Sun GlassFish Enterprise Server v3 管理ガイド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 821–1299–10
2009 年 12 月

100118@23031

目次

図目次

表目次

例目次

はじめに

『Sun GlassFish Enterprise Server v3 管理ガイド』では、Sun GlassFish Enterprise Server の設定方法と管理方法について説明します。

ここでは、Sun GlassFish™ Enterprise Server (Enterprise Server) のドキュメントセット全体に関する情報と表記規則について説明しています。

Enterprise Server v3 は、`https://glassfish.dev.java.net/` にある GlassFish プロジェクトのオープンソースコミュニティーを経て作成されたものです。この GlassFish プロジェクトは、Enterprise Server プラットフォームを開発するための構造化されたプロセスを提供して、Java EE のもっとも重要な特徴である互換性を保ちながら、これまでになく短期間で Java EE プラットフォームの新機能を実現できるようにするものです。Java 開発者が Enterprise Server のソースコードにアクセスして Enterprise Server の開発に参加できるようになっています。この GlassFish プロジェクトは、Sun の技術者とコミュニティーのコミュニケーションを助けるように設計されています。

ここでは、次のテーマを取り上げます。

Enterprise Server のドキュメントセット

Enterprise Server のマニュアルセットは、配備の計画とシステムのインストールについて説明しています。Enterprise Server ドキュメントの URL (Uniform Resource Locator) は、`http://docs.sun.com/coll/1343.9` です。Enterprise Server への導入としては、次の表に示されている順序でドキュメントを参照してください。

表 P–1　Enterprise Serverのドキュメントセットの内容

ドキュメント名	説明
『Release Notes』	ソフトウェアとマニュアルに関する最新情報を提供します。サポートされているハードウェア、オペレーティングシステム、Java™DevelopmentKit(JDK™)、およびデータベースドライバの包括的な表ベースの概要を含みます。
『Quick Start Guide』	Enterprise Server 製品の使用を開始するための手順を説明します。
『Installation Guide』	ソフトウェアおよびそのコンポーネントのインストール方法を説明します。
『Upgrade Guide』	Enterprise Server の最新バージョンにアップグレードする方法を説明します。このガイドでは、直前の製品リリースとの違いと、製品仕様との互換性がなくなる可能性のある構成オプションについても説明します。
『Administration Guide』	Enterprise Server サブシステムおよびコンポーネントを、`asadmin(1M)` ユーティリティを使用してコマンド行から設定、監視、管理する方法を説明します。これらのタスクを管理コンソール から実行する方法の説明は、管理コンソール オンラインヘルプで提供します。
『Application Deployment Guide』	Enterprise Server にアプリケーションを組み込んで配備する方法について説明し、配備記述子についての情報を提供します。
Your First Cup: An Introduction to the Java EE Platform	Java EE の初級プログラマ向けの簡単なチュートリアルです。シンプルなエンタープライズアプリケーションを開発する際の、全体的なプロセスを説明します。サンプルアプリケーションは、Enterprise JavaBeans™ 仕様に基づくコンポーネント、JAX-RS Web サービス、および Web フロントエンドの JavaServer™ Faces コンポーネントで構成される Web アプリケーションです。
『Application Development Guide』	Enterprise Server 上で実行する Java Platform, Enterprise Edition (Java EE プラットフォーム) アプリケーションの作成と実装の方法を説明します。これらのアプリケーションは、Java EE コンポーネントおよび API の Java オープンスタンダードモデルに準拠します。このガイドでは、開発者向けツール、セキュリティー、デバッグについての情報を提供します。
『 Add-On Component Development Guide』	Enterprise Server の公開されたインタフェースを使用して、Enterprise Server 向けのアドオンコンポーネントを開発する方法を説明します。このドキュメントでは 確実に Enterprise Server に適したアドオンコンポーネントとするためのタスクの実行方法のみを説明します。
『Embedded Server Guide』	組み込みの Enterprise Server でアプリケーションを実行する方法と、Enterprise Server を組み込むアプリケーションを開発する方法を説明します。

表 P–1 Enterprise Serverのドキュメントセットの内容 (続き)

ドキュメント名	説明
『Scripting Framework Guide』	Ruby on Rails や Groovy on Grails などの言語で、Enterprise Server への配備用のスクリプトアプリケーションを開発する方法を説明します。
『Troubleshooting Guide』	Enterprise Server の使用中に発生する可能性がある一般的な問題と、その解決方法を説明します。
『Error Message Reference』	Enterprise Server の使用中に表示されるエラーメッセージについて説明します。
『 Reference Manual』	Enterprise Server 管理コマンド、ユーティリティコマンド、および関連の概念についてのリファレンス情報をマニュアルページ形式で提供します。
『Domain File Format Reference』	Enterprise Server 構成ファイルである `domain.xml` の形式について説明します。
Java EE 6 Tutorial, Volume I	Java EE 6 プラットフォームテクノロジと API を使用した Java EE アプリケーションの開発方法を説明します。
『Message Queue Release Notes』	Sun GlassFish Message Queue の新機能、互換性の問題、および既存のバグについて説明します。
『Message Queue Administration Guide』	Sun GlassFish Message Queue メッセージングシステムを設定および管理する方法を説明します。
『Message Queue Developer's Guide for JMX Clients』	Sun GlassFish Message Queue のアプリケーションプログラミングインタフェースを使用して、JMX (Java Management Extensions) に従って Message Queue リソースをプログラムによって設定および監視する方法を説明します。
『 System Virtualization Support in Sun Java System Products』	Sun Java System 製品をシステム仮想化製品および機能とともに使用する場合の、Sun サポートの概要を説明します。

関連マニュアル

『Java EE 6 Tutorial, Volume II』 (`https://www.sun.com/offers/details/java_ee6_tutorial.xml`)には、Java EE 6 Tutorial, Volume Iのすべてのトピックが記載されているほか、高度なトピック、追加技術、およびケーススタディーが追加されています。このドキュメントは、Enterprise Server の登録ユーザーであればご使用になれます。

&ProductName に付属するパッケージ用の Javadoc™ ツールのリファレンスは、以下から入手できます。

- Java EE のバージョン 6 用 API 仕様書 (`http://java.sun.com/javaee/6/docs/api/`)。

- &ProductName 製品専用パッケージ用 API ドキュメント (http://javadoc.glassfish.org/v3/apidoc/)。

さらに、次のリソースが役立つことがあります。

- Java EE の仕様 (http://java.sun.com/javaee/technologies/index.jsp)
- Java EE Blueprints (http://java.sun.com/reference/blueprints/index.html)

NetBeans™ 統合開発環境 (IDE) でのエンタープライズアプリケーション開発については、http://www.netbeans.org/kb/60/index.html を参照してください。

Enterprise Server で使用する Java DB の詳細は、http://developers.sun.com/javadb/ を参照してください。

サンプルアプリケーションは、広範な Java EE テクノロジの実例を示したものです。このサンプルは Java EE Software Development Kit (SDK) に付属しています。

表記上の規則

このドキュメントでは、次のような書体を特別な意味を持つものとして使用します。

表 P–2 表記上の規則

書体	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力を示します。	.login ファイルを編集します。 ls -a を使用して、すべてのファイルを表示します。 machine_name% you have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	machine_name% **su** Password:
AaBbCc123	実際に使用する名前または値で置き換えられるプレースホルダを示します。	ファイルを削除するには、rm *filename* と入力します。
AaBbCc123	書名、新出用語、強調する用語を示します (一部の強調された項目はボールドで表示されます)。	『ユーザーズガイド』の第 6 章を参照してください。 *cache* は、ローカルに保存されたコピーです。 ファイルを保存しないでください。

記号の表記ルール

この表は、このドキュメントで使用される記号について説明したものです。

表 P–3 記号の表記ルール

記号	説明	例	意味
`[ ]`	省略可能な引数やコマンドオプションが含まれます。	`ls [-l]`	`-l` オプションは必須ではありません。
`{ | }`	必須コマンドオプションの選択項目が含まれています。	`-d {y|n}`	`-d` オプションには、`y` 引数または `n` 引数のいずれかを使用する必要があります。
`${ }`	変数参照を示します。	`${com.sun.javaRoot}`	`com.sun.javaRoot` 変数の値を参照します。
-	同時に実行する複数のキーストロークを結び付けます。	Control-A	コントロールキーを押しながら A キーを押します。
+	連続で複数のキーストロークを行います。	Ctrl + A + N	Control キーを押して離してから、次のキーを押します。
→	グラフィカルユーザーインタフェースでのメニュー項目の選択を示します。	「ファイル」→「新規」→「テンプレート」	「ファイル」メニューから「新規」を選択します。「新規」サブメニューから、「テンプレート」を選択します。

デフォルトのパスおよびファイル名

次の表は、このドキュメントで使用されているデフォルトのパス名とファイル名について説明したものです。

表 P–4 デフォルトのパスおよびファイル名

プレースホルダ	説明	デフォルト値
as-install	Enterprise Server のベースインストールディレクトリを表します。 構成ファイルでは、*as-install* は次のように表されます。 `${com.sun.aas.installRoot}`	Solaris™ オペレーティングシステム、Linux オペレーティングシステム、および Mac オペレーティングシステムへのインストールの場合: ユーザーのホームディレクトリ`/glassfishv3/glassfish` Windows のすべてのインストールの場合: システムドライブ`:\glassfishv3\glassfish`

表 P–4 デフォルトのパスおよびファイル名 (続き)

プレースホルダ	説明	デフォルト値
as-install-parent	Enterprise Server のベースインストールディレクトリを表します。	Solaris オペレーティングシステム、Linux オペレーティングシステム、および Mac オペレーティングシステムへのインストールの場合: ユーザーのホームディレクトリ/glassfishv3 Windows のすべてのインストールの場合: システムドライブ:\glassfishv3
domain-root-dir	デフォルトによるドメインの作成先ディレクトリを表します。	*as-install*/domains/
domain-dir	ドメインの構成保存先ディレクトリを表します。 構成ファイルでは、*domain-dir* は次のように表されます。 ${com.sun.aas.instanceRoot}	*domain-root-dir*/ドメイン名

ドキュメント、サポート、およびトレーニング

Sun の Web サイトには、次に示す関連情報が示されています。

- ドキュメント (http://www.sun.com/documentation/)
- サポート (http://www.sun.com/support/)
- トレーニング (http://www.sun.com/training/)

Sun 製品資料の検索

Sun 製品マニュアルは docs.sun.com℠ Web サイトで検索できるだけでなく、検索エンジンの検索フィールドに次の構文を入力することによっても検索できます。

search-term site:docs.sun.com

たとえば、「ブローカ」を検索する場合は、次のように入力します。

broker site:docs.sun.com

Sun のその他の Web サイト (たとえば、java.sun.com、www.sun.com、developers.sun.com) も検索対象とする場合は、検索フィールドで docs.sun.com の代わりに sun.com を使用します。

第三者の Web サイト参照

このドキュメント内で参照している第三者の URL は、追加の関連情報を提供します。

注 – このドキュメント内で引用する第三者の Web サイトの可用性について Sun は責任を負いません。こうしたサイトやリソース上の、またはこれらを通じて利用可能な、コンテンツ、広告、製品、その他の素材について、Sun は推奨しているわけではなく、Sun はいかなる責任も負いません。こうしたサイトやリソース上の、またはこれらを経由して利用可能な、コンテンツ、製品、サービスを利用または信頼したことに伴って発生した、あるいは発生したと主張されるいかなる損害や損失についても、Sun は一切の責任を負いません。

このマニュアルに関するコメント

弊社では、ドキュメントの改善に努め、お客様からのコメントおよびごをお受けしております。`http://docs.sun.com` にアクセスして「コメントの送信」をクリックしてください。オンラインフォームに、マニュアルの完全なタイトルと Part No. を記入してください。Part No. は 7 桁または 9 桁の数字で、マニュアルの表紙またはドキュメントの URL にあります。たとえば、このマニュアルの Part No は 821-1299-10 です。

◆ ◆ ◆ 第 1 章

Enterprise Server の管理の概要

Sun GlassFish™ Enterprise Server v3 は、Java™ アプリケーションと Web サービスの開発および配備用の環境を提供します。

Enterprise Server の管理者の主な業務は、Enterprise Serverの環境のセキュリティー保護の確立、および環境に関与するサービス、リソース、およびユーザーの監督です。主な作業として、リソースとサービスの構成、実行時の Enterprise Server の管理、サーバーに関する問題の修正などがあります。また、ソフトウェアのインストール、アドオンコンポーネントの統合、およびアプリケーションの配備を行う場合もあります。

ここでは、次のテーマを取り上げます。

デフォルトの設定と場所

インストール後、インストール機能を目的に合わせるために一部の構成タスクを即時に実行する必要がある場合があります。構成のデフォルトをそのまま使用した場合、有効になる機能と有効にならない機能があります。Enterprise Server のサービスやリソースの初期構成タスクの概要については、35 ページの「初期構成タスク」を参照してください。

さらに、デフォルトのパスワードのリセット、ファイルの名前や場所の変更などを行うこともあります。次の表に、デフォルトの管理値を示します。

注 – Enterprise Server v3 の zip バンドルの場合、デフォルトの管理用ログイン名は `admin` で、パスワードはありません。これは、ログインが不要なことを意味します。

表 1–1 デフォルトの管理値

項目	デフォルト
ドメイン名	`domain1`
マスターパスワード	`changeit`
管理用パスワード	`admin`
管理用サーバーポート	`4848`
HTTP ポート	`8080`
HTTPS ポート	`8181`
Pure JMX Clients ポート	`8686`
Message Queue ポート	`7676`
IIOP ポート	`3700`
IIOP/SSL ポート	`3820`
相互認証を使用する IIOP/SSL ポート	`3920`

表 1–2 デフォルトの場所

項目	デフォルト
コマンド行ユーティリティー (`asadmin`)	*as-install*`/bin`
構成ファイル	*domain-dir*`/config`
ログファイル	*domain-dir*`/logs`
アップグレードツール (`asupgrade` コマンド)	*as-install*`/bin`
アップグレードツールと `pkg` コマンド	*as-install-parent*`/bin`

置き換え可能な項目とデフォルトのパスとファイルの詳細については、29 ページの「デフォルトのパスおよびファイル名」を参照してください。

構成タスク

Enterprise Server の環境を意図どおりに動作させるために、インストールの直後に一部の構成タスクを実行する必要があります。たとえば、Enterprise Server とともにデータベースを使用する場合は、データベースとの接続をただちに設定する必要があります。

構成の状況の中には、進行中であるためインストールの期間中に何度も変更が必要になるものがあります。構成を変更するには、管理コンソール または `asadmin` ユーティリティーを使用できます。変更内容は、適切な構成ファイルに自動的に適用されます。

ここでは、次のテーマを取り上げます。

初期構成タスク

この節では、一般的な構成タスクと、このガイドのコマンド行手順との対応を示します。構成の状況によっては、リソースとサービスが自動的に有効になるため、構成タスクの一環として、独自のニーズに合わせたデフォルト設定の調整や変更が必要になる場合があります。

次に示すリソースとサービスは、インストールの直後に構成が必要になることがよくあります。

システムプロパティー
: 57 ページの「システムプロパティーの管理」を参照してください。

ドメイン
: インストール時に初期の `domain1` が作成されます。追加の構成タスクには、追加ドメインの構成や自動再起動の設定などのタスクが含まれることがあります。第 3 章「ドメインの管理」を参照してください。

JVM
: JVM の構成の初期タスクとして、JVM のオプションおよびプロファイラの作成があります。第 4 章「Java プラットフォームの仮想マシンの管理」を参照してください。

ログ
: デフォルトではログが有効になるので、追加構成なしで基本ログが機能します。ただし、ログレベル、プロパティーの値、またはログファイルの場所を変更することがあります。第 7 章「ロギングサービスの管理」を参照してください。

監視
デフォルトでは、監視サービスが有効になります。ただし、個々のモジュールの監視は有効にならないので、最初のタスクは、対象のモジュールについて監視を有効にすることです。第 8 章「監視サービスの管理」を参照してください。

ライフサイクルモジュール
第 9 章「ライフサイクルモジュールの管理」を参照してください。

セキュリティー

- 「システムセキュリティー」。初期構成タスクとして、パスワード、監査モジュール、および認証の設定が含まれることがあります。第 11 章「システムセキュリティーの管理」を参照してください。
- 「ユーザーセキュリティー」。初期構成タスクとして、認証レルムとファイルユーザーの作成が含まれることがあります。第 12 章「ユーザーセキュリティーの管理」を参照してください。
- 「メッセージセキュリティー」。初期構成タスクとして、Java Cryptography Extension (JCE) プロバイダの構成、デフォルトおよびそれ以外のセキュリティープロバイダの有効化、およびメッセージ保護ポリシーの構成が含まれることがあります。第 13 章「メッセージセキュリティーの管理」を参照してください。

データベースの接続
Enterprise Server と Java DB データベースとの接続を構成する初期タスクには、Java Database Connectivity (JDBC) 接続プールの作成、JDBC リソースの作成、JDBC ドライバの統合が含まれます。第 14 章「データベース接続の管理 」を参照してください。

EIS の接続
Enterprise Server とエンタープライズ情報システム (EIS) との接続を構成する初期タスクには、コネクタ接続プールの作成、コネクタリソースの作成、リソースアダプタ構成の編集、コネクタのセキュリティーマップの作成、コネクタ動作のセキュリティーマップの作成、および管理オブジェクトの作成 (必要な場合) が含まれます。第 15 章「EIS 接続の管理」を参照してください。

インターネット接続
配備した Web アプリケーションをインターネットクライアントからアクセス可能にする初期タスクには、HTTP ネットワークリスナーと仮想サーバーの作成、および SSL 用 HTTP リスナーの構成 (必要な場合) が含まれます。第 16 章「インターネット接続の管理」を参照してください。

Object Request Broker (ORB)
初期構成タスクで、IIOP リスナーを作成することがあります。第 17 章「ORB (Object Request Broker) の管理」を参照してください。

JavaMail サービス
初期構成タスクで、JavaMail リソースを作成することがあります。第 18 章「JavaMail サービスの管理」を参照してください。

Java Message Service (JMS)
初期構成タスクには、物理送信先の作成、接続ファクトリまたは宛先リソースの作成、JMS ホストの作成 (デフォルトの JMS ホストが適切でない場合)、接続プール設定の調整 (必要な場合)、および JMS 用のリソースアダプタの構成が含まれることがあります。第 19 章「JMS (Java Message Service) の管理 」 を参照してください。

JNDI サービス
初期構成タスクで、JNDI リソースを作成することがあります。第 20 章「Java Naming and Directory Interface (JNDI) サービスの管理」 を参照してください。

管理コンソール を使用してこれらのタスクを実行する方法は、管理コンソール のオンラインヘルプを参照してください。

構成でのドット表記名の機能

初期構成の適用後、Enterprise Server の稼働期間に使用中の構成を継続して管理します。生産性を向上させるためにリソースの調整が必要になったり、問題が発生して設定の変更やデフォルトへのリセットなどが必要になることがあります。`update-connector-work-security-map` サブコマンドのように、更新用の `asadmin` のサブコマンドが用意されている場合があります。しかし、たいていの更新は、`list`、`get`、および `set` のサブコマンドとドット表記名を使用して行われます。ドット表記名の詳細については、`dotted-names(5ASC)` のマニュアルページを参照してください。

注 – ドット表記名は監視にも使用しますが、メソッドが異なります。監視でのドット表記名の使用方法については、130 ページの「監視のツリー構造の仕組みについて」 を参照してください。

コマンド行で構成変更を操作する一般的なプロセスは、次のとおりです。

1. 目的のコンポーネントのモジュールを一覧表示します。

   次のシングルモードの例は、| (パイプ) 文字と `grep` コマンドを使用して、検索を絞り込んでいます。

   ```
   asadmin list "*" | grep http | grep listener
   ```

   次のような情報が返されます。

```
configs.config.server-config.network-config.network-listeners.network-listener.http-listener-1
configs.config.server-config.network-config.network-listeners.network-listener.http-listener-2
configs.config.server-config.network-config.protocols.protocol.admin-listener.http
configs.config.server-config.network-config.protocols.protocol.admin-listener.http.file-cache
configs.config.server-config.network-config.protocols.protocol.http-listener-1
configs.config.server-config.network-config.protocols.protocol.http-listener-1.http
```

```
configs.config.server-config.network-config.protocols.protocol.http-listener-1.http.file-cache
configs.config.server-config.network-config.protocols.protocol.http-listener-2
configs.config.server-config.network-config.protocols.protocol.http-listener-2.http
configs.config.server-config.network-config.protocols.protocol.http-listener-2.http.file-cache
configs.config.server-config.network-config.protocols.protocol.http-listener-2.ssl
```

2. 目的のモジュールに適用する属性を取得します。

   次のマルチモードの例は、`http-listener-1` の属性と値を取得します。

```
asadmin> get server-config.network-config.network-listeners.network-listener.http-listener-1.*
```

   次のような情報が返されます。

```
server.http-service.http-listener.http-listener-1.acceptor-threads = 1
server.http-service.http-listener.http-listener-1.address = 0.0.0.0
server.http-service.http-listener.http-listener-1.blocking-enabled = false
server.http-service.http-listener.http-listener-1.default-virtual-server = server
server.http-service.http-listener.http-listener-1.enabled = true
server.http-service.http-listener.http-listener-1.external-port =
server.http-service.http-listener.http-listener-1.family = inet
server.http-service.http-listener.http-listener-1.id = http-listener-1
server.http-service.http-listener.http-listener-1.port = 8080
server.http-service.http-listener.http-listener-1.redirect-port =
server.http-service.http-listener.http-listener-1.security-enabled = false
server.http-service.http-listener.http-listener-1.server-name =
server.http-service.http-listener.http-listener-1.xpowered-by = true
```

3. サブコマンド `set` を使用して、属性を変更します。

   この例は、`http-listener-1` の属性 `security-enabled` を true に設定します。

```
asadmin> set server.http-service.http-listener.http-listener-1.security-enabled = true
```

構成ファイル

Enterprise Server のリソース、アプリケーション、およびサーバーインスタンスの構成情報の多くは、構成ファイル `domain.xml` に保存されます。このファイルは、指定管理ドメインの中央リポジトリで、Enterprise Server のドメインモデルの XML 表現を含みます。domain.xml ファイルのデフォルトの場所は、*as-install* `/glassfish3/glassfish/domains/`*domain-name*`/config` です。`domain.xml` ファイルの詳細については、『Sun GlassFish Enterprise Server v3 Domain File Format Reference』を参照してください。

`logging.properties` ファイルは、各モジュールのログレベルの構成に使用されます。このファイルの場所は、`domain.xml` ファイルと同じディレクトリです。`logging.properties` ファイルの詳細については、124 ページの「ログレベルの設定」を参照してください。

asenv.conf ファイルの場所は、*as-install* /glassfishv3/glassfish/config です。このファイルの目的は、データベースのインストール場所、Message Queue など、GlassFish に固有の環境変数を保存することです。

注 – 変更内容は、適切な構成ファイルに自動的に適用されます。構成ファイルを直接編集しないでください。手動の編集はエラーを発生しやすく、予期しない結果となることがあります。

構成の変更の影響

変更した構成を有効にする際、Enterprise Server の再起動が必要な場合があります。Enterprise Server を再起動しなくとも、動的に変更内容が適用される場合もあります。このガイドの手順では、サーバーを再起動する必要がある場合について説明します。

- 39 ページの「サーバーの再起動が必要な構成の変更」
- 40 ページの「動的な構成の変更」

サーバーの再起動が必要な構成の変更

次の構成を変更したときには、変更内容を有効にするためにサーバーを再起動する必要があります。

- JVM オプションの変更
- ポート番号の変更
- ログハンドラ要素の変更
- 認証の構成
- HTTP、JMS、IIOP、JNDI サービスの管理
- リソースの作成または削除 (例外: 一部の JDBC、JMS、またはコネクタリソースは再起動が不要)
- 次の JDBC 接続プールプロパティーの変更
  - datasource-classname
  - associate-with-thread
  - lazy-connection-association
  - lazy-connection-enlistment
  - JDBC ドライバのベンダー固有のプロパティー
- 次のコネクタ接続プールプロパティーの変更
  - resource-adapter-name
  - connection-definition-name
  - transaction-support
  - associate-with-thread

- `lazy-connection-association`
- `lazy-connection-enlistment`
- ベンダー固有のプロパティー

動的な構成の変更

「動的な構成」によって、サーバーの動作中に変更内容が有効になります。次の構成を変更する際、サーバーを再起動する必要はありません。

- アドオンコンポーネントの追加または削除
- JDBC、JMS、コネクタのリソースおよびプールの追加または削除 (例外: 一部の接続プールプロパティーは再起動が必要)
- ファイルレルムユーザーの追加
- ログレベルの変更
- 監視の有効化と無効化
- モジュールの監視レベルの変更
- リソースとアプリケーションの有効化と無効化
- アプリケーションの配備、配備解除、および再配備

管理ツール

グラフィカルな管理コンソールと `asadmin` コマンド行ユーティリティーのどちらを使用しても、ほとんどの場合同じタスクを実行できますが、例外もあります。

この節では次の Enterprise Server の管理ツールについて説明します。

- 40 ページの「管理コンソール」
- 41 ページの「`asadmin` ユーティリティー」
- 42 ページの「REST インタフェース」
- 42 ページの「更新ツール」
- 43 ページの「OSGi モジュール管理サブシステム」
- 45 ページの「`keytool` ユーティリティー」
- 45 ページの「Java Monitoring and Management Console (JConsole)」
- 45 ページの「Application Server Management Extension (AMX)」

管理コンソール

管理コンソールはブラウザベースのユーティリティーで、ナビゲーションが容易なグラフィカルインタフェースを特長とし、管理タスク向けの広範なオンラインヘルプを用意しています。

管理コンソール を使用するには、ドメイン管理サーバー (DAS) が稼働中である必要があります。·各ドメインには、一意のポート番号を持つ DAS がそれぞれ必要です。DAS のポート番号は、Enterprise Server をインストールしたときに選択しています。選択していない場合は、デフォルトポートの 4848 を使用しています。デフォルトのログイン情報 (ユーザー名 admin、パスワードなし) をそのまま使用していない場合は、ユーザー名とパスワードも指定しています。.

管理コンソール の URL を指定する場合は、ドメインの管理ポート番号を指定します。Web ブラウザで 管理コンソール を開始するための形式は、`http://`*hostname*`:`*port* です。次に例を示します。

```
http://kindness.sun.com:4848
```

Enterprise Server をインストールしたマシンで 管理コンソール を実行する場合は、ホスト名として `localhost` を指定します。次に例を示します。

```
http://localhost:4848
```

Microsoft Windows では「スタート」メニューから Enterprise Server の 管理コンソール を開始することもできます。

管理コンソール のページのヘルプ内容を表示するには、そのページの「ヘルプ」ボタンをクリックします。最初のマニュアルページには、ページ自体の機能とフィールドの説明があります。追加ページにある関連タスクの操作方法にアクセスするには、「関連項目」リストのリンクをクリックします。

asadmin ユーティリティー

`asadmin` ユーティリティーはコマンド行ツールであり、実行する操作またはタスクを指定するサブコマンドを実行します。`asadmin` のサブコマンドは、コマンドプロンプト、またはスクリプトから実行できます。`asadmin` のサブコマンドをスクリプトから実行すると、反復したタスクを自動化するのに役立ちます。`asadmin` ユーティリティーの機能の基本情報については、asadmin(1M) のマニュアルページを参照してください。`asadmin` ユーティリティーの使用方法については、49 ページの「`asadmin` ユーティリティーの使用」を参照してください。

`asadmin` のサブコマンドを標準コマンドシェル (シングルモード) で実行するには、&InstallDir/`bin` ディレクトリに移動し、`asadmin` コマンドのあとにサブコマンドを入力します。次に例を示します。

```
asadmin list-jdbc-resources
```

マルチコマンドモード (マルチモード) を起動するには、コマンドプロンプトに `asadmin` を入力します。その後、プロンプト `asadmin>` が表示されます。マルチモードを終了して標準コマンドシェルに戻るまで、`asadmin` ユーティリティーは継続してサブコマンドを受け取ります。次に例を示します。

```
asadmin> list-jdbc-resources
```

asadmin のサブコマンドのマニュアルページを表示するには、help の後にサブコマンド名を入力します。次に例を示します。

```
asadmin> help restart-domain
```

または

```
asadmin help restart-domain
```

asadmin のマニュアルページを集めたものが、『Sun GlassFish Enterprise Server v3 Reference Manual』です。HTML および PDF の形式で用意されています。

REST インタフェース

Enterprise Server は、新規にインストールされたアドオンコンポーネントが提供するデータなど、Enterprise Server の監視データや構成データにアクセスできる表現状態転送 (REST) インタフェースを装備しています。詳細については、65 ページの「REST インタフェースによる Enterprise Server の管理」を参照してください。

更新ツール

Enterprise Server は、配備した Enterprise Server のソフトウェアを更新する Image Packaging System (IPS) のツールセットを装備しています。一般的な更新には、Enterprise Server の新規リリース、および Enterprise Server のアドオンコンポーネントまたはモジュールの新規リリースまたは改訂リリースが含まれます。

- 更新ツールのグラフィカルユーティリティーは、管理コンソール で実行することも、コマンド updatetool を使用してコマンド行から呼び出すこともできます。いずれかのツールを使用してコンポーネントを追加できます。ただし、既存のコンポーネントの更新や削除には、スタンドアロンバージョンを使用する必要があります。更新ツールのグラフィカルバージョンの使用方法は、管理コンソール のオンライン ヘルプ、およびスタンドアロンの更新ツールのオンラインヘルプに記載されています。
- コマンド pkg は、更新ツールのコマンド行バージョンです。アドオンコンポーネントとともに pkg コマンドを使用する方法は、第 10 章「Enterprise Server の拡張」に記載されています。

GlassFish Enterprise Server v3 では、Web Profile と Full Platform Profile の 2 つの配布がサポートされています。インストール後、システムのモジュールを表示するには、グラフィカルな更新ツール、または pkg コマンドを使用します。

注 – Web Profile を選択した後で、Full Platform Profile に変更するには、更新ツールの対応する Full Platform Profile パッケージを選択します。従属するモジュールがすべて自動的に追加されます。

配布について各モジュールの追加や削除は可能ですが、そのような構成はサポートされていません。

Enterprise Server の新バージョンを操作するために、ドメイン構成データのアップグレードについての情報が必要な場合は、『Sun GlassFish Enterprise Server v3 Upgrade Guide』を参照してください。

OSGi モジュール管理サブシステム

Enterprise Server に付属の OSGi モジュール管理サブシステムは、Apache Felix OSGi フレームワークです。このフレームワークを管理できるように、Enterprise Server ではデフォルトで Apache Felix Remote Shell (`http://felix.apache.org/site/apache-felix-remote-shell.html`) が有効 なっています。このシェルは Felix シェルサービスを使用して OSGi モジュール管理サブシステムと相互作用し、次のような管理タスクを実行可能にします。

- インストール済みの OSGi バンドルを参照する
- インストール済みの OSGi バンドルのヘッダーを表示する
- OSGi バンドルをインストールする
- インストールしたバンドルのライフサイクルを管理する

Apache Felix Remote Shell は、ネットワークのどこからでも telnet クライアントにアクセスできます。telnet サービスを使用して Apache Felix Remote Shell に接続するには、コマンド `telnet(1)` を次のように使用します。

`telnet` *host felix-remote-shell-port*

host
DAS が稼働中のホスト

felix-remote-shell-port
telnet サービスを使用して Apache Felix Remote Shell に接続するポート。Enterprise Server は、このためにポート 6666 を使用するように事前構成されています。

Apache Felix Remote Shell で使用できるコマンドを一覧表示するには、Apache Felix Remote Shell のプロンプトに `help` と入力します。

Apache Felix Remote Shell を終了するには、Apache Felix Remote Shell のプロンプトに `exit` と入力します。

例 1–1 Apache Felix Remote Shell に接続する

この例では、ローカルホストで動作し、telnet サービスによる Apache Felix Remote Shell への接続に事前構成されたポートを使用しているドメインで Apache Felix Remote Shell に接続します。

```
telnet localhost 6666
```

接続の確立後、次の情報が表示されます。

```
Connected to localhost.
Escape character is '^]'.

Felix Remote Shell Console:
============================

->
```

例 1–2 インストール済みの OSGi バンドルを一覧表示する

この例は、引数を指定せずに Felix Remote Shell のコマンド ps を実行して、インストール済みの OSGi バンドルを一覧表示します。見やすくするために、この例で表示される可能性のある一部のバンドルを記載していません。

```
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (2.0.2)
[   1] [Active     ] [    1] HK2 OSGi Main Bundle (1.0.0)
[   2] [Installed  ] [    1] AMX V3 Core (3.0.0.SNAPSHOT)
[   3] [Active     ] [    1] GlassFish Rest Interface (3.0.0.SNAPSHOT)
...
[ 217] [Installed  ] [    1] Admin Console JDBC Plugin (3.0.0.SNAPSHOT)
[ 218] [Resolved   ] [    1] stats77 (3.0.0.SNAPSHOT)
[ 219] [Active     ] [    1] Apache Felix Declarative Services (1.0.8)
[ 220] [Active     ] [    1] GlassFish Web Container (rfc #66) for OSGi Enabled
Web Applications (3.0.0.SNAPSHOT)
->
```

例 1–3 指定の名前を持つ OSGi バンドルを検索する

この例は、Felix Remote Shell のコマンド find を実行して、名前にテキスト rfc が含まれる OSGi バンドルを検索します。..

```
-> find rfc
START LEVEL 1
   ID   State         Level  Name
```

例 1–3 指定の名前を持つOSGiバンドルを検索する (続き)

```
[ 220] [Active     ] [    1] GlassFish Web Container (rfc #66) for OSGi Enabled
Web Applications (3.0.0.SNAPSHOT)
->
```

例 1–4 OSGiバンドルが提供するサービスを調べる

この例は、オプション `service` と `capability` を指定して Felix Remote Shell のコマンド `inspect` を実行し、OSGi バンドル 220 が提供するサービスを調べます。

```
-> inspect service capability 220
GlassFish Web Container (rfc #66) for OSGi Enabled Web Applications (220) provides services:
---------------------------------------------------------------------------------------
objectClass = org.glassfish.osgiweb.Extender
service.id = 30
----
objectClass = org.osgi.service.url.URLStreamHandlerService
service.id = 31
url.handler.protocol = webbundle
->
```

keytool ユーティリティー

`keytool` ユーティリティーは、Java Security Socket Extension (JSSE) デジタル証明書の設定と操作に使用します。`keytool` の使用方法については、210 ページの「JSSE 証明書の管理」を参照してください。

Java Monitoring and Management Console (JConsole)

Java SE は、MBean サーバーに接続してそのサーバーに登録された MBean を表示するツールを装備しています。JConsole は一般的な JMX コネクタクライアントであり、標準 Java SE ディストリビューションの一部として利用できます。Enterprise Server 環境に JConsole を実装する方法については、171 ページの「Enterprise Server の監視データを表示するための JConsole の設定」を参照してください。

Application Server Management Extension (AMX)

アプリケーションサーバー管理拡張 (AMX) API は、AMX インタフェースの使いやすいクライアント側の動的プロキシ実装として、Enterprise Server の構成、および監視中の JMX 管理対象 Bean をすべて公開します。

Enterprise Server の管理方法

コマンド行から多くの管理タスクを実行する方法とその情報は、このドキュメント、および asadmin ユーティリティーのマニュアルページに記載されています。asadmin のオンラインヘルプへのアクセス方法については、53 ページの「asadmin ユーティリティーまたはサブコマンドのヘルプ情報を表示する」を参照してください。

管理コンソール を使用してこれらのタスクを実行する方法は、管理コンソール のオンラインヘルプを参照してください。

注 – Enterprise Server のツール用に作成された命令は、コマンド内のディレクトリパスやファイル名の区切り文字として、標準 UNIX® のスラッシュ (/) を使用します。Microsoft Windows システムで Enterprise Server を実行している場合は、代わりに円記号 (\) を使用します。次に例を示します。

- UNIX: *as-install*`/bin/asadmin`
- Windows: *as-install*`\bin\asadmin`

次に示す追加ドキュメントで、特定の管理領域について説明しています。

- Enterprise Server ソフトウェアのインストール、更新ツールによるアドオンコンポーネントの更新

  『Sun GlassFish Enterprise Server v3 Installation Guide 』
- アプリケーションの確認と配備

  『Sun GlassFish Enterprise Server v3 Application Deployment Guide 』
- 問題の診断と解決

  『Sun GlassFish Enterprise Server v3 Troubleshooting Guide 』

パート I

ランタイム管理

◆ ◆ ◆ 第 2 章

管理の概要

この章では、asadmin コマンド行ユーティリティーを使用して、&ProductBrand™; Enterprise Server v3 の環境で一般的な管理タスクを実行する方法について説明します。

ここでは、次のテーマを取り上げます。

- 49 ページの「asadmin ユーティリティーの使用」
- 57 ページの「システムプロパティーの管理」
- 59 ページの「リソースの管理」
- 60 ページの「さまざまなシステム要素の一覧表示」
- 65 ページの「REST インタフェースによる Enterprise Server の管理」

管理コンソール を使用してこの章のタスクを実行する方法は、管理コンソール のオンライン ヘルプに記載されています。

asadmin ユーティリティーの使用

Sun GlassFish Enterprise Server の管理タスクを実行するには、コマンド行またはスクリプトから asadmin ユーティリティーを使用します。このユーティリティーは、管理コンソール インタフェースの代わりに使用できます。

ここでは、次のテーマを取り上げます。

- 50 ページの「asadmin ユーティリティーのパス」
- 50 ページの「asadmin ユーティリティーの構文」
- 52 ページの「シングルモードで asadmin ユーティリティーのサブコマンドを実行する」
- 53 ページの「asadmin ユーティリティーまたはサブコマンドのヘルプ情報を表示する」
- 54 ページの「マルチモードセッションを開始する」
- 55 ページの「マルチモードセッションを終了する」

- 56 ページの「ファイルから asadmin の一連のサブコマンドを実行する」

asadmin ユーティリティーのパス

asadmin ユーティリティは、*as-install* /bin ディレクトリにあります。パスを指定せずに asadmin ユーティリティーを実行するには、このディレクトリをパスに入れてください。

asadmin ユーティリティーの構文

asadmin ユーティリティーを実行するための構文は次のとおりです。

asadmin [*asadmin-util-options*] [*subcommand* [*subcommand-options*] [*operands*]]

この構文で置き換え可能な項目については、以降の項で説明します。この構文の詳細については、asadmin(1M) のマニュアルページを参照してください。

asadmin ユーティリティーのサブコマンド

次の「サブコマンド」は、実行する動作またはタスクを指定します。サブコマンドは大文字と小文字を区別します。サブコマンドは、ローカルサブコマンドとリモートサブコマンドのいずれかです。

- 「ローカルサブコマンド」は、ドメイン管理サーバー (DAS) を動作させずに実行できます。ただし、サブコマンドを実行してインストールディレクトリやドメインディレクトリにアクセスするには、ドメインのホストマシンにログインする必要があります。
- 「リモートコマンド」は常に、DAS に接続して DAS でサブコマンドを実行することにより実行されます。稼働中の DAS が必要です。

このリリースの Enterprise Server のサブコマンドリストについては、『Sun GlassFish Enterprise Server v3 Reference Manual』の第 1 節を参照してください。

asadmin ユーティリティーのオプションとサブコマンドのオプション

オプションは、asadmin ユーティリティーとそのサブコマンドの動作を制御します。オプションは大文字と小文字を区別します。

asadmin ユーティリティーには、次のタイプのオプションがあります。

- asadmin ユーティリティーのオプション。これらのオプションは、asadmin ユーティリティーの動作を制御します。サブコマンドの動作は制御しません。asadmin ユーティリティーのオプションは、サブコマンドの前後に指定できますが、サブコマンドの後に asadmin ユーティリティーのオプションを指定することは推奨しません。asadminユーティリティーのオプションは、サブコマンドの前、または後にすべて指定する必要があります。asadmin ユーティリティーのオプションをサブコマンドの前と後の両方に指定した場合、エラーが発生します。asadmin ユーティリティーのオプションの詳細については、asadmin(1M) のマニュアルページを参照してください。
- 「サブコマンドオプション」。これらのオプションは、サブコマンドの動作を制御します。asadmin ユーティリティーの動作は制御しません。サブコマンドオプションは、サブコマンドの後に指定する必要があります。サブコマンドのオプションの詳細については、『Sun GlassFish Enterprise Server v3 Reference Manual』のサブコマンドの項目を参照してください。

注 – サブコマンドオプションの一部は、このリリースの Enterprise Server ではサポートされていません。サポートされていないオプションを指定しても、構文エラーは発生しません。その代わりコマンドが正常に実行され、未サポートのオプションは単に無視されます。

サブコマンドオプションの名前が、asadmin ユーティリティーのオプションの名前と同じ場合がありますが、2 つのオプションの効果は異なります。

オプションには、長形式と省略形式があります。

- オプションの省略形式では、ダッシュ 1 つ (-) の後に文字 1 つが続きます。
- オプションの長形式では、ダッシュ 2 つ (--) の後にオプションのワードが続きます。

たとえば、terse 出力を指定するオプションの省略形式と長形式は次のとおりです。

- 省略形式: -t
- 長形式: --terse

論理型オプションを除く多くのオプションには、機能の有効と無効を切り替える引数の値が必要です。

asadmin ユーティリティーのサブコマンドのオペランド

オペランドは、サブコマンドの動作対象となる項目を指定します。オペランドは、サブコマンドオプションの引数の後に指定する必要があり、空白 1 文字、タブ 1 文字、またはダッシュ 2 文字 (--) で区切ります。asadmin ユーティリティーは、サブコマンドオプションとその値に続くものはすべて、オペランドとして扱います。

▼ シングルモードで asadmin ユーティリティーのサブコマンドを実行する

シングルモードでは、使用するサブコマンドについて個々に asadmin コマンドを入力する必要があります。サブコマンドの実行後は、オペレーティングシステムのコマンドシェルに戻ります。asadmin ユーティリティーのオプションは、実行する asadmin コマンドについて個々に指定する必要があります。複数のサブコマンドについて同じ asadmin ユーティリティーのオプションが必要な場合は、マルチモードで asadmin ユーティリティーを使用します。詳細については、54 ページの「マルチモードセッションを開始する」を参照してください。

- オペレーティングシステムのコマンドシェルで、asadmin ユーティリティーを実行してサブコマンドを指定します。

  必要に応じて、必須の asadmin ユーティリティーのオプション、サブコマンドのオプション、およびオペランドも指定します。

例 2–1 シングルモードでの asadmin ユーティリティーのサブコマンド実行

この例は、シングルモードでサブコマンド list-applications(1) を実行します。この例では、すべてのオプションでデフォルト値を使用します。

この例は、ローカルホストでアプリケーション hello が配備されていることを示します。

```
asadmin list-applications
hello <web>

Command list-applications executed successfully.
```

例 2–2 シングルモードでの asadmin ユーティリティーオプションとサブコマンドの指定

この例は、シングルモードで asadmin ユーティリティーのオプション --host とサブコマンド list-applications を指定します。この例では、DAS はホスト srvr1.example.com で稼働中です。

この例は、アプリケーション basic-ezcomp、scrumtoys、ejb31-war、および automatic-timer-ejb がホスト srvr1.example.com に配備されていることを示します。

```
asadmin --host srvr1.example.com list-applications
basic-ezcomp <web>
scrumtoys <web>
ejb31-war <ejb, web>
```

```
automatic-timer-ejb <ejb>

Command list-applications executed successfully.
```

例 2–3 シングルモードでの asadmin ユーティリティーオプションとサブコマンドオプションの指定

この例は、シングルモードで asadmin ユーティリティーオプション --host、サブコマンドオプション --type、およびサブコマンド list-applications を指定します。この例では、DAS がホスト srvr1.example.com で稼働し、web タイプのアプリケーションが一覧表示されます。

```
asadmin --host srvr1.example.com list-applications --type web
basic-ezcomp <web>
scrumtoys <web>
ejb31-war <ejb, web>

Command list-applications executed successfully.
```

▼ asadmin ユーティリティーまたはサブコマンドのヘルプ情報を表示する

Enterprise Server には、asadmin ユーティリティーとそのサブコマンドの構文、目的、およびオプションに関するヘルプ情報が用意されています。このヘルプ情報は、UNIX® プラットフォームのマニュアルページのスタイルで作成されています。このヘルプ情報は、『Sun GlassFish Enterprise Server v3 Reference Manual』にも記載されています。

1. リモートサブコマンドのヘルプ情報を表示する場合は、サーバーが稼働中であることを確認してください。
   リモートサブコマンドを実行するには、稼働中のサーバーが必要です。
2. 対象のサブコマンドを、help サブコマンドのオペランドとして指定します。
   オペランドを指定せずに help サブコマンドを実行した場合、asadmin ユーティリティーのヘルプ情報が表示されます。

例 2–4 asadmin ユーティリティーのヘルプ情報の表示

この例は、asadmin ユーティリティーのヘルプ情報を表示します。

```
asadmin help
```

例 2–5 asadmin ユーティリティーサブコマンドのヘルプ情報表示

この例は、サブコマンド create-jdbc-resource のヘルプ情報を表示します。

```
asadmin help create-jdbc-resource
```

参照 使用できるサブコマンドを表示するには、サブコマンド list-commands(1) を使用します。ローカルサブコマンドが、リモートサブコマンドの前に表示されます。サーバーが稼働していない場合は、ローカルサブコマンドのみが表示されます。

▼ マルチモードセッションを開始する

asadmin ユーティリティーは、複数コマンドモード、つまりマルチモードで使用できます。マルチモードでは、asadmin ユーティリティーを一度実行してマルチモードセッションを開始します。セッションを終了してオペレーティングシステムのコマンドシェルに戻るまでの間、asadmin ユーティリティーは継続してサブコマンドを受け取ります。マルチモードセッションで設定した asadmin ユーティリティーのオプションは、セッションの後続のサブコマンドすべてについて使用されます。

注 – マルチモードセッションを開始するには、稼働中の DAS は「不要」です。

- 次のいずれかの操作を行います。
  - サブコマンドを指定せずに asadmin ユーティリティーを実行します。
  - サブコマンド multimode(1) を使用します。

  必要に応じて、マルチモードセッション全体に適用する asadmin ユーティリティーのオプションも指定します。

  マルチモードセッションでは、コマンド行に asadmin> プロンプトが表示されます。このプロンプトに asadmin のサブコマンドを入力して Enterprise Server を管理できます。

例 2–6 asadmin ユーティリティーオプションによるマルチモードセッション開始

この例は、マルチモードセッションを開始し、asadmin ユーティリティーのオプション --user および --passwordfile をこのセッションに設定します。

```
asadmin --user admin1 --passwordfile pwd.txt multimode
```

例 2–7 サブコマンド multimode によるマルチモードセッションの開始

この例は、サブコマンド multimode を使用して、デフォルトの asadmin ユーティリティーのオプションを使用するマルチモードセッションを開始します。

```
asadmin multimode
```

コマンド行に asadmin> プロンプトが表示されます。

例 2–8 マルチモードセッションでのサブコマンドの実行

この例は、マルチモードセッションを開始し、そのセッションでサブコマンド list-domains を実行します。

```
asadmin
Enter commands one per "line", ^D to quit
asadmin> list-domains
Name: domain1 Status: Running
Command list-domains executed successfully.
asadmin>
```

参考 既存のマルチモードセッションからのマルチモードセッション開始

既存のセッションからマルチモードセッションを開始するには、既存のセッションからサブコマンド multimode を実行します。2 つ目のマルチモードセッションを終了すると、元のマルチモードセッションに戻ります。

参照 また、コマンド行に asadmin help multimode を入力して、サブコマンドの詳細構文とオプションも表示できます。

▼ マルチモードセッションを終了する

- asadmin> プロンプトに、次のコマンドまたはキーコンビネーションのいずれかを入力します。
  - **exit**
  - **quit**
  - UNIX および Linux システム: Ctrl-D
  - Windows システム: Ctrl-Z

  オペレーティングシステムのコマンドシェルに戻ります。asadmin> プロンプトは表示されなくなります。asadmin> プロンプトがまだ表示されている場合は、マルチモードセッションからマルチモードセッションを開いた可能性があります。この場合は、この手順を繰り返して残りのマルチモードセッションを終了します。

▼ ファイルから asadmin の一連のサブコマンドを実行する

ファイルから asadmin の一連のサブコマンドを実行すると、繰り返し実行するタスクを自動化できます。

1. 実行するサブコマンドのシーケンスを含むプレーンテキストファイルを作成します。
2. 作成したファイルを指定して、サブコマンド multimode(1) を実行します。
   必要に応じて、ファイル内のサブコマンドを実行可能にするために必要な asadmin ユーティリティーのオプションも指定します。

例 2–9 ファイルからの asadmin の一連のサブコマンドの実行

この例には、次のものが含まれます。

- asadmin のサブコマンドのシーケンスを含む、名前が commands_file.txt のファイルリスト
- ファイル commands_file.txt 内のサブコマンドを実行するためのコマンド

commands_file.txt ファイルには、次の動作シーケンスを実行するための asadmin ユーティリティーのサブコマンドがあります。

1. ドメイン customdomain の作成
2. ドメイン customdomain の開始
3. 使用できるすべてのサブコマンドの一覧表示
4. ドメイン customdomain の停止
5. ドメイン customdomain の削除

ファイル commands_file.txt の内容は次のとおりです。

```
create-domain --portbase 9000 customdomain
start-domain customdomain
list-commands
stop-domain customdomain
delete-domain customdomain
```

この例は、commands_file.txt ファイル内のサブコマンドのシーケンスを実行します。ファイル内のサブコマンド create-domain にオプション --portbase が指定されているので、asadmin ユーティリティーのオプション --port も設定する必要があります。

```
asadmin --port 9048 multimode --file commands_file.txt
```

参照　上の例のサブコマンドの詳細については、次のマニュアルページを参照してください。

- create-domain(1)
- delete-domain(1)
- list-commands(1)
- multimode(1)
- start-domain(1)
- stop-domain(1)

システムプロパティーの管理

共有サーバーインスタンスでは、参照される構成に定義された属性の上書きが頻繁に必要になります。任意の構成の属性を、対応する名前のシステムプロパティーによって上書きできます。

ここでは、次のテーマを取り上げます。

- 57 ページの「システムプロパティーを作成する」
- 58 ページの「システムプロパティーを一覧表示する」
- 58 ページの「システムプロパティーを削除する」

▼ システムプロパティーを作成する

ドメインまたは構成の 1 つ以上のシステムプロパティーの作成または更新を行うには、リモートモードでサブコマンド create-system-properties を使用します。 任意の構成の属性を、対応する名前のシステムプロパティーによって上書きできます。

1. サーバーが稼働中であることを確認してください。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. create-system-properties(1) サブコマンドを使用して、システムプロパティーを作成します。
   サブコマンドのプロパティーに関する情報が、このマニュアルページにあります。

例 2–10　システムプロパティーの作成

この例は、localhost の http-listener-port=1088 に関するシステムプロパティーを作成します。

```
asadmin> create-system-properties http-listener-port=1088
Command create-system-properties executed successfully.
```

参照　コマンド行に asadmin help create-system-properties と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ システムプロパティーを一覧表示する

ドメインまたは構成に適用するシステムプロパティーを一覧表示するには、リモートモードで list-system-properties サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-system-properties(1) サブコマンドを使用して、システムプロパティーを一覧表示します。
   HTTP_LISTENER_PORT や HTTP_SSL_LISTENER_PORT などの事前定義プロパティーを含む、既存のシステムプロパティーが表示されます。

例 2–11　システムプロパティーの一覧表示

この例は、ホスト localhost のシステムプロパティーを一覧表示します。

```
asadmin> list-system-properties
http-listener-port=1088
Command list-system-properties executed successfully.
```

参照　コマンド行に asadmin help list-system-properties と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ システムプロパティーを削除する

システムプロパティーを削除するには、リモートモードで delete-system-property サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-system-properties(1) サブコマンドを使用して、既存のシステムプロパティーを一覧表示します。
3. delete-system-property(1) サブコマンドを使用して、システムプロパティーを削除します。
4. 必要に応じて、システムプロパティーが削除されたことをユーザーに通知します。

例 2–12 システムプロパティーの削除

この例は、localhost から http-listener-port という名前のシステムプロパティーを削除します。

```
asadmin> delete-system-property http-listener-port
Command delete-system-property executed successfully.
```

参照 コマンド行に asadmin help delete-system-property と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

リソースの管理

この節では、Enterprise Server 環境にリソースを統合する方法を説明します。JDBC のような特定リソースの管理に関する情報は、他の章に記載されています。

ここでは、次のテーマを取り上げます。

- 59 ページの「XML ファイルからリソースを追加する」

▼ XML ファイルからリソースを追加する

指定の XML ファイル内に名前が指定されているリソースを作成するには、リモートモードで add-resources サブコマンドを使用します。サポートするリソースは、JDBC 接続プールおよびリソース、JMS、JNDI、および JavaMail のリソース、カスタムリソース、コネクタリソースおよび作業セキュリティーマップ、admin オブジェクト、およびリソースアダプタの構成です。

XML ファイルは、*as-install*/domains/domain1/config ディレクトリに配置される必要があります。相対パスを指定した場合、または単に XML ファイルの名前を指定した場合は、このサブコマンドのオペランドの前に *as-install*/domains/domain1/config が付けられます。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 add-resources(1) サブコマンドを使用して、XML ファイルからリソースを追加します。
このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3 Enterprise Server を再起動します。
94 ページの「ドメインの再起動」を参照してください。

例 2–13 リソースの追加

この例は、localhost の resource.xml の内容を使用して、リソースを作成します。

```
asadmin> add-resources c:\tmp\resource.xml
Command : JDBC resource jdbc1 created successfully.
Command : JDBC connection pool poolA created successfully.
Command add-resources executed successfully.
```

参照 コマンド行に asadmin help add-resources と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

