を指定したアプリケーション数” × 2 ＋ 14 以上
		[トレース機能を使用する場合] 上記値 ＋ 1 以上
		[スナップショット機能を使用する場合] 上記値 ＋ 1 以上
		[SSL連携機能を使用する場合] 上記値 ＋ limit_of_max_IIOP_resp_con(注1) 以上
para3	設定値	50 以上
para4	加算値	以下の値のうち、最大値を指定します。 ・ 512 ・ max_IIOP_resp_con_extend_number(注2) × 5 ＋ max_IIOP_resp_requests_extend_number(注2) ＋ max_impl_rep_entries ＋ プロセスモードのCORBAサーバアプリケーションの起動プロセス数(注3) ＋ “Buffer Size、Buffer Number(ワークユニット定義)を指定したアプリケーション数” × 2 ＋ 100 以上

注1)

limit_of_[パラメタ名]のデフォルト値は以下となります。0が指定された場合も、以下と同様になります。
[パラメタ名] × 1.3 (小数部分切り捨て)

isconfig.xmlファイルの定義項目AutoConfigurationModeにMANUALを指定し、自動拡張を行わない設定にした場合は、[パラメタ名]となります。

注2)

[パラメタ名]_extend_numberのデフォルト値は以下となります。0が指定された場合も、以下と同様になります。
(limit_of_[パラメタ名] － [パラメタ名]) ÷ [パラメタ名] (小数部分切り上げ)
isconfig.xmlファイルの定義項目AutoConfigurationModeにMANUALを指定し、自動拡張を行わない設定にした場合は、0となります。

注3)

起動プロセス数が分からない場合はmax_processesを指定してください。

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	16,384 以上
kernel.msgmnb	設定値	32,768 以上
kernel.msgmni	加算値	512 以上

インタフェースリポジトリ

インタフェースリポジトリを使用する場合に必要となるシステム資源を以下に示します。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	[ログ採取時] logging memory size ＋ 16K (注)

注)

“logging memory size”は、CORBAサービスのirconfigファイルで指定します。詳細については、“A.5 irconfig”を参照してください。

ネーミングサービス

ネーミングサービスにネーミングコンテキストを多数作成する場合に必要となるシステム資源を、以下に示します。

パラメタ	種類	必要数
(注)	加算値	ネーミングコンテキスト数 ＋ 16 以上

注)

プロセス数あたりのオープン可能なファイル数です。該当するパラメタはありません。
bash(Linux)または、ボーンシェルの場合はulimitコマンドを、Cシェルの場合はlimitコマンドを使用して、ネーミングサービスのプロセスが必要とするファイルをオープンできるだけの値を設定してください。コマンドの詳細については、OSのドキュメントを参照してください。

■アプリケーションで使用するスレッド数・プロセス数

CORBAサービスでアプリケーションを実行する場合、アプリケーションから生成されるプロセス数・スレッド数が多くなる場合には、システムパラメタを変更する必要があります。
アプリケーションをマルチスレッドで作成している場合に、生成されるスレッド数の目安を以下に示します。

分類	スレッド数
CORBAサービス	25個 ＋ クライアントアプリケーションとの接続数 ＋ CORBAアプリケーションプロセス数
サーバアプリケーション	1プロセスにつき (6個 ＋ スレッド多重度数)
クライアントアプリケーション	1プロセスにつき8個

システムパラメタで、変更が必要となるものを以下に示します。

パラメタ	内容
kernel.threads-max	システム全体で生成できるプロセス数とスレッド数の合計

システムパラメタ以外で、考慮するパラメタを以下に示します。

パラメタ	内容
(注1)	プロセスの最大スタックサイズ
(注2)	1人のユーザが使用できる最大のプロセス数

注1)

該当するパラメタはありません。
bashまたはボーンシェルの場合はulimitコマンドを、Cシェルの場合はlimitコマンドを使用して設定してください。この値にスレッド数を掛けた値が、プロセスのスタック領域として使用されます。1つのプロセスで使用可能なメモリを超過してスレッドは生成できないため、1つのプロセスが生成できるスレッド数は限界があります。
CORBAサーバアプリケーションおよびEJBアプリケーションのリクエスト処理多重度は“スレッド多重度 × プロセス多重度”で計算されます。1つのプロセスで使用可能なメモリサイズによってスレッド多重度を上げることができない場合は、プロセス多重度を上げることを検討してください。CORBAサーバアプリケーションのスレッド多重度・プロセス多重度については“リファレンスマニュアル(コマンド編)”の“OD_impl_inst”に記載されているproc_conc_max、thr_conc_init、thr_conc_maximumを参照してください。EJBアプリケーションのスレッド多重度については“EJBコンテナのチューニング”に記載されている“5.4.1 同時処理数”を参照してください。

注2)

該当するパラメタはありません。
bashまたはボーンシェルの場合はulimitコマンドを、Cシェルの場合はlimitコマンドを使用して設定してください。ユーザが生成するプロセス数とスレッド数の合計値以上の値に設定してください。

■ファイルディスクリプタ数

CORBAサービスで多数のアプリケーションを動作させ(多端末接続時など)、使用するファイルディスクリプタ数がシステムのデフォルト値を超える場合は、システムパラメタにその値を設定してください。

システムパラメタで、変更が必要となるものを以下に示します。

パラメタ	内容
fs.file-max	システム全体のファイルディスクリプタの上限値。

システムパラメタ以外で、考慮するパラメタを以下に示します。

パラメタ	内容
nofile (注1)	各ユーザのオープン可能なファイルディスクリプタの上限値

注1)

/etc/security/limits.conf ファイルを編集します。limits.conf ファイルの詳細については、OSのドキュメントを参照してください。

3.1.2 コンポーネントトランザクションサービスのシステム環境の設定

コンポーネントトランザクションサービスの動作時には、使用する機能によりシステム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ
  - コンポーネントトランザクションサービスの基本機能
  - 性能監視ツール

以降に示す値は、CORBAサービスの値を含んでいません。“3.1.1 CORBAサービスのシステム環境の設定”を参照し、必要な値を加算してください。

■システムパラメタ

コンポーネントトランザクションサービスが使用する共用メモリ、セマフォ、メッセージキューのシステムパラメタのチューニングについて説明します。

コンポーネントトランザクションサービスの基本機能のほかに各機能を使用する場合は、コンポーネントトランザクションサービスの基本機能の資源に各機能で使用する資源量を加算してください。

システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

コンポーネントトランザクションサービスの基本機能

コンポーネントトランザクションサービスの基本機能を使用する場合に必要となるシステム資源について、以下に示します。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	18,219,144 以上
kernel.shmmni	加算値	22

セマフォ

パラメタ	種類	必要数
para1	設定値	12 以上
para2	加算値	21
para3	設定値	3 以上
para4	加算値	29

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	528 以上
kernel.msgmnb	設定値	4,572 ＋ (528 × 同時実行コマンド数) (注)
kernel.msgmni	加算値	11

注)

同時実行コマンド数とは、以下のコマンドを同時に実行した数のことです。

－ isstartwu、isstopwu、tdstartwu、tdstopwu、tdinhibitobj、tdpermitobj、tdmodifyprocnum、tdmodifywu

また、Systemwalker Operation Managerを使用してワークユニットの起動/停止、オブジェクト閉塞/閉塞解除を行う場合は、同時に操作する回数が同時実行コマンド数となります。

性能監視ツール

性能監視ツールを使用する場合に追加となるシステム資源については、“3.3 性能監視ツール使用時に必要なシステム資源”を参照してください。

3.1.3 データベース連携サービスのシステム環境の設定

データベース連携サービスの動作時には、利用するシステム形態によりシステム資源を拡張する必要があります。ここでは、以下について、利用するシステム形態ごとに説明します。

- システムパラメタ
  - － OTSシステムのみが動作する場合
  - － リソース管理プログラムのみが動作する場合
  - － OTSシステムとリソース管理プログラムの両方が動作する場合

■システムパラメタ

データベース連携サービスが使用する共用メモリ、セマフォ、メッセージキューのシステムパラメタのチューニングについて、システム形態ごとに説明します。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

OTSシステムのみが動作する場合

OTSシステムのみが動作する場合に必要となるシステム資源について、以下に示します。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	17,830,204 (注)
kernel.shmmni	加算値	12

注)

以下のデフォルト環境で算出した値です。

－ データベース連携サービスの環境定義のconfigファイル
OTS_TRACE_SIZE = 4096
RECOVERY_TRACE_SIZE = 4096
OBSERVE_TRACE_SIZE = 4096

－ データベース連携サービスの環境定義のセットアップ情報ファイル
TRANMAX = 100
PARTICIPATE = 4

定義値を変更している場合、以下の計算式で求めてください。
必要数 ＝
(OTS_TRACE_SIZE ＋ RECOVERY_TRACE_SIZE ＋ OBSERVE_TRACE_SIZE) ×1,024 ＋
PARTICIPATE × TRANMAX × 2,560 ＋ TRANMAX × 284 ＋ 4,399,692

セマフォ

パラメタ	種類	必要数
para1	設定値	12 以上
para2	加算値	24
para3	設定値	3 以上
para4	加算値	8

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	528 以上
kernel.msgmnb	設定値	4,572 以上
kernel.msgmni	加算値	3

リソース管理プログラムのみが動作する場合

リソース管理プログラムのみが動作する場合に必要となるシステム資源について、以下に示します。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	12,840,416 (注)
kernel.shmmni	加算値	リソース管理プログラムの種類 × 11

注)

リソース管理プログラムの種類が1つの場合で、かつ、以下のデフォルト環境で算出した値です。

- データベース連携サービスの環境定義のconfigファイル
  RESOURCE_TRANMAX = 10
  RESOURCE_TRACE_SIZE = 4096
  OBSERVE_TRACE_SIZE = 4096
- リソース定義ファイル
  OTS_RMP_PROC_CONC = 5
- データベース連携サービスの環境定義のセットアップ情報ファイル
  TRANMAX = 100

定義値を変更している場合、以下の計算式で求めてください。
必要数 ＝
(RESOURCE_TRACE_SIZE ＋ OBSERVE_TRACE_SIZE) × 1,024 ＋
(TRANMAX ＋ 1) × 332 ＋
((リソース管理プログラムの種類 × RESOURCE_TRANMAX ×
OTS_RMP_PROC_CONC) × (144 ＋ 332)) ＋ 4,394,476

セマフォ

パラメタ	種類	必要数
para4	加算値	リソース管理プログラムの種類 × 7

OTSシステムとリソース管理プログラムの両方が動作する場合

OTSシステムとリソース管理プログラムの両方が動作する場合に必要となるシステム資源について、以下に示します。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	17,830,204 (注)
kernel.shmmni	加算値	12 ＋ リソース管理プログラムの種類 × 11

注)

リソース管理プログラムの種類が1つの場合で、かつデフォルト環境での値です。定義値を変更している場合、以下の計算式で求めてください。
必要数 ＝
OTSシステムのみが動作する場合の必要数 ＋
リソース管理プログラムのみが動作する場合の必要数 － 4,915,600

セマフォ

パラメタ	種類	必要数
para1	設定値	12 以上
para2	加算値	24
para3	設定値	3 以上
para4	加算値	8 ＋ リソース管理プログラムの種類 × 7

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	528 以上
kernel.msgmnb	設定値	4,572 以上
kernel.msgmni	加算値	3

3.1.4 イベントサービスのシステム環境の設定

イベントサービスを用いたシステムの運用時には、チャネル数、接続するコンシューマ/サプライヤ数などによりシステム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ

以降に示す値は、CORBAサービスの値を含んでいません。“3.1.1 CORBAサービスのシステム環境の設定”を参照し、必要な値を加算してください。

■システムパラメタ

一般的なイベントサービスが使用する共用メモリ、セマフォ、メッセージキューのシステムパラメタのチューニングについて説明します。
イベントサービスの他に共用メモリ、セマフォ、メッセージキューを使用するアプリケーションが存在する場合、そのアプリケーションが使用する資源にイベントサービスの資源量を加算してください。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	[揮発チャネル運用の場合] 2,064 × イベントチャネル最大作成数(注1) ＋ 600K ＋trace_sizeパラメタに指定したバイト数(注2) ＋ 3,328
		[不揮発チャネル運用の場合] 以下の値のうち、最大値を指定します。 ・ 2,064 × イベントチャネル最大作成数(注1) ＋ 184 × 同時実行可能なグローバルトランザクション数(注3) ＋ 600K ＋trace_sizeパラメタに指定したバイト数(注2) ＋ 3,328 ・ 134,217,728 以上 ・ ユニットで使用する共用メモリサイズ(複数ユニットがある場合、最大値)(注4)
kernel.shmmni	加算値	[揮発チャネル運用の場合] ・ プロセス単位で内部トレースを採取する(trace_buffer=process)場合(注2) 4 ＋ イベントチャネルのプロセス数(注5) ・ イベントサービス単位で内部トレースを採取する(trace_buffer=system)場合(注2) 4
		[不揮発チャネル運用の場合] ・ プロセス単位で内部トレースを採取する(trace_buffer=process)場合(注2) 100 以上の値 (ユニット単位に加算) ＋ イベントチャネルのプロセス数(注5) ・ イベントサービス単位で内部トレースを採取する(trace_buffer=system)場合(注2) 100 以上の値 (ユニット単位に加算)

注1)

イベントチャネル最大作成数 ＝
静的生成イベントチャネル最大作成数 ＋ 動的生成イベントチャネル最大作成数

注2)

trace_sizeパラメタおよびtrace_bufferパラメタは、イベントサービスの動作環境ファイル（traceconfig）で指定します。詳細については、“D.1 traceconfig”を参照してください。

注3)

同時実行可能なグローバルトランザクション数は、イベントサービスの構成情報管理コマンド(essetcnf)による-gtrnmaxオプションの設定値です。

注4)

ユニットで使用する共用メモリサイズは、ユニットの作成コマンド(esmkunit)のユニット定義ファイルで指定した項目shmmaxの設定値です。

注5)

イベントチャネルのプロセス数 ＝
静的イベントチャネルグループ数 ＋ 動的イベントチャネルのプロセス数

－ 動的イベントチャネルのプロセス数:
イベントサービスのセットアップコマンド(essetup)による-pオプションの設定値。
ノーティフィケーションサービスを使用している場合は、“動的イベントチャネルのプロセス数 × 2”としてください。

セマフォ

パラメタ	種類	必要数
para1	設定値	29
para2	加算値	[揮発チャネル運用の場合] 6 以上
		[不揮発チャネル運用の場合] ユニット数 × 29 ＋ 13 以上
para3	設定値	29
para4	加算値	ユニット数 × 256

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	[不揮発チャネル運用の場合] 2,048 以上
kernel.msgmnb	設定値	[不揮発チャネル運用の場合] 4,096 以上
kernel.msgmni	加算値	[不揮発チャネル運用の場合] ユニット数 × 9

3.1.5 IJServerまたはEJBサービスのシステム資源の設定

IJServerまたはEJBサービスでは以下の機能を使用する場合に、システム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ

■システムパラメタ

IJServerまたはEJBサービスを使用する際には、以下のシステムパラメタのチューニングを行ってください。
ワークユニット定義の通信バッファ数(Buffer Number)、通信バッファ長(BufferSize)を指定する場合は、“3.1.1 CORBAサービスのシステム環境の設定”についても参照してください。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	4,096 以上
kernel.msgmnb	設定値	4,096 以上
kernel.msgmni	加算値	2 以上

3.1.6 Interstage HTTP Serverのシステム資源の設定

Interstage HTTP Serverを用いたシステムの運用時には、システム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ

■システムパラメタ

Interstage HTTP Serverを運用するためのサービスが使用するセマフォのシステムパラメタのチューニングについて説明します。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

セマフォ

パラメタ	種類	必要数
para1	設定値	1以上
para2	加算値	Webサーバ数×2
para3	設定値	1以上
para4	加算値	Webサーバ数×2

3.1.7 MessageQueueDirectorのシステム資源の設定

MessageQueueDirectorを用いたシステムの運用時には、MQDのシステム数およびMQDの上位サービスの種類などによりシステム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ

■システムパラメタ

MessageQueueDirectorが使用するシステムパラメタのチューニングについて説明します。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	134,217,728 以上
kernel.shmmni	加算値	100 × MQDシステム数

セマフォ

パラメタ	種類	必要数
para1	設定値	29
para2	加算値	MQDシステム数 × 29
para3	設定値	29
para4	加算値	MQDシステム数 × 256

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	16,384 以上
kernel.msgmnb	設定値	32,768 以上
kernel.msgmni	加算値	9 × MQDシステム数

3.1.8 Interstage シングル・サインオンのシステム資源の設定

Interstage シングル・サインオンを用いたシステムの運用時には、システム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ
  - Interstage シングル・サインオンのリポジトリサーバ機能を使用する際のシステムパラメタのチューニング
  - Interstage シングル・サインオンの認証サーバ機能を使用する際のシステムパラメタのチューニング
  - Interstage シングル・サインオンの業務サーバ機能を使用する際のシステムパラメタのチューニング

■システムパラメタ

Interstage シングル・サインオンが使用するシステムパラメタのチューニングについて説明します。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

なお、以降に示す値は、Interstage HTTP Serverで必要な値を含んでいません。“Interstage HTTP Serverのシステム資源の設定”を参照し、必要な値を加算してください。

Interstage シングル・サインオンのリポジトリサーバ機能を使用する際のシステムパラメタのチューニング

リポジトリサーバでは、“Interstage ディレクトリサービス”を使用しています。“Interstage ディレクトリサービス”で必要としているチューニングについては、“3.1.9 Interstage ディレクトリサービスのシステム資源の設定”を参照してください。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	12,800,000 ＋ (164 × サイト定義数(注1)) ＋ (ロール数(注2) ＋ ロールセット数(注3) ＋ ロールセット数(注3) × ロール数(注2)) × 1,024 以上(注4)
kernel.shmmni	加算値	13

注1)

SSOリポジトリの保護リソースに登録したサイト定義の総数

注2)

SSOリポジトリに定義したロールの総数

注3)

SSOリポジトリに定義したロールセットの総数

注4)

ユーザ情報を登録するディレクトリサービスにActive Directoryを使用し、シングル・サインオンの拡張スキーマを使用しない場合は、運用に応じて以下の式で見積もった値を加算してください。
Active Directoryのロール/ロールセットに使用する属性の総数 × 524

セマフォ

パラメタ	種類	必要数
para2	加算値	9
para4	加算値	9

Interstage シングル・サインオンの認証サーバ機能を使用する際のシステムパラメタのチューニング

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	14,000,000 ＋ パス定義の総数(注) × 2,048 以上
kernel.shmmni	加算値	11

注)

SSOリポジトリの保護リソースに登録した各サイト定義に定義したパス定義の総数

セマフォ

パラメタ	種類	必要数
para2	加算値	10
para4	加算値	10

Interstage シングル・サインオンの業務サーバ機能を使用する際のシステムパラメタのチューニング

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	13,000,000 ＋ (1 + キャッシュサイズ(注1)) × 1024 × キャッシュ数(注2) + (ロール数(注3) ＋ 拡張ユーザ情報数(注4)＋ 1) × パス定義の最大数(注5) × 1,200
kernel.shmmni	加算値	10 × 業務サーバ数

注1)

セションの管理を行う運用の場合、業務サーバでキャッシュする利用者の認証情報サイズ(Kバイト)を設定します。セションの管理を行わない運用の場合は、0で計算してください。
キャッシュサイズについては、“F.4 業務サーバを構築する場合のチューニング”を参照してください。

注2)

セションの管理を行う運用の場合、システムの同時アクセス最大数以上を設定します。セションの管理を行わない運用の場合は、0で計算してください。
キャッシュ数については、“F.4 業務サーバを構築する場合のチューニング”を参照してください。

注3)

SSOリポジトリに定義したロールの総数

注4)

業務アプリケーションに対して通知するユーザ情報の数
セションの管理を行う運用の場合、リポジトリサーバのInterstage管理コンソールの以下に設定されている属性名の数を設定します。
セションの管理を行わない運用の場合は、0で計算してください。
[システム] > [セキュリティ] > [シングル・サインオン] > [認証基盤] > [リポジトリサーバ] > [環境設定] > [リポジトリサーバ詳細設定[表示]] > [業務システムに通知する情報] > [拡張ユーザ情報]

注5)

SSOリポジトリの保護リソースに登録した各サイト定義の保護パスに設定されているパス定義の最大数

セマフォ

パラメタ	種類	必要数
para2	加算値	7 × 業務サーバ数
para4	加算値	7 × 業務サーバ数

3.1.9 Interstage ディレクトリサービスのシステム資源の設定

Interstage ディレクトリサービスを用いたシステムの運用時には、システム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ(Interstage ディレクトリサービスの運用に必要なシステム資源)
- システムパラメタ(標準データベースの運用に必要なシステム資源)
- システムパラメタ(Symfoware Serverの運用に必要なシステム資源)
- システムパラメタ(Oracleデータベースの運用に必要なシステム資源)

■システムパラメタ(Interstage ディレクトリサービスの運用に必要なシステム資源)

Interstage ディレクトリサービスが使用するシステムパラメタのチューニングについて説明します。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	5 × (リポジトリ数 × 1,843,200) 以上
kernel.shmmni	加算値	4 × リポジトリ数(注)

注)

リポジトリのデータベースとして標準データベースを使用し、レプリケーション運用をする時は、5 × リポジトリ数です。

リポジトリのデータベース運用に必要なシステム資源のチューニング

リポジトリのデータベースに標準データベースを使用する場合には、さらにシステム資源のチューニングが必要です。“システムパラメタ(標準データベースの運用に必要なシステム資源)”を参照して、システムパラメタを変更してください。

リポジトリのデータベースとしてRDBを使用する場合には、さらに、RDBの運用に必要なシステム資源をチューニングしてください。以下の項目を参照して、システムパラメタを変更してください。

- “システムパラメタ(Symfoware Serverの運用に必要なシステム資源)”
- “システムパラメタ(Oracleデータベースの運用に必要なシステム資源)”

■システムパラメタ(標準データベースの運用に必要なシステム資源)

標準データベースを運用するには、omsアカウント環境の最大ファイルディスクリプタ数のハードリミットが、256以上65,536以下となるように設定してください。

最大ファイルディスクリプタ数のハードリミットを変更するには、リソース制限を変更してください。システムパラメタの変更方法については、“◆リソース制限 ”を参照してください。

ファイルディスクリプタ

パラメタ	種類	必要数
nofile	hard	256以上、65,536以下

システム全体の最大ファイルディスクリプタ数のハードリミットを、65,536より大きな値に設定したい場合、omsユーザの設定を個別に行ってください。

例

```
*    hard    nofile    524288
oms  hard    nofile    65536
```

■システムパラメタ(Symfoware Serverの運用に必要なシステム資源)

Symfoware Serverの運用に必要なシステムパラメタの設定は、Symfoware Serverをインストールしたマシンで変更してください。

- Symfoware ServerをSolarisにインストールした場合
- Symfoware ServerをLinuxにインストールした場合

ここで示す各システムパラメタの必要数の値は、Symfoware Serverのシステム用動作環境ファイル、またはRDB構成パラメタファイルのパラメタに以下の値が設定されていることを前提としています。これらのパラメタの設定値を変更する場合は、Symfoware Serverのマニュアルを参照して各システムパラメタの必要数を算出し直してください。

パラメタ	設定値
ローカル接続で使用するメモリ量(COMMUNICATION_BUFFER)	32Kバイト
ローカル接続数(MAX_CONNECT_SYS) (注1)	256
デーモン多重度(RDBCNTNUM)	712
共有メモリ量(RDBEXTMEM)	13,208Kバイト

注1)

ローカル接続数は、Interstage ディレクトリサービスの使用時に必要となる、リポジトリからRDBへの最大コネクション数の合計に、他のアプリケーション等が使用するコネクション数を加えた値を算出してください。求めた値が、この設定値(256)を超える場合には、各システムパラメタの必要数を算出し直してください。

リポジトリの最大コネクション数の詳細は、“ディレクトリサービス運用ガイド”の“データベース共用”-“最大コネクション数の設定”を参照してください。

レプリケーション運用をするときは、Linkexpressの運用に必要なシステム資源の設定をしてください。設定内容は、Linkexpressのインストールガイドを参照してください。

Solarisの場合は、「Linkexpressへの同時転送依頼数」を「1」、「Linkexpress同時ファイル転送多重度」を「4」として計算してください。また、システムパラメタ「shmmin」(共用メモリセグメントの最小サイズ)は設定しないでください。

Symfoware ServerをSolarisにインストールした場合

共用メモリ

パラメタ (注)	資源制御	種類	必要数
shmmax	project.max-shm-memory	設定値	13,524,992 以上
shmmni	project.max-shm-ids	加算値	10

注)

パラメタ欄には、“shmsys:shminfo_xxxxxx”の“xxxxxx”を記載しています。

セマフォ

パラメタ (注1)	資源制御	種類	必要数
semmni	project.max-sem-ids	加算値	300
semmns (注2)	―	加算値	1,112

パラメタ (注1)	資源制御	種類	必要数
semmnu (注2)	－	加算値	64
semmsl	process.max-sem-nsems	設定値	48 以上

注1)

パラメタ欄には、“semsys:seminfo_xxxxxx”の“xxxxxx”を記載しています。

注2)

Solaris 9でのみ有効です。

メッセージキュー

パラメタ (注1)	資源制御	種類	必要数
msgmax (注2)	－	設定値	128 以上
msgmnb	process.max-msg-qbytes	設定値	4,096 以上
msgmni	project.max-msg-ids	加算値	2
msgtql	process.max-msg-messages	加算値	64

注1)

パラメタ欄には、“msgsys:msginfo_xxxxxx”の“xxxxxx”を記載しています。

注2)

Solaris 9でのみ有効です。

Symfoware ServerをLinuxにインストールした場合

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	13,524,992 以上
kernel.shmmni	加算値	10

セマフォ

パラメタ	種類	必要数
para1	設定値	48 以上
para2	加算値	1,112
para3		すでに設定されている値
para4	加算値	300

メッセージキュー

パラメタ	種類	必要数
kernel.msgmax	設定値	128 以上
kernel.msgmnb	設定値	4,096 以上
kernel.msgmni	加算値	2

■システムパラメタ(Oracleデータベースの運用に必要なシステム資源)

Oracleデータベースの運用に必要なシステムパラメタの設定は、Oracleデータベースをインストールしたマシンで変更してください。

- OracleデータベースをSolarisにインストールした場合
- OracleデータベースをLinuxにインストールした場合

レプリケーション運用をするときは、レプリケーションの運用に必要なシステム資源の設定をしてください。設定内容は、Oracleデータベースのマニュアルを参照してください。

OracleデータベースをSolarisにインストールした場合

共用メモリ

パラメタ (注)	資源制御	種類	必要数
shmmax	project.max-shm-memory	設定値	4,294,967,295 以上
shmmni	project.max-shm-ids	設定値	100 以上

注)

パラメタ欄には、“shmsys:shminfo_xxxxxx”の“xxxxxx”を記載しています。

セマフォ

パラメタ (注1)	資源制御	種類	必要数
semmni	project.max-sem-ids	設定値	100 以上
semmns (注2)	－	設定値	1,024 以上
semmsl	project.max-sem-nsems	設定値	256 以上
semvmx (注2)	－	設定値	32,767 以上

注1)

パラメタ欄には、“semsys:seminfo_xxxxxx”の“xxxxxx”を記載しています。

注2)

Solaris 9でのみ有効です。

OracleデータベースをLinuxにインストールした場合

共用メモリ

パラメタ	種類	必要数
kernel.shmall	設定値	2,097,152 以上
kernel.shmmax	設定値	物理メモリのサイズ(バイト)の1/2 以上
kernel.shmmni	設定値	4,096 以上

セマフォ

パラメタ	種類	必要数
para1	設定値	250 以上
para2	設定値	32,000 以上
para3	設定値	100 以上
para4	設定値	128 以上

ファイルシステム

パラメタ	種類	必要数
fs.file-max (最大ファイル・ハンドル数)	設定値	65,536 以上

ネットワーク

パラメタ	種類	必要数
net.ipv4.ip_local_port_range (ポート番号の範囲)	設定値	最小:1,024 最大:65,000
net.core.rmem_default (受信用ウィンドウ・サイズのデフォルト値)	設定値	1,048,576 以上
net.core.rmem_max (受信用ウィンドウ・サイズの最大値)	設定値	1,048,576 以上
net.core.wmem_default (送信用ウィンドウ・サイズのデフォルト値)	設定値	262,144 以上
net.core.wmem_max (送信用ウィンドウ・サイズの最大値)	設定値	262,144 以上

3.1.10 Interstage管理コンソールのシステム資源の設定

Interstage管理コンソールを用いたシステムの運用時には、システム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ

■システムパラメタ

Interstage管理コンソールを運用するためのサービスが使用するセマフォのシステムパラメタのチューニングについて説明します。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

セマフォ

パラメタ	種類	必要数
para1	設定値	1以上
para2	加算値	1
para3	設定値	1以上
para4	加算値	1

3.1.11 Webサーバコネクタのシステム資源の設定

Webサーバコネクタを使用する場合には、システム資源を拡張する必要があります。ここでは、以下について説明します。

- システムパラメタ
  - Webサーバコネクタを使用する場合
  - Webサーバコネクタの故障監視機能を使用する場合

■システムパラメタ

Webサーバコネクタを使用する際には、以下のシステムパラメタのチューニングを行ってください。
Webサーバコネクタの故障監視機能を使用する場合は、Webサーバコネクタで使用する資源にWebサーバコネクタの故障監視機能で使用する資源量を加算してください。
システムパラメタの変更方法や、各パラメタの意味については、“■システムパラメタについて”を参照してください。

Webサーバコネクタ

Webサーバコネクタを使用する場合に必要となるシステム資源について、以下に示します。

セマフォ

パラメタ	種類	必要数
para1	設定値	1以上
para2	加算値	以下をWebサーバごとに求めた合算値 (注) ・ ((IJServerワークユニット数の合計値 ＋ 各IJServerワークユニットのプロセス多重度の合計値 ＋ 2)×2) ＋ 4
para3	設定値	2 以上
para4	加算値	以下をWebサーバごとに求めた合算値 (注) ・ ((IJServerワークユニット数の合計値 ＋ 各IJServerワークユニットのプロセス多重度の合計値 ＋ 2)×2) ＋ 4

注)

チューニングの例を以下に示します。

条件		
IJServerワークユニット	WU001	プロセス多重度:3
	WU002	プロセス多重度:3
	WU003	プロセス多重度:3
Webサーバ	web001	接続先:WU001、WU002
	web002	接続先:WU003

- para2

  web001の必要数 ＝
  ((2(IJServerワークユニット数の合計値) ＋
  6(各IJServerワークユニットのプロセス多重度の合計値) ＋ 2)×2) ＋ 4
  ＝ 24
  web002の必要数 ＝
  ((1(IJServerワークユニット数の合計値) ＋
  3(各IJServerワークユニットのプロセス多重度の合計値) ＋ 2)×2) ＋ 4
  ＝ 16
  para2 ＝ 24(web001の必要数) ＋ 16(web002の必要数) ＝ 40

- para4

  para2と同様に計算してください。

Webサーバコネクタの故障監視機能

Webサーバコネクタの故障監視機能を使用する場合に追加となるシステム資源について、以下に示します。

共用メモリ

パラメタ	種類	必要数
kernel.shmmax	設定値	6,720,012 以上
kernel.shmmni	加算値	Webサーバ数 + 1

セマフォ

パラメタ	種類	必要数
para1	設定値	2 以上
para2	加算値	3
para3	設定値	2 以上
para4	加算値	3

3.2 マルチサーバ管理機能を使用する時に必要なシステム資源

Interstageのマルチサーバ管理機能を使用する時に必要となるシステム資源について説明します。

なお、システムパラメタを算出するためのExcelファイルがマニュアルCDの“ApplicationServer¥tuning”フォルダに“ISAS-IPCtuning.xls”として格納されています。Microsoft(R) Excel 2000もしくは以降のバージョンのMicrosoft(R) Excelをお持ちの場合は“ISAS-IPCtuning.xls”を使用してシステムパラメタを算出することが可能です。使用方法などの詳細については、当該Excelファイル内の説明記事を参照してください。

■管理サーバ機能を使用する場合

管理サーバ機能を使用する場合に必要なシステム資源量は、“3.1 サーバ機能運用時に必要なシステム資源”の同一サービスの説明を参照してください。

■管理対象サーバとして運用する場合

管理対象サーバとして運用する場合は、使用する機能に関するシステム資源が必要です。必要量については、“3.1 サーバ機能運用時に必要なシステム資源”の同一サービスの説明を参照してください。

■共存サーバとして運用する場合

共存サーバでは管理サーバ機能とInterstageのサーバ機能(管理対象サーバ)が同一マシン上で動作しています。共存サーバ運用時に必要なシステム資源については、“■管理サーバ機能を使用する場合”と“■管理対象サーバとして運用する場合”を参照して使用するサービスを列挙してください。各サービスが必要とするシステム資源は、“3.1 サーバ機能運用時に必要なシステム資源”の同一サービスの説明を参照してください。

管理サーバ機能とInterstageのサーバ機能で同一サービスを使用する場合、そのサービスの必要資源量を2倍する必要はありません。

3.3 性能監視ツール使用時に必要なシステム資源

性能監視ツールで性能監視環境として使用する資源の見積もり方法を説明します。

3.3.1 システム構成情報の見積もり方法

システム構成情報は、以下の見積もり式を参考に見積もり、見積もり値以上の値を設定してください。

セマフォ

kernel.sem = *para1 para2 para3 para4*

システム構成情報	見積もり
para1	以下を満たす値を設定 (*) para1 ＞＝ 既存の値 ＋ セマフォ数 セマフォ数 ＝ 性能監視ツール起動時に指定する共有メモリ量(MB) × 10 ＋ 2 ただし、最大52
para2	para1と同じ値を加算 para1は、(*)で見積もった値
para3	para1と比較し大きな方の値を設定 para1は、(*)で見積もった値を元に比較します
para4	1を加算

共有メモリ

システム構成情報	見積もり
kernel.shmmax	“3.3.2 共有メモリ量の見積もり方法”を参照
kernel.shmmni	1を加算

3.3.2 共有メモリ量の見積もり方法

共有メモリ量は、以下の見積もり式を参考に必要な共有メモリ量を求め、その共有メモリ量をメガバイト単位に切り上げた値で見積もってください。

共有メモリ量の見積もり(単位:バイト)

1. 性能監視ツールの性能監視対象とする全オブジェクトに対して、**オブジェクトごとに必要な共有メモリ量**を、以下の方法で求めてください。
    1. 各アプリケーションに定義されている**プロセス多重度の合計**を求めます。
    2. 各アプリケーションに登録されている**オペレーション数の平均**を求めます。
       オペレーション数はIDL定義ファイルから求めてください。オペレーション数の平均は、小数点以下を切り上げてください。

3. オペレーション数の平均により、以下の計算で、**オブジェクトごとに必要な共有メモリ量**を求めます。

   - **オペレーション数の平均**が**3以下**の場合
     (プロセス多重度の合計 × 1536) ＋ 400
   - **オペレーション数の平均**が**3よりも多い**場合
     (オペレーション数の平均 × プロセス多重度の合計 × 546) ＋ 400

2. 以下の計算で、**必要な共有メモリ量**を求めます。

   オブジェクトごとに必要な共有メモリ量の合計 ＋ 261188

3. **共有メモリ量**を、以下の計算で求めます。

   必要な共有メモリ量 ÷ 1048576

   なお、上記の計算で端数(小数点以下)が発生した場合は、切り上げてください。

インターバル時間内に性能監視対象のオブジェクトを再起動する場合には、1)～3)で求めた「オブジェクトごとに必要な共有メモリ量の合計」について、再起動回数分、あらかじめ積算してください。

3.4 IPC資源のカスタマイズ

ここでは、その他に必要となるカスタマイズ項目について説明します。

■System V IPC資源のIPCキー値のカスタマイズについて

Interstageでは、Interstageを構成するプロセス間通信のために、OSが提供するSystem V IPC資源(メッセージキュー、セマフォ、共有メモリ)を使用しています。このIPC資源は、作成時に指定する値(IPCキー値)により、システムで一意に識別されます。
IPCキー値は、システムで一意でなければなりませんが、任意の値を使用することが可能なため、ほかのIPC資源を使用する製品およびアプリケーションプログラムと重複することがあります。

IPCキー値の重複が発生した場合、Interstageでは、以下のようなメッセージを出力して、IPCキー値の重複を通知します。

IPCキー値の重複を検出したときにコンポーネントトランザクションサービスが出力するメッセージの例

・ TD: エラー: td11038:必要なIPC資源が使用中のため獲得できませんでした(key=%x path=%s)

この場合、IPCキー値に対応したIPC資源を使用しているInterstageのサービスの各機能は使用できません。
このような状態に対処するため、Interstageでは以下に示す方法で、Interstageが使用するIPCキー値をカスタマイズすることが可能です。IPCキー値の重複発生を通知するメッセージが出力された場合は、この対処により運用することができます。

■概要

IPCキー値は4バイト(32ビット)で構成されますが、そのうちの下位12ビット(16進3桁)に任意の値を定義することで、ほかの製品が使用するIPCキー値と重複しないようにします。なお、上位残りの20ビットは、Interstageが決定します。

■IPCキー値の定義方法

以下のIPCキー値定義ファイルを新規に作成し、16進3桁でIPCキー値の下位12ビットを指定します。
MessageQueueDirectorを除くサービスは、共通定義ファイルの指定が有効です。MessageQueueDirectorは固有の定義ファイルの指定が有効です。

共通定義ファイル:

Solaris

/var/opt/FJSVisas/system/システム名/FJSVisas/etc/ipc_key

Linux

/var/opt/FJSVisas/system/default/FJSVisas/etc/ipc_key

MQDシステムの定義ファイル:

/opt/FJSVmqd/mqd/MQDシステム名/ipc_key

- 定義ファイルの内容が16進3桁以外の場合は、IPCキー値の指定がない場合と同様に動作します。
- “MQDシステム名”については、“MessageQueueDirector 説明書”を参照してください。
- Solaris
  “システム名”は、マルチシステム機能のシステム名です。マルチシステム機能を使用していない場合は、“default”となります。
- Linux
  クラスタシステムを使用する場合、MQDシステム用の定義ファイルは、mqd環境定義ファイル中の以下で指定したディレクトリの中に定義ファイル(ipc_key)を作成してください。

  ー [Cluster]
  SystemDirectory = MQDのクラスタサービスが使用するディレクトリの名前


```
FFF
```

この定義例の場合、Interstageが使用する上位20ビットを0x01280とすると、IPCキー値は、16進表示で、以下のようになります。

```
0x01280FFF
```

- IPCキー値の定義は、Interstageを全強制停止モードで停止し、さらにInterstage JMXサービスを停止してから行ってください。
  Interstageの運用中に本定義を変更しないでください。
- Solaris
  IPCキー値の定義は、システム間で重複することのないように定義してください。

第4章 ワークユニットのチューニング

ワークユニットには、様々な機能があり、これらをチューニングすることで、最適な状態での運用を行うことが可能となります。ここでは、ワークユニットのチューニングについて説明します。

4.1 プロセス強制停止時間のチューニング

プロセス強制停止時間は、ワークユニット停止においてアプリケーションプロセスの停止処理がハングアップしていると判断する時間です。
プロセス強制停止時間をチューニングする場合は、アプリケーションプロセス停止処理に必要な時間を考慮してください。以下の条件式を参考にしてください。

■条件式

プロセス強制停止時間(秒) ＞ アプリケーションプロセス停止処理時間(秒)

アプリケーションプロセス停止処理時間:
以下の場合において、アプリケーションプロセス停止処理時間を考慮してください。その他の場合は、デフォルト値で問題ありません。

- CORBAワークユニットの場合
  サーバアプリケーションの活性化後の動作モードに“SYNC_END(サーバアプリケーションを活性化しても、活性化メソッドは復帰しません。)”を設定し、かつ、活性化メソッドの後に後処理を記述している場合、後処理の実行に必要な時間
  サーバアプリケーションの活性化後の動作モードについては、“リファレンスマニュアル(コマンド編) OD_impl_inst(「CORBAアプリケーション情報定義ファイルでの登録」の「mode」)”を参照してください。
- IJServerワークユニットの場合
  停止時実行クラスを登録している場合、停止時実行クラスの実行に必要な時間
  停止時実行クラスについては、“J2EE ユーザーズガイド(「J2EEアプリケーションの設計」「J2EEアプリケーションが運用される環境(IJServer) 」の「起動/停止の実行クラス」)”を参照してください。


ポイント

ワークユニットを停止する時は、お客様の業務が終了していることを想定しています。そのため、デフォルト値には、ハングアップしていると判断して差し支えのない値として、180秒を設定しています。

業務運用中にワークユニットを停止することがある場合は、アプリケーション処理時間を考慮する必要があります。以下の条件式を参考にしてください。
プロセス強制停止時間はアプリケーションの処理時間とプロセスの停止処理時間の合計を上回るように設定してください。

プロセス強制停止時間(秒) ＞ アプリケーション処理時間(秒) ＋ アプリケーションプロセス停止処理時間(秒)

アプリケーション処理時間:
アプリケーション処理中にワークユニットを停止するような運用を実施する場合、そのアプリケーションの処理が正常に完了するために必要な、最も長い時間を設定してください。
ワークユニット同期停止は、アプリケーション処理中に実行すると、処理中の要求が完了するのを待ってからプロセスを停止します。
アプリケーション処理時間の考慮がされていないと、ワークユニット同期停止において、処理中の要求が途中で強制終了されます。
注)IJServerワークユニットの通常停止は同期停止と同じ動きとなります。

アプリケーションプロセス停止処理時間:
■条件式の**アプリケーションプロセス停止処理時間**を参照してください。


例

アプリケーション処理時間が120秒で、アプリケーションプロセス停止処理時間が5秒の場合、プロセス強制停止時間は、126秒以上を設定してください。

4.2 CORBAワークユニットのチューニング

CORBAワークユニットは、以下の製品で使用可能です。

- Interstage Application Server Enterprise Edition

ワークユニットのチューニングについては、以下のマニュアルを参照してください。

- “OLTPサーバ運用ガイド”の“ワークユニットの設計”
- “OLTPサーバ運用ガイド”の“CORBAワークユニット”

4.3 IJServerワークユニットのチューニング

IJServerワークユニットについては、以下を参照してください。

- “IJServerのチューニング”

4.4 ユーティリティワークユニットのチューニング

Solaris Linux

ユーティリティワークユニットは、以下の製品が使用可能です。

- Interstage Application Server Enterprise Edition

ユーティリティワークユニットについては、以下のマニュアルを参照してください。

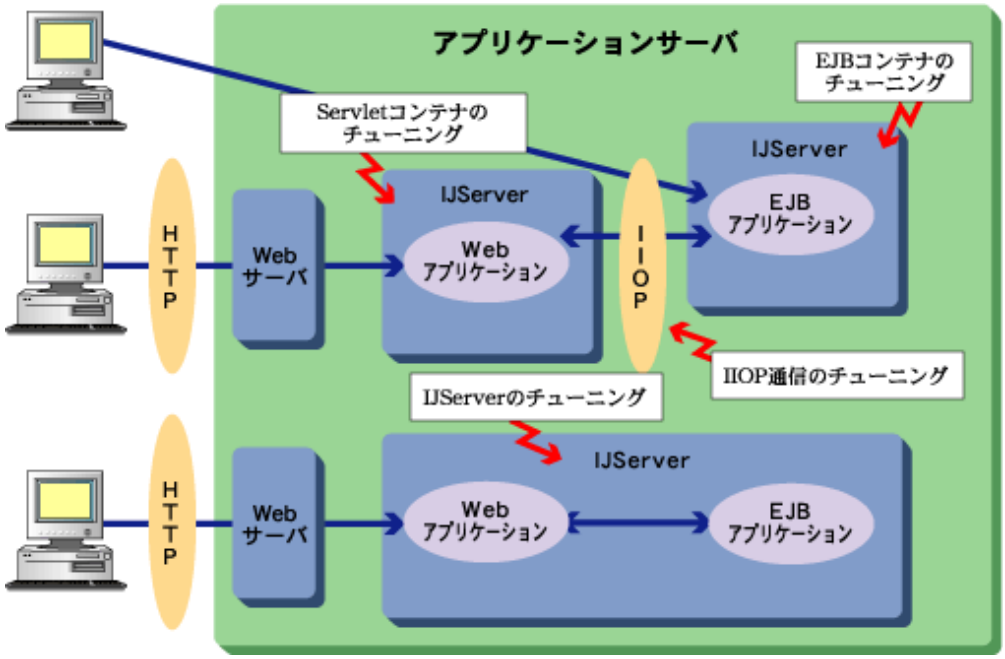
- “OLTPサーバ運用ガイド”の“ワークユニットの設計”
- “OLTPサーバ運用ガイド”の“ユーティリティのワークユニット”

第5章 J2EEのチューニング

Interstageはシステム規模の指定だけで、システム運用が可能となるようなモデルケースを設定して各サービス定義を登録しています。J2EEアプリケーションを動作させるためには、これらの定義に加え、J2EEを構成する各コンポーネントでチューニングが必要となります。

ここでは、下記のようなJ2EEアプリケーションの形態を例に、チューニングに関する設定を説明します。また、J2EEアプリケーションのチューニングに役立つ“5.1 J2EEモニタロギング機能”についても説明します。

- IJServerのチューニング
- Servletコンテナのチューニング
- EJBコンテナのチューニング
- IIOP通信のチューニング
  詳細は、“付録A CORBAサービスの動作環境ファイル”を参照してください。
- LDAPサーバとしての、ディレクトリサービスのチューニング



5.1 J2EEモニタロギング機能

J2EEモニタロギング機能とは、IJServerの性能情報をロギングする機能です。この機能を利用して、JavaVMやJDBCデータソースなどの性能情報を定期的に採取し、結果をログファイルへ出力することができます。ログファイルはCSV形式のファイルに出力されるため、容易にMicrosoft(R) Excelなどで読み込んで性能情報を分析することができ、統計情報の蓄積にも役立ちます。

isj2eemonitorコマンドを実行すると、isj2eemonitorコマンドはInterstage JMXサービスにロギングの要求を通知します。
Interstage JMXサービスではロギングの開始要求を受け付けた場合には、指定された時間間隔で定期的にIJServerにログ出力要求を通知し、IJServerプロセス上で性能情報をCSV形式のファイルに出力します。ロギングの停止要求を受け付けた場合にはロギング処理を停止します。

■J2EEモニタロギングの対象

J2EEモニタロギングは、V9.0以降のIJServerで使用できます。
本製品では、V8.0互換モードのIJServer、または8.0以前に作成されたIJServerも運用することができますが、これらのIJServerではJ2EEモニタロギングを行うことができません。
V8.0互換モードのIJServer、または8.0以前に作成されたIJServerに対してJ2EEモニタロギングのコマンドが実行された場合、エラーが出力されます。

5.1.1 J2EEモニタロギングの操作手順

J2EEモニタロギングの操作手順について、以下を説明します。

- J2EEモニタロギングの起動操作
- J2EEモニタロギングの停止操作
- 監視操作の流れ

■J2EEモニタロギングの起動操作

isj2eemonitorコマンドを以下のように実行して、J2EEモニタロギングを起動します。

```
isj2eemonitor -start -n IJServer名
```

性能監視対象のIJServerの起動前/起動後のどちらでもJ2EEモニタロギングを起動し監視を行うことが可能です。
また、コマンド実行時にログ採取間隔や採取する情報などを指定することもできます。詳細は“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。

IJServer起動前にロギングを開始した場合、初回採取データとしては、IJServer起動時から次回のログ採取間隔経過時の集計情報が出力されます。また、IJServerが停止されると、その時点でログ出力は停止します。

■J2EEモニタロギングの停止操作

isj2eemonitorコマンドを以下のように実行して、J2EEモニタロギングを停止します。

```
isj2eemonitor -stop -n IJServer名
```

また、以下のサービスを停止した場合も、J2EEモニタロギングが停止します。

Windows

“Interstage Operation Tool”サービス

Solaris Linux

Interstage JMXサービス

■監視操作の流れ

◆特定時刻のログのみ採取する場合

トラブル調査などのため、ある特定の時間のみ性能情報を採取したい場合は、以下のようにIJServer起動後に採取したいタイミングでJ2EEモニタロギング機能を開始して性能情報を分析してください。

1. IJServerの起動
   Interstage管理コンソール、またはisstartwuコマンドでIJServerを起動します。
2. J2EEモニタロギングの開始
   isj2eemonitorコマンドでJ2EEモニタロギングを開始します。
3. 性能情報の分析
   出力された情報をMicrosoft(R) Excelなどで分析します。
4. J2EEモニタロギングの停止
   isj2eemonitorコマンドでJ2EEモニタロギングを停止します。
5. 2.～4.を繰り返します。
6. IJServerの停止
   Interstage管理コンソール、またはisstopwuコマンドでIJServerを停止します。

◆継続的にログを採取する場合

継続的にログを採取して性能チューニングの妥当性の検証を行いたい場合は、以下のようにJ2EEモニタロギング機能の起動後に、IJServerを起動して性能情報を分析してください。

1. J2EEモニタロギングの開始
   isj2eemonitorコマンドでJ2EEモニタロギングを開始します。
2. IJServerの起動
   Interstage管理コンソール、またはisstartwuコマンドでIJServerを起動します。
3. 性能情報の分析
   出力された情報をMicrosoft(R) Excelなどで分析します。
4. IJServerの停止
   Interstage管理コンソール、またはisstopwuコマンドでIJServerを停止します。
5. J2EEモニタロギングの停止
   isj2eemonitorコマンドでJ2EEモニタロギングを停止します。
6. 1.～5.を繰り返します。

5.1.2 J2EEモニタロギングのログファイル

J2EEモニタロギングのログファイルについて、以下を説明します。

- ログファイル名
- ログファイルの出力条件
- ロールオーバ後のファイル名
- ログファイルのライフサイクル
- ファイルのアクセス権
- 出力ディレクトリ

■ログファイル名

モニタ情報を出力するログファイルは、プロセスごと、かつ、ロギング対象ごとに出力されます。ログファイルのファイル名は以下の命名規則で出力されます。

monitor-[ロギング対象名].log

ファイル名一覧を以下に記載します。

ファイル	ファイル名
JavaVM情報ログファイル	monitor-JavaVM.log
データソース情報ログファイル	monitor-DataSource.log
トランザクション情報ログファイル	monitor-Transaction.log
Servletコンテナ情報ログファイル	monitor-ServletContainer.log
EJBコンテナ情報ログファイル	monitor-EJBContainer.log

■ログファイルの出力条件

ログファイルの作成と性能情報の出力は、指定されたIJServerが起動している場合のみ行われます。
また、IJServerのタイプにより、出力可能なログファイルの種類が異なります。IJServerタイプごとの出力可能ログファイルを以下に示します。

	WebアプリケーションとEJBアプリケーションを同一JavaVMで運用	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	Webアプリケーションのみ運用	EJBアプリケーションのみ運用
JavaVM情報ログファイル	○	○	○	○
データソース情報ログファイル	○ (注1)	○ (注1)	○ (注1)	○ (注1)
トランザクション情報ログファイル	○	○	○	○
Servletコンテナ情報ログファイル	○	○ (注2)	○	－
EJBコンテナ情報ログファイル	○	○ (注3)	－	○

注1) データソースを使用している場合のみ、データソース情報ログファイルが出力されます。
注2) Webアプリケーションを運用するJavaVMでのみ、Servletコンテナ情報ログファイルが出力されます。
注3) EJBアプリケーションを運用するJavaVMでのみ、EJBコンテナ情報ログファイルが出力されます。

■ロールオーバ後のファイル名

出力されたログファイルは一定間隔でロールオーバされます。ロールオーバ後のログファイルは、以下のようにロールオーバした日時情報を付加してバックアップされます。ログファイル名のロギング対象名(例:“JavaVM”)と拡張子“.log”の間に、日時を示す文字列が挿入されます。また、ロギング対象名と日時文字列の間はハイフン(“-”)で区切られます。

```
monitor-[ロギング対象名]-YYYY_MM_DD-hh_mm_ss.log
```

日時情報について以下に説明します。

YYYY	年を4桁の数字(0000～9999)で表示します。
MM	月を2桁の数字(01～12)で表示します。
DD	日を2桁の数字(01～31)で表示します。
hh	時を2桁の数字(00～23)で表示します。
mm	分を2桁の数字(00～59)で表示します。
ss	秒を2桁の数字(00～59)で表示します。

例) monitor-JavaVM-2006_06_24-01_00_00.log

■ログファイルのライフサイクル

ログファイルのライフサイクルについて説明します。
IJServerプロセスがログを出力する場合、ログ出力ディレクトリに“■ログファイル名”で説明しているファイルが存在しなければファイルを新規に作成します。また、ログファイルは以下の条件の場合にロールオーバされます。

ロールオーバ条件

- ロールオーバ時刻にIJServerプロセスが起動している場合
- ログが採取されるタイミングで、以前のログファイルがログ出力ディレクトリに残存しており、かつ、そのファイルの更新日時が前回のロールオーバ日時より前の場合

ロールオーバ開始時刻はJ2EEモニタロギングを開始する時に指定できます。詳細は、“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。

ロールオーバは以下のように行われます。

1. バックアップされているファイルのファイル数がログファイルの世代数以上となっている場合にファイルの更新日時が古いファイルを削除します。バックアップされたファイルのファイル数が“世代数－1”となるまで削除します。
2. 既存のログファイルを“■ロールオーバ後のファイル名”で説明している名前に変名してバックアップします。
3. 新規に“■ログファイル名”で説明している名前の新規ファイルを作成します。

■ファイルのアクセス権 Solaris Linux

出力されるファイルの所有者はIJServerの起動ユーザとなり、ファイルの権限は「644」となります。

■出力ディレクトリ

デフォルトのログファイル出力ディレクトリは以下です。

Windows

[J2EE共通ディレクトリ]¥ijserver¥[IJServer名]¥log¥[プロセス通番]

Solaris Linux

[J2EE共通ディレクトリ]/ijserver/[IJServer名]/log/[プロセス通番]

出力先はisj2eeadminコマンドのIJServer定義、またはInterstage管理コンソールの[ワークユニット] > “ワークユニット名” > [環境設定]タブ > [詳細設定] > [ワークユニット設定] > [ログ出力ディレクトリ]で変更できます。

5.1.3 性能情報の分析と対処

ログファイルに採取した性能情報の分析方法と対処方法について説明します。

■出力情報

モニタ情報ログファイルは、以下のようにCSV形式(カンマ区切りで並べた形式)で出力します。

```
D1,D2,D3,D4,D5,・・・
```

■性能情報の項目内容

Interstage管理コンソールで参照可能なモニタ情報はIJServer起動時からの集計データを表示していましたが、J2EEモニタロギング機能で採取されるデータは性能ボトルネックなどの解析に使用可能とするためデータ採取区間内での集計データ(データ採取時間帯での最大値など)を採取します。

また、Interstage管理コンソールの場合、一部のデータ(JavaVMなど)を除いてIJServer全体のデータを集計してモニタ情報として参照できましたが、J2EEモニタロギング機能では以下のようにIJServerプロセスごとのデータを採取可能であるため、より詳細な分析が可能となります。

	Interstage管理コンソール	J2EEモニタロギング
上限値	IJServer全体で使用可能な上限値	各プロセスで使用可能な上限値
最大値	IJServer全体の最大値	各プロセスの最大値
最小値	IJServer全体の最小値	各プロセスの最小値
平均値	IJServer全体の平均値	各プロセスの平均値
現在値	IJServer全体の現在値	各プロセスの現在値

性能情報中の日時情報は、以下の値が“DD/MM/YYYY hh:mm:ss:SSS”のフォーマットで出力されます。

DD　日を2桁の数字(01～31)で表示します。

MM　月を2桁の数字(01～12)で表示します。

YYYY　年を4桁の数字(0000～9999)で表示します。

hh　時を2桁の数字(00～23)で表示します。

mm　分を2桁の数字(00～59)で表示します。

ss　秒を2桁の数字(00～59)で表示します。

SSS　ミリ秒を3桁の数字(000～999)で表示します。

例) 24/06/2006 01:00:00:200

性能情報中の「ミリ秒」単位は、以下の値が“h:mm:ss:SSS”のフォーマットで出力されます。

h　時を可変桁の数字(0～)で表示します。値が0の場合は“0”と表示します。

mm　分を2桁の数字(00～59)で表示します。

ss　秒を2桁の数字(00～59)で表示します。

SSS　ミリ秒を3桁の数字(000～999)で表示します。

例) 0:00:00:200

性能情報として出力される項目について説明します。各表の項番に書かれているD1、D2、・・・は、CSV形式で出力されるD1、D2、・・・に対応しています。

1)JavaVM情報

項番	項目名	単位	内容
D1	データ採取開始日時	－	当該レコードの性能情報の測定を開始した日時
D2	データ採取終了日時	－	当該レコードの性能情報の測定を終了した日時
D3	プロセス通番	－	IJServerが起動したプロセスの通番
D4	プロセスID	－	測定対象のIJServerのプロセスID
D5	コンテナタイプ	－	JavaVMのコンテナタイプ。以下のいずれかを出力します。 ・1VM(ServletとEJBを運用するプロセス) ・Web(Servletのみ運用するプロセス) ・EJB(EJBのみ運用するプロセス)
D6	JavaVMの運用時間	－	IJServer起動時からの総時間
D7	ヒープ現在使用量	KB	ログ採取時のJavaVMのヒープ使用量
D8	ヒープ最小使用量	KB	測定時間内でのJavaVMの最小ヒープ使用量 (注1)
D9	ヒープ最大使用量	KB	測定時間内でのJavaVMの最大ヒープ使用量 (注1)
D10	ヒープ上限値	KB	ヒープサイズの上限値(-Xmxオプションで指定した値とほぼ同等の値)
D11	Perm領域現在使用量	KB	ログ採取時のJavaVMのPerm領域使用量
D12	Perm領域最小使用量	KB	測定時間内でのJavaVMの最小Perm領域使用量 (注1)
D13	Perm領域最大使用量	KB	測定時間内でのJavaVMの最大Perm領域使用量 (注1)
D14	Perm領域上限値	KB	Perm領域サイズの上限値(-XX:MaxPermSizeオプションで指定した値とほぼ同等の値)
D15	ガーベジコレクション発生回数	回	測定時間内でのガーベジコレクションの発生回数 (注2)
D16	ガーベジコレクション処理トータル時間	ミリ秒	測定時間内でのガーベジコレクションの処理時間の合計値 (注2)

注1) JavaVMヒープ/Permanent領域の最小値/最大値については、Interstage管理コンソールのモニタ表示画面と同様に、3秒間隔でサンプリングした値の最小値/最大値を出力します。
注2) ガーベジコレクションは、Full GCの情報を使用しています。