さまざまなシステム要素の一覧表示

▼ Enterprise Server のバージョンを表示する

特定サーバーの Enterprise Server のバージョンに関する情報を表示するには、リモートモードで version サブコマンドを使用します。指定のログイン (ユーザーとパスワード) およびターゲット (ホストとポート) の情報を使用してサブコマンドがサーバーと通信できない場合は、ローカルバージョンと警告メッセージが表示されます

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 version(1) サブコマンドを使用してバージョンを表示します。

例 2–14 バージョン情報の表示

この例では、ローカルホストの Enterprise Server のバージョンを表示します。

```
asadmin> version
Version = GlassFish v3
Command version executed successfully.
```

参照　コマンド行に asadmin help version と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ アプリケーションを一覧表示する

配備した Java™ アプリケーションを一覧表示するには、リモートモードで list-applications サブコマンドを使用します。--type オプションを指定しない場合は、すべてのアプリケーションが一覧表示されます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-applications(1) サブコマンドを使用して、アプリケーションを一覧表示します。

例 2–15　アプリケーションの一覧表示

この例では、localhost の Web アプリケーションを一覧表示します。

```
asadmin> list-applications --type web
hellojsp <web>
Command list-applications executed successfully.
```

参照　コマンド行に asadmin help list-applications と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コンテナを一覧表示する

アプリケーションコンテナを一覧表示するには、リモートモードで list-containers サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-containers(1) サブコマンドを使用して、コンテナを一覧表示します。

例 2–16　コンテナの一覧表示

この例は、localhost のコンテナを一覧表示します。

```
asadmin> list-containers
List all known application containers
Container : grizzly
```

```
Container : ejb
Container : webservices
Container : ear
Container : appclient
Container : connector
Container : jpa
Container : web
Container : jruby
Container : security
Container : webbeans
Command list-containers executed successfully.
```

参照 コマンド行に `asadmin help list-containers` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ モジュールを一覧表示する

Enterprise Server のモジュールサブシステムにアクセスできるモジュールを一覧表示するには、リモートモードで `list-modules` サブコマンドを使用します。各モジュールの状態が含まれます。表示される状態には、NEW と READY があります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `list-modules(1)` サブコマンドを使用して、モジュールを一覧表示します。

例 2–17 モジュールの一覧表示

この例はアクセス可能なモジュールを一覧表示します。

```
asadmin> list-modules
```

次のような情報が表示されます (出力の一部を示す)。

```
List Of Modules
Module : org.glassfish.web.jstl-connector:10.0.0.b28
    properties=(visibility=public,State=READY,Sticky=true)
    Module Characteristics : List of Jars implementing the module
        Jar : file:/C:/Preview/v3_Preview_release/distributions/web/target/glass
fish/modules/web/jstl-connector.jar
    Module Characteristics : List of imported modules
    Module Characteristics : Provides to following services
Module : org.glassfish.admingui.console-common:10.0.0.b28
    properties=(visibility=public,State=NEW,Sticky=true)
```

```
Module : org.glassfish.admin.launcher:10.0.0.b28
    properties=(visibility=public,State=NEW,Sticky=true)
Module : org.glassfish.external.commons-codec-repackaged:10.0.0.b28
    properties=(visibility=public,State=NEW,Sticky=true)
Module : com.sun.enterprise.tiger-types-osgi:0.3.32.Preview-b28
    properties=(visibility=public,State=READY,Sticky=true)
    Module Characteristics : List of imported modules
    Module Characteristics : Provides to following services
    Module Characteristics : List of Jars implementing the module
        Jar : file:/C:/Preview/v3_Preview_release/distributions/web/target/glass
fish/modules/tiger-types-osgi.jar.
.
.
.
Command list-modules executed successfully.
```

参照　コマンド行に asadmin help list-modules と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ サブコマンドを一覧表示する

配備した asadmin のサブコマンドを一覧表示するには、リモートモードで list-commands サブコマンドを使用します。リモートサブコマンドのみ、またはローカルサブコマンドのみを一覧表示するように指定できます。このサブコマンドのデフォルトでは、ローカルサブコマンドのリストの後にリモートサブコマンドのリストが表示されます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-commands(1) サブコマンドを使用して、サブコマンドを一覧表示します。

例 2–18　サブコマンドの一覧表示

この例は、ローカルサブコマンドのみを一覧表示します。

```
asadmin> list-commands --localonly
create-domain
delete-domain
list-commands
list-domains
login
monitor
start-database
```

```
start-domain
stop-domain
stop-database
version
Command list-commands executed successfully.
```

参照　コマンド行に `asadmin help list-commands` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ タイマーを一覧表示する

タイマーサービスは、エンタープライズ Bean コンテナによって提供され、エンタープライズ Bean が使用する通知やイベントのスケジュールに使用される、持続的なトランザクション通知サービスです。ステートフルセッション Beans 以外のエンタープライズ Beans はすべて、タイマーサービスからの通知を受信できます。このサービスによって設定された持続タイマーは、サーバーのシャットダウンや再起動では破棄されません。

指定のサーバーインスタンスが所有する持続タイマーを一覧表示するには、リモートモードで `list-timers` サブコマンドを使用します。この情報を使用して、タイマーを移行するかどうかの決定、または移行が正常に完了したかどうかの検証ができます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-timers(1)` サブコマンドを使用して、タイマーを一覧表示します。

例 2–19　タイマーの一覧表示

この例は、特定のスタンドアロンサーバーインスタンスのタイマーを一覧表示します。現在、アクティブなタイマーが 1 つ設定されています。

```
asadmin> list-timers server
1
The list-timers command was executed successfully.
```

▼ コンポーネントの状態を表示する

指定した配備済みコンポーネントの状態 (有効または無効) を取得するには、リモートモードで `show-component-status` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 show-component-status(1) サブコマンドを使用して、コンポーネントの状態を表示します。

例 2–20 コンポーネントの状態の表示

この例は、MEjbApp コンポーネントの状態を表示します。

```
asadmin> show-component-status MEjbApp
Status of MEjbApp is enabled
Command show-component-status executed successfully.
```

REST インタフェースによる Enterprise Server の管理

Enterprise Server は、新規にインストールされたアドオンコンポーネントが提供するデータなど、Enterprise Server の監視データや構成データにアクセスできる表現状態転送 (REST) インタフェースを装備しています。

Enterprise Server の REST インタフェースには、次のようなクライアントアプリケーションからアクセスできます。

- Web ブラウザ
- cURL (http://curl.haxx.se/)
- GNU Wget (http://www.gnu.org/software/wget/)

また、Enterprise Server の REST インタフェースは、次のような言語で開発された REST クライアントアプリケーションでも使用できます。

- JavaScript
- Ruby
- Perl
- Java
- JavaFX

Enterprise Server の REST インタフェースの実装は、project Jersey (https://jersey.dev.java.net/) をベースにしています。Project Jersey は、Java ™ 仕様要求 (JSR) 311: JAX-RS、RESTful Web サービス用 Java API (http://jcp.org/en/jsr/summary?id=311) の参照実装です。JSR 311 に関する情報は、JSR 311 プロジェクトのホームページ (https://jsr311.dev.java.net/)にも記載されています。

ここでは、次のテーマを取り上げます。

- 66 ページの「REST URL による Enterprise Server の管理」
- 68 ページの「REST リソースメソッドによる Enterprise Server の管理」
- 78 ページの「CRUD 以外の操作用の子リソース」
- 79 ページの「REST インタフェースのセキュリティー保護」
- 79 ページの「リソースの表現形式」

REST URL による Enterprise Server の管理

構成および監視のオブジェクトツリーの各ノードは、HTTP uniform resource locator (URL) からアクセスできる REST リソースとして表現されます。Enterprise Server の監視データや構成データの REST リソースにアクセスするには、稼働中の DAS が必要です。

構成および監視のオブジェクトツリーのノードを表現するリソースの URL の形式は、次のとおりです。

- 構成: `http://` *host*`:`*port*`/management/domain/` *path*
- 監視: `http://` *host*`:`*port*`/monitoring/domain/` *path*

これらの URL で置き換え可能な項目は次のとおりです。

host
DAS が稼働中のホスト

port
管理用の HTTP ポートまたは HTTPS ポート

path
ノードへのパス。パスはノードのドット表記名で、各ドット (.) がスラッシュ (/) に置換されます。詳細については次のドキュメントを参照してください。

- `dotted-names(5ASC)` のマニュアルページ
- 130 ページの「監視のツリー構造の仕組みについて」
- 37 ページの「構成でのドット表記名の機能」
- 『Sun GlassFish Enterprise Server v3 Domain File Format Reference』の「Element Hierarchy」

Enterprise Server の監視データまたは構成データについて、REST リソースの URL を Web ブラウザで開いた場合、ブラウザには、リソースの次の情報を含む Web ページが表示されます。

- リソースの属性とその値のリストリソースが構成ツリーのノードを表現している場合は、それらの属性は、リソースの更新に使用できる HTML 形式で表示されます。監視ツリーにあるノードのリソースの属性は、読み取り専用です。
- リソースの子を示すハイパーテキストリンクのリストこのリンクのリストを使用して、リソースを含むツリー内を移動して、ツリー内のすべてのリソースを検出できます。

次の図に、ドメインを管理する REST リソースの Web ページを示します。

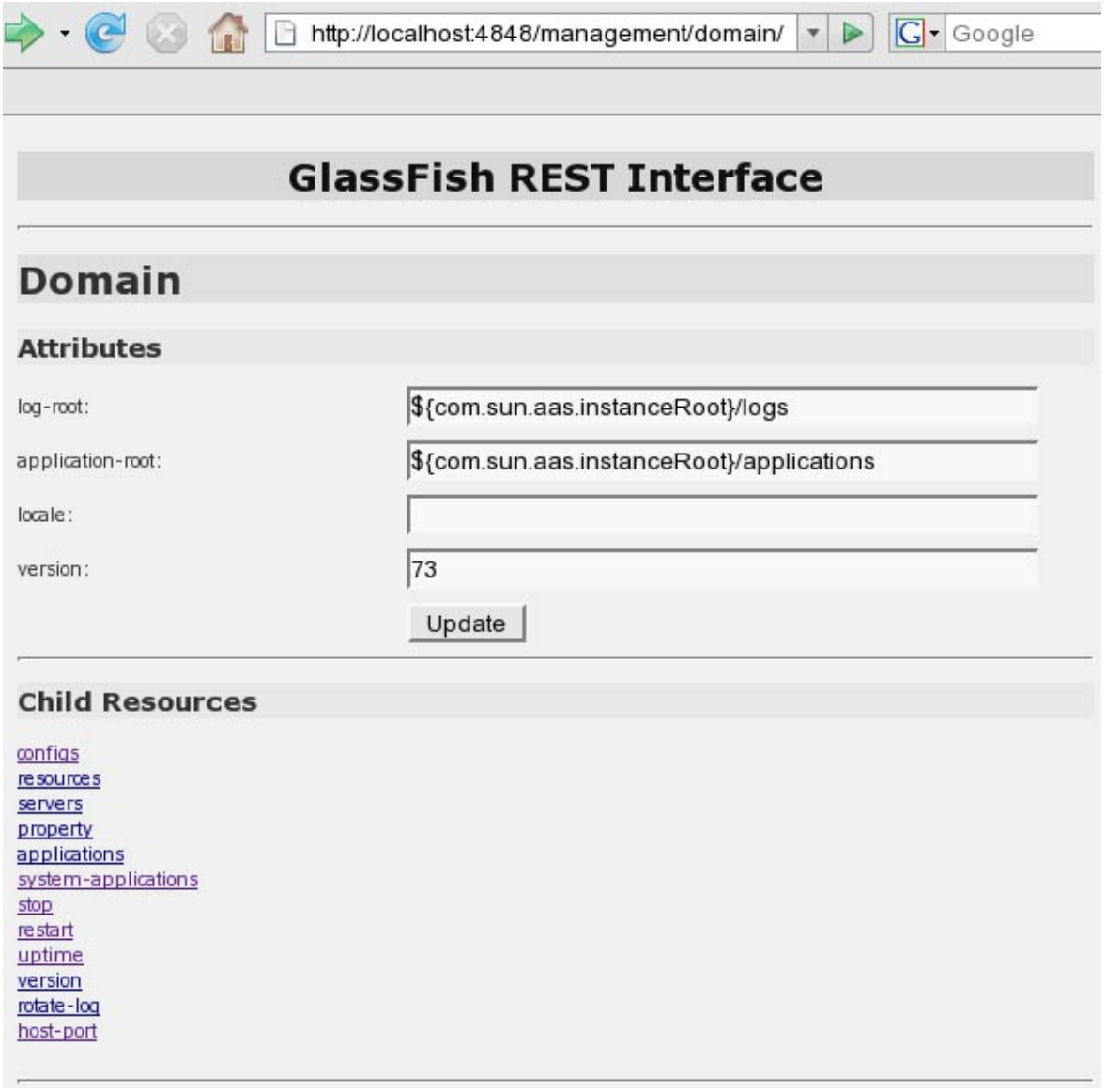

図 2–1 ドメインを管理する REST リソースの Web ページ

REST リソースメソッドによる Enterprise Server の管理

Enterprise Server の REST インタフェースは、監視および構成のオブジェクトツリーのノードにアクセスする方法をサポートしています。

次の表に、監視データや構成データを管理する REST メソッド、および各メソッドで実行できるタスクを示します。これらのメソッドは、HTTP 1.1 プリミティブです。これらのプリミティブの詳細仕様については、Hypertext Transfer Protocol -- HTTP/1.1 (`http://www.w3.org/Protocols/rfc2616/rfc2616.html`) を参照してください。

表 2–1 監視データと構成データを管理する REST リソースメソッド

作業	REST メソッド
ツリー内のノードがサポートするメソッドおよびメソッドパラメータの判定	`OPTIONS` または `GET`
ツリー内のノードのデータ取得	`GET`
ツリーへのノードの追加	`POST`
ツリー内のノードの更新	`POST`
ツリーからのノードの削除	`DELETE`

注 – `GET` メソッドを `OPTIONS` メソッドの代わりに使用して、ツリー内のノードがサポートするメソッドおよびメソッドパラメータを判定することができます。`GET` メソッドは、ノードの追加情報も表示します。詳細については、70 ページの「ツリー内のノードのデータを取得する」を参照してください。

▼ ツリー内のノードがサポートするメソッドおよびメソッドパラメータを判定する

ツリー内のノードがサポートするメソッドおよびメソッドパラメータは、ノードを表す REST リソースによって決まります。

- 監視用の REST リソースは、`GET` メソッドのみをサポートします。
- 構成用の REST リソースはすべて、`GET` メソッドおよび `OPTIONS` メソッドをサポートします。ただし、構成用の一部の REST リソースは、`POST` メソッドと `DELETE` メソッドもサポートします。

ツリーのノードで操作を実行する前に、そのノードがサポートするメソッドおよびメソッドパラメータを判定します。

この情報の表示形式を指定できます。詳細については、79 ページの「リソースの表現形式」を参照してください。

1 サーバーが実行されていることを確認します。

Enterprise Server のデータについて、REST リソースを操作するには、稼働中のサーバーが必要です。

2 ノードを表す **REST** リソースに対して、適切なメソッドを使用します。

- 監視オブジェクトツリーのノードの場合は、GET メソッドを使用します。
- 構成オブジェクトツリーのノードの場合は、 OPTIONS メソッドまたは GET メソッドを使用します。

GET メソッドと OPTIONS メソッドは、リソースがサポートするメソッドのリストを返します。各メソッドについて、使用できるメッセージパラメータのリスト、または使用できるクエリーパラメータのリストが返されます。

例 2–21 ツリー内のノードがサポートするメソッドおよびメソッドパラメータの判定

この例は、cURL ユーティリティーを使用して、ドメインのリソースがサポートするメソッドおよびメソッドパラメータを判定します。この例は、cURL ユーティリティーの次に示すオプションを使用します。

- -X: OPTIONS メソッドを使用していることを示します
- -H: リソースが JavaScript Object Notation (JSON) で記述されていることを示します

この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。OPTIONS メソッドに加えて、このリソースは POST メソッドと GET メソッドをサポートしています。

```
curl -X OPTIONS -H "Accept: application/json" http://localhost:4848/management/domain
{"Domain":
  {
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "log-root":{"Key":"false", "Type":"string", "Optional":"true"},
        "application-root":{"Key":"false", "Type":"string", "Optional":"true"},
        "locale":{"Key":"false", "Type":"string", "Optional":"true"},
        "version":{"Key":"false", "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
```

```
    }
  }
}
```

▼ ツリー内のノードのデータを取得する

ツリー内のノードのデータを取得すると、そのノードを表す REST リソースに関する次の情報が得られます。

- リソースがサポートする REST メソッドのリスト
- リソースの属性とその値のリスト
- リソースの子の URL のリスト

この情報の表示形式を指定できます。詳細については、79 ページの「リソースの表現形式」を参照してください。

1 サーバーが実行されていることを確認します。

Enterprise Server のデータについて、REST リソースを操作するには、稼働中のサーバーが必要です。

2 ノードを表す **REST** リソースに対して、GET メソッドを使用します。

例 2–22 ツリー内のノードのデータの取得

この例は、cURL ユーティリティーを使用して、ドメインのリソースデータを取得します。この例は、cURL ユーティリティーの次に示すオプションを使用します。

- -X: GET メソッドを使用していることを示します
- -H: リソースが JavaScript Object Notation (JSON) で記述されていることを示します

この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。

改行は読みやすくするためです。

```
curl -X GET -H "Accept: application/json" http://localhost:4848/management/domain
{

  "Domain":{"log-root":"${com.sun.aas.instanceRoot}/logs",
"application-root":"${com.sun.aas.instanceRoot}/applications",
"locale":"", "version":"74.1"},

  "Methods":{
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "log-root":{"Key":"false", "Type":"string", "Optional":"true"},
```

```
        "application-root":{"Key":"false", "Type":"string", "Optional":"true"},
        "locale":{"Key":"false", "Type":"string", "Optional":"true"},
        "version":{"Key":"false", "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
    }
  },

  "Child Resources":[
    "http://localhost:4848/management/domain/configs",
    "http://localhost:4848/management/domain/resources",
    "http://localhost:4848/management/domain/servers",
    "http://localhost:4848/management/domain/property",
    "http://localhost:4848/management/domain/applications",
    "http://localhost:4848/management/domain/system-applications",
    "http://localhost:4848/management/domain/stop",
    "http://localhost:4848/management/domain/restart",
    "http://localhost:4848/management/domain/uptime",
    "http://localhost:4848/management/domain/version",
    "http://localhost:4848/management/domain/rotate-log",
    "http://localhost:4848/management/domain/host-port"
  ]
```

▼ ツリーにノードを追加する

1. サーバーが実行されていることを確認します。

   Enterprise Server のデータについて、REST リソースを操作するには、稼働中のサーバーが必要です。

2. ノードの親を表すリソースの POST メソッドについて、使用できるメッセージパラメータを調べます。

   この手順の実行方法については、68 ページの「ツリー内のノードがサポートするメソッドおよびメソッドパラメータを判定する」を参照してください。

3. 追加するノードの親を表す **REST** リソースに対して、POST メソッドを使用します。

4. ノードが追加されたことを確認します。

   追加したノード (親ではない) を表すリソースに対してこの手順を実行します。この手順の実行方法については、70 ページの「ツリー内のノードのデータを取得する」を参照してください。

例 2–23 ツリーへのノードの追加

この例は、cURL ユーティリティーを使用して、JDBC リソースを表す REST リソースを作成することにより、ツリーに JDBC リソースを追加します。

この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。

改行は読みやすくするためです。

1. この手順は、リソース jdbc-resource の POST メソッドに使用できるメッセージパラメータを調べます。

```
curl -X OPTIONS -H "Accept: application/json"
http://localhost:4848/management/domain/resources/jdbc-resource
{"JdbcResource":
  {
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "id":{"Acceptable Values":"", "Default Value":"", "Type":"string",
            "Optional":"false"},
        "enabled":{"Acceptable Values":"", "Default Value":"true",
            "Type":"boolean", "Optional":"true"},
        "description":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"},
        "target":{"Acceptable Values":"", "Default Value":"", "Type":"string",
            "Optional":"true"},
        "property":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"},
        "connectionpoolid":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"false"}
      }
    },
    "Method":{
      "Name":"GET"
    }
  }
}
```

2. この手順は、jdbc-resource リソースの子としてリソースを追加します。cURL ユーティリティーのオプション -d は、必要なメッセージパラメータを次のように設定します。
   - id が jdbc/myjdbcresource に設定されます。
   - connectionpoolid が DerbyPool に設定されます。

```
curl -X POST -d "id=jdbc/myjdbcresource&connectionpoolid=DerbyPool"
http://localhost:4848/management/domain/resources/jdbc-resource
"http://localhost:4848/management/domain/resources/jdbc-resource/
jdbc/myjdbcresource" created successfully.
```

3. この手順は、ノードを表す REST リソースのデータを取得することにより、ノードが追加されたことを確認します。

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
{

  "JdbcMyjdbcresource":{"enabled":"true", "pool-name":"DerbyPool",
      "description":"", "jndi-name":"jdbc/myjdbcresource", "object-type":"user"},

  "Methods":{
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "enabled":{"Key":"false", "Default Value":"true",
            "Type":"boolean", "Optional":"true"},
        "pool-name":{"Key":"false", "Type":"string", "Optional":"true"},
        "description":{"Key":"false", "Type":"string", "Optional":"true"},
        "jndi-name":{"Key":"true", "Type":"string", "Optional":"true"},
        "object-type":{"Key":"false", "Default Value":"user",
            "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
    },
    "Method":{
      "Name":"DELETE",
      "Message Parameters":{
        "target":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"}
      }
    }
  }

}
```

▼ ツリー内のノードを更新する

1 サーバーが実行されていることを確認します。

Enterprise Server のデータについて、REST リソースを操作するには、稼働中のサーバーが必要です。

2. ノードを表すリソースの POST メソッドについて、使用できるメッセージパラメータを判定します。

   この手順の実行方法については、68 ページの「ツリー内のノードがサポートするメソッドおよびメソッドパラメータを判定する」を参照してください。

3. 更新するノードを表す **REST** リソースに対して、POST メソッドを使用します。

4. ノードが更新されたことを確認します。

   この手順の実行方法については、70 ページの「ツリー内のノードのデータを取得する」を参照してください。

例 2–24 ツリー内のノードの更新

この例は、cURL ユーティリティーを使用して、JDBC リソースを表す REST リソースを変更することにより、ツリーの JDBC リソースを更新します。

この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。

改行は読みやすくするためです。

1. この手順は、リソース jdbc-myjdbcresource の POST メソッドに使用できるメッセージパラメータを調べます。

```
curl -X OPTIONS -H "Accept: application/json"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
{"JdbcMyjdbcresource":
  {
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "enabled":{"Key":"false", "Default Value":"true",
            "Type":"boolean", "Optional":"true"},
        "pool-name":{"Key":"false", "Type":"string", "Optional":"true"},
        "description":{"Key":"false", "Type":"string", "Optional":"true"},
        "jndi-name":{"Key":"true", "Type":"string", "Optional":"true"},
        "object-type":{"Key":"false", "Default Value":"user",
            "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
    },
    "Method":{
      "Name":"DELETE",
      "Message Parameters":{
```

```
        "target":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"}
      }
    }
  }
}
```

2. この手順は、jdbc-myjdbcresource が表す JDBC リソースを無効にするように、REST リソース jdbc-myjdbcresource を更新します。cURL ユーティリティーのオプション -d は、メッセージパラメータ enabled を disabled に設定します。

```
curl -X POST -d "enabled=false"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
"http://localhost:4848/management/domain/resources/jdbc-resource/
jdbc-myjdbcresource" updated successfully.
```

3. この手順は、ノードを表す REST リソースのデータを取得することにより、ノードが更新されたことを確認します。

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
{

  "JdbcMyjdbcresource":{"enabled":"false", "pool-name":"DerbyPool",
       "description":"", "jndi-name":"jdbc/myjdbcresource", "object-type":"user"},

  "Methods":{
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "enabled":{"Key":"false", "Default Value":"true",
            "Type":"boolean", "Optional":"true"},
        "pool-name":{"Key":"false", "Type":"string", "Optional":"true"},
        "description":{"Key":"false", "Type":"string", "Optional":"true"},
        "jndi-name":{"Key":"true", "Type":"string", "Optional":"true"},
        "object-type":{"Key":"false", "Default Value":"user",
            "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
    },
    "Method":{
      "Name":"DELETE",
      "Message Parameters":{
        "target":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"}
```

```
        }
      }
    }

  }
```

▼ ツリーからノードを削除する

1. サーバーが実行されていることを確認します。

   Enterprise Server のデータについて、REST リソースを操作するには、稼働中のサーバーが必要です。

2. ノードが削除できることを確認します。

   この手順の実行方法については、68 ページの「ツリー内のノードがサポートするメソッドおよびメソッドパラメータを判定する」を参照してください。

3. ノードが削除されたことを確認します。

   削除したノードを表すリソースに対してこの手順を実行します。この手順の実行方法については、70 ページの「ツリー内のノードのデータを取得する」を参照してください。

例 2–25 ツリーからのノードの削除

この例は、cURL ユーティリティーを使用して、JDBC リソースを表す REST リソースを削除することにより、ツリーから JDBC リソースを削除します。

この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。

改行は読みやすくするためです。

1. この手順は、リソース `jdbc-myjdbcresource` がサポートする REST のメソッドを取得することにより、ノードが削除できることを確認します。

```
curl -X OPTIONS -H "Accept: application/json"
http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
{"JdbcMyjdbcresource":
  {
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "enabled":{"Key":"false", "Default Value":"true",
            "Type":"boolean", "Optional":"true"},
        "pool-name":{"Key":"false", "Type":"string", "Optional":"true"},
        "description":{"Key":"false", "Type":"string", "Optional":"true"},
```

```
        "jndi-name":{"Key":"true", "Type":"string", "Optional":"true"},
        "object-type":{"Key":"false", "Default Value":"user",
            "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
    },
    "Method":{
      "Name":"DELETE",
      "Message Parameters":{
        "target":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"}
      }
    }
  }
}
```

2. この手順は、リソース jdbc-myjdbcresource を削除します。

```
curl -X DELETE http://localhost:4848/management/domain/resources/
jdbc-resource/jdbc-myjdbcresource
```

3. この手順は、ノードの親を表す REST リソースのデータを取得することにより、ノードが削除されたことを確認します。

```
curl -X GET -H "Accept: application/json"
http://localhost:4848/management/domain/resources/jdbc-resource/
{

  "JdbcResource":{},

  "Methods":{
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "id":{"Acceptable Values":"", "Default Value":"", "Type":"string",
            "Optional":"false"},
        "enabled":{"Acceptable Values":"", "Default Value":"true",
            "Type":"boolean", "Optional":"true"},
        "description":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"true"},
        "target":{"Acceptable Values":"", "Default Value":"", "Type":"string",
            "Optional":"true"},
        "property":{"Acceptable Values":"", "Default Value":"", "Type":"string",
            "Optional":"true"},
        "connectionpoolid":{"Acceptable Values":"", "Default Value":"",
            "Type":"string", "Optional":"false"}
      }
    },
```

```
  "Method":{
    "Name":"GET"
  }
},

"Child Resources":[
  "http://localhost:4848/management/domain/resources/jdbc-resource/
      jdbc-__TimerPool",
  "http://localhost:4848/management/domain/resources/jdbc-resource/
      jdbc-__default"
]
```

CRUD 以外の操作用の子リソース

Enterprise Server の REST インタフェースは次に示すように、作成、読み取り、更新、および削除 (CRUD) 以外の操作もサポートしています。

- 状態管理
- クエリー
- アプリケーション配備

これらの操作は、操作の実行対象リソースの子リソースを通じてサポートされます。子リソースは、構成オブジェクトツリーのノードを表しません。

たとえば、ドメイン管理用リソースは、次の表に示す CRUD 以外の操作用の子リソースを提供します。

表 2–2 ドメインでの CRUD 以外の操作用の子リソース

リソース	アクション
`host-port`	DAS が稼働中のホスト、および DAS が HTTP 要求を待機するポートを表示します。
`restart`	ドメインの DAS を停止し、その後再起動します。
`rotate-log`	タイムスタンプ名を持つファイルを `server.log_`*date-and-time* の形式の名前に変更し、空のログを作成することにより、サーバーログファイルをローテーションします。
`stop`	ドメインの DAS を停止します。
`uptime`	DAS の最後の再起動後の動作時間を表示します。
`version`	Enterprise Server のバージョン情報を表示します。

REST インタフェースのセキュリティー保護

Enterprise Server の REST インタフェースは、セキュアな接続上での基本認証をサポートしています。セキュリティーを有効にしたときには、REST リソースの URL にプロトコルとして `https` を指定し、ユーザー名とパスワードを指定する必要があります。

Enterprise Server の REST インタフェースのセキュリティー保護では、次のタスクシーケンスが実行されます。

1. `admin-realm` ユーザーを `asadmin` ユーザーグループに追加
2. Secure Sockets Layer (SSL) を有効化

コマンド行からこれらのタスクを実行する方法については、次に示すドキュメントを参照してください。

- 217 ページの「認証レルムを作成する」
- 221 ページの「ファイルユーザーを作成する」
- 300 ページの「SSL の HTTP リスナーを構成する」

管理コンソール を使用してこれらのタスクを実行する方法については、管理コンソール のオンラインヘルプから次に示すトピックを参照してください。

- 管理レルムにユーザーを追加する
- プロトコルの SSL 設定を編集する

リソースの表現形式

Enterprise Server の REST インタフェースは、次に示す形式でリソースを表現します。

- JSON (`http://www.json.org/`)
- XML
- HTML

リソースの表現を指定する方法は、Enterprise Server の REST インタフェースへのアクセス方法によって異なります。たとえば、cURL ユーティリティーを使用している場合は、オプション `-H` を使用して、次のようにリソースの表現を指定します。

- JSON の場合は、 `-H "Accept: application/json"` を指定します。
- XML の場合は、 `-H "Accept: application/xml"` を指定します。
- HTML の場合は、オプション `-H` を省略します。

JSON リソースの表現

リソースの JSON 表現の一般的な形式は次のとおりです。

```
{
    "resource":{attributes},

    "Methods": {
        method-list
    }

    "Child Resources":[urls]
}
```

この形式の置き換え可能な項目は次のとおりです。

resource
リソース名。

attributes
コンマ (,) で区切られたゼロ以上の名前と値のペア。名前と値の各ペアは、"名前":値 として指定します。

method-list
リソースがサポートするメソッドを表現する、コンマ (,) で区切られた 1 つ以上のメタデータのセット。 各メタデータセットの形式については、80 ページの「メソッドリストのメソッドの JSON 表現」を参照してください。

urls
コンマ (,) で区切られたゼロ以上の子リソースの URL。

メソッドリストのメソッドの JSON 表現

メソッドリストのメソッドの JSON 表現は、次のとおりです。

```
Method":{
    "Name":"method-name",

    "Message Parameters":{
        message-parameter-list
    }

    "Query Parameters":{
        queryparameter- list
    }
}
```

この形式の置き換え可能な項目は次のとおりです。

method-name
GET、 POST、 DELETE のいずれかのメソッド名。

message-parameter-list
メソッドで使用できるメッセージパラメータを表す、コンマ (,) で区切られたゼロ以上のメタデータのセット。各メタデータセットの形式については、81 ページの「メッセージパラメータまたはクエリーパラメータの JSON 表現」.を参照してください。

query-parameter-list
メソッドで使用できるクエリーパラメータを表す、コンマ (,) で区切られたゼロ以上のメタデータのセット。各メタデータセットの形式については、81 ページの「メッセージパラメータまたはクエリーパラメータの JSON 表現」.を参照してください。

メッセージパラメータまたはクエリーパラメータの JSON 表現

メッセージパラメータまたはクエリーパラメータの JSON 表現は、次のとおりです。

"*parameter-name*" : {*attribute-list*}

この形式の置き換え可能な項目は次のとおりです。

parameter-name
パラメータ名。

attribute-list
コンマで区切られた、パラメータの属性の名前と値のペアのリスト。各ペアの形式は次のとおりです。

"*name*" : "*value*"

使用できる属性は次のとおりです。

`Default Value`
パラメータのデフォルト値。

`Acceptable Values`
パラメータに使用できる値のセットまたは範囲。

`Type`
パラメータのデータ型。次のいずれかです。

- `boolean`
- `int`
- `string`

`Optional`
パラメータが省略可能かどうか。`true` の場合、パラメータは省略可能です。`false` の場合、パラメータは必須です。

`Key`
パラメータがキーかどうか。`true` の場合、パラメータはキーです。`false` の場合、パラメータはキーではありません。

JSON リソースの表現例

この例は、ドメイン管理用リソースの JSON 表現を示します。この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。この例のリソースの URL は、`http://localhost:4848/management/domain` です。

改行は読みやすくするためです。

```
{

  "Domain":{"log-root":"${com.sun.aas.instanceRoot}/logs",
       "application-root":"${com.sun.aas.instanceRoot}/applications",
       "locale":"", "version":"73"},

  "Methods":{
    "Method":{
      "Name":"POST",
      "Message Parameters":{
        "log-root":{"Key":"false", "Type":"string", "Optional":"true"},
        "application-root":{"Key":"false", "Type":"string", "Optional":"true"},
        "locale":{"Key":"false", "Type":"string", "Optional":"true"},
        "version":{"Key":"false", "Type":"string", "Optional":"true"}
      }
    },
    "Method":{
      "Name":"GET"
    }
  },

  "Child Resources":[
    "http://localhost:4848/management/domain/configs",
    "http://localhost:4848/management/domain/resources",
    "http://localhost:4848/management/domain/servers",
    "http://localhost:4848/management/domain/property",
    "http://localhost:4848/management/domain/applications",
    "http://localhost:4848/management/domain/system-applications",
    "http://localhost:4848/management/domain/stop",
    "http://localhost:4848/management/domain/restart",
    "http://localhost:4848/management/domain/uptime",
    "http://localhost:4848/management/domain/version",
    "http://localhost:4848/management/domain/rotate-log",
    "http://localhost:4848/management/domain/host-port"
  ]

}
```

XML リソースの表現

リソースの XML 表現の一般的な形式は次のとおりです。

```
<resource  attributes>

    <Methods>
        method-list
    </Methods>
children
</type>
```

この形式の置き換え可能な項目は次のとおりです。

resource
リソース名。

attributes
空白文字 1 つで区切られたゼロ以上の名前と値のペア。名前と値の各ペアは名前="値"として指定します。

method-list
リソースがサポートするメソッドを表現する、1 つ以上の XML 要素。 各要素の形式については、83 ページの「リソースメソッドの XML 表現」を参照してください。

children
子リソースの URL を指定するゼロ以上の XML 要素。各要素は、*<child-resource> url</child-resource>* として指定します。*child-resource* は子リソースの名前、*url* は子リソースの URL です。

リソースメソッドの XML 表現

メソッドリストのメソッドの XML 表現は、次のとおりです。

```
<Method name="method-name">
    <Message-Parameters>
        message-parameter-list
    </Message-Parameters>
    <Query-Parameters>
        query-parameter-list
    </Query-Parameters>
</Method>
```

この形式の置き換え可能な項目は次のとおりです。

method-name
GET、POST、DELETE のいずれかのメソッド名。

message-parameter-list
メソッドで使用できるメッセージパラメータを表す、改行で区切られたゼロ以上の XML 要素。各要素の形式については、84 ページの「メッセージパラメータまたはクエリーパラメータの XML 表現」を参照してください。

query-parameter-list
メソッドで使用できるクエリーパラメータを表す、改行で区切られたゼロ以上の XML 要素。各要素の形式については、84 ページの「メッセージパラメータまたはクエリーパラメータの XML 表現」を参照してください。

メッセージパラメータまたはクエリーパラメータの XML 表現

メッセージパラメータまたはクエリーパラメータの XML 表現は、次のとおりです。

<parameter-name attribute-list/>

この形式の置き換え可能な項目は次のとおりです。

parameter-name
パラメータ名。

attribute-list
空白文字で区切られた、パラメータの属性の名前と値のペアのリスト。各ペアの形式は次のとおりです。

name="*value*"

使用できる属性は次のとおりです。

`Default Value`
パラメータのデフォルト値。

`Acceptable Values`
パラメータに使用できる値のセットまたは範囲。

`Type`
パラメータのデータ型。次のいずれかです。

- `boolean`
- `int`
- `string`

`Optional`
パラメータが省略可能かどうか。`true` の場合、パラメータは省略可能です。`false` の場合、パラメータは必須です。

`Key`
パラメータがキーかどうか。`true` の場合、パラメータはキーです。`false` の場合、パラメータはキーではありません。

XML リソースの表現例

この例は、ドメイン管理用リソースの XML 表現を示します。この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。この例のリソースの URL は、`http://localhost:4848/management/domain` です。

改行は読みやすくするためです。

```
<Domain log-root="${com.sun.aas.instanceRoot}/logs"
application-root="${com.sun.aas.instanceRoot}/applications" locale="" version="73">

  <Methods>
    <Method name="POST">
      <Message-Parameters>
        <log-root Key="false" Type="string" Optional="true"/>
        <application-root Key="false" Type="string" Optional="true"/>
        <locale Key="false" Type="string" Optional="true"/>
        <version Key="false" Type="string" Optional="true"/>
      </Message-Parameters>
    </Method>
    <Method name="GET">
    </Method>
  </Methods>

  <Child-Resources>
    <Child-Resource>http://localhost:4848/management/domain/configs</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/resources</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/servers</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/property</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/applications</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/system-applications</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/stop</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/restart</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/uptime</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/version</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/rotate-log</Child-Resource>
    <Child-Resource>http://localhost:4848/management/domain/host-port</Child-Resource>
  </Child-Resources>

</Domain>
```

HTML リソースの表現

リソースの HTML 表現の形式は Web ページであり、リソースに関して次の情報を提供します。

- リソースの属性とその値のリスト
- リソースがサポートするメソッドおよびメソッドパラメータのリスト各メソッドとそのパラメータは、HTML フォーム内の適切な型を持つフィールドとして表示されます。
- リソースの子を示すハイパーテキストリンクのリスト

Web ページの例については、図 2–1 を参照してください。この例では、DAS がローカルホストで稼働中で、管理用の HTTP ポートは 4848 です。この例のリソースの URL は、`http://localhost:4848/management/domain` です。

◆ ◆ ◆ 第 3 章

3 ドメインの管理

この章では、asadmin コマンド行ユーティリティーを使用して Sun GlassFish™ Enterprise Server v3 環境でドメインを管理する手順について説明します。

ここでは、以下のトピックに関して説明します。

本章で説明するタスクを 管理コンソール から実行する手順については、管理コンソール オンラインヘルプを参照してください。

ドメイン (サーバー) の管理について

ドメインは同時に管理されるインスタンスのグループです。ドメインは、事前に設定されたランタイムをユーザーアプリケーションに提供します。管理上の境界線を設けることに加え、ドメインは基本的なセキュリティー構造を提供し、これによって別々の管理者がサーバーインスタンスの特定のグループを管理できます。サーバーインスタンスを個別のドメインにグループ化することにより、さまざまな組織や管理者が 1 つの Enterprise Server インストールを共有できます。ドメインには、固有の構成、ログファイル、およびアプリケーションの配備領域があり、これらはほかのドメインとは無関係です。1 つのドメインの構成が変更されても、ほかのドメインの構成は影響を受けません。

Enterprise Server インストーラにより、デフォルトの管理ドメイン (domain1という名前) が作成されます。さらに、関連するドメイン管理サーバー (DAS) (server という名前) も作成されます。DAS は管理者を認証し、管理ツールからの要求を受け付け、ドメイン内のサーバーインスタンスと通信して要求を実行するように特別に指定されたインスタンスです。DAS はデフォルトサーバーと呼ばれることもありま

す。デフォルトサーバーと呼ばれる理由は、Enterprise Server のインストール時に作成される唯一のサーバーインスタンスで、配備に使用できるからです。

。デフォルトの管理ポートは 4848 ですが、インストール中に別のポートを指定することもできます。ドメインが作成されると、管理ユーザー名とパスワードを入力するよう求められますが、ユーザー名が admin でパスワードがない場合は、デフォルトのままにすることもできます。管理者パスワードをリセットするには、203 ページの「管理パスワードを変更する」を参照してください。

グラフィカルな 管理コンソール は、特定のDAS と通信し、その DAS と関連するドメインを管理します。管理コンソール の各セッションにより、特定のドメインを設定し、管理できます。ドメインを複数作成した場合は、別々に 管理コンソール セッションを起動して、それぞれのドメインを管理する必要があります。

ドメインの作成、ログイン、削除

ここでは、以下のトピックに関して説明します。

▼ ドメインの作成

Enterprise Server をインストールしてデフォルトドメイン (domain1) を作成してからは、ローカルの create-domain サブコマンドを使用してさらにドメインが作成できるようになります。このサブコマンドは、ドメインの構成を作成します。所定のシステムの asadmin ユーティリティーに対してアクセス権を持つユーザーは、ドメインを作成し、自分の選択するフォルダにそのドメイン構成を格納することができます。デフォルトでは、ドメイン構成はドメインのデフォルトディレクトリに作成されます。この場所をオーバーライドして、別の場所に構成を格納することもできます。

ドメインを作成すると、管理ユーザーを指定するよう求められます。または、パスワードなしでユーザー名が admin のデフォルトログイン ID のままにしておくこともできます。

始める前に　ドメインに適用するプロファイルを決定します。

1. 作成中のドメイン名を選択します。
   まだ使用されていないドメイン名かどうかを確認するには、list-domains(1) を使用します。

2 ドメインを作成するには、create-domain(1) サブコマンドを使用します。
このサブコマンドのオプションについては、このマニュアルページに記載されています。

3 ドメインの admin ユーザー名とパスワードを入力します。
admin ログインを設定しないようにするには、パスワードなしでデフォルトの admin のままにします。Return を押しても、デフォルトが選択されます。

例 3–1 ドメインの作成

この例では、domain1 というドメイン名を作成します。コマンドを入力すると、ログイン情報を入力するよう求められることがあります。

```
asadmin> create-domain --adminport 4848 domain1
Enter admin user name[Enter to accept default]>
Using port 4848 for Admin.
Default port 8080 for HTTP Instance is in use. Using 1161
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8081 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Default port 8686 for JMX_ADMIN is in use. Using 1162
Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=moonbeam.gateway.2wire.net,OU=GlassFish,O=Sun Microsystems,L=Santa Clara,ST
California,C=US]
Domain domain1 created.
Command create-domain executed successfully.
```

管理コンソール をブラウザで起動するには、次のフォーマットで URL を入力します。

```
http://hostname:5000
```

この例の場合、ドメインのログファイル、構成ファイル、および配備されたアプリケーションは次のディレクトリに置かれます。

domain-root-dir/mydomain

参照 このサブコマンドの完全な構文を確認するには、コマンド行に asadmin help create-domain と 入力してください。

▼ ドメインの一覧表示

ドメインとそのステータスを一覧表示するには、`list-domains` サブコマンドを使用します。ドメインのディレクトリが指定されていない場合は、デフォルト *as-install* の `/domains` ディレクトリにあるコンテンツが表示されます。複数のドメインが存在する場合は、ドメイン名を指定する必要があります。

別のディレクトリに作成されているドメインを一覧表示するには、`--domaindir` オプションを指定します。

- ドメインを一覧表示するには、`list-domains(1)` サブコマンドを使用します。

例 3–2 ドメインの一覧表示

この例では、デフォルトの *as-install*`/domains` ディレクトリを一覧表示しています。

```
asadmin> list-domains
Name: domain1 Status: Running
Name: domain4 Status: Not Running
Name: domain6 Status: Not Running
Command list-domains executed successfully.
```

参照 このサブコマンドの完全な構文を確認するには、コマンド行に `asadmin help list-domain` と入力してください。

▼ ドメインへのログイン

どのリモートサブコマンドにも資格が必要で、管理ユーザー名とそのパスワードを指定しなくてはなりません。ID が明示的または暗黙的に何も指定されていない場合は、デフォルトでは `asadmin` ユーザーが管理者向け操作を実行できるような ID でドメインが作成されます。デフォルト *ID* は、ユーザー名が `admin` でパスワードがない形式です。コマンド行やプロンプトにユーザー名を指定せず、`--passwordfile` オプションまたはプロンプトにパスワードを何も指定せず、`login` サブコマンドまたは `create-domain` サブコマンドに `----savelogin` オプションを付けたうちのいずれかでドメインにログインしたことがなければ、`asadmin` ユーティリティーが ID を何も指定せずに指定の管理操作を実行しようとします。サーバー (ドメイン) は、次の状況に当てはまる限り、このデフォルト ID を使用して管理操作を行えるようにします。

1. サーバー (ドメイン) が、admin ユーザーの認証にファイルレルムを使用している。
2. ファイルレルムに、ユーザーが 1 人しかいない (ユーザー名の内容は問わない)。
3. その 1 人のユーザーに、パスワードが用意されていない。

デフォルトでは、特定のユーザー名とパスワードでドメインを作成していない限り、これらの条件がすべて当てはまります。したがって、唯一の管理ユーザーは、パスワードなしの admin となります。3. に当てはまらない場合は、パスワードを指定する必要があります。2. に当てはまらない場合も、ユーザー名を指定する必要があります。1. に当てはまらない場合は、ユーザー名とパスワードを指定する必要があります。

特定ドメインへの認証 (ログイン) を行うには、ローカルモードで login サブコマンドを使用します。ログインした後は、そのドメインのそれ以降の操作に管理ユーザーやパスワードを指定しなくても済むようになります。login サブコマンドは、管理パスワードを指定する際にのみ使用できます。リモートサブコマンドから入力を求められるこれ以外のパスワードについては、--passwordfile オプションを使用するか、コマンドプロンプトでパスワードを指定してください。管理ユーザー名とパスワードの入力が常に求められます。

logout サブコマンドは一切ありません。別のドメインにログインするには、asadmin login の --host と --port に新しい値を付けて呼び出します。

1 ログインしようとしているドメイン名を指定します。

既存ドメインの一覧表示

```
asadmin list-domains
```

2 login(1)コマンドでドメインにログインします。

例 3–3 リモートマシンでのドメインへのログイン

この例では、別のマシンにあるドメインにログインします。login サブコマンドの前に、オプションを指定します。

```
asadmin> --host foo --port 8282 login
Please enter the admin user name>admin Please enter the admin password>
Trying to authenticate for administration of server at host [foo] and port [8282] ...
Login information relevant to admin user name [admin]
for host [foo] and admin port [8282] stored at [/.asadminpass] successfully.
Make sure that this file remains protected. Information stored in this
file will be used by asadmin commands to manage associated domain.
```

例 3–4 Localhost のデフォルトポートにあるドメインへのログイン

この例では、デフォルトポートの myhost にあるドメインにログインします。login サブコマンドの前に、オプションを指定します。

```
asadmin>  --host myhost login
Please enter the admin user name>admin Please enter the admin password>
Trying to authenticate for administration of server at host [myhost] and port [4848] ...
```

```
An entry for login exists for host [myhost] and port [4848], probably from
an earlier login operation.
Do you want to overwrite this entry (y/n)?y
Login information relevant to admin user name [admin] for host [myhost]
and admin port [4848] stored at [/home/joe/.asadminpass] successfully.
Make sure that this file remains protected. Information stored in this file will be used by
asadmin commands to manage associated domain.
```

参照　このサブコマンドの完全な構文を確認するには、コマンド行に `asadmin help login` と 入力してください。パスワードの詳細は、201 ページの「パスワードの管理」を参照してください。

▼ ドメインの削除

サーバーから既存のドメインを削除するには、`delete-domain` サブコマンドを使用します。このサブコマンドを実行できるのは、このドメインを管理する権限があるルートユーザー、もしくはオペレーティングシステムのユーザーだけです。

始める前に　削除する前に、ドメインがすでに停止している必要があります。

1. ドメインを一覧表示するには、`list-domains(1)` サブコマンドを使用します。
2. 必要な場合は、削除しようとしているドメインのドメインユーザーに通知してください。
3. 削除対象のドメインが停止していることを確認します。
   必要な場合は、94 ページの「ドメインの停止」を参照してください。
4. `delete-domain(1)` サブコマンドを使用して、ドメインを削除します。

例 3–5　ドメインの削除

この例では、`domain1` という名前のドメインを指定の場所から削除します。

```
asadmin> delete-domain --domaindir ..\domains domain1
Domain domain1 deleted.
Command delete-domain executed successfully.
```

参照　このサブコマンドの完全な構文を確認するには、コマンド行に `asadmin help delete-domain` と 入力してください。

ドメインの起動と停止

ここでは、以下のトピックに関して説明します。

▼ ドメインの起動

ドメインまたはサーバーの起動時に、ドメイン管理サーバー (DAS) が起動されます。DAS は、一度起動すると常時稼動となり、要求を待機して受け付けます。

ドメインのディレクトリが指定されていない場合は、デフォルトの *as-install*`/domains` ディレクトリにあるドメインが起動します。複数のドメインが存在する場合、`domain_name` オペランドを指定する必要があります。各ドメインは、別々に起動する必要があります。

起動しているドメインで `restart-domain` サブコマンドを使用できるようにするには、`--watchdog` オプションを true に設定してください (true がデフォルトです)。`--watchdog` オプションを false に設定しておくと、そのドメインで `restart-domain` サブコマンドが使用できません。

注 – Microsoft Windows の場合、ドメインを起動するにはもう 1 つ方法があります。Windows の「スタート」メニューで、「プログラム」>「Sun Microsystems」>「Enterprise Server」>「デフォルトサーバーを起動」を選択します。

このサブコマンドは、ローカルモードでのみサポートされています。

- ドメインを起動するには、`start-domain`**(1)** サブコマンドを使用します。

例 3–6 ドメインの起動

この例では、デフォルトドメインディレクトリ内の `domain2` を起動します。

```
asadmin> start-domain domain2
```

ドメインが 1 つだけの場合は、ドメイン名を省略できます。パスワードを含めていない場合は、入力を求められることがあります。

```
Name of the domain started: [domain1] and its location:
[C:\prelude\v3_prelude_release\distributions\web\target\glassfish
domains\domain1].
Admin port for the domain: [4848].
```

参照　このサブコマンドの完全な構文を確認するには、コマンド行に asadmin help start-domain と 入力してください。

▼ ドメインの停止

ドメインまたはサーバーを停止すると、そのドメインの管理サーバー (DAS) がシャットダウンします。ドメインを停止すると、その DAS は新しい接続を受け付けなくなり、未完了の接続がすべて完了するまで待機します。このシャットダウンプロセスには数秒間かかります。ドメインの停止処理中は、管理コンソール およびほとんどの asadmin サブコマンドが使用できません。このサブコマンドは、異常なサーバーを停止する際に特に有用です。制御された状態では、restart-domain(1) サブコマンドも使用できます。

注 – Microsoft Windows の場合、ドメインを停止するにはもう 1 つ方法があります。「スタート」メニューで、「プログラム」>「Sun Microsystems」>「Enterprise Server」>「デフォルトサーバー停止」を選択します。

---

1. 必要な場合は、停止しようとしているドメインのユーザーに通知します。
2. stop-domain(1) サブコマンドを使用して、ドメインを停止します。

例 3–7　ドメイン (サーバー) の停止

この例では、デフォルトディレクトリにある domain1 を停止します。ここで、domain1 はこのディレクトリに存在する唯一のドメインとします。

```
asadmin> stop-domain
Waiting for the domain to stop ...........
Command stop-domain executed successfully.
```

参照　このサブコマンドの完全な構文を確認するには、コマンド行に asadmin help stop-domain と 入力してください。

▼ ドメインの再起動

指定ホストのドメイン管理サーバー (DAS) を再起動するには、リモートモードで restart-domain サブコマンドを使用します。ドメインを再起動すると、その DAS は新しい接続を受け付けなくなり、未完了の接続がすべて完了するまで待機します。このシャットダウンプロセスには数秒間かかります。ドメインが再起動されるまで、管理コンソール およびほとんどの asadmin サブコマンドが使用できません。

このサブコマンドは、サーバーマシンがセキュリティー保護されていてアクセスしにくい環境で特に有用です。正当な資格がある状態であれば、そのサーバーをリモートからでも同一マシンからでも再起動できます。

サーバーが再起動しない場合は、stop-domain(1) サブコマンドの後に start-domain(1) サブコマンドを使用します。

始める前に　restart-domain サブコマンドを成功させるには、ドメインを起動した時に start-domain サブコマンドの --watchdog オプションを (デフォルトの) true に設定しておく必要があります。このオプションが false に設定された状態でドメインを再起動しようとすると、ドメインが停止し警告メッセージのログが記録されます。--watchdog オプションを false に設定している場合、ドメインを再起動するには stop-domain および start-domain サブコマンドを使用するほかありません。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. restart-domain(1) サブコマンドを使用して、ドメインを再起動します。

例 3–8 ドメイン (サーバー) の再起動

この例では、デフォルトディレクトリにある mydoimain4 を再起動します。

```
asadmin> restart-domain mydomain4
Waiting for the domain to restart ..........
Command restart-domain executed successfully.
```

例 3–9 ブラウザでのドメインの再起動

この例では、ブラウザで restart-domain サブコマンドを呼び出します。

```
http://yourhost:4848/__asadmin/restart-domain
```

参照　このサブコマンドの完全な構文を確認するには、コマンド行に asadmin help restart-domain と入力してください。

ドメインの自動再起動

ここでは、Solaris でドメインを自動的に再起動するようにシステムを設定する方法について説明します。

ここでは、以下のトピックに関して説明します。

- 96 ページの「Solaris 10 でのドメイン自動的再起動」

- 97 ページの「Linux 上での自動再起動」
- 97 ページの「ユーザーの Windows ログアウト時にサービスがシャットダウンされないようにする」

▼ Solaris 10 でのドメイン自動的再起動

`create-service` サブコマンドは、Solaris と Windows プラットフォームの両方でサポートされていますが、ここでは Solaris の手順のみ説明します。

Solaris 10 では、`asadmin create-service` サブコマンドを使用して、ドメイン管理サーバー (DAS) を再起動する Solaris Service Management Facility (SMF) サービスを作成できます。サービスはプロセスに、そのプロセスを実行するユーザーの特権を付与します。SMF サービスを作成する場合、デフォルトのユーザーはスーパーユーザーです。別のユーザーがプロセスを実行する必要がある場合は、`method_credential` にそのユーザーを指定します。

プロセスを Solaris 10 の特権ポートにバインドする場合、そのプロセスには `net_privaddr` 特権が必要です。Solaris オペレーティングシステムの特権ポートは、1024 より小さいポート番号です。

ユーザーが `net_privaddr` 特権を持っているかどうかを確認するには、そのユーザーとしてログインし、`ppriv -l | grep net_privaddr` コマンドを入力します。

SMF サービスを作成して有効にしたあと、ドメインが停止した場合は、SMF によって再起動されます。

始める前に `asadmin create-service` コマンドを実行するには、`solaris.smf.*` 認証が必要です。この認証の設定方法については、`useradd` および `usermod` のマニュアルページを参照してください。さらに次のディレクトリツリーでの書き込み権も必要です。`/var/svc/manifest/application/SUNWappserver`。通常、スーパーユーザーはこれらの権限をどちらも持っています。また、`svccfg`、`svcs`、`auths` などの Solaris 10 管理コマンドが PATH で使用できなければなりません。

特定の Enterprise Server ドメインにデフォルトのユーザー特権を与えないようにする場合は、サービスのマニフェストを変更し、サービスを再インポートします。

1. `create-service(1)` サブコマンドを使用して、サービスを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。
2. サービスが作成されたら、`svacdm enable` コマンドを使用してサービスを有効にします。
   次に例を示します。

```
svacdm enable /appserver/domains/domain1
```

例 3–10 Solaris 10 でドメインを自動再起動するためにサービスを作成する

この例では、サービスを作成します。

```
asadmin> create-service
The Service was created successfully. Here are the details:
Name of the service:application/GlassFish/domain1
Type of the service:Domain
Configuration location of the service:/home/gfuser/glassfish-installations
/glassfishv3/glassfish/domains
Manifest file location on the system:/var/svc/manifest/application
/GlassFish/domain1_home_gfuser_glassfish-installations_glassfishv3
_glassfish_domains/Domain-service-smf.xml.
You have created the service but you need to start it yourself.
Here are the most typical Solaris commands of interest:
* /usr/bin/svcs -a | grep domain1 // status
* /usr/sbin/svcadm enable domain1 // start
* /usr/sbin/svcadm disable domain1 // stop
* /usr/sbin/svccfg delete domain1 // uninstall
Command create-service executed successfully
```

参照 サービスを管理するときに、次の Solaris コマンドが役に立ちます。
`auths`、`smf_security`、`svcadm`、 `svccfg`、`rbac`、`useradd`、および `usermod`。

▼ Linux 上での自動再起動

Linux 上で再起動を設定するには、`/etc/inittab` を編集します。`/etc/rc.local` またはこれに相当するファイルを使用している場合は、`/etc/rc.local` に必要な `asadmin` サブコマンドを呼び出す行を追加します。

- `/etc/inittab` ファイルにテキストを 1 行追加します。
  次に例を示します。

  ```
  das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
  --passwordfile /opt/SUNWappserver/password.txt domain1
  ```

  このテキストは 1 行で記述してください。先頭の 3 文字はこのプロセスに対する一意の指示子ですが、これは変更可能です。

▼ ユーザーの Windows ログアウト時にサービスがシャットダウンされないようにする

デフォルトでは、Java Virtual Machine (JVM) は、Windows のシャットダウンまたはユーザーの Windows ログアウトが行われることを示すシグナルを Windows からキャッチし、Java VM 自身を完全にシャットダウンします。この動作により、Enterprise Server サービスがシャットダウンされます。ユーザーがログアウトす

るときにサービスがシャットダウンしないようにするには、-Xrs Java VM オプション (http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/java.html#Xrs) を設定します。

1. *as-install*\domains\ *domain-name*\config\domain.xml ファイル内の、**Java VM** オプションを定義するセクションに次の行を追加します。

   ```
   <jvm-options>-Xrs</jvm-options>
   ```

2. **Enterprise Server** サービスが稼働している場合、変更を有効にするには、そのサービスを再起動します。

その他のドメインタスク

ここでは、以下のトピックに関して説明します。

- 98 ページの「ドメインの稼働時間の表示」
- 99 ページの「サポートされている別の Java バージョンへドメインを切り換える」

▼ ドメインの稼働時間の表示

ドメイン管理サーバー (DAS) を最後に起動してから今まで稼動している期間を表示するには、リモートモードで uptime サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. uptime(1) サブコマンドで稼働時間を表示します。

例 3–11 DAS 稼働時間の表示

この例では、DAS の稼働時間を表示します。

```
asadmin> uptime
Uptime: 1 Weeks, 4 days, 0 hours, 17 minutes, 14 seconds, Total milliseconds: 951434595
Command uptime executed successfully.
```

参照 このサブコマンドの完全な構文を確認するには、コマンド行に asadmin help uptime と 入力してください。

▼ サポートされている別の Java バージョンへドメインを切り換える

Enterprise Server v3 には、Java™ プラットフォーム (Java Virtual Machine すなわち JVM™ マシン) のベースとなる仮想マシンに Version 6 Java SE プラットフォームが必要です。

注 – 新しい JVM マシンでドメインを作成したあとは、以前の Java バージョンにはダウングレードしないでください。ご使用の JVM マシンをダウングレードする必要がある場合は、そのドメインの Java バージョンだけをダウングレードしてください。

---

1 まだダウングレードしていない場合は、目的の **Java SDK (JRE** ではありません**)** をダウンロードして、システムにインストールしてください。

Java SDK は、`http://java.sun.com/j2se` でダウンロードできます。

2 **JDK** を変更するドメインを起動します。

次のフォーマットを使用します。

as-install`/bin/asadmin start-domain` *domain-name*

有効な JVM インストールの場合、次の順で場所がチェックされます。

a. `domain.xml` **(**`java-config` 内の `java-home`**)**

b. `asenv.conf` **(**`AS_JAVA="path to java home"` 設定**)**

有効な JDK が見つからない場合は、致命的エラーが発生し、その問題のレポートが返されます。

3 必要な場合は、ドメインの **JVM** マシン属性を変更します。

特に、`JAVA_HOME` 環境変数を変更する必要があります。`JAVA_HOME` 変数を変更するには、たとえば次のように入力します。

as-install`/bin/asadmin set "server.java-config.java-home=path-to-java-home"`

◆ ◆ ◆ 第 4 章

Java プラットフォームの仮想マシンの管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server v3 環境で Java™ プラットフォームの仮想マシン (Java 仮想マシン、または JVM™ マシン) を管理する手順について説明します。

ここでは、次のテーマを取り上げます。

- 101 ページの「JVM オプションの管理」
- 105 ページの「プロファイラの管理」

これらのタスクを 管理コンソール を使用して実行する場合の手順は、管理コンソール のオンラインヘルプで説明します。

JVM オプションの管理

Java 仮想マシンは、コンパイルされた Java プログラムのバイトコードを実行するための、インタプリタ型の処理エンジンです。仮想マシンは Java バイトコードをホストマシンのネイティブ命令に変換します。Java プロセスの 1 つである Enterprise Server は、Java アプリケーションの実行とサポートに仮想マシンを必要とします。JVM の設定は、Enterprise Server の構成の一部です。

ここでは、次のテーマを取り上げます。

- 102 ページの「JVM オプションを作成する」
- 102 ページの「JVM オプションを一覧表示する」
- 103 ページの「JVM オプションを削除する」
- 104 ページの「JVM レポートを生成する」

▼ JVM オプションを作成する

Java 構成または domain.xml ファイルのプロファイラ要素に JVM オプションを作成するには、リモートモードで create-jvm-options サブコマンドを使用します。プロファイラ用に作成された JVM オプションは、プロファイラを開始する設定を記録するために使用されます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. create-jvm-options(1) サブコマンドを使用して、JVM オプションを作成します。
   複数の JVM オプションを作成する場合は、コロン (:) を使用してオプションを区切ります。JVM オプション自体にコロン (:) が含まれている場合は、バックスラッシュ (\) を使用して区切り記号のコロンと区別します。

   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. 変更内容を適用するために、Enterprise Server を再起動します。94 ページの「ドメインの再起動」を参照してください。

例 4–1 JVM オプションの作成

この例では、複数の Java システムプロパティーを設定します。

```
asadmin> create-jvm-options -Dunixlocation=/root/example:
-Dvariable=\$HOME:
-Dwindowslocation=d\\:\\\sun\\\appserver:
-Doption1=-value1
created 4 option(s)
Command create-jvm-options executed successfully.
```

参照 コマンド行に asadmin help create-jvm-options と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JVM オプションを一覧表示する

既存の JVM オプションを一覧表示するには、リモートモードで list-jvm-options サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-jvm-options(1) サブコマンドを使用して、JVM オプションを一覧表示します。

例 4–2 JVM オプションの一覧表示

この例では、すべての JVM オプションを表示します。

```
asadmin> list-jvm-options
-Djava.security.auth.login.config=${com.sun.aas.instanceRoot}/config/login.conf
-XX: LogVMOutput
-XX: UnlockDiagnosticVMOptions
-Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.
config.serverbeans.AppserverConfigEnvironmentFactory
-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks
-XX:NewRatio=2
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
-Djdbc.drivers=org.apache.derby.jdbc.ClientDriver
-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks
-client
-Djava.ext.dirs=${com.sun.aas.javaRoot}/lib/ext${path.separator}${com.sun.aas.ja
vaRoot}/jre/lib/ext${path.separator}${com.sun.aas.instanceRoot}/lib/ext${path.se
parator}${com.sun.aas.derbyRoot}/lib
-Xmx512m
-XX:LogFile=${com.sun.aas.instanceRoot}/logs/jvm.log
-Djava.endorsed.dirs=${com.sun.aas.installRoot}/lib/endorsed

Command list-jvm-options executed successfully.
```

参照 コマンド行に `asadmin help list-jvm-options` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JVM オプションを削除する

Java 構成または `domain.xml` ファイルのプロファイラ要素から JVM オプションを削除するには、リモートモードで `delete-jvm-options` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-jvm-options(1)` サブコマンドを使用して、**JVM** オプションを一覧表示します。
3. 必要に応じて、**JVM** オプションを削除することをユーザーに通知します。
4. `delete-jvm-options(1)` サブコマンドを使用して、**JVM** オプションを削除します。
   複数の JVM オプションを削除する場合は、コロン (:) を使用してオプションを区切ります。JVM オプション自体にコロンが含まれている場合は、バックスラッシュ (\) を使用して区切り記号のコロンと区別します。

5 変更内容を適用するために、**Enterprise Server** を再起動します。**94** ページの「ドメインの再起動」を参照してください。

例 **4–3** 単一の JVM オプションの削除

この例では、1 つの JVM オプションを削除します。

```
asadmin> delete-jvm-options -Dopt1=A
deleted 1 option(s)
Command delete-jvm-options executed successfully.
```

例 **4–4** 複数の JVM オプションの削除

この例では、複数の JVM オプションを削除します。

```
asadmin> delete-jvm-options -Doption1=-value1:-Dvariable=\$HOME
deleted 2 option(s)
Command delete-jvm-options executed successfully.
```

参照 コマンド行に `asadmin help delete-jvm-options` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JVM レポートを生成する

指定したドメイン管理サーバー (DAS) のスレッド (スタックトレースのダンプ)、クラス、メモリー、およびロガーを示す JVM レポートを生成するには、リモートモードで `generate-jvm-report` サブコマンドを使用します。生成できるレポートのタイプは、概要 (デフォルト)、クラス、スレッド、およびログです。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 `generate-jvm-report`**(1)** サブコマンドを使用して、レポートを生成します。

例 **4–5** JVM レポートの生成

この例では、スレッド、クラス、およびメモリーの概要情報を表示します。

```
asadmin> generate-jvm-report --type summary
Operating System Information:
Name of the Operating System: Windows XP
Binary Architecture name of the Operating System: x86, Version: 5.1
Number of processors available on the Operating System: 2
```

```
System load on the available processors for the last minute: NOT_AVAILABLE.
(Sum of running and queued runnable entities per minute).
.
,
.
user.home = C:\Documents and Settings\Jennifer
user.language = en
user.name = Jennifer
user.timezone = America/New_York
user.variant =
variable = \$HOME
web.home = C:\Preview\v3_Preview_release\distributions\web\target\
glassfish\modules\web
Command generate-jvm-report executed successfully.
```

参照　コマンド行に asadmin help generate-jvm-report と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

プロファイラの管理

「プロファイラ」は、サーバーパフォーマンスの解析に使用される情報を生成します。ここでは、次のテーマを取り上げます。

- 105 ページの「プロファイラを作成する」
- 106 ページの「プロファイラを削除する」

▼ プロファイラを作成する

サーバーインスタンスは、Java 構成内のプロファイラ要素によって、特定のプロファイラと連動しています。プロファイラ用に作成された JVM オプションは、特定のプロファイラの有効化に必要な設定を記録するために使用されます。Java 構成にプロファイラ要素を作成するには、リモートモードで create-profiler サブコマンドを使用します。

存在できるプロファイラは 1 つだけです。すでにプロファイラが存在している場合はエラーメッセージが表示され、新しいプロファイラを作成する前に古いプロファイラを削除するように指示されます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. create-profiler(1) サブコマンドを使用して、プロファイラを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3 変更内容を適用するために、**Enterprise Server** を再起動します。
94 ページの「ドメインの再起動」を参照してください。

例 **4–6** プロファイラの作成

この例では、`sample_profiler` という名前のプロファイラを作成します。

```
asadmin> create-profiler --classpath=/home/appserver/ --nativelibrarypath=/u/home/lib
--enabled=false --property=defaultuser=admin:password=adminadmin sample_profiler
Command create-profiler executed successfully.
```

参照 コマンド行に `asadmin help create-profiler` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ プロファイラを削除する

Java 構成からプロファイラ要素を削除するには、リモートモードで `delete-profiler` サブコマンドを使用します。削除のあと、新しいプロファイラを作成できます。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 `delete-profiler(1)` サブコマンドを使用して、プロファイラを削除します。

3 変更内容を適用するために、**Enterprise Server** を再起動します。
94 ページの「ドメインの再起動」を参照してください。

例 **4–7** プロファイラの削除

この例では、`sample_profiler` という名前のプロファイラを削除します。

```
asadmin> delete-profiler sample_profiler
Command delete-profiler executed successfully.
```

参照 コマンド行に `asadmin help delete-profiler` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 5 章

スレッドプールの管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server v3 環境でスレッドプールを管理する手順について説明します。

ここでは、次のテーマを取り上げます。

これらのタスクを 管理コンソール を使用して実行する場合の手順は、管理コンソール のオンラインヘルプで説明します。

スレッドプールについて

Java™ プラットフォームの仮想マシン (Java 仮想マシン、または JVM™ マシン) は、多数の実行スレッドを同時にサポートすることができます。パフォーマンスの向上に役立つように、Enterprise Server は 1 つまたは複数のスレッドプールを維持します。特定のスレッドプールを、コネクタモジュール、ネットワークリスナー、または ORB (Object Request Broker) に割り当てることができます。

1 つのスレッドプールで、複数のコネクタモジュールおよびエンタープライズ Bean を処理できます。「要求スレッド」は、アプリケーションコンポーネントへのユーザーの要求を処理します。Enterprise Server は要求を受け取ると、その要求をスレッドプール内の使用可能なスレッドに割り当てます。スレッドはクライアントの要求を実行し、結果を返します。たとえば、現在ビジー状態のシステムリソースが必要な場合、スレッドはリソースが解放されるのを待ってから、リソースの使用を要求に許可します。

スレッドプールの構成

アプリケーションからの要求用に確保するスレッドの最小数と最大数を指定できます。スレッドプールはこれら 2 つの値の間で動的に調整されます。

ここでは、次のテーマを取り上げます。

▼ スレッドプールを作成する

スレッドプールを作成するには、リモートモードで `create-threadpool` サブコマンドを使用します。

サーバーは、指定された最小スレッドプールサイズに従って、アプリケーション要求用に確保するスレッドを割り当てます。その数は、指定された最大スレッドプールサイズまで増加できます。プロセスで使用可能なスレッドの数を増やすと、プロセスが同時に応答できるアプリケーション要求数が多くなります。

1 つのリソースアダプタまたはアプリケーションが Enterprise Server のすべてのスレッドを占有すると、スレッド不足が発生します。この状況は、Enterprise Server のスレッドを複数のスレッドプールに分割することで回避できます。

1. **サーバーが実行されていることを確認します。**
   リモートサブコマンドには、実行中のサーバーが必要です。

2. **`create-threadpool(1)` サブコマンドを使用して、新しいスレッドプールを作成します。**
   サブコマンドのオプションについては、このサブコマンドのマニュアルページを参照してください。

3. **変更内容を適用するために、Enterprise Server を再起動します。**
   94 ページの「ドメインの再起動」を参照してください。

注 – Web コンテナによって使用されるスレッドプールでは、再起動が必要ない場合もあります。

例 5–1 スレッドプールの作成

この例では、`threadpool-1` を作成します。

```
asadmin> create-threadpool --maxthreadpoolsize 100
      --minthreadpoolsize 20 --idletimeout 2 --workqueues 100 threadpool-1
Command create-threadpool executed successfully
```

参照　コマンド行に asadmin help create-threadpool と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ スレッドプールを一覧表示する

既存のスレッドプールを一覧表示するには、リモートモードで list-threadpools サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-threadpools(1) サブコマンドを使用して、既存のスレッドプールを一覧表示します。

例 5–2　スレッドプールの一覧表示

この例では、既存のスレッドプールを一覧表示します。

```
asadmin> list-threadpools
threadpool-1
Command list-threadpools executed successfully
```

参照　コマンド行に asadmin help list-threadpools と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ スレッドプールを更新する

set サブコマンドを使用して、指定したスレッドプールの値を更新します。

1. list-threadpools(1) サブコマンドを使用して、既存のスレッドプールを一覧表示します。
2. set(1) サブコマンドを使用して、スレッドプールの値を変更します。
   スレッドプールはドット表記名で識別されます。
3. 変更内容を適用するために、Enterprise Server を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

注 – Web コンテナによって使用されるスレッドプールでは、再起動が必要ない場合もあります。

例 5–3 スレッドプールの更新

この例では、max-thread-pool-size を元の値から 8 に変更します。

```
asadmin> set server.thread-pools.thread-pool.http-thread-pool.max-thread-pool-size=8
Command set executed successfully
```

参照 コマンド行に asadmin help set と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ スレッドプールを削除する

既存のスレッドプールを削除するには、リモートモードで delete-threadpool サブコマンドを使用します。スレッドプールがネットワークリスナーによって参照されている場合、そのプールの削除は失敗します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-threadpools(1) サブコマンドを使用して、既存のスレッドプールを一覧表示します。
3. delete-threadpool(1) サブコマンドを使用して、指定したスレッドプールを削除します。
4. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

注 – Web コンテナによって使用されるスレッドプールでは、再起動が必要ない場合もあります。

例 5–4 スレッドプールの削除

この例では、threadpool-1 を削除します。

```
asadmin> delete-threadpool threadpool-1
Command delete-threadpool executed successfully
```

参照　コマンド行に `asadmin help delete-threadpool` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 6 章

Web アプリケーションの管理

この章では、Sun GlassFish™ Enterprise Server v3 環境で Web アプリケーションを管理する方法について説明します。

ここでは、次のテーマを取り上げます。

これらのタスクの一部を 管理コンソール を使用して実行する場合の手順は、管理コンソール のオンラインヘルプで説明します。

サーブレットの呼び出し方法について

Enterprise Server に配備されたサーブレットは、ブラウザに URL を指定して呼び出すことができます。また、HTML ファイルや JSP ファイル内にリンクとして埋め込まれたサーブレットも呼び出すことができます。サーブレットの呼び出し URL の形式は次のとおりです。

`http://`*server*`:`*port*`/`*context-root*`/`*servlet-mapping*`?`*name*`=`*value*

次の表に URL の各セクションの意味を示します。

表 6–1 アプリケーション内のサーブレットを呼び出す URL のフィールド

URL 要素	説明
server:*port*	IP アドレス (またはホスト名) および省略可能のポート番号。 仮想サーバーのデフォルト Web モジュールにアクセスする場合は、この URL セクションだけを指定します。名前と値のパラメータも指定するのでない限り、*context-root* と *servlet-name* を指定する必要はありません。
context-root	アプリケーションのコンテキストルート は、`application.xml`、`sun-application.xml`、または `sun-web.xml` ファイルの `context-root` 要素で定義されます。個別に配備される Web モジュールのコンテキストルートは、配備中に指定されます。 アプリケーションと個別に配備された Web モジュールのどちらでも、デフォルトのコンテキストルートは WAR ファイルの名前から `.war` のサフィックスを除いた文字列になります。
servlet-mapping	`web.xml` ファイルで設定された `servlet-mapping`。
?*name*= *value*. . .	省略可能な要求のパラメータ。

例 6–1 URL によるサーブレットの呼び出し

この例では、`localhost` はホスト名、`MortPages` はコンテキストルート、および `calcMortgage` はサーブレットマッピングを表します。

```
http://localhost:8080/MortPages/calcMortgage?rate=8.0&per=360&bal=180000
```

例 6–2 JSP ファイル内でのサーブレットの呼び出し

JSP ファイル内でサーブレットを呼び出す場合は、相対パスを使用できます。次に例を示します。

```
<jsp:forward page="TestServlet"/><jsp:include page="TestServlet"/>
```

サーブレットのログ出力の変更

`ServletContext.log` のメッセージは、サーバーログに送信されます。デフォルトでは、サーブレットの `System.out` および `System.err` 出力はサーバーログに送信されます。起動中に、サーバーログのメッセージは `System.err` 出力にエコーされます。またデフォルトでは、`System.err` 出力用の Windows 専用コンソールはありません。

これらのデフォルト設定は、管理コンソール の「システムログに書き込み」ボックスを使用して変更できます。このボックスにチェックマークを付ける と、`System.out` 出力がサーバーログに送信されます。チェックマークを外す と、`System.out` 出力はシステムのデフォルトの場所だけに送信されます。

Web アプリケーションのグローバルな機能の定義

default-web.xml ファイルを使用して、フィルタやセキュリティー制約などの、すべての Web アプリケーションに適用される機能を定義できます。

たとえば、ディレクトリの一覧表示は、セキュリティーの強化のためにデフォルトで無効化されます。ドメインの default-web.xml ファイルでディレクトリの一覧表示を有効にするには、servlet-name が default であるサーブレットの定義を検索し、listings という名前の init-param を true に設定します。続いて、サーバーを再起動します。

```
<init-param>
   <param-name>listings</param-name>
   <param-value>true</param-value>
</init-param>
```

listings を true に設定した場合、ディレクトリの一覧表示をソートする方法も決定できます。sortedBy という名前の init-param の値を、NAME、SIZE、または LAST_MODIFIED に設定します。続いて、サーバーを再起動します。

```
<init-param>
   <param-name>sortedBy</param-name>
   <param-value>LAST_MODIFIED</param-value>
</init-param>
```

default-web.xml ファイルの mime-mapping 要素はグローバルで、すべての Web アプリケーションで継承されます。Web アプリケーションの web.xml ファイルで mime-mapping 要素を使用して、これらのマッピングを上書きするか、独自に定義することができます。mime-mapping 要素の詳細は、Servlet 仕様を参照してください。

default-web.xml ファイルは、管理コンソール を使用して編集するか、次の手順で直接編集することができます。

▼ default-web.xml ファイルを使用する

1. フィルタ、セキュリティー制約、またはその他の機能の JAR ファイルを、*domain-dir*/lib ディレクトリに配置します。
2. 配置した JAR ファイルを参照するように、*domain-dir*/config/default-web.xml ファイルを編集します。
3. 変更内容を適用するために、Enterprise Server を再起動します。
94 ページの「ドメインの再起動」を参照してください。

URL のリダイレクト

古い URL に対する要求を、新しい URL に対する要求として処理するように指定できます。この処理は、URL の「リダイレクト」と呼ばれます。

仮想サーバーのリダイレクトされる URL を指定するには、`redirect_n` プロパティーを使用します。*n* は正の整数で、複数の指定が可能です。各 `redirect_n` プロパティーは、仮想サーバーに配備されたすべての Web アプリケーションによって継承されます。

`redirect_n` プロパティーの値には 2 つの構成要素があり、任意の順番で指定できます。

- 最初の構成要素の `from` は、照合する要求された URI のプレフィックスを指定します。
- 2 番目の構成要素の `url-prefix` はクライアントに返す新しい URL プレフィックスを指定します。 from プレフィックスは、この URL プレフィックスで置き換えられます。

例 6–3 URL のリダイレクト

この例では、`from` に指定した `dummy` を `etude` にリダイレクトします。

```
<property name="redirect_1" value="from=/dummy url-prefix=http://etude"/>
```

mod_jk の管理

`mod_jk` コネクタを使用すると、Web コンテナを Apache HTTP Server などの Web サーバーに接続できます。`mod_jk` は Enterprise Server に付属するコネクタで、これを使用することにより、Enterprise Server の前に Apach HTTP Server をたてることができます。この処理の一般的な目的は、静的なリソースに対する要求を Apache HTTP Server に処理させ、サーブレットや JavaServer™ Pages (JSP) などの動的なリソースに対する要求を、Enterprise Server のバックエンドインスタンスに転送して処理することです。

負荷分散の目的で、`mod_jk` を JSP またはサーブレットエンジンで直接使用することもできます。

ここでは、次のテーマを取り上げます。

- 117 ページの「mod_jk を有効にする」
- 119 ページの「mod_jk と Enterprise Server を使用して負荷分散する」

▼ mod_jk を有効にする

ここで説明するように mod_jk コネクタを有効化することで、Enterprise Server と Apache HTTP Server を連携させることができます。ネットワークリスナーの jk-enabled 属性を使用する場合、追加の JAR ファイルを /lib ディレクトリにコピーする必要はありません。ネットワークリスナー属性の jk-enabled を使用することで、JK コネクタを別の仮想サーバーに作成することもできます。

1 **Apache HTTP Server** と mod_jk をインストールします。

- Apache HTTP Server のインストール方法については、http://httpd.apache.org/docs/2.0/install.html を参照してください。
- mod_jk のインストール方法については、http://tomcat.apache.org/connectors-doc/webserver_howto/apache.html を参照してください。

2 次のファイルを設定します。

- apache2/conf/httpd.conf (Apache の主要な構成ファイル)
- apache2/config/workers.properties または *domain-dir* /config/glassfish-jk.properties (http://tomcat.apache.org/tomcat-5.5-doc/config/ajp.html で説明されている属性のデフォルト以外の値を使用する場合)

worker.properties ファイルと glassfish-jk.properties ファイルの両方を使用する場合は、httpd.conf で参照されている (または、httpd.conf で最初に参照される) ファイルが優先されます。

3 **Apache HTTP Server** (httpd) を起動します。

4 **Web** アプリケーションが少なくとも **1** つ配備されている **Enterprise Server** を起動します。

配備済み Web アプリケーションを少なくとも 1 つ使用する Web コンテナが起動されていなければ、mod_jk コネクタは起動できません。

5 次のコマンドを実行して、**HTTP** リスナーを作成します。

```
asadmin> create-http-listener --listenerport 8009
--listeneraddress 0.0.0.0 --defaultvs server listener-name
```

6 次のコマンドを実行して、mod_jk を有効にします。

```
asadmin> set server-config.network-config.network-listeners.
network-listener.listener-name.jk-enabled=true
```

listener-name は、mod_jk を有効にするネットワークリスナーの ID です。

7 glassfish-jk.properties ファイルを使用していて、httpd.conf でこのファイルを参照していない場合は、次のコマンドを実行してファイルの場所を指定します。

```
asamin> create-jvm-options -Dcom.sun.enterprise.web.connector.enableJK.propertyFile=
domain-dir/config/glassfish-jk.properties
```

8 変更内容を適用するために、**Enterprise Server** を再起動します。

94 ページの「ドメインの再起動」を参照してください。

例 6–4 mod_jk を使用する場合の httpd.conf ファイルの設定

この例では、httpd.conf ファイルを示します。

```
LoadModule jk_module /usr/lib/httpd/modules/mod_jk.so
JkWorkersFile /etc/httpd/conf/worker.properties
# Where to put jk logs
JkLogFile /var/log/httpd/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel debug
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
JkRequestLogFormat "%w %V %T"
# Send all jsp requests to GlassFish
JkMount /*.jsp worker1
# Send all glassfish-test requests to GlassFish
JkMount /glassfish-test/* worker1
```

例 6–5 mod_jk を使用する場合の worker.properties ファイルの設定

この例では、worker.properties または glassfish-jk.properties ファイルを示します。

```
# Define 1 real worker using ajp13
worker.list=worker1
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
```

参照 Apache の詳細は、http://httpd.apache.org/ を参照してください。

▼ mod_jk と Enterprise Server を使用して負荷分散する

負荷分散は、1 台のコンピュータで実行しなければならない作業を複数のコンピュータに分配し、同じ時間でより多くの作業を行う処理です。

1 117 ページの「mod_jk を有効にする」の手順を実行します。

2 少なくとも 1 つの Web アプリケーションが配備されている、別の Enterprise Server を起動します。

配備済み Web アプリケーションを少なくとも 1 つ使用する Web コンテナが起動されていなければ、mod_jk コネクタは起動できません。

3 次のようなコマンドを実行して、HTTP リスナーを作成します。

```
asadmin> create-http-listener --listenerport 8010 --listeneraddress 0.0.0.0
--defaultvs server my-connector
```

複数のインスタンスが同じマシンで動作している場合は、別の JK ポートを選択する必要があります。ポートは worker.properties ファイルの worker.worker*.port に一致する必要があります。以下のプロパティーファイルの例を参照してください。

4 次のコマンドを実行して、mod_jk を有効にします。

```
asadmin> set server-config.network-config.network-listeners.
network-listener.listener-name.jk-enabled=true
```

listener-name は、mod_jk を有効にするネットワークリスナーの ID です。

5 変更を適用するには、Apache と Enterprise Server を再起動します。

94 ページの「ドメインの再起動」を参照してください。

例 6–6 負荷分散を行う場合の httpd.conf ファイルの設定

この例では、httpd.conf ファイルを示します。

```
LoadModule jk_module /usr/lib/httpd/modules/mod_jk.so
JkWorkersFile /etc/httpd/conf/worker.properties
# Where to put jk logs
JkLogFile /var/log/httpd/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel debug
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
```

```
JkRequestLogFormat "%w %V %T"
# Send all jsp requests to GlassFish
JkMount /*.jsp worker1
# Send all glassfish-test requests to GlassFish
JkMount /glassfish-test/* loadbalancer
```

例 6–7 負荷分散を行う場合の worker.properties ファイルの設定

この例では、worker.properties または glassfish-jk.properties ファイルを示します。worker.worker*.port は、作成した JK ポートに一致している必要があります。

```
worker.list=loadbalancer
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=1
worker.worker1.socket_keepalive=1
worker.worker1.socket_timeout=300
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8010
worker.worker2.lbfactor=1
worker.worker2.socket_keepalive=1
worker.worker2.socket_timeout=300
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=worker1,worker2
```

◆ ◆ ◆ 第 7 章

ロギングサービスの管理

この章では、Sun GlassFish™ Enterprise Server v3 環境で、ロギングを設定する手順とログ情報を表示する手順について説明します。

ここでは、次のテーマを取り上げます。

- 121 ページの「ロギングについて」
- 124 ページの「ログレベルの設定」
- 127 ページの「サーバーログのローテーション」
- 128 ページの「ログ情報の表示」

これらのタスクの実行とプロパティーの編集を 管理コンソール を使用して行う場合の手順は、管理コンソール のオンラインヘルプで説明します。

ロギングについて

「ロギング」は、構成エラー、セキュリティー障害、サーバーの不完全な動作といったサーバーの動作中に発生したイベントに関する情報を、Enterprise Server が取得するための処理です。このデータはログファイルに記録され、通常は問題が発生したときの主要な情報源となります。ログファイルの解析は、サーバーの健全性を判断する際に役立ちます。

アプリケーションコンポーネントでは Apache Commons Logging Library を使用してメッセージを記録することもできますが、より適切なログ構成を行うには、プラットフォーム標準の JSR 047 API をお勧めします。

ここでは、次のテーマを取り上げます。

- 122 ページの「ログファイル 」
- 123 ページの「ロガー名前空間」

ログファイル

Enterprise Server のログレコードはサーバーログに記録されます。デフォルトのサーバーログの名前は `server.log` で、通常は *domain-dir*`/logs` に保存されます。サーバーログのデフォルトの名前または場所は、管理コンソールを使用して変更できます。

サーバーログのほかに、*domain-dir*`/logs` ディレクトリには次のログも保存されます。

- HTTP サービスアクセスログ (`/access` サブディレクトリ)
- トランザクションサービスログ (`/tx` サブディレクトリ)

サーバーログのサイズが指定された値 (バイト単位) に達すると、ログはローテーションされ、タイムスタンプを付加した `server.log_` *date* という名前に変更されます。*date* は、ファイルがローテーションされた日時を表します。ログファイルのローテーションは、127 ページの「サーバーログのローテーション」の手順に従って手動で行うこともできます。

Enterprise Server のログレコードは、次の統一形式に従います。

`[#|`*yyyy-mm-ddThh:mm:ss.SSS-Z*`|`*Log Level*`|`*ProductName-Version*`|`*LoggerName*`|`*Key Value Pairs*`|`*Message*`|#]`

- `[#` と `#]` はレコードの開始と終了をマーク付けします。
- 垂直線 (`|`) は、レコードのフィールドを区切ります。
- *yyyy-mm-ddThh:mm:ss.SSSS-Z* は、レコードが作成された日時を表します。たとえば、`2006-10-21T13:25:53.852-0400` です。
- *Log Level* は、指定したログレベルを表します。ログレベルには、`SEVERE`、`WARNING`、`INFO`、`CONFIG`、`FINE`、`FINER`、および `FINEST` のいずれかの値を選択できます。デフォルトは、「`INFO`」です。
- *ProductName-Version* は、Enterprise Server の現在のバージョンを表します。たとえば、`glassfish` です。
- *LoggerName* は、ログモジュールのソースを識別する階層ロガーの名前空間です。たとえば、`javax.enterprise.system.core` です。
- *Key Value Pairs* は、キーの名前と値のペア (通常はスレッド ID) を表します。たとえば、`_ThreadID=14;` です。
- *Message* は、ログメッセージのテキストです。Enterprise Server のすべての `SEVERE` および `WARNING` のメッセージと、多くの `INFO` メッセージは、モジュールコードと数値で構成されるメッセージ ID で始まります。たとえば、`CORE5004` です。

ログレコードの例を次に示します。

```
[#|2006-10-21T13:25:53.852-0400|INFO|GlassFish10.0|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

管理コンソール では、ログレコードがわかりやすい形式で表示されます。

ロガー名前空間

list-logger-levels(1) サブコマンドを使用して、モジュールの既存のロガーを一覧表示できます。次に例を示します。

```
javax.enterprise.system.container.cmp: INFO
javax.enterprise.system.tools.admin: INFO
javax.enterprise.system.container.web: INFO
javax.enterprise.system.util: INFO
javax.enterprise.resource.webcontainer.jsf.timing: INFO
javax: INFO
javax.enterprise.resource.corba: INFO
javax.enterprise.system.core.naming: INFO
javax.enterprise.system.core.selfmanagement: INFO
javax.enterprise.system.container.ejb: INFO
javax.enterprise.resource.webcontainer.jsf.config: INFO
javax.enterprise.resource.javamail: INFO
org.apache.catalina: INFO
javax.enterprise.system.core.config: INFO
javax.enterprise.system.webservices.rpc: INFO
javax.enterprise.system.webservices.registry: INFO
javax.enterprise.system.tools.deployment: INFO
javax.enterprise.resource.jms: INFO
javax.enterprise.system: INFO
javax.enterprise.system.webservices.saaj: INFO
org.apache.jasper: INFO
javax.enterprise.resource.webcontainer.jsf.lifecycle: INFO
javax.enterprise.resource.jta: INFO
javax.enterprise.resource.jdo: INFO
javax.enterprise.resource.resourceadapter: INFO
javax.enterprise.system.core.transaction: INFO
javax.enterprise.resource.webcontainer.jsf.resource: INFO
javax.enterprise.system.core.security: INFO
javax.enterprise.resource.webcontainer.jsf.application: INFO
javax.enterprise.system.core.classloading: INFO
org.apache.coyote: INFO
javax.enterprise.resource.webcontainer.jsf.managedbean: INFO
javax.enterprise.system.container.ejb.mdb: INFO
javax.enterprise.resource.webcontainer.jsf.context: INFO
javax.enterprise.resource.webcontainer.jsf.renderkit: INFO
javax.enterprise.resource.webcontainer.jsf.facelets: INFO
javax.enterprise.resource.webcontainer.jsf.taglib: INFO
```

ログレベルの設定

「ログレベル」は、ログに記録するメッセージの粒度を決定します。ここには、エラーのみ (SEVERE) から詳細なデバッグ情報 (FINEST) までのいずれかを指定します。選択できる値は、SEVERE、WARNING、INFO、CONFIG、FINE、FINER、および FINEST です。各ログレベルは上位のレベルを含みます。つまり、特定のログレベル (たとえば、INFO) を設定した場合、これより上位のログレベル (SEVERE および WARNING) のメッセージも記録されます。もっとも低いログレベル (FINEST) に設定した場合、出力にはファイルのすべてのメッセージが含まれます。デフォルト設定は、INFO です。

ログレベルの設定には、グローバル設定とロガー固有の設定の 2 つを利用できます。ロガー固有の設定を選択し、この設定がグローバル設定と異なる場合は、ロガー固有の設定が優先されます。

グローバルログレベルを設定するには、logging.properties ファイルを編集します。モジュールごとのログレベルは、この節で説明するように asadmin サブコマンドを使用して設定します。

ログレベルの設定

ログレベルの設定は動的な処理であるため、変更を有効にするために Enterprise Server を再起動する必要はありません。

ここでは、次のテーマを取り上げます。

- 124 ページの「ロガーレベルを一覧表示する」
- 125 ページの「グローバルログレベルを設定する」
- 126 ページの「モジュールのロガーレベルを設定する」

▼ ロガーレベルを一覧表示する

モジュールと各モジュールの現在のログレベルを一覧表示するには、リモートモードで list-logger-levels サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-logger-levels(1) サブコマンドを使用して、既存のモジュールのロガーを一覧表示します。

例 7–1 モジュールのロガーレベルの一覧表示

この例では、既存のロガーの一部と、各ロガーで設定されているログレベルを一覧表示します。

```
asadmin> list-logger-levels
javax.enterprise.system.container.cmp: INFO
javax.enterprise.system.tools.admin: INFO
java.util.logging.ConsoleHandler: FINEST
javax.enterprise.system.container.web: INFO
javax.enterprise.system.util: INFO
javax.enterprise.resource.webcontainer.jsf.timing: INFO
javax: INFO
javax.enterprise.resource.corba: INFO
...
Command list-logger-levels executed successfully.
```

参照 コマンド行に asadmin help list-logger-levels と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ グローバルログレベルを設定する

「グローバルログレベル」は、すべてのロガーで記録されるイベントの種類を指定します。コンソールへのメッセージ出力のデフォルトレベルは INFO です。このレベルには、SEVERE と WARNING のメッセージも含まれます。

グローバルロギングは、logging.properties ファイルを編集して設定します。デフォルトの logging.properties ファイルは、domain.xml ファイルと同じディレクトリにあり、通常は *domain-dir*/config に保存されています。java.util.logging.config.file システムプロパティーにファイル名を指定して、別のファイルを選択することもできます。たとえば、次のように指定します。

```
java -Djava.util.logging.config.file=myfile
```

ConsoleHandler には、表示されるメッセージを制限する独立したログレベル設定があります。次に例を示します。

```
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter =
com.sun.enterprise.server.logging.UniformLogFormatter
```

1. テキストエディタで、ConsoleHandler のログレベルを設定している行を検索して変更を行います。
2. ファイルを保存します。

例 7–2 すべてのロガーを対象とするグローバルログレベルの変更

ログレベルをルートレベルで設定すると、すべてのロガーのレベルを設定できます。この例では、すべてのロガーのログレベルを INFO に設定します。

```
.level= INFO
```

▼ モジュールのロガーレベルを設定する

「モジュールログレベル」は、特定のロガーで記録するイベントの種類を指定します。コンソールへのメッセージ出力のデフォルトレベルは INFO です。このレベルには、SEVERE と WARNING のメッセージも含まれます。グローバルログレベルは、モジュールに固有のログレベルで上書きされます。

デフォルトでは、モジュールログレベルは FINE に設定されます。ロガーを設定する行は次のようになります。太字はモジュールを示しています。

```
#javax.enterprise.system.tools.level=FINE
#javax.enterprise.system.container.ejb.level=FINE
#javax.enterprise.system.core.security.level=FINE
#javax.enterprise.system.tools.admin.level=FINE
#javax.enterprise.level=FINE
#javax.enterprise.system.container.web.level=FINE
```

ログレベルの設定は動的な処理であるため、変更を有効にするために Enterprise Server を再起動する必要はありません。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-logger-levels(1) サブコマンドを使用して、既存のモジュールのロガーを一覧表示します。
3. set-log-level(1) サブコマンドを使用して、モジュールのログレベルを設定します。
   選択できる値は、SEVERE、WARNING、INFO、CONFIG、FINE、FINER、および FINEST です。

例 7–3 モジュールロガーのログレベルの設定

この例では、Web コンテナロガーのログレベルを FINE に設定します。

```
asadmin> set-log-level javax.enterprise.system.container.web.level=FINE
Command set-log-level executed successfully.
```

例 7–4 複数のロガーのログレベルの設定

この例では、セキュリティーロガーと Web コンテナロガーのログレベルを設定します。

```
asadmin> set-log-level javax.enterprise.system.core.security.level=FINE
javax.enterprise.system.container.web=WARNING
Command set-log-level executed successfully.
```

参照　コマンド行に `asadmin help set-log-level` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

サーバーログのローテーション

ログは `logging.properties` ファイルの設定に従って自動的にローテーションされます。これらの設定は 管理コンソール を使用して変更できます。

▼ ログファイルを手動でローテーションする

サーバーログファイルを手動でローテーションするには、リモートモードで `rotate-log` サブコマンドを使用します。デフォルトの場所にあるサーバーログは、ただちにタイムスタンプ付きの名前に変更され、新しいサーバーログが作成されます。

ログのローテーションは動的な処理であるため、変更を有効にするために Enterprise Server を再起動する必要はありません。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `rotate-log(1)` サブコマンドを使用して、ログをローテーションします。

例 7–5　手動によるログファイルのローテーション

この例では、`server.log` ファイルの名前を `yyyy-mm-dd_server.log` に変更し、デフォルトの場所に新しい `server.log` ファイルを作成します。

```
asadmin> rotate-log
Command rotate-log executed successfuly.
```

参照　コマンド行に `asadmin help rotate-log` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

ログ情報の表示

デフォルトでは、すべてのロギング情報は server.log ファイルに記録されます。このファイルは、通常 *domain-dir* /logs に保存されます。ロギング情報は、管理コンソール のログビューアを使用して表示できます。管理コンソール のロギング機能を使用する手順は、管理コンソール のオンラインヘルプで説明します。

特定のモジュールで収集された情報を表示するには、server.log ファイルをテキストエディタで開き、対象のモジュールを検索します。

◆ ◆ ◆ 第 8 章

監視サービスの管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server v3 のコンポーネントとサービスを監視する方法について説明します。Enterprise Server リソースを監視するための JConsole の設定手順についても説明します。

ここでは、次のテーマを取り上げます。

- 129 ページの「監視について」
- 137 ページの「監視の設定」
- 140 ページの「共通の監視データの表示」
- 142 ページの「総合的な監視データの表示 」
- 171 ページの「Enterprise Server の監視データを表示するための JConsole の設定」

管理コンソール を使用して監視を行う場合の手順は、管理コンソール のオンラインヘルプで説明します。

REST インタフェースを使用して監視を行う方法については、65 ページの「REST インタフェースによる Enterprise Server の管理」を参照してください。

監視について

「監視」は、パフォーマンスの向上や問題の解決のために、システムの統計を調査する処理です。監視サービスでは、1 秒あたりの要求数、平均応答時間、スループットなどの運用上の統計を追跡および表示できます。Enterprise Server に配備されたさまざまなコンポーネントとサービスの状態を監視することで、パフォーマンスのボトルネックの識別、障害の予測、根本原因の分析を行い、すべての機能を期待どおりに動作させることができます。監視によって収集されたデータは、パフォーマンスの調整や容量計画にも役立ちます。

Enterprise Server のこのリリースでは、監視はモジュール方式で公開され、多くのクライアントモジュールが監視統計にアクセスして、これらを表示することができま

す。これらのクライアントには、管理コンソール、asadmin ユーティリティー、AMX、および REST インタフェースが含まれます。

ここでは、次のテーマを取り上げます。

監視のツリー構造の仕組みについて

「監視可能なオブジェクト」は、監視の対象に指定できるコンポーネント、サブコンポーネント、またはサービスです。Enterprise Server は、ツリー構造を使って監視可能なオブジェクトを追跡します。ツリーは動的であるため、Enterprise Server のコンポーネントが追加または削除されるときに、ツリーも変更されます。

ツリー内の監視可能なオブジェクトは、そのオブジェクトで監視できる対象を正確に表す子オブジェクト (ノード) を持ちます。すべての子オブジェクトのアドレス指定にはドット (.) 文字を区切り記号として使用します。このようなドット区切りの名前は、「ドット表記名」と呼ばれます。ドット表記名の詳細は、dotted-names(5ASC) のマニュアルページを参照してください。

次のコマンドは、インスタンス server の監視可能な子オブジェクトを一覧表示します。

```
asadmin> list --monitor "server.*"

server.applications
server.connector-service
server.http-service
server.jms-service
server.containers.jruby
server.jvm
server.network
server.orb
server.resources
server.security
server.thread-pool
server.transaction-service
server.web
```

各オブジェクトはドット表記名で表されます。また、監視可能なオブジェクトの特定の属性も、ドット表記名で指定します。たとえば、jvm オブジェクトには memory 属性と maxheapsize という名前の統計があります。この属性は次のドット表記名で表します。

```
server.jvm.memory.maxheapsize
```

オブジェクトが監視可能であっても、常に監視する必要はありません。監視を有効にする手順については、137 ページの「監視の設定」を参照してください。

監視可能なオブジェクトのツリー構造について

監視可能なオブジェクトは、それぞれ階層的なツリー構造を持ちます。ツリー中の「**statistics*」などの置換可能な文字列は、統計を表示できる属性の名前を表しています。

ここでは、次のノードのツリー階層を説明します。

アプリケーションのツリー階層

`applications` ツリーには、次のノードが含まれます。

```
server.applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                 |--- bean-cache (for entity/sfsb) *
   |    |                 |--- bean-pool (for slsb/mdb/entity) *
   |    |                 |--- bean-methods
   |    |                      |---method1 *
   |    |                      |---method2 *
   |    |                 |--- timers (for s1sb/entity/mdb) *
   |    |--- web-module-1
   |    |        |--- virtual-server-1 *
   |    |                      |---servlet1  *
   |    |                      |---servlet2  *
   |--- standalone-web-module-1
   |    |        |----- virtual-server-2 *
   |    |                      |---servlet3 *
   |    |                      |---servlet4 *
```

```
|    |        |----- virtual-server-3 *
|    |                      |---servlet3 *(same servlet on different vs)
|    |                      |---servlet5 *
|--- standalone-ejb-module-1
|    |        |--- ejb2 *
|    |                 |--- bean-cache (for entity/sfsb) *
|    |                 |--- bean-pool (for slsb/mdb/entity) *
|    |                 |--- bean-methods
|    |                      |--- method1 *
|    |                      |--- method2 *
|    |                 |--- timers (for s1sb/entity/mdb) *
|--- jersey-application-1
|    |--- jersey
|    |        |--- resources
                        resource-0
                            hitcount
                                 *statistic
|--- application2
```

ドット表記名は、`server.applications.hello.server.request.maxtime` のようになります。

EJB の `method` ノード以下のドット表記名
は、`server.applications.ejbsfapp1.ejbsfapp1ejbmod1\.jar.SFApp1EJB1` のようになります。

Jersey のドット表記名
は、`server.applications.helloworld-webapp.jersey.resources.resource-0.hitcount.resourcehitcount-count` のようになります。

使用可能な統計については、146 ページの「EJB 統計」、151 ページの「Jersey の統計」、および 168 ページの「Web の統計」 を参照してください。

コネクタサービスのツリー階層

`connector-service` ツリーには、コネクタ接続プールなどの、プールの監視可能な属性が格納されます。`connector-service` ツリーには、次のノードが含まれます。

```
server.connector-service
        resource-adapter-1
             connection-pools
                  pool-1
             work-management
```

ドット表記名
は、`server.connector-service.resource-adapter-1.connection-pools.pool-1` のようになります。使用可能な統計については、152 ページの「JMS サービスおよびコネクタサービスの統計」を参照してください。

HTTP サービスのツリー階層

`http-service` ツリー階層には、次のノードが含まれます。

```
server.http-service
       virtual-server
           request
               *statistic
       _asadmin
           request
               *statistic
```

virtual-server ノード以下のドット表記名は、`server.http-service.virtual-server1.request.requestcount` のようになります。使用可能な統計については、150 ページの「HTTP サービスの統計」を参照してください。

JMS およびコンテナサービスのツリー階層

`jms-service` ツリーには、接続ファクトリ (リソースアダプタの接続プール) と作業管理 (Message Queue リソースアダプタ用) の監視可能な属性が格納されます。`jms-service` ツリーには、次のノードが含まれます。

```
server.jms-service
        connection-factories
             connection-factory-1
        work-management
```

`connection-factories` ノード以下のドット表記名は、`server.jms-service.connection-factories.connection-factory-1` のようになります。この表記名は、この接続ファクトリのすべての統計を表します。使用可能な統計については、152 ページの「JMS サービスおよびコネクタサービスの統計」を参照してください。

JRuby のツリー階層

`jruby` ツリーには、次のノードが含まれます。

```
server.containers.jruby.applications
         jruby-application
                *statistic
                http
                     *statistic
                runtime-pool
                     *statistic
```

使用可能な統計については、154 ページの「JRuby の統計」を参照してください。

JVM のツリー階層

jvm ツリーには、次のノードが含まれます。

```
server.jvm
          class-loading-system
          compilation-system
          garbage-collectors
          memory
          operating-system
          runtime
```

memory ノード以下のドット表記名は、server.jvm.memory.maxheapsize のようになります。使用可能な統計については、156 ページの「JVM の統計」を参照してください。

ネットワークのツリー階層

ネットワークの統計は、admin-listener 、http-listener-1、http-listener-2 などのネットワークリスナーに適用されます。network ツリーには、次のノードが含まれます。

```
server.network
          type-of-listener
              keep-alive
                    *statistic
              file-cache
                    *statistic
              thread-pool
                    *statistic
              connection-queue
                     *statistic
```

network ノード以下のドット表記名
は、server.network.admin-listener.keep-alive.maxrequests-count のようになります。使用可能な統計については、160 ページの「ネットワークの統計」を参照してください。

ORB のツリー階層

orb ツリーには、接続マネージャーの監視可能な属性が格納されます。orb ツリーには、次のノードが含まれます。

```
server.orb
    transport
        connectioncache
            inbound
```

```
            *statistic
        outbound
            *statistic
```

ドット表記名は、`server.orb.transport.connectioncache.inbound.connectionsidle-count` のようになります。使用可能な統計については、163 ページの「ORB の統計 (接続マネージャー)」を参照してください。

リソースのツリー階層

`resources` ツリーには、JDBC 接続プールやコネクタ接続プールなどの、プールの監視可能な属性が格納されます。`resources` ツリーには、次のノードが含まれます。

```
server.resources
       connection-pool
           request
               *statistic
```

ドット表記名は、`server.resources.jdbc-connection-pool1.numconnfree.count` のようになります。使用可能な統計については、164 ページの「リソースの統計 (接続プール)」を参照してください。

セキュリティーのツリー階層

security ツリーには、次のノードが含まれます。

```
server.security
       ejb
          *statistic
       web
          *statistic
       realm
          *statistic
```

ドット表記名は、`server.security.realm.realmcount-starttime` のようになります。使用可能な統計については、165 ページの「セキュリティーの統計」を参照してください。

スレッドプールのツリー階層

`thread-pool` ツリー階層には、接続マネージャーの監視可能な属性が格納され、次のノードが含まれます。

```
server.thread-pool
                orb
```

```
                    threadpool
                            thread-pool-1
                                *statistic
```

ドット表記名
は、server.thread-pool.orb.threadpool.thread-pool-1.averagetimeinqueue-current
のようになります。使用可能な統計については、166 ページの「スレッドプールの統計」を参照してください。

トランザクションサービスのツリー階層

transaction-service ツリーには、トランザクションをロールバックするためのトランザクションサブシステムに関して、監視可能な属性が格納されます。transaction-service ツリーには、次のノードが含まれます。

```
server.transaction-service
         statistic
```

ドット表記名は、server.tranaction-service.activeids のようになります。使用可能な統計については、168 ページの「トランザクションサービスの統計」を参照してください。

Web のツリー階層

web ツリーには、次のノードが含まれます。