2)データソース情報

項番	項目名	単位	内容
D1	データ採取開始日時	－	当該レコードの性能情報の測定を開始した日時
D2	データ採取終了日時	－	当該レコードの性能情報の測定を終了した日時
D3	プロセス通番	－	IJServerが起動したプロセスの通番
D4	プロセスID	－	測定対象のIJServerのプロセスID
D5	データソース名	－	データソース名
D6	物理コネクション現在数	個	ログ採取時に確立されている物理コネクション数 (注1)
D7	物理コネクション最小数	個	測定時間内で同時に確立したデータベースの最小物理コネクション数 (注1)
D8	物理コネクション最大数	個	測定時間内で同時に確立したデータベースの最大物理コネクション数 (注1)
D9	物理コネクション上限数	個	確立可能なデータベースへの上限物理コネクション数 (注1)
D10	使用コネクション現在数	個	ログ採取時に使用されているコネクション数 (注1)
D11	使用コネクション最小数	個	測定時間内でプーリングされているコネクションで同時に使用された最小コネクション数 (注1)
D12	使用コネクション最大数	個	測定時間内でプーリングされているコネクションで同時に使用された最大コネクション数 (注1)
D13	累積コネクション待ち回数	回	測定時間内でコネクション待ちとなった回数(コネクション待ちタイムアウトが発生した場合は測定対象外) (注2)
D14	平均コネクション待ち時間	ミリ秒	測定時間内でコネクション待ちとなった時の平均待ち時間(コネクション待ちタイムアウトが発生した場合は測定対象外) (注2)
D15	コネクション待ち最小時間	ミリ秒	測定時間内でコネクション待ちとなった時の最小待ち時間(コネクション待ちタイムアウトが発生した場合は測定対象外) (注2)
D16	コネクション待ち最大時間	ミリ秒	測定時間内でコネクション待ちとなった時の最大待ち時間(コネクション待ちタイムアウトが発生した場合は測定対象外) (注2)
D17	コネクション待ち現在スレッド数	個	ログ採取時にコネクション待ちとなっているスレッド数 (注2)
D18	コネクション待ち最小スレッド数	個	測定時間内でコネクション待ちとなった時の最小待ちスレッド数 (注2)
D19	コネクション待ち最大スレッド数	個	測定時間内でコネクション待ちとなった時の最大待ちスレッド数 (注2)
D20	コネクション待ちタイムアウト回数	回	測定時間内でコネクション待ちとなった時のコネクション待ちタイムアウトが発生した回数 (注2)
D21	物理コネクション確立回数	回	測定時間内でデータベースの物理コネクションを確立した回数 (注2)
D22	物理コネクション最大確立時間	ミリ秒	測定時間内でプーリングしていたコネクションのデータベースへの最大確立時間 (注2)

項番	項目名	単位	内容
D23	アイドルタイムアウトによるクローズ回数	回	測定時間内でアイドルタイムアウトによりクローズした物理コネクション数 (注2)
D24	例外発生によるクローズ回数	回	測定時間内でコネクション接続時もしくはクローズ時に例外発生によりクローズした物理コネクション数 (注2)
D25	コネクション獲得回数	回	測定時間内でアプリケーションがgetConnectionメソッドを実行してコネクションを獲得した回数 (注3)
D26	コネクションクローズ回数	回	測定時間内でアプリケーションがcloseメソッドを実行してコネクションを解放した回数 (注3)
D27	通信待ちタイムアウト回数	回	測定時間内で通信待ちタイムアウトが発生した回数 (注3)
D28	コネクション平均使用時間	ミリ秒	測定時間内にアプリケーションでコネクションを使用した平均時間 (注3)
D29	コネクション最小使用時間	ミリ秒	測定時間内にアプリケーションでコネクションを使用した最小時間 (注3)
D30	コネクション最大使用時間	ミリ秒	測定時間内にアプリケーションでコネクションを使用した最大時間 (注3)
D31	コネクション使用タイムアウト回数	回	測定時間内でコネクション使用タイムアウトが発生した回数 (注3)

注1) InterstageでJDBCコネクションをプーリングしている、または、データベースがOracle10g以降かつJDBCドライバでJDBCコネクションをプーリングしている場合に、値を出力します。その他の場合は、“-”(ハイフン)が出力されます。
注2) InterstageでJDBCコネクションをプーリングしている場合のみ値を出力します。JDBCドライバでJDBCコネクションをプーリングしている場合、“-”(ハイフン)が出力されます。
注3) アプリケーションの前後で動作するコンテナ制御処理で使用されるコネクションの情報も合わせて表示されます。

※ 未使用のデータソースの性能情報は出力されません。

3)トランザクション情報

項番	項目名	単位	内容
D1	データ採取開始日時	－	当該レコードの性能情報の測定を開始した日時
D2	データ採取終了日時	－	当該レコードの性能情報の測定を終了した日時
D3	プロセス通番	－	IJServerが起動したプロセスの通番
D4	プロセスID	－	測定対象のIJServerのプロセスID
D5	実行トランザクション総数	個	測定時間内にアプリケーションで実行したJ2EEトランザクション数 (注1)
D6	コミットトランザクション数	個	測定時間内にアプリケーションでコミットしたJ2EEトランザクション数 (注1)
D7	ロールバックトランザクション数	個	測定時間内にアプリケーションでロールバックしたJ2EEトランザクション数 (注1)
D8	トランザクション平均時間	ミリ秒	測定時間内にアプリケーションで実行したJ2EEトランザクションの平均時間 (注1)
D9	トランザクション最小時間	ミリ秒	測定時間内にアプリケーションで実行したJ2EEトランザクションの最小時間 (注1)
D10	トランザクション最大時間	ミリ秒	測定時間内にアプリケーションで実行したJ2EEトランザクションの最大時間 (注1)
D11	現在実行トランザクション数	個	ログ採取時の実行中であったJ2EEトランザクション数 (注1)

項番	項目名	単位	内容
D12	最小同時実行トランザクション数	個	測定時間内に同時に実行された最小J2EEトランザクション数 (注1)
D13	最大同時実行トランザクション数	個	測定時間内に同時に実行された最大J2EEトランザクション数 (注1)

注1) コンテナで制御されるトランザクションの情報も合わせて表示されます。

4)Servletコンテナ情報

項番	項目名	単位	内容
D1	データ採取開始日時	―	当該レコードの性能情報の測定を開始した日時
D2	データ採取終了日時	―	当該レコードの性能情報の測定を終了した日時
D3	プロセス通番	―	IJServerが起動したプロセスの通番
D4	プロセスID	―	測定対象のIJServerのプロセスID
D5	現在使用スレッド数		ログ採取時に使用中であったスレッド数
D6	最小同時使用スレッド数	個	測定時間内に同時に使用した最小スレッド数 (注1)
D7	最大同時使用スレッド数	個	測定時間内に同時に使用した最大スレッド数 (注2)
D8	現在トータルスレッドプール数	個	ログ採取時にプールされていたスレッド数 (注3)
D9	最小トータルスレッドプール数	個	測定時間内にプールされていたスレッドの最小数 (注3)
D10	最大トータルスレッドプール数	個	測定時間内にプールされていたスレッドの最大数 (注3)

注1) クライアントからのリクエスト受け付け用にスレッドは1個使用されるため、クライアントからのリクエストがまったくない場合でも最小同時使用スレッド数は1が出力されます。
注2) クライアントからのリクエストを受付けるServletコンテナのServerSocketは数秒おきに待ち状態から復帰し、再度クライアントからのリクエストの待ち状態になります。そのため、クライアントからのリクエストがまったくない場合でも最大同時使用スレッド数は2が出力されます。
注3) クライアントからのリクエストが一定時間ない場合は、Servletコンテナの待機中の最大値に設定された値にしたがって余分な処理スレッドが破棄されます。このとき、クライアントからのリクエストを受け付けるスレッドは破棄対象となりませんので、待機中の最大値＋1 が最小値となります。

5)EJBコンテナ情報

項番	項目名	単位	内容
D1	データ採取開始日時	―	当該レコードの性能情報の測定を開始した日時
D2	データ採取終了日時	―	当該レコードの性能情報の測定を終了した日時
D3	プロセス通番	―	IJServerが起動したプロセスの通番
D4	プロセスID	―	測定対象のIJServerのプロセスID
D5	Message-driven Bean最大同時処理数	個	Message-driven Beanでプーリングできるスレッド数の最大値
D6	Message-driven Bean現在スレッドプールサイズ	個	ログ採取時にMessage-driven Beanでプーリングしていたスレッド数
D7	Message-driven Bean最小スレッドプールサイズ	個	Message-driven Beanでプーリングした最小スレッド数
D8	Message-driven Bean最大スレッドプールサイズ	個	Message-driven Beanでプーリングした最大スレッド数

項番	項目名	単位	内容
D9	Message-driven Bean現在アイドルスレッド数	個	ログ採取時にMessage-driven Beanでプーリングしたスレッドで未使用であったスレッド数
D10	Message-driven Bean最小アイドルスレッド数	個	Message-driven Beanでプーリングしたスレッドで未使用であったスレッドの最小スレッド数
D11	Message-driven Bean最大アイドルスレッド数	個	Message-driven Beanでプーリングしたスレッドで未使用であったスレッドの最大スレッド数
D12	Message-driven Beanアイドルタイムアウト発生回数	個	Message-driven Beanでプーリングしたスレッドをアイドルタイムアウトで破棄したスレッド数

■評価方法と対処方法

1)JavaVM情報

項番	評価方法	対応/処置
1	途中ガーベジコレクションが発生しているにもかかわらず、ヒープ使用量が増加傾向にある。	メモリリークの可能性があります。 不要になったオブジェクトが参照され続けていることがないかなど、サーバアプリケーションの見直しを実施してください。 また、Webアプリケーションのセションタイムアウト時間を極端に長く設定している場合、設定値が妥当か見直しを実施してください。
2	ヒープ最小使用量がヒープ上限値に近い状態が続いている。	運用を継続するとヒープ不足になる可能性があります。 指定しているヒープサイズの上限を増やすなど、IJServerの設定を見直してください。
3	Perm領域最小使用量がPerm領域上限値に近い値となっている。	運用を継続するとPerm領域不足になる可能性があります。 指定しているPerm領域サイズを増やす等、IJServerの設定を見直してください。
4	ガーベジコレクション発生回数が多い。	ガーベジコレクションが多く発生することでアプリケーションの処理性能が劣化している可能性があります。 JavaVMヒープの最小使用量が上限値と近い値となっている場合、ヒープ不足によりガーベジコレクションが多発している可能性があります。ヒープの上限を増やすなど、IJServerの設定を見直してください。
5	ガーベジコレクションの処理トータル時間が長いときがある。	IJServerを運用するサーバのCPU負荷が高い状態が続いている可能性があります。 不要なアプリケーションを停止するなどしてCPU負荷を軽減するか、IJServerを運用するサーバのCPU性能が適正か見直してください。 また、ヒープサイズを大きくすると、1回のガーベジコレクションで回収されるオブジェクトが増えるため、1回あたりの処理時間が長くなる傾向にあります。 ヒープの使用状況を確認し、余裕がある場合はヒープサイズの上限を小さくすることを検討してください。

※ JavaVMのチューニングについては、本チューニングガイドの以下も参照してください。

－ “J2EEのチューニング”－“IJServerのチューニング”－“5.2.2 JavaVMのヒープ領域サイズ”

－ “J2EEのチューニング”－“IJServerのチューニング”－“5.2.3 ガーベジコレクション発生回数”

－ “JDK/JREのチューニング”－“チューニング方法”

2)データソース情報

項番	評価方法	対応/処置
1	物理コネクション最大数が物理コネクション上限数に近い値となっている。	運用を継続するとコネクション待ちとなる可能性があります。指定している最大コネクション数の指定が適切か見直してください。
2	使用コネクション最大数が物理コネクション最小数よりかなり小さい値となっている。	プーリングされているコネクションと比較して、使用しているコネクションが少ないため、無駄なコネクションがプーリングされている可能性があります。 指定している事前コネクト数とアイドルコネクトタイムアウト値を見直してください。事前コネクト数を小さくすることで常時接続している物理コネクションを少なくすることできます。また、アイドルコネクトタイムアウトを小さくすることで無駄なコネクションを解放することが可能です。
3	累積コネクション待ち回数が多い。	コネクション待ちが発生することでアプリケーションの処理性能を劣化させる原因になっている可能性があります。 指定している最大コネクション数の指定が適切か見直してください。
4	平均コネクション待ち時間が長い。	コネクション待ちが発生することでアプリケーションの処理性能を劣化させる原因になっている可能性があります。 指定している最大コネクション数もしくはコネクション待ち時間の指定が適切か見直してください。
5	コネクション待ち最小時間が短く、コネクション待ち最大時間が長い。	時間帯によりコネクション待ち時間にばらつきがあるため、アプリケーションの処理性能のばらつきが発生している可能性があります。 コネクション待ち時間の設定を短くして待ち時間が長時間に及ぶ場合にはエラーを返却するなどの対策を検討してください。
6	コネクション待ち最大スレッド数が増加している。	アプリケーションで同時に使用するコネクションが多くなっているため、運用を継続するとアプリケーションの処理性能を劣化させる可能性があります。指定している最大コネクション数の指定が適切か見直してください。
7	コネクション待ちタイムアウト回数が増加している。	コネクション待ちタイムアウトが発生してエラーとなるアプリケーションが増加しています。 指定している最大コネクション数もしくはコネクション待ち時間の指定が適切か見直してください。
8	物理コネクション確立回数が多い。アイドルタイムアウトによるクローズ回数が多い。	アイドルタイムアウトにより物理コネクション数が解放されるために物理コネクト回数が多くなり、アプリケーションの処理性能を劣化させる原因になっている可能性があります。 アイドルタイムアウト値が適切か見直してください。また、事前コネクト数を指定することで物理コネクションを常時確立することもできます。
9	物理コネクション確立回数が多い。例外発生によるクローズ回数が多い。	アイドルタイムアウトにより物理コネクション数が解放されるために物理コネクト回数が多くなり、アプリケーションの処理性能を劣化させる原因になっている可能性があります。 アプリケーションを見直して例外発生原因を取り除いてください。
10	コネクション獲得回数とコネクションクローズ回数の差が大きい。	コネクションのクローズ漏れが発生している可能性があります。アプリケーションを見直してください。
11	コネクション平均使用時間が長い。	データベースアクセス処理によりアプリケーションの処理性能を劣化させる原因になっている可能性があります。 アプリケーションもしくはデータベースの設定を見直してください。

項番	評価方法	対応/処置
12	コネクション最大使用時間とコネクション最小使用時間の差が大きい。	データベースアクセス処理の何かしらの理由により、アプリケーションの処理性能のばらつきが発生している可能性があります。 アプリケーションもしくはデータベースの設定を見直してください。
13	通信待ちタイムアウトの発生回数が多い。	データベースアクセス処理によりアプリケーション処理性能が劣化している可能性があります。 データベースの処理時間、ハングの有無をチェックしてください。

※ JDBCデータソースのチューニングについては、本チューニングガイドの以下も参照してください。

－ “J2EEのチューニング”－“IJServerのチューニング”－“5.2.5 JDBCのコネクション”

－ “J2EEのチューニング”－“IJServerのチューニング”－“5.2.6 Statementキャッシュ機能”

3)トランザクション情報

項番	評価方法	対応/処置
1	実行トランザクション総数と最大同時実行トランザクション数が増加し、トランザクション平均時間も増加している。	同時に実行されるトランザクションが増加したことにより、データベースの排他制御などでアプリケーションの処理性能が劣化している可能性があります。 IJServerに指定した同時処理数を小さくして処理性能を安定させるか、IJServerサーバを運用するサーバの処理性能を見直してください。
2	実行トランザクション総数は変わらないが、最大同時実行トランザクション数とトランザクション最大時間が大きい。	一定時間帯にトランザクション処理が集中したことにより、データベースの排他制御などでアプリケーションの処理遅延が発生した可能性があります。 IJServerに指定した同時処理数を小さくして処理性能を安定させるか、IJServerサーバを運用するサーバの処理性能を見直してください。
3	コネクション最大使用時間とコネクション最小使用時間の差が大きい。	データベースアクセス処理の何かしらの理由により、アプリケーションの処理性能のばらつきが発生している可能性があります。 アプリケーションもしくはデータベースの設定を見直してください。 もしくはIJServerを運用するサーバのCPU負荷が高い状態が一時的に発生している可能性があります。不要なアプリケーションを停止するなどしてCPU負荷を軽減するか、IJServerを運用するサーバのCPU性能が適正か見直してください。
4	ロールバックトランザクション数に比べてコミットトランザクション数が多い。	コミット処理はロールバック処理に比べて一般的に負荷は高いため、アプリケーションの処理性能が劣化している可能性があります。 無駄にコミット処理を実行していないかアプリケーションを見直してください。

※ JDBCデータソースのチューニングについては、本チューニングガイドの以下も参照してください。

－ “J2EEのチューニング”－“IJServerのチューニング”－“5.2.4 トランザクションアイソレーションレベル”

4)Servletコンテナ情報

項番	評価方法	対応/処置
1	アクティブ数が、長時間トータルに近い値を示している。	[CPUに余裕がある場合] ・ Servletコンテナの同時処理数が小さすぎる可能性があります。

項番	評価方法	対応/処置
		Servletコンテナの同時処理数を増やしてください。ただし、JavaVMのヒープ使用量に余裕がない場合にそのまま同時処理数を増やすとOutOfMemoryErrorが発生する可能性があります。この場合は、JavaVMのヒープ使用量の最大値を増やすなどの対処を同時に行ってください。 ・ プロセス内の要因によってスレッド間の競合が発生している可能性があります。 ワークユニットのプロセス多重度を増やしてください。
		[CPUに余裕がない場合] ・ サーバの処理能力を超えている可能性があります。 サーバの増設または、より性能の良いサーバへのリプレースを検討してください。

※ Servletコンテナのチューニングについては、本チューニングガイドの以下も参照してください。

― “J2EEのチューニング”－“5.3 Servletコンテナのチューニング”

5)EJBコンテナ情報

項番	評価方法	対応/処置
1	Message-driven Bean最大同時処理数とMessage-driven Bean最大スレッドプールサイズが近い値となっている。	運用を継続するとスレッド獲得待ちとなる可能性があります。指定しているMessage-driven Beanの同時処理数が適切か見直してください。
2	Message-driven Bean最小スレッドプールサイズとMessage-driven Bean最小スレッドプールサイズの差が大きく、Message-driven Beanアイドルタイムアウト発生回数が多い。	Message-driven Beanのスレッド生成処理でアプリケーションの処理性能が劣化している可能性があります。 指定しているMessage-driven Beanの最小同時処理数が適切か見直してください。
3	Message-driven Bean最大アイドルスレッド数が大きい。	指定しているMessage-driven Beanの最小同時処理数が大きいため、メモリ資源を無駄に消費している可能性があります。 指定しているMessage-driven Beanの最小同時処理数が適切か見直してください。

※ EJBコンテナのチューニングについては、本チューニングガイドの以下も参照してください。

― “J2EEのチューニング”－“5.4 EJBコンテナのチューニング”

5.2 IJServerのチューニング

IJServerをチューニングする時に考慮するポイントは以下です。ここに記述されたチューニングは、ServletコンテナとEJBコンテナの両方に有効です。

- プロセス多重度
- JavaVMのヒープ領域サイズ
- ガーベジコレクション発生回数
- トランザクションアイソレーションレベル
- JDBCのコネクション
- Statementキャッシュ機能
- モニタリング情報
- IPCOM連携時の注意事項

5.2.1 プロセス多重度

1つのIJServerを、複数のプロセスで起動できます。これにより、負荷を分散できます。
IJServerのプロセス多重度は、Interstage管理コンソールのワークユニット設定またはisj2eeadminコマンドで指定できます。詳細については、Interstage管理コンソールのヘルプを参照してください。isj2eeadminコマンドについては、“リファレンスマニュアル(コマンド編)”の“isj2eeadmin”を参照してください。

5.2.2 JavaVMのヒープ領域サイズ

Interstage管理コンソールまたはisj2eeadminコマンドを使用して、ワークユニット設定のJavaVMオプションを指定することで、IJServerが動作するJavaVMのパラメタを変更して動作させることができます。
パラメタを変更して、JavaVMヒープ領域サイズなどを変更できます。

JDK 1.4の場合における最大ヒープ領域サイズの例を次に示します。
最大ヒープ領域のサイズの省略値は、JavaVMによって異なりますので、JDKのドキュメントを参照してください。
java.lang.OutOfMemoryErrorが多発する場合には、本定義項目で、JavaVMの最大ヒープ領域を増加させてください。

JavaVMの最大ヒープ領域を1024メガバイトとする場合の設定

```
-Xmx1024m
```

なお、Interstageではヒープ領域の問題を警告メッセージで通知する、予兆監視機能を提供しています。
警告メッセージが出力された場合、そのまま業務を継続すると、メモリ不足やレスポンス低下などの問題が発生する可能性があります。これらの問題を解決するために、警告メッセージに記載されている不足リソースの情報を元に、チューニングを実施してください。
JavaVMで問題となる異常の原因は、ヒープ領域またはPerm領域の不足です。これを回避するために、現在の上限値を20%増加させて運用を再開します。それでも警告が出力される場合は、上限値を更に20%増加させて、警告が出力されなくなるまで繰り返しチューニングを実施してください。チューニングを繰り返し行い、警告メッセージが出力されない状態にすることで、安定稼動するシステムを構築することができます。
予兆監視機能については、“運用ガイド(基本編)”を参照してください。

5.2.3 ガーベジコレクション発生回数

IJServerでは、JavaのRMI機能による自動ガーベジコレクションが1時間隔(デフォルトの場合)で動作します。
RMI機能による自動ガーベジコレクションの発生間隔は、Interstage管理コンソールの“ワークユニット名” > [環境設定]タブ > [ワークユニット設定]のJavaVMオプションに、以下を設定します。

“-Dsun.rmi.dgc.client.gcInterval=発生間隔”および
“-Dsun.rmi.dgc.server.gcInterval=発生間隔”

発生間隔: ミリ秒単位で数値を指定します。デフォルト値は、3600000です。

isj2eeadminコマンドを使用して、設定することもできます。isj2eeadminコマンドについては、“リファレンスマニュアル(コマンド編)”の“isj2eeadmin”を参照してください。

なお、RMI機能による自動ガーベジコレクションの発生間隔をチューニングしても、ガーベジコレクション発生回数が削減されない場合、JavaVMのヒープ領域サイズが不足している可能性がありますので、JavaVMのヒープ領域サイズのチューニングを行うことで削減される場合があります。“5.2.2 JavaVMのヒープ領域サイズ”を参照してください。

5.2.4 トランザクションアイソレーションレベル

EJBアプリケーションからデータベースにアクセスする場合、EJBアプリケーションの実行多重度を上げるには、トランザクションアイソレーションレベル(以降、アイソレーションレベルと呼びます)を考慮する必要があります。アイソレーションレベルとは、データベースに対する排他整合性水準のことです。

使用できるアイソレーションレベルを以下に示します。アイソレーションレベルの詳細は、使用するデータベースのマニュアルを参照してください。

- Transaction-read-committed
- Transaction-read-uncommitted
- Transaction-repeatable-read
- Transaction-serializable

アイソレーションレベルの設定は、UserTransaction.begin()メソッドを発行してから、UserTransaction.commit()メソッドまたはUserTransaction.rollback()メソッドを発行するまでの間有効です。

■設定方法

アイソレーションレベルは、Interstage管理コンソールまたはisj2eeadminコマンドで設定します。設定方法の詳細については、Interstage管理コンソールのヘルプを参照してください。isj2eeadminコマンドについては、“リファレンスマニュアル(コマンド編)”の“isj2eeadmin”を参照してください。

DBMSにOracleを使用している場合

「ORA-8177:このトランザクションのアクセスを逐次化できません。」というエラーは、トランザクションアイソレーションレベルにTransaction-serializableが設定されているにもかかわらず、複数のユーザが同時に同一の表を更新した場合など、トランザクションのシリアル化を保障できない場合に出力され、ユーザにその旨を伝えています。

トランザクションアイソレーションレベルにTransaction-serializableを設定して、エラー「ORA-8177」が発生した場合は、アプリケーション側で単に「異常終了」と判断するのではなく、トランザクションのロールバック後に「リトライ」させるなどの対処が必要になります。

なお、トランザクションアイソレーションレベルがTransaction-read-committed(Oracleのデフォルト)の場合は、「ORA-8177」エラーが発生することはありません。特にTransaction-serializableの設定が必須ではない場合、Transaction-read-committedを設定することによって、同時実行性が向上し、「ORA-8177」エラーも発生しなくなります。

5.2.5 JDBCのコネクション

ここでは、JDBCのコネクションの以下について説明します。

- コネクションプーリングの種別
- コネクションプーリングのチューニングパラメタ

InterstageのJNDIサービスプロバイダから取得したJDBCデータソースを使用した場合、JDBCのコネクションはプーリングされて再利用されます。

■コネクションプーリングの種別

コネクションプーリングには、以下の2種類があります。

- Interstageでコネクションプーリング
  Interstageでコネクションのプーリング制御を行うため、Interstage管理コンソールでコネクションプーリングの詳細設定ができます。プーリングされている情報を、Interstage管理コンソールのモニタ機能で参照できます。Oracleの分散トランザクションを使用する場合もInterstageでコネクションプーリングを行います。
- JDBCドライバでコネクションプーリング
  JDBCドライバでコネクションのプーリング制御を行うため、使用する各JDBCドライバの機能を使用して、コネクションプーリングの設定を行います。設定方法の詳細は、JDBCドライバのマニュアルを参照してください。
  JDBCドライバ側でプーリング制御を行うため、Interstage管理コンソールで参照可能なJDBCデータソースのモニタ情報は、Oracleの場合は「コネクションプール情報」と「アプリケーションからのコネクション確立情報」、Oracle以外のデータベースは「アプリケーションからのコネクション確立情報」のみです。詳細はInterstage管理コンソールのヘルプを参照してください。

コネクションプーリングの機能概要については、“J2EE ユーザーズガイド”の“JDBC(データベース)のコネクション”を参照してください。

以下に、各データベースとコネクションプーリングの対応について記載します。
サポートしているデータベースは、使用しているプラットフォームによって異なります。“使用上の注意”の“アプリケーション実行時に必要なソフトウェア”を参照してください。

DB種別	コネクションプーリング
Oracle	データソースの種類に“Interstageのコネクションプーリングを使用する”を選択している場合、Interstageでコネクションプーリングを行います。 “Oracleのコネクションプーリングを使用する”を選択している場合、JDBCドライバがコネクションプーリングを行います。
Symfoware	データソースの種類に“Interstageのコネクションプーリングを使用する”を選択している場合、Interstageでコネクションプーリングを行います。 “Symfowareのコネクションプーリングを使用する”を選択している場合、JDBCドライバでコネクションプーリングを行います。
SQL Server	データソースの種類にかかわらずInterstageでコネクションプーリングを行います。
Windows Linux PostgreSQL	データソースの種類に“Interstageでコネクションプーリングを行う”を選択している場合に、Interstageでコネクションプーリングを行います。 “PostgreSQLでコネクションプーリングを行う”の場合は、JDBCドライバがコネクションプーリングを行います。
汎用定義	データソースクラスがjava.sql.ConnectionPoolDataSourceインタフェースを実装している場合、Interstageでコネクションプーリングを行います。 上記以外の場合、JDBCドライバがコネクションプーリングを行うかどうかはJDBCドライバの実装に依存します。

■コネクションプーリングのチューニングパラメタ

IJServerの環境設定に指定可能なDBコネクション設定はデータベースタイプおよびデータソース種別によって設定が有効となる項目が異なります。以下の表を参照してください。

	Oracle			Symfoware		SQL Server	PostgreSQL		汎用定義	
	Interstage (*1)	JDBC (*2)	XA (*3)	Interstage (*1)	JDBC (*2)	Interstage (*1)	Interstage (*1)	JDBC (*2)	Interstage (*4)	JDBC (*5)
トランザクションアイソレーションレベル	○	○	○	○	○	○	○	○	○	○
事前コネクト数	○	○	○	○	○	○	○	○	○	○
最大コネクション数	○	○	○	○	×	○	○	×	○	×
コネクションタイムアウト	○	○	○	○	×	○	○	×	○	×
アイドルタイムアウト	○	○	○	○	×	○	○	×	○	×
コネクション使用監視時間	○	○	○	○	○	○	○	○	○	○
Statementキャッシュサイズ	×	○	×	○	×	×	×	×	×	×
Statement自動クローズ	×	×	×	○	×	×	×	×	×	×
通信待ち時間	○	○	○	○	○	○	○	○	○	○
異常時の再接続	○	×	×	○	×	○	○	×	×	×

	Oracle			Symfoware		SQL Server	PostgreSQL		汎用定義	
	Interstage (*1)	JDBC (*2)	XA (*3)	Interstage (*1)	JDBC (*2)	Interstage (*1)	Interstage (*1)	JDBC (*2)	Interstage (*4)	JDBC (*5)
インターバル時間	○	×	×	○	×	○	○	×	×	×
リトライ回数	○	×	×	○	×	○	○	×	×	×

○:有効 ×:無効

*1) Interstageのコネクションプーリングを使用する場合
*2) JDBCのコネクションプーリングを使用する場合
*3) 分散トランザクションを使用する場合
*4) データソースクラスがjava.sql.ConnectionPoolDataSourceインタフェースを実装している場合
*5) データソースクラスがjava.sql.ConnectionPoolDataSourceインタフェースを実装していない場合(JDBCでコネクションプーリングを行うかはJDBCドライバの実装に依存します。)

以下に、Interstage管理コンソールまたはisj2eeadminコマンドを使用して設定可能なパラメタを示します。
パラメタは、データソースごとにIJServerの環境設定で設定します。

パラメタ	説明	設定値
事前コネクト数 (注1)(注5)	運用で必要なコネクションを、起動時にあらかじめ取得することにより、初回疎通時から2回目以降と同等の処理速度が得られます。 データベースタイプがOracle、データソースの種類が「Oracleのコネクションプーリングを使用する」で、暗黙的接続キャッシュの接続キャッシュ・プロパティにInitialLimitを指定し、かつ事前コネクト数が1以上の場合、事前コネクト数とInitialLimitの値が大きい方が有効となります。(注4)	最大値:2147483647 最小値:0 初期設定値:0
最大コネクション数	最大コネクション数を抑止することにより、メモリ資源を抑止することが可能です。 最大コネクション数すべてをJ2EEアプリケーションが使用している状態で接続リクエストがあった場合、コンテナは“コネクションタイムアウト”における設定時間の期間内で、プールにコネクションが返却されるのを待ちます。プールにコネクションが返却された場合にはそのコネクションを使用し、返却されなかった場合にはSQLExceptionを返却します。 データベースタイプがOracle、データソースの種類が「Oracleのコネクションプーリングを使用する」の場合、以下の注意があります。 ・本項目は暗黙的接続キャッシュの接続キャッシュ・プロパティMaxLimitの設定となります。 ・JDBCリソース定義時にキャッシュ・プロパティMaxLimitを指定しても値は有効になりません。本項目で指定してください。	最大値:2147483647 最小値:1 初期設定値:64
コネクションタイムアウト	最大コネクション数分のコネクションすべてがJ2EEアプリケーションで使用中の状態で、コネクションの接続要求が来た場合に、プールにコネクションが返却されるのを待つ時間を指定します。時間が超過してもコネクションが返却されなかった場合は、SQLExceptionが返却されます。0を指定した場合、コネクション待ち状態になるとすぐにSQLExceptionが返却されます。 データベースタイプがOracle、データソースの種類が「Oracleのコネクションプーリングを使用する」の場合、以下の注意があります。 ・本項目は暗黙的接続キャッシュの接続キャッシュ・プロパティConnectionWaitTimeoutの設定となります。 ・JDBCリソース定義時にキャッシュ・プロパティConnectionWaitTimeoutを指定しても値は有効になりません。本項目で指定してください。	最大値:2147483647 最小値:0 初期設定値:5 (単位:秒)

パラメタ	説明	設定値
アイドルタイムアウト	使用されていないコネクションをタイムアウトで破棄することにより、無駄なメモリ資源を解放することができます。 ただし、事前コネクトで接続されたコネクションはアイドルタイムアウトの対象となりません。 データベースタイプがOracle、データソースの種類が「Oracleのコネクションプーリングを使用する」の場合、以下の注意が必要です。 ・本項目は、暗黙的接続キャッシュの接続キャッシュ・プロパティ InactivityTimeout の設定となります。 ・JDBCリソース定義時にキャッシュ・プロパティ InactivityTimeout を指定しても値は有効になりません。本項目で指定してください。 ・タイムアウトを確認する時間間隔は、暗黙的接続キャッシュの接続キャッシュ・プロパティ PropertyCheckInterval の設定値となります。データソースの環境設定に PropertyCheckInterval を定義してください。プロパティの詳細はOracleのマニュアルを参照してください。 ・Oracleのコネクションプーリングを使用する場合、アイドルタイムアウト発生時のコネクション破棄はOracleのJDBCドライバで行われます。また事前コネクトで接続されたコネクションもアイドルタイムアウトの対象となります。	最大値:2147483647 最小値:0 初期設定値:600 (単位:秒)
コネクション使用監視時間 (注2)	トランザクション開始前にアプリケーションが獲得したコネクションに対してクローズするまでの使用時間を監視します。 コネクションの使用時間が指定した時間を超過してもクローズされない場合、警告メッセージがコンテナログやシステムログに出力されます。 0を指定した場合、コネクション使用時間の監視は行いません。	最大値:2147483647 最小値:0 初期設定値:60 (単位:分)
タイムアウトしたコネクションはクローズする	コネクション使用監視時間使用時(コネクション使用監視時間が1以上)にタイムアウトしたコネクションを自動回収する場合はONにします。OFFにした場合、タイムアウトしたコネクションは自動回収されません。	・する ・しない (デフォルト)
Statementキャッシュサイズ	Statementのキャッシュサイズを指定します。 アプリケーションが実行したStatementをクローズせずにキャッシュするサイズを指定します。 0を指定した場合、Statementのキャッシュは行いません。Statementキャッシュ機能の詳細は“5.2.6 Statementキャッシュ機能”を参照してください。	最大値:32000 最小値:0 初期設定値:10
Statement自動クローズ	ステートメントキャッシュ機能の使用時(Statementキャッシュサイズが1以上の場合)に、ステートメントのクローズをJDBCドライバが自動的に行うかどうかを指定します。	・する ・しない (デフォルト)
通信待ち時間 (注2)	以下のSQL文実行の開始から終了までの時間を監視します。 ・java.sql.Statement ー execute(String) ー execute(String, int) ー execute(String, int[]) ー execute(String, String[]) ー executeBatch() ー executeQuery(String)	最大値:2147483647 最小値:0 初期設定値:400 (単位:秒)

パラメタ	説明	設定値
	ー executeUpdate(String) ー executeUpdate(String, int) ー executeUpdate(String, int[]) ー executeUpdate(String, String[]) ・ java.sql.PreparedStatement ー execute() ー executeQuery() ー executeUpdate() SQL文の実行時間が一定時間超過しても復帰しなかった場合、警告メッセージがコンテナログやシステムログに出力されます。 0を指定した場合、通信待ち時間の監視は行いません。	
ログにSQL文を出力する	通信待ち時間使用時(通信待ち時間が1以上)に、ログに出力する警告メッセージにSQL文を出力する場合はONにします。OFFにした場合、SQL文は出力されません。	・ ON ・ OFF(デフォルト)
異常時の再接続	JDBCコネクションの自動再接続機能(注3)を使用するかどうかを指定します。 自動再接続機能を使用する場合、プーリングされているJDBCのコネクションが使用可能なコネクションであるかを判定し、使用できないコネクションの場合には自動的にDBMSに再接続します。	・ する(デフォルト) ・ しない
インターバル時間	JDBCコネクションの自動再接続機能(注3)において、プーリングされているJDBCのコネクションが使用できない場合、またはDBMSへの接続に失敗した場合、再度接続を行うまでのインターバル時間を指定します。 異常時の再接続を[する]にした場合のみ、指定した値が有効になります。	最大値:2147483647 最小値:1 初期設定値:10 (単位:秒)
リトライ回数	JDBCコネクションの自動再接続機能(注3)において、プーリングされているJDBCのコネクションが使用できない場合、またはDBMSへの接続に失敗した場合、再度接続を試みる回数を指定します。 異常時の再接続を[する]にした場合のみ、指定した値が有効になります。	最大値:2147483647 最小値:1 初期設定値:10 (単位:回数)

注1)

CMP2.0のEJBアプリケーションを配備したIJServerを起動する場合、起動時にDBMSの識別子長の最大値をチェックします。このため、事前コネクト機能を使用しない場合にも1コネクションだけDBMSへ接続します。
Interstageでコネクションをプーリングする場合には、起動処理完了後にコネクションが切断されます。

注2)

コネクション使用監視時間、および通信待ち時間のタイムアウト値は以下の計算式を参考に設定してください。

```
アプリケーション最大処理時間 ＞ コネクション使用監視時間 ＞ 通信待ち時間
```

注3)

Symfowareの場合は、Connection Managerの機能を使用することで、データベースサーバのダウンおよび通信回線の異常発生時にも同等の運用を行うことが可能です。Connection Managerの詳細については、Symfoware Serverのマニュアル“Connection Managerユーザーズガイド”を参照してください。

注4)

Oracleの場合、事前コネクトの獲得タイミングは以下の表のようになります。

Oracleのデータソース種別および設定		事前コネクト取得タイミング
Interstageでコネクションプーリング	DBコネクション設定の“事前コネクト数”に1以上の値を設定	IJServer起動時
	上記設定をしない場合	事前にコネクションは確立されません
Oracleでコネクションプーリング	DBコネクション設定の“事前コネクト数”に1以上の値を設定 ただし暗黙的接続キャッシュのプロパティに“InitialLimit”も設定されている場合は双方の値の大きい方の値分確立されます	IJServer起動時
	DBコネクション設定の“事前コネクト数”を設定せず(デフォルト値:0)、暗黙的接続キャッシュのプロパティの“InitialLimit”に1以上の値を設定	初回getConnection時
	上記以外の場合	事前にコネクションは確立されません

注5)

事前コネクト数を1以上に設定した場合、IJServer起動時にDBMSへ接続しますので対象のDBMSは起動されている必要があります。

5.2.6 Statementキャッシュ機能

以下の場合に、Statementキャッシュ機能を使用できます。

- データベースタイプが“Oracle”で、データソースの種類が“Oracleのコネクションプーリングを使用する”の場合
  この場合、Oracle10g以降のStatementキャッシュ機能が使用可能になります。
  なお、“Oracleのコネクションプーリングを使用する”データソースでは分散トランザクションが使用できないため、本機能は分散トランザクション環境では使用できません。
- データベースタイプが“Symfoware”で、データソースの種類が“Interstageのコネクションプーリングを使用する”の場合

Statementをキャッシュすることによって、SQL文の解析、作成によるオーバーヘッドの軽減や、データベースとの通信回数を削減する効果があります。

キャッシュ対象となるStatementは以下です。

- Connection.prepareStatement(String)メソッドで取得したPreparedStatementオブジェクト
- Connection.prepareStatement(String, int, int)メソッドで取得したPreparedStatementオブジェクト
- Connection.prepareStatement(String, int)メソッドで取得したPreparedStatementオブジェクト
- Connection.prepareStatement(String, int[])メソッドで取得したPreparedStatementオブジェクト
- Connection.prepareStatement(String, int, int, int)メソッドで取得したPreparedStatementオブジェクト
- Connection.prepareStatement(String, String[])メソッドで取得したPreparedStatementオブジェクト
- Connection.prepareCall(String)メソッドで取得したCallableStatementオブジェクト
- Connection.prepareCall(String, int, int)メソッドで取得したCallableStatementオブジェクト
- Connection.prepareCall(String, int, int, int)メソッドで取得したCallableStatementオブジェクト

■チューニング方法

Statementキャッシュサイズ

Statementキャッシュサイズは、使用するデータソースに対して、Interstage管理コンソールまたはisj2eeadminコマンドで設定します。
Interstage管理コンソールでは、[IJServer] > [環境設定] > [DBコネクション設定] > [Statementキャッシュサイズ]で設定します。[Statementキャッシュサイズ]に0を設定した場合は、Statementキャッシュは行なわれません。
isj2eeadminコマンドの詳細については、“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。
なお、JDBCリソース定義画面の接続オプションでStatementキャッシュサイズを設定し、同時にDBコネクション設定でキャッシュサイズを指定した場合は、DBコネクション設定で行ったキャッシュサイズの値が有効になります。

設定の指針

Statementはコネクション単位でキャッシュされ、コネクションは実行する同一IJServerプロセス内のアプリケーション全体で共通に使用されます。
アプリケーションで発行されるStatement数がStatementキャッシュサイズより大きくなるとキャッシュから削除されるStatementが増加します。アプリケーションが要求するStatementがキャッシュから削除されていると、Statementの再作成のために、データベースとの通信、SQLステートメント文の解析によるオーバーヘッドが発生します。
キャッシュからのStatement削除回数を削減するため、StatementキャッシュサイズをIJServerプロセスで発行されるStatementの総数もしくはそれに近い値を設定することを推奨します。ただし、Statementのキャッシングはマシン資源を消費します。マシンスペックを考慮の上でStatementのキャッシュサイズの設定を行う必要があります。

また、アプリケーションサーバ(コンテナ)側でStatementを発行するCMPアプリケーションの場合は、次の数の合計値をアプリケーションで発行するStatementの代わりにキャッシュサイズへ追加します。

- CMP1.1 Entity Beanの場合
  - Bean単位で算出するStatement
    - 4つ(挿入、削除、更新、検索(プライマリキー))
    - 1件検索、複数件検索メソッド数
- CMP2.0 Entity Beanの場合
  - Bean単位で算出するStatement
    - 4つ(挿入、削除、更新、検索(プライマリキー))
    - deployment descriptorに定義したEJB QLクエリ数
  - relationship単位で算出するStatement
    - CMP2.0 Entity Bean間の単方向、かつ、1:1のrelationship数
    - CMP2.0 Entity Bean間の双方向、かつ、1:1のrelationship数×2
    - CMP2.0 Entity Bean間の単方向、かつ、1:多、多:多のrelationship数×3
    - CMP2.0 Entity Bean間の双方向、かつ、1:多、多:多のrelationship数×4

OPEN_CURSORSの設定(Oracleを使用する場合)

Statementキャッシュ機能を使用する場合、コネクションのクローズ時にStatementは破棄されず、1コネクションで同時に発行しているStatementは増加します。このため、“1コネクション(トランザクション)で同時に発行可能なStatement数の上限値(以降OPEN_CURSORS/デフォルト:50)”を超える可能性があります。Statement数がOPEN_CURSORSを超えた場合、SQLExceptionが発生します。
このような場合は、OPEN_CURSORSが、Statementキャッシュサイズ以上となるように設定してください。OPEN_CURSORSの設定方法は、Oracleのマニュアルを参照して下さい。

■キャッシュされたStatementの削除契機

キャッシュされたStatementは、以下の契機で削除されます。また、実行したSQL文の数が、「Statementキャッシュサイズ」に設定した値に達した場合、それ以降実行するSQL文の扱いについては使用するJDBCドライバの仕様に依存します。(JDBCドライバの仕様によりキャッシュされたStatementが削除される場合があります。)JDBCドライバの仕様については、各JDBCドライバのマニュアルを参照してください。

- JDBCプーリングコネクションの時間監視機能
  Statementは接続インスタンス(コネクション)ごとにキャッシングされるため、物理接続が接続キャッシュ内にアイドル状態で存続できる最大期間を超過したコネクションオブジェクトを開放する時に、そのコネクションでキャッシュされていたStatementオブジェクトも削除されます。

- コネクションの物理的開放
  Statementは接続インスタンス(コネクション)ごとにキャッシングされるため、コネクションがプーリングされている時にデータベースダウンなどによりコネクションが無効となった場合、そのコネクションを開放する際にキャッシュされていたStatementオブジェクトも削除されます。

5.2.7 モニタリング情報

Interstage管理コンソールでは、運用中のIJServerの稼動情報を表示します。出力される情報により、性能のボトルネックの検出や、性能チューニングの効果の確認ができます。
以下の情報が出力されます。出力される情報の詳細は、Interstage管理コンソールのヘルプを参照してください。

- JavaVM情報
- Servletコンテナ情報
- データソース情報
- トランザクション情報
- キュー情報
- Message Driven Beanスレッド情報

なお、V9.0以降のIJServerを使用している場合は、性能情報を一定間隔でログファイルへ出力することができます。詳細は、“5.1 J2EEモニタロギング機能”を参照してください。

5.2.8 IPCOM連携時の注意事項

IPCOMを利用して、IJServerとWebサーバを分離して運用するシステムの負荷分散をする場合、故障監視用のコネクション(スレッド)が必要となります。
この場合、IJServerの同時処理数を設定するときは実際の同時処理数に監視用の数を考慮してください。設定は以下になります。

設定する同時処理数 ＝ 実際の同時処理数 ＋ 1(監視用)

なお、IPCOMのWebアクセラレーション機能を使用する場合は、実際の同時処理数を設定してください(監視用の数は加算しないでください)。Webアクセラレーション機能については、IPCOMのマニュアルを参照してください。

5.3 Servletコンテナのチューニング

Servletコンテナをチューニングする時に考慮するポイントは以下です。

- 同時処理数
- 接続数
- タイムアウト

■同時処理数

同時処理数およびプロセス多重度を増やすと、Webアプリケーションの同時実行多重度を増やせます。
同時処理数を増やすと、1プロセスあたりの実行多重度を増やせますが、同時処理数が増えることによる負荷や資源の増加により効果はみられない可能性があります。通常はデフォルト値以下で運用することを推奨しています。
アプリケーションから呼び出すEJBの同時処理数や、JDBCのコネクション数にあわせてチューニングしてください。
同時処理数は、Interstage管理コンソールのServletコンテナ設定で指定します。また、isj2eeadminコマンドを使用して設定することもできます。

同時処理数については、以下が設定できます。

- 初期値(増分値)
- 待機中(アイドル状態)の処理スレッドの最大値
- 最大値

処理スレッドの処理完了後、待機中となり使用されていない処理スレッドは、1分間隔の監視によって、待機中の最大値をこえている分が解放されます。
これにより、一時的に負荷が高くなった場合でも、負荷が下がればサーバ資源を解放し節約することができます。

■接続数

接続数に関する設定項目としては以下があります。

- WebサーバコネクタのServletコンテナへの最大接続数

  Interstage管理コンソールで指定します。

  ー Webサーバとワークユニットを同一マシンで運用する場合
  ワークユニットの環境設定のWebサーバコネクタ(コネクタ)設定

  ー Webサーバとワークユニットを同一マシンで運用しない場合
  Webサーバコネクタの環境設定

  また、isj2eeadminコマンドを使用して設定することもできます。

- Servletコンテナの接続数(最大接続数)

  Interstage管理コンソールのServletコンテナ設定で指定します。
  また、isj2eeadminコマンドを使用して設定することもできます。

上記項目の値を増やすとServletコンテナが受付けるクライアントからのリクエストの数を増やせます。
運用時に一時的に負荷が高くなりServletコンテナの同時処理数を超えるリクエストを受付けることが想定される場合は、Servletコンテナの同時処理数を抑えてServletコンテナの接続数(最大接続数)に大きな値を設定することで、サーバ全体のレスポンスの悪化を防ぐことができます。

接続数は、運用するシステムの要求の違いによって以下のように設定します。

- Servletコンテナの負荷が高く処理できないリクエストを時間がかかってもかまわないので、正常に処理させたい場合

項目	設定値
Servletコンテナの最大接続数	リクエストを処理できるだけの十分に大きな値を設定
WebサーバコネクタのServletコンテナへの接続数の制限	なし(無制限)

- Servletコンテナの負荷が高く処理できないリクエストは長時間待たせるのではなく、エラーとしてクライアントに通知したい場合

項目	設定値
Servletコンテナの最大接続数	リクエストを処理できるだけの十分に大きな値を設定
WebサーバコネクタのServletコンテナへの接続数の制限	Servletコンテナが処理可能な数を設定

- Servletコンテナへの接続数がServletコンテナの同時処理数の最大値を超えた場合、超えた分の接続はKeepAliveされませんので、Servletコンテナの同時処理数の最大値を超えない場合と比較して多くのSocketを消費します。
- 接続数の最大値に設定された値はServletコンテナが使用するSocketのbacklogに設定されます。
  Socketのbacklogに設定した値はOSによって有効となる範囲が決まっていますので、接続数の最大値に設定した値がすべて有効となるわけではありません。
  Socketのbacklogの有効範囲については各OSのドキュメントを参照してください。
  Windows (R)の場合、200より大きな値を設定しても有効になりません。

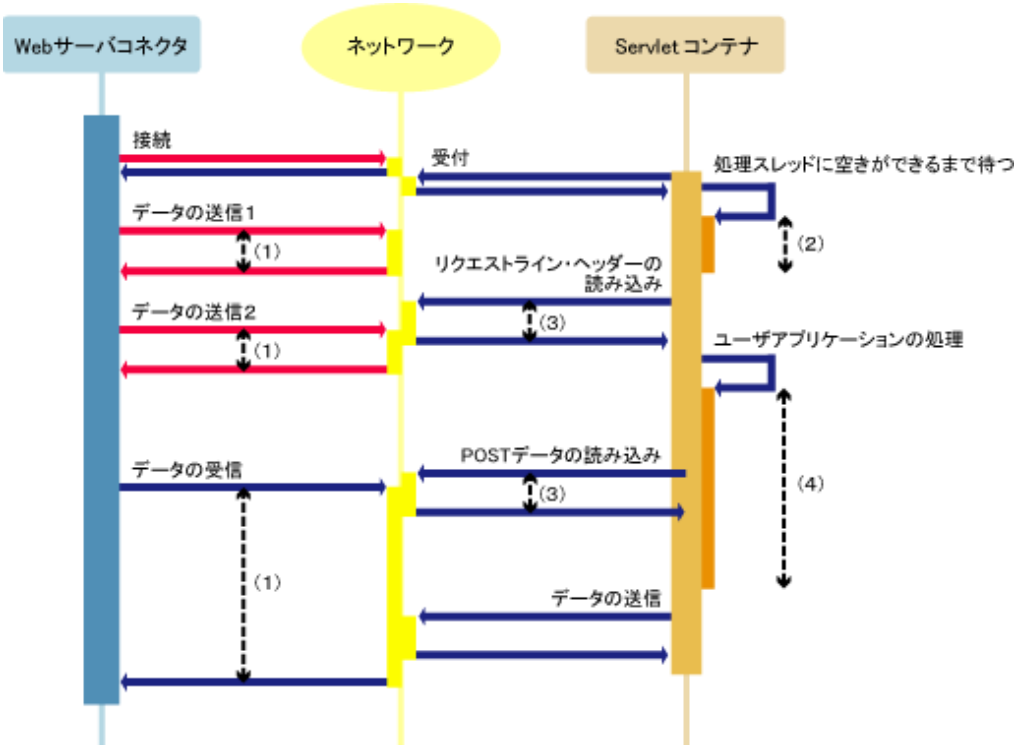
■タイムアウト

タイムアウトの監視項目

タイムアウトに関しては、以下が監視されています。

監視項目	意味
(1)	Webサーバコネクタが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時間
(2)	Servletコンテナに接続されてからリクエストの処理が開始されるまでの待ち時間(値を指定することはできません)
(3)	ServletコンテナがWebサーバコネクタからのデータパケットを受信するときに待機する時間
(4)	ユーザアプリケーションの処理時間

それぞれの項目が監視されるタイミングを下図に示します。



タイムアウトの設定項目

タイムアウトに関する設定項目には以下があります。

設定項目	意味
(a)	Webサーバコネクタの送受信タイムアウト Interstage管理コンソールの以下の項目で指定します。 ・ Webサーバとワークユニットを同一マシンで運用する場合 [ワークユニット] > [Webサーバコネクタ(コネクタ)設定] > [送受信タイムアウト] ・ Webサーバとワークユニットを同一マシンで運用しない場合 [Webサーバコネクタ] > [環境設定] > [送受信タイムアウト]
(b)	Servletコンテナのタイムアウト Interstage管理コンソールの[Servletコンテナ設定] > [タイムアウト]で指定します。
(c)	ワークユニットのアプリケーション最大処理時間 Interstage管理コンソールの[ワークユニット設定] > [アプリケーション最大処理時間]で指定します。

上記は、isj2eeadminコマンドを使用して設定することもできます。

- 以下の現象が頻繁に発生する場合、(a)、(c)の値を大きくすることで回避できる可能性があります。(a)、(c)の値を大きくしても回避できない場合は、サーバの処理能力を超えている可能性があります。サーバの増設または、より性能の良いサーバへのリプレースを検討してください。
  - コンテナログにIJServer32113が出力される
  - Webサーバコネクタのログに IJServer12035、IJServer12044が出力される
- 以下の現象が頻繁に発生する場合、(b)の値を大きくすることで回避できる可能性があります。回避できない場合はネットワークやクライアントアプリケーションに問題がある可能性があります。発生している問題を解決してください。
  - リクエストデータの読み込みでサーブレットにjava.net.SocketTimeoutExceptionがスローされる
  - Webサーバコネクタのログに IJServer12026、IJServer12027、IJServer12034、IJServer12036が出力される

タイムアウトの設定方法

タイムアウトの設定値は、以下の関係を満たすように値を設定します。

```
設定項目(a) ＞ 監視項目(2) ＋ 監視項目(3) ＋ 監視項目(4)
設定項目(b) ＞ 監視項目(3)
設定項目(c) ＞ 監視項目(4)
```

- 以下の条件に該当する場合、IJServerが強制停止されます。
  - 設定項目(b) ＞ 設定項目(c)に設定した場合、かつ、
  - [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]を“プロセスを強制停止する”に設定した場合、かつ
  - タイムアウトの監視項目の図における“データの送信2”が設定項目(c)を超えても完了しなかった場合
- 以下に設定した場合、Webブラウザには Webサーバコネクタのタイムアウトが通知されます。
  設定項目(a) ＜ 監視項目(2) ＋ 監視項目(3) ＋ 監視項目(4)
  しかし、Servletコンテナは Webサーバコネクタがタイムアウトしたことを検出することはできないため、アプリケーションを継続して実行し、監視項目(3)の時間内に処理が完了した場合は、Servletコンテナ側で異常を検出することができません。
- 監視項目(4)の時間にはPOSTデータの読み込み時間も含まれます。

5.4 EJBコンテナのチューニング

EJBコンテナをチューニングする時に考慮するポイントは以下です。

- 同時処理数
- Session Bean
- Entity Bean
- Message-driven Bean
- ローカル呼出し機能
- JNDI

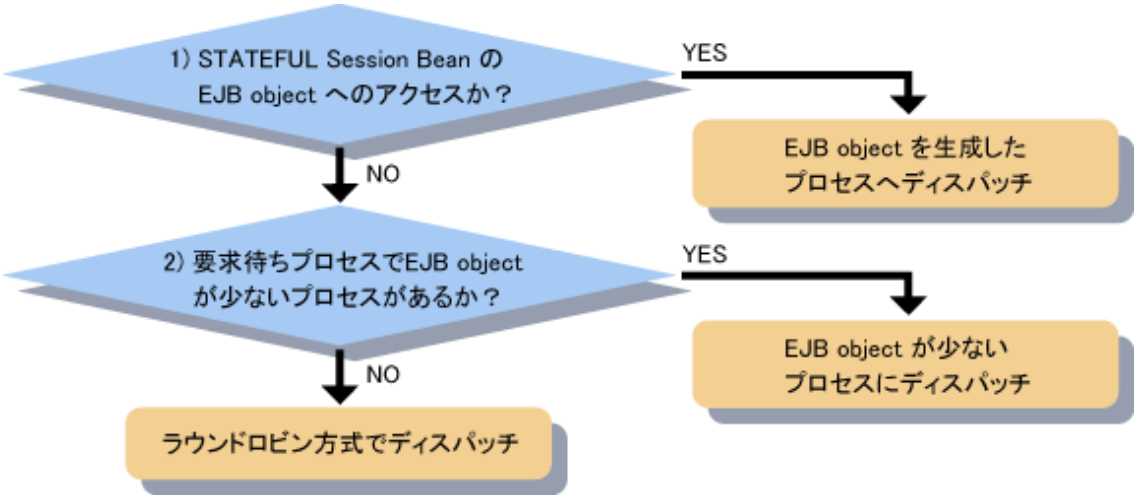
5.4.1 同時処理数

同時処理数は、1プロセスあたりの実行多重度です。IJServerで同時に処理できるクライアントのリクエスト数は、プロセス多重度と同時処理数で決まります。例えばプロセス多重度を2、同時処理数を16に設定すると、合計で2×16＝32の処理が同時に処理できます。クライアントからのリクエストが、処理できるリクエスト数を超えた場合は、キューイングされます。

- 同時処理数を変更することで、1プロセスあたりの実行多重度を増やすことができますが、同時処理数が増えることによる使用資源(CPU使用率、メモリ量など)の増加により、効果が見られない場合があります。
- 資源が不足している場合には、後から実行されたリクエストをキューイングすることにより、レスポンスの安定を図ります。

■プロセスへの処理振り分け

EJBコンテナのプロセス多重度を2以上に設定した場合、各プロセスへのディスパッチ論理は以下のように行われます。



1. STATEFUL Session Beanの場合、createメソッドで取得したEJB objectへのアクセスは、必ず同一のプロセスにディスパッチされます(同一のプロセスにディスパッチされるため、前回の処理をセションの情報として保持し、次回のアクセス時に参照できます)。
2. 1以外の場合には、EJB objectの総数が少ないプロセスに処理を振り分けます。
   各プロセスのEJB objectが同数の場合には、ラウンドロビン方式(処理要求を順番に割り振る方式)でディスパッチするプロセスを決定します。

各EJBアプリケーション種別で、EJB objectが生成されるタイミングと削除されるタイミングを以下に記載します。
EJBコンテナは、各プロセスに生成されたEJB objectの総数を判断して、適切なプロセスに処理を割り振ります。

EJBアプリケーション種別	EJB objectが生成されるタイミング	EJB objectが削除されるタイミング
STATEFUL Session Bean	createメソッド実行時	・removeメソッド実行時 ・無通信監視タイムアウト時 ・IJServer停止時
STATELESS Session Bean	IJServer起動時に1つ生成	IJServer停止時
Entity Bean	・createメソッド実行時 ・finderメソッド実行時	・removeメソッド実行時 ・Entity BeanのEJB objectタイムアウト時 ・IJServer停止時
Message-driven Bean	－ (Message-driven BeanはRemoteインタフェースがないため、EJB objectは生成されません。)	－

同時処理数には、以下のように最小値と最大値の設定ができます。
通常はデフォルト値で運用することを推奨します。

定義項目	デフォルト値	説明
最小値	16	IJServer起動直後から、EJBコンテナで同時処理可能なスレッド数です。クライアントのリクエスト数が最小値を超えた場合は、自動的に拡張して動作します。
最大値	64	拡張可能な同時処理数の最大値です。クライアントのリクエスト数が最大値を超えた場合は、リクエストはキューイングされます。

5.4.2 Session Bean

Session Beanのチューニングについて、説明します。
リソースを効果的に利用するために、Session Beanにおいては以下の設定を行います。

- STATEFULとSTATELESSの選択
- STATEFUL Session Beanの無通信監視
- Session Beanのcreateメソッド実行数の上限
- STATELESS Session Beanの起動時インスタンス生成