```
server.web
           jsp
              *statistic
           servlet
              *statistic
           session
              *statistic
           request
              *statistic
```

servlet ノードのドット表記名は、server.web.servlet.activeservletsloadedcount
のようになります。使用可能な統計については、142 ページの「Web モジュールの共通統計」を参照してください。

アドオンコンポーネントの監視について

一般的に、アドオンコンポーネントは、Enterprise Server が実行時に収集できる統計を生成します。監視機能を追加することで、アドオンコンポーネントは Enterprise Server の配布で提供されているコンポーネントと同じ方法で、Enterprise Server に統計

を提供できるようになります。結果として、コンポーネントの製造元にかかわらず、インストールされているすべての Enterprise Server コンポーネントの統計を、同じ管理インタフェースを使用して監視することができます。

Enterprise Server を監視するためのツール

Enterprise Server のサービスとコンポーネントを監視するために、次の `asadmin` サブコマンドが提供されています。

- `enable-monitoring`、`disable-monitoring`、または `get` および `set` サブコマンドは、監視を有効または無効にするために使用します。手順については、137 ページの「監視の設定」を参照してください。
- `monitor --type` サブコマンドは、特定のタイプの監視可能なオブジェクトの基本データを表示するために使用します。手順については、140 ページの「共通の監視データの表示」を参照してください。
- `list --monitor` サブコマンドは、`monitor` サブコマンドで監視できるオブジェクトを表示するために使用します。ガイドラインと手順については、143 ページの「`list` および `get` サブコマンドを監視に使用する場合のガイドライン」を参照してください。
- `get` サブコマンドは、属性や値などの総合的なデータをドット表記名で表示するために使用します。`get` サブコマンドでワイルドカードのパラメータを使用すると、任意の監視可能なオブジェクトで使用可能なすべての属性を表示できます。詳細は、143 ページの「`list` および `get` サブコマンドを監視に使用する場合のガイドライン」を参照してください。

監視の設定

デフォルトでは、Enterprise Server の監視サービスは有効になっていますが、各モジュールの監視は有効になっていません。モジュールの監視を有効にするには、モジュールの監視レベルを LOW または HIGH に変更します。監視の必要がないオブジェクトでは、監視レベルを OFF のままにすることもできます。

- **LOW**。作成数やバイト数などの簡単な統計が含まれます。
- **HIGH**。簡単な統計に加え、メソッドの数や実行時間などのメソッドの統計が含まれます。
- **OFF**。監視を行いません。パフォーマンスへの影響はありません。

ここでは、次のタスクを説明します。

- 138 ページの「監視を有効にする 」
- 139 ページの「監視を無効にする」

▼ 監視を有効にする

`enable-monitoring` サブコマンドを使用して、監視サービス自体を有効にするか、個々のモジュールの監視を有効にします。監視はただちに有効になります。Enterprise Server の再起動は必要ありません。

`set(1)` サブコマンドを使用して、モジュールの監視を有効にすることもできます。`set` コマンドの使用は動的な手順ではないため、変更を有効にするには Enterprise Server を再起動する必要があります。

1. 現在監視が有効になっているサービスとコンポーネントを確認します。

```
asadmin> get server.monitoring-service.module-monitoring-levels.*
```

この例の出力は、HTTP サービスでは監視が有効でなく (OFF)、その他のオブジェクトでは有効であることを示しています。

```
configs.config.server-config.monitoring-service.module-monitoring-levels.web-container=HIGH
       configs.config.server-config.monitoring-service.module-monitoring-levels.http-service=OFF
           configs.config.server-config.monitoring-service.module-monitoring-levels.jvm=HIGH
```

2. `enable-monitoring(1)` サブコマンドを使用して、監視を有効にします。

サーバーの再起動は必要ありません。

例 8–1 監視サービスの動的な有効化

この例では、個々のモジュールの監視に影響を与えずに、監視サービスを有効にします。

```
asadmin> enable-monitoring
Command enable-monitoring executed successfully
```

例 8–2 モジュールの監視の動的な有効化

この例では、`ejb-container` モジュールの監視を有効にします。

```
asadmin> enable-monitoring --level ejb-container=HIGH
Command enable-monitoring executed successfully
```

例 8–3 set サブコマンドによるモジュールの監視の有効化

この例では、監視レベルを HIGH に設定することで、HTTP サービスの監視を有効にします。変更を有効にするには、サーバーを再起動する必要があります。

```
asadmin> set server.monitoring-service.module-monitoring-levels.http-service=HIGH
Command set executed successfully
```

参照　コマンド行に asadmin help enable-monitoring と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 監視を無効にする

disable-monitoring サブコマンドを使用して、監視サービス自体を無効にするか、個々のモジュールの監視を無効にします。監視はただちに停止されます。Enterprise Server の再起動は必要ありません。

set(1) サブコマンドを使用して、モジュールの監視を無効にすることもできます。set コマンドの使用は動的な手順ではないため、変更を有効にするには Enterprise Server を再起動する必要があります。

1　現在監視が有効になっているサービスとコンポーネントを確認します。

asadmin get server.monitoring-service.module-monitoring-levels.*

この例の出力は、web-container 、http-service、および jvm で監視が有効であることを示しています。

```
configs.config.server-config.monitoring-service.module-monitoring-levels.web-container=HIGH
       configs.config.server-config.monitoring-service.module-monitoring-levels.http-service=HIGH
              configs.config.server-config.monitoring-service.module-monitoring-levels.jvm=HIGH
```

2　disable-monitoring(1) サブコマンドを使用して、サービスまたはモジュールの監視を無効にします。

サーバーの再起動は必要ありません。

例 8–4　監視サービスの動的な無効化

この例では、個々のモジュールの監視レベルを変更することなく、監視サービスを無効にします。

```
asadmin> disable-monitoring
Command disable-monitoring executed successfully
```

例 8–5　モジュールの監視の動的な無効化

この例では、指定したモジュールの監視を無効にします。これらのモジュールの監視レベルは OFF に設定されます。

```
asadmin> disable-monitoring --modules web-container,ejb-container
Command disable-monitoring executed successfully
```

例 8–6 set サブコマンドによる監視の無効化

この例では、HTTP サービスの監視を無効にします。変更を有効にするには、サーバーを再起動する必要があります。

```
asadmin> set server.monitoring-service.module-monitoring-levels.http-service=OFF
Command set executed successfully
```

参照 コマンド行に asadmin help disable-monitoring と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

共通の監視データの表示

monitor サブコマンドを使用して、共通で監視されるオブジェクトについての基本的なデータを表示できます。

- 140 ページの「共通の監視データを表示する」
- 141 ページの「共通の監視統計」

▼ 共通の監視データを表示する

monitor サブコマンドの --type オプションを使用して、httplistener、jvm、webmodule などの、データを表示するオブジェクトを指定します。タイプを指定せずに monitor サブコマンドを使用すると、エラーメッセージが表示されます。

サブコマンドの出力は、表形式で続けて表示されます。--interval オプションを使用すると、特定の間隔 (デフォルトでは 30 秒) で出力を表示することができます。

始める前に 監視可能なオブジェクトのデータを表示する前に、対象のオブジェクトで監視を設定する必要があります。138 ページの「監視を有効にする 」を参照してください。

1. 監視する監視可能なオブジェクトのタイプを決定します。
   v3 では、jvm、httplistener 、および webmodule を選択できます。
2. monitor(1) サブコマンドを使用して、監視データを要求します。

例 8–7 共通の監視データの表示

この例では、インスタンス server の jvm タイプの共通データを要求します。

```
asadmin> monitor --type jvm server

UpTime(ms)                          Heap and NonHeap Memory(bytes)
```

```
current                 min       max        low       high      count

9437266                 8585216   619642880  0         0         93093888
9467250                 8585216   619642880  0         0         93093888
```

参照　コマンド行に `asadmin help monitor` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

共通の監視統計

共通の監視統計について、次の節で説明します。

- 141 ページの「HTTP リスナーの共通統計」
- 141 ページの「JVM の共通統計」
- 142 ページの「Web モジュールの共通統計」

HTTP リスナーの共通統計

`httplistener` タイプに関して利用可能な統計を、次の表に示します。

表 8–1　HTTP リスナーの共通監視統計

Statistic	説明
`ec`	エラー数。エラー数の累積値です。
`mt`	最大時間。要求あたりの最長応答時間です。累積値ではなく、応答時間の中で最大の値です。
`pt`	処理時間。各要求の処理にかかった時間の累積値。処理時間は、要求全体での要求処理時間の平均になります。
`rc`	要求数。現時点までに処理された要求の累積数。

JVM の共通統計

`jvm` タイプに関して利用可能な統計を、次の表に示します。

表 8–2　JVM の共通監視統計

Statistic	説明
`count`	JVM マシンでの使用が保証されているメモリー量 (バイト)。
`high`	他のリリースとの互換性を維持するために使用されます。
`low`	他のリリースとの互換性を維持するために使用されます。
`max`	メモリー管理用として使用可能なメモリーの最大サイズ。

表 8–2 JVM の共通監視統計 (続き)

Statistic	説明
min	起動中のメモリー管理用に JVM マシンがオペレーティングシステムに要求するメモリー量の初期値 (バイト)。
UpTime	直前の起動日時からの JVM マシンの稼働時間 (ミリ秒)。

Web モジュールの共通統計

webmodule タイプに関して利用可能な統計を、次の表に示します。

表 8–3 Web モジュールの共通監視統計

Statistic	説明
ajlc	読み込まれているアクティブな JavaServer Pages™ (JSP™) テクノロジページの数。
asc	現在のアクティブなセッション。
aslc	読み込まれているアクティブなサーブレットの数。
ast	アクティブなセッションの合計数。
mjlc	読み込まれている JSP ページの最大数。
mslc	読み込まれているサーブレットの最大数
rst	拒否されたセッションの合計数。
st	セッションの合計数。
tjlc	読み込まれている JSP ページの合計数。
tslc	読み込まれているサーブレットの合計数。

総合的な監視データの表示

ツリー構造に対して list および get サブコマンドをドット表記名を使用して適用することで、各統計の説明や測定単位など、より総合的な監視データを表示することができます。

ここでは、次のテーマを取り上げます。

- 143 ページの「list および get サブコマンドを監視に使用する場合のガイドライン」
- 144 ページの「総合的な監視データを表示する」
- 146 ページの「総合的な監視統計」

list および get サブコマンドを監視に使用する場合のガイドライン

list コマンドと get コマンドでドット表記名を使用する場合は、基本として次の内容が前提となります。

- list サブコマンドにドット表記名を指定する場合、ドット表記名のあとにワイルドカード (*) を指定しないと、現在のノードの直接の子だけが一覧表示されます。たとえば、次のサブコマンドは server ノードに属する直接の子をすべて表示します。

  ```
  list --monitor server
  ```

- list サブコマンドにドット表記名を指定する場合、ドット表記名の後ろに .* の形式のワイルドカードを指定すると、指定したノードから子ノードの階層ツリーが一覧表示されます。たとえば、次のサブコマンドは applications ノード以下のすべての子ノードを、それぞれの子ノードも含めてすべて表示します。

  ```
  list --monitor server.applications.*
  ```

- list サブコマンドにドット表記名を指定する場合、ドット表記名の前または後ろに、**dottedname*、*dotted*name**、または *dottedname** の形式のワイルドカードを指定すると、指定したマッチングパターンで表される正規表現に一致する、すべてのノードとその子ノードが一覧表示されます。
- get サブコマンドのあとに .* または * を指定すると、指定したノードに含まれる属性とその値のセットが取得されます。

例として、list および get サブコマンドに resources ノードのドット表記名を使用した場合の出力を、次の表に示します。

表 8–4 リソースレベルのドット表記名の例

サブコマンド	ドット表記名	出力
list --monitor	server.resources	プール名の一覧。
list --monitor	server.resources.connection-pool1	属性は表示されず、代わりに「get --monitor サブコマンドを使用して、このノードの属性と値を表示してください」というメッセージが表示されます。
get --monitor	server.resources.connection-pool1.*	接続プールの属性に対応した属性と値のリスト。

ドット表記名の詳細は、dotted-names(5ASC) のマニュアルページを参照してください。

▼ 総合的な監視データを表示する

monitor サブコマンドは多くの状況で役に立ちますが、すべての監視可能なオブジェクトの完全なリストを表示することはできません。特定のオブジェクトタイプの総合的なデータを操作するには、list --monitor サブコマンドおよび get --monitor サブコマンドのあとに、監視可能なオブジェクトのドット表記名を指定します。

始める前に　監視可能なオブジェクトの情報を表示する前に、対象のオブジェクトで監視を設定する必要があります。必要な場合は、138 ページの「監視を有効にする」を参照してください。

1. list(1) サブコマンドを使用して、監視が有効なオブジェクトを一覧表示します。

   たとえば、次のサブコマンドは、インスタンス server で監視が有効になっているコンポーネントとサービスをすべて表示します。

```
asadmin> list --monitor "*"
server.web
server.connector-service
server.orb
server.jms-serviceserver.jvm
server.applications
server.http-service
server.thread-pools
```

2. get(1) サブコマンドを使用して、監視対象のコンポーネントまたはサービスのデータを取得します。

例 8–8　特定のタイプの属性の表示

この例では、インスタンス serverで、オブジェクトタイプ jvm のすべての属性に関する情報を取得します。

```
asadmin> get --monitor server.jvm.*
server.jvm.class-loading-system.loadedclasscount = 3715
server.jvm.class-loading-system.totalloadedclasscount = 3731
server.jvm.class-loading-system.unloadedclasscount = 16
server.jvm.compilation-system.name-current = HotSpot Client Compiler
server.jvm.compilation-system.totalcompilationtime = 769
server.jvm.garbage-collectors.Copy.collectioncount = 285
server.jvm.garbage-collectors.Copy.collectiontime = 980
server.jvm.garbage-collectors.MarkSweepCompact.collectioncount = 2
server.jvm.garbage-collectors.MarkSweepCompact.collectiontime = 383
server.jvm.memory.committedheapsize = 23498752
server.jvm.memory.committednonheapsize = 13598720
server.jvm.memory.initheapsize = 0
```

```
server.jvm.memory.initnonheapsize = 8585216
server.jvm.memory.maxheapsize = 66650112
server.jvm.memory.maxnonheapsize = 100663296
server.jvm.memory.objectpendingfinalizationcount = 0
server.jvm.memory.usedheapsize = 19741184
server.jvm.memory.usednonheapsize = 13398352
server.jvm.operating-system.arch-current = x86
server.jvm.operating-system.availableprocessors = 2
server.jvm.operating-system.name-current = Windows XP
server.jvm.operating-system.version-current = 5.1
server.jvm.runtime.classpath-current = glassfish.jar
server.jvm.runtime.inputarguments-current = []
server.jvm.runtime.managementspecversion-current = 1.0
server.jvm.runtime.name-current = 4372@ABBAGANI_WORK
server.jvm.runtime.specname-current = Java Virtual Machine Specification
server.jvm.runtime.specvendor-current = Sun Microsystems Inc.
server.jvm.runtime.specversion-current = 1.0
server.jvm.runtime.uptime = 84813
server.jvm.runtime.vmname-current = Java HotSpot(TM) Client VM
server.jvm.runtime.vmvendor-current = Sun Microsystems Inc.
server.jvm.runtime.vmversion-current = 1.5.0_11-b03
```

例 8–9 監視可能なアプリケーションの表示

この例では、インスタンス server の監視可能なアプリケーションをすべて表示します。

```
asadmin> list --monitor server.applications.*
server.applications.app1
server.applications.app2
server.applications.app1.virtual-server1
server.applications.app2.virtual-server1
```

例 8–10 アプリケーションの属性の表示

この例では、アプリケーション hello のすべての属性に関する情報を取得します。

```
asadmin> get --monitor server.applications.hello.*
server.applications.hello.server.activatedsessionstotal = 0
server.applications.hello.server.activejspsloadedcount = 1
server.applications.hello.server.activeservletsloadedcount = 1
server.applications.hello.server.activesessionscurrent = 1
server.applications.hello.server.activesessionshigh = 1
server.applications.hello.server.errorcount = 0
server.applications.hello.server.expiredsessionstotal = 0
server.applications.hello.server.maxjspsloadedcount = 1
server.applications.hello.server.maxservletsloadedcount = 0
```

```
server.applications.hello.server.maxtime = 0
server.applications.hello.server.passivatedsessionstotal = 0
server.applications.hello.server.persistedsessionstotal = 0
server.applications.hello.server.processingtime = 0.0
server.applications.hello.server.rejectedsessionstotal = 0
server.applications.hello.server.requestcount = 0
server.applications.hello.server.sessionstotal =
server.applications.hello.server.totaljspsloadedcount = 0
server.applications.hello.server.totalservletsloadedcount = 0
```

例 **8–11** 特定の属性の表示

この例では、インスタンス server の jvm 属性 runtime.vmversion-current に関する情報を取得します。

```
asadmin> get --monitor server.jvm.runtime.vmversion-current
server.jvm.runtime.vmversion-current = 10.0-b23
```

総合的な監視統計

目的の統計を表すドット表記名を指定することで、総合的な監視統計を取得できます。たとえば、次のドット表記名では、virtual-server1 の HTTP サービスに対する要求の累積数が表示されます。

```
server.http-service.virtual-server1.request.requestcount
```

監視可能なオブジェクトのそれぞれで使用可能な統計を、次の節の表に示します。

- 146 ページの「EJB 統計」
- 150 ページの「HTTP サービスの統計」
- 151 ページの「Jersey の統計」
- 152 ページの「JMS サービスおよびコネクタサービスの統計」
- 154 ページの「JRuby の統計」
- 156 ページの「JVM の統計」
- 160 ページの「ネットワークの統計」
- 163 ページの「ORB の統計 (接続マネージャー)」
- 164 ページの「リソースの統計 (接続プール)」
- 165 ページの「セキュリティーの統計」
- 166 ページの「スレッドプールの統計」
- 168 ページの「トランザクションサービスの統計」
- 168 ページの「Web の統計」

EJB 統計

EJB は、131 ページの「アプリケーションのツリー階層」に示したオブジェクトツリー内に含まれます。次のドット表記名パターンを使用して、アプリケーションの統計を取得します。

`server.applications.`*appname*`.`*ejbmodulename*`.`*ejbname*`.bean-cache.`*statistic*

アプリケーションに関して利用可能な統計を、次の節で説明します。

- 147 ページの「EJB キャッシュの統計」
- 148 ページの「EJB コンテナの統計」
- 148 ページの「EJB メソッドの統計」
- 149 ページの「EJB プールの統計」
- 150 ページの「タイマーの統計」

EJB キャッシュの統計

EJB キャッシュの統計では、次のドット表記名パターンを使用します。

`server.applications.`*appname*`.`*ejbmodulename*`.bean-cache.`*ejbname*`.`*statistic*

EJB キャッシュに関して利用可能な統計を、次の表に示します。

表 8–5　EJB キャッシュの監視統計

Statistic	データ型	説明
`cachemisses`	RangeStatistic	ユーザー要求に対する Bean がキャッシュ内で見つからなかった回数。
`cachehits`	RangeStatistic	ユーザー要求に対するエントリがキャッシュ内で見つかった回数。
`numbeansincache`	RangeStatistic	キャッシュ内の Beans 数。これは現在のキャッシュサイズです。
`numpassivations`	CountStatistic	非活性化された Bean の数。ステートフルセッション Beans にのみ適用されます。
`numpassivationerrors`	CountStatistic	非活性化中に発生したエラーの数。ステートフルセッション Beans にのみ適用されます。
`numexpiredsessionsremoved`	CountStatistic	クリーンアップスレッドによって削除された期限切れセッションの数。ステートフルセッション Beans にのみ適用されます。
`numpassivationsuccess`	CountStatistic	非活性化が正常に終了した回数。ステートフルセッション Beans にのみ適用されます。

EJB コンテナの統計

EJB コンテナの統計では、次のドット表記名パターンを使用します。

`server.applications.`*appname*`.`*ejbmodulename*`.container.`*ejbname*

EJB コンテナに関して利用可能な統計を、次の表に示します。

表 8–6　EJB コンテナの監視統計

Statistic	データ型	説明
`createcount`	CountStatistic	特定の EJB に対する `create` メソッドの呼び出し回数。
`messagecount`	CountStatistic	特定のメッセージ駆動型 Bean に対して受信されたメッセージの数。
`methodreadycount`	RangeStatistic	`MethodReady` 状態にあるステートフルまたはステートレスセッション Beans の数。
`passivecount`	RangeStatistic	`Passive` 状態にあるステートフルセッション Beans の数。
`pooledcount`	RangeStatistic	プールされた状態にあるエンティティー Bean の数。
`readycount`	RangeStatistic	実行可能状態にあるエンティティー Bean の数。
`removecount`	CountStatistic	特定の EJB に対する `remove` メソッドの呼び出し回数。

EJB メソッドの統計

EJB メソッドの統計では、次のドット表記名パターンを使用します。

`server.applications.`*appname*`.`*ejbmodulename*`.bean-methods.`*ejbname*`.`*statistic*

EJB メソッドの呼び出しに関して利用可能な統計を、次の表に示します。

表 8–7　EJB メソッドの監視統計

Statistic	データ型	説明
`executiontime`	CountStatistic	成功または失敗した最後の操作実行時にメソッド実行に費やされた時間 (ミリ秒)。この情報は、EJB コンテナの監視が有効になっている場合に、ステートレスおよびステートフルのセッション Beans とエンティティー Beans に対して収集されます。

表 8–7 EJB メソッドの監視統計 *(続き)*

Statistic	データ型	説明
methodstatistic	TimeStatistic	特定の操作の呼び出し回数。その呼び出しにかかった合計時間など。
totalnumerrors	CountStatistic	メソッド実行時に例外が発生した回数。この情報は、EJB コンテナの監視が有効になっている場合に、ステートレスおよびステートフルのセッション Beans とエンティティー Beans に対して収集されます。
totalnumsuccess	CountStatistic	メソッドが正常に実行された回数。この情報は、EJB コンテナの監視が有効になっている場合に、ステートレスおよびステートフルのセッション Beans とエンティティー Beans に対して収集されます。

EJB プールの統計

EJB プールの統計では、次のドット表記名パターンを使用します。

server.applications.*appname*.*ejbmodulename*.bean-pool.*ejbname*.*statistic*

EJB プールに関して利用可能な統計を、次の表に示します。

表 8–8 EJB プールの監視統計

Statistic	データ型	説明
jmsmaxmessagesload	CountStatistic	メッセージ駆動型 Bean のサービスを提供するために JMS セッション内に一度にロード可能なメッセージの最大数。デフォルトは 1 です。メッセージ駆動型 Beans 用のプールにのみ適用されます。
numbeansinpool	RangeStatistic	関連付けられたプール内の EJB 数。プールの変化に関する情報を提供します。
numthreadswaiting	RangeStatistic	未使用 Beans を取得するために待機しているスレッドの数。これは、要求が過剰である可能性を示します。
totalbeanscreated	CountStatistic	関連付けられたプール内でデータ収集開始後に作成された Beans の数。
totalbeansdestroyed	CountStatistic	関連付けられたプール内でデータ収集開始後に破棄された Beans の数。

タイマーの統計

タイマーの統計では、次のドット表記名パターンを使用します。

`server.applications.`*appname*`.`*ejbmodulename*`.timers.`*ejbname*`.`*statistic*

タイマーに関して利用可能な統計を、次の表に示します。

表 8–9　タイマーの監視統計

Statistic	データ型	説明
numtimerscreated	CountStatistic	システム内で作成されたタイマーの数。
numtimersdelivered	CountStatistic	システムによって配信されたタイマーの数。
numtimersremoved	CountStatistic	システムから削除されたタイマーの数。

HTTP サービスの統計

HTTP サービスは、133 ページの「HTTP サービスのツリー階層」に示したオブジェクトツリー内に含まれます。

HTTP サービスの統計を、次の節で説明します。

- 150 ページの「HTTP サービス仮想サーバーの統計」

HTTP サービス仮想サーバーの統計

HTTP サービス仮想サーバーの統計では、次のドット表記名パターンを使用します。

`server.http-service.`*virtual-server*`.request.`*statistic*

仮想サーバーに関する HTTP サービスの統計を、次の表に示します。

表 8–10　HTTP サービス仮想サーバーの監視統計

Statistic	データ型	説明
count200	CountStatistic	状態コードが 200 である応答の数。
count2xx	CountStatistic	状態コードが 2xx の範囲内にある応答の数。
count302	CountStatistic	状態コードが 302 である応答の数。
count304	CountStatistic	状態コードが 304 である応答の数。
count3xx	CountStatistic	状態コードが 3xx の範囲内にある応答の数。

表 8–10 HTTP サービス仮想サーバーの監視統計 (続き)

Statistic	データ型	説明
count400	CountStatistic	状態コードが 400 である応答の数。
count401	CountStatistic	状態コードが 401 である応答の数。
count403	CountStatistic	状態コードが 403 である応答の数。
count404	CountStatistic	状態コードが 404 である応答の数。
count4xx	CountStatistic	状態コードが 4xx の範囲内にある応答の数。
count503	CountStatistic	状態コードが 503 である応答の数。
count5xx	CountStatistic	状態コードが 5xx の範囲内にある応答の数。
countother	CountStatistic	状態コードが 2xx、3xx、4xx、および 5xx の範囲外である応答の数。
errorcount	CountStatistic	エラー回数の累計値。エラー回数は、応答コードが 400 以上になった場合の回数を表します。
hosts	StringStatistic	仮想サーバーのホスト (エイリアス) 名。
maxtime	CountStatistic	要求あたりの最長応答時間です。累積値ではなく、応答時間の中で最大の値です。
processingtime	CountStatistic	各要求の処理にかかった時間の累積値。処理時間は、要求数全体での要求処理時間の平均値になります。
requestcount	CountStatistic	現時点までに処理された要求の累積数。
state	StringStatistic	仮想サーバーの状態

Jersey の統計

Jersey は、「アプリケーションのツリー階層」に示したオブジェクトツリー内に含まれます。

Jersey の統計では、次のドット表記名パターンを使用します。

`server.applications.jersey-application.jersey.resources.`*resource-0*`.hitcount.`*statistic*

Jersey に関して利用可能な統計を、次の表に示します。

表 8–11 Jersey の統計

Statistic	データ型	説明
resourcehitcount	CountStatistic	このリソースクラスでのヒット数。
rootresourcehitcount	CountStatistic	このルートリソースクラスでのヒット数。

JMS サービスおよびコネクタサービスの統計

JMS サービスとコネクタサービスは、133 ページの「JMS およびコンテナサービスのツリー階層」に示したオブジェクトツリー内に含まれます。

JMS サービスとコネクタサービスの統計を、次の節で説明します。

- 152 ページの「コネクタ接続プールの統計 (JMS)」
- 153 ページの「コネクタ作業管理の統計 (JMS) 」

コネクタ接続プールの統計 (JMS)

JMS サービスとコネクタサービスの接続プールの統計では、次のドット表記名パターンを使用します。

server.connector-service.*resource-adapter-1*.connection-pool.*statistic*

コネクタ接続プールに関して利用可能な JMS サービスとコネクタサービスの統計を、次の表に示します。

表 8–12 コネクタ接続プールの監視統計 (JMS)

Statistic	データ型	説明
averageconnwaittime	CountStatistic	接続プールからサービスを受けるまでにかかった平均接続待ち時間。
connectionrequestwaittime	RangeStatistic	接続要求の最長待ち時間と最短待ち時間。現在の値は、プールのサービスを最後に受けた要求の待ち時間を示します。
numconnfailedvalidation	CountStatistic	開始時刻から前回のサンプリング時刻までの間に検証に失敗した接続プール内の接続の合計数。
numconnused	RangeStatistic	現在使用されている合計接続数に加え、過去に使用された接続の最大数 (ハイウォーターマーク) に関する情報も提供します。
numconnfree	RangeStatistic	前回のサンプリング時点におけるプール内の未使用接続の合計数。

表 8–12 コネクタ接続プールの監視統計 (JMS) (続き)

Statistic	データ型	説明
numconntimedout	CountStatistic	開始時刻から前回のサンプリング時刻までの間にタイムアウトしたプール内の接続の合計数。
numconncreated	CountStatistic	前回のリセット後に作成された物理接続の数。
numconndestroyed	CountStatistic	前回のリセット後に破棄された物理接続の数。
numconnacquired	CountStatistic	プールから取得された論理接続の数。
numconnreleased	CountStatistic	プールに解放された論理接続の数。
waitqueuelenght	CountStatistic	サービスを受けるためにキュー内で待機している接続要求の数。

コネクタ作業管理の統計 (JMS)

JMS サービスとコネクタサービスの作業管理の統計では、次のドット表記名パターンを使用します。

`server.connector-service.`*resource-adapter-1*`.work-management.`*statistic*

コネクタ作業管理に関して利用可能な JMS サービスとコネクタサービスの統計を、次の表に示します。

表 8–13 コネクタ作業管理の監視統計 (JMS)

Statistic	データ型	説明
activeworkcount	RangeStatistic	コネクタによって実行された作業オブジェクトの数。
completedworkcount	CountStatistic	完了した作業オブジェクトの数。
rejectedworkcount	CountStatistic	Enterprise Server によって拒否された作業オブジェクトの数。
submittedworkcount	CountStatistic	コネクタモジュールによって送信された作業オブジェクトの数。
waitqueuelength	RangeStatistic	実行される前にキュー内で待機している作業オブジェクトの数。
workrequestwaittime	RangeStatistic	作業オブジェクトが実行されるまでの最長待ち時間と最短待ち時間。

JRubyの統計

JRubyは、133ページの「JRubyのツリー階層」に示したオブジェクトツリー内に含まれます。

JRubyに関して利用可能な統計を、次の節で説明します。

- 154ページの「JRubyコンテナの統計」
- 154ページの「JRubyランタイムの統計」
- 155ページの「JRuby HTTPサービスの統計」

JRubyコンテナの統計

JRubyコンテナの統計では、次のドット表記名パターンを使用します。

`server.containers.jruby.applications.`*jruby-application*`.`*statistic*

JRubyコンテナに関して利用可能な統計を、次の表に示します。

表8–14　JRubyコンテナの統計

Statistic	データ型	説明
`environment`	StringStatistic	JRubyアプリケーション環境。
`appname`	StringStatistic	Rubyアプリケーション名。
`contextpath`	StringStatistic	Rubyアプリケーションのコンテキストパス。
`jrubyversion`	StringStatistic	JRubyのバージョン。
`rubyframework`	StringStatistic	Rubyアプリケーションフレームワーク。

JRubyランタイムの統計

JRubyランタイムの統計では、次のドット表記名パターンを使用します。

`server.containers.jruby.applications.`*jruby-application*`.runtime.`*statistic*

JRubyランタイムに関して利用可能な統計を、次の表に示します。

表8–15　JRubyランタイムの統計

Statistic	データ型	説明
`activeruntimes`	CountStatistic	現在アクティブなランタイムの数。
`appname`	StringStatistic	Rubyアプリケーション名。
`hardmaximum`	CountStatistic	アクティブなランタイムの最大数。

表 8–15 JRuby ランタイムの統計 (続き)

Statistic	データ型	説明
hardminimum	CountStatistic	アクティブなランタイムの最小数。

JRuby HTTP サービスの統計

JRuby HTTP サービスの統計では、次のドット表記名パターンを使用します。

`server.containers.jruby.applications.`*jruby-application*`.http.`*statistic*

JRuby HTTP サービスに関して利用可能な統計を、次の表に示します。

表 8–16 JRuby HTTP サービスの統計

Statistic	データ型	説明
address	StringStatistic	サーバーアドレス。
appname	StringStatistic	Ruby アプリケーション名。
averageprocessingtime	CountStatistic	平均要求処理時間 (ミリ秒)。
contextpath	StringStatistic	Ruby アプリケーションのコンテキストパス。
count2xx	CountStatistic	状態コードが 2xx の範囲内にある応答の数。
count200	CountStatistic	状態コードが 200 である応答の数。
count3xx	CountStatistic	状態コードが 3xx の範囲内にある応答の数。
count302	CountStatistic	状態コードが 302 である応答の数。
Count304	CountStatistic	状態コードが 304 である応答の数。
count4xx	CountStatistic	状態コードが 4xx の範囲内にある応答の数。
count400	CountStatistic	状態コードが 400 である応答の数。
count401	CountStatistic	状態コードが 401 である応答の数。
count403	CountStatistic	状態コードが 403 である応答の数。
count404	CountStatistic	状態コードが 404 である応答の数。
count5xx	CountStatistic	状態コードが 5xx の範囲内にある応答の数。
count503	CountStatistic	状態コードが 503 である応答の数。

表 8–16 JRuby HTTP サービスの統計 (続き)

Statistic	データ型	説明
countother	CountStatistic	状態コードがその他の値である応答の数。
errorcount	CountStatistic	状態コードが 400 より大きい応答の数。
requests/seconds	CountStatistic	1 秒あたりの要求数。

JVM の統計

JVM は、134 ページの「JVM のツリー階層」に示したオブジェクトツリー内に含まれます。

Java プラットフォームの仮想マシン (Java™ 仮想マシン、または JVM マシン) に関して利用可能な統計を、次の節で説明します。

JVM クラス読み込みシステムの統計

JVM クラス読み込みシステムの統計では、次のドット表記名パターンを使用します。

`server.jvm.class-loading-system.`*statistic*

Java SE では、JVM から追加の監視情報を取得できます。監視レベルを「低」に設定すると、この追加情報の表示が有効になります。監視レベルを「高」に設定すると、さらにシステム内の各ライブスレッドに関する情報も表示されます。Java SE で利用可能な追加監視機能の詳細は、『Monitoring and Management for the Java Platform』を参照してください。この文書は、`http://java.sun.com/javase/6/docs/technotes/guides/management/` で入手できます。

Java SE 監視ツールについては、`http://java.sun.com/javase/6/docs/technotes/tools/#manage` を参照してください。

Java SE の JVM で利用可能なクラス読み込み関連の統計を、次の表に示します。

表 8–17 Java SE のクラス読み込みに関する JVM の監視統計

Statistic	データ型	説明
loadedclasscount	CountStatistic	JVM 内に現在読み込まれているクラスの数。

表 8–17 Java SE のクラス読み込みに関する JVM の監視統計 (続き)

Statistic	データ型	説明
totalloadedclasscount	CountStatistic	JVM の実行開始後に読み込まれたクラスの合計数。
unloadedclasscount	CountStatistic	JVM の実行開始後に JVM から読み込み解除されたクラスの数。

Java SE の JVM で利用可能なスレッド関連の統計を、次の図に示します。

表 8–18 Java SE に関する JVM の監視統計 - スレッド

Statistic	データ型	説明
allthreadids	StringStatistic	すべてのライブスレッド ID のリスト。
currentthreadcputime	CountStatistic	CPU 時間の測定が有効になっている場合は、現在のスレッドに対する CPU 時間 (ナノ秒)。CPU 時間の測定が無効になっている場合は、-1 が返されます。
daemonthreadcount	CountStatistic	ライブデーモンスレッドの現在の数。
monitordeadlockedthreads	StringStatistic	監視デッドロックが発生しているスレッド ID のリスト。
peakthreadcount	CountStatistic	JVM 起動後またはピーク値リセット後におけるライブスレッドのピーク数。
threadcount	CountStatistic	ライブデーモンスレッドと非デーモンスレッドの現在の数。
totalstartedthreadcount	CountStatistic	JVM が起動されて以来、作成されたスレッド、起動されたスレッド、作成および起動されたスレッドの合計数。

JVM コンパイルシステムの統計

JVM コンパイルシステムの統計では、次のドット表記名パターンを使用します。

`server.jvm.compilation-system.`*statistic*

Java SE の JVM のコンパイルに関して利用可能な統計を、次の表に示します。

表 8–19 Java SE のコンパイルに関する JVM の監視統計

Statistic	データ型	説明
name-current	StringStatistic	現在のコンパイラの名前。

表 8–19 Java SE のコンパイルに関する JVM の監視統計 (続き)

Statistic	データ型	説明
totalcompilationtime	CountStatistic	コンパイルに費やされた時間の累計 (ミリ秒)。

JVM ガベージコレクタの統計

JVM ガベージコレクタの統計では、次のドット表記名パターンを使用します。

server.jvm.garbage-collectors.*statistic*

Java SE の JVM のガベージコレクションに関して利用可能な統計を、次の表に示します。

表 8–20 Java SE のガベージコレクタに関する JVM の監視統計

Statistic	データ型	説明
collectioncount	CountStatistic	実行されたコレクションの合計回数。
collectiontime	CountStatistic	コレクションに費やされた時間の累計 (ミリ秒)。

JVM メモリーの統計

JVM メモリーの統計では、次のドット表記名パターンを使用します。

server.jvm.memory.*statistic*

Java SE の JVM のメモリーに関して利用可能な統計を、次の表に示します。

表 8–21 Java SE のメモリーに関する JVM の監視統計

Statistic	データ型	説明
committedheapsize	CountStatistic	JVM 用としてコミットされたヒープメモリーのサイズ (バイト)。
committednonheapsize	CountStatistic	JVM 用としてコミットされた非ヒープメモリーのサイズ (バイト)。
initheapsize	CountStatistic	JVM が最初に要求したヒープのサイズ。
initnonheapsize	CountStatistic	JVM が最初に要求した非ヒープ領域のサイズ
maxheapsize	CountStatistic	メモリー管理用として使用可能なヒープメモリーの最大サイズ (バイト)。

表 8–21 Java SE のメモリーに関する JVM の監視統計 (続き)

Statistic	データ型	説明
maxnonheapsize	CountStatistic	メモリー管理用として使用可能な非ヒープメモリーの最大サイズ (バイト)。
objectpendingfinalizationcount	CountStatistic	ファイナライズを保留しているオブジェクトの概算数。
usedheapsize	CountStatistic	現在使用されているヒープのサイズ。
usednonheapsize	CountStatistic	現在使用されている非ヒープ領域のサイズ。

JVM オペレーティングシステムの統計

JVM オペレーティングシステムの統計では、次のドット表記名パターンを使用します。

`server.jvm.operating-system.`*statistic*

Java SE の JVM マシンのオペレーティングシステムに関して利用可能な統計を、次の表に示します。

表 8–22 Java SE のオペレーティングシステムに関する JVM の統計

Statistic	データ型	説明
arch-current	StringStatistic	オペレーティングシステムのアーキテクチャー。
availableprocessors	CountStatistic	JVM が使用できるプロセッサの数。
name-current	StringStatistic	オペレーティングシステムの名前。
version-current	StringStatistic	オペレーティングシステムのバージョン。

JVM ランタイムの統計

JVM ランタイムの統計では、次のドット表記名パターンを使用します。

`server.jvm.runtime.`*statistic*

Java SE の JVM ランタイムに関して利用可能な統計を、次の表に示します。

表 8–23 Java SE のランタイムに関する JVM の監視統計

Statistic	データ型	説明
classpath-current	StringStatistic	システムクラスローダーがクラスファイルの検索時に使用するクラスパス。

表 8–23 Java SE のランタイムに関する JVM の監視統計 (続き)

Statistic	データ型	説明
inputarguments-current	StringStatistic	JVM に渡される入力引数 (main メソッドへの引数は含まない)。
managementspecversion-current	StringStatistic	JVM で実装される管理仕様のバージョン。
name-current	StringStatistic	実行中の JVM を表す名前
specname-current	StringStatistic	JVM 仕様の名前。
specvendor-current	StringStatistic	JVM 仕様のベンダー。
specversion-current	StringStatistic	JVM 仕様のバージョン。
uptime	CountStatistic	JVM の稼働時間 (ミリ秒)。
vmname-current	StringStatistic	JVM 実装の名前。
vmvendor-current	StringStatistic	JVM 実装のベンダー。
vmversion-current	StringStatistic	JVM 実装のバージョン。

ネットワークの統計

ネットワークは、134 ページの「ネットワークのツリー階層」に示したオブジェクトツリー内に含まれます。

ネットワークの統計を、次の節で説明します。

- 160 ページの「ネットワークキープアライブの統計」
- 161 ページの「ネットワーク接続キューの統計 」
- 162 ページの「ネットワークファイルキャッシュの統計」
- 163 ページの「ネットワークスレッドプールの統計」

ネットワークキープアライブの統計

ネットワークキープアライブの統計では、次のドット表記名パターンを使用します。

`server.network.`*type-of-listener*`.keep-alive.`*statistic*

ネットワークキープアライブに関して利用可能な統計を、次の表に示します。

表 8–24 ネットワークキープアライブの統計

Statistic	データ型	説明
countconnections	CountStatistic	キープアライブモードの接続の数。

表 8–24 ネットワークキープアライブの統計 (続き)

Statistic	データ型	説明
counttimeouts	CountStatistic	タイムアウトしたキープアライブ接続の数。
secondstimeouts	CountStatistic	キープアライブのタイムアウト値 (秒)。
maxrequests	CountStatistic	1 つのキープアライブ接続で許可されている要求の最大数。
countflushes	CountStatistic	閉じられたキープアライブ接続の数。
counthits	CountStatistic	キープアライブモードの接続で受信した要求の数。
countrefusals	CountStatistic	拒否されたキープアライブ接続の数。

ネットワーク接続キューの統計

ネットワーク接続キューの統計では、次のドット表記名パターンを使用します。

server.network.*type-of-listener*.connection-queue.*statistic*

ネットワーク接続キューに関して利用可能な統計を、次の表に示します。

表 8–25 ネットワーク接続キューの統計

Statistic	データ型	説明
countopenconnections	CountStatistic	開いている接続またはアクティブな接続の数。
countoverflows	CountStatistic	キューがいっぱいになったために接続を格納できなかった回数。
countqueued	CountStatistic	キュー内に現在存在している接続の数。
countqueued15minutesaverage	CountStatistic	キューに格納されている接続数の過去 15 分間における平均値。
countqueued1minuteaverage	CountStatistic	キューに格納されている接続数の過去 1 分間における平均値。
countqueued5minutesaverage	CountStatistic	キューに格納されている接続数の過去 5 分間における平均値。
counttotalconnections	CountStatistic	受け付けられた接続の合計数。
counttotalqueued	CountStatistic	キューに格納された接続の合計数。
maxqueued	CountStatistic	接続キューの最大サイズ。

表 8–25 ネットワーク接続キューの統計 (続き)

Statistic	データ型	説明
peakqueued	CountStatistic	キュー内に同時に存在していた接続の最大数。
tickstotalqueued	CountStatistic	接続がキュー内で費やした合計ティック数 (未サポート)。

ネットワークファイルキャッシュの統計

ネットワークファイルキャッシュの統計では、次のドット表記名パターンを使用します。

server.network.*type-of-listener*.file-cache.*statistic*

ネットワークファイルキャッシュに関して利用可能な統計を、次の表に示します。

表 8–26 ネットワークファイルキャッシュの統計

Statistic	データ型	説明
contenthits	CountStatistic	キャッシュファイルコンテンツのヒット数。
contentmisses	CountStatistic	キャッシュファイルコンテンツの失敗数。
heapsize	CountStatistic	現在のキャッシュサイズ (バイト)。
hits	CountStatistic	キャッシュ検索のヒット数。
infohits	CountStatistic	キャッシュファイル情報のヒット数。
infomisses	CountStatistic	キャッシュファイル情報の失敗数。
mappedmemorysize	CountStatistic	キャッシュ用に割り当てられたメモリーのサイズ (バイト)。
maxheapsize	CountStatistic	キャッシュ用のヒープ領域の最大サイズ (バイト)。
maxmappedmemorysize	CountStatistic	キャッシュ用の最大メモリーマップサイズ (バイト)。
misses	CountStatistic	キャッシュ検索に失敗したデータタイプの数。
opencacheentries	CountStatistic	現在開いているキャッシュエントリの数。

ネットワークスレッドプールの統計

ネットワークスレッドプールの統計では、次のドット表記名パターンを使用します。

`server.network.`*type-of-listener*`.thread-pool.`*statistic*

ネットワークスレッドプールに関して利用可能な統計を、次の表に示します。

表 8–27　ネットワークスレッドプールの統計

Statistic	データ型	説明
corethreads	CountStatistic	スレッドプールに含まれるスレッドのコア数。
currentthreadcount	CountStatistic	リスナースレッドプール内に現在存在している要求処理スレッドの数。
currentthreadsbusy	CountStatistic	要求処理用リスナースレッドプール内で現在使用されている要求処理スレッドの数。
maxthreads	CountStatistic	スレッドプールで許可されているスレッドの最大数。
totalexecutedtasks	CountStatistic	スレッドプールで実行されたタスクの合計数。

ORB の統計 (接続マネージャー)

ORB は、134 ページの「ORB のツリー階層」に示したオブジェクトツリー内に含まれます。

ORB の統計では、次のドット表記名パターンを使用します。

`server.orb.transport.connectioncache.inbound.`*statistic*
`server.orb.transport.connectioncache.outbound.`*statistic*

ORB の接続マネージャーに関して利用可能な統計を、次の表に示します。

表 8–28　ORB の監視統計 (接続マネージャー)

Statistic	データ型	説明
connectionsidle	CountStatistic	ORB への接続のうち、アイドル状態の接続の合計数。
connectionsinuse	CountStatistic	ORB への接続のうち、使用中の接続の合計数。
totalconnections	BoundedRangeStatistic	ORB への接続の合計数。

リソースの統計 (接続プール)

接続プールリソースを監視することで、実行時にパフォーマンスの測定やリソースの使用状況の取得を行えます。接続は負荷が大きく、アプリケーションでは頻繁にパフォーマンスのボトルネックとなります。接続プールの解放状況と新しい接続の作成状況、および特定のプールから接続を取得するために待機中であるスレッドの数を監視することが重要です。

接続プールリソースは、135 ページの「リソースのツリー階層」に示したオブジェクトツリー内に含まれます。

接続プールの統計では、次のドット表記名パターンを使用します。

`server.resources.`*`connection-pool`*`.`*`statistic`*

接続プールの統計を、次の表に示します。

表 8–29　リソースの監視統計 (接続プール)

Statistic	データ型	説明
averageconnwaittime	CountStatistic	成功した接続要求あたりの平均待ち時間。
connrequestwaittime	RangeStatistic	前回のサンプリング以降の、接続要求の最長待ち時間と最短待ち時間 (ミリ秒)。現在の値は、プールで処理された直前の要求の待ち時間を示します。
numconnfailedvalidation	CountStatistic	開始時刻から前回のサンプリング時刻までの間に検証に失敗した接続プール内の接続数。
numconnused	RangeStatistic	現在使用されている接続数と、過去に使用された接続の最大数 (ハイウォーターマーク) に関する情報。
numconnfree	RangeStatistic	前回のサンプリング時点におけるプール内の未使用の接続の数。
numconntimedout	CountStatistic	開始時刻から前回のサンプリング時刻までの間にタイムアウトしたプール内の接続の数。
numconncreated	CountStatistic	前回のリセット後にプールによって作成された物理接続の数。
numconndestroyed	CountStatistic	前回のリセット後に破棄された物理接続の数。
numconnacquired	CountStatistic	前回のサンプリング以降に、プールから取得された論理接続の数。

表 8–29 リソースの監視統計 (接続プール) (続き)

Statistic	データ型	説明
numconnreleased	CountStatistic	前回のサンプリング以降に、プールに戻された接続の数。
numconnnotsuccessfullymatched	CountStatistic	マッチング中に拒否された接続の数。
numconnsuccessfullymatched	CountStatistic	マッチングに成功した接続の数。
numpotentialconnleak	CountStatistic	潜在的な接続リークの数。
waitqueuelength	CountStatistic	キュー内で処理されるのを待機している接続要求の数。

セキュリティーの統計

セキュリティーは、135 ページの「セキュリティーのツリー階層」に示したオブジェクトツリー内に含まれます。

セキュリティーに関して利用可能な統計を、次の節で説明します。

- 165 ページの「EJB セキュリティーの統計」
- 165 ページの「Web セキュリティーの統計」
- 166 ページの「レルムセキュリティーの統計」

EJB セキュリティーの統計

EJB セキュリティーの統計では、次のドット表記名パターンを使用します。

`server.security.ejb.`*statistic*

EJB セキュリティーに関して利用可能な統計を、次の表に示します。

表 8–30 EJB セキュリティーの監視統計

Statistic	データ型	説明
policyconfigurationcount	CountStatistic	ポリシー構成の数。
securitymanagercount	CountStatistic	EJB セキュリティーマネージャーの数。

Web セキュリティーの統計

Web セキュリティーの統計では、次のドット表記名パターンを使用します。

`server.security.web.`*statistic*

Web セキュリティーに関して利用可能な統計を、次の表に示します。

表 8–31 Web セキュリティーの監視統計

Statistic	データ型	説明
websecuritymanagercount	CountStatistic	セキュリティーマネージャーの数。
webpolicyconfigurationcount	CountStatistic	ポリシー構成オブジェクトの数。

レルムセキュリティーの統計

レルムセキュリティーの統計では、次のドット表記名パターンを使用します。

server.security.realm.*statistic*

レルムセキュリティーに関して利用可能な統計を、次の表に示します。

表 8–32 レルムセキュリティーの監視統計

Statistic	データ型	説明
realmcount	CountStatistic	レルムの数。

スレッドプールの統計

スレッドプールは、135 ページの「スレッドプールのツリー階層」に示したオブジェクトツリー内に含まれます。

スレッドプールに関して利用可能な統計を、次の節で説明します。

- 166 ページの「スレッドプールの監視統計」
- 167 ページの「Java SE に関する JVM の統計 - スレッド情報」

スレッドプールの監視統計

スレッドプールの統計では、次のドット表記名パターンを使用します。

server.thread-pool.*thread-pool*.*statistic*

スレッドプールに関して利用可能な統計を、次の表に示します。

表 8–33 スレッドプールの監視統計

Statistic	データ型	説明
averagetimeinqueue	BoundedRangeStatistic	キュー内の要求が処理されるまでの平均待ち時間 (ミリ秒)。
averageworkcompletiontime	BoundedRangeStatistic	割り当てが完了するまでの平均時間 (ミリ秒)。

表 8–33 スレッドプールの監視統計 *(続き)*

Statistic	データ型	説明
currentbusythreads	CountStatistic	ビジースレッドの数。
currentnumberofthreads	BoundedRangeStatistic	要求処理スレッドの現在の数。
numberofavailablethreads	CountStatistic	使用可能なスレッドの数。
numberofworkitemsinqueue	BoundedRangeStatistic	キューで待機している作業項目の現在の数。
totalworkitemsadded	CountStatistic	前回のサンプリング以降に、作業キューに追加された作業項目の合計。

Java SE に関する JVM の統計 - スレッド情報

Java SE の JVM で利用可能な `ThreadInfo` 関連の統計を、次の図に示します。

表 8–34 Java SE に関する JVM の監視統計 - スレッド情報

Statistic	データ型	説明
blockedcount	CountStatistic	このスレッドが `BLOCKED` 状態に入った合計回数。
blockedtime	CountStatistic	このスレッドが `BLOCKED` 状態に入ったあと経過した時間 (ミリ秒)。スレッド競合監視が無効になっている場合は、-1 が返されます。
lockname	StringStatistic	このスレッドが獲得をブロックされている監視ロック、またはこのスレッドが `Object.wait` メソッド経由で通知されるのを待っている監視ロックの文字列表現。
lockownerid	CountStatistic	このスレッドのブロック対象オブジェクトの監視ロックを保持しているスレッドの ID。
lockownername	StringStatistic	このスレッドのブロック対象オブジェクトの監視ロックを保持しているスレッドの名前。
stacktrace	StringStatistic	このスレッドに関連付けられているスタックトレース。
threadid	CountStatistic	スレッドの ID。
threadname	StringStatistic	スレッドの名前
threadstate	StringStatistic	スレッドの状態。

表 8–34 Java SE に関する JVM の監視統計 - スレッド情報 (続き)

Statistic	データ型	説明
waitedtime	CountStatistic	スレッドが WAITING 状態に入ったあと経過した時間 (ミリ秒)。スレッド競合監視が無効になっている場合は、-1 が返されます。
waitedcount	CountStatistic	スレッドが WAITING 状態または TIMED_WAITING 状態になった合計回数。

トランザクションサービスの統計

トランザクションサービスを使用すると、クライアントはトランザクションサブシステムをフリーズして、トランザクションをロールバックしたり、フリーズ時点で処理中であったトランザクションを特定することができます。トランザクションサービスは、136 ページの「トランザクションサービスのツリー階層」に示したオブジェクトツリー内に含まれます。

トランザクションサービスの統計では、次のドット表記名パターンを使用します。

server.transaction-service.*statistic*

トランザクションサービスに関して利用可能な統計を、次の表に示します。

表 8–35 トランザクションサービスの監視統計

Statistic	データ型	説明
activecount	CountStatistic	現在アクティブなトランザクションの数。
activeids	StringStatistic	現在アクティブなトランザクションの ID。それらの各トランザクションは、トランザクションサービスのフリーズ後にロールバックすることができます。
committedcount	CountStatistic	コミットされたトランザクションの数。
rolledbackcount	CountStatistic	ロールバックされたトランザクションの数。
state	StringStatistic	トランザクションがフリーズされたかどうかを示します。

Web の統計

Web モジュールは、136 ページの「Web のツリー階層」に示したオブジェクトツリー内に含まれます。

利用可能な Web の統計を、次の節で説明します。

- 169 ページの「Web モジュールサーブレットの統計」

Web モジュールサーブレットの統計

Web モジュールサーブレットの統計では、次のドット表記名パターンを使用します。

`server.applications.`*web-module*`.`*virtual-server*`.servlet.`*statistic*
`server.applications.`*application*`.`*web-module*`.`*virtual-server*`.servlet.`*statistic*

利用可能な Web モジュールサーブレットの統計を、次の表に示します。

表 8–36　Web モジュールサーブレットの統計

Statistic	データ型	説明
`errorcount`	CountStatistic	応答コードが 400 以上になった場合の累計件数。
`maxtime`	CountStatistic	Web コンテナの要求待ち状態の最大継続時間。
`processingtime`	CountStatistic	各要求の処理に要した時間の累計値。この処理時間は、要求処理時間を要求数で割って得られた平均値です。
`requestcount`	CountStatistic	その時点までに処理された要求の合計数。
`servicetime`	CountStatistic	応答時間の総計 (ミリ秒)。

Web JSP の統計

Web JSP の統計では、次のドット表記名パターンを使用します。

`server.applications.`*web-module*`.`*virtual-server*`.`*statistic*
`server.applications.`*application*`.`*web-module*`.`*virtual-server*`.`*statistic*

利用可能な Web JSP 統計を、次の表に示します。

表 8–37　Web JSP の監視統計

Statistic	データ型	説明
`jspcount-current`	RangeStatistic	アクティブな JSP ページの数。
`jsperrorcount`	CountStatistic	JSP ページの呼び出しでトリガーされたエラーの合計数。

表 8–37 Web JSP の監視統計 (続き)

Statistic	データ型	説明
jspreloadedcount	CountStatistic	再読み込みされた JSP ページの合計数。
totaljspcount	CountStatistic	これまでに読み込まれた JSP ページの合計数。

Web 要求の統計

Web 要求の統計では、次のドット表記名パターンを使用します。

server.applications.*web-module*.*virtual-server*.*statistic*
server.applications.*application*.*web-module*.*virtual-server*.*statistic*

利用可能な Web 要求統計を、次の表に示します。

表 8–38 Web 要求の監視統計

Statistic	データ型	説明
errorcount	CountStatistic	エラー回数の累計値。エラー回数は、応答コードが 400 以上になった場合の回数を表します。
maxtime	CountStatistic	要求あたりの最長応答時間です。累積値ではなく、応答時間の中で最大の値です。
processingtime	CountStatistic	平均要求処理時間 (ミリ秒)。
requestcount	CountStatistic	現時点までに処理された要求の累積数。

Web サーブレットの統計

Web サーブレットの統計では、次のドット表記名パターンを使用します。

server.applications.*web-module*.*virtual-server*.*statistic*
server.applications.*application*.*web-module*.*virtual-server*.*statistic*

利用可能な Web サーブレットの統計を、次の表に示します。

表 8–39 Web サーブレットの監視統計

Statistic	データ型	説明
activeservletsloadedcount	RangeStatistic	現在読み込まれているサーブレットの数。
servletprocessingtimes	CountStatistic	サーブレット処理時間の累積値 (ミリ秒)。

表 8–39 Web サーブレットの監視統計 (続き)

Statistic	データ型	説明
totalservletsloadedcount	CountStatistic	Web モジュールに読み込まれたサーブレットの累積数。

Web セッションの統計

Web セッションの統計では、次のドット表記名パターンを使用します。

server.applications.*web-module*.*virtual-server*.*statistic*
server.applications.*application*.*web-module*.*virtual-server*.*statistic*

利用可能な Web セッションの統計を、次の表に示します。

表 8–40 Web セッションの監視統計

Statistic	データ型	説明
activatedsessionstotal	CountStatistic	アクティブ化されたセッションの合計数。
activesessionscurrent	RangeStatistic	現在のアクティブセッションの数。
activesessionshigh	CountStatistic	現在のアクティブセッションの最大数。
expiredsessionstotal	CountStatistic	期限切れセッションの合計数。
passivatedsessionstotal	CountStatistic	パッシブ化されたセッションの合計数。
persistedsessionstotal	CountStatistic	持続化されたセッションの合計数。
rejectedsessionstotal	CountStatistic	拒否されたセッションの合計数。
sessionstotal	CountStatistic	作成されたセッションの合計数。

Enterprise Server の監視データを表示するための JConsole の設定

Java SE には、MBean Server に接続し、サーバーに登録されている MBean を表示するためのツールが用意されています。JConsole は一般的な JMX コネクタクライアントであり、標準 Java SE ディストリビューションの一部として利用できます。Enterprise Server で使用できるように JConsole を設定すると、Enterprise Server は JMX コネクタのサーバー側となり、JConsole は JMX コネクタのクライアント側となります。

▼ JConsole を Enterprise Server に接続する

Java SE 6 では、Platform MBean Server を含めること、および仮想マシンを設定するための管理対象 Bean (MBean) を含めることにより、仮想マシンの管理と監視を拡張します。

すべての MBean を表示するために、Enterprise Server にはシステム JMX コネクタサーバーという標準 JMX コネクタサーバーの構成が用意されています。Enterprise Server の起動時に、この JMX コネクタサーバーのインスタンスが起動します。規格に準拠する JMX コネクタクライアントはすべて、JMX コネクタサーバーを使用してサーバーに接続できます。

デフォルトでは、Enterprise Server はセキュリティー保護されていないシステム JMX コネクタサーバーを使用するように設定されています。この設定で問題がある場合は、JMX コネクタを削除できます。ただし、`address` を `locahost` に設定することにより、特定の IP アドレス (たとえば、ループバックアドレス) へのアクセスが制限される場合があります。

1. ドメインを起動します。
   手順については、93 ページの「ドメインの起動」を参照してください。

2. *JDK_HOME* `/bin/jconsole` の形式を使用して、**JConsole** を起動します。
   次に例を示します。

   ```
   /usr/java/bin/jconsole
   ```

   JConsole の「Connect to Agent」ウィンドウが表示されます。

3. 「リモート」タブをクリックし、ホスト名とポート番号を入力します。
   JConsole には常にリモートで接続してください。リモート以外で接続した場合は、MBean が自動的に読み込まれません。

4. 「接続」をクリックします。

5. 「**Remote Process**」テキストボックスに、**JMX** サービス **URL** を指定します。
   次に例を示します。

   ```
   service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
   ```

   JMX サービス URL は、サーバーによって起動時に発行され、次のような形式になります。

   ```
   [#|2009-12-03T10:25:17.737-0800|INFO|glassfishv3.0|
   javax.enterprise.system.tools.admin.org.glassfish.server|_ThreadID=20;
   _ThreadName=Thread-26;|JMXStartupService: Started JMXConnector, JMXService
   URL = service:jmx:rmi://localhost:8686/jndi/rmi://localhost:8686/jmxrmi|#]
   ```

ただし、ほとんどの場合は、簡単に `host:port` (192.168.1.150:8686 など) を入力するだけで十分です。長いサービス URL は必要ありません。

注 – `localhost` の代わりに別のホスト名を指定することもできます。`jmx-connector` の構成が変更されている場合は、デフォルトのポート番号 (8686) が変更されている場合があります。

---

6 「接続」をクリックします。

JConsole ウィンドウの各種タブに、MBean、JVM 情報などが表示されます。監視に利用できる MBean のほとんどは、`amx` および `java.lang` ドメインにあります。

参照 JConsole の詳細は、`http://java.sun.com/javase/6/docs/technotes/guides/management/jconsole.html` を参照してください。

◆ ◆ ◆ 第 9 章

ライフサイクルモジュールの管理

この章では、Sun GlassFish™Enterprise Server v3 環境でライフサイクルモジュールを管理する手順について説明します。

ここでは、次のテーマを取り上げます。

本章で説明するタスクを管理コンソールから実行する手順については、管理コンソールのオンラインヘルプを参照してください。

ライフサイクルモジュールについて

「ライフサイクルモジュール」は初期化サービスとも呼ばれ、Enterprise Server 環境内で短期的または長期的な Java ベースタスクを実行する手段として利用できます。これらのモジュールはサーバーの起動時に自動的に開始され、サーバーのライフサイクルのさまざまな段階で通知を受け取ります。ライフサイクルモジュールの設定済みのプロパティーは、サーバーの初期化中にプロパティーとして渡されます。

ライフサイクルモジュールのすべてのクラスとインタフェースは、*as-install* `/glassfish/modules/glassfish-api.jar` ファイルに格納されています。

ライフサイクルモジュールは次の Enterprise Server イベントを待機し、イベントに応じてタスクを実行します。

1. 「初期化」。サーバーは構成を読み取り、組み込みサブシステム (セキュリティーサービスやロギングサービス) を初期化して、コンテナを作成します。
2. 「起動」。サーバーは配備されたアプリケーションを読み込んで初期化します。
3. 「実行可能」。サーバーは要求の処理を開始します。

4. 「シャットダウン」。サーバーはアプリケーションをシャットダウンして停止します。
5. 「終了」。サーバーはコンテナ、組み込みサブシステム、およびサーバー実行環境を終了します。

これらのイベントは `LifecycleEvent` クラスで定義されます。ライフサイクルモジュールの作成については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 13 章「Developing Lifecycle Listeners」を参照してください。

注 – `is-failure-fatal` 設定が true に設定されている場合は (デフォルトでは false)、ライフサイクルモジュールで障害が発生したときにサーバーの初期化または起動が中止されます。ただし、シャットダウンまたは終了は実行されます。

ライフサイクルモジュールの設定

ここでは、次のテーマを取り上げます。

▼ ライフサイクルモジュールを作成する

ライフサイクルモジュールを作成するには、リモートモードで `create-lifecycle-module` サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 `create-lifecycle-module(1)` サブコマンドを使用して、新しいライフサイクルモジュールを作成します。
サブコマンドのオプションとプロパティーについては、このマニュアルページに記載されています。

3 サーバーを再起動して、変更を有効にします。
94 ページの「ドメインの再起動」を参照してください。

例 9–1 ライフサイクルモジュールの作成

この例では、`customSetup` というライフサイクルモジュールを作成します。