■STATEFULとSTATELESSの選択

STATELESS Session Beanを使用することで、メモリ資源やオブジェクトの生成回数が抑止されるため、処理性能が向上します。
STATEFULとSTATELESSには以下の違いがありますので、用途に合わせて使い分けてください。

	STATEFUL	STATELESS
対話状態	createからremoveまで同一のオブジェクトにアクセスするため、クライアントとの対話状態を保持することができる。	クライアントとの対話状態を持たないため、クライアントが情報を保持する必要がある。

	STATEFUL	STATELESS
トランザクション	Session Beanのsynchronization機能を使って、トランザクションとの同期が可能。	1メソッド内でトランザクションを完了させる必要がある。
性能	クライアントごとにEJB objectを生成するため、STATELESSに比べてメモリ使用量が多い。	インスタンスやEJB objectを使い回すため、メモリ使用量を抑止することができる。また、オブジェクトの生成回数が抑止される。

■STATEFUL Session Beanの無通信監視

createメソッドで作成したオブジェクトに対してremoveメソッドを実行せずに終了した場合、残存するオブジェクトを自動的に消去するため、不要なメモリの増加を防ぐことができます。
デフォルト値は、30(分)です。

■Session Beanのcreateメソッド実行数の上限

createメソッド実行数を高負荷の実行環境に合わせて変更できます。
以下の計算式で算出された値が、デフォルト値の1024を超過する場合には、Session Beanのcreateメソッド実行数の上限値を変更してください。createメソッド実行数はInterstage管理コンソールで設定します。

(クライアントアプリケーションのプロセス数)×(1プロセス数あたりの実行スレッド数の平均)

createメソッドで作成したオブジェクトをremoveメソッドで削除しなかった場合、無通信監視機能でタイムアウトが発生するまでオブジェクトが残存するため、メモリの使用量が多くなる可能性があります。create数の上限値は適切な値に設定してください。

設定範囲

種類	設定値
初期設定値	1024
最小値	1
最大値	2147483647

■STATELESS Session Beanの起動時インスタンス生成

IJServer起動時に、STATELESS Session Beanのインスタンスを作成しておくことでアクセス時のインスタンス作成時間が省略され、処理性能が向上します。
設定は、Interstage管理コンソールの[システム] > [ワークユニット] > “ワークユニット名” > “EJBモジュール” > “EJBアプリケーション名” > [アプリケーション環境定義]タブの[Interstage拡張情報]で行います。デフォルトは“しない”です。

設定範囲

種類	設定値
最小値(デフォルト)	0
最大値	2147483647

- 初期起動インスタンス数を増やした場合、使用するヒープ量が増えるため注意してください。
- 初期起動インスタンス数を1以上に指定した場合、setSessionContext、またはejbCreateメソッドから以下の操作、またはアクセスはできません。

  ー javax.ejb.TimerServiceメソッド

- javax.ejb.Timerメソッド
- 他のEJBアプリケーションへのアクセス
- リソースマネージャ(データベースなど)へのアクセス

・ 作成されたインスタンスは削除されないため、実行時の同時処理数以上に初期起動インスタンス数を設定しないよう、注意してください。
実行時の同時処理数の上限は、以下の式で算出してください。

- IJServerタイプが“WebアプリケーションとEJBアプリケーションを同一JavaVMで運用”の場合
Servletコンテナ設定の同時処理数 ＋ Message-driven Beanの同時処理数の最大値
- IJServerタイプが“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”または“EJBアプリケーションのみ運用”の場合
EJBコンテナ設定のIIOP呼び出しの同時処理数の最大値 ＋ Message-driven Beanの同時処理数の最大値

5.4.3 Entity Bean

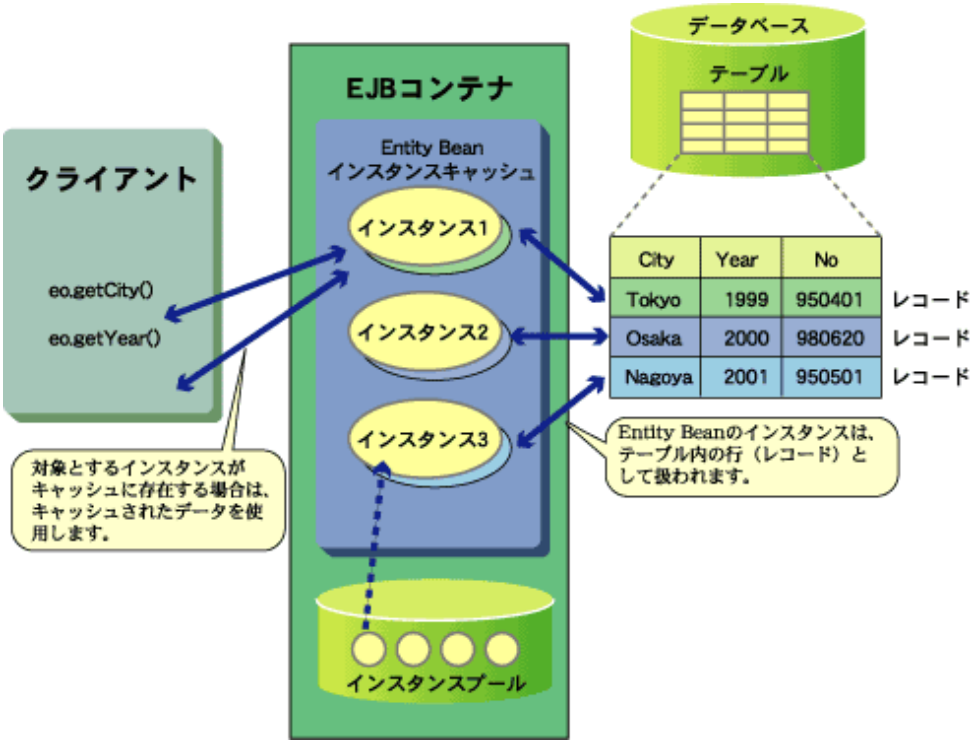
Entity Beanのチューニングについて、説明します。

■Entity Bean呼出しの注意

Entity Beanは、レコードの情報を取得するためにメソッドを頻繁に実行します。このため、プロセス外からEntity Beanを呼び出すとIIOP通信が頻繁に発生することにより、性能が劣化します。
Entity Beanは同一IJServerのアプリケーションから呼び出すことを推奨します。

■インスタンス数

インスタンスはトランザクション内でキャッシュされます。インスタンスプールにインスタンスが存在しない場合、同一トランザクション内で使用したインスタンスのレコードデータをデータベースに反映し、そのインスタンスを別のレコードデータを格納するインスタンスとして再利用します。インスタンスが頻繁に再利用されるとデータベースにアクセスする回数が増加することによって性能に影響がありますので、性能を考慮してインスタンス数を設定してください。



インスタンス数はデータベースの検索レコード数とクライアントの同時接続数に関係します。効果的な値としては、通常検索されるレコード数の1.25倍の値を設定します。
以下に設定値の計算式を表します。

Entityインスタンス数 ＝ a × b × 1.25 (安全率) a: 1度に検索されるレコード数 b: 1プロセスで同時にアクセスするクライアント数

例) 10クライアントが同時に100件を検索した場合
インスタンス数 ＝ 10 × 100 × 1.25 ＝ 1250

注) インスタンス数を増やすと、使用メモリが増えるので注意してください。

■インスタンス管理モード

Entity Beanのインスタンス管理モードによってデータベースの処理をチューニングすることができます。
以下の表に、それぞれのインスタンス管理モードと最適な処理について示します。

管理モード	最適な処理
ReadWrite	検索を実行する時、またオンラインのデータベースを更新する時に有効
ReadOnly	更新されない主要なデータを検索(参照)する時に有効
Sequential	大量のデータを一括処理する時に有効

■CMPデータのstream転送

CMPで使用するEntity Beanで、JDBCのサイズ制限以上のデータを扱う場合は、Interstage管理コンソールの以下で設定してください。
[システム] > [ワークユニット] > “ワークユニット名” > “EJBモジュール” > “EJBアプリケーション名” > [アプリケーション環境定義]タブを開き、CMFマッピング定義の「CMPデータのstream転送」に“する”を設定します。また、ejbdefexport/ejbdefimportコマンドを使用して

設定することもできます。詳細は“J2EE ユーザーズガイド”の“運用コマンドを使用してカスタマイズする方法”を参照してください。デフォルトは“しない”です。

■CMP2.0の複数件検索高速化

CMP2.0 Entity Beanで複数件finderメソッドを実行した場合に、レコードのデータを一度にすべてロードするオプションを提供しています。
全データを、DBMSからロードするような処理の場合にも、高速にDBMSからデータをロードできます。
詳細は、“J2EEユーザーズガイド”の“Entity Beanの最適化処理”の“CMP2.0の複数件検索時の高速化”を参照してください。

■CMP1.1の複数レコードの一括更新

CMP1.1の複数レコードの一括更新の設定は、Interstage管理コンソールの[システム] > [ワークユニット] > “ワークユニット名” > “EJBモジュール” > “EJBアプリケーション名” > [アプリケーション環境定義]タブの[Interstage拡張情報]で設定します。「複数レコードの一括更新」で“する”を選択してください。デフォルトは“する”です。
また、ejbdefexport/ejbdefimportコマンドを使用して設定することもできます。詳細は“J2EE ユーザーズガイド”の“運用コマンドを使用してカスタマイズする方法”を参照してください。

設定を行ったCMP1.1 Entity Beanはデータベースの更新を行う時に、以下のAPIを利用して一括更新を行います。

- java.sql.PreparedStatementクラスのaddBatchメソッド

- 使用するデータベース、および、JDBCドライバがJDBC2.0バッチ更新機能をサポートしている必要があります。
  JDBC2.0バッチ更新機能が未サポートのデータベース、および、JDBCドライバを使用した場合は、通常のデータベースへの更新処理を行います。
- 「インスタンス管理モード」が“Read-Only”の場合、データの更新自体が行われないため、「複数レコードの一括更新」の設定は無効となります。
- 「複数レコードの一括更新」は、分散トランザクションを使用しない場合にだけ有効です。分散トランザクションを使用する場合、通常のデータベースへの更新処理を行います。
- CMP1.1 Entity Beanでstream転送を行う場合、データの更新に失敗する場合があります。その場合は「複数レコードの一括更新」を“しない”に変更してください。

■CMP1.1のbyte配列更新判定

CMP1.1 Entity Beanでbyte配列を使用する場合に、byte配列のデータが更新されているかの判定方法を設定できます。
設定は、Interstage管理コンソールの[システム] > [環境設定] の 詳細設定から[EJBサービス詳細設定]タブで「CMP1.1のbyte配列更新判定」で行います。また、isj2eeadminコマンドを使用して設定することもできます。

5.4.4 Message-driven Bean

Message-driven Beanのインスタンス数を設定することにより、同時にメッセージ処理を行うことができます。
目安としては、キューにメッセージが蓄積されない程度のインスタンス数を設定します。ただし、クライアント数とMessage-driven Bean処理時間に依存するため、環境に合わせ試験運用を行い、調整後に設定してください。

■Message-driven Beanのスレッドプール

Point-To-Point、または、Publish/Subscriberメッセージングモデルの Message-driven Beanのメッセージ受信時にプールしたスレッドを利用してメッセージ受信処理を行います。
このプールされたスレッド数のことをMessage-driven Beanの同時処理数と呼びます。


注意

以下の条件に該当するMessage-driven Beanの場合、スレッド多重でのメッセージ受信をサポートしていないため、スレッドプールは使用されません。

	条件A	条件B
メッセージングモデル	Publish/Subscriber	Publish/Subscriber
トランザクション管理種別	Container	Bean
トランザクション属性	Required	－

スレッドプールの単位

IJServerプロセスに1つ作成し、プールしているスレッドはIJServerに配備されているすべてのMessage-driven Beanのメッセージ受信処理で共有されます。

スレッドの作成について

IJServerプロセス起動時にMessage-driven Beanの最小同時処理数のスレッドを作成してプールに格納します。
メッセージが配信されると、メッセージの処理開始前にプールを検索し、プールにスレッドが存在しない場合、および現在利用されているスレッド数がMessage-driven Beanの最大同時処理数より小さい場合に、スレッドを作成して受信処理を行います。
受信完了後、作成したスレッドは削除されることなくプールに返却されます。


注意

プールにスレッドが存在しない場合で、現在利用されているスレッド数がMessage-driven Beanの最大同時処理数を超過している場合は、現在利用されるスレッドがプールに返却されるまで待機し、プールに返却されたスレッドを取得して受信処理を行います。

スレッドの削除について

プーリングされたスレッドは、アイドルタイムアウトによって破棄されます。スレッドがプールに返却されてから指定した時間を超過しても使用されないスレッドが存在する場合には、そのスレッドを破棄します。
ただし、初期起動スレッド数分はプーリングされ続けますので破棄対象外となります。また、IJServerプロセスが停止するとスレッドも削除されます。

スレッドプールのチューニング

Interstage管理コンソールまたはisj2eeadminコマンドで、スレッドプールに対して以下のチューニングを行うことができます。

Message-driven Beanの同時処理数

EJBを運用するプロセス上で、Message-driven Beanで同時に処理できる処理数(スレッド数)の最大値と最小値を指定します。
また、使用されずにプーリングされたスレッドを解放するまでの、タイムアウト時間を指定します。

設定は、Interstage管理コンソールの[ワークユニット] > “ワークユニット名” > [環境設定] > [詳細設定] > [EJBコンテナ設定]で行います。
または、isj2eeadminコマンドで設定することも可能です。詳細は、“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。

項目	説明	値の説明
最小	IJServerプロセスの起動時に生成するスレッド数を指定します。 生成されたスレッドは、プーリングされてアイドルタイムアウトの対象外となります。	0～2147483647の整数値を指定できます。 デフォルト値は0です。

項目	説明	値の説明
最大	起動時に最小値だけスレッドが生成され、必要に応じて最大値までスレッドが拡張されます。	1～2147483647の整数値を指定できます。 デフォルト値は64です。
アイドルタイムアウト	アイドルタイムアウト値を指定します。 スレッドが、プールに返却されてから指定した時間を超過しても、使用されないスレッドを破棄します。ただし、最小同時処理数分は破棄対象外となります。 0を指定した場合には、タイムアウトしません。	0～2147483647の整数値を指定できます。 デフォルト値は600(秒)です。

Message-driven Beanの同時処理数をチューニングする場合、以下を参考にしてください。

- 断続的に処理を実行するような場合には、常に同時に実行されるスレッド数を最小値に設定することで、スレッドの作成、破棄の処理が軽減されるためにCPU使用率が軽減されて性能が向上します。
  Message-driven Beanの処理が頻繁に行われない場合には、最小値を小さく設定することで、使用メモリ量を抑えることができます。
- Interstage管理コンソールのモニタ情報を参照して、アイドルタイムアウト回数が断続的に増加している場合、スレッドの生成と破棄が頻繁に発生しています。
  アイドルタイムアウトの設定値が小さい可能性がありますので、アイドルタイムアウトの設定値を大きくすることを検討してください。
  ただし、アイドルタイムアウトの設定値を大きく設定すると生成したスレッドがプールに滞留する時間が長くなるため、メモリの使用量も増加します。Javaのヒープ量やシステム資源も考慮して設定してください。
- 膨大な処理要求を受け付ける可能性がある場合、最大値を小さく指定して同時処理数を抑制することで、CPU負荷を軽減できます。
  CPUの使用率が上限値に対して低い場合には、最大値を大きく設定することで、最大値に指定した値までMessage-driven Beanで同時処理できます。
- 各Message-driven Beanの初期起動スレッド数の合計値が、Message-driven Beanの同時処理数の最大値を上回っている場合、Message-driven Beanの同時処理数の最大値以上のメッセージがIJServerプロセスに配信される可能性があります。
  その場合、使用中のスレッドがプールに返却されるまで待機します。プールに返却されるまで待機させない場合には、各Message-driven Beanの初期起動インスタンス数の合計値をInterstage管理コンソール、またはisj2eeadminコマンドで、Message-driven Beanの同時処理数の最大値に指定してください。

■異常時のメッセージ退避機能

Point-To-Pointメッセージングモデルでかつ不揮発化チャネルを利用する場合、リトライ回数を超過してもメッセージ受信を繰り返す可能性があります。
イベントチャネルのイベントデータをメモリにキャッシュする数を、イベントチャネルに蓄積できるイベントデータの最大値よりも大きく設定してください。
設定はイベントサービス運用コマンドを使用して行います。詳細は“リファレンスマニュアル(コマンド編)”の“essetcnf”および“essetcnfchnl”を参照してください。

5.4.5 ローカル呼出し機能

IJServerに配備されたEJBアプリケーションが、同一のEJBコンテナ内のEJBアプリケーションからだけで呼び出される場合、ローカル呼出し機能を使用できます。
ローカル呼出し機能を使用することによって、ネットワークを介してEJBアプリケーションを呼び出されることを考慮したIJServerの処理が軽減されるため、通常よりも更に性能良く動作します。
本機能は、IJServerが“EJBアプリケーションのみ運用する場合”、または、“WebアプリケーションとEJBアプリケーションを別JavaVMで運用する場合”にだけ有効です。

ローカル呼出し機能の設定はInterstage管理コンソールの[ワークユニット] > [IJServer名] > [EJBアプリケーション] > [アプリケーション環境定義] > [Interstage拡張情報]の“ローカル呼出し”で行います。設定の詳細についてはInterstage管理コンソールのヘルプを参照してください。

- 「ローカル呼出し」を“する”に設定したEJBアプリケーションをプロセス外から呼び出した場合、“CORBA OBJ_ADAPTER”のエラーが発生します。
- Entity Beanをプロセス外から呼び出す場合には、「ローカル呼出し」を“しない”に変更してください。また、ローカル呼出しをしない場合には、EJB objectをタイマで削除してください。EJB objectのタイマ削除については“J2EE ユーザーズガイド”の“EJB objectのタイマ削除機能”を参照してください。

5.4.6 JNDI

■オブジェクトの使い回し

lookupメソッドで取得したオブジェクトは、EJBアプリケーション内で保持して使い回すことにより、lookupメソッドの実行回数を軽減できます。

■deployment descriptor定義

deployment descriptorファイルにオブジェクトの情報を定義することができます。

- 定義した場合、定義された情報を元に各ネーミングサービスからIJServer起動時にオブジェクトを取得してメモリ上に保持するため、処理性能が向上します。
- 定義しない場合、アプリケーションでlookupメソッドを実行した時にオブジェクトが各ネーミングサービスに存在しないか確認します。

deployment descriptorファイルにオブジェクトの情報を定義することを推奨しますが、すでに開発済みのEARファイルを使用する場合などでdeployment descriptorファイルの編集ができない場合、以下のオプションを使用することで、ネーミングサービスへのアクセス回数を軽減できます。

項目	設定内容
定義ファイル格納ディレクトリ	Windows Interstageインストールディレクトリ¥ejb¥etc Solaris Linux /opt/FJSVejb/etc
定義ファイル	FJEJBconfig.properties
指定するキー名	“LookupCache”(固定)
指定する値	・ 1:同一オブジェクトへの2回目以降のlookupメソッド実行時にはメモリに保持した情報を返却します。 ・ 1以外: lookupメソッドを実行する度に、各ネーミングサービスにアクセスします(デフォルト)。

本オプションを使用すると、lookupメソッド実行時に各ネーミングサービスから取得した情報をメモリ上に保持するため、同一のオブジェクトに対する2回目以降のlookupメソッドの処理性能が向上します。
なお、以下のオブジェクトを取得する場合には、本オプションを設定してIJServerを運用してください。

- 他のIJServerに配備されたEJBアプリケーションのHomeオブジェクト
- データソース
- JMSコネクションファクトリ
- JMS Destination

5.5 LDAPサーバとしての、ディレクトリサービスのチューニング

Interstage ディレクトリサービスは、以下の製品で使用することができます。

Windows Server(R) for Itanium-based Systems/RHEL-AS4(IPF)/RHEL5(IPF)の場合

- Interstage Application Server Enterprise Edition

上記以外のオペレーティングシステムの場合

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard-J Edition

Interstage ディレクトリサービスのチューニングについては、以下を参照してください。

- “Interstage ディレクトリサービスのシステム資源の設定”
- “ディレクトリサービス運用ガイド”の“検索のチューニング”

第6章 業務構成管理機能のチューニング

業務構成管理機能が管理するリポジトリの設定値をチューニングすることで、最適な状態での運用を行うことが可能となります。ここでは、業務構成管理機能が管理するリポジトリのチューニングについて説明します。

6.1 リポジトリのチューニング

リポジトリのチューニングについては、以下のマニュアルを参照してください。

・ “運用ガイド(基本編)”の“業務構成管理機能”

第7章 JDK/JREのチューニング

CやC++では、メモリに割り当てた領域は、不要になった時点でプログラマが明示的に解放する必要があります。しかし、この解放処理に漏れやミスがあると、メモリリーク、プログラム停止および暴走が発生するデメリットがあります。

Javaでは、ガーベジコレクション(GC)を導入したことにより、プログラマがメモリ管理において作業の負担を軽減できるようになりました。その反面、GCが発生するたびにJavaアプリケーションが一時停止するため、パフォーマンス劣化の要因になることがあります。また、Java独特のメモリリークも存在します。

Javaでは、スレッドを管理しやすいため、大量のスレッドを生成する傾向があります。このため、スタックが大量に生成されて、メモリ不足になることもあります。

以上から、多くの場合において、JDK/JREのチューニングは、スタックサイズ、または、GCの発生頻度とその処理時間を調整することになります。本章では、Javaアプリケーションのチューニングにあたって、必要な基礎知識やチューニング方法などを説明します。

7.1 基礎知識

本項では、JDK/JREのチューニングを行う際に必要な知識を説明します。

7.1.1 JDK関連のドキュメント

■JDKドキュメント

本製品に搭載しているJDK 1.4、5.0、6のドキュメントは、それぞれ次のURLにあります。

- JDK 1.4: http://java.sun.com/j2se/1.4/ja/docs/ja/index.html
- JDK 5.0: http://java.sun.com/j2se/1.5.0/ja/docs/ja/index.html
- JDK 6 : http://java.sun.com/javase/ja/6/docs/ja/index.html

上記のドキュメントは、オンラインマニュアルCDの次の箇所にも格納しています。

- JDK 1.4: ¥ApplicationServer¥javadocs¥jdk14-jdkdocs.zip
- JDK 5.0: ¥ApplicationServer¥javadocs¥jdk5-jdkdocs.zip
- JDK 6 : ¥ApplicationServer¥javadocs¥jdk6-jdkdocs.zip

また、javaコマンドおよびJava VMのオプションには、チューニングに関するオプションがあります。本章では、これらのオプションを随所で紹介します。
各オプションの詳細は、次を参照してください。なお、本マニュアル内で具体的に説明していないJava VMのチューニングに関するオプションは、FJVMではサポートしておりません。

- javaコマンドのオプション
  JDKドキュメンテーションの「SDKのツール」の「java」
- Java HotSpot VM Options
  http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp

■Java プラットフォーム移行ガイド

JDK 1.3で開発されたJavaプログラムを、JDK 5.0に移行する際の非互換や問題点については、「Java プラットフォーム移行ガイド (バージョン 1.3 から 5.0 へ)」を参照してください。

このガイドは、オンラインマニュアルCDの次の箇所に格納しています。

- ¥ApplicationServer¥JavaplatformMigration¥jm_white_paper_r6a-jp.pdf

JDK5.0からJDK6への移行については、以下のURLを参照してください。

- http://www.oracle.com/technetwork/java/javase/adoptionguide-137484.html

7.1.2 Java VM

■製品搭載のJava VM

JDK/JREには、Javaのバイトコードを解釈・実行するエンジン部であるJava仮想マシン(**Java VM**)があります。
“表7.1 Java VMの種類と特徴”に、**Java VM**の種類と特徴を示します。

表7.1 Java VMの種類と特徴

Java VMの名称	Java VMの特徴
Java HotSpot Client VM	アプリケーション起動時間を短縮し、メモリ消費を抑制するように設計されたクライアント環境向けの**Java VM**です。 富士通版**Java HotSpot Client VM**では、富士通独自技術によるトラブルシューティングに関する機能強化などを追加実装しています。
Java HotSpot Server VM (FJVM)	アプリケーション起動時間の短縮などよりも、安定性および、スループットの向上を考慮して設計されたサーバ環境向けの**Java VM**です。 富士通版**Java HotSpot Server VM**では、富士通独自技術による性能改善やトラブルシューティングに関する機能強化などを追加実装しています。 Interstageでデフォルトとなる富士通版**Java VM**であることから、このJava VMを特に**FJVM**と呼びます。 **FJVM**の詳細は、“7.2 FJVM”を参照してください。

Interstageでは、**Java HotSpot Client VM**と**FJVM**(=**Java HotSpot Server VM**)を搭載しています。
デフォルトの**Java VM**は、**FJVM**です。これは、javaコマンドのオプションに、“-server”または“-fjvm”を指定することと同義です。**Java HotSpot Client VM**を使用したい場合は、オプションに“-client”を指定してください。

Windows (64ビット) Linux (64ビット)
実行モードが64ビットモードのJDK/JREでは**FJVM**のみ使用可能です。**Java HotSpot Client VM**は搭載していません。

また、Windows Server(R) for Itanium-based Systems、Linux for Itaniumの富士通版JDK/JRE 6(Java VMを含む)は提供していません。

エルゴノミクス機能によるJava VMの自動選択機能

富士通版JDK/JRE 5.0、6では、エルゴノミクス機能によるJava VMの自動選択機能(マシンのCPU数や物理メモリ量などに応じて、使用するJava VMを自動的に選択する機能)は無効になっています。

■Java VM関係の資料

Java VMの詳細は、JDKドキュメンテーションの次を参照してください。

- JDK 1.4の場合: [新機能の概要] > [Java 仮想マシン]
- JDK 5.0の場合: [J2SEの概要] > [Java 仮想マシン] および [新機能と拡張機能] > [仮想マシン]
- JDK 6の場合 : [Java SE 6 概要] > [Java 仮想マシン] および
- [新機能と拡張機能] > [Java SE 6のテクノロジーの主な変更]
- > [VM(Java仮想マシン)]

他にも、**Java VM**に関連する資料があります。

- The Java Virtual Machine Specification (Java VM仕様書)
  http://java.sun.com/docs/books/vmspec/
- Java HotSpot VM Options
  http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp
- Tuning Garbage Collection with the 1.4.2 Java Virtual Machine
  http://java.sun.com/docs/hotspot/gc1.4.2/index.html
- Tuning Garbage Collection with the 5.0 Java Virtual Machine
  http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html

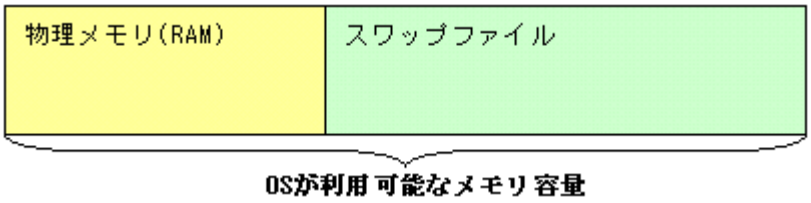
7.1.3 仮想メモリと仮想アドレス空間

Javaアプリケーションを開発・運用するにあたり、OSのメモリ管理の概要を知っておく必要があります。本節では、OSの一般的なメモリ管理技法の1つである**仮想アドレス空間**を説明します。
なお、**仮想アドレス空間**の具体的なアーキテクチャはOSごとに異なりますが、本節では各OSに共通する内容を説明します。

■仮想メモリ

OSは、物理メモリ(RAM)だけでなくスワップファイルを活用することにより、多くのメモリ領域を使用することができます。このテクノロジーを**仮想メモリ**といいます。**仮想メモリ**の容量は、RAMとスワップファイルのサイズの合計になります。

図1 OSが利用可能なメモリ容量



ただし、ハードディスクへのアクセスはRAMより低速なので、メモリのスワッピングは性能に大きく影響を与えますので、注意が必要になります。

■仮想アドレス空間

OS上でプログラムを起動すると、OSがプログラムを実行・管理する単位としてのプロセスが生成されます。同様にして、Javaアプリケーションを起動すると、Javaプロセスが生成されます。

OS上で生成されたプロセスには、**仮想アドレス空間**が割り当てられます。**仮想アドレス空間**は、それぞれのプロセスで独立したものであり、あるプロセスから別のプロセスの**仮想アドレス空間**にアクセスすることはできません。マシンに積んでいる物理メモリ(RAM)のサイズとは関係なく、**仮想アドレス空間**のサイズは、常に一定です。たとえば、32ビットアーキテクチャのOSの場合、物理メモリ(RAM)のサイズに依存せず、**仮想アドレス空間**のサイズは常に4GB(2の32乗バイト)です。

このため、大量の**仮想メモリ**を用意しても、1つのプロセスで使用できるメモリ量の上限は**仮想アドレス空間**の上限になりますから、プロセスが**仮想アドレス空間**の大きさを越えるメモリ量を必要とする場合は、メモリ不足の状態になります。

また、大量の**仮想メモリ**を用意しても、OS上で大量のプロセスが動作していて**仮想メモリ**を大量に消費している状態であれば、各プロセスに割り当てられた**仮想アドレス空間**を使い切っていなくても、メモリ不足の状態になる場合があります。

■ユーザ空間

プロセスが持つ**仮想アドレス空間**のうち、実際にプロセスが使用できる空間を、**ユーザ空間**といいます。**ユーザ空間**には、プログラムの実体(Windows(R)でJavaアプリケーションを実行する場合は、java.exeなど)がコピーされるだけでなく、スタックやヒープなどのさまざまなセグメントがあります。更に**ユーザ空間**は、実行するプログラムだけでなく、そのプログラムを実行させるためのOS側のプログラムなどでも使用します。Javaプロセスの**ユーザ空間**の場合には、前述の各セグメントの他に、Javaオブジェクトを格納するセグメント(=Javaヒープ)があります。このため、Javaアプリケーションをチューニングする際の対象となるJavaヒープのサイズの上限値は、**ユーザ空間**のサイズよりも少なくなります。

Javaアプリケーションのチューニングを実施する際は、仮想メモリの容量やプロセスの使用状況など、システムの状態を考慮する必要があります。なおOSの種類によって、**ユーザ空間**として使用できる上限値が異なるため、注意が必要です。

7.1.4 スタック

ここでは、**スタック**について簡単に説明します。

プログラムがスレッドを生成すると、OSがそのスレッドに対して**スタック**と呼ばれるメモリ領域をそのスレッドの終了時まで自動的に割り当てます。**スタック**は、スレッド上で実行されるメソッドや関数が使用するローカル変数などの一時的なデータを格納するための作業域として使用されます。

スレッド上では多くのメソッドや関数が入れ子状態で呼び出されるので、これらのメソッドや関数が使用する一時的な作業域を管理するために、OSは**フレーム**と呼ばれる区切りで個々の作業域をスタック上に積み上げることで管理しています。(メソッドや関数が呼び出されるごとに、これらが使用する作業域を**フレーム**という区切りでスタック上に積み上げ、**フレーム**内のデータは呼び出したメソッドや関数からの復帰時に破棄されます。)

このため、あるメソッドを再帰的に無限に呼び出すなどしてメソッドを深く呼び出した場合や、非常に大きなサイズのローカル変数などを使用するメソッドを呼び出した場合などには、**スタック**域の枯渇により、スタック上に**フレーム**を積み上げることができなくなり、スタックオーバーフローが発生する場合があります。

Javaアプリケーションの場合、通常、スタックオーバーフローが発生すると、java.lang.StackOverflowErrorがスローされます。ただし、Javaプロセス中のネイティブモジュール実行時の処理でスタックオーバーフローが発生した場合には、java.lang.StackOverflowErrorはスローされません。なお、FJVMを使用しているJavaプロセス中のネイティブモジュール実行時にスタックオーバーフローが発生した場合は、"スタックオーバーフロー検出時のメッセージ出力機能"によってスタックオーバーフローの発生を検出できる場合があります。

ユーザ空間のメモリ不足によるスレッド生成エラー

スタックは、プロセス内のユーザ空間から割り当てられます。
そして、ユーザ空間のメモリ不足により**スタック**などを割り当てることができなった場合には、スレッド生成処理でエラーが発生します。
Javaプロセスの場合、Javaヒープなどのサイズを大きく確保すると、その分だけ**スタック**として割当て可能な領域が減少するため、Javaプロセス内で生成可能なスレッドの個数も減少します。
このため、Javaプロセス内のスレッド生成処理がエラーとなる要因の1つとして、Javaヒープなどのサイズを大きく確保したことによるユーザ空間のメモリ不足が考えられます。

Javaプロセスの消滅 Windows

Windows(R)では、Javaプロセス内でスタックオーバーフローが発生した場合、システム状態やプログラム状態によっては、OSからFJVMやワトソン博士に制御が渡らず、痕跡を残さずにJavaプロセスが消滅する場合があります。
なお、ワトソン博士の説明は、"7.3.4 クラッシュダンプ・コアダンプ"を参照してください。

スタックサイズについて Windows **(64ビット)**

Windows Server(R) for Itanium-based Systemsでは、スタックとして以下の2種類の領域を使用します。

- メモリ用スタック域
- レジスタ用スタック域

そのためOSは、プログラムがスレッド生成時に指定したスタックサイズの値を用いて、その両方のスタック域を新たに生成するスレッドへ割り当てます。
つまりスレッドに対しては、スタックサイズとして指定された値の2倍の大きさがスタックとして割り当てられます。
たとえば、スタックサイズを2MBとしてスレッドを起動した場合には、そのスレッドに対するスタック域として、両スタック域の合計である4MBが割り当てられます。

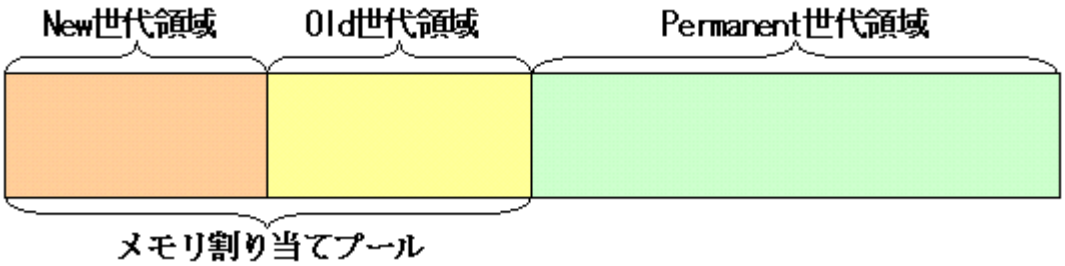
7.1.5 Javaヒープとガーベジコレクション

ここでは、**Javaヒープ**と**ガーベジコレクション(GC)**を説明します。

Javaヒープは、Javaプロセス内に存在するJavaオブジェクトを格納するための領域です。
Javaヒープは、**New世代領域**、**Old世代領域**および**Permanent世代領域**に大別され、Java VMが管理・制御します。なお、**New世代領域**と**Old世代領域**は、**メモリ割り当てプール**という形で各領域を合わせて一括的に管理・制御します。

Java VMは、Javaアプリケーションの実行時に、Javaオブジェクトを**Javaヒープ**の各領域に格納します。**Javaヒープ**の空き容量がなくなった場合は、java.lang.OutOfMemoryErrorがスローされます。
また、不要となったJavaオブジェクトは**GC**により回収され、**Javaヒープ**の空き領域が増加します。なお、**Old世代領域**や**Permanent世代領域**に存在する不要となったJavaオブジェクトを回収する**GC**処理を、特に**FullGC**処理といいます。

図1 Javaヒープの構造



- **New世代領域とOld世代領域(メモリ割り当てプール)**

  New世代領域と**Old世代領域**は、インスタンスや配列などのJavaオブジェクトを管理する領域です。
  New世代領域は寿命の短いJavaオブジェクトを管理します。通常、Javaアプリケーションで要求されたオブジェクトは、**New世代領域**に生成されます。一定期間**New世代領域**で生存したJavaオブジェクトは、**GC**処理によって**Old世代領域**に移動されます。そして、**Old世代領域**に存在する不要となったJavaオブジェクトは、**FullGC**処理によって回収されます。領域がNew世代とOld世代に分かれるのは、世代別に**GC**処理を実行するためです。
  なお、**メモリ割り当てプール**全体のサイズの初期値および最大値は、それぞれ、“-Xms”オプションおよび“-Xmx”オプションで指定することができます。

- **Permanent世代領域**

  Permanent世代領域は、Javaのクラス、メソッドや定数など、永続的に参照される静的オブジェクトを管理する領域です。
  Permanent世代領域に存在する不要となったオブジェクトは、**FullGC**処理によって回収されます。
  なお、**Permanent世代領域**の大きさの初期値および最大値は、それぞれ、“-XX:PermSize”オプションおよび“-XX:MaxPermSize”オプションで指定することができます。

ユーザ空間

Javaヒープは、Javaプロセスのユーザ空間内に割り当てられます。
また指定された最大値まで**Javaヒープ**が利用できる環境にするため、Java VM起動時に、**メモリ割り当てプール**および**Permanent世代領域**を、各最大値の大きさで連続領域としてリザーブします(同一プロセス内の他の処理から使用できないようにします)。
このため、**Javaヒープ**の最大値を大きく設定すると、その分だけスタックなど他の処理に割り当てられるメモリ領域が減少します。

なおJava VM起動時に指定された**Javaヒープ**の大きさが利用できない場合、Java VMは以下のメッセージを出力し、Javaプロセスを終了させます。

```
Error occurred during initialization of VM
Could not reserve enough space for object heap
```

このメッセージが出力された場合は、**Javaヒープ**を小さくするチューニングを行ってください。

またSolaris用およびLinux用のJava VMにおいて、Java VM起動時またはJavaアプリケーション実行中に「ユーザ空間不足」または「仮想メモリ不足」が発生した場合、Java VMは以下のメッセージを出力し、Javaプロセスを終了させます。

Java VM起動時の場合:

```
制御名: mmap failed: errno=エラー情報, 制御情報....
Error occurred during initialization of VM
mmap failure
```

Javaアプリケーション実行中の場合:

```
制御名: mmap failed: errno=エラー情報, 制御情報....
(上記メッセージに続いて、java.lang.OutOfMemoryErrorメッセージが出力される場合があります。)
```

制御名　：メモリ不足が発生した際のJava VMの制御名
エラー情報:メモリ不足が発生した際のJava VMのエラー情報
制御情報 ：メモリ不足が発生した際のJava VMの制御情報

ユーザ空間が不足している場合は、**Javaヒープ**を小さくするチューニングを行ってください。
仮想メモリが不足している場合は、他の不要なプロセスを終了して仮想メモリに余裕を持たせるか、物理メモリ(RAM)またはスワップファイルを拡張して仮想メモリを増やすようにチューニングを行ってください。

ユーザ空間

Java VMは、OSの仮想メモリ資源を効率的に利用するために、起動時に**Javaヒープ**の各領域の初期値を割り当て、各領域の最大値まで段階的に拡大する制御方法を用いています。
具体的には、Javaプロセスの起動時は**メモリ割り当てプール**および**Permanent世代領域**の各初期値のサイズを割り当てます。その後、**FullGC**処理の結果、各領域が不足した場合、各領域の最大値まで、段階的に拡大していきます。
なお、各領域を拡大していく過程で、OS側で物理メモリの資源をディスクにスワップする処理が発生する場合があります。このスワップ処理により、各領域を拡大する**FullGC**処理に時間がかかる場合があります。**FullGC**処理中のスワップ処理発生による時間遅延が問題になる場合は、**Javaヒープ**の各領域に対する初期値と最大値は同じ値に指定してください。

■ガーベジコレクションのログ

GCのログを採取する方法は、“7.3.1 ガーベジコレクションのログ出力”を参照してください。

■ガーベジコレクション処理の実行抑止

図2に示す機能を使用するJavaアプリケーションを実行すると、Javaヒープ内に存在する全オブジェクトの移動が禁止されるクリティカルセクションと呼ばれる状態が、当該機能の利用状況に応じて発生します。
クリティカルセクション状態発生時は、オブジェクトの移動が禁止されるため、オブジェクト移動が必須となるGC処理の実行が抑止される状態となります。
Java VMは、JavaアプリケーションによりGC処理の実行が抑止されている際に発生したオブジェクト生成要求に対し、New世代領域に空きが無い場合には、Old世代領域の空き領域を一時的に使用して対応します。
そして、要求されたオブジェクト量を満たす空きがOld世代領域にない場合には、java.lang.OutOfMemoryError例外を発生させます。
そのため図2の機能を多用するJavaアプリケーションの場合は、GC処理実行が抑止される状態も多数発生する可能性が高くなり、当該機能を多用しないJavaアプリケーションに比べ、GC処理実行抑止に因るjava.lang.OutOfMemoryError例外が発生しやすい状態にあります。
特にOld世代領域が小さい状態でJavaアプリケーションを実行した場合は、Old世代領域の空きとなり得る最大値（仮にOld世代領域を全く使用しない場合だと、Old世代領域自身の大きさ）もその大きさに比例して小さいため、その傾向が強まることがあります。

なおFJVMでは、GC処理の実行抑止によりjava.lang.OutOfMemoryError例外が発生したかどうかの情報を出力する機能を提供しています。
詳しくは“7.2.5 メモリ領域不足事象発生時のメッセージ出力機能の強化”を参照してください。

図2 ガーベジコレクション処理の実行が抑止される機能

【GC処理の実行が抑止される状態となるJNI関数】

- GetPrimitiveArrayCritical()実行からReleasePrimitiveArrayCritical()実行までの間
- GetStringCritical()実行からReleaseStringCritical()実行までの間

【GC処理の実行が抑止される状態となるJVMPI関数】

- DisableGC()実行からEnableGC()実行までの間

【GC処理の実行が抑止される状態となるJVMPIイベント】

- JVMPI_EVENT_THREAD_START
- JVMPI_EVENT_CLASS_LOAD
- JVMPI_EVENT_CLASS_UNLOAD
- JVMPI_EVENT_JNI_GLOBALREF_ALLOC
- JVMPI_EVENT_JNI_GLOBALREF_FREE
- JVMPI_EVENT_JNI_WEAK_GLOBALREF_ALLOC

- JVMPI_EVENT_JNI_WEAK_GLOBALREF_FREE
- JVMPI_EVENT_OBJECT_ALLOC
- JVMPI_EVENT_MONITOR_CONTENDED_ENTER
- JVMPI_EVENT_MONITOR_CONTENDED_ENTERED
- JVMPI_EVENT_MONITOR_CONTENDED_EXIT
- JVMPI_EVENT_MONITOR_WAIT
- JVMPI_EVENT_MONITOR_WAITED
- JVMPI_EVENT_HEAP_DUMP
- JVMPI_EVENT_METHOD_ENTRY
- JVMPI_EVENT_METHOD_ENTRY2
- JVMPI_EVENT_METHOD_EXIT

7.1.6 ネイティブモジュール

Windows（64ビット）

Windows Server(R) for Itanium-based Systemsで動作するJavaプロセス内で実行されるネイティブモジュールについては、以下の注意事項があります。

- Java VMが行なうベクタ化例外制御との競合発生の可能性があるため、Javaプロセス内で実行されるネイティブモジュールでは、ベクタ化例外制御を用いないでください。
- Javaプロセス内で実行されるネイティブモジュールでC++による構造化例外を使用していても、当該モジュールがその例外を受け取れない場合(C++による構造化例外処理を使用できない場合)があります。

  特に次の例外については、C++により構造化例外の対象として設定していたとしても、Java VM側で異常終了対象例外として処理し、Javaプロセスを異常終了させます。

  ー EXCEPTION_ACCESS_VIOLATION

  ー EXCEPTION_STACK_OVERFLOW

  ー EXCEPTION_ILLEGAL_INSTRUCTION

  ー EXCEPTION_INT_OVERFLOW

  ー EXCEPTION_INT_DIVIDE_BY_ZERO

  ー 例外コード (0xC000001E)

  またJavaメソッドを実行するスレッド上において、Javaメソッドの中から呼び出された ネイティブモジュール内で発生した例外を、当該Javaメソッドの呼出し元のネイティブ モジュール内でキャッチすることはできません。

7.2 FJVM

FJVMは、InterstageのデフォルトJava VMです。

FJVMは、Oracle CorporationのJava VMであるJava HotSpot Server VMをベースに、富士通独自技術による性能改善やJava VMのトラブルシューティングに関する機能強化などを追加実装したJava VMです。このため、**FJVM**は、Java HotSpot Server VMと機能的な互換性を基本的にもっています。

ここでは、富士通独自技術によるFJVM固有の機能について説明します。

なお以下の固有機能についてはJava HotSpot Client VMにおいても対応しています。

- FJVMでサポートされるガーベジコレクション処理（FJGCに関する内容を除く）
- Java VM異常終了時のログ出力機能の強化
- ガーベジコレクション処理の結果ログ出力機能の強化
- メモリ領域不足事象発生時のメッセージ出力機能の強化
- Java VM終了時における状態情報のメッセージ出力機能
- java.lang.System.gc()実行時におけるスタックトレース出力機能

・ スタックオーバーフロー検出時のメッセージ出力機能

・ FJVMログ

7.2.1 FJVMでサポートされるガーベジコレクション処理

FJVMでサポートされるガーベジコレクション(GC)処理には、Javaヒープの**New**世代領域に対するGC制御方法の違いにより、以下の4種類があります。

■標準GC(シリアルGC)

New世代領域用GC制御に対して何も付加機能を追加していない「標準機能のみ」で構成されたGC処理です。後述の**パラレルGC**との対比から、標準GCのことを**シリアルGC**と呼ぶ場合もあります。

Java HotSpot Client VM、実行モードが64ビットモードのJDK/JRE 1.4(ただし、Windows Server(R) 2003 for Itanium-based SystemsおよびRHEL-AS4(IPF)にのみ対応)の**FJVM**、および図1のオプションを指定した場合の**FJVM**では、**シリアルGC**が実行されます。なお、図1のオプションは、後述の**FJGC**によるGC制御を無効にする、またはシリアルGCによるGC制御を有効にするオプションです。

図1 FJVMでシリアルGCを有効にする場合に指定するオプション

```
JDK/JRE 1.4の場合: -XX:-UseFJGC
JDK/JRE 5.0、6の場合: -XX:+UseSerialGC
```

図2は**シリアルGC**使用時に利用可能となる「Javaヒープのチューニング用オプション」です。

図2 Javaヒープチューニング用オプション(シリアルGC使用時)

```
-Xms
-Xmx
-XX:NewSize
-XX:MaxNewSize
-XX:NewRatio
-XX:SurvivorRatio
-XX:TargetSurvivorRatio
-XX:PermSize
-XX:MaxPermSize
```

■New世代領域サイズ自動調整機能付きGC(FJGC)

New世代領域用GC制御に対し、富士通独自技術による「**New世代領域サイズ自動調整機能**」を追加して構成されたGC処理です。富士通独自技術によるGC制御を用いていることから、このGCを**FJGC**と呼ぶ場合もあります。

実行モードが32ビットモードのJDK/JRE 1.4の**FJVM**の場合は、デフォルトでこのGC処理が実行されます。

JDK/JRE 5.0の**FJVM**の場合は、図3のオプションを指定した場合に、このGC処理が実行されます。

なお、Java HotSpot Client VMおよびJDK/JRE 6のFJVMでは、本機能を提供していません。

図3 JDK/JRE 5.0のFJVMでNew世代領域サイズ自動調整機能を有効にするオプション

```
-XX:+UseFJGC
```

図4は**FJGC**使用時に利用可能となる「Javaヒープのチューニング用オプション」です。

図4 Javaヒープチューニング用オプション(FJGC使用時)

```
-Xms
-Xmx
-XX:PermSize
-XX:MaxPermSize
```

New世代領域サイズ自動調整機能未提供のFJVM

次のFJVMには、New世代領域サイズ自動調整機能を提供していません。

・ 実行モードが64ビットモードの**FJVM**

・ JDK/JRE6の**FJVM**

この**FJVM**に図3のオプションを指定した場合は、図5のワーニングメッセージが標準エラー出力へ出力され、オプションの指定は無効になります。

図5 -XX:+UseFJGCを指定した際に出力されるメッセージ

【実行モードが64ビットモードの**FJVMの場合】**

```
warning: -XX:+UseFJGC is not supported in Java HotSpot 64-Bit Server VM.
```

【実行モードが32ビットモードでJDK/JRE 6の**FJVMの場合】**

```
warning: -XX:+UseFJGC is not supported in Java HotSpot Server VM.
```

■New世代領域用制御処理並列化機能付きGC（パラレルGC）

New世代領域用GC制御に対し、「当該処理を並列化して実行する機能」を追加して構成されたGC処理です。New世代領域用のGC制御を並列化して実行することから、このGCを**パラレルGC**と呼ぶ場合もあります。
図6のオプションは、パラレルGCによるGC制御を有効にするオプションです。

JDK/JRE 5.0、6の**FJVM**の場合は、デフォルトでこのGC処理が実行されます。

図6 JDK/JRE 5.0、6でパラレルGCを有効にする場合に指定するオプション

```
-XX:+UseParallelGC
```

図7は**パラレルGC**使用時に利用可能となる「Javaヒープのチューニング用オプション」です。

図7 Javaヒープチューニング用オプション（パラレルGC使用時）

```
-Xms
-Xmx
-XX:NewSize
-XX:MaxNewSize
-XX:NewRatio
-XX:PermSize
-XX:MaxPermSize
```

JDK/JRE 1.4の場合

JDK/JRE 1.4の**FJVM**では、**パラレルGC**はサポート対象外のGC処理です。

JVMPIとJVMTI

次の場合、Java Virtual Machine Profiling Interface(JVMPI)をサポートしていません。

・ JDK/JRE 5.0を使用し、GC処理としてパラレルGCまたはCMS付きパラレルGCが選択されている場合

・ JDK/JRE 6を使用している場合

JVMPI相当の機能を使用する場合には、JDK/JRE 5.0から導入されたJava Virtual Machine Tool Interface(JVMTI)を使用してください。

JDK/JRE 5.0でJVMPI機能は推奨されていません。**やむを得ず使用する場合**は、GC処理として**シリアルGC**を使用してください。

GC処理用スレッド数

パラレルGCを使用した場合は、実行するハードウェアに搭載しているCPU数分のGC処理用スレッドがJavaプロセス内に作成されます。そのため、GC処理用スレッドの数分だけ、スタック域などのスレッド用のメモリ領域が必要となります。

Javaプロセス内でのメモリ量を抑えるためなど、GC処理用スレッドの数を抑制する場合には、図8のオプションでGC処理用スレッドの数を指定することにより、GC処理用スレッドの数を抑制することができます。

なおGC処理用スレッドの数を抑制した分だけGC処理における性能がおちる場合もありますので、このオプションを用いる場合には、十分な性能確認を実施してください。また、一般的に、CPU数以上の数のGC処理用スレッドを作成しても、GC処理における性能向上にはつながりません。

図8 パラレルGCで使用するGC処理用スレッドの数を指定するオプション

-XX:ParallelGCThreads=GC処理用スレッドの数(0を指定した場合は搭載CPU数)

メモリ割り当てプールの省略値自動調整機能

FJVMの**パラレルGC**では、JDK/JRE 5.0、6の機能であるエルゴノミクス機能によるメモリ割り当てプールの初期値(-Xms)および最大値(-Xmx)の省略値自動調整機能（マシンの物理メモリ量などに応じて、-Xmsおよび-Xmxの各オプションに対する省略値を自動的に決定する機能）を無効にしています。

JDK/JRE 5.0、6の**FJVM**で、エルゴノミクス機能によるメモリ割り当てプールの省略値自動調整機能を有効にする場合は、図9のオプションを指定してください。

ただし、このオプション指定は、システムのメモリ資源不足の要因となる場合があるため、システム内に複数のJavaプロセスを起動、実行する場合には使用しないでください。

図9 JDK/JRE 5.0、6でメモリ割り当てプールの省略値自動調整機能を有効にするオプション

-XX:+AutomaticallyJavaHeapSizeSetting

メモリ領域不足事象の検出機能

FJVMの**パラレルGC**では、JDK/JRE 5.0、6の機能であるエルゴノミクス機能によるメモリ領域不足事象の検出機能（図10の各オプション指定値による条件が同時に成立した場合に、メモリ領域不足事象（java.lang.OutOfMemoryError）として検出する機能）を無効にしています。

JDK/JRE 5.0、6の**FJVM**でエルゴノミクス機能によるメモリ領域不足事象検出機能を有効にする場合は、図11のオプション（使用するJDK/JREの違いにより異なるため注意が必要です）を指定してください。

ただし、このオプション指定で検出されるメモリ領域不足事象は、Javaヒープの使用量だけではなく、メモリ領域不足事象検出用オプションで指定された値、およびガーベジコレクション処理の動作状況から得られた統計情報などを元に決定されるため、Javaヒープの使用量が不足していない状態であっても、メモリ領域不足事象が検出される場合がありますので注意してください。

図10 メモリ領域不足事象検出条件

-XX:GCTimeLimit=GC処理に要する時間の上限値 (デフォルトは98) Javaアプリケーションの合計処理時間に対して、GC処理に要した時間の上限値をパーセント単位(%)で指定します。 指定された上限値を超えた場合、検出条件の一方が成立します。 **-XX:GCHeapFreeLimit=GC処理後のJavaヒープ量の空きスペースの下限値** (デフォルトは2) メモリ割り当てプールの最大値に対する、GC処理後のJavaヒープ量の空きスペースの下限値をパーセント単位(%)で指定します。 指定された下限値を下回った場合、検出条件の一方が成立します。

図11 JDK/JRE 5.0、6でメモリ領域不足事象検出を有効にするオプション

【JDK/JRE 5.0の場合】

```
-XX:+UseGCTimeLimit
```

【JDK/JRE 6の場合】

```
-XX:+UseGCOverheadLimit
```

メモリ割り当てプールの最小使用量保証機能

FJVMで**パラレルGC**を使用した場合、JDK/JRE 5.0、6の機能であるパラレルGC処理のエルゴノミクス機能（メモリ割り当てプール内の各世代領域サイズの動的変更機能）により、Javaアプリケーションの実行状況や負荷／GC処理に掛かる時間などの情報から、GC処理としての最適動作状態になるように、メモリ割り当てプール内の各世代領域サイズが自動的に調整・変更および最適化されます。

その際、メモリ割り当てプールの使用量が-Xmsオプションで指定した値（メモリ割り当てプールの初期値）よりも小さくなることあります（-Xmsオプションで指定した値よりも下回るメモリ割り当てプールの使用量で、GC処理としての最適動作状態になることがあります）。

パラレルGCを使用してJavaアプリケーションを実行する際、メモリ割り当てプールの使用量についての操作を行なう場合は、以下のオプションを指定してください。

メモリ割り当てプールの最小使用量保証機能を操作するオプション

【-Xmsオプションで指定した値よりもメモリ割り当てプールの使用量を小さくしない場合】

```
-XX:+UseAdaptiveSizePolicyMinHeapSizeLimit
```

パラレルGC処理のエルゴノミクス機能動作時、-Xmsオプション指定値によるメモリ割り当てプールの最小使用量保証機能を有効にします。

Javaアプリケーション実行時に使用されるメモリ割り当てプールの大きさは，「-Xmsオプション指定値」から「-Xmxオプション指定値」の範囲で変動します。

（-Xmsオプション指定値と-Xmxオプション指定値が等しい場合、使用中となるメモリ割り当てプールの大きさは変動しません。）

この状態は、JDK/JRE 6でパラレルGCを使用する場合のデフォルト状態です。

【-Xmsオプションで指定した値よりもメモリ割り当てプールの使用量を小さくして良い場合】

```
-XX:-UseAdaptiveSizePolicyMinHeapSizeLimit
```

パラレルGC処理のエルゴノミクス機能動作時、-Xmsオプション指定値によるメモリ割り当てプールの最小使用量保証機能を無効にします。

Javaアプリケーション実行時に使用されるメモリ割り当てプールの大きさは，「Java VMとしての下限値」から「-Xmxオプション指定値」の範囲で変動します。

（-Xmsオプション指定値と-Xmxオプション指定値が等しい場合であっても、使用中となるメモリ割り当てプールの大きさは変動します。）

この状態は、JDK/JRE 5.0でパラレルGCを使用する場合のデフォルト状態です。

なおメモリ割り当てプールの最小使用量保証機能の有効／無効に関わらず、Javaプロセス起動時に使用されるメモリ割り当てプールの大きさは、-Xmsオプションで指定された値となります。

■コンカレント・マーク・スイープGC付きパラレルGC（CMS付きパラレルGC）

New世代領域用GC制御を並列化して実行する機能、およびJavaアプリケーションと同時並列に動作するOld世代領域用GC制御「コンカレント・マーク・スイープGC(CMS-GC)機能」を追加して構成されたGC処理です。CMS-GC機能を追加されたパラレルGC制御であることから、このGCを**CMS付きパラレルGC**と呼ぶ場合もあります。
図12のオプションは、**CMS付きパラレルGC**によるGC制御を有効にするオプションです。

CMS付きパラレルGC

CMS付きパラレルGCによるGC制御は、以下のエディションに搭載されたJDK/JRE 5.0、6にだけ提供しています。

・ Interstage Application Server Enterprise Edition

CMS-GC

CMS-GCは、Javaアプリケーションが Full GCによって停止されることにより影響を受ける「アプリケーションの応答性能の平準化」を改善するために実行される「Old世代領域内の不要オブジェクト回収用のGC機構」です。

Javaアプリケーションを停止させて実行するFull GCに対し、CMS-GCはJavaアプリケーションと同時並列に動作し、Old世代領域内の不要オブジェクトを回収します。CMS-GCの実行により、Old世代領域(New世代領域にあるオブジェクトの移動先や巨大オブジェクトの生成先となる領域)の空きを、Javaアプリケーション動作と並行して増加させることができるため、Full GCの発生を抑えることができます。これにより、JavaアプリケーションはFull GCにより停止される影響を受けにくくなり、応答性能平準化の改善が期待できます。

なおCMS-GC動作中は、NewGC処理/Full GC処理の実行開始が遅延する場合があります。そして遅延期間中は、Javaアプリケーションとしての動作も停止します。そのため、ガーベジコレクション処理の結果ログ内のNewGC処理/Full GC処理に対して出力されたGC処理実行時間よりも長い間、Javaアプリケーションとしての動作が停止している場合があります。

図12 JDK/JRE 5.0、6でCMS付きパラレルGCを有効に場合に指定するオプション

-XX:UseFJcmsGC=タイプ タイプとして以下の値が指定できます type0 type1 type2

-XX:UseFJcmsGC=type0が指定された場合

図13は-XX:UseFJcmsGC=type0指定時に利用可能となる「Javaヒープのチューニング用オプション」です。図13に示すチューニング用オプションを用いて、利用者が細かくチューニング作業を行なうことが可能なCMS付きパラレルGCを使用する場合に指定します。

図13 Javaヒープチューニング用オプション(-XX:UseFJcmsGC=type0指定時)

-Xms -Xmx -XX:NewSize -XX:MaxNewSize -XX:SurvivorRatio -XX:TargetSurvivorRatio -XX:PermSize -XX:MaxPermSize