```
asadmin> create-lifecycle-module --classname "com.acme.CustomSetup"
--classpath "/export/customSetup" --loadorder 1 --failurefatal=true
--description "this is a sample customSetup"
--property rmi="Server\=acme1\:7070":timeout=30 customSetup
Command create-lifecycle-module executed successfully
```

参照 コマンド行に asadmin help create-lifecycle-module と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ ライフサイクルモジュールを一覧表示する

既存のライフサイクルモジュールを一覧表示するには、リモートモードで list-lifecycle-modules サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-lifecycle-modules(1) サブコマンドを使用して、ライフサイクルモジュールを一覧表示します。

例 9–2 ライフサイクルモジュールの一覧表示

この例では、既存のライフサイクルモジュールを一覧表示します。

```
asadmin> list-lifecycle-modules
WSTCPConnectorLCModule
Command list-lifecycle-modules executed successfully
```

参照 コマンド行に asadmin help list-lifecycle-modules と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ ライフサイクルモジュールを更新する

set サブコマンドを使用して、既存のライフサイクルモジュールを更新します。

1 get(1) サブコマンドを使用して、ライフサイクルモジュールの更新可能なプロパティーを一覧表示します。
次に例を示します (シングルモード)。

```
asadmin get "*" | grep sampleLCM
applications.application.sampleLCMmodule.availability-enabled=false
applications.application.sampleLCMmodule.directory-deployed=false
```

```
applications.application.sampleLCMmodule.enabled=true
applications.application.sampleLCMmodule.name=sampleLCMmodule
applications.application.sampleLCMmodule.object-type=user
applications.application.sampleLCMmodule.property.class-name=example.lc.SampleModule
applications.application.sampleLCMmodule.property.classpath=/build/lcm.jar
applications.application.sampleLCMmodule.property.is-failure-fatal=false
applications.application.sampleLCMmodule.property.isLifecycle=true
```

2 set(1) サブコマンドを使用して、ライフサイクルモジュールを更新します。

3 サーバーを再起動して、変更を有効にします。
94 ページの「ドメインの再起動」を参照してください。

例 9–3 ライフサイクルモジュールの更新

この例では、classpath プロパティーを更新します。

```
sadmin> set applications.application.sampleLCMmodule.
property.classpath=/build/lcm_new.jarapplications.application.
sampleLCMmodule.property.classpath=/build/lcm_new.jarCommand set executed successfully.
```

参照 コマンド行に asadmin help set と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ ライフサイクルモジュールを削除する

ライフサイクルモジュールを削除するには、リモートモードで delete-lifecycle-module サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-lifecycle-modules(1) サブコマンドを使用して、現在のライフサイクルモジュールを一覧表示します。

3 delete-lifecycle-module(1) サブコマンドを使用して、ライフサイクルモジュールを削除します。

例 9–4 ライフサイクルモジュールの削除

この例では、customSetup というライフサイクルモジュールを削除します。

```
asadmin> delete-lifecycle-module customSetup
Command delete-lifecycle-module executed successfully
```

参照　コマンド行に `asadmin help delete-lifecycle-module` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 10 章

10 Enterprise Server の拡張

この章では、配備された Sun GlassFish™ Enterprise Server v3 の拡張および更新に関するタスクを、更新ツールを使用して実行する手順を説明します。pkg コマンドを使用する手順もこの章で説明します。Enterprise Server には、pkg コマンドの使用方法を説明するリファレンスページが用意されています。

ここでは、次のテーマを取り上げます。

この章のタスクを管理コンソールを使用して実行する場合の手順は、管理コンソールオンラインヘルプの更新ツールの項目で説明します。スタンドアロンの更新ツールにも、オンラインヘルプが用意されています。更新ツールの詳細は、http://wikis.sun.com/display/IpsBestPractices/Screenshots を参照してください。

アドオンコンポーネントについて

Enterprise Server は、機能をモジュール形式で提供するように設計されています。したがって、必要な機能だけをインストールして、必要がない機能のインストールは省略することができます。「*OSGi* モジュール」は、バンドルとも呼ばれ、配備した Enterprise Server にアドオン機能を提供します。サーバーの再起動を必要とせず、実行時にアドオンコンポーネントを追加または削除することができます。

時間の経過とともに、新しいアドオンコンポーネントが開発され、既存のコンポーネントは変更されていきます。Enterprise Server では、配備したサーバー上のソフトウェアを更新できるように、一連の Image Packaging System (IPS) ツールが提供されています。更新ツールを次の方法で使用して、新しいアドオンコンポーネンや更新されたアドオンコンポーネントをインストールできます。

- グラフィカルな Enterprise Server 管理コンソール で選択して起動します。

- コマンド行から開始できるスタンドアロンのグラフィカルツールとして起動します。コマンドの形式は、*as-install-parent* `/bin/updatetool` です。
- コマンド行ユーティリティー (`pkg` コマンド) として起動します。グラフィカルな更新ツールで実行できるほとんどのタスクを実行できます。`pkg` コマンドを使用する主な理由は次のとおりです。
  - アップデートスクリプトを作成する際の基礎として使用する
  - モニター、グラフィックカード、またはキーボードが接続されていないシステム (ヘッドレスシステム) で、システムを操作する手段として使用する

グラフィカルなバージョンの更新ツールには、どちらにも詳細なオンラインヘルプが用意されています。

コンポーネントの追加

この節では、`pkg` コマンドを使用して、配備済みの Enterprise Server に Enterprise Server アドオンコンポーネントをインストールする手順について説明します。

▼ アドオンコンポーネントをインストールする

`pkg` コマンドを使用して、アドオンコンポーネントをシステムにインストールできます。複数のバージョンのパッケージを使用できる場合は、指定しないかぎり最新のバージョンが適用されます。`pkg` コマンドは、*as-install-parent*`/bin` ディレクトリに格納されています。

注 – `pkg` コンポーネント、`updatetool` コンポーネント、またはコマンド行から呼び出すその他の有効なコンポーネントが、配備済みの Enterprise Server にまだインストールされていない場合は、コンポーネントをインストールするかどうかを確認するメッセージが表示されます。「Y」を入力すると、コンポーネントがインストールされます。

始める前に　追加コンポーネントをインストールする前に、Enterprise Server v3 が完全に配備されている必要があります。インストールの手順については、『Sun GlassFish Enterprise Server v3 Installation Guide』を参照してください。

1. インストール済みのコンポーネントを一覧表示します。

   `pkg list`

   次のような情報が表示されます。

```
NAME (AUTHORITY)                        VERSION         STATE      UFIX
glassfishv3-common                      0-1             installed  ----
glassfishv3-ejb                         0-1             installed  u---
```

```
glassfishv3-nucleus                          0-1             installed  ----
glassfishv3-web                              0-1             installed  ----
grails                                       1.0-1.0         installed  ----
jersey                                       0.7-0.1         installed  u---
jmaki                                        1.8.0-1.0       installed  ----
jruby                                        1.1.1-1.0       installed  ----
metro                                        1.2-1           installed  u---
pkg                                          0.1.4-6.564     installed  u---
python2.4-minimal                            2.4.4-6.564     installed  u---
updatetool (glassfish.org)                   2.0-6.564       installed  u---
wxpython2.8-minimal                          2.8.7.1-6.564   installed  u---
```

2 使用可能なパッケージをすべて表示します。

```
pkg list -a
```

リポジトリから次のような情報が表示されます。

```
NAME (AUTHORITY)                             VERSION         STATE      UFIX
glassfishv3-common                           0-1             known      u---
glassfishv3-common                           0-1             known      u---
glassfishv3-common                           0-1             installed  ----
glassfishv3-ejb                              0-1             known      u---
glassfishv3-ejb                              0-1             known      u---
glassfishv3-ejb                              0-1             known      u---
glassfishv3-ejb                              0-1             installed  u---
glassfishv3-ejb                              0-1             known      ----
glassfishv3-nucleus                          0-1             known      u---
glassfishv3-nucleus                          0-1             known      u---
glassfishv3-nucleus                          0-1             installed  ----
glassfishv3-web                              0-1             known      u---
glassfishv3-web                              0-1             known      u---
glassfishv3-web                              0-1             installed  ----
grails                                       1.0-1.0         installed  ----
javadb                                       0-1             known      u---
javadb                                       0-1             known      u---
javadb                                       0-1             known      ----
jersey                                       0.7-0.1         installed  u---
jersey                                       0.7-0.2         known      u---
jersey                                       0.7-0.3         known      u---
jersey                                       0.8-0.1         known      ----
jmaki                                        1.8.0-1.0       installed  ----
jruby                                        1.1.1-1.0       installed  ----
metro                                        1.2-1           installed  u---
metro                                        1.2-2           known      u---
metro                                        1.2-3           known      ----
pkg                                          0.1.4-6.564     installed  u---
pkg                                          0.1.5-8.724     known      ----
python2.4-minimal                            2.4.4-6.564     installed  u---
```

```
python2.4-minimal                         2.4.4-8.724     known      ----
updatetool                                2.0-6.564       installed  u---
updatetool                                2.0-8.724       known      ----
wxpython2.8-minimal                       2.8.7.1-6.564   installed  u---
wxpython2.8-minimal                       2.8.7.1-8.724   known      ----
```

3. *as-install*ディレクトリに移動します。

4. 使用可能なパッケージのリストから、パッケージをインストールします。

   `pkg install` *package-name* という構文を使用します。次に例を示します。

   ```
   pkg install javadb
   ```

   コンポーネントの最新のバージョンがインストールされ、次のような情報が表示されます。

```
DOWNLOAD                                  PKGS       FILES     XFER (MB)
javadb                                    0/1      61/200     2.10/7.26

PHASE                                     ACTIONS
Install Phase                             222/222
```

5. 変更内容を適用するために、**Enterprise Server** を再起動します。

   94 ページの「ドメインの再起動」を参照してください。

参照 `pkg` コマンドを使用する際の完全な構文とオプションは、*as-install*`/pkg/man/` ディレクトリにあるリファレンスページで説明されています。これらのリファレンスページは、`man` コマンドでは表示されません。代わりに、`more` または `cat` などのコマンドを使用してください。

インストール済みのコンポーネントの更新

この節では、Enterprise Server のコンポーネントをインストールしたあとに、これらのコンポーネントを更新する次の手順について説明します。

- 184 ページの「インストール済みのコンポーネントを更新する」
- 185 ページの「イメージ内のすべてのインストール済みコンポーネントを更新する」

▼ インストール済みのコンポーネントを更新する

更新されたバージョンのコンポーネントをインストールする場合は、変更されたファイルだけがダウンロードおよびインストールされます。更新されたパッケージで削除されているファイルは、更新プロセス中に削除されます。

1 使用可能な更新があるインストール済みのパッケージだけを一覧表示します。

```
pkg list -u
```

次のような情報が表示されます。

```
NAME (AUTHORITY)                                VERSION         STATE      UFIX
glassfishv3-ejb                                 0-1             installed  u---
jersey                                          0.7-0.1         installed  u---
metro                                           1.2-1           installed  u---
pkg                                             0.1.4-6.564     installed  u---
python2.4-minimal                               2.4.4-6.564     installed  u---
updatetool (glassfish.org)                      2.0-6.564       installed  u---
wxpython2.8-minimal                             2.8.7.1-6.564   installed  u---
```

2 新しいバージョンのパッケージをインストールします。

`pkg install` *package-name* という構文を使用します。次に例を示します。

```
pkg install metro
```

次のような情報が表示されます。

```
DOWNLOAD                                  PKGS       FILES     XFER (MB)
Completed                                 1/1         5/5     0.49/0.49

PHASE                                     ACTIONS
Removal Phase                                2/2
Update Phase                                 7/7
Install Phase                                2/2
```

3 変更内容を適用するために、**Enterprise Server** を再起動します。

94 ページの「ドメインの再起動」を参照してください。

参照 `pkg` コマンドを使用する際の完全な構文とオプションは、*as-install* `/pkg/man/` ディレクトリにあるリファレンスページで説明されています。これらのリファレンスページは、`man` コマンドでは表示されません。代わりに、`more` または `cat` などのコマンドを使用してください。

▼ イメージ内のすべてのインストール済みコンポーネントを更新する

Enterprise Server では、単一のシステムで複数のインストールイメージを保守できます。インストールイメージを更新するときに、イメージ内にあるすべてのコンポーネントは、新しいバージョンを利用できる場合、そのバージョンに更新されます。更新されたバージョンのコンポーネントをインストールするときに、変更され

たファイルだけがダウンロードおよびインストールされます。更新されたパッケージで削除されているファイルは、更新プロセス中に削除されます。

1 イメージのすべてのパッケージをインストールします。

`pkg install` *image-name* の構文を使用します。次に例を示します。

```
pkg image-update
```

次のような情報が表示されます。

```
DOWNLOAD                                  PKGS       FILES     XFER (MB)
Completed                                 6/6      729/729   21.59/21.59

PHASE                                     ACTIONS

Removal Phase                             887/887
Update Phase                              253/253
Install Phase                             584/584
```

2 変更内容を適用するために、**Enterprise Server** を再起動します。

94 ページの「ドメインの再起動」を参照してください。

参照 `pkg` コマンドを使用する際の完全な構文とオプションは、*as-install* `/pkg/man/` ディレクトリにあるリファレンスページで説明されています。これらのリファレンスページは、`man` コマンドでは表示されません。代わりに、`more` または `cat` などのコマンドを使用してください。

インストール済みのコンポーネントの削除

使用していないコンポーネントをシステムから削除する場合は、`uninstall` コマンドを使用します。コンポーネントを以前のバージョンに戻す必要がある場合は、現在のバージョンをアンインストールしたあとに、バージョン番号を指定して以前のバージョンをインストールする必要があります。

- 186 ページの「インストール済みのコンポーネントをアンインストールする」
- 187 ページの「コンポーネントをアンインストールして古いバージョンに戻す」

▼ インストール済みのコンポーネントをアンインストールする

始める前に インストール済みのコンポーネントを削除する前に、削除するコンポーネントに依存関係がないことを確認します。

1 インストール済みのコンポーネントをすべて表示します。

```
pkg list
```

```
NAME (AUTHORITY)                        VERSION          STATE      UFIX
glassfishv3-common                      0-1              installed  ----
glassfishv3-ejb                         0-1              installed  u---
glassfishv3-nucleus                     0-1              installed  ----
glassfishv3-web                         0-1              installed  ----
grails                                  1.0-1.0          installed  ----
jersey                                  0.7-0.1          installed  u---
jmaki                                   1.8.0-1.0        installed  ----
jruby                                   1.1.1-1.0        installed  ----
metro                                   1.2-1            installed  u---
pkg                                     0.1.4-6.564      installed  u---
python2.4-minimal                       2.4.4-6.564      installed  u---
updatetool (glassfish.org)              2.0-6.564        installed  u---
wxpython2.8-minimal                     2.8.7.1-6.564    installed  u---
```

2 システムから削除するコンポーネントをアンインストールします。

pkg uninstall *package-name* の構文を使用します。次に例を示します。

```
pkg uninstall jruby
```

3 変更内容を適用するために、**Enterprise Server** を再起動します。

94 ページの「ドメインの再起動」を参照してください。

参照 pkg コマンドを使用する際の完全な構文とオプションは、*as-install* /pkg/man/ ディレクトリにあるリファレンスページで説明されています。これらのリファレンスページは、man コマンドでは表示されません。代わりに、more または cat などのコマンドを使用してください。

▼ コンポーネントをアンインストールして古いバージョンに戻す

インストールしたコンポーネントが正しく動作しない場合は、コンポーネントを古いバージョンに戻すことができます。古いバージョンのコンポーネントを復元するには、現在のバージョンのコンポーネントをアンインストールしたあと、元に戻す古いバージョンをインストールします。

始める前に 現在のバージョンのコンポーネントをアンインストールする前に、古いバージョンのコンポーネントがリポジトリにあることを確認する必要があります。

1 古いバージョンのコンポーネントがまだ使用可能であることを確認します。

```
pkg list -a
```

2. インストール済みのコンポーネントを一覧表示します。

   ```
   pkg list
   ```

3. 置換対象の現在インストールされているコンポーネントをアンインストールします。

   `pkg uninstall` *package-name* の構文を使用します。次に例を示します。

   ```
   pkg uninstall jersey
   ```

4. 古いバージョンのコンポーネントをインストールします。

   `pkg install` *package-name* `–version` *version-number* の構文を使用します。次に例を示します。

   ```
   pkg install jersey -version 0.7-0.2
   ```

5. 古いバージョンがインストールされたことを確認します。

   ```
   pkg list
   ```

参照 `pkg` コマンドを使用する際の完全な構文とオプションは、*as-install* `/pkg/man/` ディレクトリにあるリファレンスページで説明されています。これらのリファレンスページは、`man` コマンドでは表示されません。代わりに、`more` または `cat` などのコマンドを使用してください。

パート II

セキュリティー管理

◆ ◆ ◆ 第 11 章

11 システムセキュリティーの管理

ここでは、次のテーマを取り上げます。

これらのタスクの多くを 管理コンソール を使用して実行する場合の手順は、管理コンソール のオンラインヘルプで説明します。

セキュリティーの設定に関するその他の手順は、第 12 章「ユーザーセキュリティーの管理」および第 13 章「メッセージセキュリティーの管理」で説明します。

アプリケーションのセキュリティーについては、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 5 章「Securing Applications」で説明します。

Enterprise Server のシステムセキュリティーについて

セキュリティーとはデータを保護すること、つまり、記憶領域のデータまたは搬送中のデータに対する許可されていないアクセスや損傷を防ぐための方法です。Enterprise Server は Java セキュリティーモデルをベースに構築され、アプリケーションが安全に動作できるサンドボックスを使用するため、システムやユーザーにリスクが及ぶ可能性がありません。「システムセキュリティー」は、Enterprise Server 環境内のすべてのアプリケーションに影響します。

システムセキュリティーには次のような機能があります。

■ 200 ページの「システムセキュリティー管理用ツール」

認証

「認証」とは、あるエンティティー (ユーザー、アプリケーション、またはコンポーネント) が、別のエンティティーが主張している本人であることを確認する方法です。エンティティーは、セキュリティー「資格」を使用して自らを認証します。資格には、ユーザー名、パスワード、デジタル証明書などが含まれます。通常、サーバーまたはアプリケーションはクライアントに自身を認証するように要求します。また、クライアントがサーバーに自身を認証するように要求する場合もあります。双方向で認証する場合は、これを「相互認証」と呼びます。

エンティティーが保護対象リソースにアクセスを試行する場合、Enterprise Server はそのリソースに対して設定されている認証メカニズムを使用してアクセスを認可するかどうかを決定します。たとえば、ユーザーが Web ブラウザでユーザー名およびパスワードを入力でき、アプリケーションがその資格を確認する場合、そのユーザーは認証されます。それ以降のセッションで、ユーザーはこの認証済みのセキュリティー ID に関連付けられます。

認証のタイプ

アプリケーションは、使用する認証のタイプを配備記述子内で指定します。Enterprise Server では、次のタイプの認証がサポートされます。

BASIC	サーバーに組み込まれているログインダイアログボックスを使用します。通信プロトコルは HTTP です (SSL を指定可能)。SSL を使用しなければ、ユーザー証明書は暗号化されません
FORM	アプリケーションが独自仕様のカスタムログインおよびエラーページを提供します。通信プロトコルは HTTP です (SSL を指定可能)。SSL を使用しなければ、ユーザー証明書は暗号化されません。
CLIENT-CERT	サーバーは公開鍵証明書を使用してクライアントを認証します。通信プロトコルは HTTPS (HTTP over SSL) です。ユーザー認定された暗号化は SSL です。
DIGEST	サーバーは、ユーザー名とパスワードに基づいてユーザーを認証します。認証はパスワードを暗号化された形式で送信することによって実行され、この暗号化形式は BASIC 認証で使用される単純な Base64 エンコーディングよりも安全です。通信プロトコルは HTTPS です。

パスワード

パスワードは、Enterprise Server のコンポーネントおよびデータへの許可されていないアクセスを防ぐもっとも重要な手段です。Enterprise Server でパスワードを使用する方法については、201 ページの「パスワードの管理」を参照してください。

マスターパスワードとキーストア

「マスターパスワード」は、全体で共有されるパスワードで、システムでもっとも重要なデータです。これを認証に使用したり、ネットワークを介して送信したりすることは決してありません。マスターパスワードは要求されたときに手動で入力するか、ファイルに隠蔽化することができます。

マスターパスワードは、セキュリティー保護されたキーストアのパスワードです。新しいアプリケーションサーバードメインが作成されると、新しい自己署名付き証明書が生成されて、関連キーストアに格納されます。このキーストアは、マスターパスワード (デフォルトのパスワードは `changeit`) を使用してロックされます。マスターパスワードが変更済みで、デフォルト以外の値に設定されている場合は、マスターパスワードの入力を要求されます。正しいマスターパスワードを入力したあと、ドメインが起動します。

管理用パスワード

管理パスワードは、管理コンソールおよび `asadmin` ユーティリティーの呼び出しに使用されます。通常、このパスワードはインストール時に設定されますが、変更可能です。手順については、203 ページの「管理パスワードを変更する」を参照してください。

符号化されたパスワード

符号化されたパスワードを含むファイルは、ファイルシステムのアクセス権を使用して保護する必要があります。これらのファイルには次のものが含まれます。

- *domain-dir*`/master-password`

  このファイルにはエンコード化されたマスターパスワードが含まれているので、ファイルシステムのアクセス権 600 で保護する必要があります。
- `asadmin` ユーティリティーに `--passwordfile` 引数を使用して渡すために作成された、すべてのパスワードファイル。これらは、ファイルシステムのアクセス権 600 で保護する必要があります。

手順については、204 ページの「パスワードをファイルから設定する」を参照してください。

パスワードエイリアス

パスワードをドメイン構成ファイルに平文で保存するのを避けるために、パスワードのエイリアスを作成できます。この処理は、パスワードの「暗号化」とも呼ばれます。詳細は、205 ページの「パスワードエイリアスの管理」を参照してください。

シングルサインオン

「シングルサインオン」によって、1 つのアプリケーションにログインしたユーザーは、同じ認証情報が必要なほかのアプリケーションに暗黙的にログインするようになります。シングルサインオンはグループに基づいています。配備記述子が同じグループを定義し、かつ同じ認証方法 (BASIC、FORM、または CLIENT-CERT) を使用するすべての Web アプリケーションは、シングルサインオンを共有します。

Enterprise Server では、仮想サーバーのシングルサインオンがデフォルトで有効に設定されます。これにより、同じ仮想サーバー内の複数のアプリケーションが、ユーザーの認証状態を共有できます。

承認

「承認」は、アクセス制御とも呼ばれ、データにアクセスする許可または操作を実行する許可を、ユーザーに与えるための手段です。ユーザーが承認されたあと、ユーザーの承認のレベルにより、その所有者が実行できる操作の範囲が決定されます。ユーザーの承認は、ユーザーのロールに基づきます。

ロール

「ロール」は、ユーザーがアクセスできるアプリケーションおよび各アプリケーションの部分、およびこれらのユーザーまたはグループがアプリケーションで実行できる操作の内容を定義します。たとえば、人事アプリケーションの場合、すべての社員は電話番号とメールアドレスの情報を表示できますが、給与情報にアクセスできるのは管理職だけです。このアプリケーションでは、`employee` と `manager` の少なくとも 2 つのロールを定義しています。`manager` ロールのユーザーだけが、給与情報の表示を許可されています。

ロールはアプリケーション内での役割を定義するのに対し、グループはある方法で関連付けられているユーザーの集まりに過ぎません。この点で、ロールとグループは異なります。たとえば、人事アプリケーションで、`full-time`、`part-time`、および `on-leave` といったグループがあるとします。これらのグループのユーザーは、すべて社員 (`employee` ロール) です。これに加え、各ユーザーには追加の雇用レベルを定義する各自の役職が指定されます。

ロールは、アプリケーションの配備記述子内で定義されます。アプリケーションの開発者または配備担当者は、各アプリケーションの配備記述子で、ロールを 1 つまたは複数のグループにマッピングします。アプリケーションがパッケージ化されて配備される場合、次の図に例示されているように、アプリケーションはユーザーまたはグループとロールとの間のマッピングを指定します。

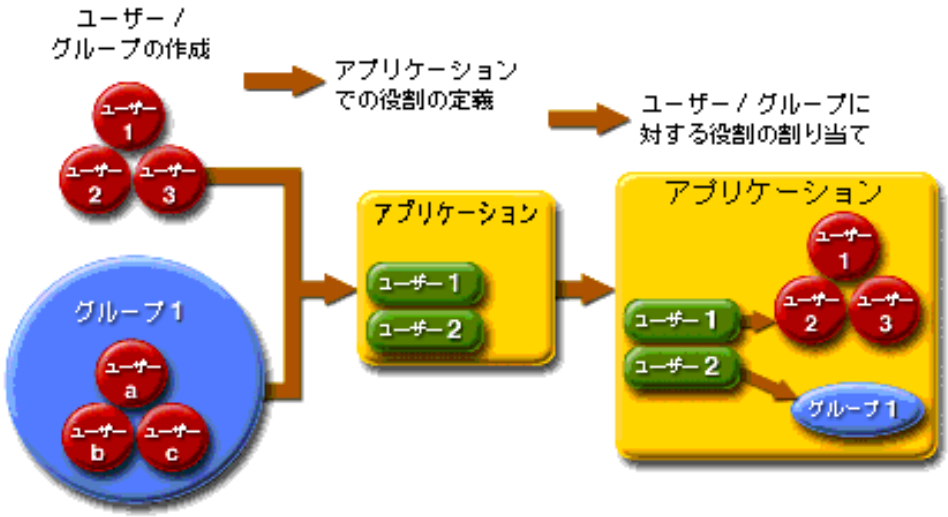

図 11–1　ロールマッピング

Java Authorization Contract for Containers

JACC (Java Authorization Contract for Containers) は Java EE 仕様の一部で、プラグイン可能な承認プロバイダ用のインタフェースを定義します。これにより、認証を行うためにサードパーティー製のプラグインモジュールを設定できます。デフォルトで、Enterprise Server は JACC 仕様に準拠する単純なファイルベースの承認エンジンを提供します。サードパーティー製の JACC プロバイダを追加指定することもできます。

JACC プロバイダは JAAS (Java Authentication and Authorization Service) の API を使用します。JAAS によって、サービスが認証およびユーザーに対するアクセス制御を行うことが可能になります。JAAS は、標準 PAM (Pluggable Authentication Module) フレームワークの Java テクノロジバージョンを実装しています。

JSR 196 により、別のレイヤーでプラグインを開発できます。`AuthConfigProvider` や `AuthConfigFactory` などの、新しい認証メカニズムを設定する方法を変更するプラグインを定義できます。また、`ServerAuthModule` や `ClientAuthModule` などの、新しい認証メカニズムを定義することもできます。

監査

「監査」は、セキュリティー対策の効果を評価するために、セキュリティーに関するイベントを取得するための手段です。Enterprise Server は、監査モジュールを使用して、すべての認証と承認の決定に関する監査証跡を取得します。Enterprise Server は、デフォルトの監査モジュールのほか、監査モジュールのカスタマイズ機能も提供します。

管理の手順については、208 ページの「監査モジュールの管理」を参照してください。

ファイアウォール

「ファイアウォール」は、2 つ以上のネットワーク間のデータフローを制御し、ネットワーク間のリンクを管理します。ファイアウォールは、ハードウェア要素およびソフトウェア要素で構成できます。Enterprise Server では、主に次のようなガイドラインが適しています。

- 一般的に、ファイアウォールは、クライアントが必要な TCP/IP ポートにアクセスできるように設定します。

  たとえば、HTTP リスナーがポート 8080 で動作している場合は、HTTP 要求をポート 8080 だけで受け付けるようにファイアウォールを設定します。同様に、HTTPS 要求がポート 8081 に設定されている場合は、HTTPS 要求をポート 8081 で受け付けるようにファイアウォールを設定する必要があります。
- インターネットから EJB モジュールへの直接の RMI-IIOP (Remote Method Invocations over Internet Inter-ORB Protocol) アクセスが必要な場合は、同様に RMI-IIOP リスナーのポートを開きます。

  注 – RMI-IIOP リスナーポートはセキュリティーリスクの原因となるため、このポートは開かないようにすることをお勧めします。

- 二重のファイアウォールのアーキテクチャーでは、HTTP および HTTPS トランザクションを受け付けるように外部ファイアウォールを設定する必要があります。また、ファイアウォールの背後の Enterprise Server と通信する HTTP サーバープラグインを受け付けるように内部ファイアウォールを設定する必要があります。

証明書と SSL

ここでは、次のテーマを取り上げます。

管理の手順については、210 ページの「JSSE 証明書の管理 」を参照してください。

証明書

「証明書」 (または、デジタル証明書) は、インターネット上の人物やリソースを一意に識別する電子ファイルです。さらに証明書は 2 つのエンティティー間の安全で機密保護された通信を可能にします。証明書にはさまざまな種類があります。

- 「個人証明書」は、個人によって使用されます。
- 「サーバー証明書」は、SSL (Secure Sockets Layer) テクノロジを通して、サーバーとクライアント間でセキュリティー保護されたセッションを確立するために使用されます。

証明書は「公開鍵暗号化」に基づき、意図した受信者だけが情報を解読できるように、デジタルの鍵 (非常に長い数値) のペアを使用して暗号化または符号化します。そして受信者は、情報を「復号化」して解読します。「鍵のペア」には公開鍵と非公開鍵が含まれます。所有者は公開鍵を配布して、だれでも利用できるようにします。しかし、所有者は非公開鍵を決して配布せず、常時秘密にしておきます。鍵は数学的に関連付けられているので、一方の鍵で暗号化されたデータは、そのペアのもう一方の鍵でしか復号化することができません。

証明書は、「証明書発行局」(CA) と呼ばれる、信頼できるサードパーティーが発行します。CA はパスポートセンターに似ています。CA は、証明書の所有者の身元を確認したあと、偽造や改ざんができないように証明書に署名します。CA が証明書に署名したあと、所有者は ID の証明としてこれを提出することで、暗号化され、機密保護された通信を確立できます。もっとも重要な点は、証明書によって所有者の公開鍵が所有者の ID と結び付けられることです。

公開鍵のほかに、通常、証明書には次のような情報が含まれています。

- 所有者の名前、および証明書を使用する Web サーバーの URL や個人のメールアドレスなどその他の識別情報
- 証明書を発行した CA の名前
- 有効期限の日付

証明書は、X.509 形式の技術仕様で管理されます。certificate レルムのユーザー ID を検証するために、 certificate 認証サービスは X.509 証明書の共通名フィールドを主体名として使用して、X.509 証明書を検証します。

証明書チェーン

「証明書チェーン」とは、最後がルート CA 証明書で終わる、継続的な CA によって発行される一連の証明書です。

Web ブラウザは、ブラウザが自動的に信頼する一連の「ルート」CA 証明書で事前に設定されます。別の場所で発行されたすべての証明書は、有効性を検証するために「証明書チェーン」を備えている必要があります。

証明書が最初に生成される場合、それは「自己署名付き」証明書です。自己署名付き証明書とは、発行者 (署名者) が被認証者 (公開鍵が証明書で認証されているエンティティー) と同じものです。所有者は、証明書の署名要求 (CSR) を CA に送信するとき、その応答をインポートし、自己署名付き証明書が証明書のチェーンによって置き換えられます。チェーンの元の部分には、被認証者の公開鍵を認証する CA によって発行された証明書 (応答) があります。このチェーンの次の証明書は、CA の公開鍵を認証するものです。通常、これは自己署名付き証明書 (つまり、自らの公開鍵を認証する CA からの証明書) およびチェーンの最後の証明書です。

CA が証明書のチェーンに戻ることができる場合もあります。この場合、チェーンの元の証明書は同じ (キーエントリの公開鍵を認証する、CA によって署名された証明書) ですが、チェーン 2 番目の証明書が、CSR の送信先の CA の公開鍵を認証する、異なる CA によって署名された証明書です。そして、チェーンのその次の証明書は 2 番目の鍵を認証する証明書というように、自己署名付き「ルート」証明書に到達するまで続きます。こうして、チェーンの最初以降の各証明書は、チェーンの前にある証明書の署名者の公開鍵を認証します。

証明書ファイル

Enterprise Server のインストール中に、証明書が内部テストに適した JSSE (Java Secure Socket Extension) 形式で生成されます。デフォルトでは、Enterprise Server は証明書情報を *domain-dir*/config ディレクトリの証明書データベースに格納します。

キーストアファイル

key3.db ファイルには、非公開鍵を含む Enterprise Server の証明書が格納されます。キーストアファイルはパスワードで保護されています。

各キーストアエントリには一意のエイリアスがあります。インストール後の Enterprise Server のキーストアには、s1as のエイリアスを持つ 1 つのエントリが含まれています。

トラストストアファイル　`cert8.db` ファイルには、ほかのエンティティーの公開鍵を含む、Enterprise Server の信頼できる証明書が格納されます。信頼できる証明書では、サーバーは証明書の公開鍵が証明書の所有者に属していることを確認しています。通常、信頼できる証明書には CA の証明書も含まれています。

デフォルトでは、Enterprise Server は、サンプルアプリケーションで開発目的のために動作するキーストアおよびトラストストアを使用して設定されています。

Secure Sockets Layer

「SSL」(Secure Sockets Layer) とは、インターネットの通信およびトランザクションのセキュリティー保護でもっとも普及している標準仕様です。セキュリティー保護された Web アプリケーションは HTTPS (HTTP over SSL) を使用します。HTTPS プロトコルは証明書を使用して、サーバーとクライアント間のセキュリティー保護された機密通信を保証します。SSL 接続では、クライアントとサーバーの両方が送信の前にデータを暗号化します。データは受信時に復号化されます。

クライアントの Web ブラウザがセキュリティー保護されたサイトに接続するときに、次のように「*SSL* ハンドシェーク」が行われます。

1. ブラウザはネットワークを介してセキュアなセッションを要求するメッセージを送信します。通常は、`http` ではなく `https` で始まる URL を要求します。
2. サーバーは、公開鍵を含む証明書を送信することで応答します。
3. ブラウザは、サーバーの証明書が有効であること、またサーバーの証明書が証明書をブラウザのデータベースに持つ信頼されている CA によって署名されていることを検証します。さらに、CA の証明書の有効期限が切れていないことも検証します。
4. 証明書が有効な場合、ブラウザは 1 回かぎりの一意の「セッション鍵」を生成し、サーバーの公開鍵でそれを暗号化します。そして、ブラウザは暗号化されたセッション鍵をサーバーに送信し、両方でコピーを持てるようにします。
5. サーバーは、非公開鍵を使用してメッセージを復号化し、セッション鍵を復元します。

ハンドシェークの後、クライアントは Web サイトの ID を検証し、クライアントと Web サーバーだけがセッション鍵のコピーを持ちます。これ以降、クライアントとサーバーはセッション鍵を使用して互いにすべての通信を暗号化します。こうすると、通信は確実にセキュアになります。

SSL 標準の最新バージョンは TLS (Transport Layer Security) と呼ばれています。Enterprise Server は、SSL 3.0 および TLS 1.0 の暗号化プロトコルをサポートしています。

SSL を使用するには、セキュリティー保護された接続を受け付ける外部インタフェースまたは IP アドレスごとに、Enterprise Server が証明書を所持しておく必要があります。ほとんどの Web サーバーの HTTPS サービスは、証明書がインストールされるまで実行されません。HTTP リスナーに SSL を適用する手順については、300 ページの「SSL の HTTP リスナーを構成する」を参照してください。

暗号化方式

「暗号化方式」とは、暗号化と復号化に使用される暗号化アルゴリズムです。SSL および TLS プロトコルは、サーバーとクライアントでお互いを認証するために使用される多くの暗号化方式のサポート、証明書の送信、およびセッション鍵の確立を行います。

安全度は、暗号化方式によって異なります。クライアントとサーバーは異なる暗号化方式群をサポートできます。クライアントとサーバーは安全な接続のために、双方で通信に使用可能なもっとも強力な暗号化方式を使用します。したがって、通常はすべての暗号化方式を有効にすれば十分です。

名前ベースの仮想ホスト

セキュアなアプリケーションに名前ベースの仮想ホストを使用すると、問題が発生する場合があります。これは、SSL プロトコル自体の設計上の制約です。クライアントブラウザがサーバーの証明書を受け付ける SSL ハンドシェークは、HTTP 要求がアクセスされる前に行われる必要があります。その結果、認証より前に仮想ホスト名を含む要求情報を特定できないので、複数の証明書を単一の IP アドレスに割り当てできません。

単一の IP すべての仮想ホストが同じ証明書に対して認証を必要とする場合、複数の仮想ホストを追加しても、サーバーの通常の SSL 動作を妨害する可能性はありません。ただし、証明書 (主に正式な CA の署名済みの証明書が該当) に表示されているドメイン名がある場合、ほとんどのブラウザがサーバーのドメイン名をこのドメイン名と比較することに注意してください。ドメイン名が一致しない場合、これらのブラウザは警告を表示します。一般的には、アドレスベースの仮想ホストだけが本稼働環境の SSL で広く使用されています。

システムセキュリティー管理用ツール

Enterprise Server には、次のシステムセキュリティー管理用ツールがあります。

管理コンソール	管理コンソールは、サーバー全体のセキュリティーを設定するための、ブラウザベースのユーティリティーです。証明書、ユーザー、グループ、およびレルムの管理や、システム全体に関

	するその他のセキュリティータスクの実行に使用します。管理コンソール の概要については、40 ページの「管理コンソール」を参照してください。
asadmin ユーティリティー	asadmin コマンド行ユーティリティーでは、管理コンソール と同じ多数のタスク 実行できます。管理コンソール では実行できない操作を、asadmin ユーティリティーで実行できる場合もあります。asadmin の概要については、41 ページの「asadmin ユーティリティー」を参照してください。
keytool ユーティリティー	keytool Java 2 Platform, Standard Edition (J2SE) コマンド行ユーティリティーは、デジタル証明書および鍵のペアを管理するために使用します。詳細は、210 ページの「JSSE 証明書の管理 」を参照してください。
policytool ユーティリティー	policytool J2SE グラフィカルユーティリティーは、システム全体の Java セキュリティーポリシーを管理するために使用します。管理者が policytool を使用することはほとんどありません。

keytool、policytool、およびその他の Java セキュリティーツールの使用方法については、http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security の『Java 2 SDK Tools and Utilities』を参照してください。

パスワードの管理

パスワードの管理には複数の方法があります。各管理者には、パスワードを秘密にして定期的に変更するように指示します。ユーザーにコマンドを入力させず、asadmin のサブコマンドがパスワードを格納したファイルにアクセスできるように、これらのファイルを設定することができます。domain.xml ファイルで重要なパスワードが表示されないように、エイリアスを設定してパスワードを暗号化することができます。

ここでは、次のテーマを取り上げます。

- 202 ページの「マスターパスワードを変更する」
- 203 ページの「管理パスワードを変更する」
- 204 ページの「パスワードをファイルから設定する」
- 205 ページの「パスワードエイリアスの管理」

▼ マスターパスワードを変更する

マスターパスワードは、ドメインで使用される暗号化ストア (NSS `cert8.db` トラストストアまたは Java JKS キーストア) へのアクセスを許可します。このパスワードは、UNIX ユーザーに結び付けられません。この全体で共有されるパスワードは、システムでもっとも重要なデータです。マスターパスワードを認証に使用したり、ネットワークを介して送信したりすることは決してありません。

パスワードを要求されたときに手動で入力するか、パスワードファイルに隠蔽化することができます。パスワードファイルがない場合は、マスターパスワードの入力を要求されます。パスワードファイルがある場合に、入力を必要とするようにアクセスを変更する場合は、ファイルを削除します。デフォルトのマスターパスワードは `changeit` です。

`change-master-password` サブコマンドをローカルモードで使用して、マスターパスワードを変更します。

マスターパスワードを変更すると、マスターパスワードキーストアにパスワードが再保存されます。これは、Java JCEKS タイプのキーストアです。

始める前に　ドメインが停止していない場合、このサブコマンドは機能しません。

1. パスワードを変更するドメインを停止します。
   94 ページの「ドメインの停止」を参照してください。
2. `change-master-password(1)` サブコマンドを使用して、ドメインのマスターパスワードを変更します。
   旧パスワードと新しいパスワードの入力を要求されます。依存しているすべての項目が再暗号化されます。
3. ドメインを起動します。
   93 ページの「ドメインの起動」を参照してください。

例 11–1 マスターパスワードの変更

`change-master-password` サブコマンドは対話形式で動作し、旧マスターパスワードと新しいマスターパスワードの入力を要求します。この例では、`domain44ps` のマスターパスワードを変更します。

```
asadmin> change-master-password domain44ps
```

`login(1)` サブコマンドを使用してすでにドメインにログインしている場合は、新しいマスターパスワードの入力を要求されます。

```
Please enter the new master password>
Please enter the new master password again>
```

ドメインにログインしていない場合は、旧マスターパスワードと新しいマスターパスワードの両方の入力を要求されます。

```
Please enter the master password again>
Please enter the new master password>
Please enter the new master password again>
```

次のような情報が表示されます。

```
Master password changed for domain44ps
```

参照　コマンド行に asadmin help change-master-password と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 管理パスワードを変更する

管理パスワードを変更するには、リモートモードで change-admin-password サブコマンドを使用します。デフォルトの管理パスワードは admin です。確認のために、旧パスワードと新しいパスワードの入力を要求されます。

注 – ZIP インストール中に、パスワードが設定されていないデフォルトの admin ユーザーを受け入れている場合は、このユーザーにパスワードを追加できます。パスワードが設定されていない admin というユーザーしか存在しない場合は、ログイン情報を要求されません。その他の状況ではログインが必要です。

管理パスワードの暗号化は強く推奨されています。

始める前に　パスワードのエイリアスを作成する (暗号化する) 前に管理パスワードを変更する場合は、set サブコマンドを使用できます。次に例を示します。

```
asadmin set --user admin server.jms-service.jms-host.default_JMS_host.admin-password=
new_pwd
```

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. change-admin-password(1) サブコマンドを使用して、管理パスワードを変更します。

3 要求に従って、旧管理パスワードと新しい管理パスワードを入力します。

例 11–2 管理パスワードの変更

この例では、ユーザー anonymous の管理パスワードを、adminadmin から newadmin に変更します。

```
asadmin> change-admin-password --user anonymous
```

旧管理パスワードと新しい管理パスワードの入力を要求されます。

```
Enter admin password>adminadmin
Enter new admin password>newadmin
Enter new admin password again>newadmin
```

次のような情報が表示されます。

```
Command change-admin-password executed successfully.
```

参照 コマンド行に asadmin help change-admin-password と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ パスワードをファイルから設定する

コマンド行にパスワードを入力する代わりに、passwords.txt などのファイルからコマンドのパスワードにアクセスできます。asadmin ユーティリティーの --passwordfile オプションは、パスワードを含むファイルの名前を受け取ります。ファイル内のパスワードのエントリには、パスワード名の前に AS_ADMIN_ というプレフィックス (大文字) を付ける必要があります。

次のタイプのパスワードも指定できます。

```
AS_ADMIN_MASTERPASSWORD
AS_ADMIN_USERPASSWORD
AS_ADMIN_ALIASPASSWORD
```

1 パスワードファイルを編集します。

たとえば、ドメイン管理サーバー (DAS) のパスワードを指定するには、パスワードファイルに次のようなエントリを追加します。adminadmin は管理者パスワードです。

```
AS_ADMIN_PASSWORD=adminadmin
```

2 パスワードファイルを保存します。

これで、asadmin サブコマンドにパスワードファイルを指定できるようになりました。この例では、passwords.txt がパスワードを含むファイルです。

```
asadmin>delete-jdbc-resource --user admin --password passwords.txt jdbc/DerbyPool
```

注意事項 AS_ADMIN_PASSWORD がグローバル環境にエクスポートされている場合、--passwordfile オプションを指定すると、--passwordfile オプションの使用に関する警告が表示されます。この警告が生成されないようにするには、AS_ADMIN_PASSWORD を設定解除します。

パスワードエイリアスの管理

パスワードエイリアスは、パスワード自体が構成ファイルに現れないように、パスワードに間接的にアクセスするために使用します。

ここでは、次のテーマを取り上げます。

▼ パスワードエイリアスを作成する

パスワードのエイリアスをドメインのキーストアに作成するには、リモートモードで create-password-alias サブコマンドを使用します。エイリアス名に対応するパスワードは、ドメイン構成ファイルに暗号化された形式で保存されます。create-password-alias サブコマンドの形式には、ユーザーにすべての情報を要求するセキュリティー保護された対話形式と、パスワードがコマンド行で伝達されるスクリプトに適した形式があります。

set(1) サブコマンドを使用して、構成ファイル中のパスワードを削除および置換することもできます。次に例を示します。

```
asadmin set --user admin server.jms-service.jms-host.default_JMS_host.
admin-password='${ALIAS=jms-password}'
```

1 サーバーが実行されていることを確認します。

リモートサブコマンドには、実行中のサーバーが必要です。

2 構成ファイルが保存されているディレクトリに移動します。

デフォルトでは、構成ファイルは *domain-dir* /config に保存されています。

3. create-password-alias(1) サブコマンドを使用して、パスワードエイリアスを作成します。
4. 要求に応じて、エイリアスのパスワードを入力します。
5. エイリアスをパスワードファイルに追加します。
   パスワードファイル (たとえば、passwords.txt) に、AS_ADMIN_PASSWORD=${ALIAS=*admin-password-alias*} という行を追加します。*admin-password-alias* は新しいパスワードエイリアスです。
6. **Enterprise Server** ドメインを停止します。
   94 ページの「ドメインの停止」を参照してください。
7. エイリアスを含むファイルを指定して、ドメインを起動します。
   次に例を示します。