-XX:UseFJcmsGC=type1が指定された場合

New世代領域用GCでのオブジェクト回収を重視した調整が成されたCMS付きパラレルGCを使用する場合に指定します。

実行されるアプリケーションの特徴が「大半のオブジェクトが、少ない回数のNew世代領域用GCによって回収されるオブジェクト」である場合に、CMS-GCによる応答性能平準化の改善効果が得やすい調整になっています。
Javaヒープのチューニングを行なう場合は、まず-Xms/-Xmxおよび-XX:PermSize/-XX:MaxPermSizeの各オプションにより、メモリ割り当てプールおよびPermanent世代領域の大きさを調整します。そして必要に応じて、-XX:NewSize/-XX:MaxNewSizeの各オプションでNew世代領域の大きさを調整します。

なおNew世代領域サイズとして、メモリ割り当てプール最大サイズ未満の値を指定できます。ただしNew世代領域サイズを大きくしすぎると、Full GCが発生しやすくなってしまうので注意が必要です。

図14は-XX:UseFJcmsGC=type1指定時に利用可能となる「Javaヒープのチューニング用オプション」です。

図14 Javaヒープチューニング用オプション(-XX:UseFJcmsGC=type1指定時)

-Xms -Xmx -XX:NewSize

```
-XX:MaxNewSize
-XX:PermSize
-XX:MaxPermSize
```

-XX:UseFJcmsGC=type2が指定された場合

CMS-GCでのオブジェクト回収を重視した調整が成されたCMS付きパラレルGCを使用する場合に指定します。

実行されるアプリケーションの特徴が「大半のオブジェクトが、何回かのGCを経てから回収される(長期間常駐せず、ある程度の短期間で回収される)オブジェクト」である場合に、CMS-GCによる応答性能平準化の改善効果が得やすい調整になっています。

Javaヒープのチューニングを行なう場合は、まず-Xms/-Xmxおよび-XX:PermSize/-XX:MaxPermSizeの各オプションにより、メモリ割り当てプールおよびPermanent世代領域の大きさを調整します。そして必要に応じて、-XX:NewSize/-XX:MaxNewSizeの各オプションでNew世代領域の大きさを調整します。

なおNew世代領域サイズとして、メモリ割り当てプール最大サイズ未満の値を指定できます。ただしNew世代領域サイズを大きくしすぎると、Full GCが発生しやすくなってしまうので注意が必要です。

図15は-XX:UseFJcmsGC=type2指定時に利用可能となる「Javaヒープのチューニング用オプション」です。

図15 Javaヒープチューニング用オプション(-XX:UseFJcmsGC=type2指定時)

```
-Xms
-Xmx
-XX:NewSize
-XX:MaxNewSize
-XX:PermSize
-XX:MaxPermSize
```

JDK/JRE 1.4の場合

JDK/JRE 1.4の**FJVM**では、**CMS付きパラレルGC**はサポート対象外のGC処理です。

JVMPIとJVMTI

次の場合、Java Virtual Machine Profiling Interface(JVMPI)をサポートしていません。

- JDK/JRE 5.0を使用し、GC処理としてパラレルGCまたはCMS付きパラレルGCが選択されている場合
- JDK/JRE 6を使用している場合

JVMPI相当の機能を使用する場合には、JDK/JRE 5.0から導入されたJava Virtual Machine Tool Interface(JVMTI)を使用してください。

JDK/JRE 5.0でJVMPI機能は推奨されていません。**やむを得ず使用する場合**は、GC処理として**シリアルGC**を使用してください。

JDK/JRE 5.0のJVMTI

JDK/JRE 5.0を使用し、GC処理としてCMS付きパラレルGCが選択されている場合、Java Virtual Machine Tool Interface(JVMTI)で、以下の権限を必要とする機能は使用できません。

- can_tag_objects
- can_generate_object_free_events

GC処理用スレッド数

CMS付きパラレルGCを使用した場合は、実行するハードウェアに搭載しているCPU数に依存した数のGC処理用スレッドがJavaプロセス内に作成されます。そのため、GC処理用スレッドの数分だけ、スタック域などのスレッド用のメモリ領域が必要となります。
Javaプロセス内でのメモリ量を抑えるためなど、GC処理用スレッドの数を調整する場合には、図16のオプションでGC処理用スレッドの

数を指定することにより、GC処理用スレッドの数を調整することができます。
なおGC処理用スレッドの数を抑制した分だけGC処理における性能がおちる場合もありますので、このオプションを用いる場合には、十分な性能確認を実施してください。また、一般的に、CPU数以上の数のGC処理用スレッドを作成しても、GC処理における性能向上にはつながりません。

図16 CMS付きパラレルGCで使用するGC処理用スレッドの数を指定するオプション

-XX:ParallelGCThreads=New世代領域GC用スレッドの数 New世代領域用GC処理を行なうGCスレッドの数を指定します。 最小値は「2」です。 「0」または「1」が指定された場合は、デフォルト値となります。 デフォルト値は以下のとおりです。 - 実行するハードウェアに搭載しているCPU数が1の場合 = 2 - 実行するハードウェアに搭載しているCPU数が2以上7以下の場合 = CPU数分 - 実行するハードウェアに搭載しているCPU数が8以上の場合 = 8

■デフォルトのGC

GCは、デフォルトのGCの使用を推奨します。
通常、デフォルトのGCを変更する必要はありません。

以下に、各環境におけるデフォルトのGCを示します。

【JDK/JREの実行モードが32ビットモードの場合】

	JDK/JRE 1.4		JDK/JRE 5.0 または JDK/JRE 6	
VMの種類	Client VM	FJVM	Client VM	FJVM
デフォルトのGC	シリアルGC	FJGC	シリアルGC	パラレルGC

【JDK/JREの実行モードが64ビットモードの場合】

	JDK/JRE 1.4	JDK/JRE 5.0 または JDK/JRE 6
VMの種類	FJVM	FJVM
デフォルトのGC	シリアルGC	パラレルGC

※実行モードが64ビットモードのClient VMを提供していません。

※Windows Server(R) for Itanium-based Systems、Linux for ItaniumのJDK/JRE 6を提供していません。

7.2.2 New世代領域サイズ自動調整機能

ここでは、**FJVM**で実装している**New世代領域サイズ自動調整機能**について説明します。

なお、次のFJVMでは、**New世代領域サイズ自動調整機能**を提供していません。

- 実行モードが64ビットモードのFJVM
- JDK/JRE 6のFJVM

JDK/JRE 1.4のFJVM

JDK/JRE 1.4の**FJVM**におけるデフォルトのGC処理は、FJGC（実行モードが32ビットモードの場合）またはシリアルGC（実行モードが64ビットモードの場合（ただし、Windows Server(R) 2003 for Itanium-based SystemsおよびRHEL-AS4(IPF)にのみ対応））です。

JDK/JRE 5.0、6のFJVM

JDK/JRE 5.0、6の**FJVM**におけるデフォルトのGC処理は、パラレルGCです。

■New世代領域サイズ自動調整機能

FJVMでは、富士通独自技術により、JavaヒープのNew世代領域のサイズを自動的に調整および最適化する機能を実装しています。この機能を、**New世代領域サイズ自動調整機能**(以降、**自動調整機能**と略する)といいます。

■New世代領域サイズ自動調整機能が有効となる条件

自動調整機能は、“メモリ割り当てプールの最大サイズ”が図1に示す値の範囲内である場合に有効となります。(“メモリ割り当てプールの最大サイズ”は、“-Xmx”オプションで指定できます。)

“メモリ割り当てプールの最大サイズ”が図1で示す範囲よりも大きな値である場合は、**FJVM**起動時に図2のメッセージが標準出力へ出力され、**自動調整機能**は停止状態となります。

自動調整機能が停止状態となった場合は、New世代領域の大きさに対する自動調整処理は行なわれず、表1の注1で示すオプションに関する制御処理を除き、シリアルGCでの制御処理と同じ処理方法でNew世代領域の大きさが制御されるようになります。

図1 自動調整機能が有効となるメモリ割り当てプールの最大サイズ

Windows

【Windows(R) 2000の場合】
JDK/JRE 1.4: 920MB以下
JDK/JRE 5.0: 900MB以下

【Windows Server(R) 2003、Windows Server(R) 2008、Windows(R) XPまたはWindows Vista(R)の場合】
JDK/JRE 1.4: 700MB以下（ただし、Windows Server(R) 2008を除く）
JDK/JRE 5.0: 700MB以下

Solaris

JDK/JRE 1.4: 2340MB以下
JDK/JRE 5.0: 2250MB以下

Linux

JDK/JRE 1.4: 1600MB以下
JDK/JRE 5.0: 1500MB以下

図2 自動調整機能停止時に出力されるメッセージ

```
Heap too large for dynamic eden: using static eden
```

■New世代領域サイズ自動調整機能で無効となるオプション

自動調整機能が有効な場合、New世代領域の大きさに関する値が自動的に調整・最適化されるため、表1の各オプションへの指定値は無効となります。

なお、実行モードが64ビットモードのFJVMでは**自動調整機能**を提供していないため、Java HotSpot Server VMと同様、“表7.2 自動調整機能で無効となるオプション”の各オプションへの指定値は有効となります。

表7.2 自動調整機能で無効となるオプション

オプション	オプションの機能
-XX:NewSize	New世代領域のヒープサイズを指定します。
-XX:MaxNewSize	New世代領域の最大ヒープサイズを設定します。
-XX:NewRatio	New世代領域とOld世代領域のサイズ比率を指定します。
-XX:SurvivorRatio (注1) (注2)	New世代領域を構成するEden領域とSurvivor領域のサイズ比率を指定します。
-XX:TargetSurvivorRatio (注1) (注2)	GC処理後の生存オブジェクトがSurvivor領域を占める割合を、指定したパーセンテージ値に調整します。

注1) 図1で示す範囲よりも大きな値を“-Xmx”オプションで指定して**自動調整機能**が停止した場合でも、このオプションへの指定値は無効となります。
注2) パラレルGCの場合は、このオプションへの指定値は無効となります。

■New世代領域サイズ自動調整機能の無効化

自動調整機能は図3のオプションを指定することで無効にすることができます。

図3 自動調整機能を無効にするオプション

```
-XX:-UseFJGC
```

JDK/JRE 1.4の**FJVM**を使用している場合で、かつ以下のような場合には、**自動調整機能**を無効にし、シリアルGCを使用してください。

また、JDK/JRE 5.0のFJVMで**自動調整機能**を有効にしている場合で、かつ以下のような場合には、**自動調整機能**を有効にするオプションを削除し、パラレルGCまたはシリアルGCを使用してください。

- Javaヒープに対するチューニングを手動で行いたい場合:

  Java HotSpot Server VMを使用する場合と同じように、表1の各オプションを用いてJavaヒープに対するチューニングを行いたい場合には、**自動調整機能**を無効にしてください。

- ネイティブモジュールで使用するメモリ量をある程度確保する必要がある場合:

  自動調整機能が有効な場合、当該機能が使用する作業域(“-Xmx”オプションで指定した値の2/3の大きさ)をユーザ空間から確保(リザーブ処理だけの確保)しています。
  そのため、“-Xmx”オプションで図1に示す範囲内でかつその上限値近くの値を指定した場合、**自動調整機能**が使用する作業域確保のため、ネイティブモジュールから使用できるスタックやヒープなどのメモリ量が非常に少なくなっています。
  図1に示す範囲の上限値近くの値のメモリ割り当てプールの大きさを確保し、かつネイティブモジュールで使用するスタックやヒープなどのセグメントに対するメモリ量もある程度確保する必要がある場合には、**自動調整機能**を無効にしてください。

- Javaプロセスのユーザ空間から連続域の状態で確保できるメモリ量が小さいシステムの場合:

  自動調整機能が有効な場合、メモリ割り当てプールと**自動調整機能**用の作業域を連続域としてユーザ空間から確保しています。
  また、ユーザ空間から連続域として確保できる最大メモリ量は、システムごとに異なります。
  そのため、“-Xmx”オプションで指定したメモリ割り当てプールの大きさが図1に示す値の範囲内であっても、Javaプロセスのユーザ空間から、メモリ割り当てプールと**自動調整機能**用の作業域が連続域として確保できない場合には(Javaプロセスのユーザ空間から、連続域の状態で確保できるメモリ量が小さいシステムの場合には)、Java VMは図4のメッセージを出力して、Javaアプリケーションの実行を中止します。
  その場合は、より小さな値を“-Xmx”オプションで指定するか、**自動調整機能**を無効にしてください。

- GC処理実行抑止によりjava.lang.OutOfMemoryErrorが発生した場合:

  自動調整機能では、Javaヒープ域におけるNew世代領域とOld世代領域の比率をJavaアプリケーションからのオブジェクト生成要求および解放タイミングの状態変動により動的に変更します。
  そしてNew世代領域とOld世代領域を管理するメモリ割り当てプールの大きさには上限があるため、ある時点においてNew世代領域のサイズを調整・最適化した結果、New世代領域が大きくなった場合は、Old世代領域のサイズは小さくなります(逆にNew世代領域が小さくなった場合は、Old世代領域のサイズは大きくなります)。
  そのため**自動調整機能**が有効な場合、以下の条件に合致するJavaアプリケーションを実行すると、Old世代領域が小さくなっている時とJavaアプリケーションによりGC処理の実行が抑止されている時が重なるタイミングが多くなることがあり、結果的にGC処理実行抑止によるjava.lang.OutOfMemoryErrorの発生が、**自動調整機能**を無効とした場合よりも多くなることがあります。

  ー 寿命の短いオブジェクトを多用するJavaアプリケーション。かつ、

  ー GC処理の実行が抑止される機能を多用するJavaアプリケーション。

  GC処理実行抑止によるjava.lang.OutOfMemoryErrorの発生傾向が強い場合は、**自動調整機能**を無効にしてください。

  なおGC処理の実行が抑止される機能については“7.1.5 Javaヒープとガーベジコレクション”を参照してください。

■補足

ユーザ空間内で使用できるメモリ量には、システムごとに異なる上限値があります。

そして、Javaアプリケーションを実行する場合には、Javaヒープ(“-Xmx”オプションで指定するメモリ割り当てプールなど)のほかに、Javaアプリケーション自身、Java VM、そしてネイティブモジュールが動作するために使用するメモリ領域も必要です。

そのため、ユーザ空間内で使用できるメモリ量の上限から、Javaアプリケーション自身、Java VM、そしてネイティブモジュールが使用するメモリ量などを差し引いた値がJavaヒープとして使用できる上限となります。

“-Xmx”オプションなどにより、その上限を超える値をJavaヒープの大きさとして指定した場合、Java VMは以下のメッセージを出力し、Javaアプリケーションの実行を中止します。
また、**自動調整機能**が有効な場合には、メモリ割り当てプールと**自動調整機能**用の作業域が連続域として確保できない場合にも、Java VMは以下のメッセージを出力し、Javaアプリケーションの実行を中止します。

図4 Javaヒープなどが確保できない場合のメッセージ

```
Error occurred during initialization of VM
Could not reserve enough space for object heap
```

またSolaris用およびLinux用のJava VM起動時またはJavaアプリケーション実行中にメモリ不足(「ユーザ空間不足」または「仮想メモリ不足」)が発生した場合、Java VMは以下のメッセージを出力し、Javaアプリケーションの実行を中止します。

図5 Java VM起動時またはJavaアプリケーション実行中にメモリ不足が発生した場合のメッセージ

Java VM起動時の場合:

```
制御名: mmap failed: errno=エラー情報, 制御情報....
Error occurred during initialization of VM
mmap failure
```

Javaアプリケーション実行中の場合:

```
制御名: mmap failed: errno=エラー情報, 制御情報....
(上記メッセージに続いて、java.lang.OutOfMemoryErrorメッセージが出力される場合があります。)
```

制御名　：メモリ不足が発生した際のJava VMの制御名
エラー情報:メモリ不足が発生した際のJava VMのエラー情報
制御情報　：メモリ不足が発生した際のJava VMの制御情報

ユーザ空間が不足している場合は、Javaヒープを小さくするチューニングを行ってください。
仮想メモリが不足している場合は、他の不要なプロセスを終了して仮想メモリに余裕を持たせるか、物理メモリ(RAM)またはスワップファイルを拡張して仮想メモリを増やすようにチューニングを行ってください。

7.2.3 Java VM異常終了時のログ出力機能の強化

FJVMでは「Java VM異常終了時のログ出力機能」の強化を行っています。

Java VM異常終了時に**FJVM**が出力するログ情報(FJVMログ)の見方については、“7.2.11 FJVMログ”を参照してください。

7.2.4 ガーベジコレクション処理の結果ログ出力機能の強化

FJVMでは、“-verbose:gc”オプション指定時にガーベジコレクション(GC)処理の結果ログを出力する「**ガーベジコレクション処理の結果ログ出力機能**」の強化を行っています。

“-verbose:gc”オプションを指定してGC処理の結果ログを出力する際、図1のオプションを追加指定することにより、GC処理の結果ログとして出力される情報が、図2から図5に示す形式に拡張されます。また、図6から図8に出力例を示します。

図1 GC処理の結果ログとして出力される情報を拡張するオプション

```
-XX:+UseFJverbose
```

図2 GC処理の結果ログとして出力される情報の拡張形式

```
$1: [$2, [$3 : $4->$5($6)], [$7 : $8->$9($10)] $11->$12($13), [$14 : $15->$16($17)], $18 secs]
```

図2の各要素について、以下で説明します。

$1: 経過時間

GC処理の実行開始時間をJava VMが起動されてからの経過時間(秒)で示します。

$2: GC種別

実行したGC処理の種別を以下の名称で示します。

- GC
  New世代領域を対象とするGC処理(NewGC処理またはマイナーGC処理)における結果情報です。
- Full GC
  Javaヒープ域全体(メモリ割り当てプール(New世代領域、Old世代領域)およびPermanent世代領域)を対象とするGC処理(FullGC処理)における結果情報です。
- Full GC*
  使用されているGC処理がシリアルGC、FJGCまたはCMS付きパラレルGCの場合で、かつ、直前に実行されたNewGC処理では十分な空き領域が確保できず、そのNewGC処理に続けて実行されたFullGC処理における結果情報です("Full GC"の後に"*"の付く表示になります)。
  なおこの名称は“-verbose:gc”オプションだけを指定した場合には出力されません(単に"Full GC"として出力されます)。
- CMS initial-mark
  Old世代領域を対象とするCMS-GC処理のうち、initialマーク処理における結果情報です。
  CMS-GCでは、不要オブジェクトを検出するために行なう処理(initialマーク処理)を実行する際、極わずかな時間だけJavaアプリケーションを停止させます。
  注)不要オブジェクトの検出処理だけであるため、GC処理開始前後での各世代領域におけるオブジェクト量に変化はありません。
- CMS remark
  Old世代領域を対象とするCMS-GC処理のうち、finalマーク処理における結果情報です。
  CMS-GCでは、不要オブジェクトを検出するために行なう処理(finalマーク処理)を実行する際、極わずかな時間だけJavaアプリケーションを停止させます。
  注)不要オブジェクトの検出処理だけであるため、GC処理開始前後での各世代領域におけるオブジェクト量に変化はありません。

$3: New世代領域の識別名

New世代領域の識別名を示します。
使用されているGC処理の違いにより、以下の識別名が出力されます。

- DefNew: シリアルGCの場合
- SplitEden: FJGCの場合
- PSYoungGen: パラレルGCの場合
- ParNew: CMS付きパラレルGCの場合

$4: GC処理前のオブジェクト量(New世代領域)

GC処理実行前のNew世代領域に存在したオブジェクトの総量(バイト)です。

$5: GC処理後のオブジェクト量(New世代領域)

GC処理実行後のNew世代領域に存在するオブジェクトの総量(バイト)です。

$6: New世代領域の大きさ

New世代領域の大きさ(バイト)です。
注)使用されているGC処理がシリアルGC、パラレルGCまたはCMS付きパラレルGCの場合は、この大きさに「to space」領域の大きさが含まれません。
(シリアルGC、パラレルGCまたはCMS付きパラレルGCの場合、GC処理はNew世代領域を「eden space」、「from space」および「to space」の3つの内部領域に細分割して制御しています。)

$7: Old世代領域の識別名

Old世代領域の識別名を示します。
使用されているGC処理の違いにより、以下の識別名が出力されます。

- Tenured: シリアルGCの場合

- Tenured: FJGCの場合
- PSOldGen: パラレルGCの場合
- CMS: CMS付きパラレルGCの場合

$8: GC処理前のオブジェクト量(Old世代領域)

GC処理実行前のOld世代領域に存在したオブジェクトの総量(バイト)です。

$9: GC処理後のオブジェクト量(Old世代領域)

GC処理実行後のOld世代領域に存在するオブジェクトの総量(バイト)です。

$10: Old世代領域の大きさ

Old世代領域の大きさ(バイト)です。

$11: GC処理前のオブジェクト量(メモリ割り当てプール)

GC処理実行前のメモリ割り当てプールに存在したオブジェクトの総量(バイト)です。
$4+$8の値です。

$12: GC処理後のオブジェクト量(メモリ割り当てプール)

GC処理実行後のメモリ割り当てプールに存在するオブジェクトの総量(バイト)です。
$5+$9の値です。

$13: メモリ割り当てプールの大きさ

メモリ割り当てプールの大きさ(バイト)です。
$6+$10の値です。
注)使用されているGC処理がシリアルGC、パラレルGCまたはCMS付きパラレルGCの場合は、この大きさにNew世代領域の「to space」領域の大きさが含まれません。
(シリアルGC、パラレルGCまたはCMS付きパラレルGCの場合、GC処理はNew世代領域を「eden space」、「from space」および「to space」の3つの内部領域に細分割して制御しています。)

$14: Permanent世代領域の識別名

Permanent世代領域の識別名 です。
使用されているGC処理の違いにより、以下の識別名が出力されます。

- Perm: シリアルGCの場合
- Perm: FJGCの場合
- PSPermGen: パラレルGCの場合
- CMS Perm: CMS付きパラレルGCの場合

$15: GC処理前のオブジェクト量(Permanent世代領域)

GC処理実行前のPermanent世代領域に存在したオブジェクトの総量(バイト)です。

$16: GC処理後のオブジェクト量(Permanent世代領域)

GC処理実行後のPermanent世代領域に存在するオブジェクトの総量(バイト)です。

$17: Permanent世代領域の大きさ

Permanent世代領域の大きさ(バイト)です。

$18: GC処理実行時間

GC処理実行に要した時間(秒)です。
注)GC処理は、Javaアプリケーションとしての動作を停止させて実行されます。

$2、$11、$12、$13、および$18に対する出力情報は、GC処理の結果ログ出力機能として“-verbose:gc”オプションだけを指定した際に出力される情報と対応します。

図3 GC処理の結果ログとして出力される情報の拡張形式(CMS-GCの開始)

```
$1: CMS start
```

図3の各要素について、以下で説明します。

$1: 経過時間

CMS-GC処理の実行開始時間をJava VMが起動されてからの経過時間(秒)で示します。

図4 GC処理の結果ログとして出力される情報の拡張形式(FullGC発生によるCMS-GCの終了要求)

```
$1: CMS stop-req
```

図4の各要素について、以下で説明します。

$1: 経過時間

Full GC要求発生時にCMS-GC処理が実行中であった場合、CMS-GC処理の終了要求時点をJava VMが起動されてからの経過時間(秒)で示します。

注)CMS-GC処理が動作している最中にJavaヒープ不足やjava.lang.System.gc()実行などによりFull GC要求が発生した場合、Full GC処理は、実行中のCMS-GC処理の終了を待ってから開始されます。その際、CMS-GCが処理しているデータの整合性を維持させるため、CMS-GC処理の終了は強制的な打ち切り終了ではなく、CMS-GC内で打ち切り可能な処理まで完了させてからの終了になります。そのため、CMS-GC処理が実行中であった場合、Full GC処理要求の発生から実際にFull GC処理が開始されるまでに、ある程度の時間差が生じる可能性があります。

なおCMS-GC処理の終了要求時点からCMS-GC処理の実行が終了するまで、Javaアプリケーションとしての動作は停止します。そのため、CMS-GC処理の終了要求時点からFull GC処理の実行完了までが、この出力があった場合のFull GC処理によるJavaアプリケーション動作の実際の停止期間となります。

図5 GC処理の結果ログとして出力される情報の拡張形式(CMS-GCの終了)

パターン1:

```
$1: CMS stop($2), [CMS : $3->$4($5)], $6 secs
```

パターン2:

```
$1: CMS stop($2), $6 secs
```

図5の各要素について、以下で説明します。

$1: 経過時間

CMS-GC処理の実行終了時間をJava VMが起動されてからの経過時間(秒)で示します。

$2: 終了コード

CMS-GC処理の実行結果に対する終了コードを示します。
終了コードの違いにより、情報の出力形式パターンが異なります。
終了コードの種類および意味は以下のとおりです。

- 00: CMS-GC処理が終了しました。
  Old世代領域内で検出済の不要オブジェクトは回収しました。
  パターン1の出力形式で情報が出力されます。
- 10: Javaヒープ不足によるFull GC要求があったため、実行中のCMS-GC処理を終了しました。
  Old世代領域内で検出済の不要オブジェクトは回収しました。
  パターン1の出力形式で情報が出力されます。
- 20: java.lang.System.gc()などの外部要因によるFull GC要求があったため、実行中のCMS-GC処理を終了しました。
  Old世代領域内で検出済の不要オブジェクトは回収しました。
  パターン1の出力形式で情報が出力されます。

- 11: Javaヒープ不足によるFull GC要求があったため、実行中のCMS-GC処理を終了しました。
  Old世代領域内の不要オブジェクトは回収していません。
  パターン2の出力形式で情報が出力されます。
- 21: java.lang.System.gc()などの外部要因によるFull GC要求があったため、実行中のCMS-GC処理を終了しました。
  Old世代領域内の不要オブジェクトを回収していません。
  パターン2の出力形式で情報が出力されます。

$3: CMS-GC処理前のオブジェクト量(Old世代領域)

CMS-GC処理実行前のOld世代領域に存在したオブジェクトの総量(バイト)です。
通常はCMS-GC処理のfinalマーク処理実行時のOld世代領域に存在したオブジェクトの総量と等しくなります。
finalマーク処理実行時のOld世代領域に存在したオブジェクトの総量と等しくない場合は、CMS-GCによる不要オブジェクトの回収処理前にNew世代領域用GCが実行されたことを示します。

$4: CMS-GC処理後のオブジェクト量(Old世代領域)

CMS-GC処理実行後のOld世代領域に存在するオブジェクトの総量(バイト)です。

$5: Old世代領域の大きさ

Old世代領域の大きさ(バイト)です。

$6: CMS-GC処理実行時間

CMS-GC処理実行に要した時間(秒)です(CMS-GC開始からの経過時間です)。

有効なGC処理

このオプション指定により出力形式が拡張されるのは、使用するGC処理が以下の場合です。

- シリアルGC
- FJGC（JDK/JRE 1.4、5.0）
- パラレルGC (JDK/JRE 5.0、6)
- CMS付きパラレルGC (JDK/JRE 5.0、6)

クラスのアンロード情報

Javaアプリケーションがクラスのアンローディング処理を行なっていた場合には、FullGC処理中に該当するクラスのアンロード情報”[Unloading class アンロードされたクラス名]”が出力の途中に挿入される場合があります.

ガーベジコレクション処理の結果ログに対するクラスのアンロード情報出力を抑止する場合は、次のオプションを指定してください。

GC処理の結果ログに対するクラスのアンロード情報出力を抑止するオプション

```
-XX:-ClassUnloadingInfo
```

ログ出力量の増加

本オプションの指定により、ログ出力が増大します。
本オプションを指定する場合は、ログ出力量についての注意が必要です。

■ ログの見方

図6から図8に、GC処理の結果ログとして出力される情報の拡張形式の出力例を示します。

図6 GC処理の結果ログとして出力される情報の拡張形式の出力例

```
1.495: [Full GC*, [SplitEden : 384K->0K(704K)], [Tenured : 47835K->32752K(47872K)] 48219K->32752K(48576K), [Perm : 4081K->4081K(16384K)], 0.6623532 secs]
```

この出力情報から、以下のことが分かります。

- Java VMが起動されてから1.495秒後にFullGC処理が実行された。
- 使用されているGC処理はFJGCである。
- GC処理後のNew世代領域の大きさは704KBである。
- GC処理により、New世代領域に存在するオブジェクト量は384KBから0KBになった。
  （不要オブジェクトが削除され、また必要に応じてOld世代領域へ生存オブジェクトが移動した）
- GC処理後のOld世代領域の大きさは47872KBである。
- GC処理により、Old世代領域に存在するオブジェクト量は47835KBから32752KBになった。
  （不要オブジェクトが削除され、また必要に応じてNew世代領域から生存オブジェクトが移動してきた）
- GC処理後のメモリ割り当てプールの大きさは48576KBである。
- GC処理により、メモリ割り当てプールに存在するオブジェクト総量は48219KBから32752KBになった。
  （不要オブジェクトが削除された）
- GC処理後のPermanent世代領域の大きさは16384KBである。
- GC処理により、Permanent世代領域に存在するオブジェクト量は変化していない。
- GC処理に要した時間は0.6623532秒である。

図7 GC処理の結果ログとして出力される情報の拡張形式の出力例(CMS付きパラレルGCの場合-1)

```
150.207: CMS start
150.208: [CMS initial-mark, [ParNew : 1863K->1863K(14784K)], [CMS : 53791K->53791K(65536K)] 55654K->55654K(80320K), [CMS Perm : 4664K->4664K(16384K)], 0.0030212 secs]
150.351: [GC, [ParNew : 14782K->1598K(14784K)], [CMS : 53791K->57981K(65536K)] 68573K->59579K(80320K), [CMS Perm : 4664K->4664K(16384K)], 0.0328537 secs]
150.466: [CMS remark, [ParNew : 8277K->8277K(14784K)], [CMS : 57981K->57981K(65536K)] 66258K->66258K(80320K), [CMS Perm : 4664K->4664K(16384K)], 0.0097905 secs]
150.549: [GC, [ParNew : 14782K->1598K(14784K)], [CMS : 50163K->54371K(65536K)] 64946K->55969K(80320K), [CMS Perm : 4664K->4664K(16384K)], 0.0303271 secs]
150.583: CMS stop(00), [CMS : 57981K->54200K(65536K)], 0.3753996 secs
```

この出力情報から、以下のことが分かります。

- 使用されているGC処理はCMS付きパラレルGCである。
- Java VMが起動されてから150.207秒後にCMS-GC処理が開始され、150.583秒後に終了した。
- 終了したCMS-GCにより不要オブジェクトが回収された。
- CMS-GC処理後のOld世代領域の大きさは65536KBである。
- CMS-GC処理により、Old世代領域に存在するオブジェクト量は57981KBから54200KBになった。
- CMS-GC処理に要した時間は0.3753996秒である。
- CMS-GC処理中にNew世代領域用GC処理が実行された。またその実行開始は、遅延している可能性がある。

図8 GC処理の結果ログとして出力される情報の拡張形式の出力例(CMS付きパラレルGCの場合-2)

```
137.803: CMS start
137.804: [CMS initial-mark, [ParNew : 206690K->206690K(314560K)], [CMS : 655731K->655731K(699072K)] 862421K->862421K(1013632K), [CMS Perm : 3892K->3892K(16384K)], 0.4101250 secs]
139.069: [GC, [ParNew : 279616K->34943K(314560K)], [CMS : 655731K->673280K(699072K)] 935347K->708223K(1013632K), [CMS Perm : 3892K->3892K(16384K)], 0.2177910 secs]
142.140: CMS stop-req
142.501: CMS stop(11), 4.6984060 secs
142.501: [Full GC, [ParNew : 314559K->0K(314560K)], [CMS : 673280K->657037K(699072K)] 987839K->657037K(1013632K), [CMS Perm : 3892K->3892K(16384K)], 1.8642510 secs]
```

この出力情報から、以下のことが分かります。

- 使用されているGC処理はCMS付きパラレルGCである。
- Java VMが起動されてから137.803秒後にCMS-GC処理が開始され、142.501秒後に終了した。
- Java VMが起動されてから142.140秒後にFullGC要求があり、そのため実行中のCMS-GCが終了した。
- 終了したCMS-GCによる不要オブジェクト回収は実行されていない。
- CMS-GC処理の終了要求時点(142.140)からCMS-GC処理の実行が終了(142.501)するまでの0.361秒、および142.501に実行開始したFullGCの実行時間1.8642510秒の合計2.225251秒の間、Javaアプリケーション動作が停止された。

7.2.5 メモリ領域不足事象発生時のメッセージ出力機能の強化

FJVMでは、メモリ領域不足事象発生時に出力されるメッセージ情報の強化を行なっています。
これにより**FJVM**では、メモリ領域不足事象が発生した場合に、java.lang.OutOfMemoryErrorの例外メッセージ情報に加え、不足した領域の種別情報を図1の形式で出力します。

図1 メモリ領域不足事象が発生した場合に出力される不足した領域の種別情報

```
The memory was exhausted area_name
Java heap size / max Java heap size = heap_size / max_heap_size
Java perm size / max Java perm size = perm_size / max_perm_size
```

area_name: メモリ領域不足事象が発生した領域の名前や領域不足となったオブジェクトの要求サイズ(不足領域情報)を表示します。
不足領域情報としては以下の項目があります。

- **on Java heap space. : requested <NNNN> bytes**
  NNNNバイトのオブジェクト生成要求において、メモリ割り当てプール(New世代領域またはOld世代領域)に対してメモリ領域不足事象が発生した場合です。
  なお、JDK/JRE 5.0、6のエルゴノミクス機能によるメモリ領域不足事象の検出機能が有効な場合で、かつ当該機能によりメモリ領域不足事象を検出した場合、この項目になる場合があります。
- **on Java heap space. : requested <NNNN> bytes (in critical section)**
  意味は**「on Java heap space. : requested <NNNN> bytes」**と同じですが、メモリ領域不足事象発生時、クリティカルセクション状態でGC処理の実行が抑止されていたことを示しています。
- **on Java perm space. : requested <NNNN> bytes**
  NNNNバイトのオブジェクト生成要求において、Permanent世代領域に対してメモリ領域不足事象が発生した場合です。
  なお、JDK/JRE 5.0、6のエルゴノミクス機能によるメモリ領域不足事象の検出機能が有効な場合で、かつ当該機能によりメモリ領域不足事象を検出した場合、この項目になる場合があります。
- **on Java perm space. : requested <NNNN> bytes (in critical section)**
  意味は**「on Java perm space. : requested <NNNN> bytes」**と同じですが、メモリ領域不足事象発生時、クリティカルセクション状態でGC処理の実行が抑止されていたことを示しています。
- **(なし)**
  スタックやヒープなど、Javaヒープ以外の領域に対してメモリ領域不足事象が発生した場合です。特にjava.lang.OutOfMemoryErrorの例外メッセージ情報が“java.lang.OutOfMemoryErrorがスローされた場合”の「ユーザ空間不足」または「仮想メモリ不足」の場合に出力される形式の場合は、スタックやヒープなど、Javaヒープ以外の領域に対してメモリ領域不足事象が発生した場合と断定できます。
  またはJavaアプリケーション実行時における配列生成式の評価の段階で、配列オブジェクトの長さ(配列要素の数)から、当該配列オブジェクトを割り当て

るための領域が十分でないと評価された場合（配列の長さ（配列要素の数）が大きすぎ、配列オブジェクトとしての大きさが2ギガバイト程度もしくはそれ以上の大きさになる配列の定義がある場合）。
またはクラスのロード処理でメモリ不足が発生した場合。
なお、JDK/JRE 6のエルゴノミクス機能によるメモリ領域不足事象の検出機能が有効な場合で、かつ当該機能によりメモリ領域不足事象を検出した場合、この項目になる場合があります。

heap_size: メモリ領域不足事象が発生した際に使用中となっているメモリ割り当てプールのサイズ(単位:byte)。

max_heap_size: 利用可能なメモリ割り当てプールの最大サイズ(単位:byte)。

perm_size: メモリ領域不足事象が発生した際に使用中となっているPermanent世代領域のサイズ(単位:byte)。

max_perm_size: 利用可能なPermanent世代領域の最大サイズ(単位:byte)。

メモリ領域不足事象が発生した際に出力される各領域の使用中サイズ(heap_size、perm_size)には、メモリ領域不足の原因となったオブジェクトの大きさは含まれません。
そのため巨大サイズのオブジェクト生成要求などによりメモリ領域不足事象が発生した場合には、「最大サイズ」と「使用中サイズ」の差が大きい場合（空き領域がたくさんあるように見える場合）がありますので注意してください。

メモリ割り当てプールに対してメモリ領域不足事象が発生した場合に出力されるheap_sizeの値は、New世代領域での使用中サイズとOld世代領域での使用中サイズの合計値です。
New世代領域とOld世代領域は別々のオブジェクト格納域として管理・制御されますから、max_heap_sizeとheap_sizeの差の大きさが、そのまま生成要求できるオブジェクトの最大サイズにはなりませんので注意してください。

図2 メモリ領域不足事象が発生した場合に出力されるメッセージ出力例

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
The memory was exhausted on Java heap space. : requested 4016 bytes
  Java heap size / max Java heap size = 495974032 / 536870912
  Java perm size / max Java perm size = 1678376 / 67108864
```

図2の出力例の場合、4016バイトのオブジェクト生成要求において、メモリ割り当てプールに対してメモリ領域不足が発生したことを確認することができます。

7.2.6 Java VM終了時における状態情報のメッセージ出力機能

特別なメッセージ出力などがないまま、Javaプロセスが予想外の状態で終了してしまった場合の原因の1つとして、Javaアプリケーションが以下のいずれかの処理を実行した場合が考えられます。

- java.lang.System.exit()を予想外の箇所で実行した
- java.lang.Runtime.exit()を予想外の箇所で実行した
- java.lang.Runtime.halt()を予想外の箇所で実行した

以降は、java.lang.System.exit()(以降、System.exit()と略する)で代表して説明します。

System.exit()の実行によりJavaアプリケーションが明示的にJavaプロセスを終了させた場合、Java VM側から見ると正常な仕様動作であるため、Java VMとして特別なメッセージ出力などは行いません。このため、ソースがないなど、内部処理動作の詳細が不明なJavaアプリケーションが予想外の状態で終了した場合には、ソース確認などが行えないため、System.exit()が実行されたかどうかを確認することができません。

このため**FJVM**では、Javaプロセス終了時にSystem.exit()が実行されたかどうかを確認可能にするための機能を、「**Java VM終了時における状態情報のメッセージ出力機能**」として実装しています。

Java VM終了時における状態情報のメッセージ出力機能は、図1のオプションを指定した場合に有効となります。

図1 Java VM終了時における状態情報のメッセージ出力機能を有効にするオプション

```
-XX:+VMTerminatedMessage
```

本機能が有効な場合、JavaプロセスがSystem.exit()の実行で終了した場合には、System.exit()を実行したスレッドのスタックトレース情報などが、図2のような形で標準出力へ出力されます。スタックトレース情報の出力の有無および内容を確認することにより、System.exit()実行の有無を確認することができます。

なお、標準出力への出力結果は、**FJVM**のログ情報としてファイルへも格納されます。ファイル名や格納先はJava VM異常終了時のログ出力時と同じです。“7.2.11 FJVMログ”を参照してください。

そして、本機能が有効な場合で、かつJavaプロセス終了時に図2のようなスタックトレースの情報が出力されなかった場合(「#### JavaVM terminated: …」から始まるメッセージ出力だけの場合)は、System.exit()が使用されず、Javaアプリケーション側の制御論理として終了したと考えられます。

また、本機能が有効な場合で、かつJavaプロセス終了時に図2のような情報が何も出力されなかった場合は、ネイティブモジュールの中からCランタイムのexit()関数呼び出しによりJavaプロセスが終了したなど、別原因によるJavaプロセスの終了だと考えられます。

図2 Java VM終了時における状態情報のメッセージ出力機能による出力例

【JDK/JRE 1.4または5.0の場合】

```
Thread dump at JVM_Halt(status code=1234):
"main" prio=5 tid=0x00286240 nid=0x394 runnable [6f000..6fc10]
                        at java.lang.Shutdown.halt(Native Method)
                        at java.lang.Shutdown.exit(Shutdown.java:211)
      - locked <0x16b66d48> (a java.lang.Class)
                        at java.lang.Runtime.exit(Runtime.java:90)
                        at java.lang.System.exit(System.java:715)
                        at JVM_Halt.main(JVM_Halt.java:5)

#### JavaVM terminated: Java HotSpot(TM) Server VM
(1.5.0_FUJITSU_MODIFIED-B** mixed mode), [pid=21388] TimeMillis=1169451586436
Time=Mon Jan 22 16:39:46 2007
```

【JDK/JRE 6の場合】

```
Thread dump at JVM_Halt(status code=1234):
"main" prio=3 tid=0x00030000 nid=0x2 runnable [0xfe5ff000..0xfe5ffd80]
   java.lang.Thread.State: RUNNABLE
                        at java.lang.Shutdown.halt0(Native Method)
                        at java.lang.Shutdown.halt(Shutdown.java:105)
       - locked <0xfa81e7d0> (a java.lang.Shutdown$Lock)
                        at java.lang.Shutdown.exit(Shutdown.java:179)
       - locked <0xf3dc8dd8> (a java.lang.Class for java.lang.Shutdown)
                        at java.lang.Runtime.exit(Runtime.java:90)
                        at java.lang.System.exit(System.java:906)
                        at JVM_Halt.main(JVM_Halt.java:5)

#### JavaVM terminated: Java HotSpot(TM) Server VM
(11.3.02_FUJITSU_MODIFIED-B** mixed mode), [pid=29500] TimeMillis=1243580170483
Time=Fri May 29 15:56:10 2009
```

図2の出力例の場合、JVM_Halt.main()がSystem.exit()を実行し、その結果としてJavaプロセスが終了したことを確認することができます。

JDK/JRE 6からスタックトレース情報にスレッドの状態を表す情報(トレース情報の前の行)が表示されるようになりました。

図中の例では「java.lang.Thread.State: RUNNABLE」となっていますが、"RUNNABLE"の部分が、"NEW"、"TIMED_WAITING (sleeping)"、"WAITING (on object monitor)"、"TIMED_WAITING (on object monitor)"、"WAITING (parking)"、"TIMED_WAITING (parking)"、"BLOCKED (on object monitor)"、"TERMINATED"、"UNKNOWN"などの表示の場合があります。

この情報は**Java VM終了時における状態**の判定には使用しません。

7.2.7 java.lang.System.gc()実行時におけるスタックトレース出力機能

Javaアプリケーションが以下のJavaメソッドを頻繁に実行すると、Java VMに負荷がかかり、アプリケーションの応答性能を低下させる要因になることがあります。

- java.lang.System.gc()
- java.lang.Runtime.gc()

以降、java.lang.System.gc()(以降、System.gc()と略する)で代表して説明します。

FJVMでは、Javaアプリケーション実行時にSystem.gc()メソッドの実行状態の確認ができるように、当該メソッドを実行したJavaスレッドのスタックトレースを出力する機能を「**java.lang.System.gc()実行時におけるスタックトレース出力機能**」として実装しています。
java.lang.System.gc()実行時におけるスタックトレース出力機能は、図1のオプションを指定した場合に有効となります。

本機能が有効な場合、Javaアプリケーション内でSystem.gc()メソッドが実行された場合には、当該メソッドを実行したJavaスレッドのスタックトレース情報が、図2のような形で標準出力へ出力されます。
なお、標準出力への出力結果は、**FJVM**のログ情報としてファイルへも格納されます。また“-verbose:gc”オプション指定などでガーベジコレクション処理の結果ログ出力を指定していた場合は、java.lang.System.gc()実行時におけるスタックトレース出力後に出力された結果ログ出力も、合わせてファイルへ格納されます。ファイル名や格納先はJava VM異常終了時のログ出力時と同じです。“7.2.11 FJVMログ”を参照してください。

図1 java.lang.System.gc()実行時におけるスタックトレース出力機能を有効にするオプション

```
-XX:+PrintJavaStackAtSystemGC
```

図2 java.lang.System.gc()実行時におけるスタックトレース出力機能による出力例

【JDK/JRE 1.4または5.0の場合】

```
"main" prio=5 tid=0x0002d0a8 nid=0x1 runnable [ffbee000..ffbeee7c]
                        at java.lang.Runtime.gc(Native Method)
                        at java.lang.System.gc(System.java:737)
                        at SystemGC.main(SystemGC.java:8)

 "main" prio=5 tid=0x0002d0a8 nid=0x1 runnable [ffbee000..ffbeee7c]
                        at java.lang.Runtime.gc(Native Method)
                        at SystemGC.foo(SystemGC.java:4)
                        at SystemGC.main(SystemGC.java:10)
```

【JDK/JRE 6の場合】

```
"main" prio=10 tid=0x087c1c00 nid=0xd2a runnable [0xb75ab000..0xb75ab214]
   java.lang.Thread.State: RUNNABLE
                        at java.lang.Runtime.gc(Native Method)
                        at java.lang.System.gc(System.java:928)
                        at SystemGC.main(SystemGC.java:8)

"main" prio=10 tid=0x087c1c00 nid=0xd2a runnable [0xb75ab000..0xb75ab214]
   java.lang.Thread.State: RUNNABLE
                        at java.lang.Runtime.gc(Native Method)
                        at SystemGC.foo(SystemGC.java:4)
                        at SystemGC.main(SystemGC.java:10)
```

図2の出力例の場合、SystemGC.mainからjava.lang.System.gc()が実行され、SystemGC.fooからjava.lang.Runtime.gc()が実行されたことを確認することができます。

参考

JDK/JRE 6からスタックトレース情報にスレッドの状態を表す情報（トレース情報の前の行）が表示されるようになりました。

図中の例では「java.lang.Thread.State: RUNNABLE」となっていますが、"RUNNABLE"の部分が、"NEW"、"TIMED_WAITING (sleeping)"、"WAITING (on object monitor)"、"TIMED_WAITING (on object monitor)"、"WAITING (parking)"、"TIMED_WAITING (parking)"、"BLOCKED (on object monitor)"、"TERMINATED"、"UNKNOWN"などの表示の場合があります。

7.2.8 スタックオーバーフロー検出時のメッセージ出力機能

Javaプロセスが不当なメモリアクセスにより異常終了した場合の原因の1つとして、スレッドに対するスタックのサイズ不足、すなわちスタックオーバーフローの発生が考えられます。

FJVMでは、スタックオーバーフローが原因と考えられる不当なメモリアクセスによりJavaプロセスが異常終了した場合、その旨を原因調査情報としてFJVMログへ出力する機能を、「**スタックオーバーフロー検出時のメッセージ出力機能**」として実装しています。
FJVMログの見方については、“7.2.11 FJVMログ”を参照してください。

スタックオーバーフロー発生の原因が、Java APIで生成されたスレッドに対するスタックのサイズにある場合は、“7.5.1 スタックのチューニング”を参照して、Java APIで生成されるスレッドに対するスタックのサイズをチューニングしてください。

注意 **検出対象となるスレッド**

本機能でスタックオーバーフローの検出対象となるスレッドは、原則Java APIで生成されたスレッドです。
次のスレッドは、本機能による検出対象スレッドとはなりません。

- ネイティブモジュールからOSのAPIを直接使用して生成されたスレッド
- mainメソッドを実行するスレッド(Java APIで生成されたスレッドではないため)

  スタックオーバフローの検出には、OSの機能を利用しています。ご使用中のOSがWindowsの場合は、上記スレッドの場合であっても、そのスレッドで実行されるJavaメソッドの中から呼び出されたネイティブメソッド内で直接発生したスタックオーバーフローについては、本機能による検出の対象となります。

注意 **注意事項** Windows

スタックオーバーフローが発生しても、OS側から**FJVM**側の処理へ制御が渡らないことがあります。その場合は**FJVM**ログが出力されません。
OSの制御処理がワトソン博士へ直接例外制御を渡した場合には、ワトソン博士が出力するログファイルを確認してください。ワトソン博士の説明は、“7.3.4 クラッシュダンプ・コアダンプ”を参照してください。

なお、スタックオーバーフロー発生時のスタック残量が少ない場合には、以下の状態でJavaアプリケーションが終了する場合があるので注意してください。

- 以下のメッセージが標準出力へ出力されただけで、FJVMログが出力されないままJavaアプリケーションが終了してしまうことがあります。
  スタックオーバーフロー検出に関するFJVMログが出力されない場合であっても、以下のメッセージが出力された場合は、スタックオーバーフローが発生したことを示します。

  ```
  An unrecoverable stack overflow has occurred.
  ```

  または、

  ```
  Register stack overflow, addr:??, stack_base:??
  ```

  (??=16進数) [Windows Server(R) for Itanium-based Systemsの場合のみ]

- スタックオーバーフロー発生例外に対する制御が、OS側からFJVM側の処理およびワトソン博士のどちらにも渡らず、そのままJavaアプリケーションが終了してしまうことがあります。
  その場合は、スタックオーバーフロー発生を検出することができません。

RHEL-AS4(IPF)/RHEL5(IPF)における注意事項 Linux

Linux for Itanium用のネイティブモジュールにおいて、メモリアクセスを伴わない再帰処理(例:スタックポインタの更新処理だけが行われるような再帰処理)でスタックオーバーフローが発生した場合には、OS側から**FJVM**側の処理へ正しく制御が渡りません。その場合は、スタックオーバーフロー発生を検出することができません。

7.2.9 コンパイラ異常発生時の自動リカバリ機能

Java VMはJavaアプリケーションとして実行されるJavaメソッドに対して必要に応じて自動的にコンパイル処理を行いますが、コンパイル処理を行っている際にコンパイラ内で何らかの異常が発生すると、当該Javaメソッドに対するコンパイル処理だけでなくJava VMとしての動作も異常状態として停止させてしまう場合があります。

FJVMでは、コンパイラ内で何らかの異常が発生した場合に自動的にリカバリ処理を行い、Java VMとしての動作を継続させる機能を「コンパイラ異常発生時の自動リカバリ機能」として実装しています。

なお、本機能によるリカバリ処理が行なわれた際にコンパイル対象となっていたJavaメソッドは、以降、コンパイル処理の対象とはなりません。当該Javaメソッドについてはコンパイルされず、インタプリタモードのままJavaアプリケーションとしての実行が継続されます。

また、本機能は**FJVM**の内部処理として動作する機能であるため、コンパイラ内で何らかの異常が発生してもリカバリ処理が正常に行われた場合には、外部に対する通知などは何も行いません。リカバリ処理が正常に行われた場合でもコンパイラ内で何らかの異常が発生したことを情報として受け取る場合には、図1のオプションを指定してください。

図1のオプションを指定した場合、リカバリ処理の情報が図2の形式で標準出力に出力されます。

図1 リカバリ処理実施後に通知を受け取るオプション

```
-XX:+PrintCompilerRecoveryMessage
```

図2 リカバリ処理実施後に通知される情報

【JDK/JRE 1.4の場合】

```
CompilerRecovery: Information: The compilation order was canceled in compiling method method_name
Reason for the cancellation: reason [code:c, addr:xxxxxxxx]
```

【JDK/JRE 5.0または6場合】

```
CompilerRecovery: Information:The compilation was canceled for method method_name
Reason for the cancellation: reason [code:c, addr:xxxxxxxx]
```

method_name: コンパイル処理で異常が発生した際にコンパイル対象となっていたJavaメソッドの名前。

reason: コンパイル処理で発生した異常の原因情報。原因情報として以下の項目があります。

- **assert:**
  コンパイル処理で内部処理矛盾を検出した
- **error:**
  コンパイル処理で何らかの誤りを検出した
- **stack overflow:**
  コンパイル処理でスタックオーバーフローを検出した

c: 異常コード。

xxxxxxxx: コンパイル処理で異常が発生した際のアドレス。

Windows Server(R) for Itanium-based Systemsでの注意事項

Windows

コンパイル処理でスタックオーバーフローが発生した場合、その発生例外に対する制御がOS側から**FJVM**側の処理へ渡らず、そのままJavaアプリケーションが終了してしまうことがあります。
その場合は、本機能によるリカバリ処理は行われません。

なおその際、以下のメッセージが標準出力へ出力される場合があります。

```
An unrecoverable stack overflow has occurred. (compiler thread)
```

7.2.10 長時間コンパイル処理の検出機能

Java VMはJavaアプリケーションとして実行されるJavaメソッドに対して必要に応じて自動的にコンパイル処理を行い、通常は極短時間でその処理を完了します。
しかし、コンパイル処理自身や同じJavaプロセス内で動作させている他の処理の障害などによりCPU資源が占有され続けてしまうと、数分を経過してもコンパイル処理が終了しない場合が考えられます。このような状態が継続すると、システム全体に対して悪影響を与える可能性が考えられます。
このため、**FJVM**では、各Javaメソッドのコンパイル処理に要している時間を監視し、コンパイル処理で必要と考えられる程度の時間を経過してもコンパイル処理が終了していない場合には、Javaプロセス内の処理で何らかの問題が発生していると判断し、当該Javaプロセスを強制的に終了させる機能を「**長時間コンパイル処理の検出機能**」として実装しています。

本機能は、図1のオプションでコンパイル処理に対する監視時間(コンパイル処理に要する時間の上限値)を指定した場合に有効となります。ただし、オプション値として「0」を指定した場合には、本機能は有効となりません。
図1のオプションで指定された時間を超過してもコンパイル処理が終了していない場合、本機能はJavaプロセス内の処理で何からの問題が発生していると判断し、当該Javaプロセスを強制的に終了させます。

図1 長時間コンパイル処理の検出機能を有効にするオプション

```
-XX:CompileTimeout=<nn>
```

<nn>には、本機能による異常有無の判断条件とされるコンパイル処理に要する時間の上限値(単位:秒)を指定します。デフォルト値は「0」であり、本機能は無効となっています。
なお、本機能による時間監視の最小単位は30秒であるため、その単位での時間誤差があります。

なお、本機能によってJavaプロセスを強制終了する場合、**FJVM**は図2のメッセージを標準出力に出力してから終了します。また、本機能によるJavaプロセスの強制終了時にはコアダンプも出力されます。

図2 長時間コンパイル処理の検出機能によるJavaプロセス強制終了時の出力メッセージ

```
CompilerRecovery: Information: CompilerRecovery got the VM aborted
because the compiler thread(nnnnnnnn) has not completed.
(compiling method: method_name)
```

nnnnnnnn: コンパイラスレッドの内部識別子
method_name: 本機能によるチェックでJavaプロセス内に異常が検出された際にコンパイル対象となっていたJavaメソッドの名前

長時間コンパイル処理の検出機能に対する注意事項

何らかの要因によりJavaプロセス内のコンパイル処理へCPU資源が十分に割り当てられず、コンパイル処理自体が進んでいない場合でも、コンパイル処理開始から-XX:CompileTimeoutオプションで指定された監視時間を超過した場合には、本機能による当該Javaプロセスの強制終了となります。
このため、当該Javaプロセスを実行するシステムのCPU負荷が高い場合には、コンパイル処理に対してCPU資源が十分に割り当てられず、この結果として本機能による強制終了が発生する可能性が考えられます。
本機能による強制終了が発生した場合には、まず以下の事項を確認してください。

- 当該Javaプロセスを実行しているシステムのCPU資源量は十分か。
- 当該Javaプロセス以外の他のプロセスでCPU資源が占有されていないか。

・ “-XX:CompileTimeout=0”を指定した場合に本機能による強制終了が回避され、かつ、当該Javaプロセスが正常に終了する、または未負荷時に正常なアイドル状態に遷移するか。

上記事項に合致する場合は、コンパイル処理に対してCPU資源が十分に割り当てられなかった結果として発生した強制終了と考えられます。
長時間コンパイル処理の検出機能を有効にしてこの状態が発生した場合には、“-XX:CompileTimeout”オプションで指定する監視時間として、より大きな値に設定する形で調整してください。

長時間コンパイル処理の検出機能に対する監視メッセージの出力

図3のオプションを指定した場合、Javaメソッドのコンパイル処理において1分が経過すると、図4の形式で監視メッセージが出力されます。
その後は、30秒経過するごとに同じ監視メッセージが出力されます。
なお、本機能による時間監視の最小単位は30秒であるため、その単位での時間誤差があります。

図3 長時間コンパイル処理の検出機能の監視メッセージ出力を有効にするオプション

```
-XX:+PrintCompilerRecoveryMessage
```

図4 長時間コンパイル処理の検出機能が出力する監視メッセージ

```
CompilerRecovery: Information: The compiler thread(0xnnnnnnnn) might not return from compiling
method method_name.
```

nnnnnnnn: コンパイラスレッドの内部識別子
method_name: 本機能によるチェックで検出された際にコンパイル対象となっていたJavaメソッドの名前

7.2.11 FJVMログ

何らかの原因でJavaプロセスが異常終了した場合、**FJVMログ**が出力されます。
Javaプロセスが異常終了した原因の調査のために、この**FJVMログ**を活用することができます。

■FJVMログの出力先

FJVMログは、Javaプロセスのカレントディレクトリに、図1のファイル名で出力されます。

図1 FJVMのファイル名