```
asadmin start-domain --user admin --passwordfile /path-to/passwords.txt domain1
```

例 **11–3** パスワードエイリアスの作成

この例では、admin ユーザーの新しい jms-password エイリアスを作成します。

```
asadmin> create-password-alias --user admin jms-password
```

エイリアスのパスワードを入力するように要求されます。

```
Please enter the alias password>secret-password
Please enter the alias password again>secret-password
Command create-password-alias executed successfully.
```

参照 コマンド行に asadmin help create-password-alias と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ パスワードエイリアスを一覧表示する

既存のパスワードエイリアスを一覧表示するには、リモートモードで list-password-aliases サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-password-aliases(1) サブコマンドを使用して、パスワードエイリアスを一覧表示します。

例 11–4 パスワードエイリアスの一覧表示

この例では、既存のパスワードエイリアスを一覧表示します。

```
asadmin> list-password aliases
jmspassword-alias
Command list-password-aliases executed successfully
```

参照 コマンド行に asadmin help list-password-aliases と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ パスワードエイリアスを削除する

既存のパスワードエイリアスを削除するには、リモートモードで delete-password-alias サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-password-aliases(1) サブコマンドを使用して、エイリアスをすべて表示します。
3. list-password-aliases(1) サブコマンドを使用して、パスワードエイリアスを削除します。

例 11–5 パスワードエイリアスの削除

この例では、パスワードエイリアス jmspassword-alias を削除します。

```
asadmin> delete-password-alias jmspassword-alias
Command list-password-aliases executed successfully
```

参照 コマンド行に asadmin help delete-password-alias と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ パスワードエイリアスを更新する

既存のパスワードエイリアスのパスワードを変更するには、リモートモードで update-password-alias サブコマンドを使用します。update-password-alias サブコマンドでは、セキュリティー保護された対話形式 (ユーザーがすべての情報の入力を求められる) と、スクリプトの処理しやすい形式 (パスワードがコマンド行で伝達される) の両方を使用できます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. update-password-alias(1) サブコマンドを使用して、エイリアスを更新します。
3. 要求に応じて、パスワードを入力します。

例 11–6 パスワードエイリアスの更新

この例では、jmspassword-alias エイリアスのパスワードを更新します。

```
asadmin> update-password-allias /home/password.txt jsmpassword-alias
```

エイリアスの新しいパスワードを入力するように要求されます。

```
Please enter the alias password>new-secret-password
Please enter the alias password again>new-secret-password
Command update-password-alias executed successfully
```

参照 コマンド行に asadmin help update-password-alias と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

監査モジュールの管理

ここでは、次のテーマを取り上げます。

- 208 ページの「監査モジュールを作成する」
- 209 ページの「監査モジュールを一覧表示する」
- 210 ページの「監査モジュールを削除する」

▼ 監査モジュールを作成する

監査機能を実装するアドオンコンポーネントの監査モジュールを作成するには、リモートモードで create-audit-module サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. create-audit-module(1) サブコマンドを使用して、監査モジュールを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

例 11–7 監査モジュールの作成

この例では、sampleAuditModule という名前の監査モジュールを作成します。

```
asadmin> create-audit-module
--classname com.sun.appserv.auditmodule --property defaultuser=
admin:Password=admin sampleAuditModule
Command create-audit-module executed successfully.
```

参照 コマンド行に asadmin help create-audit-module と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 監査モジュールを一覧表示する

次のいずれか対象の監査モジュールを一覧表示するには、リモートモードで list-audit-modules サブコマンドを使用します。

- サーバーインスタンス、server (デフォルト)
- 指定したサーバーインスタンス
- 指定した構成

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-audit-modules(1) サブコマンドを使用して、監査モジュールを一覧表示します。

例 11–8 監査モジュールの一覧表示

この例では、localhost 上の監査モジュールを一覧表示します。

```
asadmin> list-audit-modules
audit-module : default
audit-module : sampleAuditModule
Command list-audit-modules executed successfully.
```

参照 コマンド行に asadmin help list-audit-modules と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 監査モジュールを削除する

既存の監査モジュールを削除するには、リモートモードで `delete-audit-module` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-audit-modules(1)` サブコマンドを使用して、監査モジュールを一覧表示します。
3. `delete-audit-module(1)` サブコマンドを使用して、監査モジュールを削除します。

例 **11–9** 監査モジュールの削除

この例では、`sampleAuditModule` を削除します。

```
asadmin> delete-audit-module sampleAuditModule
Command delete-audit-module executed successfully.
```

JSSE 証明書の管理

開発者プロファイルでは、Enterprise Server v3 はサーバー側で JSSE 形式を使用して証明書とキーストアを管理します。すべてのプロファイルで、クライアント側 (アプリケーションクライアントまたはスタンドアロン) は JSSE 形式を使用します。

J2SE SDK には JSSE (Java Secure Socket Extension) デジタル証明書を設定および操作することができる `keytool` ユーティリティーが付属しています。公開鍵と非公開鍵のペアおよび関連する証明書を管理し、通信しているピアの公開鍵を (証明書の形式で) キャッシュすることができます。

ここでは、次のテーマを取り上げます。

- 210 ページの「`keytool` による証明書の生成」
- 212 ページの「`keytool` を使用して証明書に署名する」
- 214 ページの「`keytool` を使用して証明書を削除する」

▼ keytool による証明書の生成

デフォルトでは、`keytool` ユーティリティーは実行元のディレクトリにキーストアファイルを作成します。

始める前に　keytool ユーティリティーを実行するには、シェル環境を設定して、J2SE の /bin ディレクトリがパスに含まれるようにする必要があります。そうでない場合は、ユーティリティーのフルパスをコマンド行に指定する必要があります。

1. キーストアファイルおよびトラストストアファイルが格納されているディレクトリに移動します。

   証明書の生成は常に、キーストアファイルとトラストストアファイルが格納されているディレクトリ内で行います。デフォルトは、*domain-dir*/config です。

2. 次のコマンド形式を使用して、キーストアファイル keystore.jks に証明書を生成します。

   ```
   keytool -genkey -alias keyAlias-keyalg RSA
    -keypass changeit
    -storepass changeit
   keystore keystore.jks
   ```

   keyAlias には任意の一意名を指定します。キーストアまたは非公開鍵のパスワードをデフォルト (changeit) から変更している場合は、changeit を新しいパスワードで置き換えてください。デフォルトのキーパスワードエイリアスは s1as です。

   名前、組織、およびその他の情報を尋ねるプロンプトが表示されます。

3. 次のコマンド形式を使用して、生成された証明書を server.cer ファイル (または、必要に応じて client.cer) にエクスポートします。

   ```
   keytool -export -alias keyAlias-storepass changeit
    -file server.cer
    -keystore keystore.jks
   ```

4. 認証局によって署名された証明書が必要な場合は、212 ページの「keytool を使用して証明書に署名する」を参照してください。

5. 次のコマンド形式を使用して、cacerts.jks トラストストアファイルを作成し、証明書をトラストストアに追加します。

   ```
   keytool -import -v -trustcacerts
   -alias keyAlias
    -file server.cer
   -keystore cacerts.jks
    -keypass changeit
   ```

   キーストアまたは非公開鍵のパスワードをデフォルト (changeit) から変更している場合は、新しいパスワードで置き換えてください。

   証明書に関する情報が表示され、証明書を信頼するかどうかを確認するプロンプトが表示されます。

6 yes と入力し、続いて **Enter** キーを押します。
次のような情報が表示されます。

```
Certificate was added to keystore
[Saving cacerts.jks]
```

7 変更内容を適用するために、**Enterprise Server** を再起動します。**94** ページの「ドメインの再起動」を参照してください。

例 11–10 RSA 鍵アルゴリズムを使用した JKS キーストアへの自己署名付き証明書の作成

RSA は RSA Data Security, Inc. によって開発された、公開鍵暗号化テクノロジです。

```
keytool  -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}
-dname  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

例 11–11 デフォルトの鍵アルゴリズムを使用した JKS キーストアへの自己署名付き証明書の作成

```
keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
${keystore.pass}
```

例 11–12 JKS キーストア内の使用可能な証明書の表示

```
keytool -list -v  -keystore ${keystore.file} -storepass ${keystore.pass}
```

例 11–13 JKS キーストア内の証明書情報の表示

```
keytool -list -v  -alias ${cert.alias} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

参照 `keytool` の詳細は、`keytool` のドキュメント (`http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html`) を参照してください。

▼ keytool を使用して証明書に署名する

証明書の作成後、所有者は証明書に署名して偽造を防止する必要があります。E コマースのサイト、または ID の認証が重要であるサイトは、既知の証明書発行局 (CA) から証明書を購入できます。

注 – 認証に心配がない場合 (たとえば、非公開の安全な通信だけが必要な場合) は、自己署名付き証明書を使用して、CA 証明書の取得に必要な時間と費用を節約することができます。

1. **CA の Web サイトの指示に従って、証明書の鍵のペアを生成します。**
2. 生成された証明書の鍵のペアをダウンロードします。
   キーストアファイルとトラストストアファイルが格納されているディレクトリに、証明書を保存します。デフォルトは、*domain-dir*/config です。
3. 使用しているシェルで、証明書を含むディレクトリに変更します。
4. 次のコマンド形式を使用して、証明書をローカルキーストアと、必要な場合はローカルトラストストアにインポートします。

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
-storepass changeit
```

   キーストアまたは非公開鍵のパスワードがデフォルト以外の値である場合は、デフォルトのパスワード (changeit) を新しいパスワードで置き換えてください。
5. 変更内容を適用するために、**Enterprise Server** を再起動します。94 ページの「ドメインの再起動」を参照してください。

例 11–14 RFC/テキスト形式の証明書の JKS キーストアへのインポート

証明書は、バイナリエンコーディングではなく、RFC (Internet Request for Comments) 1421 標準によって定義された印刷可能なエンコーディング形式を使って格納されることがしばしばあります。Base 64 エンコーディングとしても知られるこの証明書形式を使用すれば、電子メールなどの機構を使って証明書をほかのアプリケーションにエクスポートしやすくなります。

```
keytool -import -noprompt -trustcacerts -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

例 11–15 JKS キーストアからの PKCS7 形式による証明書のエクスポート

「Public Key Cryptography Standards #7, Cryptographic Message Syntax Standard」によって定義された応答形式には、発行される証明書に加え、それをサポートする証明書チェーンも含まれます。

```
keytool -export -noprompt -alias ${cert.alias} -file ${cert.file}
-keystore ${keystore.file} -storepass ${keystore.pass}
```

例 11–16 JKS キーストアからの RFC/テキスト形式による証明書のエクスポート

```
keytool -export -noprompt -rfc -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

参照 keytool の詳細は、keytool のドキュメント (http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html) を参照してください。

▼ keytool を使用して証明書を削除する

keytool -delete コマンドを使用して、既存の証明書を削除します。

- 次のコマンド形式を使用して、証明書を削除します。

```
keytool -delete
 -alias keyAlias
 -keystore keystore-name
 -storepass password
```

例 11–17 JKS キーストアからの証明書の削除

```
keytool -delete -noprompt -alias ${cert.alias}  -keystore ${keystore.file}
-storepass ${keystore.pass}
```

参照 keytool の詳細は、keytool のドキュメント (http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html) を参照してください。

◆ ◆ ◆ 第 12 章

12 ユーザーセキュリティーの管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server 環境でユーザーセキュリティーを管理する手順について説明します。Enterprise Server は、レルム、ユーザー、およびグループに対して、認証と承認のポリシーを適用します。この章の説明では、認証、承認、証明書などのセキュリティー機能について理解していることを前提とします。これらのセキュリティー機能については、第 11 章「システムセキュリティーの管理」を参照してください。

ここでは、次のテーマを取り上げます。

- 215 ページの「認証レルムの管理」
- 220 ページの「ファイルユーザーの管理」

これらのタスクを 管理コンソール を使用して実行する場合の手順は、管理コンソール のオンラインヘルプで説明します。

認証レルムの管理

「認証レルム」は、セキュリティーポリシードメインまたはセキュリティードメインとも呼ばれ、Enterprise Server によって共通のセキュリティーポリシーが定義および適用される範囲を表します。Enterprise Server は、ファイル、証明書、および管理のレルムで事前設定されます。これらに加え、LDAP、JDBC、ダイジェスト、Solaris、またはカスタムのレルムを設定できます。アプリケーションは使用するレルムを配備記述子で指定できます。アプリケーションでレルムが指定されていない場合、Enterprise Server はデフォルトのレルム (file) を使用します。

ファイルレルム	Enterprise Server は、ユーザー資格を keyfile という名前のファイルにローカルに格納します。ファイルレルムは、初期状態のデフォルトレルムです。
管理レルム	管理レルムはファイルレルムでもあり、管理者のユーザー資格を admin-keyfile という名前のファイルにローカルに格納します。

証明書レルム	Enterprise Server はユーザー資格を証明書データベースに格納します。証明書レルムを使用する場合、サーバーは HTTPS プロトコルで証明書を使用して Web クライアントを認証します。
LDAP レルム	Enterprise Server は、ユーザー資格を DirectoryServer などの LDAP (Lightweight Directory Access Protocol) サーバーから取得します。LDAP とは、一般のインターネットまたは会社のイントラネットのどちらであっても、ネットワークでの組織、個人、およびファイルやデバイスなどその他のリソースの検出をだれにでもできるようにするプロトコルです。LDAP レルムのユーザーとグループの管理については、LDAP サーバーのドキュメントを参照してください。
JDBC レルム	Enterprise Server はユーザー資格をデータベースから取得します。サーバーは、データベース情報と構成ファイル内の有効な JDBC レルムオプションを使用します。
ダイジェストレルム	ダイジェスト認証は、ユーザー名とパスワードに基づいてユーザーを認証します。ただし、認証はパスワードを暗号化された形式で送信することで実行されます。
Solaris レルム	Enterprise Server はユーザー資格を Solaris オペレーティングシステムから取得します。このレルムは、Solaris 9 および Solaris 10 オペレーティングシステムでサポートされます。Solaris レルムのユーザーおよびグループの管理については、Solaris のドキュメントを参照してください。
カスタムレルム	リレーショナルデータベースや他社のコンポーネントなど、その他のリポジトリを作成してユーザー資格に使用できます。カスタムレルムの詳細は、管理コンソール のオンラインヘルプを参照してください。カスタムレルムを作成する手順については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の「Creating a Custom Realm」を参照してください。

Enterprise Server の認証サービスは、複数のレルムでユーザーを管理できます。

次のタスクと情報を使用して、認証レルムを管理します。

- 217 ページの「認証レルムを作成する」
- 217 ページの「認証レルムを一覧表示する」
- 218 ページの「認証レルムを更新する」
- 218 ページの「認証レルムを削除する」
- 219 ページの「JDBC レルムまたはダイジェスト認証レルムを設定する」

▼ 認証レルムを作成する

認証レルムを作成するには、リモートモードで `create-auth-realm` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `create-auth-realm(1)` サブコマンドを使用して、レルムを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

例 12–1 レルムの作成

この例では、`db` という名前のレルムを作成します。

```
asadmin> create-auth-realm --classname com.iplanet.ias.security.
auth.realm.DB.Database --property defaultuser=admin:Password=admin db
Command create-auth-realm executed successfully.
```

参照 コマンド行に `asadmin help create-auth-realm` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

カスタムレルムの作成については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の「Creating a Custom Realm」を参照してください。

▼ 認証レルムを一覧表示する

既存の認証レルムを一覧表示するには、リモートモードで `list-auth-realms` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-auth-realms(1)` サブコマンドを使用して、レルムを一覧表示します。

例 12–2 レルムの一覧表示

この例では、`localhost` 上の認証レルムを一覧表示します。

```
asadmin> list-auth-realms
db
certificate
file
admin-realm
Command list-auth-realms executed successfully.
```

参照　コマンド行に asadmin help list-auth-realms と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 認証レルムを更新する

set サブコマンドを使用して、既存の認証レルムを変更します。

注– カスタムレルムでは、サーバーの再起動は必要ありません。

---

1. list-auth-realms(1) サブコマンドを使用して、レルムを一覧表示します。
2. set(1) サブコマンドを使用して、指定したスレッドプールの値を変更します。
   スレッドプールはドット表記名で識別されます。
3. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

▼ 認証レルムを削除する

既存の認証レルムを削除するには、リモートモードで delete-auth-realm サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-auth-realms(1) サブコマンドを使用して、レルムを一覧表示します。
3. 必要に応じて、レルムを削除することをユーザーに通知します。
4. delete-auth-realm(1) サブコマンドを使用して、レルムを削除します。
5. 変更内容を適用するために、**Enterprise Server** を再起動します。**94** ページの「ドメインの再起動」を参照してください。

例 12–3 レルムの削除

この例では、db という名前の認証レルムを削除します。

```
asadmin> delete-auth-realm db
Command delete-auth-realm executed successfully.
```

参照 コマンド行に asadmin help delete-auth-realm と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JDBC レルムまたはダイジェスト認証レルムを設定する

Enterprise Server では、接続プールの代わりに JDBC レルムにユーザー資格 (ユーザー名とパスワード) を指定できます。接続プールの代わりに jdbc タイプのレルムを使用すると、ほかのアプリケーションがユーザー資格のデータベース表を参照するのを防止できます。

注 – JDBC レルムでは、デフォルトでは平文によるパスワードの保存はサポートされません。通常の状況では、パスワードを平文で保存しないでください。

---

1. レルムのユーザー資格を格納するデータベース表を作成します。
   データベース表の作成方法は、使用しているデータベースによって異なります。
2. 作成したデータベース表にユーザー資格を追加します。
   データベース表にユーザー資格を追加する方法は、使用しているデータベースによって異なります。
3. データベースの JDBC 接続プールを作成します。
   248 ページの「JDBC 接続プールを作成する」を参照してください。
4. データベースの JDBC リソースを作成します。
   253 ページの「JDBC リソースを作成する」
5. レルムを作成します。
   手順については、217 ページの「認証レルムを作成する」を参照してください。

注 – JAAS コンテキストは、ダイジェスト認証では jdbcDigestRealm に、その他の認証タイプでは jdbcRealm になります。

---

6 配備記述子を変更して、jdbc レルムを指定します。
アプリケーションに関連付けられている配備記述子を変更します。

- **Enterprise Archive (EAR)** ファイルのエンタープライズアプリケーションの場合は、sun-application.xml ファイルを変更します。
- **Web Application Archive (WAR)** ファイルの **Web** アプリケーションの場合は、web.xml ファイルを変更します。
- **EJB JAR** ファイルのエンタープライズ **Bean** の場合は、sun-ejb-jar.xml ファイルを変更します。

レルムの指定方法については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の「How to Configure a Realm」を参照してください。

7 レルム内のユーザーにセキュリティーロールを割り当てます。
ユーザーにセキュリティーロールを割り当てるには、変更した配備記述子に security-role-mapping 要素を追加します。

8 データベースが動作していることを確認します。
必要に応じて、245 ページの「データベースを起動する」を参照してください。

9 認証を適用するには、サーバーを再起動します。
94 ページの「ドメインの再起動」を参照してください。

例 12–4 セキュリティーロールの割り当て

この例では、セキュリティーロール Employee をユーザー Calvin に割り当てる security-role-mapping 要素を示します。

```
<security-role-mapping>
    <role-name>Employee</role-name>
    <principal-name>Calvin</principal-name>
  </security-role-mapping>
```

ファイルユーザーの管理

「ユーザー」は、Enterprise Server で定義される個人またはアプリケーションプログラムの ID です。認証されたユーザーを「主体」と呼ぶ場合もあります。

管理者には、ユーザーを Enterprise Server 環境に統合し、これらのユーザーの資格を安全に確立して、ユーザーが使用権限を持つアプリケーションおよびサービスにアクセスできるようにする責任があります。

次のタスクを使用してユーザーを管理します。

▼ ファイルユーザーを作成する

keyfile に新しいエントリを追加し、新規ユーザーを作成するには、リモートモードで create-file-user サブコマンドを使用します。エントリにはユーザー名、パスワード、およびユーザーのグループが含まれます。グループ名をコロン (:) で区切ることで、複数のグループを指定できます。

新しい file レルムユーザーの作成は動的なイベントであり、サーバーの再起動は必要ありません。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 ユーザーを特定のグループに所属させる場合は、list-file-groups(1) サブコマンドを使用して現在のグループを表示します。

3 create-file-user(1) サブコマンドを使用して、ファイルユーザーを作成します。

例 12–5 ユーザーの作成

この例では、デフォルトレルムの file に Jennifer というユーザーを作成します。グループは指定しません。

```
asadmin> create-file-user --user admin
--passwordfile=c:\tmp\asadminpassword.txt Jennifer
Command create-file-user executed successfully.
```

参照 コマンド行に asadmin help create-file-user と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ ファイルユーザーを一覧表示する

keyfile 内のユーザーを一覧表示するには、リモートモードで list-file-users サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-file-users(1) サブコマンドを使用して、ユーザーを一覧表示します。

例 12–6 ファイルユーザーの一覧表示

この例では、デフォルトの file レルムファイルのファイルユーザーを一覧表示します。

```
asadmin> list-file-users
Jennifer
Command list-file-users executed successfully.
```

参照 コマンド行に asadmin help list-file-users と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ ファイルグループを一覧表示する

「グループ」は、肩書きや顧客のプロファイルなどの共通の特性で分類されたユーザーのカテゴリです。たとえば、E コマースアプリケーションのユーザーは customer グループに属し、お得意様は preferred グループにも属します。ユーザーをグループに分類すると、大量のユーザーによるアクセスを容易に制御できます。グループはサーバーおよびレルム全体で定義されます。ユーザーは複数のユーザーグループに関連付けることができます。

ロールはアプリケーション内での役割を定義するのに対し、グループはある方法で関連付けられているユーザーの集まりに過ぎません。この点で、グループとロールは異なります。たとえば、人事アプリケーションで、full-time、part-time、および on-leave といったグループがあるとします。これらのグループのユーザーは、すべて社員 (employee ロール) です。これに加え、各ユーザーには追加の雇用レベルを定義する各自の役職が指定されます。

ファイルユーザーのグループを一覧表示するには、リモートモードで list-file-groups サブコマンドを使用します。--name オプションを指定しない場合は、ファイルグループがすべて表示されます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-file-groups(1) サブコマンドを使用して、ファイルグループを一覧表示します。

例 12–7 ユーザーのグループの一覧表示

この例では、ユーザー joesmith のグループを一覧表示します。

```
asadmin> list-file-groups --name joesmith
staff
manager
Command list-file-groups executed successfully
```

▼ ファイルユーザーを更新する

指定したユーザーの keyfile の情報を変更するには、リモートモードで update-file-user サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. update-file-user(1) サブコマンドを使用して、ユーザー情報を更新します。
3. 変更内容を適用するために、Enterprise Server を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 12–8 ユーザーの更新

次のサブコマンドは、ユーザー Jennifer のグループを更新します。

```
asadmin> update-file-user --passwordfile c:\tmp\asadminpassword.txt --groups
staff:manager:engineer Jennifer
Command update-file-user executed successfully.
```

参照 コマンド行に asadmin help update-file-user と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ ファイルユーザーを削除する

指定したユーザーのエントリを keyfile から削除するには、リモートモードで delete-file-user サブコマンドを使用します。自分自身を削除することはできません。つまり、ログインしているユーザーを、そのセッション中に削除することはできません。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-file-users(1) サブコマンドを使用して、ユーザーを一覧表示します。
3. delete-file-user(1) サブコマンドを使用して、ユーザーを削除します。

例 12–9 ユーザーの削除

この例では、ユーザー Jennifer をデフォルトの file レルムから削除します。

```
asadmin> delete-file-user Jennifer
Command delete-file-user executed successfully.
```

参照 コマンド行に asadmin help delete-file-user と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 13 章

メッセージセキュリティーの管理

この章では、Sun GlassFish™ Enterprise Server 環境で、Web サービスのメッセージレイヤーセキュリティーを設定する際の情報と手順について説明します。

注 – メッセージセキュリティー (JSR 196) は、Enterprise Server の Full Platform Profile のみでサポートされ、Web Profile ではサポートされません。

ここでは、次のテーマを取り上げます。

この章の一部の内容は、セキュリティーと Web サービスに関する基本概念の理解を前提としてます。セキュリティーの詳細は、191 ページの「Enterprise Server のシステムセキュリティーについて」を参照してください。

本章で説明するタスクを 管理コンソール から実行する手順については、管理コンソール オンラインヘルプを参照してください。

Enterprise Server のメッセージセキュリティーについて

「メッセージセキュリティー」により、サーバーはメッセージレイヤーで Web サービスの呼び出しと応答をエンドツーエンドで認証できます。セキュリティー情報がメッセージ内に挿入され、その情報が完全なメッセージとともにネットワークレイヤー経由でメッセージの送信先に届けられます。メッセージセキュリティーはトランスポートレイヤーセキュリティーと異なり、メッセージの伝送と保護を分離できるため、メッセージは伝送後も保護されたままとなります。

Enterprise Server 上に配備された Web サービスをセキュリティー保護するには、アプリケーションの配備先コンテナ、またはそのアプリケーションがサービスを提供する Web サービスエンドポイントのいずれかに対し、SOAP レイヤーメッセージセキュリティープロバイダとメッセージ保護ポリシーをバインドします。Enterprise Server のクライアント側コンテナで SOAP レイヤーメッセージセキュリティー機能を設定するには、クライアントコンテナ、またはクライアントアプリケーションによって宣言されたポータブルサービス参照のいずれかに対し、SOAP レイヤーメッセージセキュリティープロバイダとメッセージ保護ポリシーをバインドします。

メッセージレベルのセキュリティーは、Enterprise Server 全体に対して、または特定のアプリケーションやメソッドに対して設定できます。アプリケーションレベルでのメッセージセキュリティーの設定については、『Sun GlassFish Enterprise Server v3 Application Development Guide』を参照してください。

ここでは、次のテーマを取り上げます。

セキュリティートークンとセキュリティーメカニズム

WS-Security は、Web サービスにセキュリティーを適用するための通信プロトコルを提供する仕様です。セキュリティーメカニズムはこの仕様を実装します。WSIT (Web Services Interoperability Technologies) は、WS-Security を実装して、メッセージが送信先のエンドポイントに達するまでに中間ノードを通過する場合でも、相互運用可能なメッセージコンテンツの完全性と機密性を提供します。WSIT によって提供される WS-Security は、既存のトランスポートレベルのセキュリティーに付加され、トランスポートレベルのセキュリティーもそのまま使用できます。Enterprise Server とともにインストールされる SOAP (Simple Object Access Protocol) レイヤーメッセージセ

キュリティープロバイダを使用すると、ユーザー名とパスワードのセキュリティートークンおよびX.509証明書セキュリティートークンを利用して、SOAP Webサービスメッセージを認証および暗号化することができます。

- 「ユーザー名トークン」。Enterprise Serverは、SOAPメッセージでユーザー名トークンを使用して、メッセージの送信者を認証します。パスワードが埋め込まれたユーザー名トークンを含むメッセージの受信者は、送信者がユーザーのパスワードを知っているかどうかを確認して、メッセージの送信者がトークンで識別されるユーザーとして振る舞うことを承認されているかどうかを検証します。

  ユーザー名トークンを使用する場合は、Enterprise Server上に有効なユーザーデータベースを設定する必要があります。
- 「デジタル署名」。Enterprise Serverは、XMLデジタル署名を使ってメッセージのコンテンツに認証IDをバインドします。クライアントはデジタル署名を使用して、呼び出し側のIDを確立します。デジタル署名は、メッセージコンテンツのソースを認証するために、メッセージ受信者によって検証されます。このソースは、メッセージ送信者と異なる可能性があります。

  デジタル署名を使用する場合は、Enterprise Server上に有効なキーストアおよびトラストストアファイルを設定する必要があります。
- 「暗号化」。暗号化の目的は、意図した相手だけが理解できるようにデータを変更することです。これは、元のコンテンツを暗号化された要素に置き換えることにより行われます。公開鍵暗号方式に基づく場合、暗号化はメッセージの読み取りを承認する関係者のIDを確立するために使用されます。

  暗号化を使用する場合は、暗号化をサポートするJava暗号化拡張機能(JCE)プロバイダをインストールする必要があります。

認証プロバイダ

「認証レイヤー」とは、認証処理を実行する必要のあるメッセージレイヤーです。Enterprise Serverは、SOAPレイヤーでWebサービスメッセージセキュリティーを適用します。サポートされている認証には次のタイプが含まれます。

- ユーザー名とパスワード認証を含む送信者認証
- XMLデジタル署名を含むコンテンツ認証

Enterprise Serverは、「認証プロバイダ」を呼び出してSOAPメッセージレイヤーセキュリティーを処理します。メッセージセキュリティープロバイダは、要求メッセージおよび応答メッセージに必要な認証のタイプなどの情報を提供します。Enterprise Serverには、次のメッセージセキュリティープロバイダが用意されています。

- 「クライアント側プロバイダ」。署名またはユーザー名とパスワードを使って要求メッセージのソースIDを確立したり、意図した受信者だけがメッセージを参照できるように暗号化を使って要求メッセージを保護したりします。また、クライアント側プロバイダは、受信した応答を正常に復号化することで、その許可さ

れた受信者としてコンテナを確立したり、応答内のパスワードまたは署名を検証してその応答に関連付けられたソース ID を認証したりもします。Enterprise Server 内に設定されているクライアント側プロバイダを使えば、ほかのサービスのクライアントとして機能するサーバー側コンポーネント (サーブレットと EJB コンポーネント) によって送信される要求メッセージと受信される応答メッセージを保護することができます。

「デフォルトクライアントプロバイダ」は、特定のクライアントプロバイダがバインドされていない任意のアプリケーションに対して呼び出されるクライアント側プロバイダを識別するために使用されます。

- 「サーバー側プロバイダ」。受信した要求を正常に復号化することで、許可された受信者としてコンテナを確立したり、要求内のパスワードまたは署名を検証してその要求に関連付けられたソース ID を認証したりします。また、サーバー側プロバイダは、署名またはユーザー名/パスワードを使って応答メッセージのソース ID を確立したり、対象の受信者だけがメッセージを参照できるように暗号化を使って要求メッセージを保護したりもします。サーバー側プロバイダを呼び出すのはサーバー側コンテナだけです。

「デフォルトサーバープロバイダ」は、特定のサーバープロバイダがバインドされていない任意のアプリケーションに対して呼び出されるサーバー側プロバイダを識別するために使用されます。

メッセージ保護ポリシー

「要求ポリシー」は、認証プロバイダが実行する要求処理に関連付けられた認証ポリシー要件を定義します。ポリシーはメッセージ送信者による順序で示されるので、「コンテンツのあと」ではメッセージ受信者が署名の検証前にメッセージを復号化することを意味します。「応答ポリシー」は、認証プロバイダが実行する応答処理に関連付けられた認証ポリシー要件を定義します。

メッセージ保護ポリシーは、要求メッセージの処理および応答メッセージの処理に対して定義されます。ポリシーは、ソース認証または受信者認証に関する要件として表現されます。プロバイダは、特定のメッセージセキュリティーメカニズムを適用することで、SOAP Web サービスメッセージにおけるメッセージ保護ポリシーを実現します。

- 「ソース認証ポリシー」。メッセージを送信したエンティティーまたはメッセージのコンテンツを定義したエンティティーの ID がメッセージ内で確立され、その ID をメッセージ受信者が認証できる、という要件を表します。
- 「受信者認証ポリシー」。メッセージを受信可能なエンティティーの ID をメッセージ送信者が確立できるようにメッセージが送信される、という要件を表します。

要求および応答メッセージ保護ポリシーは、セキュリティープロバイダがコンテナ内に設定されるときに定義されます。また、アプリケーションやアプリケーション

クライアントのSun固有の配備記述子内で、アプリケーション固有のメッセージ保護ポリシー(Webサービスのポートまたは処理の範囲でのポリシー)を設定することも可能です。メッセージ保護ポリシーを定義する場合は常に、クライアントの要求と応答に対するメッセージ保護ポリシーが、サーバー側の要求と応答に対するメッセージ保護ポリシーと同じである必要があります。アプリケーション固有のメッセージ保護ポリシーの定義については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第5章「Securing Applications」を参照してください。

アプリケーション固有のWebサービスセキュリティー

アプリケーション固有のWebサービスセキュリティー機能を(アプリケーション構築上で)設定するには、対象アプリケーションのSun固有の配備記述子内で`message-security-binding`要素を定義します。これらの`message-security-binding`要素は、特定のセキュリティープロバイダまたはメッセージ保護ポリシーをWebサービスエンドポイントまたはサービス参照に関連付けるために使用されます。また、この要素を修飾することで、それらのプロバイダやポリシーが対応するエンドポイントまたは参照サービスの特定のポートやメソッドに適用されるようにすることも可能です。

アプリケーション固有のメッセージ保護ポリシーの定義については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第5章「Securing Applications」を参照してください。

メッセージセキュリティーの管理

Enterprise Serverのインストール時に、SOAPレイヤーメッセージセキュリティープロバイダがEnterprise Serverのクライアント側コンテナとサーバー側コンテナ内に設定され、コンテナまたはコンテナ内に配備された個々のアプリケーションまたはクライアントからバインドして利用できるようになります。インストール中、デフォルトのプロバイダには単純なメッセージ保護ポリシーが設定されます。このポリシーをコンテナまたはコンテナ内のアプリケーションまたはクライアントにバインドした場合、すべての要求メッセージと応答メッセージに含まれるコンテンツのソースが、XMLデジタル署名によって認証されるようになります。

Enterprise Serverの管理インタフェースを使用して、次の操作を実行できます。

- プロバイダによって適用されるメッセージ保護ポリシーを変更します。
- Enterprise Serverのサーバー側コンテナで使用できるように、既存のプロバイダをバインドします。
- 別のメッセージ保護ポリシーを使用して新しいセキュリティープロバイダの構成を作成します。

アプリケーションクライアントコンテナの SOAP メッセージレイヤーセキュリティー構成でも、これと同様の管理操作を実行できます。Enterprise Server に配備されたすべての Web サービスアプリケーションを Web サービスセキュリティーで保護する場合は、240 ページの「アプリケーションクライアントのメッセージセキュリティーの有効化」を参照してください。

Enterprise Server では、メッセージレイヤーセキュリティーはデフォルトで無効になっています。Enterprise Server でメッセージレイヤーセキュリティーを設定するには、232 ページの「Web サービスのデフォルトメッセージセキュリティープロバイダの有効化」を参照してください。

ほとんどの場合、管理タスクを実行したあとに Enterprise Server を再起動する必要があります。特に、操作実行時に Enterprise Server にすでに配備されているアプリケーションに対して管理上の変更を適用する場合は、これに該当します。

メッセージセキュリティーのタスク

メッセージセキュリティーの一般的な実装タスクには、次のタスクの一部またはすべてが含まれます。

1. バージョン 1.5.0 より前のバージョンの Java SDK を使用し、暗号化テクノロジを使用する場合は、JCE プロバイダを設定します。
2. ユーザー名トークンを使用している場合は、ユーザーデータベースが適切なレルムに対して設定されていることを確認します。

   ユーザー名およびパスワードトークンを使用する場合は、適切なレルムを設定し、このレルムに対してユーザーデータベースを設定する必要があります。
3. 必要に応じて証明書と非公開鍵を管理します。
4. Enterprise Server のデフォルトのプロバイダを有効にします。
5. 新しいメッセージセキュリティープロバイダを設定します。

メッセージセキュリティーのロール

Enterprise Server では、メッセージセキュリティー設定の主要責任者として、管理者とアプリケーション配備担当者が適任です。状況に応じて、アプリケーション開発者もこれに加わります。

システム管理者

システム管理者は、次のメッセージセキュリティータスクに責任を持ちます。

- サーバーセキュリティー設定と証明書データベースの管理
- キーストアおよびトラストストアファイルの管理
- Enterprise Server 上のメッセージセキュリティープロバイダの設定
- メッセージセキュリティーの有効化
- サンプルサーバーのインストール (必要な場合のみ)

アプリケーション配備担当者

アプリケーション配備担当者は、次のメッセージセキュリティータスクに責任を持ちます。

- 必要なすべてのアプリケーション固有メッセージ保護ポリシーをアプリケーションの再アセンブリ時に指定 (それらのポリシーが開発者またはプログラマによって指定されていない場合)。
- Sun 固有の配備記述子を変更し、アプリケーション固有メッセージ保護ポリシー情報 (message-security-binding 要素) を Web サービスエンドポイントとサービス参照に指定。

アプリケーション開発者およびアセンブリ担当者

アプリケーション開発者およびアセンブリ担当者は、次のメッセージセキュリティータスクに責任を持ちます。

- アプリケーションに固有のメッセージ保護ポリシーがアプリケーションで必要かどうかの判断

  必要な場合は、開発者がアプリケーションのアセンブリ時に必要なポリシーを指定する必要があります。
- メッセージセキュリティーに対する Web サービスの設定方法の指定

  メッセージセキュリティーの設定を管理者が行う場合、すべての Web サービスがセキュリティー保護されます。コンテナにバインドされているセキュリティープロバイダまたは保護ポリシーと異なるものをアプリケーションにバインドする必要がある場合は、アプリケーション配備担当者がメッセージセキュリティーの設定を行います。
- メッセージセキュリティーの有効化 (管理者によって承認されている場合)

Web サービスのサンプルアプリケーション

Enterprise Server には、`xms` という名前のサンプルアプリケーションが用意されています。`xms` アプリケーションは、Java EE EJB エンドポイントと Java サーブレットエンドポイントの両方を使って実装された、単純な Web サービスです。両エンドポイントは同一のサービスエンドポイントインタフェースを共有しています。サービスエンドポイントインタフェースには、`sayHello` という処理のみが定義されています。この処理は、文字列の引数を 1 つ受け取り、その呼び出し引数の前に `Hello` を付加した `String` を返します。

`xms` サンプルアプリケーションは、Enterprise Server の WS-Security 機能を使って、既存の Web サービスアプリケーションをセキュリティー保護する方法を示すことを目的としています。サンプルに付属する手順では、Enterprise Server の WS-Security 機能を有効にして `xms` アプリケーションを保護する方法が説明されています。また、こ

のサンプルは、229 ページの「アプリケーション固有の Web サービスセキュリティー」アプリケーションで説明されているように、WS-Security 機能をアプリケーションに直接バインドする方法も示しています。

xms サンプルアプリケーションのコンパイル、パッケージ化、および実行については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 5 章「Securing Applications」を参照してください。

xms サンプルアプリケーションは、*as-install*/samples/webservices/security/ejb/apps/xms/ にインストールされます。

Web サービスのデフォルトメッセージセキュリティープロバイダの有効化

Enterprise Server では、メッセージセキュリティーはデフォルトで無効になっています。デフォルトメッセージセキュリティープロバイダは作成されていますが、有効化するまではアクティブになりません。プロバイダを有効にしたあと、メッセージセキュリティーが有効になります。

ここでは、次のテーマを取り上げます。

- 232 ページの「デフォルトサーバープロバイダを有効にする」
- 233 ページの「デフォルトクライアントプロバイダを有効にする」

▼ デフォルトサーバープロバイダを有効にする

Enterprise Server に配備された Web サービスエンドポイントのメッセージセキュリティーを有効にするには、サーバー側でデフォルトで使用されるセキュリティープロバイダを指定する必要があります。メッセージセキュリティーのデフォルトのプロバイダを有効にする場合、Enterprise Server に配備された Web サービスのクライアントが使用するプロバイダも有効にする必要があります。

1 set(1) サブコマンドを使用して、デフォルトサーバープロバイダを指定します。

次の構文を使用します。

```
asadmin  set --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

2 すでに実行されているアプリケーションに変更を適用するには、**Enterprise Server** を再起動します。

94 ページの「ドメインの再起動」を参照してください。

▼ デフォルトクライアントプロバイダを有効にする

配備済みエンドポイントからの Web サービス呼び出しに対してメッセージセキュリティーを有効にするには、デフォルトクライアントプロバイダを指定する必要があります。Enterprise Server のデフォルトクライアントプロバイダを有効にした場合、Enterprise Server に配備されたエンドポイントから呼び出されるすべてのサービスが、メッセージ層セキュリティーと互換性を持つように設定されていることを確認する必要があります。

1 `set(1)` サブコマンドを使用して、デフォルトクライアントプロバイダを指定します。
次の構文を使用します。

```
asadmin  set --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

2 すでに実行されているアプリケーションに変更を適用するには、**Enterprise Server** を再起動します。
94 ページの「ドメインの再起動」を参照してください。

メッセージ保護ポリシーの設定

メッセージ保護ポリシーは、要求メッセージの処理および応答メッセージの処理に対して定義されます。ポリシーは、ソース認証または受信者認証に関する要件として表現されます。プロバイダは、特定のメッセージセキュリティーメカニズムを適用することで、SOAP Web サービスメッセージにおけるメッセージ保護ポリシーを実現します。

ここでは、次のテーマを取り上げます。

メッセージ保護ポリシーの設定と処理の対応

次の表に、メッセージ保護ポリシーの構成と、その設定の結果として WS-Security SOAP メッセージセキュリティープロバイダが実行するメッセージセキュリティー処理を示します。

表 13–1　メッセージ保護ポリシーと WS-Security SOAP 処理の対応

メッセージ保護ポリシー	結果として実行される **WS-Security SOAP** メッセージ保護処理
auth-source="sender"	メッセージに `wsse:Security` ヘッダーが格納され、そのヘッダー内に `wsse:UsernameToken` (パスワード付き) が格納されます。
auth-source="content"	SOAP メッセージ本体のコンテンツが署名されます。メッセージに `wsse:Security` ヘッダーが格納され、そのヘッダー内にメッセージ本体の署名が `ds:Signature` として格納されます。
auth-source="sender" *auth-recipient="before-content"* または *auth-recipient="after-content"*	SOAP メッセージ本体のコンテンツが暗号化され、その結果得られた `xend:EncryptedData` で置換されます。メッセージに `wsse:Security` ヘッダーが格納され、そのヘッダー内に `wsse:UsernameToken` (パスワード付き) と `xenc:EncryptedKey` が格納されます。また、`xenc:EncryptedKey` には SOAP メッセージ本文の暗号化に使用する鍵が含まれます。この鍵は、受信者の公開鍵内で暗号化されています。
auth-source="content" *auth-recipient="before-content"*	SOAP メッセージ本体のコンテンツが暗号化され、その結果得られた `xend:EncryptedData` で置換されます。`xenc:EncryptedData` は署名されています。メッセージに `wsse:Security` ヘッダーが格納され、そのヘッダー内に `xenc:EncryptedKey` と `ds:Signature` が格納されます。また、`xenc:EncryptedKey` には SOAP メッセージ本文の暗号化に使用する鍵が含まれます。この鍵は、受信者の公開鍵内で暗号化されています。
auth-source="content" *auth-recipient="after-content"*	SOAP メッセージ本体のコンテンツが、署名されたあと暗号化され、その結果得られた `xend:EncryptedData` で置換されます。メッセージに `wsse:Security` ヘッダーが格納され、そのヘッダー内に `xenc:EncryptedKey` と `ds:Signature` が格納されます。また、`xenc:EncryptedKey` には SOAP メッセージ本文の暗号化に使用する鍵が含まれます。この鍵は、受信者の公開鍵内で暗号化されています。
auth-recipient="before-content" または *auth-recipient="after-content"*	SOAP メッセージ本体のコンテンツが暗号化され、その結果得られた `xend:EncryptedData` で置換されます。メッセージには、`xenc:EncryptedKey` を含む `wsse:Security` ヘッダーが含まれます。また、`xenc:EncryptedKey` には SOAP メッセージ本文の暗号化に使用する鍵が含まれます。この鍵は、受信者の公開鍵内で暗号化されています。

表 13–1　メッセージ保護ポリシーと WS-Security SOAP 処理の対応　(続き)

メッセージ保護ポリシー	結果として実行される WS-Security SOAP メッセージ保護処理
ポリシーを何も指定しない。	モジュールはセキュリティー処理を一切行いません。

▼ プロバイダのメッセージ保護ポリシーを設定する

一般的には、プロバイダを再設定することはありません。ただし必要な場合は、プロバイダのタイプ、実装クラス、およびプロバイダ固有の構成プロパティーを変更して、プロバイダのメッセージ保護ポリシーを変更できます。設定の組み合わせの結果については、表 13–1 を参照してください。

set(1) サブコマンドを使用して応答ポリシーを設定します。そのあと、各コマンドの request の文字を response に置き換えて実行します。

1. set(1) サブコマンドを使用して、要求ポリシーをクライアントに追加し、認証のソースを設定します。

   次に例を示します。

   ```
   asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ClientProvider.request-policy.auth_source=[sender | content]
   ```

2. set サブコマンドを使用して、要求ポリシーをサーバーに追加し、認証のソースを設定します。

   次に例を示します。

   ```
   asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ServerProvider.request-policy.auth_source=[sender | content]
   ```

3. set サブコマンドを使用して、要求ポリシーをクライアントに追加し、認証の受信者を設定します。

   次に例を示します。

```
asadmin> set server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=[before-content | after-content]
```

4. set サブコマンドを使用して、要求ポリシーをサーバーに追加し、認証の受信者を設定します。

   次に例を示します。

```
asadmin> set server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=[before-content | after-content]
```

アプリケーションクライアント構成の要求および応答ポリシーの設定

「要求および応答ポリシー」は、認証プロバイダが実行する要求および応答処理に関連付けられた認証ポリシー要件を定義します。ポリシーはメッセージ送信者による順序で示されるので、「コンテンツのあと」ではメッセージ受信者が署名の検証前にメッセージを復号化することを意味します。

メッセージセキュリティーを実現するには、サーバーとクライアントの両方で要求ポリシーと応答ポリシーが有効化されている必要があります。クライアントおよびサーバーのポリシーを設定する場合は、クライアントポリシーがアプリケーションレベルのメッセージのバインドで要求および応答保護のサーバーポリシーと一致する必要があります。

アプリケーションクライアント構成の要求ポリシーを設定するには、240 ページの「アプリケーションクライアントのメッセージセキュリティーの有効化」の説明に従って、アプリケーションクライアントコンテナの Enterprise Server 固有の構成を変更します。

例 13–1　アプリケーションクライアントのメッセージセキュリティーのポリシー設定

アプリケーションクライアント構成ファイルの `request-policy` 要素および `response-policy` 要素は、次のコード例に示すように、要求ポリシーの設定に使用されます (要素を定義している箇所以外のコードは説明用のものです。実際のインストールでは異なる場合があります。これらのコードは変更しないでください)。

```
<client-container>
  <target-server name="your-host" address="your-host"
      port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
      default-client-provider="ClientProvider">
    <provider-config
        class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
        provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
           value="as-install/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

auth-source の有効な値には、sender と content があります。auth-recipient の有効な値には、before-content と after-content があります。これらの値の組み合わせの結果については、233 ページの「メッセージ保護ポリシーの設定」の表を参照してください。

要求または応答ポリシーを指定しない場合は、この要素を空白のままにします。次に例を示します。

```
<response-policy/>
```

デフォルト以外のメッセージセキュリティープロバイダの管理

ここでは、次のテーマを取り上げます。

▼ メッセージセキュリティープロバイダを作成する

セキュリティーサービスの新しいメッセージプロバイダを作成するには、リモートモードで create–message–security–provider サブコマンドを使用します。メッセージレイヤーが存在しない場合は、メッセージレイヤーが作成され、そのレイヤーにプロバイダが作成されます。

1. **サーバーが実行されていることを確認します。**
   リモートサブコマンドには、実行中のサーバーが必要です。

2. **create-message-security-provider(1) サブコマンドを使用して、メッセージセキュリティープロバイダを作成します。**
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. **(省略可能) 必要な場合は、サーバーを再起動します。**
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 13–2 メッセージセキュリティープロバイダの作成

この例では、mySecurityProvider という名前の新しいメッセージセキュリティープロバイダを作成します。

```
asadmin> create-message-security-provider
--classname com.sun.enterprise.security.jauth.ClientAuthModule
--providertype client mySecurityProvider
Command create-message-security-provider executed successfully.
```

参照 コマンド行に asadmin help create–message–security–provider と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ メッセージセキュリティープロバイダを一覧表示する

セキュリティーレイヤーのメッセージプロバイダを一覧表示するには、リモートモードで list–message–security–providers サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-message-security-providers(1) サブコマンドを使用して、メッセージセキュリティープロバイダを一覧表示します。

例 13–3 メッセージセキュリティープロバイダの一覧表示

この例では、メッセージレイヤーのメッセージセキュリティープロバイダを一覧表示します。

```
asadmin> list-message-security-providers --layer SOAP
XWS_ClientProvider
ClientProvider
XWS_ServerProvider
ServerProvider
Command list-message-security-providers executed successfully.
```

参照 コマンド行に asadmin help list–message–security–providers と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ メッセージセキュリティープロバイダを更新する

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-message-security-providers(1)` サブコマンドを使用して、メッセージセキュリティープロバイダを一覧表示します。
3. `set(1)` サブコマンドを使用して、指定したメッセージセキュリティープロバイダの値を変更します。
   メッセージセキュリティープロバイダは、ドット表記名で識別されます。

▼ メッセージセキュリティープロバイダを削除する

メッセージセキュリティープロバイダを削除するには、リモートモードで `delete-message-security-provider` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-message-security-providers(1)` サブコマンドを使用して、メッセージセキュリティープロバイダを一覧表示します。
3. `delete-message-security-provider(1)` サブコマンドを使用して、メッセージセキュリティープロバイダを削除します。

例 13–4 メッセージセキュリティープロバイダの削除

この例では、`myServerityProvider` という名前のメッセージセキュリティープロバイダを削除します。

```
asadmin> delete-message-security-provider --layer SOAP myServerityProvider
Command delete-message-security-provider executed successfully.
```

参照 コマンド行に `asadmin help delete–message–security–provider` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

アプリケーションクライアントのメッセージセキュリティーの有効化

クライアントプロバイダのメッセージ保護ポリシーは、通信相手となるサーバー側プロバイダのメッセージ保護ポリシーと等しくなるように設定する必要があります。Enterprise Server のインストール時に、プロバイダはデフォルトでそのように設定されます (ただし、有効化されていません)。

クライアントアプリケーションのメッセージセキュリティーを有効にするには、アプリケーションクライアントコンテナの Enterprise Server 固有の構成を変更します。処理は、233 ページの「メッセージ保護ポリシーの設定」の処理と同様です。

メッセージセキュリティープロバイダに関する追加情報

メッセージセキュリティーの追加情報については、次を参照してください。

- 『*Java EE 6.0 Tutorial*』の「*Security*」の章 (`http://java.sun.com/javaee/6/docs/tutorial/doc/index.html`)
- 『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 5 章「Securing Applications」

パート III

リソースとサービスの管理

◆ ◆ ◆ 第 14 章

データベース接続の管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server v3 環境でデータベース接続タスクを実行する手順について説明します。

ここでは、次のテーマを取り上げます。

- 243 ページの「データベース接続について」
- 244 ページの「データベースの設定」
- 248 ページの「データベースへのアクセスの設定」
- 256 ページの「JDBC ドライバに固有の構成」

これらのタスクを管理コンソールを使用して実行する場合の手順は、管理コンソールのオンラインヘルプで説明します。

データベース接続について

データベース管理システム (DBMS) は、データを格納、編成、および取得するための機能を提供します。多くの場合、データベース内の情報は持続的なデータとして表現されますが、これはデータがディスク上に保存され、アプリケーションプロセスが終了したあとも存在するためです。ほとんどのビジネスアプリケーションは、データをリレーショナルデータベースに格納します。アプリケーションは JDBC (Java Database Connectivity) API を使用して、データベース情報にアクセスできます。

データベース接続の主な要素は次のとおりです。

- 「データベース」。エンタープライズのデータが格納されるリポジトリです。Java EE アプリケーションは、JDBC API を介してリレーショナルデータベースにアクセスします。管理の手順については、244 ページの「データベースの設定」を参照してください。
- 「**JDBC** 接続プール」。特定のデータベースのための再利用可能な接続のグループです。管理の手順については、248 ページの「JDBC 接続プールの管理」を参照してください。

- 「**JDBC** リソース」(データソース)。データベースに接続する手段をアプリケーションに提供します。JDBC リソースを作成するには、関連した接続プールを指定します。複数の JDBC リソースが 1 つの接続プールを指定することもできます。JDBC リソースは、Java Naming Directory Interface (JNDI) 名によって識別されます。管理の手順については、253 ページの「JDBC リソースの管理」を参照してください。
- 「**JDBC** ドライバ」。データベースドライバは、Java アプリケーションがデータベース接続 API とやり取りを行うためのソフトウェアコンポーネントです。データベースごとに専用のドライバが必要です。管理の手順については、255 ページの「JDBC ドライバの統合」を参照してください。

実行時には、アプリケーションがデータベースに接続するときに次の一連の処理が発生します。

1. アプリケーションは JNDI API を通して呼び出しを行い、データベースに関連付けられた JDBC リソースを取得します。

   リソースの JNDI 名を使用して、ネームサービスとディレクトリサービスが JDBC リソースを検索します。JDBC リソースはそれぞれ接続プールを指定します。
2. アプリケーションは JDBC リソースを使用してデータベース接続を取得します。

   Enterprise Server は、データベースに対応する接続プールから物理接続を取得します。プールは、データベース名 (URL)、ユーザー名、パスワードなどの接続属性を定義します。
3. データベース接続が確立されると、アプリケーションはデータベースに対してデータの読み取り、変更、および追加を実行できるようになります。

   アプリケーションは JDBC API を呼び出すことにより、データベースにアクセスします。JDBC ドライバはアプリケーションの JDBC 呼び出しをデータベースサーバーのプロトコルに変換します。
4. データベースへのアクセスが完了すると、アプリケーションは接続を閉じ、接続を接続プールに戻します。

データベースの設定

多くのアプリケーションは、リレーショナルデータベースを使用してデータを保存、編成、および取得します。アプリケーションは、JDBC (Java™ Database Connectivity) API を通してリレーショナルデータベースにアクセスします。

ここでは、次のテーマを取り上げます。

▼ データベースおよびデータベースドライバをインストールする

1 サポートされたデータベース製品をインストールします。

Enterprise Server でサポートされているデータベース製品の最新のリストを確認するには、『Sun GlassFish Enterprise Server v3 リリースノート』を参照してください。

2 データベース製品用のサポートされている **JDBC** ドライバをインストールします。

Enterprise Server でサポートされているドライバのリストについては、256 ページの「JDBC ドライバに固有の構成」を参照してください。

3 ドメイン管理サーバー **(DAS)** が **JDBC** ドライバの **JAR** ファイルにアクセスできるようにします。

255 ページの「JDBC ドライバの統合」を参照してください。

4 データベースを作成します。

通常はアプリケーションプロバイダが、データベースを作成しデータを生成するためのスクリプトを提供しています。

▼ データベースを起動する

Enterprise Server には、Java DB (以前の名称は Derby) の実装が含まれていますが、JDBC に準拠した任意のデータベースも使用できます。データベースは Enterprise Server の起動時に自動では起動されません。したがって、データベースを必要とするアプリケーションを使用する場合は、`start-database` サブコマンドを使用して Java DB を手動で起動する必要があります。

- `start-database(1)` サブコマンドを使用して、データベースを起動します。

  データベースサーバーの起動時、またはクライアントがデータベースサーバーに正常に接続したときに、`--dbhome` オプションで指定された場所に次のファイルが作成されます。

  - `derby.log` ファイルには、データベースサーバープロセスのログが、標準出力および標準エラー情報とともに保存されます。
  - データベースのファイルには、使用するスキーマ (たとえば、データベース表) が保存されます。

例 14–1 データベースの起動

この例では、host1 というホストのポート 5001 で Derby を起動します。