```
fjvm_pid***.log (***は異常終了したJavaプロセスのプロセスID)
```

IJServer使用時のカレントディレクトリの詳細は、“J2EE ユーザーズガイド”の“J2EEアプリケーションが運用される環境(IJServer)”を参照してください。

■FJVMログの調査

FJVMログとしてJavaプロセス異常終了時の各種情報が格納されますが、その中から以下の情報を原因調査用の情報として使用することができます。

・ 異常終了箇所の情報
・ 異常終了時のシグナルハンドラ情報(Solaris版/Linux版)
・ 異常終了時のJavaヒープに関する情報

各情報の内容について、以下で説明します。

7.2.11.1 異常終了箇所の情報

異常終了箇所に関する図1の情報が確認できます。

図1 異常終了箇所に関する情報

1. 異常終了時に発生した例外に関する情報(シグナルコードおよび例外発生アドレス)
   「Unexpected Signal :」から始まる情報です。
2. 異常終了した関数名(実際には異常終了したアドレスに一番近いシンボル名)
   「Function name=」から始まる情報です。
3. 異常終了した関数を含むライブラリ名
   「Library=」から始まる情報です。
4. 異常終了時のJavaスレッドのスタックトレース
   「Current Java thread:」から始まる情報です。
5. 異常終了時のダイナミックライブラリ一覧
   「Dynamic libraries:」から始まる情報です。
6. 発生時間
   「Local Time =」から始まる情報です。

■調査手順

まず、図1の1～3の情報で異常終了した関数を特定し、実行しているJavaアプリケーションから呼び出す関数かどうかを確認します。ただし、図1の2の異常終了した関数名として出力される名前は、異常終了したアドレスに一番近いシンボル名情報であるため、実際に異常終了した関数とは別の名前が出力されている場合がありますので注意してください。
そして、実行しているJavaアプリケーションが使用する関数の場合には、当該関数使用に際して何らかの問題がないか確認します。
実行しているJavaアプリケーションで使用していない関数の場合には、図1の4のスタックトレースを調査します。

スタックトレース情報の最初のメソッドがネイティブメソッドだった場合（メソッド名の後ろに「(Native Method)」が付加されている場合）はJNI処理に関係した問題である可能性が高いため、スタックトレース情報で出力された処理のJNI処理に関わる制御で何らかの問題がないか確認します。

また異常終了した関数を含むライブラリ名が利用者作成のライブラリである場合は、利用者側作成のライブラリ内の問題である可能性が高いため、当該ライブラリ内の処理および当該ライブラリを呼び出すJNI処理に何らかの問題がないか確認します。

スタックトレースの調査方法は、“7.3.2 スタックトレース”を参照してください。

■スタックオーバーフローの検出

図1の1の異常終了時に発生した例外に関する情報に、図2のシグナルコードの表記がある場合、例外が発生したスレッドでスタックオーバーフローが発生した（図2の1、3の表記）、または発生した可能性がある（図2の2、4の表記）ことを示しています。
この場合、例外が発生したスレッドに対するスタックのサイズを大きくすることで問題が解決する可能性があります。

スタックオーバーフロー発生の原因が、Java APIで生成されたスレッドに対するスタックのサイズにある場合は、“7.5.1 スタックのチューニング”を参照して、Java APIで生成されるスレッドに対するスタックのサイズをチューニングしてください。

図2 スタックオーバーフローを示すシグナルコード

Windows
1)「EXCEPTION_STACK_OVERFLOW」
2)「EXCEPTION_ACCESS_VIOLATION (Stack Overflow ?)」

Solaris Linux
3)「SIGSEGV (Stack Overflow)」
4)「SIGSEGV (Stack Overflow ?)」

注意 ワトソンログの分析 Windows

スタックオーバーフローが原因で発生した異常終了の場合、OS側から**FJVM**側の処理へ制御が渡らず、そのままワトソン博士へ制御が渡されることがあります。この場合は、**FJVMログ**が出力されないため、ワトソン博士のログファイルを確認してください。
ワトソンログに図3の例外番号が出力されている場合には、スタックオーバーフローが原因と考えられます。
なお、ワトソン博士の説明は、“7.3.4 クラッシュダンプ・コアダンプ”を参照してください。

図3 スタックオーバーフローを示す例外番号

c00000fd (スタックオーバーフロー)

7.2.11.2 異常終了時のシグナルハンドラ情報 Solaris Linux

Java VMの実行制御で必要となる“表7.3 Java VMの制御で必要となるシグナル”の各シグナルに対するシグナルハンドラ情報が確認できます。

表7.3 Java VMの制御で必要となるシグナル

Solaris版Java VM	Linux版Java VM
SIGSEGV SIGPIPE SIGBUS SIGILL SIGFPE INTERRUPT_SIGNAL (デフォルトはSIGUSR1) ASYNC_SIGNAL (デフォルトはSIGUSR2) SIGQUIT (注1) SIGINT (注1) SIGHUP (注1) SIGTERM (注1) SIGXFSZ (注3)	SIGSEGV SIGPIPE SIGBUS SIGILL SIGFPE INTERRUPT_SIGNAL (デフォルトはSIGUSR1) (注2) SR_SIGNUM (デフォルトはSIGUSR2) SIGQUIT (注1) SIGINT (注1) SIGHUP (注1) SIGTERM (注1) SIGXFSZ (注3)

注1) -Xrsオプションで操作対象となるシグナルです。
注2) Java VMでリザーブしているシグナルです。
注3) JDK/JRE 6のJava VMで使用されているシグナルです。
これらのシグナルに関するシグナルハンドラ情報は出力しません。

シグナルハンドラ情報として、以下の情報が出力されています。

- 登録されているシグナルハンドラのアドレス
- 登録されているシグナルハンドラがJava VMで登録したシグナルハンドラかどうかの正否情報(Java VM以外の処理で登録されたシグナルハンドラの場合には、該当するシグナルハンドラ情報の行に“(not in VM)”が出力されます)

“表7.3 Java VMの制御で必要となるシグナル”のシグナルハンドラがJava VM以外の処理で登録されていた場合、Java VMは正常に動作しません。この場合、当該シグナルハンドラを登録しないようにアプリケーションを修正してください。

7.2.11.3 異常終了時のJavaヒープに関する情報

異常終了時のJavaヒープの使用状況が確認できます。

Javaヒープのサイズによる異常終了の場合、異常終了時にどのJavaヒープの枯渇により異常終了が発生したのかが確認できます。

7.2.11.4 出力例と調査例

ここでは、Solaris版JDK/JRE 5.0での出力例を元に説明します。

```
--------------------------------------------------------------------------------
#### Java VM: Java HotSpot(TM) Server VM (1.5.0_FUJITSU_MODIFIED-B**[*****] mixed mode)
>>>> Logging process start. [pid=27758] Time=Fri Jan 12 20:37:57 2007
```

(1) 異常終了箇所の情報 異常終了箇所に関する情報が確認できます。 libjvm.soのsysThreadAvailableStackWithSlack関数の近くでSIGSEGV(メモリアクセスで不正なセグメ

ントを参照)が発生しています。
本例の場合、Java VM内で異常が発生していると判断します。
異常終了箇所がJavaアプリケーション内でないため、異常発生時のスタックトレース情報を調査します。
本例の場合、com.appli.ap.business.AL02ABB00000.toStringの延長で不正なアクセスが発生しているので、そこからAL02ABB00000.javaの489行目で不正なアクセスを招きそうな箇所がないか調べます。

```
Unexpected Signal : SIGSEGV [0xb] occurred at PC=0xff092068, pid=27758, nid=1
Function name=sysThreadAvailableStackWithSlack
Library=/opt/FJSVawjbk/jdk5/jre/lib/sparc/fjvm/libjvm.so

Current Java thread:
0xfb8e2850 - 0xfb8e4b7c at com.appli.ap.business.AL02ABB00000.toString(AL02ABB00000.java:489)
0xfb8e2850 - 0xfb8e4b7c at com.appli.ap.business.AL02ABB00000.toString(AL02ABB00000.java:520)
                         at java.lang.String.valueOf(String.java:1942)
                         at java.lang.StringBuffer.append(StringBuffer.java:365)
- locked <f6db38d8> (a java.lang.StringBuffer)
                         at com.appli.ap.business.AL02ABB25201.doExecute(AL02ABB25201.java:774)
                         at com.appli.ap.formula.AFCC6842.doDelegate(AFCC6842.java:221)
                         at com.appli.ap.formula.ejb.session.AFSF6801.doExecuteOrdinarily(AFSF6801.java:381)
                         at
com.appli.ap.formula.ejb.session.FJAFSF6801_AFSF6801RemoteImpl.doExecuteOrdinarily(FJAFSF6801_AFSF6801RemoteImpl.java:
464)
- locked <df672838> (a com.appli.ap.formula.ejb.session.FJAFSF6801_AFSF6801RemoteImpl)
                         at
com.appli.ap.formula.ejb.session._FJAFSF6801_AFSF6801RemoteImpl_Tie._invoke(_FJAFSF6801_AFSF6801RemoteImpl_Tie.java:76)
0xfb98c930 - 0xfb98cc68 at com.fujitsu.ObjectDirector.CORBA.ServerRequest.call_invoke(ServerRequest.java:961)
                         at com.fujitsu.ObjectDirector.PortableServer.POA.MsgRecv(POA.java:2578)
                         at com.fujitsu.ObjectDirector.PortableServer.POAManager.MsgRecv(POAManager.java:1061)
                         at com.fujitsu.ObjectDirector.PortableServer.POAnc.MsgRecv(POAnc.java:163)

Dynamic libraries:
0x10000 /opt/FJSVawjbk/jdk5/bin/java
0xff370000 /usr/lib/libthread.so.1
0xff3fa000 /usr/lib/libdl.so.1
～～～～～～～～
略
～～～～～～～～
0xbef70000 /lib/libgen.so.1
0xbd6b0000 /lib/libextpiswu.so
0xbef50000 /opt/FJSVawjbk/jdk5/jre/lib/sparc/libioser12.so

Local Time = Fri Jan 12 20:37:57 2007
Elapsed Time = 9885
```

注意:

Error IDの行が出力されている場合、Error IDとして出力されている値は、Java VMが内部処理矛盾を自己検出した場合に出力する内部情報コードです。SIGSEGVやSIGBUSなどOSが検出した異常の場合には、常に同じ値(4F530E435050****)が出力されます。そのためError IDの先頭が「4F530E435050」で始まるコードの場合は、通常、意味を持ちません。Error IDの先頭が「4F530E435050」以外で始まるコードの場合には、障害情報検索時や判断時のキーワード情報としての意味を持ちます。

```
#
# HotSpot Virtual Machine Error : SIGSEGV (0xb)
#  [ pc=0xff092068, pid=27758(0x6c6e), nid=1(0x00000001), tid=0x00034d10 ]
#
# Please report this error to FUJITSU
#
# Java VM: Java HotSpot(TM) Server VM (1.5.0_FUJITSU_MODIFIED-B** mixed mode)
～～～～～～～～
```

略
～～～～～～～

(2)異常終了時のシグナルハンドラ情報 Solaris Linux

異常終了時のシグナルハンドラに関する情報が確認できます。
本例では、すべて「(in VM)」表示なので、シグナルハンドラの登録変更に関する問題はありません。

```
##>> Signal Handlers
 VM signal handler[1]=0xfe1ec0a0, VM signal handler[2]=0xfe4ff780, SIG_DFL= 0x00000000, SIG_IGN=0x00000001,
INT_SIG=(16,16), ASYNC_SIG=(17,17)
  SIGSEGV :signal handler=0xfe4ff780 (in VM *)
  SIGPIPE :signal handler=0xfe1ec0a0 (in VM)
  SIGBUS :signal handler=0xfe1ec0a0 (in VM *)
  SIGILL :signal handler=0xfe1ec0a0 (in VM)
  SIGFPE :signal handler=0xfe1ec0a0 (in VM)
  INTERRUPT_SIGNAL :signal handler=0xfe4ff010 (in VM +)
  ASYNC_SIGNAL :signal handler=0xfe1ec0a0 (in VM)
```

(3)異常終了時のJavaヒープ領域に関する情報

異常終了時のJavaヒープ領域に関する情報が確認できます。

JDK/JRE 5.0、6の**FJVM**の場合:
パラレルGC使用時:
「PSYoungGen」が「New世代領域」、
「PSOldGen」が「Old世代領域」、
「PSPermGen」が「Permanent世代領域」
に関する情報です。

CMS付きパラレルGC使用時:
「par new generation」が「New世代領域」、
「concurrent mark-sweep generation」が「Old世代領域」、
「concurrent-mark-sweep perm gen」が「Permanent世代領域」
に関する情報です。

なお「Old世代領域」および「Permanent世代領域」における「object space」についての情報は出力されません。

FJGC使用時(JDK/JRE 5.0の場合のみ):
「split eden generation」が「New世代領域」、
「tenured generation」が「Old世代領域」、
「compacting perm gen」が「Permanent世代領域」
に関する情報です。

シリアルGC使用時:
「def new generation」が「New世代領域」、
「tenured generation」が「Old世代領域」、
「compacting perm gen」が「Permanent世代領域」
に関する情報です。

JDK/JRE 1.4の**FJVM**の場合:
FJGC使用時:
「split eden generation」が「New世代領域」、
「tenured generation」が「Old世代領域」、
「compacting perm gen」が「Permanent世代領域」
に関する情報です。

シリアルGC使用時:
「def new generation」が「New世代領域」、
「tenured generation」が「Old世代領域」、

「compacting perm gen」が「Permanent世代領域」
に関する情報です。

本例の場合、異常終了時点における「New世代領域」＋「Old世代領域」の領域(-Xmxで最大量が指定される領域)には、空きがあることがわかります。
また「Permanent世代領域」に対しても、余裕があることがわかります。

注意:
パーセントで示されている値は、異常終了した時点でFJVMがJavaヒープ用に利用可能な状態にしている(コミットしている)メモリ量に対する比率です。利用可能な上限値に対する比率ではありません。
パーセントで示されている値は参照せず、K(キロ)単位で表示されているメモリ使用量の値と、オプションで指定された値(デフォルト値を含む)とを比較して判断してください。

注意:
パラレルGCを使用していた場合、以下の「-Xms=」「-Xmx=」に続いて表示される値は、-Xmsオプション/-Xmxオプションで指定された値などを元に、**FJVM**がJavaヒープの初期値として計算し直した値であるため、-Xmsオプションで指定した値とは異なりますので注意してください。

```
##>> Heap
  PSYoungGen      total 15488K, used 14480K [0xf62b0000, 0xf76e0000, 0xf7800000]
  eden space 15232K, 94% used [0xf62b0000,0xf70cc180,0xf7190000]
  from space 256K, 12% used [0xf7660000,0xf7668000,0xf76a0000]
  to   space 192K, 0% used [0xf76b0000,0xf76b0000,0xf76e0000]
 PSOldGen        total 1408K, used 156K [0xf3800000, 0xf3960000, 0xf62b0000]
  object space 1408K, 11% used [0xf3800000,0xf3827198,0xf3960000]
 PSPermGen       total 16384K, used 2173K [0xef800000, 0xf0800000, 0xf3800000]
  object space 16384K, 13% used [0xef800000,0xefa1f698,0xf0800000]
(-Xms=5504K, -Xmx=65536K, -XX:PermSize=16384K, -XX:MaxPermSize=65536K)
```

7.3 チューニング/デバッグ技法

ここでは、チューニング技法およびデバッグ技法を紹介します。

7.3.1 ガーベジコレクションのログ出力

ガーベジコレクション(GC)のログを採取するには、“-verbose:gc”オプションを指定します。本オプションの指定により、**GC**が発生するたびに、標準出力に1行ずつ出力されます。
出力フォーマットを図1、出力例を図2に示します。
また、本節では、“Javaヒープの中のメモリ割り当てプール”を“ヒープ”と略記します。

図1 GCログの出力フォーマット

```
[GCの種類 GC前のヒープ使用量->GC後のヒープの使用量(ヒープのサイズ), GCの処理時間]
```

GCの種類が“GC”の場合は**マイナーGC**で、“FullGC”の場合は**FullGC**であることを示します。
CMS付きパラレルGCを使用している場合は、以下の出力フォーマットのGCログもあります。

```
[GC ヒープ使用量(ヒープのサイズ), マーク処理時間]
```

このフォーマットで出力された場合は、CMS-GC処理のinitialマーク処理またはfinalマーク処理が実行されたことを示します。

図2 GCログの出力例

```
[GC 80229K->31691K(259776K), 0.4795163 secs]
[FullGC 57654K->4623K(259776K), 0.3844278 secs]
```

なおFJVMでは、GC処理の結果ログ出力機能の強化を行っています。
より詳細なGC処理の結果ログ情報を得る場合には、当該機能を使用してください。詳細は、“7.2.4 ガーベジコレクション処理の結果ログ出力機能の強化を参照してください。

ログ出力量の増加

本オプションの指定により、ログ出力が増大します。
本オプションを指定する場合は、ログ出力量についての注意が必要です。

7.3.2 スタックトレース

Javaアプリケーションで例外(java.lang.Throwableのインスタンス)がスローされた場合などに出力される**スタックトレース**は、エラーが発生するまでの経緯(メソッドの呼び出し順番)が示されています。このスタックトレースを解析することにより、エラーが発生した箇所と原因を確認することができます。

■スタックトレースの出力先

スタックトレースの出力先は、標準エラーです。通常のJavaアプリケーションの場合は、コンソールに出力されますが、Servlet/JSP/EJBアプリケーションの場合は、ログファイル(標準出力、標準エラー出力あるいはJava VMの出力を格納するファイルなど)に出力されます。

■スタックトレースの出力方法

Javaでスローされた例外をcatch節でキャッチし、例外のprintStackTraceメソッドを実行することにより、**スタックトレース**を出力することができます。
java.lang.Throwable.printStackTrace()で**スタックトレース**を出力する方法を、図1に示します。

図1 printStackTraceメソッドでスタックトレースを出力する方法

```
try {
    SampleBMPSessionRemote bmpSessionRemote = bmpSessionHome.create();
} catch(Exception e) {
    e.printStackTrace();
}
```

なお、スローされた例外をtry-catch構文で処理するメソッドがスレッドにない場合、そのスレッドは停止され、Java VMによって**スタックトレース**が出力されます。

■スタックトレースの出力フォーマット

スタックトレースの出力フォーマットを、図1に示します。

図1 スタックトレースの出力フォーマット

```
例外クラス名: エラーメッセージ
  at クラス名.メソッド名1(ソース名:行番号)     呼び出し先
  at クラス名.メソッド名2(ソース名:行番号)
              :
              :
  at クラス名.メソッド名N(ソース名:行番号)     呼び出し元
```

- 最初の1行目は、スローされた例外のクラス名とエラーメッセージです。
  エラーメッセージがない場合もあります。
- 2行目以降は、メソッドの呼び出し元(クラス名.メソッド名N)から呼び出し先(クラス名.メソッド名1)に向かって下から上に出力されます。
  2行目(クラス名.メソッド名1)が、例外をスローしたメソッドの情報です。
- “メソッド名”が“<init>”の場合、コンストラクタを示します。
- “メソッド名”が“<cinit>”が場合、static initializerを示します。
- “(ソース名:行番号)”が“(Native Method)”の場合、Javaのネイティブメソッド(.soや.dllファイル)を示します。
- クラスのコンパイル時にデバッグ情報を削除した場合、“(ソース名:行番号)”には、ソース名しか表示されなかったり、“Unknown Source”と表示されたりする場合があります。

7.3.2.1 スタックトレースの解析方法(その1)

図1の出力例をもとにして、解析方法を説明します。
図1の先頭の“数字:”は、説明の便宜上、付加しています。

図1 スタックトレースの出力例

```
1:java.lang.NullPointerException
2:  at agency.attestation.CheckLoginInfo.doCheck(CheckLoginInfo.java:150)
3:  at agency.attestation.AttestationServlet.doGet(AttestationServlet.java:96)
4:  at agency.attestation.AttestationServlet.doPost(AttestationServlet.java:161)
5:  at javax.servlet.http.HttpServlet.service(HttpServlet.java:772)
6:  at javax.servlet.http.HttpServlet.service(HttpServlet.java:865)
              :
```

■読み方

図1の**スタックトレース**は、6行目から上方向に読むと、次の流れで例外が発生したことがわかります。

1. javax.servlet.http.HttpServlet.service()が、HttpServlet.javaの865行目で、javax.servlet.http.HttpServlet.service()を実行し、
2. javax.servlet.http.HttpServlet.service()が、HttpServlet.javaの772行目で、agency.attestation.AttestationServlet.doPost()を実行し、
3. agency.attestation.AttestationServlet.doPost()が、AttestationServlet.javaの161行目で、agency.attestation.AttestationServlet.doGet()を実行し、
4. agency.attestation.AttestationServlet.doGet()が、AttestationServlet.javaの96行目で、agency.attestation.CheckLoginInfo.doCheck()を実行した結果、
5. agency.attestation.CheckLoginInfo.doCheck()内のCheckLoginInfo.javaの150行目で、java.lang.NullPointerExceptionという例外が発生した

■解析方法

図1の**スタックトレース**の解析例を、次に示します。

1. 1行目の例外情報から、原因を特定できるかどうか確認します。
   NullPointerExceptionがスローされていることがわかります。
2. 2行目のCheckLoginInfo.javaの開発担当者であれば、150行目の実装に問題がないかどうかを確認します。
3. 2行目のCheckLoginInfo.javaの開発担当者でない場合、**スタックトレース**中で最上行にある開発担当者が開発したクラスを探します。そして、そのクラスの実装に問題がないかどうかを確認します。それでも、原因を特定できない場合は、開発したクラスが使用しているクラスの提供元に調査を依頼します。

または、**スタックトレース**が、想定された流れでメソッドを実行しているかどうかを確認するのも1つの方法です。

7.3.2.2 スタックトレースの解析方法(その2)

図1の出力例をもとにして、解析方法を説明します。
図1の先頭の“数字:”は、説明の便宜上、付加しています。

図1 スタックトレースの出力例

```
1:java.util.MissingResourceException: Can't find bundle for base name sample.SampleResource, locale ja_JP
2:  at java.util.ResourceBundle.throwMissingResourceException(Unknown Source)
3:  at java.util.ResourceBundle.getBundleImpl(Unknown Source)
4:  at java.util.ResourceBundle.getBundle(Unknown Source)
5:  at sample.SampleMessage.getMessage(SampleMessage.java:15)
6:  at sample.SampleServlet.doGet(SampleServlet.java:10)
7:  at javax.servlet.http.HttpServlet.service(HttpServlet.java:696)
8:  at javax.servlet.http.HttpServlet.service(HttpServlet.java:809)
    :
    :
```

■解析方法

図1の**スタックトレース**の解析例を、次に示します。

1. 1行目の例外情報から、原因を特定できないかを確認します。

   APIリファレンスによると、java.util.MissingResourceExceptionは、Javaのリソースがない場合に発生する例外です。また、エラーメッセージによると、sample.SampleResourceというリソースファイルの日本語版(ja_JP)がないということがわかります。

2. リソースファイルを確認します。

   a. リソースファイル名を誤っていないか
   SampleMessage.javaの15行目のsample.SampleMessage.getMessage ()内で、java.util.ResourceBundle.getBundle()を実行した結果、例外がスローされています。したがって、そこでjava.util.ResourceBundle.getBundle()に渡しているリソースファイル名に誤りがないかどうかを確認します。

   b. リソースファイルが、所定のディレクトリ構成内に存在するか
   a)のリソースファイル名が正しい場合、所定のディレクトリ構成(/sample/)に、次のいずれかのリソースファイルがあるかどうかを確認します。

      - SampleResource_ja_JP.properties
      - SampleResource_ja_JP.class
      - SampleResource_ja.properties
      - SampleResource_ja.class
      - SampleResource.properties
      - SampleResource.class

7.3.2.3 スタックトレースの解析方法(その3)

JDK/JRE 1.4になって、java.lang.Throwableに次のコンストラクタとメソッドが追加されました。

- Throwable(java.lang.String, java.lang.Throwable)
- Throwable(java.lang.Throwable)
- initCause(java.lang.Throwable)

これにより、**スタックトレース**には、原因となる例外の**スタックトレース**も出力されるようになりました。

以降、図1のサンプルを使って、説明します。

図1 サンプルプログラム

```
1 :public class Test {
2 :
3 :    public static void main(String[] args) {
4 :       new Test();
5 :    }
6 :
7 :    Test() {
8 :        try{
9 :            parentMethod();
10:        } catch (Exception e) {
11:            e.printStackTrace();
12:        }
13:    }
14:
15:    void parentMethod() throws HiLevelException {
16:        try {
17:            childMethod();
18:        } catch (Exception e) {
19:            throw new HiLevelException("HiLevel", e);
20:        }
```

```
21:     }
22:
23:     void childMethod() throws LowLevelException {
24:         throw new LowLevelException("LowLevel");
25:     }
26:}
27:
28:class HiLevelException extends Exception {
29:     HiLevelException(String msg, Throwable cause) {
30:         super(msg, cause);
31:     }
32:}
33:
34:class LowLevelException extends Exception {
35:     LowLevelException(String msg) {
36:         super(msg);
37:     }
38:}
```

図1のサンプルを実行すると、図2の**スタックトレース**が出力されます。

図2 スタックトレース

```
HiLevelException: HiLevel
    at Test.parentMethod(Test.java:19)
    at Test.<init>(Test.java:9)
    at Test.main(Test.java:4)
Caused by: LowLevelException: LowLevel
    at Test.childMethod(Test.java:24)
    at Test.parentMethod(Test.java:17)
    ... 2 more
```

HiLevelExceptionに続いて、“Caused by:”以降に、原因となるLowLevelExceptionのスタックトレースが出力されています。最終行の“... 2 more”は、“Caused by:”の直前の2行が続きの**スタックトレース**であることを示しています。
つまり、図3のように解釈することができます。

図3 原因となる例外の解釈

```
Caused by: LowLevelException: LowLevel
    at Test.childMethod(Test.java:24)
    at Test.parentMethod(Test.java:17)
    at Test.<init>(Test.java:9)
    at Test.main(Test.java:4)
```

以上から、次のことがわかります。

- **スタックトレース**の原因は、LowLevelExceptionである
- Test.main(Test.javaの4行目)が、最初の呼び出し元である。

詳細は、Java APIリファレンスのjava.lang.ThrowableのprintStackTraceメソッドの解説を参照してください。

7.3.3 スレッドダンプ

スレッドダンプには、Javaプロセスの各スレッドの情報(スタックトレース形式)が含まれているため、ハングアップやデッドロックなどの動作具合を調査することができます。

スレッドダンプの出力先は、標準出力です。**スレッドダンプ**が出力される契機および出力先を、“表7.4 スレッドダンプの出力契機と出力先”に示します。

表7.4 スレッドダンプの出力契機と出力先

プログラムの種類	出力契機	出力先
J2EEアプリケーション	一定の条件を満たした場合、コンテナの機能により自動的に採取される場合と利用者の任意のタイミングで手動による採取があります。 自動採取: アプリケーションがタイムアウトまたは無応答になった場合 手動採取: Windows “スレッドダンプツール”で採取することができます。スレッドダンプツールの詳細は、“トラブルシューティング集”の“スレッドダンプツール”を参照してください。 Solaris Linux kill -QUIT [プロセスID]でJava VMに対してQUITシグナルを送り採取することができます。	標準出力、JavaVMの出力をロギングしているファイル
J2EE以外のJavaプログラム	利用者の任意のタイミングで手動で採取することができます。 Windows コマンドプロンプトからJavaプログラムを起動した場合: 以下、いずれかの方法で採取できます。 1)[Ctrl]+[Break]キー押下 2)“スレッドダンプツール” コマンドプロンプト以外からJavaプログラムを起動した場合: “スレッドダンプツール”で採取します。 スレッドダンプツールの詳細は、“トラブルシューティング集”の“スレッドダンプツール”を参照してください。 Solaris Linux ターミナルからJavaプログラムを起動した場合: 以下、いずれかの方法で採取できます。 1)[Ctrl]+[¥]キー(英語キーボードの場合バックスラッシュキー)押下 2)kill -QUIT [プロセスID] ターミナル以外からJavaプログラムを起動した場合: kill -QUIT [プロセスID]で採取します。 注意 “-Xrs”オプションが指定されたJavaプロセスの場合、当該プロセスへ送られた[Ctrl]+[Break]キー押下またはQUITシグナルに対する動作は、OSのデフォルト動作になります。 そのため、“-Xrs”オプションを指定したJavaプロセスに対して[Ctrl]+[Break]キー押下またはQUITシグナルが送られると、当該Javaプロセスは強制終了または異常終了します。 スレッドダンプを出力する可能性があるJavaプロセスに対	コンソール(標準出力)

プログラムの種類	出力契機	出力先
	して、“-Xrs”オプションは指定しないでください。 ただし、Windows(R)でサービスとして登録されるJavaプロセスの場合は、“-Xrs”オプションを指定しない場合、ログオフ時に強制終了してしまいます。これが不都合な場合は、“-Xrs”オプションを指定してください。	

図1の出力例をもとにして、**スレッドダンプ**の解析方法を説明します。

図1 サンプルプログラム

```
1 :public class DeadlockSample {
2 :    static boolean flag;
3 :    static Thread1 thread1;
4 :    static Thread2 thread2;
5 :
6 :    public static void main(String[] args) {
7 :        thread1 = new Thread1();
8 :        thread2 = new Thread2();
9 :        thread1.start();
10:        thread2.start();
11:    }
12:}
13:
14:class Thread1 extends Thread {
15:    public Thread1(){
16:        super("Thread1");
17:    }
18:
19:    public void run(){
20:        synchronized(this){
21:            System.out.println("Thread1開始");
22:            while(DeadlockSample.flag==false){  // Thread2が開始するのを待つ
23:                yield();
24:            }
25:            DeadlockSample.thread2.method();
26:            notify();
27:        }
28:    }
29:
30:    public synchronized void method(){
31:        try{wait(1000);}catch(InterruptedException ex){}
32:        System.out.println("Thread1.method()終了");
33:    }
34:}
35:
36:class Thread2 extends Thread {
37:    public Thread2(){
38:        super("Thread2");
39:    }
40:
41:    public void run() {
42:        synchronized(this){
43:            DeadlockSample.flag = true;
44:            System.out.println("Thread2開始");
45:            DeadlockSample.thread1.method();
46:            notify();
47:        }
48:    }
49:
50:    public synchronized void method() {
51:        try{wait(1000);}catch(InterruptedException ex){}
```

```
52:         System.out.println("Thread2.method()終了");
53:     }
54:}
```

図1のサンプルでは、Thread1とThread2がお互いに排他処理を行っています。
このサンプルを実行すると、次のように処理が進められます。

1. Thread1で、Thread1のロックを獲得する(20行目のsynchronized節)
2. Thread2で、Thread2のロックを獲得する(42行目のsynchronized節)
3. Thread1で、Thread2.method()を実行しようとして、ロック解放待ちになる(50行目のsynchronized修飾子)
4. Thread2で、Thread1.method()を実行しようとして、ロック解放待ちになる(50行目のsynchronized修飾子)

この結果、Thead1とThread2がお互いに、解放されないロックを待ち続けるデッドロック状態になります。
デッドロック状態で、**スレッドダンプ**を採取したものを、図2に示します。

図2 スレッドダンプ

```
"DestroyJavaVM" prio=5 tid=0x002856c8 nid=0x5f4 waiting on condition [0..6fad8]

"Thread2" prio=5 tid=0x0092f4d8 nid=0x640 waiting for monitor entry [182ef000..182efd64]
    at Thread1.method(DeadlockSample.java:31)
    - waiting to lock <0x1002ffe8> (a Thread1)
    at Thread2.run(DeadlockSample.java:45)
    - locked <0x10030ca0> (a Thread2)

"Thread1" prio=5 tid=0x0092f370 nid=0x294 waiting for monitor entry [182af000..182afd64]
    at Thread2.method(DeadlockSample.java:51)
    - waiting to lock <0x10030ca0> (a Thread2)
    at Thread1.run(DeadlockSample.java:25)
    - locked <0x1002ffe8> (a Thread1)

"Signal Dispatcher" daemon prio=10 tid=0x0098eb80 nid=0x634 waiting on condition [0..0]

"Finalizer" daemon prio=9 tid=0x0092a540 nid=0x5e8 in Object.wait() [1816f000..1816fd64]
    at java.lang.Object.wait(Native Method)
    - waiting on <0x10010498> (a java.lang.ref.ReferenceQueue$Lock)
    at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:111)
    - locked <0x10010498> (a java.lang.ref.ReferenceQueue$Lock)
    at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:127)
    at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:159)

"Reference Handler" daemon prio=10 tid=0x0096da70 nid=0x5e4 in Object.wait() [1812f000..1812fd64]
    at java.lang.Object.wait(Native Method)
    - waiting on <0x10010388> (a java.lang.ref.Reference$Lock)
    at java.lang.Object.wait(Object.java:429)
    at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:115)
    - locked <0x10010388> (a java.lang.ref.Reference$Lock)

"VM Thread" prio=5 tid=0x0096c950 nid=0x624 runnable

"VM Periodic Task Thread" prio=10 tid=0x0092c008 nid=0x2a0 waiting on condition
"Suspend Checker Thread" prio=10 tid=0x0098e118 nid=0x478 runnable

Found one Java-level deadlock:
=============================
"Thread2":
  waiting to lock monitor 0x00929c3c (object 0x1002ffe8, a Thread1),
  which is held by "Thread1"
"Thread1":
  waiting to lock monitor 0x00929c5c (object 0x10030ca0, a Thread2),
  which is held by "Thread2"
```

```
Java stack information for the threads listed above:
===================================================
"Thread2":
    at Thread1.method(DeadlockSample.java:31)
    - waiting to lock <0x1002ffe8> (a Thread1)
    at Thread2.run(DeadlockSample.java:45)
    - locked <0x10030ca0> (a Thread2)
"Thread1":
    at Thread2.method(DeadlockSample.java:51)
    - waiting to lock <0x10030ca0> (a Thread2)
    at Thread1.run(DeadlockSample.java:25)
    - locked <0x1002ffe8> (a Thread1)

Found 1 deadlock.
```

■解析方法

スレッドダンプの各スレッドの情報は、スタックトレース形式です。
Thread1とThread2の両方のスタックトレースには、“locked”と“waiting to lock”があります。また、**スレッドダンプ**の下の方にも、“deadlock”の文字列があり、デットロックが発生していることが確認できます。

このように、**スレッドダンプ**で全スレッドの動作状況を確認するにより、Javaプロセスがハングアップしているか、あるいは、デッドロック状態かを確認することができます。特に、短い間隔で複数のスレッドダンプを採取し、スレッドに動きがなければ、ハングアップの可能性があります。

スレッドダンプの詳細は、“トラブルシューティング集”の“スレッドダンプが出力された場合の対処”も参照してください。

オブジェクトをロックしているスレッドがスレッドダンプ上に出ない

通常スレッドダンプ上のあるスレッドで次のように表示される場合があります。

```
- waiting to lock <オブジェクトID> (a クラス名)
```

このような場合、別のスレッドがそのオブジェクトIDのロックを持っていて、そのスレッドのトレース上のどこかで次の表示がされています。

```
- locked <オブジェクトID> (a クラス名)
```

しかし、スレッドダンプを表示するタイミングによっては“- locked <オブジェクトID> (a クラス名)”の表示がどのスレッドにも現われず、“- waiting to lock <オブジェクトID> (a クラス名)”だけ表示される場合があります。

以下のプログラムを例とします。

```
 1  class NoLockOwner extends Thread
 2  {
 3    static Object lock = new Object();
 4
 5    public static void main(String[] arg)
 6    {
 7      new NoLockOwner().start();
 8      new NoLockOwner().start();
 9    }
10
11    public void run()
12    {
13      while (true) {
14        synchronized (lock) {
15          dumb();
16        }
17      }
18    }
19
20    void dumb()
21    {
```

```
22      int n = 0;
23      for (int i = 0 ; i < 1000 ; ++i)
24        n += i;
25    }
26
27  }
```

(0)スレッドダンプを取ると、通常はこのようになります。

```
"Thread-1" prio=1 tid=0x10 nid=0x5 waiting for monitor entry [0x3000..0x4000]
    at NoLockOwner.run(NoLockOwner.java:14)
    - waiting to lock <0x800> (a java.lang.Object)
"Thread-0" prio=1 tid=0x20 nid=0x6 runnable [0x5000..0x6000]
    at NoLockOwner.dumb(NoLockOwner.java:23)
    at NoLockOwner.run(NoLockOwner.java:15)
    - locked <0x800> (a java.lang.Object)
```

(1)先頭フレームでオブジェクトをロックしている場合は“- locked”ではなく、“- waiting to lock”と表示されることがあります。

```
"Thread-1" prio=1 tid=0x10 nid=0x5 waiting for monitor entry [0x3000..0x4000]
    at NoLockOwner.run(NoLockOwner.java:14)
    - waiting to lock <0x800> (a java.lang.Object)
"Thread-0" prio=1 tid=0x20 nid=0x6 runnable [0x5000..0x6000]
    at NoLockOwner.run(NoLockOwner.java:14)
    - waiting to lock <0x800> (a java.lang.Object)
```

この場合、スレッドの状態を見て、runnableであれば、ロック待ちではなく、ロック取得後同じフレームを実行中の状態であると考えます。

Thread-0、Thread-1ともに、“- waiting to lock <0x800> (a java.lang.Object)”と表示されているので、どちらもロック待ちのように見えます。

しかし、Thread-0の状態は、runnbleなので、ロック待ち状態ではありません。

(2) “- waiting to lock”と表示されるのは、ロック待ちの状態だけではなく、ロック獲得処理の最中でもそのように表示されます。

```
"Thread-1" prio=1 tid=0x10 nid=0x5 waiting for monitor entry [0x3000..0x4000]
    at NoLockOwner.run(NoLockOwner.java:14)
    - waiting to lock <0x800> (a java.lang.Object)
"Thread-0" prio=1 tid=0x20 nid=0x6 waiting for monitor entry [0x5000..0x6000]
    at NoLockOwner.run(NoLockOwner.java:14)
    - waiting to lock <0x800> (a java.lang.Object)
```

Thread-0、Thread-1ともに、“- waiting to lock <0x800> (a java.lang.Object)”と表示されているので、どちらもロック待ちのように見えます。

さらに、スレッドの状態も、どちらも、“waiting for monitor entry”になっています。

“- waiting to lock”と表示されるのは、ロック待ちの状態だけではなく、ロック獲得処理の最中でもそのように表示されます。

したがって、この場合、Thread-0またはThread-1のいずれか一方、あるいは両方がロック獲得処理の最中であると考えます。

しかし、この状態は長く続かず、短時間で(0)または(1)に移行します。

(3) ロックを開放した直後の状態

```
"Thread-1" prio=1 tid=0x10 nid=0x5 waiting for monitor entry [0x3000..0x4000]
    at NoLockOwner.run(NoLockOwner.java:14)
    - waiting to lock <0x800> (a java.lang.Object)
"Thread-0" prio=1 tid=0x20 nid=0x6 runnable [0x5000..0x6000]
    at NoLockOwner.run(NoLockOwner.java:16)
```

Thraed-0がちょうどロックを開放した直後の状態です。

この状態も長く続くとはなく、短時間で(0)または(1)に移行します。

スレッドダンプからアプリケーションの状態を判断する場合、ひとつのスレッドダンプから状態を判断することは困難です。

適切な判断をするためには、複数回のスレッドダンプを総合的に見ることが必要です。

スレッドダンプ中に表示されるsynchronizedメソッドの行番号について

次のようなプログラムを考えます。

```
 1    class SyncMethod extends Thread
 2    {
 3      static volatile int k;
 4
 5      public static void main(String[] arg)
 6      {
 7        new SyncMethod().start();
 8      }
 9
10      public void run()
11      {
12        while (true) {
13          dumb();
14        }
15      }
16
17      synchronized void dumb()
18      {
19        /*
20          meaningless comments
21         */
22        int i = 0;
23        for ( ; i < 10 ; ++i)
24          k += i;
25      }
26
27    }
```

このようなプログラムのスレッドダンプを取ると以下のように出力されることがあります。

```
"Thread-0" prio=1 tid=0x300 nid=0x61 runnable [1000..2000]
    at SyncMethod.dumb(SyncMethod.java:23)
    - waiting to lock <0xa00> (a SyncMethod)
    at SyncMethod.run(SyncMethod.java:13)
```

23行目でロックを獲得しているように見えますが、23行目は、“for (; i < 10 ; ++i)”であり、何もロック獲得に関係しているようにソースコード上は見えません。

これは、synchronizedメソッドがロックを獲得する行番号は、最初に実行される行番号になるためです。

スレッドダンプ中、複数のスレッドがロックを獲得している場合ついて

次のようなプログラムを考えます。

```
 1    class NoNotify extends Thread
 2    {
 3      static Object o = new Object();
 4
 5      public static void main(String[] arg)
 6      {
 7        new NoNotify().start();
 8        new NoNotify().start();
 9      }
10
11      public void run()
12      {
13        try {
14          synchronized (o) {
15            o.wait();
16          }
```

```
17          } catch (Exception e) {}
18      }
19
20  }
```

このプログラムには誤りがあります。

このプログラムでは誰もnotifyするスレッドがないため、どちらのスレッドも永久に起こされることはありません。

このプログラムのスレッドダンプを採取すると次のように示される場所があります。

```
"Thread-1" prio=1 tid=0x800 nid=0x6 in Object.wait() [1000..2000]
    at java.lang.Object.wait(Native Method)
    - waiting on <0x200> (a java.lang.Object)
    at java.lang.Object.wait(Object.java:429)
    at NoNotify.run(NoNotify.java:15)
    - locked <0x200> (a java.lang.Object)
"Thread-0" prio=1 tid=0x900 nid=0x7 in Object.wait() [3000..4000]
    at java.lang.Object.wait(Native Method)
    - waiting on <0x200> (a java.lang.Object)
    at java.lang.Object.wait(Object.java:429)
    at NoNotify.run(NoNotify.java:15)
    - locked <0x200> (a java.lang.Object)
```

このスレッドダンプから分かることは、複数のスレッドがロックを獲得しているのではなく、どのスレッドもロックを獲得していないということです。

Thread-0、Thraed-1ともに、同じオブジェクト“ID<0x200>”のオブジェクトをロックしているように見えます。

しかし“- locked”と表示されているスレッドが、現在のロックオーナであることは意味しません。(“- locked”の正確な意味するところは、そのフレームにおいてロックしたということに過ぎません。)

先頭フレームでは、“- waiting on”と表示されています。

これは、ロックが解放されたら起こされる可能性のある“- waiting to lock”とは異なり、ロックが解放されても自動的に起こされることを意味しません。

7.3.4 クラッシュダンプ・コアダンプ

Javaアプリケーションが異常終了(プロセスが消滅)したときに、各OS上に用意されたクラッシュダンプやコアダンプを採取することにより、異常終了の原因を調査することができる場合があります。

7.3.4.1 クラッシュダンプ

Windows

ここでは、Windows(R)上で異常を調査する場合に採取する、クラッシュダンプの採取方法を説明します。

■ワトソン博士について

ワトソン博士はMicrosoft Corporationのソフトウェアで、プログラムエラーのためのデバッガです。
プログラムエラーが発生すると、**ワトソン博士**が自動的にログファイルにデバッグ情報を出力します。なお、ログファイル名は、「drwtsn32.log」です。また、ログファイルの出力先は、**ワトソン博士**を起動して、設定することができます。
ワトソン博士の詳細は、Microsoft CorporationのWebページを参照してください。

■ワトソン博士の設定

クラッシュダンプの採取には、Windows(R)に同梱されている「**ワトソン博士**」を使用します。
次の例を参考にして、「**ワトソン博士**」を設定してください。この設定を行うことにより、異常終了時に、自動的にクラッシュダンプが出力されるようになります。

例 **ワトソン博士の設定例 (Windows(R) 2000の場合)**

1. MS-DOSコマンドプロンプトなどで“drwtsn32 -i”コマンドを投入します。[ワトソン博士が既定のアプリケーション デバッガとしてインストールされました。]のメッセージが表示されます。
2. 更に、MS-DOSコマンドプロンプトなどで、“drwtsn32”コマンドを実行します。[Windows 2000 ワトソン博士]の設定画面が表示されますので、以下を確認してください。
   - [ログファイルパス(L)]、[クラッシュダンプ(P)]が正しく指定されているか
   - [すべてのスレッド コンテキストをダンプ(A)]のチェックボックスがチェックされているか
   - [既定のログ ファイルに追加(E)]のチェックボックスがチェックされているか
   - [メッセージ ボックスによる通知(U)]のチェックボックスがチェックされているか
   - [クラッシュ ダンプ ファイルの作成(T)]のチェックボックスがチェックされているか

Windows(R) XP、Windows Server(R) 2003の場合

drwtsn32を立ち上げて、[クラッシュダンプの種類]を[完全]に設定しておく必要があります。

Windows Vista(R) SP1、Windows Server(R) 2008の場合

Windows Vista（R）、Windows Server(R) 2008ではワトソン博士の機能が提供されていません。

ワトソン博士の代わりにWindows Error Reporting（WER）の機能を使用します。

WERに関する設定の方法はOSのマニュアル、ヘルプを参照ください。

■注意事項

Windows Server(R) 2003の初期版においては、ユーザダンプが出力されない問題をはじめとして、その他にもJavaの実行動作に影響を及ぼす問題などがあります。
たとえば、次のような問題があります。

- http://support.microsoft.com/kb/836080/en-us
- http://support.microsoft.com/kb/837018/en-us
- http://support.microsoft.com/kb/841176/en-us

Windows Server(R) 2003を使用する場合は、Service Pack 1以降またはHotfixを適用してください。

7.3.4.2 コアダンプ（Solaris）

Solaris

ここでは、Solaris上でのコアダンプ採取のための注意事項を説明します。

■コアダンプが出力されない場合の確認

コアダンプが出力されない場合の原因として、システムリソース等の問題がまず考えられます。カレントディレクトリの書込み権、ディスク容量、limit(1)コマンド結果を確認してください。

7.3.4.3 コアダンプ（Linux）

Linux

ここでは、Linux上でのコアダンプ採取のための注意事項を説明します。

■コアダンプが出力されない場合の確認

コアダンプが出力されない場合の原因として、システムリソース等の問題がまず考えられます。カレントディレクトリの書込み権、ディスク容量、limit(1)コマンド結果を確認してください。

また、Linuxではハード/OSの出荷時もしくはOSのUpdate適用により、デフォルトでコアダンプの出力が設定されていない場合があります。以下を実施してコアダンプが出力されるようにしてください。

《コアダンプ出力設定方法》

・ isstartコマンドでInterstageを起動させる場合

sh(bash)で"ulimit -c unlimited"コマンド実行後、Interstageを起動させます。ワークユニット起動ユーザがInterstage起動ユーザと違う場合は、ワークユニット起動前に"ulimit -c unlimited"コマンドを実行してから、ワークユニットを起動させます。

・ RCプロシジャでOS起動時に自動的にInterstageが起動するように設定されている場合

以下の方法を実施することで、OS再起動後にcoreが出力されるようになります。

/etc/init.d/functionsファイルに、

```
# make sure it doesn't core dump anywhere; while this could mask
# problems with the daemon, it also closes some security problems
ulimit -S -c 0 >/dev/null 2>&1
  または、
ulimit -S -c ${DAEMON_COREFILE_LIMIT:-0} >/dev/null 2>1
```

と記述されていますので、上記の設定で"0"を、"unlimited"に変更してください。

```
ulimit -S -c unlimited >/dev/null 2>&1
```

/etc/rc2.d/S99startisに、以下の<---の記述を追加してください。

```
#!/bin/sh# Interstage Application Server
# S99starttis : Interstage Application Server start procedure

OD_HOME=/opt/FJSVod
export OD_HOME

ulimit -c unlimited   <---

/opt/FJSVod/bin/odalive > /dev/null
while [ "$?" != "0" ]
do
    sleep 1
    /opt/FJSVod/bin/odalive > /dev/null
done

/opt/FJSVtd/bin/isstart
```

7.3.5 JNI処理異常時のメッセージ出力

Java以外の言語と連携する場合、Java Native Interface(JNI)を使用します。
しかし、JNIの使用方法を誤ると、Javaプロセスの終了(異常終了)などの原因となります。

このとき、図1のオプションを指定することにより、JNI処理で異常が発生した場合にメッセージが出力されますので、JNIのパラメータなどの確認に活用してください。

図1 JNI処理異常時にメッセージを出力するオプション

```
-Xcheck:jni
```

上記の“-Xcheck:jni”パラメータを指定したときに、図2のメッセージが出力されることがあります。

図2 JNI処理異常時に出力されるメッセージ

```
FATAL ERROR in native method: (詳細メッセージ)
```

“(詳細メッセージ)”の部分には、以降で説明する文字列が出力されます。

以降、図2の“詳細メッセージ”が出力される例と注意事項を説明します。
以降の説明を参考にして、JNIの処理部分を見直してください。

■メッセージの説明

JNI received a class argument that is not a class

[異常例]

```
char buf[1];
(*env)->AllocObject(env, (jclass)buf); //jclass型の第2引数に違う型を指定
```

JNI received a null class

[異常例]

```
(*env)->AllocObject(env, NULL); //jclass型の第2引数にnullを指定
```

JNI string operation received a non-string

[異常例]

```
(*env)->GetStringUTFChars(env, NULL, 0); //jstring型の第2引数にNULLを指定
```

Non-array passed to JNI array operations

[異常例]

```
(*env)->GetArrayLength(env, (jarray)(*env)->NewStringUTF(env, "abc")); //jarray型の第2引数に配列でない型を指定
```

[注意事項]
次の場合は、“-Xcheck:jni”オプションによるメッセージは出力されません。

```
char buf[1];
(*env)->GetArrayLength(env, (jarray)buf);
```

Static field ID passed to JNI

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
jfieldID fid = (*env)->GetFieldID(env, cls, "static_data", "I"); (*env)->GetIntField(env, obj, fid); //
jfieldID型の第3引数にstaticフィールドを指定
```

Null object passed to JNI

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
jfieldID fid = (*env)->GetFieldID(env, cls, "instance_data", "I");
(*env)->GetIntField(env, NULL, fid); //object型の第2引数にNULLを指定
```

[注意事項]
instance変数かどうかのチェック時のみに出力されるメッセージです。
次の場合は、“-Xcheck:jni”オプションによるメッセージは出力されません。

```
(*env)->GetObjectClass(env, NULL); //objectの第2引数にNULLを指定
```

Wrong field ID passed to JNI

[異常例]

```
(*env)->GetIntField(env, obj, -1); //jfieldID型の第3引数に数値を指定
```

[注意事項]
instance変数かどうかのチェック時のみに出力されるメッセージです。

Non-static field ID passed to JNI

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
(*env)->GetStaticIntField(env, cls, -1); //jfiedlID型第2引数に数値を指定
```

[注意事項]
次の場合は、“-Xcheck:jni”オプションによるメッセージは出力されません。

```
jclass cls = (*env)->GetObjectClass(env, obj);
jfieldID fid = (*env)->GetStaticFieldID(env, cls, "instance_data", "I");
(*env)->GetStaticIntField(env, cls, fid);
```

Array element type mismatch in JNI

[異常例]

```
jintArray intarray = (*env)->NewIntArray(env, 2);
(*env)->GetFloatArrayElements(env, intarray, 0); //floatArray型の第2引数にjintArrayを指定
```

Object array expected but not received for JNI array operation

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
jobjectArray objarray = (*env)->NewObjectArray(env, 1, cls, obj);
(*env)->GetIntArrayElements(env, objarray, 0); //intArray型の第2引数にjobjectArray型を指定
```

Field type (static) mismatch in JNI get/set field operations

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
jfieldID fid = (*env)->GetStaticFieldID(env, cls, "static_data", "I");
(*env)->GetStaticFloatField(env, cls, fid); //GetStaticFloatFieldではなくGetStaticIntFieldでなければ
ならない
```

Field type (instance) mismatch in JNI get/set field operations

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
jfieldID fid = (*env)->GetFieldID(env, cls, "instance_data", "I");
(*env)->GetFloatField(env, obj, fid); //GetFloatFieldではなくGetIntFieldでなければならない
```

Wrong static field ID passed to JNI

[異常例]

```
jclass cls = (*env)->GetObjectClass(env, obj);
jclass cls2 = (*env)->GetObjectClass(env, (*env)->NewStringUTF(env, "abc"));
```

```
jfieldID fid = (*env)->GetStaticFieldID(env, cls, "static_data", "I");
(*env)->GetStaticObjectField(env, cls2, fid); //第2引数はcls2ではなくclsでなければならない
```

Using JNIEnv in the wrong thread

[解説]

実行しているスレッドのためのものではないJNIEnvを使用したためのエラーです。
Java VMは、JNIインタフェースポインタ(JNIEnv)が参照する領域を、スレッド固有のデータ領域に割り当てることがあります。このため、JNIインタフェースポインタは、カレントスレッドに対してのみ有効です。ネイティブメソッドは、JNIインタフェースポインタを別のスレッドに渡すといった使い方はできません。

7.3.6 例外発生時のスタックトレース出力

FJVMを使用してJavaアプリケーションを実行している場合、以下の例外については、実行性能の観点から、Java VMの動的コンパイル処理が行なう最適化処理により例外発生時のスタックトレース出力処理が省略され、例外発生時のスタックトレースが出力されない場合があります（例外発生時のスタックトレース出力抑止）。

JDK/JRE 1.4のFJVMの場合

- java.lang.NullPointerException
- java.lang.ArithmeticException

JDK/JRE 5.0、6のFJVMの場合

- java.lang.NullPointerException
- java.lang.ArithmeticException
- java.lang.ArrayIndexOutOfBoundsException
- java.lang.ArrayStoreException
- java.lang.ClassCastException

動的コンパイル処理により例外発生時のスタックトレース出力処理が省略されないようにする場合は、“-XX:-OmitStackTraceInFastThrow”オプションを指定します。

ただし該当する例外の発生頻度が高い場合に当該オプションを指定し、例外発生時のスタックトレース出力を行なった場合は、Javaアプリケーションの実行性能が低下する場合があります。

当該オプションを指定する場合は、性能検証を行った上で使用するか、開発作業において例外が発生している場所を特定したい場合においてだけ使用してください。

7.4 異常発生時の原因振り分け

本項では、異常が発生したときに、原因を振り分ける方法を説明します。

7.4.1 java.lang.StackOverflowErrorがスローされた場合

StackOverflowErrorがスローされた場合、スタックオーバーフローが原因です。

スタックのサイズをチューニングしてください。
スタックのチューニング方法は、“7.5.1 スタックのチューニング”を参照してください。

7.4.2 java.lang.OutOfMemoryErrorがスローされた場合

本節では、**OutOfMemoryError**がスローされた場合、考えられる原因とその対処方法を説明します。

なお、FJVMを使用でOutOfMemoryErrorがスローされた場合に出力されるメッセージ情報については、“メモリ領域不足事象発生時のメッセージ出力機能の強化”も参照してください。

■想定される原因(メモリリーク)

VMがガーベジコレクションを繰り返しても、時間の経緯とともにメモリ消費量が増大していく場合、プログラム中メモリリークを起こしている可能性があります。

メモリリークの結果、Javaのヒープ不足が発生し**OutOfMemoryError**がスローされる場合があります。

この場合、ガーベジコレクションのログを採取して、Javaヒープの消費状況を確認してください。ガーベジコレクションのログを採取する方法は、“7.3.1 ガーベジコレクションのログ出力”を参照してください。

■想定される原因(Javaヒープ不足)

通常、**OutOfMemoryError**は、Javaヒープ不足が原因でスローされます。

ガーベジコレクションのログを採取して、Javaヒープの消費状況を確認してください。
Javaヒープの空き容量がないことが確認されたら、Javaヒープをチューニングしてください。

ガーベジコレクションのログを採取する方法は、“7.3.1 ガーベジコレクションのログ出力”を参照してください。
Javaヒープのチューニング方法は、“7.5.2 Javaヒープのチューニング”を参照してください。

■想定される原因(ユーザ空間不足)

多量のスレッドを生成して、多量のスタックがユーザ空間内に割り当てられ、ユーザ空間不足になった場合、次の**OutOfMemoryError**がスローされる、あるいはエラーメッセージとして表示を行いプロセスが終了します。

```
java.lang.OutOfMemoryError: unable to create new native thread
```

また、JavaヒープやOSの仮想メモリに余裕があるにもかかわらず、ユーザ空間内にメモリを確保できなかった場合、次の**OutOfMemoryError**が出力されプログラムが終了します。

```
java.lang.OutOfMemoryError: requested サイズ bytes 制御名. Out of swap space?
```

サイズ:　　確保できなかったメモリの大きさ

制御名:　　メモリが確保できなかったJava VMの制御名(該当情報がある場合にだけ表示)

ユーザ空間が不足している場合は、Javaヒープまたはスタックのサイズを小さくするなどのチューニングを行ってください。
スタックのサイズをチューニングする方法は、“7.5.1 スタックのチューニング”を参照してください。
Javaヒープのチューニング方法は、“7.5.2 Javaヒープのチューニング”を参照してください。

なお、仮想メモリに余裕がある場合は、Javaプロセスを複数起動して、プロセス多重度を上げる方法もあります。J2EEアプリケーションまたはJavaEEアプリケーションの場合、J2EEまたはJavaEEのチューニングを行ってください。J2EEまたはJavaEEのチューニング方法の詳細は、それぞれのマニュアルを参照してください。

■想定される原因(仮想メモリ不足)

仮想メモリが不足してスレッドが生成できない場合、次の**OutOfMemoryError**がスローされる、あるいはエラーメッセージとして表示を行いプロセスが終了します。

```
java.lang.OutOfMemoryError: unable to create new native thread
```

また、OSの仮想メモリが不足した場合、次の**OutOfMemoryError**が出力されプログラムが終了します。