```
asadmin> start-database --dbhost host1 --dbport 5001 --terse=true
Starting database in the background.
Log redirected to /opt/SUNWappserver/databases/javadb.log.
Command start-database executed successfully.
```

参照　コマンド行に asadmin help start-database と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ データベースを停止する

ローカルの stop-database サブコマンドを使用して、指定したポートの Java DB を停止します。同一のホストのほかのポートで、複数のデータベースサーバープロセスが動作している場合があります。

1. 必要に応じて、データベースを停止することをユーザーに通知します。
2. stop-database(1) サブコマンドを使用して、データベースを停止します。

例 14–2　データベースの停止

この例では、localhost のポート 5001 で動作している Java DB を停止します。

```
asadmin> stop-database --dbhost=localhost --dbport=5001
onnection obtained for host: localhost, port number 5001.
Apache Derby Network Server - 10.2.2.1 - (538595) shutdown at 2008-10-17 23:34:2
7.218 GMT
Command stop-database executed successfully.
```

注意事項　ネットワーク間を移動するノートパソコンでは、データベースのシャットダウンに関して問題が発生する場合があります。Java DB を起動したあとに IP アドレスを変更した場合は、--dbhost 引数を指定しなければ Java DB を停止できません。たとえば、asadmin start-database --dbhost = 0.0.0.0 を実行したあと、Ethernet を切断してワイヤレス接続に切り替えた場合、データベースを停止するには次のようなコマンドを実行する必要があります。

```
asadmin stop-database --dbhost localhost
```

参照　コマンド行に asadmin help stop-database と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

Java DB ユーティリティースクリプト

Enterprise Server で利用できる Java DB の構成には、Java DB の使用に役立つスクリプトが含まれます。次のスクリプトが、*as-install*/javadb/frameworks/NetworkServer/bin ディレクトリに格納されています。

`startNetworkServer`、`startNetworkServer.bat`
: ネットワークサーバーを起動するスクリプト

`stopNetworkServer`、`stopNetworkServer.bat`
: ネットワークサーバーを停止するスクリプト

`ij`、`ij.bat`
: 対話式の JDBC スクリプト作成ツール

`dblook`、`dblook.bat`
: データベースのすべてまたは一部の DDL を表示するスクリプト

`sysinfo`、`sysinfo.bat`
: Java DB 環境に関するバージョン情報を表示するスクリプト

`NetworkServerControl`、`NetworkServerControl.bat`
: `NetworkServerControl` API でコマンドを実行するスクリプト

▼ Java DB ユーティリティースクリプトを実行するための環境を設定する

1. `JAVA_HOME` 環境変数が **JDK** のインストールディレクトリを指定していることを確認します。
2. *as-install*/derby ディレクトリをポイントするように `JAVADB_HOME` 環境変数を設定します。

参照 これらのユーティリティーの詳細については、次のドキュメントを参照してください。

- 『*Derby Tools and Utilities Guide*』 (`http://db.apache.org/derby/docs/10.1/tools/`)
- 『*Derby Server and Administration Guide*』 (`http://db.apache.org/derby/docs/10.1/adminguide/`)

データベースへのアクセスの設定

データベース接続を確立したら、Enterprise Server アプリケーションのアクセス設定を実行できます。データベースにアクセスする前に、アプリケーションは接続を取得する必要があります。

ここでは、次のテーマを取り上げます。

JDBC 接続プールの管理

「*JDBC* 接続プール」は、特定のデータベースのための再利用可能な接続のグループです。新しい物理接続の作成には時間がかかるため、Enterprise Server は使用可能な接続のプールを維持します。アプリケーションが接続を要求すると、プールから 1 つの接続が取得されます。アプリケーションが接続を閉じると、接続はプールに返されます。

JDBC リソースは、リソースが関連付けられている接続プールを指定することで作成されます。複数の JDBC リソースが 1 つの接続プールを指定することもできます。接続プールのプロパティーは、データベースベンダーによっては異なる場合もあります。共通のプロパティーには、データベースの名前 (URL)、ユーザー名、パスワードなどがあります。

次のタスクと情報を使用して、JDBC 接続プールを管理します。

▼ JDBC 接続プールを作成する

指定した JDBC 接続プール名で新しい JDBC 接続プールを登録するには、リモートモードで `create-jdbc-connection-pool` サブコマンドを使用します。。JDBC 接続プールまたはコネクタ接続プールは、認証を使用して作成できます。`asadmin` ユーティリティーでユーザー、パスワード、またはその他の接続情報を指定するサブコマンドオプションを使用するか、XML 記述子ファイルで接続情報を指定します。

各データベースには接続プールが 1 つ必要です。アプリケーションによっては、複数の接続プールが必要な場合もあります。接続プールを構築するときに、JDBC ドライバとデータベースベンダーに固有のデータが必要となります。次に示す固有データの例の一部は、256 ページの「JDBC ドライバに固有の構成」にも示してあります。

- データベースベンダー名
- `javax.sql.DataSource` (ローカルトランザクションのみ) や `javax.sql.XADataSource` (グローバルトランザクション) などのリソースタイプ
- データソースクラス名
- データベース名 (URL)、ユーザー名、およびパスワードなどの必要なプロパティー

JDBC 接続プールの作成は動的なイベントであり、サーバーの再起動は必要ありません。ただし、パラメータの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。

始める前に　接続プールを作成する前に、データベースとデータベースに関連する JDBC ドライバをインストールして統合しておく必要があります。手順については、244 ページの「データベースの設定」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-jdbc-connection-pool(1)` サブコマンドを使用して、**JDBC** 接続プールを作成します。

3. (省略可能) 必要な場合は、サーバーを再起動します。
   一部のパラメータはサーバーの再起動を必要とします。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。

例 14–3 JDBC 接続プールの作成

この例では、`sample_derby_pool` という名前の接続プールを `localhost` に作成します。

```
asadmin> create-jdbc-connection-pool
--datasourceclassname org.apache.derby.jdbc.ClientDataSource
--restype javax.sql.XADataSource
--property portNumber=1527:password=APP:user=APP:serverName=
localhost:databaseName=sun-appserv-samples:connectionAttribut
es=\;create\\=true sample_derby_pool
Command create-jdbc-connection-pool executed successfully.
```

参照　コマンド行に asadmin help create-jdbc-connection-pool と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JDBC 接続プールを一覧表示する

既存の JDBC 接続プールをすべて表示するには、リモートモードで list-jdbc-connection-pools サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-jdbc-connection-pools(1) サブコマンドを使用して、JDBC 接続プールを一覧表示します。

例 14–4　JDBC 接続プールの一覧表示

この例では、localhost 上の JDBC 接続プールを一覧表示します。

```
asadmin> list-jdbc-connection-pools
sample_derby_pool2
poolA
__TimerPool
DerbyPool
sample_derby_pool
Command list-jdbc-connection-pools executed successfully.
```

参照　コマンド行に asadmin help list-jdbc-connection-pools と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 接続プールと通信する (ping を実行する)

接続プールが使用可能かどうかをテストするには、リモートモードで ping-connection-pool サブコマンドを使用します。たとえば、あとで配備する予定のアプリケーション用に新しい JDBC 接続プールを作成した場合、そのアプリケーションを配備する前に、このコマンドを使用して JDBC プールをテストすることができます。ping を実行すると、プールがまだ作成されていない場合は作成を強制されます。

始める前に　接続プールと通信する前に、認証を使用して接続プールを作成し、サーバーまたはデータベースを実行しておく必要があります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 ping-connection-pool(1) サブコマンドを使用して、接続プールに **ping** を実行します。

例 **14–5** 接続プールとの通信

この例では、DerbyPool 接続プールが使用可能かどうかを確認します。

```
asadmin> ping-connection-pool DerbyPool
Command ping-connection-pool executed successfully
```

参照 コマンド行に asadmin help ping-connection-pool と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 接続プールをリセット (フラッシュ) する

指定した接続プールで確立されたすべての接続を再初期化するには、リモートモードで flush-connection-pool を使用します。JDBC 接続プールまたはコネクタ接続プールは、初期状態にリセットされます。既存の動作中の接続はすべて破棄され、これらの接続に関連付けられているトランザクションは失われます。続いてプールの初期接続が再作成され、プールは通常プールサイズに復元されます。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 flush-connection-pool(1) サブコマンドを使用して、接続プールをリセットします。

例 **14–6** 接続プールのリセット (フラッシュ)

この例では、__TimerPool という名前の JDBC 接続プールを通常プールサイズにリセットします。

```
asadmin> flush-connection-pool __TimerPool
Command flush-connection-pool executed successfully.
```

参照 コマンド行に asadmin help flush-connection-pool と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JDBC 接続プールを更新する

get および set サブコマンドを使用して、JDBC 接続プールのプロパティーの値を表示および変更します。

1 list-jdbc-connection-pools(1) サブコマンドを使用して、**JDBC** 接続プールを一覧表示します。

2. **get** サブコマンドを使用して、**JDBC** 接続プールの属性を表示します。
   次に例を示します。

   ```
   asadmin get resources.jdbc-connection-pool.DerbyPool.property
   ```

3. **set** サブコマンドを使用して、**JDBC** 接続プールの属性を設定します。
   次に例を示します。

   ```
   asadmin set resources.jdbc-connection-pool.DerbyPool.steady-pool-size=9
   ```

4. (省略可能) 必要な場合は、サーバーを再起動します。
   一部のパラメータはサーバーの再起動を必要とします。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。

▼ JDBC 接続プールを削除する

既存の JDBC 接続プールを削除するには、リモートモードで `delete-jdbc-connection-pool` サブコマンドを使用します。JDBC 接続プールの削除は動的なイベントであり、サーバーの再起動は必要ありません。

始める前に　JDBC 接続プールを削除する前に、リソースのすべての関連付けを削除する必要があります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `list-jdbc-connection-pools(1)` サブコマンドを使用して、**JDBC** 接続プールを一覧表示します。

3. 必要に応じて、**JDBC** 接続プールを削除することをユーザーに通知します。

4. `delete-jdbc-connection-pool(1)` サブコマンドを使用して、接続プールを削除します。

例 14–7　JDBC 接続プールの削除

この例では、`DerbyPool` という名前の JDBC 接続プールを削除します。

```
asadmin> delete-jdbc-connection-pool jdbc/DerbyPool
Command delete-jdbc-connection-pool executed successfully.
```

参照　コマンド行に `asadmin help delete-jdbc-connection-pool` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

JDBC リソースの管理

「*JDBC* リソース」はデータソースとも呼ばれ、アプリケーションがデータベースに接続する手段を提供します。一般的には、ドメインに配備されたアプリケーションがアクセスするデータベースごとに 1 つの JDBC リソースを作成します。1 つのデータベースに複数の JDBC リソースを指定することもできます。

JDBC リソースは、リソースを関連付ける接続プールを指定することで作成されます。一意の Java Naming and Directory Interface (JNDI) 名を使用して、リソースを識別します。たとえば、給与データベースのリソースには、`java:comp/env/jdbc/payrolldb` のような JNDI 名を付けることができます。

次のタスクと情報を使用して、JDBC リソースを管理します。

- 253 ページの「JDBC リソースを作成する」
- 254 ページの「JDBC リソースを一覧表示する」
- 254 ページの「JDBC リソースを更新する」
- 255 ページの「JDBC リソースを削除する」

▼ JDBC リソースを作成する

JDBC リソースを作成するには、リモートモードで `create-jdbc-resource` サブコマンドを使用します。JDBC リソースの作成は動的なイベントであり、サーバーの再起動は必要ありません。

すべての JNDI 名は `java:comp/env` サブコンテキストにあるので、管理コンソールで JDBC リソースの JNDI を指定するときは、`jdbc/`*name* の形式だけを使用します。たとえば、先に述べた給与データベースは、`jdbc/payrolldb` のように指定できます。

始める前に JDBC リソースを作成する前に、JDBC 接続プールを作成する必要があります。手順については、248 ページの「JDBC 接続プールを作成する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `create-jdbc-resource(1)` サブコマンドを使用して、**JDBC** リソースを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。
3. 必要に応じて、新しいリソースを作成したことをユーザーに通知します。

例 14–8 JDBC リソースの作成

この例では、`DerbyPool` という名前の JDBC リソースを作成します。

```
asadmin> create-jdbc-resource --connectionpoolid DerbyPool jdbc/DerbyPool
Command create-jdbc-resource executed successfully.
```

参照　コマンド行に asadmin help create-jdbc-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JDBC リソースを一覧表示する

既存の JDBC リソースを一覧表示するには、リモートモードで list-jdbc-resources サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-jdbc-resources(1) サブコマンドを使用して、JDBC リソースを一覧表示します。

例 14–9　JDBC リソースの一覧表示

この例では、localhost の JDBC リソースを一覧表示します。

```
asadmin> list-jdbc-resources
jdbc/__TimerPool
jdbc/DerbyPool
jdbc/__default
jdbc1
Command list-jdbc-resources executed successfully.
```

参照　コマンド行に asadmin help list-jdbc-resources と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JDBC リソースを更新する

set サブコマンドを使用して、JDBC リソースを有効または無効にできます。JDBC リソースはドット表記名で識別されます。

1. list-jdbc-resources(1) サブコマンドを使用して、JDBC リソースを一覧表示します。
2. set(1) サブコマンドを使用して、指定した JDBC リソースの値を変更します。
   次に例を示します。

例 14–10　JDBC リソースの更新

この例では、res1 の enabled の設定を false に変更します。

```
asadmin>set resources.jdbc-resource.res1.enabled=false
```

▼ JDBC リソースを削除する

既存の JDBC リソースを削除するには、リモートモードで `delete-jdbc-resource` サブコマンドを使用します。JDBC リソースの削除は動的なイベントであり、サーバーの再起動は必要ありません。

始める前に　JDBC リソースを削除する前に、削除するリソースのすべての関連付けを削除する必要があります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-jdbc-resources(1)` サブコマンドを使用して、**JDBC** リソースを一覧表示します。
3. 必要に応じて、**JDBC** リソースを削除することをユーザーに通知します。
4. `delete-jdbc-resource(1)` サブコマンドを使用して、**JDBC** リソースを削除します。

例 **14–11**　JDBC リソースの削除

この例では、`DerbyPool` という名前の JDBC リソースを削除します。

```
asadmin> delete-jdbc-resource jdbc/DerbyPool
Command delete-jdbc-resource executed successfully.
```

参照　コマンド行に `asadmin help delete-jdbc-resource` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

JDBC ドライバの統合

接続プールとリソースを設定したあと、次のいずれかの方法で JDBC ドライバを統合します。

- 共通のクラスローダーがドライバにアクセスできるようにして、ドメインを再起動します。
  ドライバの JAR および ZIP ファイルを、*domain-dir*`/lib` ディレクトリまたは `glassfish_home/lib` にコピーするか、ドライバのクラスファイルを *domain-dir*`/lib/ext` ディレクトリにコピーします。ドライバの JAR ファイルの完全修飾パス名が認識されます。

JDBC ドライバに固有の構成

Enterprise Server は、対応する JDBC ドライバを使用して、すべてのデータベース管理システムに接続できるように設計されています。

完全にサポートされている JDBC ドライバ

次の JDBC ドライバとデータベースの組み合わせはテスト済みで、コンテナ管理による持続性がサポートされています。

サポートされている JDBC ドライバの最新のリストについては、『Sun GlassFish Enterprise Server v3 リリースノート』を参照してください。

DB2 データベース用の Sun GlassFish JDBC ドライバ

このドライバの JAR ファイルは `smdb2.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: DB2
- 「データソースクラス名」: `com.sun.sql.jdbcx.db2.DB2DataSource`
- 「プロパティー」:
  - **`serverName`** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **`portNumber`** – データベースサーバーのポート番号を指定します。
  - **`databaseName`** – 必要に応じて設定します。
  - **`user`** – 必要に応じて設定します。
  - **`password`** – 必要に応じて設定します。

Oracle 11 データベース用の Sun GlassFish JDBC ドライバ

このドライバの JAR ファイルは `smoracle.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Oracle
- 「データソースクラス名」: `com.sun.sql.jdbcx.oracle.OracleDataSource`
- 「プロパティー」:
  - **`serverName`** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **`portNumber`** – データベースサーバーのポート番号を指定します。
  - **`user`** – 必要に応じて設定します。
  - **`password`** – 必要に応じて設定します。

Microsoft SQL Server データベース用の Sun GlassFish JDBC ドライバ

このドライバの JAR ファイルは `smsqlserver.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Microsoft SQL Server
- 「データソースクラス名」: `com.sun.sql.jdbcx.sqlserver.SQLServerDataSource`
- 「プロパティー」:
  - **`serverName`** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **`portNumber`** – データベースサーバーのポート番号を指定します。
  - **`user`** – 必要に応じて設定します。
  - **`password`** – 必要に応じて設定します。
  - **`selectMethod`** – `cursor` に設定します。

MySQL Server データベース用の Sun GlassFish JDBC ドライバ

Sun MySQL ドライバは、MySQL Enterprise のみで動作します。このドライバの JAR ファイルは `smmysql.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: SQL Server
- 「データソースクラス名」: `com.sun.sql.jdbcx.mysql.MySQLDataSource`

- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します。
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。
  - **selectMethod** – cursor に設定します。

Sybase データベース用の Sun GlassFish JDBC ドライバ

このドライバの JAR ファイルは smsybase.jar です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Sybase
- 「データソースクラス名」: com.sun.sql.jdbcx.sybase.SybaseDataSource
- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します。
  - **databaseName** – 必要に応じて設定します。これは任意指定です。
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。

IBM DB2 Type 2 ドライバ

この DB2 ドライバの JAR ファイルは db2jcc.jar、db2jcc_license_cu.jar、および db2java.zip です。環境変数を設定してください。次に例を示します。

```
LD_LIBRARY_PATH=/usr/db2user/sqllib/lib:${Java EE.home}/lib
DB2DIR=/opt/IBM/db2/V8.2
DB2INSTANCE=db2user
INSTHOME=/usr/db2user
VWSPATH=/usr/db2user/sqllib
THREADS_FLAG=native
```

次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: DB2
- 「データソースクラス名」: com.ibm.db2.jcc.DB2SimpleDataSource
- 「プロパティー」:

- **databaseName** - 必要に応じて設定します。
- **user** – 必要に応じて設定します。
- **password** – 必要に応じて設定します。
- **driverType** – 2 に設定します。
- **deferPrepares** – false に設定します。

Java DB/Derby Type 4 ドライバ

Java DB ドライバの JAR ファイルは derbyclient.jar です。Java DB は Apache Derby に基づいています。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: JavaDB
- 「データソースクラス名」: 次のいずれかを指定します。

```
org.apache.derby.jdbc.ClientDataSource
org.apache.derby.jdbc.ClientXADataSource
```

- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します (デフォルトと異なる場合)。
  - **databaseName** – データベースの名前を指定します。
  - **user** - データベースユーザーを指定します。

    これは、Java DB が認証を使用するように設定されている場合にのみ必要です。Java DB は、デフォルトで認証を使用しません。ユーザーを指定すると、その名前が、テーブルが属するスキーマの名前になります。
  - **password** – データベースパスワードを指定します。

    これは、Java DB が認証を使用するように設定されている場合にのみ必要です。

MySQL Type 4 JDBC ドライバ

MySQL™ ドライバの JAR ファイルは mysql-connector-java-5.1.7-bin.jar です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Microsoft SQL Server
- 「データソースクラス名」:

```
com.mysql.jdbc.jdbc2.optional.MysqlDataSource
com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
```

- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します。
  - **databaseName** – 必要に応じて設定します。
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。

PostgreSQL ドライバ

PostgreSQL ドライバの JAR ファイルは `postgresql-8.4-701.jdbc4.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: PostgreSQL Server
- 「データソースクラス名」:

```
* org.postgresql.ds.PGSimpleDataSource
```

- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します。
  - **databaseName** – 必要に応じて設定します。
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。

限定的にサポートされている JDBC ドライバ

次の JDBC ドライバも Enterprise Server で使用できますが、これらのドライバは完全にはテストされていません。Sun では、これらのドライバの製品サポートは提供していませんが、Enterprise Server &; での使用についての限定サポートを提供しています。

- 261 ページの「IBM Informix Type 4 ドライバ」
- 261 ページの「Oracle データベース用の Inet Oraxo JDBC ドライバ」
- 262 ページの「Microsoft SQL Server データベース用の Inet Merlia JDBC ドライバ」
- 262 ページの「Sybase データベース用の Inet Sybelux JDBC ドライバ」
- 263 ページの「Sybase ASE 12.5 データベース用の JConnect Type 4 ドライバ」
- 263 ページの「Oracle 11 データベース用の Oracle Thin Type 4 JDBC ドライバ」
- 264 ページの「Oracle データベース用の OCI Oracle Type 2 ドライバ」

注 – Oracle データベースユーザーが `capture-schema` コマンドを実行するには、そのユーザーがスキーマの所有者でないかぎり、ANALYZE ANY TABLE 特権が必要です。この特権は、データベース管理者がユーザーに付与します。`capture-schema` の詳細は、『Sun GlassFish Enterprise Server v3 Reference Manual』を参照してください。

IBM Informix Type 4 ドライバ

次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Informix
- 「データソースクラス名」: 次のいずれかを指定します。

```
com.informix.jdbcx.IfxDataSource
com.informix.jdbcx.IfxXADataSource
```

- 「プロパティー」:
  - **`serverName`** – Informix データベースサーバー名を指定します。
  - **`portNumber`** – データベースサーバーのポート番号を指定します。
  - **`databaseName`** – 必要に応じて設定します。これは任意指定です。
  - **`user`** – 必要に応じて設定します。
  - **`password`** – 必要に応じて設定します。
  - **`IfxIFXHost`** – データベースサーバーのホスト名または IP アドレスを指定します。

Oracle データベース用の Inet Oraxo JDBC ドライバ

この Oracle ドライバの JAR ファイルは `Oranxo.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Oracle
- 「データソースクラス名」: `com.inet.ora.OraDataSource`
- 「プロパティー」:
  - **`serverName`** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **`portNumber`** – データベースサーバーのポート番号を指定します。
  - **`user`** – データベースユーザーを指定します。
  - **`password`** – データベースパスワードを指定します。
  - **`serviceName`** – データベースの URL を指定します。構文は次のとおりです。

`jdbc:inetora:`*server*`:`*port*`:`*dbname*

次に例を示します。

```
jdbc:inetora:localhost:1521:payrolldb
```

この例では、`localhost` は Oracle サーバーを実行しているマシンのホスト名、`1521` は Oracle サーバーのポート番号、`payrolldb` はデータベースの SID を表します。データベース URL の構文の詳細については、Oracle のドキュメントを参照してください。

- **`streamstolob`** - BLOB または CLOB データ型のサイズが 4K バイトを超え、このドライバが CMP で使用される場合は、このプロパティーを `true` に設定してください。
- **`xa-driver-does-not-support-non-tx-operations`** - `true` の値に設定します。同じ接続プールから非 XA 接続と XA 接続の両方が取得される場合にのみ必要です。パフォーマンスが低下する可能性があります。

  このプロパティーを設定する代わりに、非 XA 接続用とXA 接続用の 2 つの接続プールを作成することもできます。

Microsoft SQL Server データベース用の Inet Merlia JDBC ドライバ

この Microsoft SQL Server ドライバの JAR ファイルは `Merlia.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Microsoft SQL Server
- 「データソースクラス名」: `com.inet.tds.TdsDataSource`
- 「プロパティー」:
  - **`serverName`** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **`portNumber`** – データベースサーバーのポート番号を指定します。
  - **`user`** – 必要に応じて設定します。
  - **`password`** – 必要に応じて設定します。

Sybase データベース用の Inet Sybelux JDBC ドライバ

この Inet Sybase ドライバの JAR ファイルは `Sybelux.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Sybase
- 「データソースクラス名」: `com.inet.syb.SybDataSource`

- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します。
  - **databaseName** – 必要に応じて設定します。完全な URL ではなく、データベース名のみを指定してください。
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。

Sybase ASE 12.5 データベース用の JConnect Type 4 ドライバ

この Sybase ドライバの JAR ファイルは `jconn4.jar` です。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Sybase
- 「データソースクラス名」: 次のいずれかを指定します。

```
com.sybase.jdbc4.jdbc.SybDataSource
com.sybase.jdbc4.jdbc.SybXADataSource
```

- 「プロパティー」:
  - **serverName** – データベースサーバーのホスト名または IP アドレスを指定します。
  - **portNumber** – データベースサーバーのポート番号を指定します。
  - **databaseName** – 必要に応じて設定します。完全な URL ではなく、データベース名のみを指定してください。
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。
  - **BE_AS_JDBC_COMPLIANT_AS_POSSIBLE** – `true` に設定します。
  - **FAKE_METADATA** – `true` に設定します。

Oracle 11 データベース用の Oracle Thin Type 4 JDBC ドライバ

この Oracle ドライバの JAR ファイルは `ojdbc6.jar` です。

注 – このドライバを使用する場合は、1 つの列に 2000 バイトを超えるデータを挿入できないことに注意してください。この問題を回避するには、OCI ドライバ (JDBC Type 2) を使用します。

次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Oracle
- 「データソースクラス名」: 次のいずれかを指定します。

```
oracle.jdbc.pool.OracleDataSource
oracle.jdbc.xa.client.OracleXADataSource
```

- 「プロパティー」:
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。
  - **xa-driver-does-not-support-non-tx-operations** - true の値に設定します。これは任意指定です。 同じ接続プールから非 XA 接続と XA 接続の両方が取得される場合にのみ必要です。パフォーマンスが低下する可能性があります。

    このプロパティーを設定する代わりに、非 XA 接続用と XA 接続用の 2 つの接続プールを作成することもできます。

注 – Oracle thin ドライバでは、`XAResource.recover` メソッドが、入力フラグに関係なく同じ未確定の Xid のセットを繰り返し返します。XA 仕様に従って、トランザクションマネージャーは最初に `TMSTARTSCAN` でこのメソッドを呼び出したあと、`TMNOFLAGS` で、Xid が返されなくなるまで繰り返しこのメソッドを呼び出します。`XAResource.commit` メソッドにもいくつかの問題があります。

この Enterprise Server の回避方法を無効にするには、`oracle-xa-recovery-workaround` プロパティーの値を `false` に設定する必要があります。

Oracle データベース用の OCI Oracle Type 2 ドライバ

この OCI Oracle ドライバの JAR ファイルは `ojdbc14.jar` です。`LD_LIBRARY_PATH` を介して共用ライブラリが使用可能であること、および `ORACLE_HOME` プロパティーが設定されていることを確認してください。次のように接続プールを設定します。

- 「名前」: あとで JDBC リソースを設定するときに、この名前を使用します。
- 「リソースタイプ」: 適切な値を指定します。
- 「データベースベンダー」: Oracle

- 「データソースクラス名」:次のいずれかを指定します。

  ```
  oracle.jdbc.pool.OracleDataSource
  oracle.jdbc.xa.client.OracleXADataSource
  ```

- 「プロパティー」:
  - **user** – 必要に応じて設定します。
  - **password** – 必要に応じて設定します。
  - **xa-driver-does-not-support-non-tx-operations** - `true` の値に設定します。同じ接続プールから非 XA 接続と XA 接続の両方が取得される場合にのみ必要です。パフォーマンスが低下する可能性があります。

    このプロパティーを設定する代わりに、非 XA 接続用とXA 接続用の 2 つの接続プールを作成することもできます。

◆ ◆ ◆ 第 15 章

EIS 接続の管理

この章では、Sun GlassFish™ Enterprise Server v3 環境で、`asadmin` コマンド行ユーティリティーを使用し、エンタープライズ情報システム (EIS) のデータとの接続を管理する方法について説明します。

注 – Web Profile をインストールした場合は、発信機能だけを使用するコネクタモジュールと、着信機能を伴わない作業管理がサポートされます。その他のコネクタ機能は、Full Platform Profile でのみサポートされます。

ここでは、以下のトピックに関して説明します。

本章で説明するタスクを管理コンソールから実行する手順については、管理コンソールのオンラインヘルプを参照してください。

データベース接続については、第 14 章「データベース接続の管理 」を参照してください。

EIS 接続とは

エンタープライズ情報システム (EIS) は、組織のデータを保持する任意のシステムです。メインフレーム、メッセージングシステム、データベースシステム、またはアプリケーションがこれに使用できます。アプリケーションとモジュールが EIS ソフトウェアにアクセスするには、接続リソースが使用されます。

EIS 接続の主な要素は、次のとおりです。

- 「コネクタモジュール」。コネクタモジュールは、アプリケーションと EIS ソフトウェアとの対話を可能にする Java EE コンポーネントで、リソースアダプタとも呼ばれます。コネクタモジュールは、Enterprise Server が Java™ Message Service (JMS) を実装する際に使用されます。ほかの Java EE モジュールと同様に、コネクタモジュールをインストールするには、これを配備する必要があります。コネクタモジュールの作成方法については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 12 章「Developing Connectors」を参照してください。
- 「コネクタ接続プール」。コネクタ接続プールとは、特定の EIS のための再利用可能な接続のグループです。コネクタ接続プールを作成するには、プールに関連付けるコネクタモジュールを指定します。管理手順については、269 ページの「コネクタ接続プールの管理」を参照してください。
- 「コネクタリソース」。コネクタリソースとは、アプリケーションに EIS への接続を提供するプログラムオブジェクトです。コネクタリソースを作成するには、JNDI 名と関連する接続プールを指定します。EIS 用コネクタリソースの JNDI 名は、通常 `java:comp/env/`*eis-specific* サブコンテキストにあります。管理手順については、273 ページの「コネクタリソースの管理」を参照してください。
- 「コネクタモジュール構成」。コネクタモジュール構成とは、ドメイン構成ファイル (`domain.xml`) 内にある、特別なコネクタモジュール (リソースアダプタ) の情報です。管理手順については、276 ページの「リソースアダプタ構成の管理」を参照してください。
- 「コネクタセキュリティーマップ」。コネクタセキュリティーマップとは、アプリケーションの (主体またはユーザーグループの) 呼び出し側 ID を、適切な EIS 主体またはグループに関連付けるものです。管理手順については、278 ページの「コネクタセキュリティーマップの管理」を参照してください。
- 「コネクタの作業セキュリティーマップ」。コネクタの作業セキュリティーマップは、コネクタモジュール (リソースアダプタ) の EIS 主体または EIS グループが提出した作業の呼び出し側 ID を、Enterprise Server セキュリティードメインの適切な主体またはユーザーグループに関連付けます。管理手順については、282 ページの「コネクタ作業セキュリティーマップの管理」を参照してください。

- 「管理対象オブジェクト」。管理対象オブジェクトは、アプリケーションの特殊な機能を提供します。たとえば、管理対象オブジェクトは、コネクタモジュールおよびそれに関連付けられた EIS に固有なパーサーへのアクセスを提供できます。管理手順については、285 ページの「管理対象オブジェクトの管理」を参照してください。

実行時に、アプリケーションが EIS に接続されると次のことが行われます。

1. JNDI API を介して呼び出しを行うことにより、アプリケーションは EIS に関連したコネクタリソース (データソース) を取得します。

   コネクタリソースの JNDI 名を基に、ネーミングおよびディレクトリサービスがリソースを検索します。EIS リソースはそれぞれコネクタ接続プールを指定します。
2. コネクタリソースを経由して、アプリケーションは EIS 接続を取得します。

   Enterprise Server は EIS リソースに対応した接続プールから物理接続を取得します。プールは、EIS 名、ユーザー名、パスワードなどの接続属性を定義します。
3. EIS 接続が確立されると、アプリケーションは EIS のデータの読み込み、変更、および追加を行うことができます。

   アプリケーションは JMS API を呼び出すことにより、EIS 情報にアクセスします。
4. EIS へのアクセスが完了すると、アプリケーションは接続を終了して、接続を接続プールに返します。

コネクタ接続プールの管理

コネクタモジュールを配備すると、これにコネクタ接続プールを作成できるようになります。

ここでは、以下のトピックに関して説明します。

- 269 ページの「コネクタ接続プールを作成する」
- 271 ページの「コネクタ接続プールを一覧表示する」
- 271 ページの「コネクタ接続プールに接続 (ping) するかコネクタ接続プールをリセット (フラッシュ) する」
- 271 ページの「コネクタ接続プールを更新する」
- 272 ページの「コネクタ接続プールを削除する」

▼ コネクタ接続プールを作成する

配備したコネクタモジュールにコネクタ接続プールを作成するには、リモートモードで `create-connector-connection-pool` サブコマンドを使用します。コネクタ接続プールを構築する際に、その EIS に固有の所定データを入力するよう求められます。必須の `--connectiondefintion` オプションの値が、EIS 情報となります。

複数のコネクタリソースで 1 つの接続プールを指定できます。

コネクタ接続プールの作成は動的イベントで、サーバーの再起動は求められません。ただし、パラメータの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。

始める前に　コネクタ接続プールを作成する前に、コネクタをインストールしておいてください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `create-connector-connection-pool(1)` サブコマンドを使用して、コネクタ接続プールを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。
3. (省略可能) 必要な場合は、サーバーを再起動します。
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。
4. (省略可能) 接続プールが使用可能であることを確認するには、`ping-connection-pool` サブコマンドを使用します。
   手順については、250 ページの「接続プールと通信する (ping を実行する)」を参照してください。

例 15–1 コネクタ接続プールの作成

この例では、`javax.jms.QueueConnectionFactory` コネクタモジュールに `jms/qConnPool` プールを新規作成します。

```
asadmin> create-connector-connection-pool --steadypoolsize 20 --maxpoolsize 100
--poolresize 2 --maxwait 60000 --raname jmsra --connectiondefinition
javax.jms.QueueConnectionFactory jms/qConnPool
Command create-connector-connection-pool executed successfully
```

参照　コマンド行に `asadmin help create-connector-connection-pool` と 入力して、サブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタ接続プールを一覧表示する

作成済みのプールを一覧表示するには、リモートモードで `list-connector-connection-pools` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-connector-connection-pools(1)` サブコマンドを使用して、コネクタ接続プールを一覧表示します。

例 **15–2** コネクタ接続プールの一覧表示

この例では、既存のコネクタ接続プールを一覧表示します。

```
asadmin> list-connector-connection-pools
jms/qConnPool
Command list-connector-connection-pools executed successfully
```

参照 コマンド行に `asadmin help list-connector-connection-pools` と 入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタ接続プールに接続 (ping) するかコネクタ接続プールをリセット (フラッシュ) する

リモートモードで接続プールにこれらのタスクを実行するには、`ping-connection-pool` または `flush-connection-pool` サブコマンドを使用します。手順については、250 ページの「接続プールと通信する (ping を実行する)」または251 ページの「接続プールをリセット (フラッシュ) する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `flush-connection-pool(1)` サブコマンド、または `ping-connection-pool(1)` サブコマンドを使用して、コネクタ接続プールに接続、またはコネクタ接続プールをリセットします。

▼ コネクタ接続プールを更新する

コネクタ接続プールのプロパティー値を表示および変更するには、 `get` および `set` サブコマンドを使用します。

1. list-connector-connection-pools(1) サブコマンドを使用して、コネクタ接続プールを一覧表示します。
2. get(1) サブコマンドを使用して、コネクタ接続プールのプロパティーを表示します。
   次に例を示します。

```
asadmin> get domain.resources.connector-connection-pool.conectionpoolname.*
```

3. set(1) サブコマンドを使用して、コネクタ接続プールのプロパティーを設定します。
   次に例を示します。

```
asadmin> set domain.resources.connector-connection-pool
.conectionpoolname.validate-atmost-once-period-in-seconds=3
```

4. (省略可能) 必要な場合は、サーバーを再起動します。
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

▼ コネクタ接続プールを削除する

コネクタ接続プールを削除するには、リモートモードで delete-connector-connection-pool サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-connector-connection-pools(1) サブコマンドを使用して、コネクタ接続プールを一覧表示します。
3. 必要な場合は、コネクタ接続プールが削除されることをユーザーに通知してください。
4. delete-connector-connection-pool(1) サブコマンドを使用して、コネクタ接続プールを削除します。

例 15–3 コネクタ接続プールの削除

この例では、jms/qConnPool という接続プールを削除します。

```
asadmin> delete-connector-connection-pool --cascade=false jms/qConnPool
Command delete-connector-connection-pool executed successfully
```

参照　コマンド行に `asadmin help delete-connector-connection-pool` と 入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

コネクタリソースの管理

コネクタリソースとは、アプリケーションまたはモジュールに EIS への接続手段を提供するものです。通常、ドメインに配備するアプリケーションのアクセス対象となる EIS ごとに、コネクタリソースを作成します。

ここでは、以下のトピックに関して説明します。

▼ コネクタリソースを作成する

新しいコネクタリソースとその JNDI 名を登録するには、リモートモードで `create-connector-resource` サブコマンドを使用します。

コネクタリソースの作成は動的イベントで、サーバーの再起動は求められません。ただし、パラメータの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。

始める前に　コネクタリソースを作成する前に、まずコネクタ接続プールを作成しておく必要があります。手順については、269 ページの「コネクタ接続プールを作成する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-connector-resource(1)` サブコマンドを使用して、コネクタリソースを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. (省略可能) 必要な場合は、サーバーを再起動します。
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 15–4 コネクタリソースの作成

この例では、jms/qConnPool 接続プールに jms/qConnFactory というリソースを新規作成します。

```
asadmin> create-connector-resource --poolname jms/qConnPool
--description "creating sample connector resource" jms/qConnFactory
Command create-connector-resource executed successfully
```

参照 コマンド行に asadmin help create-connector-resource と 入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタリソースを一覧表示する

作成済みのコネクタリソースを一覧表示するには、リモートモードで list-connector-resources サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-connector-resources(1) サブコマンドを使用して、コネクタ接続プールを一覧表示します。

例 15–5 コネクタリソースの一覧表示

この例では、既存のコネクタリソースを一覧表示します。

```
asadmin> list-connector-resources
jms/qConnFactory
Command list-connector-resources executed successfully
```

参照 コマンド行に asadmin help list-connector-resources と 入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタリソースを更新する

コネクタリソースのプロパティー値を表示および変更するには、 get および set サブコマンドを使用します。

1 list-connector-resources(1) サブコマンドを使用して、コネクタ接続プールを一覧表示します。

2 get(1) サブコマンドを使用して、コネクタリソースのプロパティーを表示します。
次に例を示します。

```
asadmin> get domain.resources.connector-resource.jms/qConnFactory
```

3 set(1) サブコマンドを使用して、コネクタリソースのプロパティーを設定します。
次に例を示します。

```
asadmin> set domain.resources.connector-resource.jms/qConnFactory.enabled=true
```

4 (省略可能) 必要な場合は、サーバーを再起動します。
プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

▼ コネクタリソースを削除する

JNDI 名を指定してコネクタリソースを削除するには、リモートモードで delete-connector-resource サブコマンドを使用します。

始める前に リソースを削除する前に、そのリソースに関連付けられているものをすべて削除しておく必要があります。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-connector-resources(1) サブコマンドを使用して、コネクタ接続プールを一覧表示します。

3 必要な場合は、コネクタリソースが削除されることをユーザーに通知してください。

4 delete-connector-resource(1) サブコマンドを使用して、コネクタリソースを削除します。

例 15–6 コネクタリソースの削除

この例では、jms/qConnFactory コネクタリソースを削除します。

```
asadmin> delete-connector-resource jms/qConnFactory
Command delete-connector-resources executed successfully
```

参照 コマンド行に asadmin help delete-connector-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

リソースアダプタ構成の管理

ここでは、以下のトピックに関して説明します。

▼ リソースアダプタの構成情報を作成する

リソースアダプタ (コネクタモジュール) の構成情報を作成するには、リモートモードで `create-resource-adapter-config` サブコマンドを使用します。配備するときに構成情報を使用できるように、このサブコマンドを実行してからリソースアダプタを配備することができます。このリソースアダプタ構成は、リソースアダプタを配備したあとでも作成できます。この場合、リソースアダプタは新しい構成で再起動されます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-resource-adapter-config(1)` サブコマンドで、構成情報を作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

例 **15–7** リソースアダプタ構成の作成

この例では、リソースアダプタ `ra1` の構成を作成します。

```
asadmin> create-resource-adapter-config --property foo=bar
--threadpoolid mycustomerthreadpool ra1
Command create-resource-adapter-config executed successfully
```

参照 コマンド行に `asadmin help create-resource-adapter-config` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ リソースアダプタ構成の一覧表示

指定したリソースアダプタ (コネクタモジュール) のドメイン構成ファイル (`domain.xml`) に含まれている構成情報を一覧表示するには、リモートモードで `list-resource-adapter-configs` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-resource-adapter-configs(1) サブコマンドで、リソースアダプタの設定を一覧表示します。

例 15–8 リソースアダプタ構成の一覧表示

この例では、リソースアダプタ構成を一覧表示します。

```
asadmin> list-resource-adapter-configs
ra1
ra2
Command list-resource-adapter-configs executed successfully
```

参照 コマンド行に asadmin help list-resource-adapter-configs と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ リソースアダプタ構成を更新する

リソースアダプタ構成のプロパティー値を表示および変更するには、 get および set サブコマンドを使用します。

1. list-resource-adapter-configs(1) サブコマンドで、リソースアダプタの構成を一覧表示します。

2. get(1) サブコマンドを使用して、コネクタリソースのプロパティーを表示します。
   次に例を示します。

```
asadmin> get domain.resources.resource-adapter-config.ra1.*
```

3. set(1) サブコマンドを使用して、コネクタリソースのプロパティーを設定します。
   次に例を示します。

```
asadmin> set domain.resources.resource-adapter-config.ra1.raSpecificProperty=value
```

▼ リソースアダプタ構成を削除する

指定したリソースアダプタ (コネクタモジュール) のドメイン構成ファイル (domain.xml) に含まれている構成情報を削除するには、リモートモードで delete-resource-adapter-config サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 list-resource-adapter-configs(1) サブコマンドで、リソースアダプタの構成を一覧表示します。

3 delete-resource-adapter-config(1) サブコマンドで、リソースアダプタの構成を削除します。

例 15–9 リソースアダプタ構成の削除

この例では、リソースアダプタ ra1 の構成を削除します。

```
asadmin> delete-resource-adapter-config ra1
Command delete-resource-adapter-config executed successfully
```

参照 コマンド行に asadmin help delete-resource-adapter-config と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

コネクタセキュリティーマップの管理

EIS は、組織のデータを保持する任意のシステムです。メインフレーム、メッセージングシステム、データベースシステム、またはアプリケーションがこれに使用できます。コネクタセキュリティーマップは、アプリケーションの証明を EIS 証明にマッピングするのに使用します。

セキュリティーマップは、個々のコネクタ接続プールに適用されます。1 つ以上の指定したセキュリティーマップをコネクタ接続プールに関連付けることができます。

ここでは、以下のトピックに関して説明します。

▼ コネクタセキュリティーマップを作成する

指定したコネクタ接続プールにセキュリティーマップを作成するには、リモートモードで create-connector-security-map サブコマンドを使用します。セキュリティーマップが存在しない場合は、新規に作成されます。バックエンド EIS 主体、またはバックエンド EIS ユーザーグループを指定できます。コネクタセキュリティーマップの構成では、ワイルドカード文字としてアスタリスク (*) を使用し、すべてのユーザーまたはすべてのユーザーグループを示すことができます。

このサブコマンドを使用して、コンテナ管理トランザクションベースのシナリオで、アプリケーション (主体またはユーザーグループ) の呼び出し側 ID を適切な EIS 主体に割り当てることもできます。

始める前に　このサブコマンドを正常に実行するためには、最初にコネクタ接続プールを作成しておく必要があります。手順については、269 ページの「コネクタ接続プールを作成する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. create-connector-security-map(1) サブコマンドで、コネクタセキュリティーマップを作成します。
   このサブコマンドのオプションについては、このマニュアルページに記載されています。

3. (省略可能) 必要な場合は、サーバーを再起動します。
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 15–10 コネクタセキュリティーマップの作成

この例では、connection-pool1 に、securityMap1 コネクタセキュリティーマップを作成します。

```
asadmin> create-connector-security-map --poolname connector-pool1
--principals principal1, principal2 --mappedusername backend-username securityMap1
Command create-connector-security-map executed successfully
```

▼ コネクタセキュリティーマップの一覧表示

指定したコネクタ接続プールに属する既存のセキュリティーマップを一覧表示するには、リモートモードで list-connector-security-maps サブコマンドを使用します。コネクタ接続プールにコネクタセキュリティーマップの簡易リストを取得することも、マップの主体を表示するような総合リストを取得することもできます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-connector-connection-pools(1) サブコマンドを使用して、既存のコネクタ接続プールを一覧表示します。

3. list-connector-security-maps(1) サブコマンドで、指定のコネクタ接続プールのセキュリティーマップを一覧表示します。

例 15–11 コネクタ接続プールのコネクタセキュリティーマップの一覧表示

この例では、connector-Pool1 に関連付けられているコネクタセキュリティーマップを一覧表示します。

```
asadmin> list-connector-security-maps connector-Pool1
securityMap1
Command list-connector-security-maps executed successfully.
```

例 15–12 コネクタ接続プールの指定セキュリティーマップの主体の一覧表示

この例では、securityMap1 に関連付けられている主体を一覧表示します。

```
asadmin> list-connector-security-maps --securitymap securityMap1 connector-Pool1
principal1
principal1
Command list-connector-security-maps executed successfully.
```

例 15–13 コネクタ接続プールのコネクタセキュリティーマップの主体の一覧表示

この例では、connector-Pool1 に関連付けられているコネクタセキュリティーマップを一覧表示します。

```
asadmin> list-connector-security-maps --verbose connector-Pool1
securityMap1
principal1
principal1
Command list-connector-security-maps executed successfully.
```

▼ コネクタセキュリティーマップを更新する

指定したコネクタ接続プールにセキュリティーマップを作成または変更するには、リモートモードで update-connector-security-map サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-connector-security-maps(1) サブコマンドを使用して、既存のコネクタセキュリティーマップを一覧表示します。
3. update-connector-security-map(1) サブコマンドで、指定のコネクタ接続プールのセキュリティーマップを変更します。

4 (省略可能) 必要な場合は、サーバーを再起動します。

プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 15–14 コネクタセキュリティーマップの更新

この例では、securityMap1 に主体を追加します。

```
asadmin> update-connector-security-map --poolname connector-pool1
--addprincipals principal1, principal2 securityMap1
Command update-connector-security-map executed successfully.
```

▼ コネクタセキュリティーマップを削除する

指定したコネクタ接続プールにセキュリティーマップを削除するには、リモートモードで delete-connector-security-map サブコマンドを使用します。

1 サーバーが実行されていることを確認します。

リモートサブコマンドには、実行中のサーバーが必要です。

2 list-connector-connection-pools(1) サブコマンドを使用して、既存のコネクタ接続プールを一覧表示します。

3 delete-connector-security-map(1) サブコマンドで、指定のコネクタ接続プールのセキュリティーマップを削除します。

このサブコマンドのオプションについては、このマニュアルページに記載されています。

例 15–15 コネクタセキュリティーマップの削除

この例では、connector-pool1 から securityMap1 を削除します。

```
asadmin> delete-connector-security-map --poolname connector-pool1 securityMap1
Command delete-connector-security-map executed successfully
```

コネクタ作業セキュリティーマップの管理

EIS は、組織のデータを保持する任意のシステムです。メインフレーム、メッセージングシステム、データベースシステム、またはアプリケーションがこれに使用できます。コネクタ作業セキュリティーマップは、EIS 証明を Enterprise Server セキュリティードメインの証明にマッピングするのに使用します。

セキュリティーマップは、個々のコネクタ接続プールに適用されます。1 つ以上の指定したセキュリティーマップをコネクタ接続プールに関連付けることができます。

ここでは、以下のトピックに関して説明します。

▼ コネクタ作業セキュリティーマップを作成する

コネクタモジュール (リソースアダプタ) の EIS 主体または EIS グループが提出した作業の呼び出し側 ID を、Enterprise Server セキュリティードメインの適切な主体またはユーザーグループにマッピングするには、リモートモードで `create-connector-work-security-map` サブコマンドを使用します。1 つ以上の作業セキュリティーマップをコネクタモジュールに関連付けることができます。

コネクタセキュリティーマップの構成では、ワイルドカード文字としてアスタリスク (*) を使用し、すべてのユーザーまたはすべてのユーザーグループを示すことができます。

始める前に　コネクタリソースを作成する前に、まずコネクタ作業セキュリティーマップを作成しておく必要があります。手順については、269 ページの「コネクタ接続プールを作成する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `create-connector-work-security-map(1)` サブコマンドで、コネクタ作業セキュリティーマップを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3 (省略可能) 必要な場合は、サーバーを再起動します。
プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 15–16 コネクタ作業セキュリティーマップを作成する

次の例は、my-resource-adapter-name に workSecurityMap1 および workSecurityMap2 を作成します。

```
asadmin> create-connector-work-security-map --raname my-resource-adapter-name
--principalsmap eis-principal-1=server-principal-1,eis-principal-2=server-principal-2,
eis-principal-3=server-principal-1 workSecurityMap1

asadmin> create-connector-work-security-map --raname my-resource-adapter-name
--groupsmap eis-group-1=server-group-1,eis-group-2=server-group-2,
eis-group-3=server-group-1 workSecurityMap2
Command create-connector-work-security-map executed successfully
```

参照 コマンド行に asadmin help create-connector-work-security-map と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタ作業セキュリティーマップを一覧表示する

指定のコネクタモジュールに属する作業セキュリティーマップを一覧表示するには、リモートモードで list-connector-work-security-maps サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-connector-work-security-maps(1) サブコマンドで、コネクタ作業セキュリティーマップを一覧表示します。

例 15–17 コネクタ作業セキュリティーマップの一覧表示

この例では、一般的な作業セキュリティーマップを一覧表示します。

```
asadmin> list-connector-work-security-maps generic-ra
generic-ra-groups-map: EIS group=eis-group, mapped group=glassfish-group
generic-ra-principals-map: EIS principal=eis-bar, mapped principal=bar
```

```
generic-ra-principals-map: EIS principal=eis-foo, mapped principal=foo
Command list-connector-work-security-maps executed successfully.
```

参照 コマンド行に asadmin help list-connector-work-security-maps と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタ作業セキュリティーマップを更新する

指定のリソースアダプタ (接続モジュール) に属する作業セキュリティーマップを変更するには、リモートモードで update-connector–work-security-map サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-connector-work-security-maps(1) サブコマンドで、コネクタ作業セキュリティーマップを一覧表示します。
3. 必要な場合は、コネクタ作業セキュリティーマップが変更されることをユーザーに通知してください。
4. update-connector-work-security-map(1) サブコマンドで、コネクタ作業セキュリティーマップを更新します。

例 15–18 コネクタ作業セキュリティーマップの更新

この例では、作業セキュリティーマップから主体を削除します。

```
asadmin> update-connector-work-security-map --raname generic-ra
--removeprincipals eis-foo generic-ra-principals-map
Command update-connector-work-security-map executed successfully.
```

参照 コマンド行に asadmin help update-connector-work-security-map と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ コネクタ作業セキュリティーマップを削除する

指定のコネクタモジュール (リソースアダプタ) に属する作業セキュリティーマップを削除するには、リモートモードで delete-connector–work-security-map サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 list-connector-work-security-maps(1) サブコマンドで、コネクタ作業セキュリティーマップを一覧表示します。

3 delete-connector-work-security-map(1) サブコマンドで、コネクタ作業セキュリティーマップを削除します。

例 15–19 コネクタ作業セキュリティーマップの削除

この例では、my_ra コネクタモジュールから worksecuritymap1 マップを削除します。