```
java.lang.OutOfMemoryError: requested サイズ bytes 制御名. Out of swap space?
```

サイズ:　　確保できなかったメモリの大きさ

制御名:　　メモリが確保できなかったJava VMの制御名(該当情報がある場合にだけ表示)

仮想メモリが不足した場合は、他の不要なプロセスを終了して仮想メモリに余裕を持たせるか、物理メモリ(RAM)またはスワップファイルを拡張して仮想メモリを増やすようにチューニングを行ってください。

7.4.3 ハングアップ(フリーズ)した場合

本節では、Javaプロセスが残っているにもかかわらず、プログラムが無応答になった場合、考えられる原因とその対処方法を説明します。

■想定される原因(デッドロック)

デッドロックが発生した場合、そのスレッドが停止されます。
ハングアップしたときに、スレッドダンプを採取して、デッドロックがないかどうかを確認してください。

スレッドダンプの採取方法および解析方法の詳細は、“7.3.3 スレッドダンプ”を参照してください。

■想定される原因(ガーベジコレクション)

ガーベジコレクションが発生すると、ガーベジコレクションが終了するまでの間、Javaアプリケーションのすべてのスレッドが停止されます。これにより、Javaアプリケーションがハングアップしたかのように見える場合があります。

ガーベジコレクションのログを採取して、ガーベジコレクションが動作したタイミングを照合してください。ガーベジコレクションが原因で無応答のような現象になる場合は、Javaヒープをチューニングして、ガーベジコレクションの動作具合を改善してください。

ガーベジコレクションのログを採取する方法は、“7.3.1 ガーベジコレクションのログ出力”を参照してください。
Javaヒープのチューニング方法は、“7.5.2 Javaヒープのチューニング”を参照してください。

■想定される原因(JNI処理の異常)

JNI経由でJava以外の言語で開発したネイティブモジュールと連携する際、JNIの使用方法を誤ると、ハングアップの原因となります。

このようなときは、“-Xcheck:jni”オプションを指定して、JNI処理でメッセージが出力されないかどうかを確認してください。“-Xcheck:jni”オプションの詳細は、“7.3.5 JNI処理異常時のメッセージ出力”を参照してください。

JNI処理に誤りがなくても、JNIモジュールで異常終了またハングアップが発生すると、Javaアプリケーションがハングアップする場合があります。たとえば、スレッドアンセーフな関数を使用している場合は、注意が必要です。

スレッドアンセーフな関数の例

Solaris

次の関数を使用したときに、ハングした事例があります。

- vfork

7.4.4 プロセスが消滅(異常終了)した場合

本節では、何の痕跡も残さずに突然プロセスが消滅した場合に、考えられる原因とその対処方法を説明します。

■想定される原因(スタックオーバーフロー)

FJVMには、**スタックオーバーフロー**検出時にメッセージを出力する機能を備えています。FJVMログを分析することにより、スタックオーバーフローが発生したかどうかを確認することができます。FJVMログの分析方法は、“7.2.8 スタックオーバーフロー検出時のメッセージ出力機能”を参照してください。

スタックオーバーフローが発生したことを確認できた場合、該当するスタックのサイズをチューニングしてください。スタックのチューニング方法は、“7.5.1 スタックのチューニング”を参照してください。

Windows

通常、スタックオーバーフローが発生した場合、java.lang.StackOverflowErrorがスローされ、ワトソン博士が検知してユーザダンプおよびワトソンログを出力します。

しかし、OSが高負荷状態になったり、スタックオーバーフロー発生時のスタック残量が少なかったりすると、OSからFJVMにもワトソン博士にも制御が渡らないまま、痕跡を残さずにプロセスが消滅することがあります。

したがって、プロセスが消滅した原因が不明な場合は、スタックのサイズを拡張して、現象が改善できるかどうかを確認してください。スタックのサイズを拡張しても改善できない場合は、別の原因を調査してください。

なお、ワトソン博士の説明は、“7.3.4 クラッシュダンプ・コアダンプ”を参照してください。

■想定される原因(長時間コンパイル処理の検出機能による終了)

FJVMの“長時間コンパイル処理の検出機能”による終了の可能性があります。
詳細は、“7.2.10 長時間コンパイル処理の検出機能”を参照してください。

Javaアプリケーションを“-XX:CompileTimeout”オプションで起動した場合は、標準出力にFJVMからのメッセージが出力されていないかどうかを確認してください。

■想定される原因(シグナルハンドラ)

Solaris Linux

Java VM以外のモジュールで、シグナルハンドラを登録した場合、Javaアプリケーションが正常に動作せずに、異常終了することがあります。詳細は、“7.2.11.2 異常終了時のシグナルハンドラ情報 ”を参照してください。

FJVMを使用している場合は、FJVMログにシグナルハンドラ情報が出力されますので、それを確認してください。

■想定される原因(JNI処理の異常)

JNI経由でJava以外の言語で開発したネイティブモジュールと連携する際、JNIの使用方法を誤ると、プロセス消滅の原因となります。

このようなときは、“-Xcheck:jni”オプションを指定して、JNI処理でメッセージが出力されないかどうかを確認してください。“-Xcheck:jni”オプションの詳細は、“7.3.5 JNI処理異常時のメッセージ出力”を参照してください。

JNI処理に誤りがなくても、ネイティブモジュールで異常終了またハングアップが発生すると、Javaアプリケーションのプロセスが消滅する場合があります。たとえば、スレッドアンセーフな関数を使用している場合は、注意が必要です。

スレッドアンセーフな関数の例

Solaris

次の関数を使用したときに、障害が発生した事例があります。

- vfork

■想定される原因(プログラムによる終了)

Javaプロセスが、特別なメッセージ出力などがないまま、予想外の状態で終了した場合、原因の1つとして、次のいずれかが考えられます。

- java.lang.Runtime.exit()を予想外の箇所で実行した
- java.lang.Runtime.halt()を予想外の箇所で実行した
- java.lang.System.exit()を予想外の箇所で実行した

FJVMを使用している場合は、“7.2.6 Java VM終了時における状態情報のメッセージ出力機能”を参照して、対処してください。

■想定される原因(Windows Server(R) 2003の問題)

Windows

Windows Server(R) 2003の初期版においては、ユーザダンプが出力されない問題をはじめとして、その他にもJavaの実行動作に影響を及ぼす問題などがあります。
たとえば、次のような問題があります。

- http://support.microsoft.com/kb/836080/en-us
- http://support.microsoft.com/kb/837018/en-us
- http://support.microsoft.com/kb/841176/en-us

Windows Server(R) 2003を使用する場合は、Service Pack 1以降またはHotfixを適用してください。

7.4.5 スローダウンが発生した場合

本節では、Javaアプリケーションの動きが遅くなる現象(スローダウン)が発生した場合に、考えられる原因とその対処方法を説明します。

■想定される原因(ガーベジコレクション)

ガーベジコレクションが発生すると、ガーベジコレクションが終了するまでの間、Javaアプリケーションのすべてのスレッドが停止されます。このため、Javaアプリケーションのレスポンス(応答)が遅くなる場合があります。

ガーベジコレクションのログを採取して、ガーベジコレクションが動作したタイミングとスローダウンが発生したタイミングを照合してください。ガーベジコレクションが原因でスローダウンになる場合は、Javaヒープをチューニングして、ガーベジコレクションの動作具合を改善してください。

ガーベジコレクションのログを採取する方法は、“7.3.1 ガーベジコレクションのログ出力”を参照してください。
Javaヒープのチューニング方法は、“7.5.2 Javaヒープのチューニング”を参照してください。

スローダウンの事例

同じソフトウェアとJavaアプリケーションが動作している複数のWebサーバのうち、一部のサーバだけがスローダウンしたという事例があります。この原因は、マシンに搭載されている物理メモリ(RAM)の容量の違いでした。
物理メモリ(RAM)が少ないマシンの場合、ガーベジコレクションの実行に伴い、ディスクのスワッピングが発生し、スローダウンすることがあります。

7.4.6 SIGBUS発生により異常終了した場合

Solaris

Solaris上で実行しているプロセスが、以下の状態のSIGBUS発生により異常終了した場合は、システムのメモリ資源／スワップ不足により発生した異常終了です。

signal no : 10(SIGBUS)

signal code : 3(BUS_OBJERR)

signal error: 12(ENOMEM)

そして、Javaプロセスが異常終了した場合に出力されるFJVMログにおいて、以下の情報が出力されている場合が、上記状態に相当します。

```
siginfo:si_signo=10, si_errno=12, si_code=3, si_addr=16進数
```

異常終了時の情報として上記情報が出力された場合は、不要なプロセスを終了して仮想メモリに余裕を持たせるか、物理メモリ(RAM)またはスワップファイルを拡張して仮想メモリを増やすようにチューニングを行ってください。

7.4.7 EXTP4435メッセージまたはISJEE_OM1018メッセージが出力された場合

Interstage Application Server配下のJavaアプリケーション実行時において、以下のメッセージが出力された場合は、メモリ領域不足事象が原因による異常となります。

Javaヒープをチューニングしてください。

Javaヒープのチューニング方法は、“7.5.2 Javaヒープのチューニング”を参照してください。

- EXTP4435メッセージ
- ISJEE_OM1018メッセージ

各メッセージの内容については、“メッセージ集”または“Java EE運用ガイド”のメッセージ情報を参照してください。
なお、IJServerのコンテナ情報ログ(info.log)およびIJServerクラスタのJava VMログ(jvm.log)に出力される「Java VMのヒープ域不足の詳細情報」の出力形式は、図1のとおりです。

図1 Java VMのヒープ域不足の詳細情報の出力形式

```
--------------------------------------------------------------------------------
OutOfMemory Log
--------------------------------------------------------------------------------
pid=process_id
heap_type=heap_type_code
heap_size=heap_size
max_heap_size=max_heap_size
perm_size=perm_size
max_perm_size=max_perm_size
--------------------------------------------------------------------------------
VM is terminated by occurred OutOfMemoryError on heap_type.
stack_trace
```

process_id: メモリ領域不足事象が発生したJavaプロセスのプロセス番号。

heap_type_code: メモリ領域不足事象が発生した領域に対応する番号です。表示される番号については、**heap_type**の説明を参照してください。

heap_size: メモリ領域不足事象が発生した際に使用中となっているメモリ割り当てプールのサイズ(単位:byte)。

max_heap_size: 利用可能なメモリ割り当てプールの最大サイズ(単位:byte)。

perm_size: メモリ領域不足事象が発生した際に使用中となっているPermanent世代領域のサイズ(単位:byte)。

max_perm_size: 利用可能なPermanent世代領域の最大サイズ(単位:byte)。

heap_type: 不足領域情報(メモリ領域不足事象が発生した領域の名前など)を表示します。不足領域情報としては以下の項目があります。

- **Java heap**
  オブジェクト生成要求において、メモリ割り当てプール(New世代領域またはOld世代領域)に対してメモリ領域不足事象が発生した場合です。
  この項目のときの**heap_type_code**は「1」になります。
  なお、JDK/JRE 5.0、6のエルゴノミクス機能によるメモリ領域不足事象の検出機能が有効な場合で、かつ当該機能によりメモリ領域不足事象を検出した場合、この項目になる場合があります。

- **Java heap in critical section**
  意味は**「Java heap」**と同じですが、メモリ領域不足事象発生時、クリティカルセクション状態でGC処理の実行が抑止されていたことを示しています。(注1)

- **Java Perm**
  オブジェクト生成要求において、Permanent世代領域に対してメモリ領域不足事象が発生した場合です。
  この項目のときの**heap_type_code**は「2」になります。
  なお、JDK/JRE 5.0、6のエルゴノミクス機能によるメモリ領域不足事象の検出機能が有効な場合で、かつ当該機能によりメモリ領域不足事象を検出した場合、この項目になる場合があります。

- **Java Perm space in critical section**
  意味は**「Java Perm」**と同じですが、メモリ領域不足事象発生時、クリティカ

ルセクション状態でGC処理の実行が抑止されていたことを示しています。(注1)

- **C heap 制御情報**
  スタックやヒープなど、Javaヒープ以外の領域に対してメモリ領域不足事象が発生した場合です。
  この項目のときの**heap_type_code**は「0」になります。
  また、次行に制御情報(メモリが確保できなかったJava VM制御の情報)が出力される場合(注1)があります。

- **unknown space**
  Javaアプリケーション実行時における配列生成式の評価の段階で、配列オブジェクトの長さ(配列要素の数)から、当該配列オブジェクトを割り当てるための領域が十分でないと評価された場合(配列の長さ(配列要素の数)が大きすぎ、配列オブジェクトとしての大きさが2ギガバイト程度もしくはそれ以上の大きさになる配列の定義がある場合)。
  またはクラスのロード処理でメモリ不足が発生した場合。
  なお、JDK/JRE 6のエルゴノミクス機能によるメモリ領域不足事象の検出機能が有効な場合で、かつ当該機能によりメモリ領域不足事象を検出した場合、この項目になる場合があります。
  この項目のときの**heap_type_code**は「-1」になります。

stack_trace: メモリ領域不足事象が発生したスレッドがJavaアプリケーションを実行しているスレッドだった場合は、当該スレッドのスタックトレースが出力されます。それ以外のスレッドの場合、またはリソース不足によりスタック情報が取り出せない場合は、スタックトレースは出力されません。(注1)

(注1) RHEL6(x86)/RHEL6(Intel64)に対応する富士通版JDK/JREを用いたJavaプロセスの場合に、この情報が出力される可能性があります。

図2 Java VMのヒープ域不足の詳細情報の出力例

【RHEL5(x86)上で動作するJDK/JRE 5.0の場合の出力例】

```
--------------------------------------------------------------------------------
OutOfMemory Log
--------------------------------------------------------------------------------
pid=4636
heap_type=1
heap_size=136816
max_heap_size=4194304
perm_size=1811104
max_perm_size=67108864
--------------------------------------------------------------------------------
```

【RHEL6(x86)上で動作するJDK/JRE 6の場合の出力例】

```
--------------------------------------------------------------------------------
OutOfMemory Log
--------------------------------------------------------------------------------
pid=4696
heap_type=1
heap_size=136800
max_heap_size=6291456
perm_size=2052320
max_perm_size=67108864
--------------------------------------------------------------------------------
```

```
VM is terminated by occurred OutOfMemoryError on Java heap.
"main" prio=6 tid=0x00307000 nid=0x12a8 runnable [0x0092f000]
   java.lang.Thread.State: RUNNABLE
        at test.<init>(test.java:10)
        at test.main(test.java:5)
```

JDK/JRE 6からスタックトレース情報にスレッドの状態を表す情報(トレース情報の前の行)が表示されるようになりました。

図中の例では「java.lang.Thread.State: RUNNABLE」となっていますが、"RUNNABLE"の部分が、"NEW"、"TIMED_WAITING (sleeping)"、"WAITING (on object monitor)"、"TIMED_WAITING (on object monitor)"、"WAITING (parking)"、"TIMED_WAITING (parking)"、"BLOCKED (on object monitor)"、"TERMINATED"、"UNKNOWN"などの表示の場合があります。

7.5 チューニング方法

チューニングのポイントとして、次があります。

- メモリ消費量と処理速度は深い関係にあり、メモリ消費量を抑制すれば処理速度が低下するのが一般的です。しかし、JDK/JREにおいては、Javaヒープのサイズを必要以上に大きく確保した場合、New世代GCの発生頻度が少なくなる反面、FullGCの処理時間が増大し、処理速度が低下するという特徴があります。
- プロセスに割り当てられたメモリ資源は限られています。JDK/JREにおいては、スタック、Javaヒープおよびネイティブモジュールの動作に必要な領域などの各セグメントがユーザ空間に割り当てられます。このため、あるセグメントの領域を大きく確保すれば、その分だけ他のセグメントの領域が少なくなります。

上のポイントを踏まえた上で、JDK/JREのチューニングを行います。

7.5.1 スタックのチューニング

本節では、Javaアプリケーションで使用するスレッドの**スタック**のチューニング方法および、チューニングによる影響範囲を説明します。

■チューニング方法

Java APIで生成するスレッドのスタックサイズは、“-Xss”オプションで指定することができます。

“-Xss”オプションは、バイト単位でスタックサイズを指定します。例えば、スタックサイズを512KBに設定する場合、“-Xss512k”と指定します。

なお、mainメソッドが実行されるスレッドはJava VMの起動前に作成されたスレッドである(Java APIで作成されたスレッドではない)ため、“-Xss”オプションによるスタックサイズの指定は効果がありません。

またJDK/JRE内で実行されるJavaメソッドを自動的にコンパイルする専用スレッド(自動コンパイル用スレッド)のスタックサイズは、“-XX:CompilerThreadStackSize”オプションで指定することができます。

通常、自動コンパイル用スレッドのスタックサイズを指定する必要はありません。

“-XX:CompilerThreadStackSize”オプションは、キロバイト(Kバイト)単位で自動コンパイル用スレッドのスタックサイズ指定します。例えば、スタックサイズを1024KBに設定する場合、”-XX:CompilerThreadStackSize=1024”と指定します。

Java APIで生成したスレッドおよび自動コンパイル用スレッドのデフォルトのスタックサイズを、“表7.5 Java APIで生成したスレッドおよび自動コンパイル用スレッドのデフォルトのスタックサイズ”に示します。

表7.5 Java APIで生成したスレッドおよび自動コンパイル用スレッドのデフォルトのスタックサイズ

JDK/JREのバージョン	OS	JDK/JREの実行モード	Java APIで生成したスレッド(注2)	自動コンパイル用スレッド (Client VM)(注2)	自動コンパイル用スレッド (FJVM)
JDK/JRE 1.4	Windows (ただし、Windows Server(R) for Itanium-based Systemsを除く)	32ビットモード	256KB	256KB	2048KB

JDK/JREのバージョン	OS	JDK/JREの実行モード	Java APIで生成したスレッド(注2)	自動コンパイル用スレッド (Client VM)(注2)	自動コンパイル用スレッド (FJVM)
	RHEL-AS4(x86)/AS4(EM64T) Solaris	32ビットモード	512KB	512KB	2048KB
	Windows Server(R) 2003 for Itanium-based Systems (注1) RHEL-AS4(IPF)	64ビットモード	1024KB	－(注3)	4096KB
JDK/JRE 5.0	Windows （ただし、Windows Server(R) for Itanium-based Systemsを除く）	32ビットモード	256KB	256KB	2048KB
	Linux for x86 Linux for Intel64 Solaris	32ビットモード	512KB	512KB	2048KB
	Windows Server(R) x64 Editions Windows Server(R) for Itanium-based Systems (注1) RHEL5(Intel64) RHEL6(Intel64) Linux for Itanium	64ビットモード	1024KB	－(注3)	4096KB
JDK/JRE 6	Windows （ただし、Windows Server(R) for Itanium-based Systemsを除く）	32ビットモード	320KB	320KB	2048KB
	Linux for x86 Linux for Intel64	32ビットモード	320KB	512KB	2048KB
	Solaris	32ビットモード	512KB	512KB	2048KB
	Windows Server(R) x64 Editions RHEL5(Intel64) RHEL6(Intel64)	64ビットモード	1024KB	－(注3)	4096KB

注1) Windows Server(R) for Itanium-based Systemsでは、デフォルト値または“-Xss”オプション/“-XX:CompilerThreadStackSize”オプションで指定した値の2倍の大きさが、スレッドに対して割り当てられるスタックの実際のサイズとなります。詳細は、“基礎知識”の“7.1.4 スタック”内にある注意事項“スタックサイズについて”を参照してください。

注2) Windows版JDK/JREにおけるスタックサイズは、java.exeなどWindows版JDK/JREが提供するJDKツールを用いた場合の値です。JNIを用いて独自にJava VMを起動しているWindowsアプリケーションの場合は、Java VMを起動したプログラムのメインスレッドに対するスタックサイズと同じ値になります。

注3) 実行モードが64ビットモードのJDK/JREでは、Java HotSpot Client VMは搭載していません。

注4) スタック領域の実際の管理はOSが行います。そのためスタック領域に関する管理方法/動作仕様については、JDK/JREを実行する各OSの仕様に依存します。

■チューニングの影響範囲

スタックのサイズを変更した場合の影響範囲を、次に示します。

- ***スタック***のサイズを縮小した場合、スタックオーバーフローが発生することがあります。
- ***スタック***のサイズを拡張した場合、その分ユーザ空間や仮想メモリが少なくなるため、Javaヒープやネイティブモジュールの動作に必要な領域を確保できず、メモリ不足になることがあります。

7.5.2 Javaヒープのチューニング

本節では、**Javaヒープ**のチューニング方法および、チューニングによる影響範囲を説明します。

■チューニング方法

Javaヒープの各領域のサイズは、“表7.6 Javaヒープに関するオプション”に示すオプションをJava起動時に指定することで設定ができます。

なお、**メモリ割り当てプール**のデフォルトの初期値および最大値を、“表7.7 メモリ割り当てプールのデフォルトのサイズ”に示します。
また、**Permanent世代領域**のデフォルトの初期値および最大値を、“表7.8 Permanent世代領域のデフォルトのサイズ”に示します。
また各オプションにおいて、“表7.6 Javaヒープに関するオプション”中に記載されていないデフォルト値を、“表7.9 Javaヒープに関するオプション(-XX:NewSize/-XX:NewRatio)のデフォルト値”に示します。

表7.6 Javaヒープに関するオプション

オプション	オプションの機能
-Xms	**メモリ割り当てプール**の初期値を指定します。 たとえば、**メモリ割り当てプール**の初期値を128MBに設定する場合、”-Xms128m”と指定します。 本オプションのデフォルト値は“表7.7 メモリ割り当てプールのデフォルトのサイズ”のとおりです。
-Xmx	**メモリ割り当てプール**の最大値を指定します。 たとえば、**メモリ割り当てプール**の最大値を256MBに設定する場合、”-Xmx256m”と指定します。 本オプションのデフォルト値は“表7.7 メモリ割り当てプールのデフォルトのサイズ”のとおりです。
-XX:NewSize (注1)	**New世代領域**のヒープサイズを指定します。 たとえば、**New世代領域**のヒープサイズを128MBに設定する場合、”-XX:NewSize=128m”と指定します。 本オプションの指定が有効な場合のデフォルト値は以下のとおりです。 ・type0のCMS付きパラレルGCを使用している場合は、メモリ割り当てプールの初期値の1/8です。 ・type1のCMS付きパラレルGCを使用している場合は、メモリ割り当てプールの初期値の1/3です。 ・type2のCMS付きパラレルGCを使用している場合は、メモリ割り当てプールの初期値の1/16です。 ・上記以外の場合のデフォルト値は“表7.9 Javaヒープに関するオプション(-XX:NewSize/-XX:NewRatio)のデフォルト値”のとおりです。
-XX:MaxNewSize (注1)	**New世代領域**の最大ヒープサイズを設定します。 たとえば、**New世代領域**の最大ヒープサイズを128MBに設定する場合、”-XX:MaxNewSize=128m”と指定します。 本オプションの指定が有効な場合のデフォルト値は以下のとおりです。 ・type0のCMS付きパラレルGCを使用している場合は、メモリ割り当てプールの最大値の1/8です。 ・type1のCMS付きパラレルGCを使用している場合は、メモリ割り当てプールの最大値の1/3です。 ・type2のCMS付きパラレルGCを使用している場合は、メモリ割り当てプールの最大値の1/16です。 ・上記以外の場合、デフォルト値はありません(本オプションの値を用いてNew世代領域の最大ヒープサイズは決定されません)。

オプション	オプションの機能
-XX:NewRatio (注1)(注4)	**New世代領域**と**Old世代領域**のサイズ比率を指定します。 たとえば、**New世代領域**と**Old世代領域**のサイズ比率を2とする場合、”-XX:NewRatio=2”と指定します。 本オプションの指定が有効な場合のデフォルト値は“表7.9 Javaヒープに関するオプション(-XX:NewSize/-XX:NewRatio)のデフォルト値”のとおりです。
-XX:SurvivorRatio (注1)(注2)(注5)	**New世代領域**を構成する**Eden領域**と**Survivor領域**のサイズ比率を指定します。 たとえば、**Eden領域**と**Survivor領域**のサイズ比率を8とする場合、”-XX:SurvivorRatio=8”と指定します。 本オプションの指定が有効な場合のデフォルト値は以下のとおりです。 ・ Solaris版JDK/JRE 1.4、5.0でシリアルGCを使用している場合のデフォルト値は32です。 ・ 上記以外の場合のデフォルト値は8です。
-XX:TargetSurvivorRatio (注1)(注2)(注5)	**ガーベジコレクション(GC)**処理後の生存オブジェクトが**Survivor領域**を占める割合を、指定したパーセンテージ値に調整します。 たとえば、GC処理後の生存オブジェクトが**Survivor領域**を占める割合を半分とする場合、”-XX:TargetSurvivorRatio=50”と指定します。 本オプションの指定が有効な場合のデフォルト値は50です。
-XX:PermSize	**Permanent世代領域**の初期値を指定します。 たとえば、**Permanent世代領域**の初期値を32MBに設定する場合、”-XX:PermSize=32m”と指定します。 本オプションのデフォルト値は“表7.8 Permanent世代領域のデフォルトのサイズ”のとおりです。
-XX:MaxPermSize	**Permanent世代領域**の最大値を指定します。 たとえば、**Permanent世代領域**の最大値を128MBに設定する場合、”-XX:MaxPermSize=128m”と指定します。 本オプションのデフォルト値は“表7.8 Permanent世代領域のデフォルトのサイズ”のとおりです。

注1) FJGCを使用する場合、このオプションへの指定値は無効となります。
注2) パラレルGCを使用する場合、このオプションへの指定値は無効となります。
注3) サイズを指定するオプションでは単位として次の文字を指定できます。
　KB(キロバイト)を指定する場合:”k”または”K”
　MB(メガバイト)を指定する場合:”m”または”M”
注4) CMS付きパラレルGCを使用する場合、このオプションへの指定値は無効となります。
注5) CMS付きパラレルGCを使用する場合、かつ-XX:UseFJcmsGC=type0指定でない場合、このオプションへの指定値は無効となります。

表7.7 メモリ割り当てプールのデフォルトのサイズ

JDK/JREのバージョン	OS	JDK/JREの実行モード	GC処理	初期値	最大値
JDK/JRE 1.4	Windows (ただし、Windows Server(R) for Itanium-based Systemsを除く) RHEL-AS4(x86)/AS4(EM64T)	32ビットモード	シリアルGC FJCG (注1)	2.0MB	64MB

JDK/JREのバージョン	OS	JDK/JREの実行モード	GC処理	初期値	最大値
	Solaris	32ビットモード	シリアルGC FJCG (注1)	3.5MB	
	Windows Server(R) 2003 for Itanium-based Systems RHEL-AS4(IPF)	64ビットモード	シリアルGC (注1)	3.5MB	
JDK/JRE 5.0	Windows （ただし、Windows Server(R) for Itanium-based Systemsを除く） Linux for x86 Linux for Intel64	32ビットモード	シリアルGC FJCG	2.0MB	64MB
			パラレルGC (注1)	5.375MB	
			CMS付きパラレルGC	(注2)	
	Solaris	32ビットモード	シリアルGC FJCG	3.5MB	
			パラレルGC (注1)	5.375MB	
			CMS付きパラレルGC	(注2)	
	Windows Server(R) x64 Editions RHEL5(Intel64) RHEL6(Intel64)	64ビットモード	シリアルGC	4.3125MB	84MB
			パラレルGC (注1)	5.75MB	
			CMS付きパラレルGC	(注2)	
	Windows Server(R) for Itanium-based Systems Linux for Itanium	64ビットモード	シリアルGC	4.5MB	88MB
			パラレルGC (注1)	5.75MB	
JDK/JRE 6	Windows(ただし、Windows Server(R) for Itanium-based Systemsは除く) Linux for x86 Linux for Intel64	32ビットモード	シリアルGC	5.0MB	64MB
			パラレルGC (注1)	8.0MB	
			CMS付きパラレルGC	(注2)	
	Solaris	32ビットモード	シリアルGC	6.125MB	
			パラレルGC (注1)	8.0MB	
			CMS付きパラレルGC	(注2)	
	Windows Server(R) x64 Editions	64ビットモード	シリアルGC	7.75MB	84MB

JDK/JREのバージョン	OS	JDK/JREの実行モード	GC処理	初期値	最大値
	RHEL5(Intel64) RHEL6(Intel64)		パラレルGC (注1)	9.1875MB	
			CMS付きパラレルGC	(注2)	

注1) デフォルトで使用されるGC処理です。

注2) メモリ割り当てプールの最大値が64MB未満の場合は、メモリ割り当てプールの最大値と等しくなります。メモリ割り当てプールの最大値が64MB以上の場合は、64MBとなります。

表7.8 Permanent世代領域のデフォルトのサイズ

JDK/JREのバージョン	OS	JDK/JREの実行モード	Java VM	初期値	最大値
JDK/JRE 1.4	Windows (ただし、Windows Server(R) for Itanium-based Systemsを除く) RHEL-AS4(x86)/AS4(EM64T) Solaris	32ビットモード	Java HotSpot Client VM	4MB	64MB
			FJVM (注1)	16MB	
	Windows Server(R) 2003 for Itanium-based Systems RHEL-AS4(IPF)	64ビットモード	FJVM (注1)	16MB	
JDK/JRE 5.0	Windows (ただし、Windows Server(R) for Itanium-based Systemsを除く) Linux for x86 Linux for Intel64 Solaris	32ビットモード	Java HotSpot Client VM	8MB	64MB
			FJVM (注1)	16MB	
	Windows Server(R) x64 Editions RHEL5(Intel64) RHEL6(Intel64)	64ビットモード	FJVM (注1)	20.75MB	84MB
	Windows Server(R) for				88MB

JDK/JREのバージョン	OS	JDK/JREの実行モード	Java VM	初期値	最大値
	Itanium-based Systems Linux for Itanium				
JDK/JRE 6	Windows(ただし、Windows Server(R) for Itanium-based Systemsは除く) Solaris Linux for x86 Linux for Intel64	32ビットモード	Java HotSpot Client VM	12MB	64MB
			FJVM (注1)	16MB	
	Windows Server(R) x64 Editions RHEL5(Intel64) RHEL6(Intel64)	64ビットモード	FJVM (注1)	20.75MB	84MB

注1) デフォルトで使用されるJava VMです。

表7.9 Javaヒープに関するオプション(-XX:NewSize/-XX:NewRatio)のデフォルト値

JDK/JREのバージョン	OS	JDK/JREの実行モード	Java VM	-XX:NewSize	-XX:NewRatio
JDK/JRE 1.4	Windows (ただし、Windows Server(R) for Itanium-based Systemsを除く) RHEL-AS4(x86)/AS4(EM64T)	32ビットモード	Java HotSpot Client VM	640KB	12
			FJVM (注1)		8
	Solaris	32ビットモード	Java HotSpot Client VM	2176KB	8
			FJVM (注1)		2
	Windows Server(R) 2003 for Itanium-based Systems RHEL-AS4(IPF)	64ビットモード	FJVM (注1)	2176KB	2
JDK/JRE 5.0	Windows (ただし、Windows Server(R) for Itanium-based Systemsを除く) Linux for x86 Linux for Intel64	32ビットモード	Java HotSpot Client VM	640KB	12
			FJVM (注1)		8
	Solaris	32ビットモード	Java HotSpot Client VM	2176KB	8
			FJVM (注1)		2
	Windows Server(R) x64 Editions	64ビットモード	FJVM (注1)	2624KB	2

JDK/JREのバージョン	OS	JDK/JREの実行モード	Java VM	-XX:NewSize	-XX:NewRatio
	RHEL5(Intel64) RHEL6(Intel64)				
	Windows Server(R) for Itanium-based Systems Linux for Itanium	64ビットモード	FJVM (注1)	2816KB	2
JDK/JRE 6	Windows(ただし、Windows Server(R) for Itanium-based Systemsは除く) Linux for x86 Linux for Intel64	32ビットモード	Java HotSpot Client VM	1024KB	12
			FJVM (注1)		8
	Solaris	32ビットモード	Java HotSpot Client VM	2176KB	8
			FJVM (注1)		2
	Windows Server(R) x64 Editions RHEL5(Intel64) RHEL6(Intel64)	64ビットモード	FJVM (注1)	2624KB	2

注1) デフォルトで使用されるJava VMです。

■チューニングの方針

Javaヒープをチューニングする際、次の方針があります。

1. **FullGC**を実行したにもかかわらず、メモリ不足が発生する場合、**GC**のログを採取し、メモリ割り当てプールまたは**Permanent世代領域**のいずれかの領域が不足しているかどうかを確認します。
2. **FullGC**はコストがかかります。このため、メモリ不足が発生しなくても、Javaアプリケーションがハングアップしたかのように一時的に無反応になる場合、**FullGC**の影響を受けている場合があります。**GC**のログを採取し、**Javaヒープ**が必要以上に大きなサイズになっていれば、**Javaヒープ**のサイズを縮小する方針でチューニングします。
3. 効率的な**New世代GC**に対して、**FullGC**はコストがかかります。このため、**New世代領域**と**Old世代領域**のサイズのバランスを考慮する必要があります。なお、FJVMでFJGCを使用する場合、New世代領域サイズ自動調整機能を実装しているため、通常はこのバランスを考慮する必要はありません。
4. 仮想メモリに余裕がある場合は、Javaプロセスを複数起動して、プロセス多重度を上げる方法を検討します。プロセス多重度を上げることにより、プロセスごとのユーザ空間を有効に使うことが可能になります。ただし、メモリのスワッピングによるスローダウンに注意する必要があります。

■チューニングの影響範囲

Javaヒープ全体のサイズを変更した場合の影響範囲を、次に示します。

1. **Javaヒープ**全体のサイズを縮小した場合、**GC**が頻発することがあります。
2. **Javaヒープ**全体のサイズを拡張した場合、**FullGC**に時間がかかることがあります。
3. **Javaヒープ**全体のサイズを拡張した場合、その分ユーザ空間や仮想メモリが少なくなるため、スタックやネイティブモジュールの動作に必要な領域を確保できず、メモリ不足になることがあります。

overcommit memory機能が有効な場合の注意事項 Linux

「overcommit memory機能」が有効な場合、Linuxは、Javaヒープの各領域の最大値に相当する仮想メモリ資源を、Java VMの起動時に、Javaプロセスに対して予約します。

このため、-Xms値と-Xmx値を異なる値にしてJavaプロセスを起動する場合、本機能の有効/無効によって、Javaプロセス起動時にJavaヒープとして必要となる仮想メモリの量が異なります。

- overcommit memory機能が無効、またはovercommit memory機能がないシステムの場合
  Javaヒープ用仮想メモリ量 ＝「-Xms値」＋「Perm域初期値」
- **overcommit memory機能が有効なシステムの場合**
  Javaヒープ用仮想メモリ量 ＝「-Xmx値」＋「-XX:MaxPermSize値」

  この結果、仮に同量の仮想メモリ資源を持つシステムの場合であっても、本機能の有効/無効によって、同時に起動できるJavaプロセスの数が異なる場合があります。
  Linuxで仮想メモリ資源の見積もりを行う場合には、overcommit memory機能の有無に注意してください。

7.6 Javaツール機能

本製品では、Javaプログラムのチューニングやトラブルシューティングに有用なツールを提供しています。ツールには、次の4つがあります。

- メソッドのトレースを出力するメソッドトレース機能
- チューニングやメモリリーク検出のためのQualyzer
- Javaヒープ使用量を出力するjheap
- スレッドダンプを出力するスレッドダンプツール(Windows(R)のみ)

ツール本体は、次に格納してあります。

Windows

- JDK 1.4使用時: [Interstageインストールフォルダ]¥jdk14¥tools
- JRE 1.4使用時: [Interstageインストールフォルダ]¥jre14¥tools
- JDK 5.0使用時: [Interstageインストールフォルダ]¥jdk5¥tools
- JRE 5.0使用時: [Interstageインストールフォルダ]¥jre5¥tools
- JDK 6使用時 : [Interstageインストールフォルダ]¥jdk6¥tools
- JRE 6使用時 : [Interstageインストールフォルダ]¥jre6¥tools

Solaris Linux

以下の“$DIR”はインストール時に指定する相対ディレクトリ名です。“$DIR”のシステム推奨名は“opt”です。

- JDK 1.4使用時: /$DIR/FJSVawjbk/jdk14/tools
- JRE 1.4使用時: /$DIR/FJSVawjbk/jre14/tools
- JDK 5.0使用時: /$DIR/FJSVawjbk/jdk5/tools
- JRE 5.0使用時: /$DIR/FJSVawjbk/jre5/tools
- JDK 6使用時 : /$DIR/FJSVawjbk/jdk6/tools
- JRE 6使用時 : /$DIR/FJSVawjbk/jre6/tools

各ツールの詳細は、“トラブルシューティング集”の“Javaツール機能”を参照してください。

Qualyzerの制限

Qualyzerは機能実現のためJVMPIを使用しています。
JVMの提供する機能範囲の関係で使用するJDK/JREにより次の制限があります。

- JDK/JRE 5.0の場合:GC処理がシリアルGCである必要があります

・ JDK/JRE 6の場合：使用できません（未提供）

付録A　CORBAサービスの動作環境ファイル

CORBAサービスの動作環境ファイルについて説明します。各ファイルは、以下に格納されます。

格納パス

Windows

C:¥INTERSTAGE¥ODWIN¥etc (インストールパスはデフォルト)

Solaris

/etc/opt/FSUNod
(インストールパスはデフォルト、
インストール時に動作環境ディレクトリ(“Fixed configuration install directory”)として変更可能)

Linux

/etc/opt/FJSVod

ファイル

(Interstage Application Server Enterprise Editionにおいて提供)

config
gwconfig
inithost/initial_hosts
nsconfig
irconfig

(Interstage Application Server Standard-J Editionにおいて提供)

config
inithost/initial_hosts
nsconfig
irconfig

(Interstage Web Serverにおいて提供)

config
inithost/initial_hosts

- 上記以外のファイルは、CORBAサービスの動作環境としてカスタマイズできません。エディタなどで編集しないでください。
- 上記のファイル内に日本語は記載できません。
- システムの異常停止などに起因して動作環境ファイルなどの資源が破壊されると、CORBAサービスが正常に起動できない場合があります。
  正常に起動できない、かつ以下のメッセージが発生する場合は、資源が破壊されている可能性があるので、環境の再構築を行うか、バックアップした資源を復元してCORBAサービスの再起動を行う必要があります。
  メッセージ番号:od10400, od10402, od10404, od10406, od10504, od10509, od10510
  万が一のため、運用環境を構築したら資源のバックアップを行うことを推奨します。
  バックアップについては、“運用ガイド(基本編)”の“メンテナンス(資源のバックアップ)”で説明されています。

A.1 config

■概要

configファイルは、CORBAサービスの各種動作環境に関する定義が格納されたファイルです。

■ファイル名

Windows

C:¥INTERSTAGE¥ODWIN¥etc¥config (インストールパスはデフォルト)

Solaris

Solarisサーバ:
/etc/opt/FSUNod/config (インストールパスはデフォルト)
Windows(R)クライアント:
C:¥INTERSTAGE¥ODWIN¥etc¥config (インストールパスはデフォルト)

Linux

Linuxサーバ:
/etc/opt/FJSVod/config (インストールパスはデフォルト)
Windows(R)クライアント:
C:¥INTERSTAGE¥ODWIN¥etc¥config (インストールパスはデフォルト)

■ファイル内情報

configファイルは、以下の形式で値を設定します。

◆形式:

パラメタ名 = 設定値

半角のシャープ(#)を行の先頭に指定した場合は、その行はコメントとして扱われます。また、空行は解析時に無視されます。
コメント

◆記述例:

```
# comment
period_receive_timeout = 72
```

◆パラメタ:

以下の動作環境について、パラメタ設定値を変更することができます。

- ホスト情報に関する動作環境
- ネットワーク環境に関する動作環境
- アプリケーション資源に関する動作環境
- タイムアウト監視に関する動作環境
- セキュリティ機能に関する動作環境
- コード変換機能に関する動作環境
- 保守機能に関する動作環境

- パラメタ値を変更した場合、次回のCORBAサービス起動時より有効となります。
- configファイル内に日本語は記載できません。
- 下表に記載していないパラメタは初期値を変更しないでください。
- 備考欄に“必須パラメタ”と記載されているパラメタは省略することはできません。(Solaris版および Linux版には必須パラメタはありません)
- 数値を指定するパラメタ(period_receive_timeout等)に数値以外の文字列(“abc”など)を指定した場合、“0”が設定されたものとして扱われます。

◆ホスト情報に関する動作環境

パラメタ名	初期値 省略値 指定範囲	意味	備考
IIOP_hostname	－ － －	マシンにIPアドレス(またはホスト名)が複数設定されている場合に、CORBAサーバアプリケーションで使用するIPアドレスを限定した運用を行う場合に設定します。 IPアドレス(またはホスト名)を設定すると、サーバアプリケーションのオブジェクトリファレンスの生成時には、ここで設定したIPアドレスが組み込まれ、クライアントからの接続時に使用されます。また、CORBAサービスは指定されたIPアドレスのみでバインドを行います。 本パラメタが省略された場合はすべてのIPアドレスに対してバインドが行われます。ただし、IPv4/IPv6のどちらの(または両方の)IPアドレスをバインドするかどうかはIP-versionパラメタの設定に依存します。	サーバ機能のみ有効。 (注1) (注2) (注3)
IIOP_port	8002 － －	CORBAサービスが使用するポート番号。 Windowsの場合、省略できません。 Solaris、Linuxの場合、初期値(8002)以外を指定するときは必ず指定してください。	(注4)

注1)
Interstage Web Serverでは指定不可。

注2)
例えばLANカードが複数装着されたマシンにおいて、1つのLANカードからのみ接続要求を受け付けることができます。
ホスト名が指定された場合はIP-versionの値に従って名前解決が行われます。
IP-versionがv4-dualの場合はIPv4での名前解決が優先的に行われます。
IP-versionがv6の場合はIPv6での名前解決が優先的に行われます。
Windows版においてリンクローカルおよびサイトローカルのIPv6アドレスを指定する場合はscope-idも記載する必要があります。(例: fe80::1234:5678:9abc:def0%4)

注3)
必要のない限り本パラメタを設定しないでください。注2)に記述されているような特殊用途以外では設定する必要はありません。誤ったホスト名を設定するとInterstageの起動が失敗します。
また、“localhost”を設定すると“127.0.0.1”(IPv4環境の場合)のみでバインドが行われ、他ホストからのリクエストが受け付けられなくなりますので“localhost”と設定しないで下さい。“127.0.0.1”(IPv4環境の場合)のIPアドレスで定義されているホスト名を設定した場合も同様に他ホストからのリクエストが受け付けられなくなります。
LinuxではOSインストール直後の状態では自ホスト名に対するIPアドレスが127.0.0.1に設定されており、自ホスト名をIIOP_hostnameに設定すると他ホストからの接続を受け付けることができなくなります。

注4)
Windowsの場合、必須パラメタです。
Solaris、Linuxの場合、この値が無効になると/etc/servicesの設定値が有効になります。

◆ネットワーク環境に関する動作環境

パラメタ名	初期値 省略値 指定範囲	意味	備考
con_accept	all all all, localhost	・ all: すべてのマシンからの接続を受け付けます。 ・ localhost: クライアントからの接続受付を自ホストに制限する場合に指定します。自ホストからの接続のみを受け付け、他ホストからの接続を受け付けません。 システムセキュリティなどの理由で、他ホストからの接続要求を許可しない場合に指定してください。	サーバ機能のみ有効。 (注)
IP-version	v4-dual v4-dual v4, v4-dual, v6	運用するIPバージョンを設定します。 ・ v4: IPv4のみを利用してCORBAアプリケーションを運用します(IPv6は使用しません)。 ・ v4-dual: IPv4およびIPv6を利用してCORBAアプリケーションを運用します。 サーバとして動作する際はIPv4およびIPv6の両方を受け付けます。 クライアントとしての動作する際にIPv4を優先的に利用します。 ・ v6: IPv4およびIPv6環境で、CORBAアプリケーションを運用します。 サーバとして動作する際はIPv4およびIPv6の両方を受け付けます。 クライアントとしての動作する際にIPv6を優先的に利用します。 IPv6に対応していない環境で“v4-dual”もしくは“v6”を指定した場合、“v4”が設定されます。	
read_interval_timeout	30 30 0～100000000	ソケットに対する読み込みの待機時間。 この時間を超えても読み込みできない状態が持続する場合、アプリケーションにシステム例外(COMM_FAILURE)が通知されます。 この値が実際の時間(秒)となります。0を設定した場合、時間監視をしません。 このパラメタによる監視は電文の受信処理が始まってから開始されます。例えば、リプライの受信待ちの状態においてパケットが1つも届かなければread_interval_timeoutによる監視は行いません。この場合はperiod_receive_timeoutによる監視が行われます。パケットが1つでも届くと電文の受信処理を開始するためread_interval_timeoutによる監視が行われます。	
write_interval_timeout	30 30 0～100000000	ソケットに対する書き込みの待機時間。 この時間を超えて書き込みできない状態が持続する場合、アプリケーションにシステム例外(COMM_FAILURE)が通知されます。 この値が実際の時間(秒)となります。0を設定した場合、時間監視をしません。	

パラメタ名	初期値 省略値 指定範囲	意味	備考
		このパラメタによる監視は電文の送信処理が始まってから開始されます。	
tcp_nodelay	no no yes, no	TCP_NODELAY機能を有効にするか無効にするかを設定します。 ・ yes: 電文送信時においてNagleアルゴリズムを無効にします。 ・ no : Nagleアルゴリズムは有効となります。 Nagleアルゴリズムが有効の場合、送信データのバッファリングを行うためネットワーク使用効率が上がります。Nagleアルゴリズムを無効にした場合、送信データのバッファリングを行わないためネットワーク使用効率が下がり通信全体の性能が下がる可能性がありますが、データの送受信に発生するタイムラグが減少し、応答性能が向上する場合があります。	

注)
Interstage Web Serverでは初期値から変更しないでください。

◆アプリケーション資源に関する動作環境(プロセス/スレッド多重度、使用コネクション数など)

これらのパラメタに実際に指定可能な値はOSの資源によって制限されます。

パラメタ名	初期値 省略値 指定範囲	意味	備考
max_exec_instance	512 (注8) 256 16～1000000	サーバアプリケーションのリクエスト実行用スレッド(またはプロセス)の最大数。	サーバ機能のみ有効。 (注1) (注3) (注4)
max_IIOP_local_init_con	256 256 1～1000000	クライアントアプリケーションが使用するサーバホストへのコネクションの最大値。	ポイント参照
max_IIOP_local_init_requests	4096 4096 1～1000000	クライアントアプリケーションが同時に送信できるリクエスト数の最大値。	
max_IIOP_resp_con	8 (注8) 8 1～500000	クライアントアプリケーションと確立できる接続の最大値。	サーバ機能のみ有効。 (注2) (注4) ポイント参照
limit_of_max_IIOP_resp_con	0 (意味参照) 0 (意味参照) 0～1000000	max_IIOP_resp_conの自動拡張の最大値。 0またはmax_IIOP_resp_con以上の値を指定してください。 0を指定すると以下の値が設定されます。 max_IIOP_resp_con × 1.3 (小数部分切り捨て)	サーバ機能のみ有効。 (注1) (注4) (注6)
max_IIOP_resp_con_extend_number	0 (意味参照) 0 (意味参照)	max_IIOP_resp_conの自動拡張の拡張数。 0を指定すると以下の値が設定されます。	サーバ機能のみ有効。

パラメタ名	初期値 省略値 指定範囲	意味	備考
	0～1000000	(limit_of_max_IIOP_resp_con － max_IIOP_resp_con) ÷ max_IIOP_resp_con (小数部分切り上げ)	(注1) (注6) (注7)
max_IIOP_resp_requests	128 (注8) 128 1～500000	サーバホストにおいて同時に受信できるリクエスト数の最大値。	サーバ機能のみ有効。 (注2) (注4) (注10)
limit_of_max_IIOP_resp_requests	0 (意味参照) 0 (意味参照) 0～1000000	max_IIOP_resp_requestsの自動拡張の最大値。 0またはmax_IIOP_resp_requests以上の値を指定してください。 0を指定すると以下の値が設定されます。 max_IIOP_resp_requests × 1.3 (小数部分切り捨て)	サーバ機能のみ有効。 (注1) (注4) (注6) (注10)
max_IIOP_resp_requests_extend_number	0 (意味参照) 0 (意味参照) 0～1000000	max_IIOP_resp_requestsの自動拡張の拡張数。 0を指定すると以下の値が設定されます。 (limit_of_max_IIOP_resp_requests － max_IIOP_resp_requests) ÷ max_IIOP_resp_requests (小数部分切り上げ)	サーバ機能のみ有効。 (注1) (注6) (注7)
max_processes	20 (注8) 16 1～1000000	最大プロセス数。(起動クライアント ＋ サーバ数)	サーバ機能のみ有効。 (注4) (注5)
max_impl_rep_entries	512 256 100～1000000	インプリメンテーションリポジトリの最大登録数。	サーバ機能のみ有効。 (注2)
number_of_common_buffer	0 (意味参照) 0 (意味参照) 0～500000 (注10)	CORBAサービスのキュー制御で使用するデフォルトバッファのバッファ数を指定します。 ワークユニット運用されているCORBAアプリケーションでワークユニット定義の“Buffer Number:通信バッファ数”を指定しているCORBAアプリケーションを除く、CORBAサービスの通信で使用します。 サーバマシン上で同時に処理される最大リクエスト数を指定してください。 0を指定すると、以下の値が設定されます。 max_IIOP_resp_requests × 0.2 (小数部分切り捨て)	サーバ機能のみ有効。 (注2) (注4)
limit_of_number_of_common_buffer	0 (意味参照) 0 (意味参照) 0～1000000 (注10)	number_of_common_bufferの自動拡張の最大値。 0またはnumber_of_common_buffer以上の値を指定してください。 0を指定すると以下の値が設定されます。 limit_of_max_IIOP_resp_requests	サーバ機能のみ有効。 (注1) (注4) (注6)
number_of_common_buffer_extend_number	0 (意味参照) 0 (意味参照) 0～1000000	number_of_common_bufferの自動拡張の拡張数。 0を指定すると以下の値が設定されます。 (limit_of_number_of_common_buffer － number_of_common_buffer) ÷ number_of_common_buffer (小数部分切り上げ)	サーバ機能のみ有効。 (注1) (注6) (注7)

パラメタ名	初期値 省略値 指定範囲	意味	備考
max_bind_instances	0 (意味参照) 0 (意味参照) 0～1000000	CORBAサービスに登録可能な「サーバプロセスとオブジェクトのバインド関係」の数。 0を指定すると以下の値が設定されます。 1024 × max_processes (計算結果が65535を超えた場合は65535)	サーバ機能のみ有効。 (注2) (注4) (注9)

注1)
Interstage Web Serverでは指定不可。

注2)
Interstage Web Serverでは初期値から変更しないでください。

注3)
設定値の目安:
登録アプリケーション数(*1) × プロセス最大多重度(*2) × スレッド最大多重度(*3)＋ 接続クライアント数(*4) ＋ 64

*1) OD_impl_instコマンドで登録したアプリケーション数
*2) OD_impl_instコマンドで指定するproc_conc_max値
*3) OD_impl_instコマンドで指定するthr_conc_maximum値
*4) isgendefコマンドのscale-valueに対応した接続クライアント数

注4)
サーバ機能では、本パラメタの設定値および実際の消費量をodprtcurparamコマンドにより確認することができます。

Solaris Linux
初期値より増加させる場合、システム資源(共用メモリなど)のチューニングが必要です。詳細については、“3.1.1 CORBAサービスのシステム環境の設定”を参照してください。
また、Linuxの場合は、bashまたはボーンシェルの場合はulimitコマンドを、Cシェルの場合はlimitコマンドを使用して、ファイルディスクリプタ数を“max_IIOP_resp_con値 ＋ max_processes値”だけ拡張してからCORBAサービスおよびCORBAアプリケーションを起動してください。

注5)
CORBAサービスのプロセス(CORBAサービス、ネーミングサービス、インタフェースリポジトリサーバ、インタフェースリポジトリキャッシュサーバ)も含みます。見積もりを行う場合、Interstageのサービスの使用分(20)にアプリケーション使用分を加算してください。
CORBAサービスのコマンドも含みます。コマンドを同時に複数起動する場合は、その数を加算してください。

注6)
自動拡張について
自動拡張を行うパラメタについてはlimit_of_[パラメタ名]というパラメタと[パラメタ名]_extend_numberというパラメタが存在します。例えば、max_IIOP_resp_conというパラメタについてはlimit_of_max_IIOP_resp_con・max_IIOP_resp_con_extend_numberが存在します。
そして、各パラメタは初期値を[パラメタ名]、最大値をlimit_of_[パラメタ名]として、[パラメタ名]_extend_number分割で必要に応じて拡張を行います。
以下に例を示します。

```
----------------------------------------------------------------------
max_IIOP_resp_con = 100
limit_of_max_IIOP_resp_con = 140
max_IIOP_resp_con_extend_number = 2
----------------------------------------------------------------------
```

上記のパラメタの場合、max_IIOP_resp_conは初期値を100として、120、140と最大2回の拡張を行います。
なお、isconfig.xmlファイルの定義項目AutoConfigurationModeにMANUALを指定した場合、自動拡張に関するパラメタは無視され拡張は行いません。isconfig.xmlについての詳細に関しては“運用ガイド(基本編)”を参照してください。

注7)
一度の拡張処理で増加できるサイズは初期値のサイズに制限されます。

一度の拡張サイズが初期値のサイズを超える拡張を行う設定がされた場合、拡張数は0が設定された場合と同様の値に補正されます。
また、自動拡張の最大値と初期値との差分よりも拡張数が大きい場合は、拡張数は自動拡張の最大値と初期値との差分の値に補正されます。

```
----------------------------------------------------------------------
max_IIOP_resp_con = 100
limit_of_max_IIOP_resp_con = 300
max_IIOP_resp_con_extend_number = 1
----------------------------------------------------------------------
```

上記のパラメタの場合、max_IIOP_resp_con_extend_numberは2に補正されます。

注8)
以下の場合は初期値が異なります。

Windows

Interstage Application Server Enterprise Edition/Standard-J Editionの場合。
Interstage Web Serverでは初期値に変更はありません。

Solaris

Interstage Application Server Enterprise Edition/Standard-J Editionで標準インストール・カスタムインストール・GUIインストーラを使用したインストールを行った場合(pkgaddコマンドによるインストールを行わない場合)。
Interstage Application Server Enterprise Editionの拡張システムおよびInterstage Web Serverでは初期値に変更はありません。

Linux

Interstage Application Server Enterprise Edition/Standard-J Editionで標準インストール・カスタムインストール・GUIインストーラを使用したインストールを行った場合(rpmコマンドによるインストールを行わない場合)。
Interstage Web Serverでは初期値に変更はありません。

初期値は以下のように変更されています。

パラメタ名	初期値
max_IIOP_resp_con	512
max_IIOP_resp_requests	2048
max_processes	512
max_exec_instance	16384

注9)
C++のCORBAアプリケーションがCORBA::ORB::bind_object関数を発行して登録するオブジェクト数と別JavaVMから呼び出されるEJBアプリケーションについてSession BeanとEntity BeanのEJB objectのインスタンス数の加算値よりも大きな値を設定してください。

注10)

Solaris

number_of_common_bufferとlimit_of_number_of_common_bufferの設定可能な値はSolaris 9ではシステムパラメタのsemvmx値、Solaris 10では65535が最大値になります。

Linux

number_of_common_bufferとlimit_of_number_of_common_bufferの設定可能な値はSEMVMXのOS実装値(32767)が最大値になります。

なお、number_of_common_bufferとlimit_of_number_of_common_bufferの設定を省略した場合は、max_IIOP_resp_requestsとlimit_of_max_IIOP_resp_requestsの値から求まる各パラメタの値が設定可能な最大値を超過しないか確認をお願いします。

ポイント: max_IIOP_local_init_con、max_IIOP_resp_conについて

CORBAサービスは、サーバアプリケーションが動作しているマシンごとに1つのコネクションを使用します。
max_IIOP_local_init_conは、各アプリケーションが使用するサーバホストへのコネクション数の最大値を指定します。
max_IIOP_resp_conは、各ホストで使用するアプリケーション間のコネクション数を指定します。

原則として、アプリケーション間のコネクションはクライアントアプリケーションのプロセス単位に生成されます。例えば、クライアントアプリケーションから1つのサーバアプリケーションに複数のリクエストが同時に発行されても、コネクション数は1になります。
SSL連携機能を使用する場合、SSL接続のコネクションとSSL接続でないコネクションは別コネクションとして数える必要があります。例えば、クライアントアプリケーションから、1つのサーバマシンにSSL接続のコネクションとSSL接続でないコネクションを使用した場合、コネクション数は2になります。

なお、以下の場合にはコネクションを使用するので必要に応じて加算する必要があります。

- CORBAサービスのコマンド実行時は1コネクション使用します。コマンドを同時に複数起動する場合は、その数を加算して指定しておいてください。
- インタフェースリポジトリ動作時は、1コネクションを使用します。
- インタフェースリポジトリ、ネーミングサービスなどの各サービスはサーバアプリケーションです。そのため、ネーミングサービスへ参照、登録などのリクエストを発行すると1コネクションを使用します。他ホスト上のインタフェースリポジトリ、ネーミングサービスを参照する設定の場合は、参照先ホストのIIOP_resp_conが1消費されます。
  例えばTYPE3 EJBで初期化したホスト上でネーミングサービスへアクセスするアプリケーションを起動すると、ネーミングサービスが起動しているホスト上のIIOP_resp_con資源が1消費されることになります。

以下に、各パラメタのコネクション数のカウント方法を示します。

max_IIOP_local_init_con:

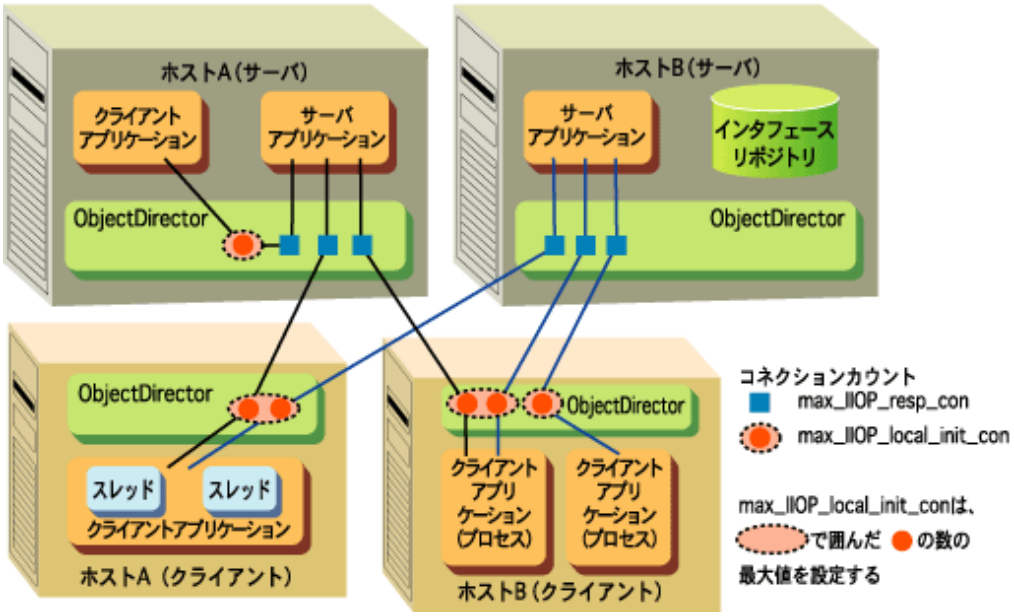
クライアントアプリケーションが動作しているホスト上で、クライアントアプリケーション(プロセス単位)からサーバアプリケーション(ホスト単位)へのコネクション数の最大値を指定します。

- **設定値の目安(インタフェースリポジトリ動作時):**
  max_IIOP_local_init_con ＝
  [1つのクライアントアプリケーションが接続するサーバホスト数の最大値]と
  256のうちの最大値
- **設定値の目安(インタフェースリポジトリ動作時、SSL連携機能を使用):**
  max_IIOP_local_init_con ＝
  [1つのクライアントアプリケーションが接続するサーバホスト数の最大値×2]と
  256のうちの最大値

max_IIOP_resp_con:

サーバアプリケーションが動作しているホスト上で、接続するクライアントアプリケーションのプロセス数の合計を指定します。同一ホスト上で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が接続する場合も、そのコネクション数を加算する必要があります。

- **設定値の目安(インタフェースリポジトリ動作時):**
  max_IIOP_resp_con ＝
  接続するクライアントアプリケーションのプロセス数 ＋ 2
- **設定値の目安(インタフェースリポジトリ動作時、SSL連携機能を使用):**
  max_IIOP_resp_con ＝
  (接続するクライアントアプリケーションのプロセス数 × 2) ＋ 2



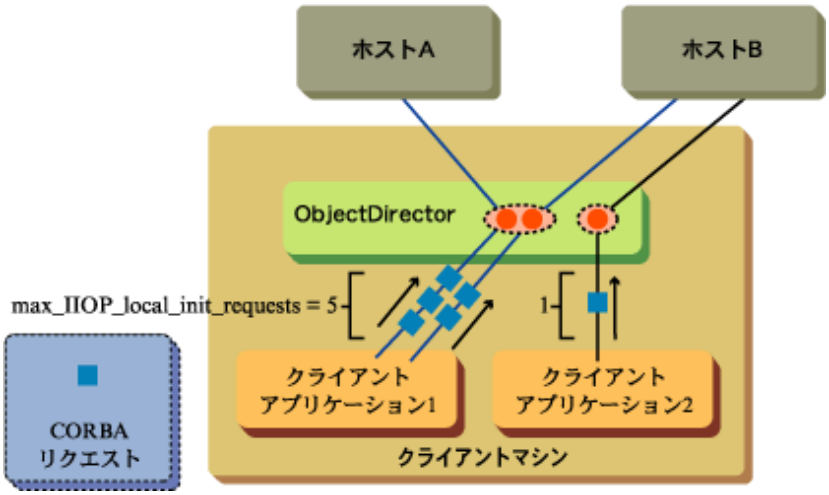
ポイント: max_IIOP_local_init_requests、max_IIOP_resp_requestsについて

CORBAサービスでは、クライアントアプリケーションが同時に送信するリクエスト数に応じてmax_IIOP_local_init_requestsを設定する必要があります。また、サーバアプリケーションが同時に受信するリクエスト数に応じてmax_IIOP_resp_requestsを設定する必要があります。

max_IIOP_local_init_requests:

クライアントアプリケーションが同時に送信できるリクエスト数の最大値を指定します。下の図ではクライアントアプリケーション1が5個のリクエストを同時に送信し、アプリケーション2が1個のリクエストを同時に送信しています。このため、max_IIOP_local_init_requestsは5以上の値を設定する必要があります。

ただし、算出された値が4096以下の場合は初期値の4096のままで問題ありません。この例では4096に満たないのでmax_IIOP_local_init_requestsはデフォルトの4096から変更の必要はありません。

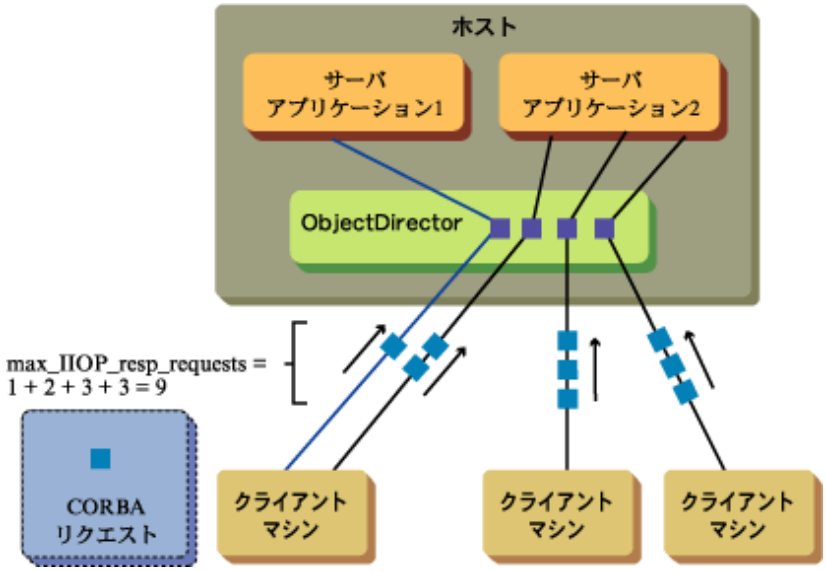


max_IIOP_resp_requests:

CORBAサーバアプリケーションが同時に受信できるリクエスト数の最大値を指定します。

それぞれのクライアントマシンから発行されたリクエストがサーバマシンに到達し、CORBAサーバアプリケーションで同時に処理される数になるので、個々のクライアントマシンから同時に発行されるリクエストの合計値を見積もる必要があります。

下の図ではそれぞれのクライアントマシンから発行されたリクエストが同時に9個サーバマシンに到達しているので、max_IIOP_resp_requestsには9以上を設定する必要があります。



◆タイムアウト監視に関する動作環境

パラメタ名	初期値 省略値 指定範囲	意味	備考
period_client_idle_con_timeout	96 (480秒) 96 (480秒) 0～20000000	クライアントにおける、無通信状態(サーバへのリクエスト送信なし)の監視時間(リクエスト返信完了後のコネクション維持時間)。 この時間を超えてもサーバへのリクエスト送信がない場合、サーバとのコネクションの切断を行います。(注3) この値に5を乗じた値が実際の時間(秒)となります。0を設定すると無通信監視を行いません。	
period_idle_con_timeout	120 (600秒) 1 (5秒) 0～20000000	サーバにおける、無通信状態(クライアントからのリクエスト送信なし)の監視時間(リクエスト返信完了後のコネクション維持時間)。 この時間を超えてもクライアントからのリクエスト送信がない場合、クライアントとのコネクションを切断し、リクエスト処理に使用したメモリ資源を解放します。 この値に5を乗じた値が実際の時間(秒)となります。0を設定すると無通信監視を行いません。	サーバ機能のみ有効。 (注1) (注4)
period_receive_timeout	72 (360秒) 72 (360秒) 0～20000000	クライアントにおける、リクエスト送信から返信までの待機時間。 この時間を超えてもサーバからの返信がない場合、クライアントにタイムアウトが通知されます。 この値に5を乗じた値が実際の時間(秒)となります。	
period_server_timeout	120 (600秒) (注2) 120 (600秒) 1～20000000	Persistentタイプ以外のサーバアプリケーションとその他のアプリケーションで意味が異なります。 Persistentタイプ以外のサーバアプリケーションにおいてはアプリケーション起動からCORBA_ORB_initメソッド完了までの監視時間となります。この時間以内にCORBA_ORB_initメソッドが完了しないとクライアントにシステム例外(NO_IMPLEMENT)が通知されます。 クライアントアプリケーションとPersistentタイプのサーバアプリケーションにおいてはCORBA_ORB_initメソッド発行からCORBA_ORB_initメソッド完了までの監視時間となります。	サーバ機能のみ有効。 (注1)

パラメタ名	初期値 省略値 指定範囲	意味	備考
		この値に5を乗じた値が実際の時間(秒)となります(0は指定できません)。	

注1)
Interstage Web Serverでは初期値から変更しないでください。

注2)
初期値より減少させた場合は、インタフェースリポジトリの起動に失敗することがあります。

注3)
次回リクエスト送信時にサーバとのコネクションの再接続を行います。
なお、クライアントアプリケーションがプロセスモードの場合は、時間超過のタイミングではコネクション切断は行わず、次回リクエスト送信時にサーバとのコネクションの切断・再接続を行います。

注4)
サーバ側の無通信監視時間超過によるコネクション切断のタイミングでクライアントがリクエストを発行すると、クライアントに通信異常が通知される場合があります。この問題を回避するためには、クライアント側のperiod_client_idle_con_timeoutにサーバ側のperiod_idle_con_timeoutよりも小さな値を設定してください。


ポイント
タイムアウト時間は、連携するアプリケーションに適用されるタイムアウト時間を考慮して設定する必要があります。詳細は“OLTPサーバ運用ガイド”(Interstage Application Server Enterprise Editionで提供)の“CORBAアプリケーションのタイマ監視”を参照してください。

◆セキュリティ機能に関する動作環境(アプリケーション間通信)

パラメタ名	初期値 省略値 指定範囲	意味	備考
http_proxy	proxy_host	HTTPプロキシサーバのホスト名。	(注1) (注2)
	null (値は設定されません)		
	―		
http_proxy_port	8080	HTTPプロキシサーバのポート番号。	(注1) (注2)
	0		
	―		
http_proxy_use	no	HTTPプロキシサーバの使用を指定。 ・ yes:使用する ・ no :使用しない	(注1) (注2)
	no		
	yes, no		
UNO_IIOP_ssl_use	no	SSL連携の有効/無効を選択。 ・ yes:有効 ・ no :無効	(注3)
	no		
	yes, no		
UNO_IIOP_ssl_port	4433	SSL連携で使用するポート番号。 UNO_IIOP_ssl_useが“yes”の場合に有効です。	
	4433		
	―		

注1)
Interstage Web Serverでは初期値から変更しないでください。

注2)
プレインストール型ランタイム(Portable-ORB以外の実行環境)でHTTPプロキシサーバを経由してHTTPトンネリングを使用する場合に指定します。http_proxy、http_proxy_portは、“http_proxy_use=yes”のときに有効であり、Webブラウザで使用しているHTTPプロキシサーバのホスト名とポート番号を指定します。

注3)
SSL接続のコネクションとSSL接続でないコネクションは別コネクションとして数える必要があります。max_IIOP_resp_con、max_IIOP_local_init_conパラメタを見積もる際には注意してください。

◆セキュリティ機能に関する動作環境(CORBAサービス資源) Solaris Linux

パラメタ名	初期値 省略値 指定範囲	意味	備考
iss_use	no no yes, no	CORBAサービス資源のセキュリティ強化機能の有効/無効を指定。“yes”を指定すると、CORBAアプリケーションはiss_groupのグループに属するユーザ(またはroot)のみが起動可能となります。	(注1) (注2)
iss_group	root(0) root(0) —	CORBAサービス資源のセキュリティ強化機能有効時（iss_use=yes指定）のアプリケーション動作のグループIDを指定します。	(注1) (注2) (注3) (注4)

注1)
インストール時のセキュリティ設定として“強化セキュリティモード”を選択した場合、初期値は以下のように変更されています。

パラメタ名	初期値
iss_use	yes
iss_group	インストール時に指定した“Interstage運用グループ名”

注2)
CORBAサービス資源のセキュリティ強化機能に関する設定を変更する場合、issetsecuritymodeコマンドの使用を推奨します。詳細は“セキュリティシステム運用ガイド”の“共通の対策”および“リファレンスマニュアル(コマンド編)”の“issetsecuritymode”を参照してください。

注3)
すでにシステムに登録されているグループを指定してください。
なお、指定したグループのエントリ情報の取得処理が行われます。また、指定した値に関わらず、sysとotherグループのエントリ情報の取得処理も行われます。このとき、OSの設定(nsswitch.conf)によってはLDAP等と通信する場合もあります。詳細はOSのマニュアルを参照してください。

注4)
CORBAアプリケーションの実行は、iss_groupに指定したグループに属するユーザまたはrootに限定され、他の一般ユーザは実行できなくなりますので、アプリケーションの実行ユーザに注意してください(“リファレンスマニュアル(コマンド編)”の“OD_impl_inst”を参照)。

◆コード変換機能に関する動作環境

パラメタ名	初期値 省略値 指定範囲	意味	備考
undefined_char_conversion	single single single, multi, fail	未定義文字をコード変換した場合の動作を設定します。 ・ single: デフォルト すべての未定義文字は半角のアンダースコアに変換します。(注1) ・ multi 半角の未定義文字は半角のアンダースコアに変換し、全角の未定義文字は全角のアンダースコアに変換します。 ・ fail 未定義文字の変換を行うとアプリケーションにシステム例外DATA_CONVERSIONを通知します。	(注2)

注1)
文字コードがUNICODEまたはUTF8の場合は、“multi”を指定した場合と同様に全角の未定義文字は全角のアンダースコアに変換します。

注2)
クライアントおよびサーバの両方で設定してください。コード変換機能の詳細については“OLTPサーバ運用ガイド”の“コード変換”を参照してください。

◆保守機能に関する動作環境

パラメタ名	初期値 省略値 指定範囲	意味	備考
access_log_policy	start (初期値を推奨) start start, standby	CORBAサービス起動時のアクセスログの採取/非採取の状態。 ・ start: 起動時からログ採取する ・ standby: ログ採取しない	サーバ機能のみ有効 (注1)
access_log_size	3000000 3000000 1～2147483647 (longの最大値)	アクセスログファイルの最大サイズ。(バイト単位)	サーバ機能のみ有効 (注1)
access_log_level	send_stex: recv_stex: send_userex: recv_userex: close_resp_info send_stex: recv_stex: send_userex: recv_userex: close_resp_info	アクセスログ採取レベルのキーワードを連結して指定。 (区切り文字はコロン(“:”)、空白は指定不可) “all”を指定すると、すべての採取レベルを指定したものとみなされます。	サーバ機能のみ有効 (注1) (注2)

パラメタ名	初期値 省略値 指定範囲	意味	備考
	―		
error_log_policy	start (初期値を推奨) start start, standby	CORBAサービス起動時のエラーログの採取/非採取の状態。 ・ start: 起動時からログ採取する ・ standby: ログ採取しない	(注1)
error_log_size	3000000 3000000 1～2147483647 (longの最大値)	エラーログファイルの最大サイズ。(バイト単位)	(注1)
info_log_policy	start (初期値を推奨) start start, standby	CORBAサービス起動時のインフォメーションログの採取/非採取の状態。 ・ start: 起動時からログ採取する ・ standby: ログ採取しない	(注1)
info_log_size	3000000 3000000 1～2147483647 (longの最大値)	インフォメーションログファイルの最大サイズ。(バイト単位)	(注1)
logging	no no yes, no	内部ログの採取を指定。 ・ yes: 採取する ・ no : 採取しない	(注3)
log_file_size	10000000 -1 4096～2147483647 (longの最大値)	内部ログのファイルサイズの上限値。(バイト単位) “logging = yes”とする場合、本パラメタは省略しないでください。	(注3)
process_log_policy	start (初期値を推奨) start start, standby	CORBAサービス起動時のプロセスログの採取/非採取の状態。 ・ start: 起動時からログ採取する ・ standby: ログ採取しない	(注1)
process_log_size	3000000 3000000 1～2147483647	プロセスログファイルの最大サイズ。(バイト単位)	(注1)

パラメタ名	初期値 省略値 指定範囲	意味	備考
	(longの最大値)		
log_file_path	― ― ―	ログファイルの出力先を絶対パスで指定します。本パラメタで指定したパスには、以下のログファイルが出力されます。 ・ アクセスログ ・ エラーログ ・ インフォメーションログ ・ プロセスログ ・ 内部ログ 本パラメタで指定したパスが存在しなかった場合、CORBAサービスの起動に失敗します。 128バイトより長いパスは指定できません。128バイトより長いパスを指定した場合、無効となります。 また、空白および“=”を含むパスは指定できません。空白または“=”を含むパスを指定した場合、その直前までが有効となります。 Windows “¥”と“/”は区別されず、共にフォルダの区切りとして使用されます。	(注1) (注3) (注4)
snap_size	40000 40000 1024～2147483647 (longの最大値) (注5)	スナップショットサイズの上限値。(バイト単位)	サーバ機能のみ有効
snap_use	yes yes yes, no	スナップショットの採取を指定。 ・ yes: 採取する ・ no : 採取しない	サーバ機能のみ有効
trace_file_synch_level	stop stop ―	トレースファイルへの出力タイミングを指定。複数指定可能(セパレータは“&”)。 ・ none:odformtraceコマンド実行時のみ出力。 ・ exit:アプリケーション正常終了時、終了したアプリケーションのトレース情報を出力。 ・ vanish:アプリケーション異常終了時、終了したアプリケーションのトレース情報を出力。 ・ stop:CORBAサービス終了時、すべてのアプリケーションのトレース情報を出力。 ・ loop:メモリ上に採取されたトレース情報のサイズがtrace_size_per_processを超えた場合に出力。	サーバ機能のみ有効
trace_size_per_process	10000 10000	プロセスごとのトレース情報サイズの最大値(バイト単位)。	サーバ機能のみ有効

パラメタ名	初期値 省略値 指定範囲	意味	備考
	1024～ 100000000 (注5)		
trace_size_of_daemon	0 0 0, 1024～ 100000000	CORBAサービスのデーモンプロセスに対するトレース情報サイズの最大値(バイト単位)。 0を指定すると、“trace_size_per_process × 32”が設定されます。 なお、計算結果が100000000を超えた場合は100000000が設定されます。 trace_size_per_processよりも小さな値を設定した場合は、trace_size_per_processの値に補正されます。	
trace_use	yes (初期値を推奨) yes yes, no	トレース情報の採取を指定 ・ yes: 採取する ・ no : 採取しない	サーバ機能のみ有効

注1)
アクセスログ・プロセスログ・エラーログ・インフォメーションログは、“log_file_path”で指定したパスに採取されます。“log_file_path”を指定しなかった場合は、以下のパスに採取されます。
また、ディスク領域として、以下のログファイルサイズの合計分が必要となります。

ログファイル格納パス

Windows

C:¥INTERSTAGE¥ODWIN¥var 配下 (インストールパスはデフォルト)

Solaris

/var/opt/FSUNod 配下 (インストールパスはデフォルト)

Linux

/var/opt/FJSVod 配下

ログファイル名とファイルサイズ

ログ名	ログファイル名	ログファイルサイズ
アクセスログ	accesslog accesslog.old	access_log_size×2
プロセスログ (サーバ用ライブラリ(ODSV.DLL, libOM.so)をリンクしている場合)	proclog proclog.old	process_log_size×2
プロセスログ (クライアント用ライブラリ(ODWIN.DLL)をリンクしている場合)	proclogcl proclogcl.old	process_log_size×2
エラーログ (サーバ用ライブラリ(ODSV.DLL, libOM.so)をリンクしている場合)	errlog errlog.old	error_log_size×2

ログ名	ログファイル名	ログファイルサイズ
エラーログ (クライアント用ライブラリ(ODWIN.DLL)をリンクしている場合)	errlogcl errlogcl.old	error_log_size×2
インフォメーションログ (サーバ用ライブラリ(ODSV.DLL, libOM.so)をリンクしている場合)	infolog infolog.old	info_log_size×2
インフォメーションログ (クライアント用ライブラリ(ODWIN.DLL)をリンクしている場合)	infologcl infologcl.old	info_log_size×2

以下のログファイルを採取するためには、ログファイル格納パスに、管理者権限グループに対する書き込みのアクセス許可が必要です。

- アクセスログ
- プロセスログ(サーバ用ライブラリをリンクしている場合)
- エラーログ(サーバ用ライブラリをリンクしている場合)
- インフォメーションログ(サーバ用ライブラリをリンクしている場合)

また、以下のログファイルを採取するためには、ログファイル格納パスに、クライアントアプリケーションを実行するユーザが所属しているグループに対する書き込みのアクセス許可が必要です。

- プロセスログ(クライアント用ライブラリをリンクしている場合)
- エラーログ(クライアント用ライブラリをリンクしている場合)
- インフォメーションログ(クライアント用ライブラリをリンクしている場合)

上記のアクセス許可がない場合はログファイルの出力に失敗しますが、この際、特にエラーメッセージなどが表示されない場合があります。このため、ログファイルを採取する際には、運用前にログファイル格納パスのアクセス許可が正しく設定されているか確認してください。

注2)

access_log_level(アクセスログ採取レベル)に指定可能なキーワードは、“トラブルシューティング集”の“障害調査資料の採取”－“CORBAサービスのログ情報の採取”を参照してください。

注3)

“logging=yes”を指定した場合、内部ログファイルへの出力処理に時間を要するため、CORBAサービスの性能が劣化し、CORBAアプリケーションのレスポンス性能が低下します。また、インタフェースリポジトリやネーミングサービスの起動に時間がかかります。

なお、インタフェースリポジトリおよびネーミングサービスの起動に1分以上かかった場合、Interstageの起動に失敗しますので注意してください。“logging=yes”を指定してInterstageの起動に失敗した場合は、“トラブルシューティング集”の“Interstageの起動/停止時の異常”を参照して対処を実施してください。

“logging=yes”を指定した場合、内部ログは“log_file_path”で指定したパスに採取されます。“log_file_path”を指定しなかった場合は、以下のパスに出力されます。ファイル名は“log_file_path”の指定にかかわらず共通です。

Windows

パス: C:¥INTERSTAGE¥ODWIN¥var
ファイル名:
・log (log.old)
・アプリケーションごとのappNNNN.log(appNNNN.old) (NNNNは英数字)

Solaris

パス: /var/opt/FSUNod
ファイル名: /log (log.old)

Linux

パス: /var/opt/FJSVod
ファイル名: /log (log.old)

プレインストール型Javaライブラリ使用時は、上記に加えて、以下のファイルに出力されます。“log_file_path”の値は影響しません。

ー ユーザ作業ディレクトリ(Java VMのシステムプロパティのuser.dirの指す位置)配下
JVxxxxxxxxxx.log (xxxxxxxxxxは数字)
アプレット運用の場合、user.dirはJava VMの起動オプションで変更可能です。

注4) Solaris

マルチシステムを使用する場合、デフォルトシステムとマルチシステムで同じ値を指定しないでください。ログファイルが正常に出力されない場合があります。

注5) Linux

Interstage Web Serverでは、初期値より大きい値は指定しないでください。

◆旧バージョンの互換に関する動作環境

パラメタ名	初期値 省略値 指定範囲	意味	備考
msg_output_compatible	no no od10924, od10926, od10941, od11101, no	od10924、od10926、od10941、od11101の各メッセージについて、システムログに出力するメッセージIDを“&”で連結して指定します。 ・“od10924”:od10924メッセージを出力します。 ・“od10926”:od10926メッセージを出力します。 ・“od10941”:od10941メッセージを出力します。 ・“od11101”:od11101メッセージを出力します。 ・“od10924&od10926&od10941&od11101”: 4つのメッセージすべてを出力します。 ・“no”:システムログへの出力は抑止されます。	

A.2 gwconfig

■概要

gwconfigファイルは、HTTPトンネリング使用時に、Webサーバで起動されるHTTP-IIOPゲートウェイの動作環境を定義するファイルです。
CORBAサービスのタイムアウト監視に関連する項目を修正した場合に、同様の項目について定義を修正する必要があります。
なお、初期値から変更しない場合、gwconfigファイルは必要ありません。

■ファイル名

Windows

C:¥INTERSTAGE¥ODWIN¥etc¥gwconfig (インストールパスはデフォルト)

Solaris

/etc/opt/FSUNod/gwconfig (インストールパスはデフォルト)

Linux

/etc/opt/FJSVod/gwconfig

■ファイル内情報

gwconfigファイルは、以下の形式で値を設定します。

◆形式:

パラメタ名 = 設定値

半角のシャープ(#)を行の先頭に指定した場合は、その行はコメントとして扱われます。また、空行は解析時に無視されます。
コメント

◆記述例:

timeout_response=60

◆パラメタ:

設定値を変更することのできるパラメタを下表に示します。

パラメタ名	初期値 指定範囲	意味
timeout_response	360 0～100000000	リクエストの返信待ち時間 HTTP-IIOPゲートウェイにおける、リクエスト送信から返信までの待機時間(秒)。この時間内にサーバメソッドから返信されないと、クライアントにタイムアウトが通知されます。クライアント側で、CORBAサービスのperiod_receive_timeout(クライアント側のリクエスト送信から返信までの待機時間。configファイルで定義)を変更した場合、このパラメタは、period_receive_timeout値以下になるよう変更する必要があります。0を指定するとタイムアウトを行いません。
timeout_session	180 0～2147483647	セション保持時間(クライアント間無通信監視時間) HTTP-IIOPゲートウェイにおける、クライアントの管理単位の保持時間(秒)。 サーバからの返信待ちがない状態で、この時間内にクライアントから新たなリクエスト送信がない場合には、クライアントの管理情報は破棄されます。0を指定するとタイムアウトを行いません。
timeout_connection	60 0～2147483647	コネクション保持時間(サーバ間無通信監視時間) HTTP-IIOPゲートウェイにおける、無通信状態の監視時間。 サーバからの返信待ちがない状態で、この時間内にクライアントから新たなリクエスト送信がない場合には、サーバとのコネクションを切断します。0を指定するとタイムアウトを行いません。
logmode	5 1～5	HTTP-IIOPゲートウェイの内部ログ採取の有無 (注) 以下から指定します。 ・ 5:内部ログ情報を採取しません。 ・ 3:リクエストデータおよびエラー発生時の情報を採取します。 ・ 2:“3”で採取する情報に加えて、リプライデータおよび内部処理の情報を採取します。 ・ 1:“2”で採取する情報に加えて、トレース情報を採取します。
max_log_file_size	1048576 1048576～10485760	ログファイルサイズ(バイト単位)

注)

- HTTP-IIOPゲートウェイの内部ログは、以下に出力されます。
  内部ログの出力先は環境変数OD_HTTPGW_HOMEまたはOD_HOMEより変更可能です。

  Windows

  C:¥INTERSTAGE¥ODWIN¥var¥httpgw*_0.log(httpgw*_1.log) (*は英数字)

  Solaris

  /opt/FSUNod/var/httpgw*_0.log(httpgw*_1.log) (*は英数字)

  Linux

  /opt/FJSVod/var/httpgw*_0.log(httpgw*_1.log) (*は英数字)

- 内部ログ採取を終了するためには、gwconfigにパラメタを設定した後、Webサーバを再起動する必要があります。

- Solaris Linux
  HTTP-IIOPゲートウェイの内部ログが出力されるディレクトリにInterstage HTTP Serverのサーバプロセスを実行するユーザ名/グループ名の書き込み権を設定してください。書き込み権がない場合、システムログにメッセージod40102が出力されます。

- 定義情報を変更した場合、次回のWebサーバ起動時より有効となります。
- gwconfigファイルの格納位置
  gwconfigファイルの格納場所は、環境変数OD_HTTPGW_HOMEまたはOD_HOMEで指定することが可能です。両方指定されている場合にはOD_HTTPGW_HOMEが優先され、指定されたディレクトリ配下のetcディレクトリに格納します。また、同ディレクトリ配下にvarディレクトリを作成してください。
- 最終行には、必ず改行を入れてください。
- 設定値に数字以外が含まれた場合、数字までを有効とします。
- SolarisまたはLinuxの場合、パラメタtimeout_sessionおよびtimeout_connectionは無視されます。

A.3 inithost/initial_hosts

■概要

inithost/initial_hostsファイルは、ネーミングサービス、インタフェースリポジトリのホスト情報を定義するファイルです(ネーミングサービス、インタフェースリポジトリは、CORBAアプリケーション連携を行う場合に必要となる、アプリケーションの所在、インタフェース情報が登録されているサービスです)。

inithost/initial_hostsには、サービスが存在するホスト名(またはIPアドレス)とCORBAサービスのポート番号(デフォルト8002)を指定します。ホスト名・ポート番号は最大16組まで指定できます。
サービスの問い合わせは定義順に行われ、参照するサービスが存在しない場合は次行に定義されているホストに問い合わせを行います。
なお、ネーミングサービス・インタフェースリポジトリをローカルホスト上で動作させる場合は、ホスト名・ポート番号の設定は必要ありません。

■ファイル名

Windows

C:¥INTERSTAGE¥ODWIN¥etc¥inithost (インストールパスはデフォルト)

Solaris

Solarisサーバ:
/etc/opt/FSUNod/initial_hosts (インストールパスはデフォルト)
Windows(R)クライアント:
C:¥INTERSTAGE¥ODWIN¥etc¥inithost (インストールパスはデフォルト)

Linux

Linuxサーバ:
/etc/opt/FJSVod/initial_hosts (インストールパスはデフォルト)
Windows(R)クライアント:
C:¥INTERSTAGE¥ODWIN¥etc¥inithost (インストールパスはデフォルト)

■ファイル内情報

inithost/initial_hostsファイルは、以下の形式で値を設定します。

◆形式:

ホスト名 ポート番号

半角のシャープ(#)を行の先頭に指定した場合は、その行はコメントとして扱われます。また、空行は解析時に無視されます。
コメント

◆記述例:

```
# comment
 hostname 8002
```

◆パラメタ:

設定値を変更することのできるパラメタを下表に示します。

パラメタ名	初期値	意味
ホスト名	なし	ネーミングサービス、またはインタフェースリポジトリサービスが動作しているホスト名(またはIPアドレス)を指定します。ホスト名の長さとして、64バイトまで指定できます。(注1)(注2)
ポート番号	なし	上記サービスが動作しているホストで定義されているCORBAサービスのポート番号を指定します。

注1)
サーバ側のイニシャルサービスやネーミングサービスに登録されているオブジェクトリファレンスに設定されているホスト名に対して名前解決(IPアドレスへの変換)が可能である必要があります。登録されているオブジェクトリファレンスの情報を参照する場合は、サーバ側でOD_or_admコマンド(-lオプション)及びodlistnsコマンド(-lオプション)を実行してください。
なお、自ホスト側/サーバ側(サービスが動作しているホスト)のホスト名定義と統一させて指定する必要があります。

[自ホスト側] Windows

Windows(R)システムフォルダ¥system32¥drivers¥etc配下のlmhostsまたはhosts

[自ホスト側] Solaris Linux

/etc/hostsまたはNIS+など

[サーバ側]
サーバ側のホスト名定義

注2)
ホスト名に自ホストのホスト名やIPアドレスは指定しないでください。自ホスト名を指定した場合、サービスの問い合わせがループする場合があります。

- 定義情報の変更について
  定義情報を変更した場合、次回のCORBAサービス起動時より有効となります。

- isinit、ismodifyserviceコマンドの実行および設定について
  - isinitおよびismodifyserviceコマンドを実行する場合は、inithost/initial_hostsファイルに指定されているインタフェースリポジトリサービスおよびネーミングサービスのホスト名を、コメントまたは削除してください。
  - inithost/initial_hostsファイルの設定は、isinitおよびismodifyserviceコマンドの実行後に可能になります。
  - inithost/initial_hostsファイルにホスト名が設定されている場合でも、isinitおよびismodifyserviceコマンドで設定したホスト名が優先されます。
    また、isinitおよびismodifyserviceコマンドで設定したホスト名は、inithost/initial_hostsファイルに設定する必要はありません。
  - isinitおよびismodifyserviceコマンドを実行して環境設定を行った場合、inithost/initial_hostsファイルを使用してネーミングサービスのリモートホスト運用を行うことはできません。それは、isinitおよびismodifyserviceコマンドを使用すると、ネーミングサービスのリモートホスト名がイニシャルサービスに設定されるため、inithost/initial_hostsファイルが使用されないためです。
  - inithost/initial_hostsファイルを使用してネーミングサービスのリモートホスト運用が行えるのは、isinitおよびismodifyserviceコマンドが含まれないCORBAサービスクライアントでの運用に限られます。

- 不要なホスト情報定義について
  Windows(R)クライアントのinithostに、存在しない、または通信不可状態のホストが定義された場合、クライアントアプリケーションの動作が遅くなることがあります。不要なホスト名は削除してください。

ホスト名・ポート番号の設定は、odsethostコマンドでも行うことができます。

A.4 nsconfig

■概要

nsconfigファイルは、ネーミングサービスの動作環境を設定するファイルです。

■ファイル名

Windows

C:¥INTERSTAGE¥ODWIN¥etc¥nsconfig (インストールパスはデフォルト)

Solaris

/etc/opt/FSUNod/nsconfig (インストールパスはデフォルト)

Linux

/etc/opt/FJSVod/nsconfig

■ファイル内情報

nsconfigファイルは、以下の形式で値を設定します。

◆形式:

パラメタ名 = 設定値

パラメタ名と設定値の間の文字列は、“ = ”(半角スペース＋半角イコール＋半角スペース)を設定します。
半角のシャープ(#)を行の先頭に指定した場合は、その行はコメントとして扱われます。また、空行は解析時に無視されます。
コメント

◆記述例:

```
file_sync = yes
trace_level = update
bl_how_many = 65536
ogl_how_many = 256
ext_intf = yes
cn_userexception_log_use = yes
cn_userexception_log_size = 2000000
```

◆パラメタ:

設定値を変更することのできるパラメタを下表に示します。なお、指定が必須となるパラメタはありません。

パラメタ名	初期値 指定範囲	意味
file_sync	yes yes, no	ネーミングサービスのデータベースへの更新処理でファイルの同期書き込みを行うかを設定します。 ・ yes: ファイルの同期書き込みを行う。 ・ no : ファイルの同期書き込みを行わない。 初期創成などの大量データの更新時に、このパラメタをnoにすることにより処理を高速化することができます。ネーミングサービスの運用中には、信頼性を向上させるため、本パラメタはyesにしてください。
trace_level	update update, all	メソッド実行の自動トレースを採取するトレースレベルを指定します。 ・ update: 更新ログだけを採取する。 ・ all: すべてのトレースを採取する。
bl_how_many	65536 0～65536	NamingContext::list, BindingIterator::next_nで返されるバインディング数の最大値を指定します。
ext_intf	yes yes, no	ネーミングサービスの拡張機能を使用するかを指定します。 ・ yes: ネーミングサービスの拡張機能を使用する。 ・ no : ネーミングサービスの拡張機能を使用しない。 noを指定すると、ネーミングコンテキスト拡張インタフェース(NamingContextExtインタフェース)を使用できません。V2.X以前のクライアントアプリケーションを動作させる場合は、noを指定する必要があります。
cn_userexception_log_use	yes	ユーザ例外ログを採取するかを指定します。 ・ yes: ユーザ例外ログを採取する。

パラメタ名	初期値 指定範囲	意味
	yes, no	・ no : ユーザ例外ログを採取しない。
cn_userexception_log_size	2000000 1000～2000000	ユーザ例外ログのファイルサイズを指定します。

- 変更した値は、次回のネーミングサービス起動時より有効となります。
- パラメタext_intfに“no”を設定してV2.X以前のクライアントアプリケーションを動作させる場合は、運用に応じた対処が必要となる場合があります。V2.X以前のネーミングサービスを使用する場合の注意事項については、“アプリケーション作成ガイド(CORBAサービス編)”の“旧バージョンからの移行上の注意”－“ネーミングサービスに関する注意事項(V2以前のバージョンからの移行)”を参照してください。

A.5 irconfig

■概要

irconfigファイルは、インタフェースリポジトリのバックアップやログ情報などの動作環境を設定するファイルです。

■ファイル名

Windows

C:¥INTERSTAGE¥ODWIN¥etc¥irconfig (インストールパスはデフォルト)

Solaris

/etc/opt/FSUNod/irconfig (インストールパスはデフォルト)

Linux

/etc/opt/FJSVod/irconfig

■ファイル内情報

irconfigファイルは、以下の形式で値を設定します。

◆形式:

パラメタ名 = 設定値

半角のシャープ(#)を行の先頭に指定した場合は、その行はコメントとして扱われます。また、空行は解析時に無視されます。
コメント

◆記述例:

auto backup = no(yes)
auto backup path =
auto recovery = no(yes)

logging = no(yes)
logging memory size = 512
logfile path =
sync = no
select cache obj =

◆パラメタ:

設定値を変更することのできるパラメタを下表に示します。なお、指定が必須となるパラメタはありません。

パラメタ名	初期値 指定範囲	意味
auto backup	no yes, no	インタフェースリポジトリ起動時に自動的にバックアップを行うかを指定します。 ・ yes: 自動的にバックアップを行う。 ・ no : 自動的にバックアップを行わない。 注)バックアップは、インタフェースリポジトリ起動時に1回だけ行います。
auto backup path	― ―	バックアップデータの格納場所を指定します。auto backup=yesと指定した場合、必ずパスを指定する必要があります。パスを指定しないとバックアップは行われません。 注)格納場所には、作成したデータベースサイズ以上の空き領域が必要です。
auto recovery	no yes, no	トランザクション処理中のシステムダウンなどによりデータベースの異常を検出した場合に、バックアップデータを元に自動的にリカバリを行うかを指定します。 ・ yes: 自動的にリカバリを行う。 ・ no : 自動的にリカバリを行わない。 注)本機能を使用する場合は“auto backup=yes”と指定し、auto backup pathを指定する必要があります。
ir_timeout	1800(秒) 0～100000000(秒)	IDLコンパイル(IDLc)およびインタフェース情報移入(odimportir)において、インタフェースリポジトリへのリクエストの復帰までの待機時間を指定します。0を指定すると、リクエスト復帰までの待機時間は監視されません。
Solaris Linux iss_use	no yes, no	資源保護機能の有効/無効を指定します。 ・ yes: 資源保護機能を有効とする。 ・ no : 資源保護機能を無効とする。 “yes”を指定すると、インタフェースリポジトリはデータベース管理者(デフォルト:root)のみが運用可能となります。 注)インストール時のセキュリティ設定として“強化セキュリティモード”を選択した場合、初期値は“yes”になります。
logging	no yes, no	トラブル発生時にログ情報を採取するかを指定します。 ・ yes : ログ情報を採取する。 ・ no : ログ情報を採取しない。 採取したログは、irlogdumpコマンドによりファイルに出力できます。通常は、初期値(no)で運用します。
logging memory size	512(KB) 1～4096(KB)	ログ情報を格納する共用メモリのサイズを指定します。logging=noとした場合、この値は意味を持ちません。

パラメタ名	初期値 指定範囲	意味
logfile path	－ －	“logging=yes”とした場合、irlogdumpコマンドにより出力されるログ情報の格納ディレクトリをフルパスで指定します。パスを指定しない場合は、CORBAサービスの動作ディレクトリと同一のディレクトリに格納されます(“A.1 config”参照)。 “logging=no”と指定した場合、この値は意味を持ちません。
select cache obj	－ －	インタフェースリポジトリ起動時にキャッシュ対象とするオブジェクトを指定します。 キャッシュ対象オブジェクトは、テキストファイル内にキャッシュ対象オブジェクトのリポジトリIDを記述し、そのファイル名をフルパスで指定します。 ファイル名が指定されない場合は、全登録オブジェクトがキャッシュ対象となります。 ファイルの作成例および注意事項については、以下のキャッシュ対象オブジェクトの指定方法を参照してください。
sync	no yes, no	同期モードを指定します。 yes:同期モードを指定します。 no:同期モードを指定しません。 “no”を指定すると、データベースへの書き込みと同期しないモードで動作するため、インタフェースリポジトリ更新処理のスループットが向上します。ただし、更新中のシステムダウンなどで発生するデータベース破壊を認識できない場合があります。 “yes”を指定すると、トランザクション単位での書き込みを保証するため、同期モードで動作します。信頼性が要求されるシステムを構築する場合に設定します(データベース破壊の認識可)。 同期モードを指定した場合、指定しない場合に比べて更新処理のスループットは低下します。このとき、クライアントへのサーバメソッドの復帰時間が長くなるため、タイムアウトが発生する可能性があります。これを防ぐには、“A.1 config”のperiod_receive_timeout値をチューニングしてください。

◆キャッシュ対象オブジェクトの指定方法

インタフェースリポジトリ起動時、キャッシュ対象オブジェクトを限定することにより、インタフェースリポジトリに大量のオブジェクトが登録されている場合の起動性能を改善することができます。

ただし、キャッシュ対象オブジェクトを指定した場合は、キャッシュ対象としないオブジェクトに対する参照性能は低下するため、運用に関しては注意が必要です。

また、キャッシュ対象オブジェクトを指定してインタフェースリポジトリを起動した場合は、以後、インタフェースリポジトリに対する登録/更新(IDLc, tdc, odimportir)を行うことができません。運用時(インタフェースリポジトリの登録/更新を行わない)に使用してください。

キャッシュ対象オブジェクトは、ユーザがテキストファイル内にキャッシュ対象オブジェクトのリポジトリIDを記述します。

リポジトリIDは、Repositoryオブジェクト(ルートオブジェクト)に直接包含されるModuleDefオブジェクトまたはInterfaceDefオブジェクトのみ指定可能で、指定されたオブジェクトに包含されるオブジェクトすべてがキャッシュ対象となります。

キャッシュ対象オブジェクトが継承またはスコープ参照で他のモジュールと関連付けられている場合は、そのモジュールもキャッシュ対象として指定する必要があります。

インタフェースリポジトリで管理するオブジェクトの種類、およびインタフェースリポジトリオブジェクトの包含/継承関係については“アプリケーション作成ガイド(CORBAサービス編)”(Interstage Application Server Enterprise Editionで提供)の“インタフェースリポジトリサービスのプログラミング”を参照してください。

なお、インタフェースリポジトリに登録されているオブジェクトと包含関係については、odlistirコマンドで表示できます。

キャッシュ対象オブジェクトを指定するためのファイルの記述例を以下に示します。

```
IDL:testmodule1:1.0
IDL:testmodule2:1.0
IDL:testmodule3:1.0
```

注) 1行に、キャッシュ対象オブジェクトのリポジトリIDを1つだけ記述できます。
コメントは、使用できません。

- 変更した値は、次回のインタフェースリポジトリサービス起動時より有効となります。
- ismodifyserviceで環境設定を行う際に“auto backup”が指定されていた場合は、データベースが空の状態でバックアップが行われます。この状態でバックアップされたデータベースは、使用できません。
- auto backup機能はインタフェースリポジトリ起動時の状態でバックアップが行われます。起動後に更新されたインタフェース定義情報はバックアップに反映されません。起動後に更新された情報が必要な場合は、Interstage(インタフェースリポジトリ)を再起動するか、またはodbackupsysコマンドを使用してバックアップを行ってください。
- odbackupsysコマンドでは、キャッシュ対象オブジェクトを指定したファイルをバックアップできません。ファイルのコピーコマンドなどでバックアップを行ってください。
- インタフェースリポジトリのデータベース管理者がroot（デフォルト）以外であり、かつirconfigファイルに“iss_use=yes”を設定すると、ログ情報採取機能の有効化（irconfigファイルの“logging=yes”）を指定しても、インタフェースリポジトリ（データベースアクセス機能）のログ情報が採取されなくなります（キャッシュサーバのログ情報は採取されます）。
  また、irconfigファイルに"iss_use=yes"を設定した場合は、irlogdumpコマンド（ログ情報の出力・制御）は管理者権限（root）で実行する必要があります。

付録B　コンポーネントトランザクションサービスの環境定義

本定義は以下の製品でだけサポートしています。

・ Interstage Application Server Enterprise Edition

コンポーネントトランザクションサービスの環境定義ファイルについて説明します。本定義ファイルは、以下に格納されます。

格納パス

Windows

C:¥Interstage¥etc¥sysdef

Solaris

/var/opt/FSUNtd/etc/sysdef

Linux

/var/opt/FJSVtd/etc/sysdef

本定義ファイルは、コンポーネントトランザクションサービスが停止している時に、変更可能です。
コンポーネントトランザクションサービスの環境定義の記述形式と制御文について説明します。

備考

万一のため、運用環境を構築したら資源のバックアップを行うことを推奨します。
バックアップについては、“運用ガイド(基本編)”の“メンテナンス(資源のバックアップ)”を参照してください。

B.1　記述形式

通常ファイル記述形式は、以下の構文により構成されます。なお、通常ファイルの記述形式に誤りがあった場合、構文エラーとなります。構文エラー時には、そのファイルに記述されているすべての内容が無効となります。

・ ステートメント

・ セクション

・ コメント行

・ 空行

B.1.1　ステートメント

ステートメントとは、情報を設定するための行であり、以下の形式で記述します。

```
キーワード: 設定内容(¥n)
```

ステートメントは、キーワード、“:”(コロン)、および設定内容から構成されています。
ステートメントの記述規則を以下に示します。

・ ステートメントを省略する場合、対象ステートメントを削除するか、設定内容だけ省略します。

・ ステートメントを記述している行にコメントは記述できません。

ステートメントを構成する情報の詳細を以下に説明します。

■キーワード

固有のキーワードを設定します。キーワードには、以下の規則があります。

・ キーワードは対象行の先頭の英数字から“:”(コロン)の直前までを意味します。

・ キーワードには英数字で始まる英数字とスペースで構成されている文字列を指定できます。

・ キーワード中に指定されている英字の大文字/小文字の区別はしません。

・ キーワード中にスペースを含むことができます。

・ キーワード中に連続したスペースが指定された場合、1つのスペースが指定されたものとみなされます。

・ 対象行の先頭にスペースやタブが指定されている場合、そのスペースやタブは無視されます。

■:(コロン)

キーワードと設定内容の区切りをあらわす文字として使用し、以下の規則があります。

・ コロンは必ず半角で指定する必要があります。

・ コロンの前後にスペースやタブが記述されている場合、そのスペースやタブは無視されます 。

■設定内容

キーワードに対応する内容を設定します。設定内容には、以下の規則があります。

・ 設定内容は、対象行の“:”の直後から指定できます。それ以降の“:”は設定内容に含まれる文字として扱われます。

・ 設定内容の終わりはスペース、タブ、改行('¥n')、またはEOFで示します。

・ 設定内容に指定されている英字の大文字/小文字は区別されます。

・ 設定内容は、1つの文字列しか指定できません。

・ 設定内容中にスペースやタブが含まれる場合、二重引用符で囲む必要があります。

・ 設定内容を複数記述する場合、ステートメントを繰り返し記述します。

ステートメントの設定例

例1)キーワード“Keyword:”には“Information”を設定します。

```
Keyword: Information(¥n)
KEYWORD: Information(¥n)
KeyWord: Information(¥n)
Keyword: Information(¥n)
Keyword: Information(¥n)
```

以上のステートメントはすべて同じように解析されます。

例2)キーワード“This is a Keyword:”には“Information Area”を設定します。

```
This is a Keyword: "Information Area"(¥n)
THIS IS A KEYWORD: "Information Area"(¥n)
This is a keyword: "Information Area"(¥n)
This is a Keyword: "Information Area"(¥n)
```

以上のステートメントはすべて同じように解析されます。

構文エラーとなるステートメントの例

設定内容が2つ指定されている(¥n)

```
Keyword: Information Area(¥n)
```

ステートメントを記述している行にコメントが指定されている(¥n)

```
Keyword: Information # This is a Statement(¥n)
```

二重引用符で終了していない(¥n)

```
Keyword: "START Information. (¥n)
```

キーワードと設定内容が2行で指定されている(¥n)

```
Keyword: "START Information. (¥n)
Information END" (¥n)
```

また、登録されていないキーワードを指定した場合も構文エラーとなります。

B.1.2 セクション

セクションとは、ステートメントの集合(グループ)であり、以下の形式で記述します。

```
[セクション名](¥n)
    キーワード: 設定内容 (¥n)
    キーワード: 設定内容 (¥n)
```

セクションは、“[セクション名]”と複数のステートメントから構成されており、以下の規則があります。

- セクションとは“[セクション名]”で始まり、次のセクション、またはEOFまでを意味します。
- セクションを省略する場合、セクションをすべて削除するか、コメントにする必要があります。
- “[セクション名]”だけのセクションは指定できません。
- “[セクション名]”を記述する行に“[セクション名]”以外は記述できません。
- セクション名は必ず[ ](角括弧)で囲む必要があります。
- セクション名には英数字で始まる英数字とスペースで構成されている文字列を指定できます。
- セクション名に指定されている英字の大文字/小文字の区別はしません。
- “[セクション名]”を記述している行にコメントは指定できません。

正常なセクションの設定例

セクションにステートメントが1つある(¥n)

```
[Section] (¥n)
  Keyword: Information(¥n)
```

セクションにステートメントが3つある(¥n)

```
[Section] (¥n)
  Keyword1: Information(¥n)
  Keyword2: Information(¥n)
  Keyword3: Information(¥n)
```

構文エラーとなるセクションの例

[セクション名]が記述されている行にコメントが指定されている(¥n)

```
[Section]          # This is a Section(¥n)
  Keyword:  Information(¥n)
```

[セクション名]の角括弧が正しく終了していない(¥n)

```
[Section(¥n)
  Keyword:  Information. (¥n)
```

また、登録されていないセクション名を指定した場合も構文エラーとなります。

B.1.3 コメント行

コメント行は、通常ファイル中にコメント(注釈)を記述する時に使用します。以下の形式で記述します。

```
# コメント(¥n)
```

コメント行には以下の規則があります。

- コメント行の先頭に#(シャープ)を指定します。
- #は半角文字で指定する必要があります。

B.1.4 空行

空行を記述することができます。
空行は解析時無視されます。

B.2 環境定義ファイルの制御文

B.2.1 [SYSTEM ENVIRONMENT]セクション

◆形式

[SYSTEM ENVIRONMENT]セクションは以下の形式で記述します。なお、本セクションは省略できません。

Windows

[SYSTEM ENVIRONMENT]

System Scale:	システムの規模
Number of Maximum WRAPPER Hold Session:	システム最大保留セション数
Using Interface Check:	インタフェース情報チェック機能使用の有無

Solaris

[SYSTEM ENVIRONMENT]

System Scale:	システムの規模
Number of Maximum WRAPPER Hold Session:	システム最大保留セション数
Using Interface Check:	インタフェース情報チェック機能使用の有無
IP version:	使用するネットワークのバージョン

Linux

[SYSTEM ENVIRONMENT]

System Scale:	システムの規模
Using Interface Check:	インタフェース情報チェック機能使用の有無

◆設定内容

System Scale:システムの規模

システムの規模を指定します。
Interstage統合コマンドにより初期化を行った場合には、適切なシステム規模が設定されますので、修正する必要はありません。
システム規模は、接続クライアント数より選択します。各定義値の意味を、以下に示します。

設定値	システム規模	接続クライアント数	
		Windows	Solaris Linux
small	小規模	1～5	1～50
moderate	中規模	6～10	51～100
large	大規模	11～50	101～500
super	超大規模	51～100	501～1000

Number of Maximum WRAPPER Hold Session:システム最大保留セション数 Windows Solaris

システム全体でのセション保留数を指定します。
0を指定した場合にはセションの抑制は行いません。
本定義は、[WRAPPER]セクションで指定するPSYS NameステートメントおよびNumber of Maximum Sessionステートメントを定義した場合に有効となります。
指定できる値は、0～1000です。省略値は、0です。

Using Interface Check:インタフェース情報チェック機能使用の有無

インタフェース情報チェック機能使用の有無を指定します。

- YES: インタフェース情報チェック機能を使用する。
- NO : インタフェース情報チェック機能を使用しない。省略値。

IP version:使用するネットワークのバージョン Solaris

使用するネットワークのバージョンを指定します。

- v6: IPv6ネットワークを使用する。
- v4: 従来のIPv4ネットワークを使用する。省略値。

B.2.2 [WRAPPER]セクション Windows Solaris

◆形式

[WRAPPER]セクションは以下の形式で記述します。

[WRAPPER]

PSYS Name: 負荷抑制対象DPCF通信パス名
Number of Maximum Session: 最大同時通信セション数

◆設定内容

PSYS Name:負荷抑制対象DPCF通信パス名

負荷抑制の対象とするDPCF通信パス名を8文字以内の英数字で指定します。
DPCF通信パス名については、以下を参照してください。

Windows

IDCMヘルプ

Solaris

IDCM使用手引書

Number of Maximum Session:最大同時通信セション数

PSYS Nameステートメントで設定したDPCF通信パスにおける同時に通信できるセション数の最大値を指定します。
指定できる値は、1～1000です。この値を超えたセションが、負荷抑制の対象となります。

- 負荷抑制機能を使用する場合、[SYSTEM ENVIRONMENT]セクションと[WRAPPER]セクションの各ステートメントをすべて定義してください。

- [WRAPPER]セクションのPSYS Nameステートメントと Number of Maximum Sessionステートメントはペアで指定します。

- 複数のDPCF通信パスに対して負荷抑制を行う場合には、複数の[WRAPPER]セクションにPSYS Nameステートメントと Number of Maximum Sessionステートメントをペアで指定します。

- Number of Maximum Sessionステートメントに指定する値は、IDCMネットワーク定義で指定した優先会話コネクション数以下の値を指定してください。優先会話コネクション数より大きな値を指定した場合、負荷抑制されないことがあります。
  IDCMネットワーク定義の詳細は、以下を参照してください。

  Windows

  IDCMヘルプ

  Solaris

  IDCM使用手引書

付録C　データベース連携サービスの環境定義

データベース連携サービスの環境定義について説明します。

C.1　configファイル

■概要

configファイルは、OTSシステム起動時に反映される情報を管理している定義ファイルです。configファイルの修正を反映させるには、OTSシステムを再起動する必要があります。

■ファイル名

configファイルは、インストール時に、以下に格納されます。

Windows

```
C:¥Interstage¥ots¥etc¥config
```

Solaris

```
/opt/FSUNots/etc/config
```

Linux

```
/opt/FJSVots/etc/config
```

注)本製品のインストールパスがデフォルトの場合のパスです。

■ファイル内情報

◆形式:

```
キー名=設定値
```

◆設定例

```
OBSERVE_CYCLE_TIME=6 (注)
TRAN_TIME_OUT=300
2PC_TIME_OUT=60
COM_RETRY_TIME=2 (注)
COM_RETRY_MAX=3 (注)
RECOVER_RETRY_TIME=30 (注)
RECOVER_RETRY_MAX=60 (注)
RESOURCE_TRANMAX=5
OTS_TRACE_SIZE=4096 (注)
RESOURCE_TRACE_SIZE=4096 (注)
RECOVERY_TRACE_SIZE=4096 (注)
OBSERVE_TRACE_SIZE=4096 (注)
DATABASE_RETRY_TIME=5 (注)
DATABASE_RETRY_MAX=5 (注)
MEM_RETRY_TIME=5 (注)
MEM_RETRY_MAX=5 (注)
RSCSTOP_CHECK_COUNT=100 (注)
OTS_VERSION=5 (注)
```

```
JTS_VERSION=5 (注)
TRACE_MODE=1
TRACE_LEVEL=1
JAVA_VERSION=14
PATH=C:¥Interstage¥JDK14¥JRE¥bin¥java.exe....(Windows(R)の場合)
PATH=/opt/FJSVawjbk/jdk14/jre/bin/java.......(Solaris/Linuxの場合)
```

注)インストール時に作成されたconfigファイルには、該当項目は記載されていません。項目を指定すると値は有効となりますが、省略値から変更しないことを推奨します。

- タイムアウトの詳細については、“OLTPサーバ運用ガイド”を参照してください。
- configファイルの項目は、すべて省略可能です。省略時は、省略値が有効となります。

◆キー一覧

キー	意味
OBSERVE_CYCLE_TIME	監視周期の指定
TRAN_TIME_OUT	トランザクションタイムアウト検出時間の指定
2PC_TIME_OUT	フェーズ間タイムアウト検出時間の指定
COM_RETRY_TIME	トランザクション処理エラー時のリトライ間隔指定
COM_RETRY_MAX	トランザクション処理リトライ上限回数の指定
RECOVER_RETRY_TIME	OTSシステムリカバリ処理リトライ間隔指定
RECOVER_RETRY_MAX	OTSシステムリカバリ処理リトライ上限回数の指定
RESOURCE_TRANMAX	1リソース管理プログラムのトランザクションの最大多重度
OTS_TRACE_SIZE	OTSシステムのトレースログサイズ指定
RESOURCE_TRACE_SIZE	リソース管理プログラムのトレースログサイズ指定
RECOVERY_TRACE_SIZE	リカバリプロセスのトレースログサイズ指定
OBSERVE_TRACE_SIZE	監視プロセスのトレースログサイズ指定
DATABASE_RETRY_TIME	データベースシステムアクセスのリトライ間隔指定
DATABASE_RETRY_MAX	データベースシステムアクセスのリトライ上限回数指定
MEM_RETRY_TIME	OTSシステム処理中のエラーでのリトライ間隔指定
MEM_RETRY_MAX	OTSシステム処理中のエラーでのリトライ上限回数指定
RSCSTOP_CHECK_COUNT	通常停止からのトランザクション待ち合わせ回数指定
OTS_VERSION	OTSのバージョン
JTS_VERSION	JTSのバージョン
JAVA_VERSION	JDK/JREのバージョン
PATH	JDK/JREのパス
TRACE_MODE	トレースの出力形式
TRACE_LEVEL	トレースの出力レベル

◆キー詳細

OBSERVE_CYCLE_TIME:監視周期の指定

OTSシステムが監視を実施する周期(秒単位)を指定します。以下の点に注意して設定してください。

- 本値に小さい値を設定すると、頻繁に監視が行われるため、システムのパフォーマンスが低下します。
- 本値に大きい値を設定すると、監視の間隔が長くなるため、異常の検出が遅くなります。

指定可能な範囲は、1～60です。省略値は、5です。

TRAN_TIME_OUT:トランザクションタイムアウト検出時間の指定

データベース連携サービスがトランザクションタイムアウト(beginからcommitまで)を検出する時間(秒単位)を指定します。アプリケーションにおいてset_timeoutメソッドでタイムアウト時間を設定した場合は、アプリケーションの設定が有効となります。
指定可能な範囲は、1～2147483647です。省略値は、300です。

2PC_TIME_OUT:フェーズ間タイムアウト検出時間の指定

OTSシステムがトランザクションの2PC(2フェーズコミット)で、リソース管理プログラムにおいて1フェーズと2フェーズ間のタイムアウトを検出する時間(秒単位)を指定します。
指定可能な範囲は、1～2147483647です。省略値は、60です。

CORBAサービスのクライアント側の無通信監視時間(CORBAサービスの動作環境ファイルのパラメタ“period_client_idle_con_timeout”の設定値×5)が“0”ではない場合、本値にはこの値よりも小さい値を設定してください。

COM_RETRY_TIME:トランザクション処理エラー時のリトライ間隔指定

トランザクション処理で通信異常などのエラーが発生した場合に、その通信をリトライする間隔(秒単位)を指定します。
指定可能な範囲は、1～600です。省略値は、2です。

COM_RETRY_MAX:トランザクション処理リトライ上限回数の指定

トランザクション処理で通信異常などのエラーが発生した場合に、その通信をリトライする上限回数を指定します。
指定可能な範囲は、1～2147483647です。省略値は、3です。

RECOVER_RETRY_TIME:OTSシステムリカバリ処理リトライ間隔指定

OTSシステムのリカバリ処理で通信異常などのエラーが発生した場合に、その通信をリトライする間隔(秒単位)を指定します。
指定可能な範囲は、1～600です。省略値は、30です。

RECOVER_RETRY_MAX:OTSシステムリカバリ処理リトライ上限回数の指定

OTSシステムのリカバリ処理で通信異常などのエラーが発生した場合に、その通信をリトライする上限回数を指定します。
指定可能な範囲は、1～2147483647です。省略値は、60です。

RESOURCE_TRANMAX:1リソース管理プログラムのトランザクションの最大多重度

1リソース管理プログラムのトランザクションの最大多重度を指定します。
指定可能な範囲は、1～2147483647です。省略値は、5です。

OTSシステムのスレッド多重度と1リソース管理プログラムのトランザクションの最大多重度は、以下の関係を保つように設定してください。

OTSシステムのスレッド多重度 ＝< 1リソース管理プログラムのトランザクションの最大多重度

OTS_TRACE_SIZE:OTSシステムのトレースログサイズ指定

OTSシステムのトレースログサイズ（Kバイト単位）を指定します。
指定可能な範囲は、128～2147483647です。省略値は、4096（Kバイト）です。

RESOURCE_TRACE_SIZE:リソース管理プログラムのトレースログサイズ指定

リソース管理プログラムのトレースログサイズ（Kバイト単位）を指定します。
指定可能な範囲は、128～2147483647です。省略値は、4096（Kバイト）です。

RECOVERY_TRACE_SIZE:リカバリプロセスのトレースログサイズ指定

リカバリプロセスのトレースログサイズ（Kバイト単位）を指定します。
指定可能な範囲は、128～2147483647です。省略値は、4096（Kバイト）です。

OBSERVE_TRACE_SIZE:監視プロセスのトレースログサイズ指定

監視プロセスのトレースログサイズ（Kバイト単位）を指定します。
指定可能な範囲は、128～2147483647です。省略値は、4096です。

DATABASE_RETRY_TIME:データベースシステムアクセスのリトライ間隔指定

OTSシステムでのデータベースシステムへのアクセス時に、メモリ資源不足などのリトライ可能なエラーが発生した場合のリトライ間隔（秒単位）を指定します。
指定可能な範囲は、1～600です。省略値は、5です。

DATABASE_RETRY_MAX:データベースシステムアクセスのリトライ上限回数指定

OTSシステムでのデータベースシステムへのアクセス時に、メモリ資源不足などのリトライ可能なエラーが発生した場合にリトライする上限回数を指定します。
指定可能な範囲は、1～2147483647です。省略値は、5です。

MEM_RETRY_TIME:OTSシステム処理中のエラーでのリトライ間隔指定

OTSシステムの処理中に、メモリ資源不足などのリトライ可能なエラーが発生した場合のリトライ間隔（秒単位）を指定します。
指定可能な範囲は、1～600です。省略値は、5です。

MEM_RETRY_MAX:OTSシステム処理中のエラーでのリトライ上限回数指定

OTSシステムの処理中に、メモリ資源不足などのリトライ可能なエラーが発生した場合にリトライする上限回数を指定します。
指定可能な範囲は、1～2147483647です。省略値は、5です。

RSCSTOP_CHECK_COUNT:通常停止からのトランザクション待合せ回数指定

トランザクション処理中にリソース管理プログラムを通常停止し、トランザクション完了をOBSERVE_CYCLE_TIMEの監視同期に合わせた待合せ回数を指定します。
OBSERVE_CYCLE_TIME × RSCSTOP_CHECK_COUNT秒間、トランザクションの完了を待ち合わせ、時間内に完了しない場

合は、リソース管理プログラムの停止を通常停止から強制停止に切り替えます。
指定可能な範囲は、1～2147483647です。省略値は、100です。

OTS_VERSION:OTSのバージョン

OTSのバージョンを指定します。通常、変更しないでください。
省略値は、5です。

JTS_VERSION:JTSのバージョン

JTSのバージョンです。通常、変更しないでください。
省略値は、5です。

JAVA_VERSION:JDK/JREのバージョン

JTS用リソース管理プログラムが利用するJavaのバージョンを指定します。
14を指定します。

PATH:JDK/JREのパス

JTS用リソース管理プログラムが利用するjavaコマンドへのパスをフルパスで指定します。java実行体を含めるパスを指定してください。
初期値は、インストール時に指定したJDK/JREのバージョンに対応したパスです。

- JTS用のリソース管理プログラムを利用する場合は、必ず指定してください。
- Interstage Application Serverに同梱されているJDK/JREのパスを指定してください。
- Solaris Linux
  JDK/JREを入れ替えた場合は、JDK/JREのバージョンに合わせて、再度指定してください。

TRACE_MODE:トレースの出力形式

JTSを利用した環境で出力されるトレースの出力形式を、以下から選択して指定します。

- 異常発生時に、OTSのインストールパス/var配下にトレースファイルを出力する場合:“1”
  注)通常、“1”を選択してください。
- システムの状態に関係なく、常時、OTSのインストールパス/var配下にトレースファイルを出力する場合:“2”
  注)常時出力されるため、ファイルサイズに注意してください。
- トレースファイルを出力しない場合:“3”

TRACE_LEVEL:トレースの出力レベル

JTSを利用した環境で出力されるトレースのモードを、“1”から“5”までの数値で指定します。数値が大きいほど、詳細なトレース情報を出力します。
通常運用時は、“1”が指定されます。パフォーマンスに影響があるため、通常、変更しないでください。

C.2 セットアップ情報ファイル

■概要

セットアップ情報ファイルは、otssetupコマンドによるOTSシステム動作環境の設定時に指定するファイルです。isinitコマンドでInterstageの初期化を行う場合は、“Interstage動作環境定義”をカスタマイズする必要があります。Interstage動作環境定義については“運用ガイ

ド(基本編)”を参照してください。
セットアップ情報ファイルは、otssetupコマンド実行時に作成します。1度作成したセットアップ情報ファイルは、次回のセットアップ時にも使用可能であるため、保管しておいてください。

■ファイル内情報

◆形式:

```
パラメタ名=設定値
```

◆設定例

```
MODE=SYS
LOGFILE=c:¥ots¥logfile.....(Windows(R)の場合)
LOGFILE=/dev/rdsk/c0t0d0s0....(Solarisの場合)
LOGFILE=/dev/raw/raw1.....(Linuxの場合)
TRANMAX=10
PARTICIPATE=4
OTS_FACT_THR_CONC=5
OTS_RECV_THR_CONC=2
JTS_RMP_PROC_CONC=5
JTS_RMP_THR_CONC=16
HOST=otshost
PORT=8002
LOCALE=EUC
```


ポイント

太字以外の項目は、省略可能です。省略時は、省略値が有効となります。

C.2.1 MODE: セットアップ種別

OTSシステムが動作するホストか、リソース管理プログラムだけが動作するホストかを、以下から選択して指定します。大文字で指定してください。

- OTSシステムおよびリソース管理プログラムが動作するホストの場合:“SYS”
  OTSシステム動作環境のセットアップ、リソース管理プログラムの動作環境のセットアップ、およびシステムログファイルの作成を行います。
- リソース管理プログラムだけが動作するホストの場合:“RMP”
  リソース管理プログラムの動作環境のセットアップだけを行います。
  注)“RMP”を設定してセットアップした環境では、OTSシステムを起動できません。

Interstage動作環境定義ファイルの“OTS Setup mode”に相当します。

“RMP”を設定する場合、リソース管理プログラムを正しく動作させるために、OTSシステムが動作するホストのネーミングサービスを参照する必要があります。以下のいずれかの方法でセットアップを行ってください。

- “RMP”を設定すると同時に、OTSシステムが動作しているホストの“HOST”、“PORT”、“LOCALE”を指定してセットアップを行います。この場合、ネーミングサービスはOTSシステムではなく、“RMP”を設定したホストのネーミングサービスが利用されます。“RMP”を設定したホストとOTSシステムが動作するホストのネーミングサービスをそれぞれ独立させて運用させることが可能となります。
- isinitコマンドに“type3”を指定してInterstageの初期化(NS Use、NS Host、NS Port Number、NSに、OTSシステムが動作するホストのネーミングサービスを設定)を行った後、“RMP”を設定してotssetupコマンドでセットアップを行ってください。これにより、OTSシステムが動作するホストと、“RMP”を設定したホストのネーミングサービスは、共有されます。


注意

・ “SYS”を設定したホスト同士のネーミングサービスを共有することはできません。必ず1つのネーミングサービスには1つの“SYS”を設定したホストになるようにしてください。“RMP”を設定する場合は、複数のホストでネーミングサービスを共有できます。

・ “RMP”を設定した場合、otsstartコマンドではOTSシステムを起動できません。otsstartrscコマンドでリソース管理プログラムの起動だけを行うことができます。

C.2.2 LOGFILE: システムログファイルのパス

OTSシステムのシステムログファイルを指定します。

Windows

OTSシステムのシステムログファイルへのパス(ドライブ名を含む絶対パス)を、制御文字(ShiftJISコードの0x00～0x1F,0x7F)を除く文字列で指定します。半角英文字の大文字と小文字、全角英文字の大文字と小文字は区別されません。

Solaris Linux

OTSシステムのシステムログファイルと使用するローデバイス・ファイル名を、スラッシュ(/)で始まる空白文字と半角カナを除く文字列で指定します。

Linux

RHEL-AS4の場合は、rawデバイス名を指定してください。
RHEL5／RHEL6の場合は、ブロックデバイス名を指定してください。

“MODE”に“SYS”を設定した場合に有効となります。
最大長は、255文字です。
Interstage動作環境定義ファイルの“OTS path for system log”に相当します。

Linux

以下にローデバイスの作成手順を示します。

■ RHEL-AS4の場合

1. オペレーティングシステムのpartedコマンド／fdiskコマンドで、ローデバイスのパーティションを作成します。
2. 作成したパーティションをバインドします。
   partedコマンドを使用した場合の例を以下に示します(#:プロンプト)。

```
# parted /dev/sdb
(parted) p
/dev/sdbの Disk geometry: 0.000-34732.890 メガバイト
ディスクラベルの種類: gpt
マイナー 開始         終了     ファイルシステム 名前       フラグ
1           0.017   2048.002  linux-swap
2        2048.002  12048.002  ext3
3       12048.002  13072.000
(parted) q
# udevinfo -q path -n /dev/sdb3
/block/sdb/sdb3
# udevinfo -q symlink -p /block/sdb/sdb3
disk/by-path/pci-0000:20:01.0-scsi-0:0:1:0p3  disk/by-id/SHP_36.4GST336753LC_3HX2BF0R000074446H48p3
# raw /dev/raw/raw1 /dev/disk/by-path/pci-0000:20:01.0-scsi-0:0:1:0p3
```

   なお、rawコマンドによるバインドはマシンを起動するたびに毎回実施する必要があります。この処理を自動化するには、以下の方法があります。
   /etc/sysconfig/rawdevices に上記で示したrawコマンドに渡したパラメタと同じものを記載してください。

```
/dev/raw/raw1 /dev/by-path/pci-0000:20:01.0-scsi-0:0:1:0p3
```

3. udevによりローデバイスのアクセス権限が正しく設定されるように、/etc/udev/permissions.d/ディレクトリにある追加パーミッションルールファイルを必要に応じて編集します。

- ローデバイスをバインドするブロックデバイスはパーティションを指定してください。パーティション番号のないハードディスクデバイス(/dev/sdgなど)は、ディスクラベル(パーティションテーブル)を含んでいるため、ローデバイスとして使用しないでください。
- rawコマンドは、マウント済みのシステム用デバイスを指定しても正常にキャラクタデバイスへのバインド処理を実施します。rawコマンドの第2パラメタに指定するデバイス名は誤りのないよう指定してください。
  誤って指定した場合、システムおよびユーザ資産を破壊する可能性があります。
- セットアップ情報ファイルのログファイルの指定には、必ずrawコマンドでキャラクタデバイスにバインドしたデバイス名を指定してください。

■ RHEL5の場合

1. オペレーティングシステムのpartedコマンド／fdiskコマンドで、ローデバイスのパーティションを作成します。
2. ディスクのパーティションに対応するudevのブロックデバイス名を特定します。
   partedコマンドを使用した場合の例を以下に示します(#:プロンプト)。

```
# parted /dev/sda
(parted) p
      :

番号  開始     終了     サイズ  タイプ     ファイルシステム  フラグ
1     32.3kB   107MB    107MB   プライマリ  ext3             boot
2     107MB    9656MB   9550MB  プライマリ                   lvm
3     9656MB   10.7GB   1078MB  プライマリ                   lvm

(parted) q
# udevinfo -q path -n /dev/sda3
/block/sda/sda3
# udevinfo -q env -p /block/sda/sda3 | grep ID_PATH
ID_PATH=pci-0000:00:10.0-scsi-0:0:0:0
```

3. udevの設定ファイル(/etc/udev/rules.d/60-raw.rules)を編集し、作成したパーティションをバインドします。

```
ACTION=="add", KERNEL=="sda3", ENV{ID_PATH}=="pci-0000:00:10.0-scsi-0:0:0:0", RUN+="/bin/raw /dev/raw/raw1 %N"
```

4. udevによりローデバイスのアクセス権限が正しく設定されるように、/etc/udev/rules.d/配下の追加パーミッションルールファイルを必要に応じて編集します。

- ローデバイスをバインドするブロックデバイスはパーティションを指定してください。パーティション番号のないハードディスクデバイス(/dev/sdgなど)は、ディスクラベル(パーティションテーブル)を含んでいるため、ローデバイスとして使用しないでください。
- セットアップ情報ファイルのログファイルの指定には、必ずキャラクタデバイスにバインドしたデバイス名を指定してください。

■ RHEL6の場合

1. オペレーティングシステムのpartedコマンド／fdiskコマンドで、ローデバイスのパーティションを作成します。
2. ディスクのパーティションに対応するudevのブロックデバイス名を特定します。
   partedコマンドを使用した場合の例を以下に示します(#:プロンプト)。

```
# parted /dev/sda
(parted) p
      :
```

```
番号  開始    終了    サイズ  タイプ    ファイルシステム  フラグ
1     1049kB  211MB   210MB   primary   ext4              boot
2     211MB   32.4GB  32.2GB  primary   ext4
      :
8     77.5GB  78.5GB  974MB   logical

(parted) q
# udevadm info --query=path --name=/dev/sda8
/devices/pci0000:00/0000:00:1f.2/host0/target0:0:0/0:0:0:0/block/sda/sda8
# udevadm info --query=property --path=/devices/pci0000:00/0000:00:1f.2/host0/target0:0:0/0:0:0:0/block/sda/sda8 |
grep ID_PATH
ID_PATH=pci-0000:00:1f.2-scsi-0:0:0:0
```

3. udevの設定ファイル(/etc/udev/rules.d/60-raw.rules)を編集し、作成したパーティションをバインドします。

```
ACTION=="add", KERNEL=="sda8", ENV{ID_PATH}=="pci-0000:00:1f.2-scsi-0:0:0:0", RUN+="/bin/raw /dev/raw/raw1 %N"
```

4. udevによりローデバイスのアクセス権限が正しく設定されるように、/etc/udev/rules.d/配下の追加パーミッションルールファイルを必要に応じて編集します。

- ローデバイスをバインドするブロックデバイスはパーティションを指定してください。パーティション番号のないハードディスクデバイス(/dev/sdgなど)は、ディスクラベル(パーティションテーブル)を含んでいるため、ローデバイスとして使用しないでください。
- セットアップ情報ファイルのログファイルの指定には、必ずキャラクタデバイスにバインドしたデバイス名を指定してください。

C.2.3 TRANMAX: 最大トランザクション多重度

トランザクションの最大数を指定します。必ず指定する必要があります。
また、MODEに“RMP”を設定した場合は、連携するOTSシステム(MODEに“SYS”を設定しているシステム)と同じ値を設定してください。

Windows

設定可能な範囲は、1～256です。

Solaris Linux

設定可能な範囲は、1～1024です。

Interstage動作環境定義ファイルの“OTS maximum Transaction”に相当します。

C.2.4 PARTICIPATE: 1トランザクションに参加するリソース数

1トランザクションに参加するリソースの最大数を指定します。
MODEに“SYS”を設定した場合に有効となります。
設定可能な範囲は、2～32です。省略値は、4です。
Interstage動作環境定義ファイルの“OTS Participate”に相当します。

C.2.5 OTS_FACT_THR_CONC: OTSシステムのスレッド多重度

OTSシステムのスレッド多重数を指定します。指定した数だけ、begin/commit/rollbackなどのCurrentインタフェース、およびUserTransactionインタフェースを同時に動作させることが可能となります。
MODEに“SYS”を設定した場合に有効となります。
設定可能な範囲は、1～31です。省略値は、5です。
Interstage動作環境定義ファイルの“OTS Multiple degree”に相当します。
最大値を超えた場合は、警告メッセージ(ots9013)が出力され、自動的に31が設定されます。


ポイント

OTSシステムのスレッド多重度は、トランザクション処理性能を最大限に引き出すようにチューニングされているため、省略値から変更する必要はありません。
変更する場合は、以下の関係を保つように設定してください。

```
OTSシステムのスレッド多重度 ＝＜ リソース管理プログラムの多重度(注)
OTSシステムのスレッド多重度 ＝＜ 1リソース管理プログラムのトランザクションの最大多重度
```

注)JTS用リソース管理プログラムにおける多重度は、以下の算出式で見積もってください。

JTS用のリソース管理プログラムのプロセス多重度:JTS_RMP_PROC_CONC ×
JTS用のリソース管理プログラムのスレッド多重度:JTS_RMP_THR_CONC

C.2.6 OTS_RECV_THR_CONC: リカバリプロセスのスレッド多重度

リカバリプログラムのスレッド多重数を指定します。指定した数だけ、リカバリ処理を同時に動作させることが可能となります。
MODEに“SYS”を設定した場合に有効となります。
設定可能な範囲は、1～2147483647です。省略値は、2です。
Interstage動作環境定義ファイルの“OTS Recovery”に相当します。

C.2.7 JTS_RMP_PROC_CONC: JTS用のリソース管理プログラムのプロセス多重度

JTS用のリソース管理プログラムのプロセス多重度を指定します。使用するリソース(データベース、リソースアダプタなど)の数だけ指定することを推奨します。
設定可能な範囲は、1～32です。省略値は、2です。5以下の場合は、変更する必要はありません。
Interstage動作環境定義ファイルの“OTS JTS's RMP Multiple degree of Process”に相当します。
最大値を超えた場合は、警告メッセージ(ots9017)が出力され、自動的に32が設定されます。


ポイント

リソース管理プログラムの多重度は、トランザクション処理性能を最大限に引き出すようにチューニングされているため、省略値から変更する必要はありません。
変更する場合は、OTSシステムのスレッド多重度とリソース管理プログラムの多重度の関係を以下のように設定してください。

```
OTSシステムのスレッド多重度 ＝＜ リソース管理プログラムの多重度(注)
```

注)JTS用リソース管理プログラムにおける多重度は、以下の算出式で見積もってください。
JTS用のリソース管理プログラムのプロセス多重度:JTS_RMP_PROC_CONC ×
JTS用のリソース管理プログラムのスレッド多重度:JTS_RMP_THR_CONC

C.2.8 JTS_RMP_THR_CONC: JTS用のリソース管理プログラムのスレッド多重度

JTS用のリソース管理プログラムのスレッド多重度を指定します。
設定可能な値は、1～2147483647です。省略値は、16です。
通常、変更する必要はありません。
Interstage動作環境定義ファイルの“OTS JTS's RMP Multiple degree of Thread”に相当します。


ポイント

リソース管理プログラムの多重度は、トランザクション処理性能を最大限に引き出すようにチューニングされているため、省略値から変更する必要はありません。
変更する場合は、OTSシステムのスレッド多重度とリソース管理プログラムの多重度の関係を以下のように設定してください。

```
OTSシステムのスレッド多重度 ＝＜ リソース管理プログラムの多重度(注)
```

注)JTS用リソース管理プログラムにおける多重度は、以下の算出式で見積もってください。
JTS用のリソース管理プログラムのプロセス多重度:JTS_RMP_PROC_CONC ×
JTS用のリソース管理プログラムのスレッド多重度:JTS_RMP_THR_CONC

C.2.9 HOST: OTSシステムが動作するホスト名

OTSシステムが利用するネーミングサービスのホスト名を、英数字、マイナス記号(-)、またはピリオド(.)から構成される64文字以内の英字から始まる文字列で指定します。最後の文字には、マイナス記号(-)、またはピリオド(.)を記述できません。
MODEに“RMP”を設定した場合に有効となります。
本ステートメントは、省略可能です。設定する場合は、PORTを同時に設定する必要があります。
本ステートメントの利用方法については、MODEを参照してください。
Interstage動作環境定義ファイルの“OTS Host”に相当します。

“isinit”コマンドで“type3”を選択している場合は、使用しないでください。

C.2.10 PORT: OTSシステムが動作するホストのCORBAサービスのポート番号

OTSシステムが利用するネーミングサービスのポート番号を指定します。
MODEに“RMP”を設定した場合に有効となります。
設定可能な範囲は、1024～65535です。
本ステートメントは、省略可能です。設定する場合は、HOSTを同時に設定する必要があります。
本ステートメントの利用方法については、MODEを参照してください。
Interstage動作環境定義ファイルの“OTS Port”に相当します。

isinitコマンドに“type3”を指定してInterstageを初期化した場合は、使用しないでください。

C.3 RMPプロパティ

■概要

JTS用リソース管理プログラムに対するプロパティファイルです。
RMPプロパティは、Javaのプロパティリストの形式で、その他のキーを設定することもできます。設定されたキーと値は、JTS用リソース管理プログラムが動作するJavaVMのシステムプロパティに反映されます。

■ファイル名

RMPプロパティファイルは、インストール時に、以下に格納されます。

Windows

```
C:¥Interstage¥ots¥etc¥RMP.properties
```

Solaris

```
/opt/FSUNots/etc/RMP.properties
```

Linux

```
/opt/FJSVots/etc/RMP.properties
```

注）本製品のインストールパスがデフォルトの場合のパスです。

■ファイル内情報

◆形式:

```
パラメタ名=設定値
```

◆パラメタ

RecoveryTarget: リカバリ対象

JTS用のリソース管理プログラムの起動時にリカバリを行う対象をリソース定義名で指定します。リカバリ対象が設定されていない場合、リカバリ対象が複数の場合は、空白で区切って指定してください。

RecoveryTargetを省略した場合でも、登録済みリソースに対してリカバリ処理を行います。

リカバリ対象が3つの場合の記述例

```
RecoveryTarget=Oracle_resource1 Oracle_resource2 Oracle_resource3
```

JavaPath: Javaコマンドへのパス

必要な場合に、記述を追加します。通常、指定しないでください。指定した場合、動作の保証はできません。

JavaCommandOption: Javaコマンドに受け渡すオプション

必要な場合に、記述を追加します。通常、指定しないでください。指定した場合、動作の保証はできません。

Classpath: 追加するクラスパス

Windows

必要な場合に、記述を追加します。通常、指定しないでください。指定した場合、動作の保証はできません。

Solaris Linux

クラスタサービス機能利用時に、リソースと連携するために必要となるクラスライブラリへのパスを設定します。クラスライブラリの詳細については、“アプリケーション作成ガイド(データベース連携サービス編)”の“リソース管理プログラムの作成から起動まで”を参照してください。

C.4 リソース定義ファイル

■概要

OTSおよびJTSが連携するリソース（データベース、リソースアダプタなど）に接続するための情報を定義するファイルです。otssetrscコマンドを利用して、リソース単位に登録します。

■ファイル内情報

◆形式:

```
キー名=設定値
```

◆設定例

OTS用リソース定義ファイル

```
# 環境変数
 ENVIRON ORACLE_SID=orac
 ENVIRON ORACLE_HOME=/opt/oracle.............(Solaris/Linuxの場合)
 ENVIRON LD_LIBRARY_PATH=/opt/oracle/lib.....(Solaris/Linuxの場合)
# 使用するデータベースシステム名とOPENINFO文字列、CLOSEINFO文字列
 NAME=oracle_rmp_thread
 RMNAME=Oracle_XA
 OPENINFO=Oracle_XA+Acc=P/system/manager+SesTm=0+Threads=true
 CLOSEINFO=
 THREADS=TRUE................................(Solaris/Linuxの場合)
```

JTS用リソース定義ファイル

```
# database1
 name=xads1
 rscType=JTS
 type=JDBC
 lookUpName=jdbc/XADataSource
 initialContextFactory=com.sun.jndi.fscontext.RefFSContextFactory
 providerURL=file:/tmp/JNDI
 user=dbuser
 password=dbpass
 logfileDir=c:¥interstage¥ots¥var............(Windows(R)の場合)
 logfileDir=/opt/FSUNots/var.................(Solarisの場合)
 logfileDir=/opt/FJSVots/var.................(Linuxの場合)
```

キーの名前は、OTSおよびJTSで大文字、小文字が異なりますが、同じ意味を持ちます。

◆キー一覧

キー(OTS)	キー(JTS)	意味
ENVIRON	－	環境変数の設定
NAME	name	リソース定義名
RMNAME	－	リソースマネージャ名
OPENINFO	－	オープン文字列
CLOSEINFO	－	クローズ文字列
THREADS	－	スレッドモード
OTS_RMP_PROC_CONC	－	OTS用のリソース管理プログラムの多重度
RSCTYPE	rscType	リソース定義ファイルの種類
－	type	リソースの種類
－	lookupName	リソースの検索名
－	initialContextFactory	initialContextFactory名

キー(OTS)	キー(JTS)	意味
－	providerURL	プロバイダURL
USER	－	ユーザ名
－	user	ユーザ名
－	password	パスワード
GROUP	－	グループ名
－	logfileDir	リソースのログファイル格納ディレクトリ

◆キー詳細

ENVIRON: 環境変数の設定(OTS)

値*data*に、リソース管理プログラム、またはリソース管理プログラムと同じプロセス内で動作するデータベースライブラリに渡す環境変数envを指定します。省略可。

Solaris Linux

リソース管理プログラムを使用するサーバアプリケーションの起動時に指定するデータベースへの環境変数と同一の環境変数を指定してください。

また、リソース定義ファイルには、以下のような$指定できません。

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/oracle/lib
```

使用するデータベースがSymfoware/RDBの場合は、環境変数LD_LIBRARY_PATHにSymfoware/RDBの必須製品であるライブラリのパス名を指定してください。

NAME、name: リソース定義名(OTS、JTS)

otssetrscコマンドでの登録時に、本パラメタに指定したリソース定義名で登録します。1度登録されたリソース定義ファイルは、すべてリソース定義名で扱うことができます。リソース定義名は、32文字以内で指定します。省略不可。

“JTSRMP”は予約語のため、リソース定義名として使用できません(一部またはすべてを小文字にしても使用できません)。

JTS用リソース定義ファイルでは、isj2eeadminコマンドで登録するJ2EEリソース定義の接続対象となるリソースの“定義名”を指定してください。詳細については、“リファレンスマニュアル(コマンド編)”の“isj2eeadmin”を参照してください。

RMNAME: リソースマネージャ名(OTS)

*system_name*に、データベースのシステム名を以下から選択して指定します。

- Oracleの場合:“Oracle_XA”
- Symfoware/RDBの場合:“RDBII”
- Windows
  SQL Serverの場合:“MS_SQL_Server”
- Windows Solaris
  MQDの場合:“XA_MQD”

OPENINFO: オープン文字列(OTS)

*open_data*に、データベースのベンダが公開する、データベースをオープンする場合に必要なopen文字列を、256文字以内の文字列で指定します。

指定する内容については、各データベースのマニュアルを参照してください。

OPENINFOに指定するユーザ名は、各データベースに対するアクセス権限がないと、リソース管理プログラムの起動に失敗します。必要な権限については、各データベースのマニュアルを参照してください。

Solaris Linux

プロセスモード／スレッドモードのタイプが、リソース管理プログラム作成時と動作時（リソース定義ファイル内のスレッド指定）とで異なる場合、リソース管理プログラムの起動が誤動作する可能性があります。必ずタイプをあわせて運用してください。

CLOSEINFO: クローズ文字列(OTS)

*close_data*に、データベースのベンダが公開する、データベースをクローズする場合に必要なclose文字列を、256文字以内の文字列で指定します。
指定する内容については、各データベースのマニュアルを参照してください。

THREADS: スレッドモード(OTS) Solaris Linux

リソース管理プログラムのモードを以下から選択して指定します。

- プロセスモードの場合：“FALSE”（省略時）
- スレッドモードの場合：“TRUE”

OTS_RMP_PROC_CONC: OTS用のリソース管理プログラムの多重度(OTS)

OTS用のリソース管理プログラムの多重数を指定します。通常、変更する必要がありません。
指定可能な範囲は、1～31です。省略値は、5です。
最大値を超えた場合は、警告メッセージ(ots9017)を出力し、31を自動的に設定します。

リソース管理プログラムの多重度は、トランザクション処理性能を最大限に引き出すようにチューニングされているため、省略値から変更する必要はありません。
変更する場合は、OTSシステムのスレッド多重度とリソース管理プログラムの多重度の関係を以下のように設定してください。

OTSシステムのスレッド多重度 ＝＜ リソース管理プログラムの多重度

RSCTYPE、rscType: リソース定義ファイルの種類(OTS、JTS)

リソース定義ファイルの種別を以下から選択して指定します。

- OTSを利用する場合：“OTS”（省略時）
- JTSを利用する場合：“JTS”
  注）JTSを利用する場合は、必ず“JTS”を指定してください。

type: リソースの種類(JTS)

リソースの種類を以下から選択して指定します。省略不可。

- JDBCを利用してデータベースと接続する場合：“JDBC”／“DBMS”(旧バージョンでの指定方法)
- J2EE Connector Architectureを利用してリソースアダプタと接続する場合は、“JCA”

lookupName: リソースの検索名(JTS)

JDBCを利用してデータベースと接続する場合は、データベースが提供するデータソースをバインドした名前を指定します。isj2eeadminコマンドで登録するJ2EEリソース定義で設定したデータソース名と同じ値を指定してください。

J2EE Connector Architectureを利用してリソースアダプタと接続する場合は、リソースアダプタの配備時に設定した“リソース名”を指定してください。

initialContextFactory: initialContextFactory名(JTS)

バインドされたデータソース参照時に使用するinitialContextFactoruy名を指定します。isj2eeadminコマンドで登録するJ2EEリソース定義で設定したクラス名と同じ値を指定してください。JDBCを利用してデータベースと接続する場合は、必ず指定してください。

providerURL: プロバイダURL(JTS)

バインドされたデータソース参照時に使用するprovider URLを指定します。isj2eeadminコマンドで登録するJ2EEリソース定義で設定したクラス名と同じ値を指定してください。

USER: ユーザ名(OTS) Solaris Linux

リソース管理プログラムの実行ユーザを指定します。otssetrscコマンド実行時に、-uオプションを指定した場合は、オプションに指定されたユーザ名が有効となります。

“GROUP”と同時に指定する必要があります。

指定したユーザは、“GROUP”に指定するグループに所属している必要があります。

強化セキュリティモードの場合は、強化セキュリティモード設定時に指定したグループに所属している必要があります。

user: ユーザ名(JTS)

リソースとの接続時にユーザ名が必要な場合に指定します。isj2eeadminコマンドで登録するJ2EEリソース定義によって設定したユーザ名を指定してください。

password: パスワード(JTS)

リソースとの接続時にパスワードが必要な場合に指定します。isj2eeadminコマンドで登録するJ2EEリソース定義によって設定したユーザのパスワードを指定してください。

GROUP: グループ名(OTS) Solaris Linux

リソース管理プログラムの実行ユーザを指定します。otssetrscコマンド実行時に、-gオプションを指定した場合は、オプションに指定されたグループ名が有効となります。

“USER”と同時に指定する必要があります。

強化セキュリティモードの場合、本設定は無効となり、強化セキュリティモード設定時に指定したグループ名が有効となります。

logfileDir: リソースのログファイル格納ディレクトリ(JTS)

接続したリソースのトラブルを調査する場合は、トレースログを採取するディレクトリを指定してください。ディレクトリ名の最後に、セパレータを付加しないでください。

通常、指定しません。

C.5 OTSシステム用業務システム情報定義ファイル

■概要

トランザクションサービスがSystemwalker Resource Coordinatorと連携して動的ダウンリカバリを実現するためのOTSシステム用業務システム情報定義ファイルです。

■ファイル内情報

◆形式:

```
キー名=設定値
```

◆パラメタ

factory_quemode: トランザクション処理閉塞の動作(OTS、JTS)

トランザクションサービスのトランザクション処理閉塞の動作を、以下から選択して指定します。

- トランザクション開始処理を受け付けず、トランザクション開始要求をエラーとする場合:“que”
- トランザクション開始処理を受け付け、トランザクション処理を待ち状態とする場合:“dsp”

C.6 アプリケーション用業務システム情報定義ファイル

■概要

トランザクションサービスがSystemwalker Resource Coordinatorと連携して動的ダウンリカバリを実現するためのアプリケーション用業務システム情報定義ファイルです。

■ファイル内情報

◆形式:

```
キー名=設定値
```

◆パラメタ

resource: リソース管理プログラムの情報(OTS)

対象となる業務で使用しているリソース管理プログラムのフルパスとリソース定義名を、カンマ(,)で区切って指定します。

- リソース管理プログラムのフルパスは、必ずダブルクォーテーション("")で囲んでください。
- 複数指定する場合は、複数行で指定してください。

workUnit: ワークユニットの情報(OTS、JTS)

対象となる業務のワークユニット名を指定します。

複数指定する場合は、カンマ(,)で区切ります。複数行で指定することもできます。

付録D　イベントサービスの環境定義

イベントサービスの動作環境ファイル、およびイベントチャネル・サプライヤ・コンシューマ総数の見積もり方法について説明します。
イベントサービスの動作環境ファイルは、以下に格納されます。

格納パス

Windows

C:¥Interstage¥eswin¥etc (インストールパスはデフォルト)

Solaris

/etc/opt/FJSVes (インストールパスはデフォルト)

Linux

/etc/opt/FJSVes

ファイル

traceconfig

上記以外のファイルは、イベントサービスの動作環境としてカスタマイズできません。エディタなどで編集しないでください。

D.1　traceconfig

■概要

traceconfigファイルは、イベントサービスのトレース動作環境に関する定義が格納されたファイルです。

■ファイル名

Windows

C:¥Interstage¥eswin¥etc¥traceconfig (インストールパスはデフォルト)

Solaris

/etc/opt/FJSVes/traceconfig (インストールパスはデフォルト)

Linux

/etc/opt/FJSVes/traceconfig

■ファイル内情報

traceconfigファイルは、以下の形式で値を設定します。

◆形式:

パラメタ名 = 設定値

◆パラメタ:

以下の動作環境について、パラメタ設定値を変更することができます。

パラメタ値を変更した場合、次回のイベントサービス起動時より有効となります。

パラメタ名	初期値 省略値 指定範囲	意味
trace_size	1024 512 512～102400	トレース情報の採取に使用するバッファのサイズをキロバイト単位で指定します。(注1)
trace_file_number	Windows Solaris 50 Linux 10	採取するトレース情報ファイルの最大数を指定します。トレース情報ファイルの数が指定値を超えた場合は、古いトレース情報ファイルに上書きします。
	Windows Solaris 50 Linux 10	
	Windows Solaris 50～1000 Linux 10～1000	
trace_auto	yes yes yes, no	トレース情報の自動採取を有効にするかを指定します。 ・ yes: トレース情報の自動採取を有効とする。(注2) ・ no : トレース情報の自動採取を無効とする。
Linux trace_buffer	process process process, system	内部トレースを採取する単位を指定します。 ・ process: プロセス単位で内部トレースを採取する。 ・ system: イベントサービス単位で内部トレースを採取する。

注1)

Windows Solaris

トレース情報の出力サイズはチャネル数、コンシューマ数、サプライヤ数、および通信頻度によって異なります。起動処理系、通信処理系、および停止処理系で使用するトレース情報のバッファサイズを以下に記載します。

Linux

プロセス単位で内部トレースを採取する(trace_buffer = process)場合、トレース情報の出力サイズはチャネル数、コンシューマ数、サプライヤ数、および通信頻度によって異なります。起動処理系、通信処理系、および停止処理系で使用するトレース情報のバッファサイズを以下に記載します。
なお、イベントサービス単位で内部トレースを採取する(trace_buffer = system)場合は、起動処理系、通信処理系、および停止処理系で使用するトレース情報を1つのバッファに格納されるため、それぞれを加算してください。

- 起動処理系
  イベントチャネル起動: 3.2KB
  サプライヤ起動処理(pushメソッドを出すまで): 1.0KB
  コンシューマ起動処理(pullメソッドを出すまで): 1.0KB

- 通信処理系
  pushメソッド: 0.8KB
  pullメソッド(受信成功): 1.2KB
  pullメソッド(COMM_FAILURE[minor=0x464a09c1]): 1.0KB

- 停止処理系
  イベントチャネル停止: 3.4KB
  サプライヤdisconnect処理: 0.5KB
  コンシューマdisconnect処理: 0.8KB

トレース情報バッファサイズを初期設定で運用した場合の計算例を以下に示します。

【1チャネルで、コンシューマ数:サプライヤ数が1:1の場合】

1回の通信(push/pull)で2.0KB(0.8KB＋1.2KB)のバッファサイズが必要となります。
トレース情報バッファは、バッファを半分ずつサイクリックに使用するため、トレース情報バッファ(サイズ:1024KB)に格納できる通信のトレース情報数は256回となります。
(トレース情報バッファサイズ ÷ 2) ÷ 1回の通信に必要なバッファサイズ ＝
(1024KB ÷ 2) ÷ 2.0KB ＝ 256回

40秒に1回の通信を行うと仮定した場合、約2.8時間の通信をロギングできることになります。
256 × 40秒 ＝ 10240秒 ＝ 約2.8時間
上記の例では、トレース情報を自動採取する事象が発生するまでの約2.8時間分のトレース情報を採取することができます。

トレース情報バッファサイズは、少なくとも5分以上のトレース情報が採取可能なサイズを指定してください。
トレース情報バッファサイズを初期値から変更した場合、その増分だけ共用メモリ使用量が増加します(キロバイト単位)。

注2)
トレース情報の自動採取を有効とする(trace_auto = yes)場合、トレースファイルは以下のファイル名で出力されます。(XXX:3桁の10進数の数値)

Windows

C:¥Interstage¥eswin¥var¥ESLOGXXX

Solaris

/var/opt/FJSVes/ESLOGXXX

Linux

プロセス単位で内部トレースを採取する(trace_buffer = process)場合

ー イベントサービスのデーモンプロセスのログ情報
/var/opt/FJSVes/ESLOGDUMPDAEMONXXX

ー イベントファクトリプロセスのログ情報
/var/opt/FJSVes/ESLOGDUMPFACTORYXXX

ー 静的イベントチャネルプロセスのログ情報
/var/opt/FJSVes/ESLOGDUMPグループ名XXX

ー 動的イベントチャネルプロセスのログ情報
/var/opt/FJSVes/ESLOGDUMPインプリメンテーション名XXX

イベントサービス単位で内部トレースを採取する(trace_buffer = system)場合
/var/opt/FJSVes/ESLOGXXX

D.2 イベントチャネル・サプライヤ・コンシューマ総数の見積もり方法

不揮発チャネル運用時に、1つのユニットで使用可能なイベントチャネル、サプライヤ、およびコンシューマの総数は、以下の見積もり式を参考にして見積もってください。

Windows

Σn(最大接続数 × 2)（注） ＋ 10(コマンド用の余裕値) ＜ ユニット定義ファイルで指定したtranmax

Solaris Linux

Σn(最大接続数 ＋ 16)（注） ＋ 10(コマンド用の余裕値) ＜ ユニット定義ファイルで指定したtranmax

n: イベントチャネルグループの総数

最大接続数: イベントチャネルに接続するサプライヤおよびコンシューマの合計値(esmkchnlコマンドで設定した-mオプションの指定値。省略値は16。)

注)本値が“256”よりも小さい場合は、“256”として計算してください。

付録E　Interstage HTTP Serverの環境定義

Interstage HTTP Serverの運用状態をチューニングするには、Interstage管理コンソールを使用して設定する方法と、Interstage HTTP Serverの環境定義ファイル(httpd.conf)を使用して設定する方法があります。

Interstage管理コンソールを使用して設定する場合、以下の手順で環境設定を行います。Interstage管理コンソールの起動については“運用ガイド（基本編）”を、Interstage管理コンソールの定義詳細についてはInterstage管理コンソールのヘルプを参照してください。

- Interstage管理コンソールのスタンドアロンサーバで環境設定を行う場合
  1. スタンドアロンサーバのInterstage管理コンソールにログインします。
  2. [システム] > [サービス] > [Webサーバ] > [Webサーバ名] > [環境設定]タブ > [Webサーバ:環境設定](詳細設定[表示])画面を使用して環境設定を行います。
- Interstage管理コンソールの管理サーバで環境設定を行う場合
  1. 管理サーバのInterstage管理コンソールにログインします。
  2. [一括操作] > [Interstage管理コンソール] > [Interstage Application Server] > [サービス] > [Webサーバ] > [FJapache(サーバグループ名またはサーバ名)] > [環境設定]タブ > [Webサーバ:環境設定](詳細設定[表示])画面を使用して環境設定を行います。

ここでは、環境定義ファイル(httpd.conf)の定義方法について説明します。
なお、Interstage HTTP Serverの環境定義ファイルは、以下に格納されています。

Windows (インストールパスはデフォルト)

```
C:¥Interstage¥F3FMihs¥servers¥(Webサーバ名)¥conf¥httpd.conf
```

Solaris (インストールパスはデフォルト)

```
/var/opt/FJSVihs/servers/(Webサーバ名)/conf/httpd.conf
```

Linux

```
/var/opt/FJSVihs/servers/(Webサーバ名)/conf/httpd.conf
```

E.1　タイムアウト時間

■タイムアウト時間の設定

タイムアウト時間を設定する場合、環境定義ファイル(httpd.conf)の以下のディレクティブを編集します。

Timeout クライアント送受信タイムアウト時間(秒)

クライアント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(秒)を指定します。待機時間には、0から65535までを指定できます。
Interstage HTTP Serverは、指定された時間に達してもパケットを受信できない場合、接続を閉じます。接続しているネットワークのトラフィックが増大し、接続が頻繁に中断される場合には、本時間を増やすことにより中断回数を減少させることができます。
クライアント送受信タイムアウト時間の初期値は“600”、省略値は“300”です。

SSLHandshakeTimeout SSLコネクション確立時の送受信タイムアウト時間(秒)

SSLコネクションの確立処理でクライアントからの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(秒)を設定します。待機時間には、0から65535までを指定できます(0を指定した場合、無制限)。

Interstage HTTP Serverは、指定された時間に達してもパケットを受信できない場合、接続を閉じます。通常、SSLコネクションの確立処理にかかる時間をチューニングしたい場合に設定します。

SSLコネクション確立時の送受信タイムアウト時間の初期値は、ありません。省略値は、Timeoutディレクティブの設定値です。

■HTTP Keep-Alive機能の設定

HTTP Keep-Alive機能を設定する場合、環境定義ファイル(httpd.conf)の以下のディレクティブを編集します。

KeepAlive On|Off

Interstage HTTP Serverでは、クライアント(Webブラウザ)との間で持続的な接続を維持できます。

“Off”を指定した場合は、1つの要求が完了するたびに接続を閉じて、次の要求に対して新しく接続しますが、“On”を指定することにより複数の要求を同じ接続で繰り返し使えるため、クライアントのレスポンスが向上します。

初期値・省略値は、“On”です。

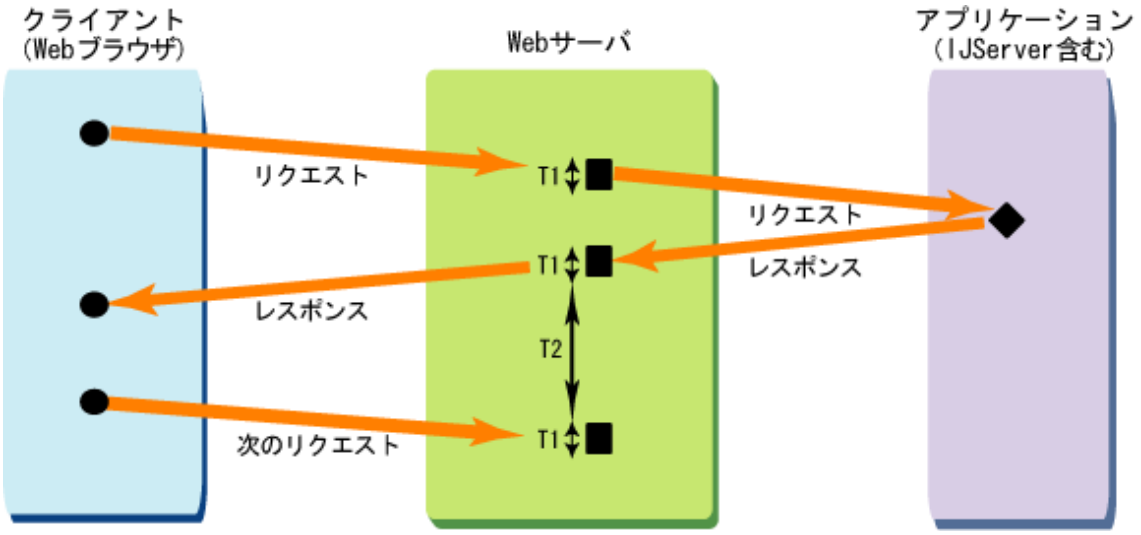
KeepAliveTimeout 次のリクエストまでのタイムアウト時間(秒)

クライアントの1つのリクエストが完了してから、コネクションを閉じずに次の新しいリクエストを待つ時間(秒)を指定します(“KeepAlive On”の場合のみ有効)。接続維持時間には、0から65535までを指定できます。この時間を経過しても次のリクエストがない場合は、接続を閉じます。

接続維持時間の初期値・省略値は、“15”です。

■タイムアウト時間の構成図

タイムアウト時間(TimeoutおよびKeepAliveTimeout)の構成図を以下に示します。



T1:Timeout(初期値 600秒)

クライアント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(秒)。

注) POSTまたはPUTリクエストにおいて、データを分割して複数回送受信される場合は、個々のデータの送受信時にかかる最長の時間となります。

T2:KeepAliveTimeout(初期値 15秒)

クライアントの1つのリクエストが完了してから、コネクションを閉じずに次の新しいリクエストを待つ時間(秒)。

E.2 クライアントの同時接続数

クライアントの同時接続数を設定する場合、環境定義ファイル(httpd.conf)の以下のディレクティブを編集します。
クライアントの同時接続数の構成図については、“Interstage HTTP Server 運用ガイド”の“Webサーバのプロセス構成”を参照してください。

ThreadsPerChild クライアントの同時接続数 Windows

Interstage HTTP Serverが、クライアント(Webブラウザ)からの接続要求を同時に受け付けることができる最大数を指定します。最大数は、1からThreadLimitディレクティブに指定した値までを指定できます。
この値を大きくすることにより同時アクセス可能な数は多くなりますが、メモリ資源や一時ファイルなどの消費に伴いシステム全体の性能が劣化する可能性があります。
同時アクセス最大数の初期値は“50”、省略値は“64”です。

ThreadLimit 通信スレッドの上限数 Windows

ThreadsPerChildディレクティブに設定するクライアント数の上限値を設定します。1から15000までを指定できます。1より小さな値を指定した場合は、1で動作します。15000より大きな値を指定した場合は、15000で動作します。
本ディレクティブは、ThreadsPerChildディレクティブに1920よりも大きな値に設定する必要がある場合にだけ使用してください。
通信スレッドの上限数の初期値は、ありません。省略値は“1920”です。

MaxClients クライアントの同時接続数 Solaris Linux

Interstage HTTP Serverが、クライアント(Webブラウザ)からの接続要求を同時に受け付けることができる最大数を指定します。最大数は、1からServerLimitディレクティブで指定した値までを指定できます。1より小さな値を指定した場合は、1で動作します。ServerLimitディレクティブで指定した値より大きな値を指定した場合は、ServerLimitディレクティブで指定した値で動作します。
この値を大きくすることにより同時アクセス可能な数は多くなりますが、メモリ資源や一時ファイルなどの消費に伴いシステム全体の性能が劣化する可能性があります。
クライアントの同時接続数の初期値は“50”、省略値は“256”です。
なお、この設定値を超過するクライアントの接続要求があった場合は、接続待ちキューに保存されます。接続待ちキュー数は、ListenBacklogディレクティブで設定します。

ServerLimit 通信プロセスの上限数 Solaris Linux

MaxClientsディレクティブに設定するクライアントの同時接続数の上限値を設定します。1から20000までを指定できます。1より小さな値を指定した場合は、1で動作します。20000より大きな値を指定した場合は、20000で動作します。
本ディレクティブは、MaxClientsディレクティブに256よりも大きな値に設定する必要がある場合にだけ使用してください。
通信プロセスの上限数の初期値は、ありません。省略値は“256”です。

ListenBacklog 接続待ちキューの最大数 Windows

ThreadsPerChildディレクティブで設定したクライアントの同時接続数よりも多くの接続要求があった場合に、オペレーティングシステム内にキューイングする数の最大数を設定します。最大数には、1から200までを指定できます。
接続待ちキューの最大数の省略値は、“200”です。

ListenBacklog 接続待ちキューの最大数 Solaris

MaxClientsディレクティブで設定したクライアントの同時接続数よりも多くの接続要求があった場合に、Solarisシステム内にキューイングする数の最大数を設定します。最大数には、1から2147483647までを指定できます。

本ディレクティブの設定値がSolarisシステムに設定されている待機中TCPコネクションの最大値（tcp_conn_req_max_q）よりも大きい場合は、待機中TCPコネクションの最大値が有効となります。

Solarisシステムに設定されている待機中TCPコネクションの最大値を確認する場合は、nddコマンドを使用して以下のように実行してください。

```
/usr/sbin/ndd  /dev/tcp  tcp_conn_req_max_q
```

なお、待機中TCPコネクションの設定およびnddコマンドの詳細については、Solarisシステムのドキュメントを参照してください。

接続待ちキューの最大数の省略値は、“511”です。ただし、待機中TCPコネクションの最大値よりも大きい場合は、待機中TCPコネクションの最大値が有効となります。

ListenBacklog 接続待ちキューの最大数 Linux

MaxClientsディレクティブで設定したクライアントの同時接続数よりも多くの接続要求があった場合に、Linuxシステム内にキューイングする数の最大数を設定します。最大数には、1から2147483647までを指定できます。

本ディレクティブの設定値がLinuxシステムに設定されている待機中TCPコネクションの最大値（/proc/sys/net/core/somaxconn）よりも大きい場合は、待機中TCPコネクションの最大値が有効となります。

Linuxシステムに設定されている待機中TCPコネクションの最大値を確認する場合は、sysctlコマンドを使用して以下のように実行してください。

```
/sbin/sysctl -n net.core.somaxconn
```

なお、待機中TCPコネクションの設定およびsysctlコマンドの詳細については、Linuxシステムのドキュメントを参照してください。

接続待ちキューの最大数の省略値は、“511”です。ただし、待機中TCPコネクションの最大値よりも大きい場合は、待機中TCPコネクションの最大値が有効となります。

付録F　Interstage シングル・サインオンの環境定義

Interstage シングル・サインオンを運用するための、環境定義のチューニングについて説明します。

F.1　1台のサーバにリポジトリサーバを構築する場合のチューニング

1台のサーバに、リポジトリサーバを構築する場合のチューニングについて説明します。

■Webサーバ(Interstage HTTP Server)のチューニング

リポジトリサーバのチューニングは、Webサーバ(Interstage HTTP Server)の環境定義を変更することにより行います。
詳細については、“付録E Interstage HTTP Serverの環境定義”を参照してください。

クライアントの同時接続数

以下のディレクティブに、想定する同時アクセス最大数以上を設定します。(初期値:50)

Windows

ThreadsPerChild

Solaris Linux

MaxClients

クライアント送受信タイムアウト時間

Timeoutにクライアント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(秒)を指定します。(初期値:600)

◆チューニングの例

例

想定する同時アクセス最大数が256ユーザのシステムの場合

Windows

Interstage HTTP Server
ThreadsPerChild＝256＋α(注1)
Timeout＝600(注2)

Solaris Linux

Interstage HTTP Server
MaxClients＝256＋α(注1)
Timeout＝600(注2)

注1) システムを安定稼動させるため、αには10～100までの値を設定してください。
注2) 接続しているネットワークのトラフィックが増大し、接続が頻繁に中断される場合には、本時間を増やしてください。

■TCP/IPパラメタのチューニング Windows

セション管理の運用を行う場合は、リポジトリサーバ(更新系)のマシンのTCP/IPパラメタのチューニングを行います。(注)

注) リポジトリサーバ(更新系)への同時アクセス最大数が増大し、セション管理サーバとの通信に失敗し、以下のメッセージが出力される場合があります。この場合には、TCP/IPパラメタのチューニングを行ってください。

- sso00114
- sso00119

F.2　1台のサーバに認証サーバを構築する場合のチューニング

1台のサーバに、認証サーバを構築する場合のチューニングについて説明します。

■Webサーバ(Interstage HTTP Server)のチューニング

認証サーバのチューニングは、Webサーバ(Interstage HTTP Server)の環境定義を変更することにより行います。
詳細については、“付録E Interstage HTTP Serverの環境定義”を参照してください。

クライアントの同時接続数

以下のディレクティブに、想定する同時アクセス最大数以上を設定します。(初期値:50)

Windows

ThreadsPerChild

Solaris Linux

MaxClients

クライアント送受信タイムアウト時間

Timeoutにクライアント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(秒)を指定します。(初期値:600)

◆チューニングの例

想定する同時アクセス最大数が256ユーザのシステムの場合

Windows

Interstage HTTP Server
ThreadsPerChild＝256＋α(注1)
Timeout＝600(注2)

Solaris Linux

Interstage HTTP Server
MaxClients＝256＋α(注1)
Timeout＝600(注2)

想定する同時アクセス最大数500ユーザを、5台の認証サーバにて負荷分散するシステムにおける認証サーバ1台あたりのチューニング

Windows

Interstage HTTP Server
ThreadsPerChild＝100＋α(注1)
Timeout＝600(注2)

Solaris Linux

Interstage HTTP Server
MaxClients＝100＋α(注1)
Timeout＝600(注2)

注1) システムを安定稼動させるため、αには10～100までの値を設定してください。
注2) 接続しているネットワークのトラフィックが増大し、接続が頻繁に中断される場合には、本時間を増やしてください。

F.3　1台のサーバにリポジトリサーバと認証サーバを構築する場合のチューニング

1台のサーバに、リポジトリサーバと認証サーバを構築する場合のチューニングについて説明します。

■Webサーバ(Interstage HTTP Server)のチューニング

リポジトリサーバ、および認証サーバのチューニングは、Webサーバ(Interstage HTTP Server)の環境定義を変更することにより行います。
詳細については、“付録E Interstage HTTP Serverの環境定義”を参照してください。

クライアントの同時接続数

以下のディレクティブに、想定する同時アクセス最大数×2以上を設定します。(初期値:50)

Windows

ThreadsPerChild

Solaris Linux

MaxClients

クライアント送受信タイムアウト時間

Timeoutにクライアント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(秒)を指定します。(初期値:600)

◆チューニングの例


例

想定する同時アクセス最大数が256ユーザのシステムの場合

Windows

Interstage HTTP Server
ThreadsPerChild＝256×2＋α(注1)
Timeout＝600(注2)

Solaris Linux

Interstage HTTP Server
MaxClients＝256×2＋α(注1)
Timeout＝600(注2)

注1) システムを安定稼動させるため、αには10～100までの値を設定してください。
注2) 接続しているネットワークのトラフィックが増大し、接続が頻繁に中断される場合には、本時間を増やしてください。

■TCP/IPパラメタのチューニング Windows

セション管理の運用を行う場合は、リポジトリサーバ(更新系)のマシンのTCP/IPパラメタのチューニングを行います。(注)

注) リポジトリサーバ(更新系)への同時アクセス最大数が増大し、セション管理サーバとの通信に失敗し、以下のメッセージが出力される場合があります。この場合には、TCP/IPパラメタのチューニングを行ってください。

- sso00114
- sso00119

F.4　業務サーバを構築する場合のチューニング

業務サーバを構築する場合のチューニングについて説明します。
キャッシュサイズ、キャッシュ数、および使用するWebサーバのチューニングを行ってください。

■キャッシュサイズ、キャッシュ数のチューニング

セションの管理を行う運用の場合、認証した利用者の認証情報を業務サーバでキャッシュすることで認可性能を向上させることができます。キャッシュを効果的に行うためにはシステムの同時利用者数、および利用者の認証情報サイズに応じて、キャッシュサイズ、キャッシュ数を適切に設定する必要があります。

キャッシュサイズ、キャッシュ数の設定については、業務サーバのInterstage管理コンソールを使用して、[システム] > [セキュリティ] > [シングル・サインオン] > [業務システム] > [業務システム名] > [環境設定] > [詳細設定[表示]]をクリックし、以下より行います。

- [認証情報のキャッシュ]の[キャッシュサイズ]
- [認証情報のキャッシュ]の[キャッシュ数]

キャッシュサイズ

認証情報のサイズを以下の計算式で見積り、Kバイト単位で設定します。

認証情報サイズ = (150 + DNの文字列長
+ ユーザIDの文字列長
+ 認証方式の文字列長
+ ロールサイズ(注1)
+ 拡張ユーザ情報サイズ(注2)) × 1.4バイト

注1)設定するロール名の文字列長の総和 + 10×ロール数
注2)設定する属性名の文字列長の総和 + 属性値の文字列長の総和 + 10×拡張ユーザ情報数

キャッシュ数

認証情報のキャッシュは、利用者の最終アクセス時からアイドル監視時間が経過するまで保持します。アイドル監視時間内で想定する同時アクセス最大数 + α(注)を設定してください。

注)1人の利用者が、アイドル監視時間内にサインオン、サインオフを繰り返した場合でも、キャッシュ数を消費するため、想定する同時アクセス最大数よりも少し大きめの値を設定してください。

設定したキャッシュサイズ、キャッシュ数を超える利用があった場合も継続して利用可能ですが、システムのログにsso03062、もしくはsso03063のメッセージが出力されます。
認可性能が低下する可能性があるため、メッセージに従い対処してください。

◆チューニングの例

利用者の情報が以下の場合を例に説明します。

項目	文字列長	値
DN	55	cn=Fujitsu Tarou,ou=User,ou=interstage,o=fujitsu,dc=com
ユーザID	5	tarou
認証方式	19	basicAuthOrCertAuth
ロール名	5	Admin
ロール名	6	Leader
拡張ユーザ情報(属性名)	4	mail
拡張ユーザ情報(属性値)	20	tarou@jp.fujitsu.com
拡張ユーザ情報(属性名)	14	employeeNumber

拡張ユーザ情報(属性値)	6	100001

ロールサイズ

2つのロールが設定されるため、以下のようになります。
(Admin(5文字) + Leader(6文字)) + 10×ロール数(2個) = 31

拡張ユーザ情報サイズ

2つの属性が設定されるため、以下のようになります。
(mail(4文字) + employeeNumber(14文字)) + (tarou@jp.fujitsu.com(20文字) + 100001(6文字)) + 10×拡張ユーザ情報数(2個) = 64

上記より、認証情報サイズは以下のようになります。

認証情報サイズ = (150 + DNの文字列長(55文字)
+ ユーザIDの文字列長(5文字)
+ 認証方式の文字列長(19文字)
+ ロールサイズ(31文字)
+ 拡張ユーザ情報サイズ(64文字)) × 1.4バイト
= 約454バイト

[キャッシュサイズ]には、上記認証情報サイズを切り上げ、1Kバイトを設定します。

■Webサーバ(Interstage HTTP Server)のチューニング

詳細については、“付録E Interstage HTTP Serverの環境定義”を参照してください。

クライアントの同時接続数

以下のディレクティブに、想定する同時アクセス最大数以上を設定します。(初期値:50)

Windows

ThreadsPerChild

Solaris Linux

MaxClients

クライアント送受信タイムアウト時間

Timeoutにクライアント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(秒)を指定します。(初期値:600)

■Microsoft(R) Internet Information Servicesのチューニング Windows

Microsoft(R) Internet Information Servicesのチューニングについては、“Microsoft(R) Internet Information Services”のプロパティ情報に以下の定義を設定することによりチューニングを行います。

最大接続数

最大接続数フィールドに想定する同時アクセス数以上を設定します。(default:1,000)

付録G　マルチサーバ管理の環境定義

マルチサーバ管理の環境定義ファイルのチューニングについて説明します。

G.1　マルチサーバ管理定義ファイル

■概要

マルチサーバ管理定義ファイルは、マルチサーバ管理の動作環境に関する定義が格納されたファイルです。

■ファイル名

Windows

%IS_HOME%¥jmx¥etc¥ssv.xml

Solaris　Linux

/etc/opt/FJSVisjmx/ssv.xml

■ファイル形式

<ssv>をルートタグとするXML形式。

■タグ・属性一覧

タグ名:site

属性名	内容	デフォルト値	値の範囲
server.limit	サイトに所属する管理対象サーバ数の上限	100	1～INT_MAX (注)
servergroup.limit	サイトに所属するサーバグループ数の上限	100	1～INT_MAX (注)
server.limit.inservergroup	サーバグループに所属する管理対象サーバの上限	100	1～INT_MAX (注)

注) マルチサーバ管理定義ファイルに記述された値が範囲内に納まっていない場合、デフォルト値で動作します。

タグ名:ijserver

属性名	内容	デフォルト値	値の範囲
deploy.path	配備対象アーカイブファイルの一時格納ディレクトリ	Windows %IS_HOME%¥jmx¥var¥ssv_ijs Solaris　Linux /etc/opt/FJSVisjmx/var/ssv_ijs	―

付録H　Portable-ORBの環境設定

Portable-ORBの動作環境ファイルは、porbeditenvコマンドを使用して設定します。Portable-ORBを使用する場合は、porbeditenvコマンドを使用して以下の環境設定を行ってください。

- 動作環境情報
- ホスト情報
- セキュリティ
- ネットワーク環境

porbeditenvコマンドの詳細については、“リファレンスマニュアル(コマンド編)”の“porbeditenv”を参照してください。Portable-ORBの動作環境ファイルの詳細については、“アプリケーション作成ガイド(CORBAサービス編)”の“Portable-ORB動作環境ファイルの指定”を参照してください。

また、運用環境を構築した場合は、万が一に備えて、資源のバックアップを行うことを推奨します。資源のバックアップについては、“運用ガイド（基本編）”の“メンテナンス(資源のバックアップ)”を参照してください。

索 引