```
asadmin> delete-connector-work-security-map --raname my_ra worksecuritymap1
Command delete-connector-work-security-map executed successfully.
```

参照 コマンド行に asadmin help delete-connector-work-security-map と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

管理対象オブジェクトの管理

コネクタモジュールにパッケージ化されている管理対象オブジェクトは、アプリケーションの特殊な機能を提供します。たとえば、管理対象オブジェクトは、コネクタモジュールおよびそれに関連付けられた EIS に固有なパーサーへのアクセスを提供できます。

ここでは、以下のトピックに関して説明します。

▼ 管理対象オブジェクトリソースを作成する

管理対象オブジェクトを作成するには、create-admin-object サブコマンドを使用します。管理対象オブジェクトリソースを作成すると、名前と値のペアが作成され、そのオブジェクトが JNDI 名と関連付けられます。

始める前に このサブコマンド (jmsrar.rar) を実行する前に、リソースアダプタを配備しておいてください。

1 create-admin-object(1) サブコマンドで、管理対象オブジェクトを作成します。
このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

2 (省略可能) 必要な場合は、サーバーを再起動します。

プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 15–20 管理対象オブジェクトの作成

この例では、ra.xml ファイルから javax.jms.Queue リソースの型が取得されます。新たに作成される管理対象オブジェクトの JNDI 名は、jms/samplequeue です。

```
asadmin> create-admin-object --restype javax.jms.Queue
--raname jmsra  --description "sample administered object"
--property Name=sample_jmsqueue jms/samplequeue
Command create-admin-object executed successfully
```

参照 コマンド行に asadmin help create-admin-object と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 管理対象オブジェクトを一覧表示する

既存の管理対象オブジェクトを一覧表示するには、リモートモードで list-admin-object サブコマンドを使用します。

1 サーバーが実行されていることを確認します。

リモートサブコマンドには、実行中のサーバーが必要です。

2 list-admin-objects(1) サブコマンドで、管理対象オブジェクトを一覧表示します。

例 15–21 管理対象オブジェクトの一覧表示

この例では、既存の管理対象オブジェクトを一覧表示します。

```
asadmin> list-admin-objects
jms/samplequeue
Command list-admin-objects executed successfully
```

参照 コマンド行に asadmin help list-admin-object と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 管理対象オブジェクトを更新する

管理対象オブジェクトのプロパティー値を表示および変更するには、 `get` および `set` サブコマンドを使用します。

1. `list-admin-objects(1)` サブコマンドで、管理対象オブジェクトを一覧表示します。
2. `get(1)` サブコマンドを使用して、管理対象オブジェクトのプロパティーを表示します。

   次に例を示します。

   ```
   asadmin> get domain.resources.admin-object-resource.jms/samplequeue.*
   ```
3. `set(1)` サブコマンドを使用して、管理対象オブジェクトのプロパティーを設定します。

   次に例を示します。

   ```
   asadmin> set domain.resources.admin-object-resource.jms/samplequeue.enabled=false
   ```
4. (省略可能) 必要な場合は、サーバーを再起動します。

   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

▼ 管理対象オブジェクトリソースを削除する

管理対象オブジェクトを削除するには、`delete-admin-object` サブコマンドを使用します。

1. `list-admin-objects(1)` サブコマンドで、管理対象オブジェクトを一覧表示します。
2. 必要な場合は、管理対象オブジェクトが削除されることをユーザーに通知してください。
3. `delete-admin-object(1)` サブコマンドで、管理対象オブジェクトを削除します。

例 15–22 管理対象オブジェクトの削除

この例では、JNDI 名 `jms/samplequeue` が付いた管理対象オブジェクトを削除します。

```
asadmin> delete-admin-object jms/samplequeue
Command delete-admin-object executed successfully
```

参照　コマンド行に `asadmin help delete-admin-object` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 16 章

16 インターネット接続の管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server v3 の環境でインターネット接続のタスクを実行する手順について説明します。

ここでは、次のテーマを取り上げます。

- 289 ページの「インターネット接続について」
- 291 ページの「HTTP ネットワークリスナーの管理」
- 302 ページの「仮想サーバーの管理」

本章で説明するタスクを 管理コンソール から実行する手順については、管理コンソール オンラインヘルプを参照してください。

インターネット接続について

HTTP サービスは、Web アプリケーションを配備する機能と、配備した Web アプリケーションをインターネットクライアントからアクセス可能にする機能を提供します。HTTP サービスは、リスナーと仮想サーバーという 2 種類の関連オブジェクトにより提供されます。

ここでは、次のテーマを取り上げます。

- 289 ページの「HTTP ネットワークリスナーについて」
- 290 ページの「仮想サーバーについて」

HTTP ネットワークリスナーについて

「*HTTP* リスナー」はインターネットプロトコル (IP) アドレス、ポート番号、サーバー名、およびデフォルトの仮想サーバーを持つリスナーソケットで、「ネットワークリスナー」とも呼ばれます。各仮想サーバーは、1 つまたは複数

のリスナーを通じてサーバーとクライアントの間の接続を提供します。各リスナーは、ポート番号と IP アドレスの一意の組み合わせを持つ必要があります。たとえば、IP アドレス 0.0.0.0 を指定すると、HTTP リスナーは設定されたすべての IP アドレスのホストを特定のポートで待機できます。また、同一ポートを使用して、各リスナーに一意の IP アドレスを指定することもできます。

HTTP リスナーは IP アドレスとポート番号の組み合わせなので、同じ IP アドレスと異なるポート番号の組み合わせ、または異なる IP アドレスと同じポート番号の組み合わせ (これらのアドレスに対応するようにホストが構成されている場合) で、複数の HTTP リスナーを持つことができます。ただし、HTTP リスナーに単一のポート上ですべての IP アドレスを待機する 0.0.0.0 を使用する場合は、この同じポート上に、特定の IP アドレスを待機する HTTP リスナーを作成できません。たとえば、HTTP リスナーが 0.0.0.0: 8080 (ポート 8080 のすべての IP アドレス) を使用する場合、別の HTTP リスナーが 1.2.3.4: 8080 を使用することはできません。Enterprise Server が稼働中のホストは通常、1 つの IP アドレスにのみアクセスします。HTTP リスナーは通常、IP アドレス 0.0.0.0 と複数のポート番号を使用し、各ポート番号が異なる役割に使用されます。ただし、システムが複数の IP アドレスにアクセスできる場合は、各アドレスを異なる役割に使用できます。

Enterprise Server に配備された Web アプリケーションにアクセスするには、Web アプリケーション用に指定したコンテキストルートとともに、`http://localhost:8080/` (セキュリティー保護されたアプリケーションでは `https://localhost:8081/`) という URL を使用します。

管理コンソール にアクセスするには、`https://localhost:4848/` か `http://localhost:4848/asadmin/` (コンソールのデフォルトコンテキストルート) の URL を使用します。

仮想サーバーについて

「仮想サーバー」は、複数のインターネットドメイン名を同一の物理サーバーでホストするためのオブジェクトで、仮想ホストとも呼ばれます。同一物理サーバーでホストされるすべての仮想サーバーが、その物理サーバーの IP アドレスを共有します。仮想サーバーは、サーバーのドメイン名 (`www.aaa.com` など) と、Enterprise Server が稼動するサーバーを関連付けます。各仮想サーバーは、ネットワークの DNS サーバーに登録する必要があります。

注 – インターネットドメインと Enterprise Server の管理ドメインを混同しないでください。

たとえば、物理サーバーの `www.aaa.com`、`www.bbb.com`、および `www.ccc.com` のドメインをホストする場合を考えます。これらのドメインがそれぞれ、Web モジュール `web1`、`web2`、および `web3` に関連付けられていると仮定します。つまり、その物理サーバーでは、次の URL が処理されます。

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

最初の URL は仮想サーバー `www.aaa.com`、2 番目の URL は仮想サーバー `www.bbb.com`、3 番目の URL は仮想サーバー `www.ccc.com` にそれぞれマッピングされます。このマッピングが機能するためには、`www.aaa.com`、`www.bbb.com`、および `www.ccc.com` がすべて、物理サーバーの IP アドレスに解決され、仮想サーバーがネットワークの DNS サーバーに登録される必要があります。さらに、UNIX システムでは、これらのドメインを `/etc/hosts` ファイルに追加します (`/etc/nsswitch.conf` ファイルの `hosts` の設定に `files` が含まれる場合)。

HTTP ネットワークリスナーの管理

デフォルトでは、Enterprise Server の起動時に次の HTTP リスナーが自動的に開始されます。

- `server` の名前を持つ仮想サーバーに関連付けられた HTTP リスナー
  - リスナー `http-listener-1` では、セキュリティーが有効になっていません。
  - リスナー `http-listener-2` では、セキュリティーが有効になっています。
- 仮想サーバー `__asadmin` に関連付けられた HTTP リスナー `admin-listener`。このリスナーでは、セキュリティーが有効になっていません。

次の表に、ポートを使用するリスナーについて、Enterprise Server のデフォルトポートを示します。

表 16–1 リスナーのデフォルトポート

リスナー	デフォルトポート	説明
管理サーバー	4848	ドメイン管理サーバーには、管理コンソールと `asadmin` ユーティリティーを使ってアクセスします。管理コンソールには、ブラウザの URL にポート番号を指定します。リモートから `asadmin` サブコマンドを実行する場合は、オプション `--port` を使用してポート番号を指定します。
HTTP	8080	Web サーバーはポート上で HTTP 要求を待機します。配備した Web アプリケーションとサービスにアクセスするために、クライアントはこのポートに接続します。
HTTPS	8181	セキュリティー保護された通信用に設定された Web アプリケーションは、個別のポートで待機します。

ここでは、次のテーマを取り上げます。

- 292 ページの「インターネット接続を作成する」

▼ インターネット接続を作成する

リスナーのオプションをすべて指定してインターネット接続を作成するには、この手順でサブコマンドを使用します。ネットワークリスナーは、バックグラウンドで作成されます。このプロセスの簡略バージョンについては、298 ページの「HTTP ネットワークリスナーを作成する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-protocol(1)` サブコマンドにオプション `--securityenabled` を指定して、**HTTP** または **HTTPS** プロトコルを作成します。
   組み込みの `http-listener-1` HTTP プロトコルまたは `http-listener-2` HTTPS プロトコルを使用する場合は、この手順を省略します。

3. `create-http(1)` サブコマンドを使用して、**HTTP** 構成を作成します。
   組み込みプロトコルを使用する場合は、この手順を省略します。

4. `create-transport(1)` サブコマンドを使用して、トランスポートを作成します。
   組み込みの `tcp` トランスポートを使用する場合は、この手順を省略します。

5. (省略可能) `create-threadpool(1)` サブコマンドを使用して、スレッドプールを作成します。
   スレッドプールを使用しない場合、または組み込みの `http-thread-pool` スレッドプールを使用する場合は、この手順を省略します。

   スレッドプールの詳細については、第 5 章「スレッドプールの管理」を参照してください。

6. `create-network-listener(1)` サブコマンドを使用して、**HTTP** リスナーを作成します。
   プロトコルとトランスポートを指定します。スレッドプールは省略可能です。

7. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

参照 コマンド行に `asadmin help create-http-listener` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

HTTP プロトコルの管理

各 HTTP リスナーは HTTP プロトコルを持ち、この HTTP プロトコルは `create-protocol` サブコマンドを使用するか、298 ページの「HTTP ネットワークリスナーを作成する」の手順に従った場合に適用される組み込みプロトコルを使用して作成されます。

ここでは、次のテーマを取り上げます。

▼ プロトコルを作成する

プロトコルを作成するには、リモートモードで `create-protocol` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-protocol(1)` を使用して、プロトコルを作成します。
   サブコマンドのオプションとプロパティーについては、このマニュアルページに記載されています。

例 16–1 HTTP プロトコルの作成

この例は、セキュリティーを有効にして、プロトコル `http-1` を作成します。

```
asadmin> create-protocol --securityenabled=true http-1
Command create-protocol executed successfully.
```

参照 コマンド行に `asadmin help create-protocol` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ プロトコルを一覧表示する

既存の HTTP プロトコルを一覧表示するには、リモートモードで `list-protocols` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `list-protocols(1)` サブコマンドを使用して、既存のプロトコルを一覧表示します。

例 16–2 プロトコルの一覧表示

この例は、既存のプロトコルを一覧表示します。

```
asadmin> list-protocols
admin-listener
http-1
http-listener-1
http-listener-2
Command list-protocols executed successfully.
```

参照 コマンド行に asadmin help list-protocols と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ プロトコルを削除する

プロトコルを削除するには、リモートモードで delete-protocol サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. delete-protocol(1) サブコマンドを使用して、プロトコルを削除します。

例 16–3 プロトコルの削除

この例は、プロトコル http-1 を削除します。

```
asadmin> delete-protocol http-1
Command delete-protocol executed successfully.
```

参照 コマンド行に asadmin help delete-protocol と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

HTTP 構成の管理

各 HTTP リスナーは HTTP 構成を持ち、この HTTP 構成は、create-http サブコマンドを使用するか、298 ページの「HTTP ネットワークリスナーを作成する」の手順に従う場合に適用される組み込み構成を使用して作成されます。

ここでは、次のテーマを取り上げます。

- 295 ページの「HTTP 構成を作成する」
- 295 ページの「HTTP 構成を削除する」

▼ HTTP 構成を作成する

プロトコルの HTTP パラメータセットを作成するには、リモートモードで create-http サブコマンドを使用します。このパラメータセットは、ネットワークリスナーを 1 つ以上構成します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. create-http(1) サブコマンドを使用して、HTTP 構成を作成します。
   サブコマンドのオプションとプロパティーについては、このマニュアルページに記載されています。

例 16–4 HTTP 構成の作成

この例は、http-1 というプロトコルの HTTP パラメータセットを作成します。

```
asadmin> create-http --timeout-seconds 60 --default-virtual-server server http-1
Command create-http executed successfully.
```

参照 コマンド行に asadmin help create-http と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ HTTP 構成を削除する

プロトコルから HTTP のパラメータセットを削除するには、リモートモードで delete-http サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. delete-http(1) サブコマンドを使用して、プロトコルから HTTP のパラメータを削除します。

例 16–5 HTTP 構成の削除

この例は、http-1 というプロトコルから HTTP のパラメータセットを削除します。

```
asadmin> delete-http http-1
Command delete-http executed successfully.
```

参照 コマンド行に asadmin help delete-http と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

HTTP トランスポートの管理

各 HTTP リスナーは HTTP トランスポートを持ち、この HTTP トランスポートは、`create-transport` サブコマンドを使用するか、298 ページの「HTTP ネットワークリスナーを作成する」の手順に従う場合に適用される組み込みトランスポートを使用して作成されます。

ここでは、次のテーマを取り上げます。

- 296 ページの「トランスポートを作成する」
- 296 ページの「トランスポートを一覧表示する」
- 297 ページの「トランスポートを削除する」

▼ トランスポートを作成する

ネットワークリスナーのトランスポートを作成するには、リモートモードで `create-transport` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-transport(1)` サブコマンドを使用して、トランスポートを作成します。
   サブコマンドのオプションとプロパティーについては、このマニュアルページに記載されています。

例 16–6 トランスポートの作成

この例は、アクセプタスレッド数としてデフォルト以外の値を使用する、`http1-trans` というトランスポートを作成します。

```
asadmin> create-transport --acceptorthreads 100 http1-trans
Command create-transport executed successfully.
```

参照 コマンド行に `asadmin help create-transport` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ トランスポートを一覧表示する

既存の HTTP トランスポートを一覧表示するには、リモートモードで `list-transports` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 list-transports(1) サブコマンドを使用して、既存のトランスポートを一覧表示します。

例 16–7 HTTP トランスポートの一覧表示

この例は、既存のトランスポートを一覧表示します。

```
asadmin> list-transports
http1-trans
tcp
Command list-transports executed successfully.
```

参照 コマンド行に asadmin help list-transports と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ トランスポートを削除する

トランスポートを削除するには、リモートモードで delete-transport サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 delete-transport(1) サブコマンドを使用して、トランスポートを削除します。

例 16–8 トランスポートの削除

この例は、http1-trans というトランスポートを削除します。

```
asadmin> delete-transport http1-trans
Command delete-transport executed successfully.
```

参照 コマンド行に asadmin help delete-transport と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

HTTP ネットワークリスナーの管理

ここでは、次のテーマを取り上げます。

- 300 ページの「SSL の HTTP リスナーを構成する」
- 301 ページの「HTTP リスナーから SSL を削除する」
- 301 ページの「HTTP リスナーにデフォルト仮想サーバーを割り当てる」

▼ HTTP ネットワークリスナーを作成する

リスナーを作成するには、リモートモードで `create-http-listener` サブコマンドまたは `create-network-listener` サブコマンドを使用します。これらのサブコマンドは後方互換性を提供し、さらに HTTP プロトコルを使用するネットワークリスナーを作成するためのショートカットも提供します。ネットワークリスナー、関連プロトコル、トランスポート、および HTTP 構成がバックグラウンドで作成されます。このメソッドは便利なショートカットですが、一部のオプションのみにアクセスできます。リスナーのオプションをすべて指定する場合は、292 ページの「インターネット接続を作成する」の手順に従ってください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-network-listener(1)` サブコマンドまたは `create-http-listener(1)` サブコマンドを使用して、**HTTP** ネットワークリスナーを作成します。

3. 必要な場合は、サーバーを再起動します。
   `admin-listener` という名前の特別な HTTP ネットワークリスナーを編集する場合は、変更を有効にするためにサーバーを再起動する必要があります。94 ページの「ドメインの再起動」を参照してください。

例 16–9 HTTP リスナーの作成

この例は、アクセプタスレッド数としてデフォルト以外の値を使用する、`sampleListener` という HTTP リスナーを作成します。実行時にセキュリティーは有効になりません。

```
asadmin> create-http-listener --listeneraddress 0.0.0.0
--listenerport 7272 --defaultvs server --servername host1.sun.com
--acceptorthreads 100 --securityenabled=false
--enabled=false sampleListener
Command create-http-listener executed successfully.
```

例 16–10 ネットワークリスナーの作成

この例は、実行時に有効にならない `sampleListener` というネットワークリスナーを作成します。

```
asadmin> create-network-listener --listenerport 7272 protocol http-1
--enabled=false sampleListenerCommand create-network-listener executed successfully.
```

参照　コマンド行に asadmin help create-http-listener または asadmin help create-network-listener と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ HTTP ネットワークリスナーを一覧表示する

既存の HTTP リスナーを一覧表示するには、リモートモードで list-http-listeners サブコマンドまたは list-network-listeners サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-http-listeners(1) サブコマンドまたは list-network-listeners(1) サブコマンドを使用して、HTTP リスナーを一覧表示します。

例 16–11　HTTP リスナーの一覧表示

この例は、HTTP リスナーを一覧表示します。list-network-listeners サブコマンドを使用しても、同じ結果が得られます。

```
asadmin> list-http-listeners
admin-listener
http-listener-2
http-listener-1
Command list-http-listeners executed successfully.
```

参照　コマンド行に asadmin help list-http-listeners または asadmin help list-network-listeners と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ HTTP ネットワークリスナーを更新する

1. list-http-listeners(1) サブコマンドまたは list-network-listeners(1) サブコマンドを使用して、HTTP リスナーを一覧表示します。
2. set(1) サブコマンドを使用して、指定リスナーの値を変更します。
   リスナーは、そのドット表記名で指定します。

例 16–12　HTTP ネットワークリスナーの更新

この例は、security-enabled を false に変更します。

```
asadmin> set "server.network-config.protocols.protocol.
http-listener-2.security-enabled=false"server.network-config.
protocols.protocol.http-listener-2.security-enabled=falseCommand set executed successfully.
```

▼ HTTP ネットワークリスナーを削除する

既存の HTTP リスナーを削除するには、リモートモードで `delete-http-listener` サブコマンドまたは `delete-network-listener` サブコマンドを使用します。これにより、リスナーの通信のセキュリティーが無効になります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-http-listeners(1)` サブコマンドを使用して、**HTTP** リスナーを一覧表示します。
3. `delete-http-listener(1)` サブコマンドまたは `delete-network-listener(1)` サブコマンドを使用して、**HTTP** リスナーを削除します。
4. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 16–13 HTTP リスナーの削除

この例は、`sampleListener` という HTTP リスナーを削除します。

```
asamin> delete-http-listener sampleListener
Command delete-http-listener executed successfully.
```

参照 コマンド行に `asadmin help delete-http-listener` または `asadmin help delete-network-listener` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ SSL の HTTP リスナーを構成する

指定リスナー内に SSL 要素を作成して構成するには、`create-ssl` サブコマンドを使用します。これにより、リスナーの通信のセキュリティーが有効になります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `create-ssl(1)` サブコマンドを使用して、**HTTP** リスナーを構成します。
3. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 16–14 SSL の HTTP リスナーの構成

この例は、`http-listener-1` という SSL の HTTP リスナーを有効にします。

```
asadmin> create-ssl --type http-listener --certname sampleCert http-listener-1
Command create-ssl executed successfully.
```

参照　コマンド行に asadmin help create-ssl と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ HTTP リスナーから SSL を削除する

指定リスナーの SSL 要素を削除するには、リモートモードで delete-ssl サブコマンドを使用します。これにより、リスナーの通信のセキュリティーが無効になります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. delete-ssl(1) サブコマンドを使用して、HTTP リスナーから SSL を削除します。
3. 変更内容を適用するために、Enterprise Server を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 16–15　HTTP リスナーからの SSL の削除

この例は、http-listener-1 という HTTP リスナーの SSL を無効にします。

```
asadmin> delete-ssl --type http-listener http-listener-1
Command delete-http-listener executed successfully.
```

参照　コマンド行に asadmin help delete-ssl と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ HTTP リスナーにデフォルト仮想サーバーを割り当てる

未定

1. 管理コンソールで、関連する構成の下にある HTTP サービスコンポーネントを開きます。
2. HTTP サービスコンポーネントの下にある HTTP リスナーコンポーネントを開きます。
3. HTTP リスナーを選択するか、新規に作成します。
4. 「デフォルト仮想サーバー」ドロップダウンリストから選択します。
   詳細については、305 ページの「デフォルトの Web モジュールを仮想サーバーに割り当てる」を参照してください。

参照　詳細については、管理コンソールの「HTTP リスナー」ページの「ヘルプ」ボタンをクリックしてください。

仮想サーバーの管理

仮想サーバーは、特定の URL を対象にコンテンツを提供する仮想 Web サーバーです。複数の仮想サーバーが、同一または異なるホスト名、ポート番号、または IP アドレスを使用して、コンテンツを提供できます。HTTP サービスは、受信した Web 要求を URL に基づいて異なる仮想サーバーに送信します。

Enterprise Server を初めてインストールするときに、デフォルト仮想サーバーが作成されます。新規作成する個々の HTTP サーバーに、デフォルト仮想サーバーを割り当てることができます。

Web コンポーネント (Web モジュール) を含む Web アプリケーションと Java EE アプリケーションを、配備時に仮想サーバーに割り当てることができます。Web モジュールは複数のサーバーに割り当てることができ、仮想サーバーには複数の Web モジュールを割り当てることができます。Web アプリケーションを配備するときに仮想サーバーを指定していない場合、その Web アプリケーションが現在定義されている仮想サーバーのすべてに割り当てられます。その後、追加の仮想サーバーを作成して既存の Web アプリケーションを割り当てる場合は、Web アプリケーションを再配備する必要があります。配備の詳細については、『Sun GlassFish Enterprise Server v3 Application Deployment Guide』を参照してください。

仮想サーバーのプロパティーを定義するには、`asadmin set` コマンドを使用します。次に例を示します。

```
asadmin set server-config.http-service.virtual-server.MyVS.property.sso-enabled="true"
```

指定の Web アプリケーションについて、一部の仮想サーバーのプロパティーを設定できます。詳細については、『Sun GlassFish Enterprise Server v3 Application Deployment Guide』の「sun-web-app」を参照してください。

ここでは、次のテーマを取り上げます。

▼ 仮想サーバーを作成する

デフォルトでは、Enterprise Server の起動時に次の仮想サーバーが自動的に開始されます。

- `server`: ユーザー定義のすべての Web モジュールをホスティングする仮想サーバー。

  本稼動環境以外での Web サービスの開発、テスト、配備で必要となる仮想サーバーは、通常、`server` だけです。
- `__asadmin`: すべての管理関連 Web モジュール (具体的には &AdminCon ole;) をホスティングする仮想サーバー。このサーバーの使用は制限されています。つまり、この仮想サーバーに Web モジュールを配備することはできません。

ただし本稼動環境では、同一物理サーバー上でユーザーと顧客のそれぞれが専用の Web サーバーを持つように見せる機能をホスティングするため、通常は追加の仮想サーバーも使用されます。

名前付きの仮想サーバーを作成するには、リモートモードで `create-virtual-server` サブコマンドを使用します。

始める前に　仮想サーバーは、既存の HTTP リスナーを指定する必要があります。仮想サーバーは、すでに別の仮想サーバーが使用している HTTP リスナーを指定できないので、仮想サーバーを新規作成する前に、HTTP リスナーを 1 つ以上作成します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `create-virtual-server(1)` サブコマンドを使用して、仮想サーバーを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。
3. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 16–16　仮想サーバーの作成

この例は、 `localhost` に `sampleServer` という仮想サーバーを作成します。

```
asadmin> create-virtual-server sampleServer
Command create-virtual-server executed successfully.
```

参照　コマンド行に `asadmin help create-virutal-server` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 仮想サーバーを一覧表示する

既存の仮想サーバーを一覧表示するには、リモートモードで `list-virtual-servers` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-virtual-servers(1)` サブコマンドを使用して、仮想サーバーを一覧表示します。

例 **16–17** 仮想サーバーの一覧表示

この例は、`localhost` の仮想サーバーを一覧表示します。

```
asadmin> list-virtual-servers
sampleListener
admin-listener
http-listener-2
http-listener-1

Command list-http-listeners executed successfully.
```

参照 コマンド行に `asadmin help list-virutal-servers` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 仮想サーバーを更新する

1. `list-virtual-servers(1)` サブコマンドを使用して、仮想サーバーを一覧表示します。
2. `set(1)` サブコマンドを使用して、指定した仮想サーバーの値を変更します。
   仮想サーバーは、ドット表記名で指定します。

▼ 仮想サーバーを削除する

既存の仮想サーバーを削除するには、リモートモードで `delete-virtual-server` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 `list-virtual-servers(1)` サブコマンドを使用して、仮想サーバーを一覧表示します。

3 必要に応じて、仮想サーバーが削除されることをユーザーに通知します。

4 `delete-virtual-server(1)` サブコマンドを使用して、仮想サーバーを削除します。

5 変更内容を適用するために、**Enterprise Server** を再起動します。
94 ページの「ドメインの再起動」を参照してください。

例 16–18 仮想サーバーの削除

この例は、`localhost` から `sampleServer` という仮想サーバーを削除します。

```
asadmin> delete-virtual-server sampleServer
Command delete-virtual-server executed successfully.
```

参照 コマンド行に `asadmin help delete-virutal-server` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

デフォルトの Web モジュールを仮想サーバーに割り当てる

デフォルト Web モジュールは、デフォルト仮想サーバーや個々の新しい仮想サーバーに割り当てることができます。仮想サーバーのデフォルト Web モジュールにアクセスするには、ブラウザで仮想サーバーの URL をポイントします。ただし、コンテキストルートは指定しないでください。次に例を示します。

```
http://myvserver:3184/
```

デフォルト Web モジュールが割り当てられていない仮想サーバーは、ドキュメントルート (通常は *domain-dir* `/docroot`) から HTML または JavaServer Pages™ (JSP™) のコンテンツを提供します。この HTML または JSP のコンテンツにアクセスするには、ブラウザで仮想サーバーの URL をポイントします。コンテキストルートは指定せず、ターゲットのファイルを指定します。

次に例を示します。

```
http://myvserver:3184/hellothere.jsp
```

▼ 仮想サーバーをアプリケーションまたはモジュールに割り当てる

配備済みのアプリケーションや Web モジュールに仮想サーバーを割り当てることができます。

始める前に　アプリケーションやモジュールは、すでに配備済みである必要があります。詳細については、『Sun GlassFish Enterprise Server v3 Application Deployment Guide』を参照してください。

1. 管理コンソールで、関連する構成の下にある **HTTP** サービスコンポーネントを開きます。
2. **HTTP** サービスコンポーネントの下にある仮想サーバーコンポーネントを開きます。
3. デフォルト **Web** モジュールを割り当てる仮想サーバーを選択します。
4. 「デフォルト **Web** モジュール」ドロップダウンリストから、アプリケーションまたは **Web** モジュールを選択します。

   詳細については、305 ページの「デフォルトの Web モジュールを仮想サーバーに割り当てる」を参照してください。

◆ ◆ ◆ 第 17 章

ORB (Object Request Broker) の管理

Sun GlassFish™ Enterprise Server は、相互運用性を保証する一連の標準的なプロトコルと形式をサポートします。これらのプロトコルの中には、CORBA で定義されているものがあります。ORB (Object Request Broker) は、CORBA の中枢となるコンポーネントです。ORB は、オブジェクトの特定と検索、接続管理、およびデータと要求の配信に必要なインフラストラクチャーを提供します。この章では、ORB と IIOP リスナーの設定方法について説明します。

ここでは、次のテーマを取り上げます。

本章で説明するタスクを 管理コンソール から実行する手順については、管理コンソール オンラインヘルプを参照してください。

ORB について

CORBA (Common Object Request Broker Architecture) モデルのベースになっているのは、明確に定義されたインタフェースを介して分散型のオブジェクトやサーバーにサービスを要求するクライアントです。こうしたクライアントは、リモートメソッド要求の形式でオブジェクトに対して要求を発行します。「リモートメソッド要求」では、実行する必要のある操作に関する情報が伝送されます。この情報には、サービスプロバイダのオブジェクト名 (オブジェクト参照) と、存在する場合は、起動メソッドのパラメータが含まれます。CORBA は、オブジェクトの登録、オブジェクトの配置、オブジェクトのアクティブ化、要求の多重分離、エラー処理、整列化、操作のディスパッチをはじめとするネットワークプログラミングのタスクを自動的に処理します。

ORB の設定

個々の CORBA オブジェクトが相互に対話することはありません。代わりに、オブジェクトはリモートスタブを通して、ローカルホストで動作している IIOP (Internet Inter-Orb Protocol) に要求を行います。続いて、ローカルの ORB が IIOP を使用して、ほかのホスト上の ORB に要求を渡します。リモート ORB は、適切なオブジェクトを検出し、要求を処理して、結果を返します。

アプリケーションやオブジェクトでは、RMI-IIOP により、IIOP を RMI (Remote Method Invocation) として使用することが可能になっています。エンタープライズ Bean (EJB モジュール) のリモートクライアントは、RMI-IIOP を使用して Enterprise Server と通信します。

IIOP リスナーの管理

「*IIOP* リスナー」は、Enterprise JavaBeans のリモートクライアントおよびほかの CORBA ベースのクライアントから受信する接続を受け付ける待機ソケットです。Enterprise Server では、複数の IIOP リスナーを設定できます。各リスナーに対して、ポート番号 (オプション、デフォルトは 1072)、ネットワークアドレス、およびセキュリティー属性 (オプション) を指定してください。複数のリスナーを作成する場合は、リスナーごとに異なるポート番号を割り当てる必要があります。

ここでは、次のテーマを取り上げます。

- 308 ページの「IIOP リスナーを作成する」
- 309 ページの「IIOP リスナーを一覧表示する」
- 309 ページの「IIOP リスナーを更新する」
- 310 ページの「IIOP リスナーを削除する」

▼ IIOP リスナーを作成する

IIOP リスナーを作成するには、リモートモードで `create-iiop-listener` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-iiop-listener(1)` サブコマンドを使用して、**IIOP** リスナーを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 17–1 IIOP リスナーの作成

この例では、sample_iiop_listener という名前の IIOP リスナーを作成します。

```
asadmin> create-iiop-listener --listeneraddress 192.168.1.100
--iiopport 1400 sample_iiop_listener
Command create-iiop-listener executed successfully.
```

参照 コマンド行に asadmin help create-iiop-listener と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ IIOP リスナーを一覧表示する

既存の IIOP リスナーを一覧表示するには、リモートモードで list-iiop-listeners サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-iiop-listeners(1) サブコマンドを使用して、IIOP リスナーを一覧表示します。

例 17–2 IIOP リスナーの一覧表示

この例では、サーバーインスタンスの IIOP リスナーをすべて表示します。

```
asadmin> list-iiop-listeners
 orb-listener-1
SSL
SSL_MUTUALAUTH
sample_iiop_listener
Command list-iiop-listeners executed successfully.
```

参照 コマンド行に asadmin help list-iiop-listeners と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ IIOP リスナーを更新する

1 list-iiop-listeners(1) サブコマンドを使用して、IIOP リスナーを一覧表示します。

2 set(1) サブコマンドを使用して、指定した IIOP リスナーの値を変更します。
リスナーは、そのドット表記名で指定します。

例 17–3 IIOP リスナーの更新

この例では、SSL を有効から無効に変更します。

```
asadmin> set "server.iiop-service.iiop-listener.SSL.enabled"
server.iiop-service.iiop-listener.SSL.enabled=false
Command set executed successfully.
```

▼ IIOP リスナーを削除する

IIOP リスナーを削除するには、リモートモードで delete-iiop-listener サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-iiop-listeners(1) サブコマンドを使用して、IIOP リスナーを一覧表示します。
3. delete-iiop-listener(1) サブコマンドを使用して、IIOP リスナーを削除します。
4. 変更内容を適用するために、Enterprise Server を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 17–4 IIOP リスナーの削除

この例では、sample_iiop_listener という名前の IIOP リスナーを削除します。

```
asadmin> delete-iiop-listener sample_iiop_listener
 Command delete-iiop-listener executed successfully.
```

参照 コマンド行に asadmin help delete-iiop-listener と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 18 章

18 JavaMail サービスの管理

Sun Java™ Enterprise Server には、JavaMail API、およびアプリケーションコンポーネントでインターネット上で電子メール通知を送信し、IMAP および POP3 メールサーバーから電子メールを読み取ることを可能にする JavaMail サービスプロバイダが含まれています。

ここでは、次のテーマを取り上げます。

本章で説明するタスクを 管理コンソール から実行する手順については、管理コンソール オンラインヘルプを参照してください。

JavaMail について

JavaMail API はメールシステムをモデル化する一連の抽象 API です。JavaMail API は、プラットフォームやプロトコルに依存しないフレームワークを提供して、メールおよびメッセージングアプリケーションを構築し電子メッセージの読み取りと送信の機能を提供します。サービスプロバイダは特定のプロトコルを実装します。API を使用して、アプリケーションに電子メール機能を追加できます。JavaMail は Java アプリケーションから、ネットワークまたはインターネット上の IMAP (Internet Message Access Protocol) および SMTP (メール転送プロトコル) 対応のメールサーバーにアクセスできます。API にはメールサーバー機能がないため、JavaMail を使用するにはメールサーバーにアクセスする必要があります。

JavaMail API は、Java プラットフォームのオプションパッケージとして実装され、Java EE プラットフォームの一部としても使用できます。

JavaMail API の詳細は、JavaMail Web サイト `http://java.sun.com/products/javamail/` を参照してください。

JavaMail リソースの管理

メールセッションを作成すると、サーバー側のコンポーネントとアプリケーションが、割り当てたセッションプロパティーを使用して JavaMail サービスと JNDI にアクセス可能になります。メールセッションを作成するときに、メールホスト、トランスポートプロトコルとストアプロトコル、およびデフォルトのメールユーザーを指定できるので、JavaMail を使用するコンポーネントはこれらのプロパティーを設定する必要がありません。Enterprise Server は単一のセッションオブジェクトを作成して、セッションオブジェクトを必要とするすべてのコンポーネントから使用できるようにするので、電子メールを大量に使用するアプリケーションにメリットがあります。

次のような JavaMail 設定を指定できます。

- 「**JNDI** 名」。メールセッションの一意の名前。JavaMail リソースのネーミングサブコンテキストプレフィックス mail/ を使用することをお勧めします。例: `mail/MySession`
- 「メールホスト」。デフォルトメールサーバーのホスト名。プロトコル固有のホストプロパティーが提供されていない場合に、Store オブジェクトと Transport オブジェクトの接続メソッドはこの値を使用します。この名前は実際のホスト名として解決可能でなければいけません。
- 「デフォルトユーザー」。メールサーバーへの接続時に渡されるユーザー名。プロトコル固有の username プロパティーが提供されていない場合に、Store オブジェクトと Transport オブジェクトの接続メソッドはこの値を使用します。
- 「デフォルトの返信用アドレス」。デフォルトユーザーの電子メールアドレス。*username@host.domain* の形式で入力します。
- 「説明」。コンポーネントの説明。
- 「セッション」。今回メールセッションを有効にするか、無効にするかを指定します。

ここでは、次のテーマを取り上げます。

- 312 ページの「JavaMail リソースを作成する」
- 313 ページの「JavaMail リソースを一覧表示する」
- 314 ページの「JavaMail リソースを更新する」
- 314 ページの「JavaMail リソースを削除する」

▼ JavaMail リソースを作成する

JavaMail セッションリソースを作成するには、リモートモードでサブコマンド `create-javamail-resource` を使用します。JavaMail セッションリソースの JNDI 名は通例、mail/ の命名サブコンテキスト (例: `mail/MyMailSession.`) を含みます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. create-javamail-resource(1) サブコマンドを使用して、**JavaMail** リソースを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 **18–1** JavaMail リソースの作成

この例は、mail/MyMailSession という名前の JavaMail リソースを作成します。--fromaddress オプションでエスケープ文字 (\) を使用して、ドット (.) とアットマーク (@) を区別します。

```
asadmin> create-javamail-resource --mailhost localhost
--mailuser sample --fromaddress sample\@sun\.com mail/MyMailSession
Command create-javamail-resource executed successfully.
```

参照 コマンド行に asadmin help create-javamail-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JavaMail リソースを一覧表示する

既存の JavaMail セッションリソースを一覧表示するには、リモートモードで list-javamail-resources サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-javamail-resources(1) サブコマンドを使用して、**JavaMail** リソースを一覧表示します。

例 **18–2** JavaMail リソースの一覧表示

この例は、localhost の JavaMail リソースを一覧表示します。

```
asadmin> list-javamail-resources
mail/MyMailSession
Command list-javamail-resources executed successfuly.
```

参照　コマンド行に asadmin help list-javamail-resources と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JavaMail リソースを更新する

1. list-javamail-resources(1) サブコマンドを使用して、**JavaMail** リソースを一覧表示します。
2. set(1) サブコマンドを使用して、指定した **JavaMail** リソースの値を変更します。
   リソースは、ドット表記名で指定します。

例 **18–3**　JavaMail リソースの更新

この例は、joeserver を joe に変更します。

```
asadmin> set server.resources.mail-resource.mail/
MyMailSession.user=joeserver.resources.mail-resource.mail/
MyMailSession.user=joe
Command set executed successfully.
```

▼ JavaMail リソースを削除する

JavaMail セッションリソースを削除するには、リモートモードで delete-javamail-resource サブコマンドを使用します。

始める前に　サブコマンド delete-javamail-resource を実行するには、指定のリソースへの参照を削除しておく必要があります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. list-javamail-resources(1) サブコマンドを使用して、**JavaMail** リソースを一覧表示します。
3. delete-javamail-resource(1) サブコマンドを使用して、**JavaMail** リソースを削除します。
4. 変更内容を適用するために、**Enterprise Server** を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 18–4 JavaMail リソースの削除

この例は、mail/MyMailSession という名前の JavaMail セッションリソースを削除します。

```
asadmin> delete-javamail-resource  mail/MyMailSession
Command delete-javamail-resource executed successfully.
```

参照 コマンド行に asadmin help delete-javamail-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

◆ ◆ ◆ 第 19 章

JMS (Java Message Service) の管理

Sun では、Sun GlassFish™ Message Queue ソフトウェアを Sun GlassFish Enterprise Server に統合することで、JMS (Java™ Message Service) API を実装しています。この章では、`asadmin` コマンド行ユーティリティーを使用して、Enterprise Server 環境で JMS リソースを管理する手順について説明します。

注 – JMS リソースは、Enterprise Server の Full Platform Profile のみでサポートされ、Web Profile ではサポートされません。

ここでは、次のテーマを取り上げます。

- 317 ページの「JMS について」
- 319 ページの「JMS 物理送信先の管理」
- 322 ページの「JMS 接続ファクトリと送信先の管理」
- 326 ページの「JMS ホストの管理」
- 329 ページの「接続のアドレス指定の管理」
- 330 ページの「JMS のリソースアダプタの設定」
- 331 ページの「JMS に関するトラブルシューティング」

この章のタスクを 管理コンソール を使用して実行する場合の手順は、管理コンソールのオンラインヘルプで説明します。

JMS について

JMS API は、Java EE アプリケーションおよびコンポーネントで、メッセージの作成、送信、受信、および読み取りを可能にするメッセージング標準です。この API によって、緩やかに結合され、信頼性が高く、非同期の分散通信が可能となります。

一般的に、Enterprise Server の JMS メッセージングのサポートや、特にメッセージ駆動型 Bean のサポートでは、「*JMS* プロバイダ」が必要です。Enterprise Server

は、Sun GlassFish メッセージキュー ソフトウェアをネイティブの JMS プロバイダとして使用し、透過的な JMS メッセージングのサポートを提供します。このサポートは、Enterprise Server では「*JMS* サービス」と呼ばれます。JMS で必要になるのは最低限の管理のみです。JMS クライアントが JMS 管理対象オブジェクトにはじめてアクセスするときに、クライアントの JVM が Enterprise Server から JMS 構成を受け取ります。

JMS リソースはコネクタの一種です。Message Queue は、「コネクタモジュール」によって Enterprise Server と統合されます。コネクタモジュールはリソースアダプタとも呼ばれ、Java EE Connector Architecture Specification 1.6 で定義されています。Enterprise Server に配備されるすべての Java EE コンポーネントは、コネクタモジュールによって統合された JMS プロバイダを使用して、JMS メッセージを交換します。JMS リソースが Enterprise Server に作成されるときに、コネクタリソースがバックグラウンドで作成されます。JMS 操作はそれぞれコネクタのランタイムを呼び出し、Message Queue コネクタモジュールをバックグラウンドで使用します。Enterprise Server は JMS 接続を自動的にプールします。

すべての JMS 接続で使用されるプロパティーを設定できます。これらのプロパティーを実行時に更新する場合は、プロパティーの更新後に作成された接続ファクトリのみに、更新した値が適用されます。既存の接続ファクトリは元のプロパティー値のままになります。ほとんどの値は、Enterprise Server を再起動して有効にする必要があります。手順については、94 ページの「ドメインの再起動」を参照してください。Enterprise Server を再起動せずに更新できるプロパティーは、デフォルト JMS ホストだけです。

Message Queue ブローカのモード

Message Queue は、Enterprise Server に LOCAL、REMOTE、または EMBEDDED のモードで統合できます。これらのモードは、JMS の `type` 属性で表されます。

- 「**LOCAL** モード」。Enterprise Server は、デフォルト JMS ホストとして指定された Message Queue ブローカを起動または停止します。Message Queue プロセスは、Enterprise Server プロセスとは別の仮想マシンで開始されます。Enterprise Server はブローカに追加ポートを提供し、ブローカはこのポートを使用して RMI レジストリを起動します。このポート番号は、そのインスタンス用に設定された JMS ポートの番号に 100 を足した値になります。たとえば、JMS ポート番号が 37676 である場合、この追加ポートの番号は 37776 になります。

  LOCAL モードでは、「起動引数」属性を使用して Message Queue ブローカの起動パラメータを指定します。

- 「**REMOTE** モード」。type 属性が REMOTE に設定されている場合は、Message Queue ブローカを Enterprise Server から独立して起動および停止する必要があります。Message Queue のツールを使用して、ブローカを設定および調整する必要があ

ります。この状況では、Enterprise Server は外部で設定されたブローカを使用するか、ブローカクラスタを使用します。「REMOTE」 の type 属性はクラスタ環境にもっとも適しています。

REMOTE モードでは、Message Queue のツールを使用して、Message Queue ブローカの起動パラメータを指定する必要があります。「起動引数」属性は無視されます。

- 「**EMBEDDED** モード」 (デフォルト)。JMS の type 属性が EMBEDDED に設定されている場合、Enterprise Server と JMS ブローカは同じ仮想マシンに共存します。JMS サービスは、Enterprise Server によってインプロセスで起動され管理されます。

  EMBEDDED モードでは、JMS 操作はネットワークスタックの処理を省略するため、パフォーマンスが最適化されます。

Message Queue の管理については、『Sun GlassFish Message Queue 4.4 Administration Guide』を参照してください。

JMS 物理送信先の管理

メッセージは、JMS プロバイダの「物理送信先」を使用して、コンシューマにルーティングおよび配信されます。物理送信先は、管理対象オブジェクト (`Topic` または `Queue` 送信先リソースなど) によって識別およびカプセル化されます。アプリケーションコンポーネントはこのオブジェクトを使用して、生成しているメッセージの送信先と、消費しているメッセージの発信元を指定します。送信先リソースを設定する手順については、323 ページの「接続ファクトリまたは送信先リソースを作成する」を参照してください。

メッセージ駆動型 Bean が配備され、この Bean が待機する物理送信先が存在しない場合、Enterprise Server は自動的に物理送信先を作成し、`maxNumActiveConsumers` プロパティーの値を `-1` に設定します。ただし、あらかじめ物理送信先を作成しておく方がよい方法です。アプリケーションが最初に送信先リソースにアクセスすると、Message Queue は、送信先リソースの名前プロパティーで指定した物理送信先を自動的に作成します。物理送信先は一時的なものなので、Message Queue の構成プロパティーで指定した期限が切れると効力を失います。

ここでは、次のテーマを取り上げます。

▼ JMS 物理送信先を作成する

本稼動環境では、必ず物理送信先を作成する必要があります。ただし、開発およびテスト段階では、この手順は不要です。物理送信先を作成するには、リモートモードで `create-jmsdest` サブコマンドを使用します。

物理送信先は、実際にはサーバーオブジェクトというよりも Message Queue オブジェクトであるため、プロパティーを更新する場合は Message Queue ブローカのコマンドを使用します。Message Queue プロパティーの詳細は、『Sun GlassFish Message Queue 4.4 Administration Guide』を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-jmsdest(1)` サブコマンドを使用して、JMS 物理送信先を作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. (省略可能) 必要な場合は、サーバーを再起動します。
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 **19–1** JMS 物理送信先の作成

この例では、`PhysicalQueue` という名前のキューを作成します。

```
asadmin> create-jmsdest --desttype queue --property
User=public:Password=public PhysicalQueue
Command create-jmsdest executed successfully.
```

参照 コマンド行に `asadmin help create-jmsdest` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JMS 物理送信先を一覧表示する

既存の JMS 物理送信先を一覧表示するには、リモートモードで `list-jmsdest` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 list-jmsdest(1) サブコマンドを使用して、既存の JMS 物理送信先を一覧表示します。

例 19–2 JMS 物理送信先の一覧表示

この例では、デフォルトサーバーインスタンスの物理送信先を一覧表示します。

```
asadmin> list-jmsdest
PhysicalQueue queue {}
PhysicalTopic topic {}
Command list-jmsdest executed successfully.
```

参照 コマンド行に asadmin help list-jmsdest と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 物理送信先からメッセージを消去する

指定したターゲットの JMS サービス構成の物理送信先からメッセージを消去するには、リモートモードで flush-jmsdest サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 flush-jmsdest(1) サブコマンドを使用して、JMS 物理送信先からメッセージを消去します。

3 (省略可能) 必要な場合は、サーバーを再起動します。
プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 19–3 JMS 物理送信先からのメッセージのフラッシュ

この例では、PhysicalQueue という名前のキューからメッセージを消去します。

```
asadmin> flush-jmsdest --desttype queue PhysicalQueue
Command flush-jmsdest executed successfully
```

参照 コマンド行に asadmin help flush-jmsdest と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JMS 物理送信先を削除する

指定した JMS 物理送信先を削除するには、リモートモードで `delete-jmsdest` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. `list-jmsdest(1)` サブコマンドを使用して、既存の JMS 物理送信先を一覧表示します。
3. `delete-jmsdest(1)` サブコマンドを使用して、物理送信先を削除します。

例 19–4 物理送信先の削除

この例では、`PhysicalQueue` という名前のキューを削除します。

```
asadmin> delete-jmsdest --desttype queue PhysicalQueue
Command delete-jmsdest executed successfully
```

参照 コマンド行に `asadmin help delete-jmsdest` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

JMS 接続ファクトリと送信先の管理

JMS API は、2 種類の管理対象オブジェクトを使用します。「接続ファクトリオブジェクト」は、アプリケーションがプログラムによってほかの JMS オブジェクトを作成できるようにします。「送信先オブジェクト」は、メッセージのリポジトリとして動作します。これらのオブジェクトがどのように作成されるかは、JMS の実装ごとに異なります。Enterprise Server では、次のタスクの実行により JMS が実装されます。

- 接続ファクトリの作成。
- 送信先の作成。送信先の作成には、物理送信先の作成と、物理送信先が参照する送信先リソースの作成が必要です。

JMS アプリケーションは、Java Naming and Directory Interface (JNDI) API を使用して、接続ファクトリと送信先リソースにアクセスします。通常、JMS アプリケーションは 1 つ以上の接続ファクトリと 1 つ以上の宛先を使用します。アプリケーションについて確認するか、アプリケーション開発者に問い合わせることで、作成が必要なリソースを決定できます。リソースを作成する順序は重要ではありません。

Enterprise Server は、次の接続ファクトリオブジェクトを提供します。

- ポイントツーポイント通信で使用する QueueConnectionFactory オブジェクト
- パブリッシュ - サブスクライブ通信で使用する TopicConnectionFactory オブジェクト
- ポイントツーポイント通信とパブリッシュ - サブスクライブ通信の両方で使用できる ConnectionFactory オブジェクト (新しいアプリケーションで推奨)

Enterprise Server は、次の送信先オブジェクトを提供します。

- ポイントツーポイント通信で使用する Queue オブジェクト
- パブリッシュ - サブスクライブ通信で使用する Topic オブジェクト

ここでは、次のテーマを取り上げます。

- 323 ページの「接続ファクトリまたは送信先リソースを作成する」
- 324 ページの「JMS リソースを一覧表示する」
- 325 ページの「接続ファクトリまたは送信先リソースを削除する」

この節で使用するサブコマンドは、接続ファクトリリソースと送信先リソースのどちらの管理にも使用できます。物理送信先を管理する手順については、319 ページの「JMS 物理送信先の管理」を参照してください。

▼ 接続ファクトリまたは送信先リソースを作成する

作成する JMS 接続ファクトリごとに、Enterprise Server はコネクタ接続プールとコネクタリソースを作成します。作成する JMS 送信先ごとに、Enterprise Server はコネクタ管理オブジェクトリソースを作成します。JMS リソースを削除すると、Enterprise Server は自動的にコネクタリソースを削除します。

JMS 接続ファクトリリソースまたは送信先リソースを作成するには、リモートモードで create-jms-resource コマンドを使用します。

ヒント – asadmin create-jms-resource コマンドで addresslist プロパティーを (host:mqport,host2:mqport,host3:mqport の形式で) 指定するには、コロン (:) を \\ を使用してエスケープします。たとえば、host1\\: mqport,host2\\: mqport,host3\\: mpqport のようになります。エスケープ文字の使用方法については、asadmin(1M) の概念のページを参照してください。

JMS 接続ファクトリを更新するには、更新する接続ファクトリのコネクタ接続プールに対して set サブコマンドを使用します。271 ページの「コネクタ接続プールを更新する」を参照してください。

送信先を更新するには、管理オブジェクトリソースに対して set サブコマンドを使用します。287 ページの「管理対象オブジェクトを更新する」を参照してください。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. `create-jms-resource(1)` コマンドを使用して、JMS リソースを作成します。
   このサブコマンドのプロパティーについては、このマニュアルページに記載されています。

3. (省略可能) 必要な場合は、サーバーを再起動します。
   プロパティーの中には、サーバーの再起動を求めるものもあります。39 ページの「サーバーの再起動が必要な構成の変更」を参照してください。サーバーを再起動する必要がある場合は、94 ページの「ドメインの再起動」を参照してください。

例 **19–5** JMS 接続ファクトリの作成

この例では、JNDI 名が `jms/DurableConnectionFactory` の `javax.jms.ConnectionFactory` タイプの接続ファクトリリソースを作成します。`ClientId` プロパティーは、接続ファクトリのクライアント ID を設定し、接続ファクトリを永続サブスクリプションで使用できるようにします。JMS リソースの JNDI 名には、慣習的に `jms/` のネーミングサブコンテキストを含めます。

```
asadmin> create-jms-resource --restype javax.jms.ConnectionFactory
--description "connection factory for durable subscriptions"
--property ClientId=MyID jms/DurableConnectionFactory
Command create-jms-resource executed successfully.
```

例 **19–6** JMS 送信先の作成

この例では、JNDI 名が `jms/MyQueue` の送信先リソースを作成します。

```
asadmin> create-jms-resource --restype javax.jms.Queue
--property Name=PhysicalQueue jms/MyQueue
Command create-jms-resource executed successfully.
```

参照 コマンド行に `asadmin help create-jms-resource` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JMS リソースを一覧表示する

既存の接続ファクトリと送信先リソースを一覧表示するには、リモートモードで `list-jms-resources` サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 list-jms-resources(1) サブコマンドを使用して、既存の JMS リソースを一覧表示します。

例 19–7 すべての JMS リソースの表示

この例では、既存の JMS 接続ファクトリと送信先リソースをすべて表示します。

```
asadmin> list-jms-resources
jms/Queue
jms/ConnectionFactory
jms/DurableConnectionFactory
jms/Topic
Command list-jms-resources executed successfully
```

例 19–8 指定したタイプの JMS リソースの一覧表示

この例では、リソースタイプが javax のリソースを一覧表示します。

```
asadmin> list-jms-resources --restype javax.jms.TopicConnectionFactory
jms/DurableTopicConnectionFactory
jms/TopicConnectionFactory
Command list-jms-resources executed successfully.
```

参照 コマンド行に asadmin help list-jms-resources と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 接続ファクトリまたは送信先リソースを削除する

指定した接続ファクトリまたは送信先リソースを削除するには、リモートモードで delete-jms-resource サブコマンドを使用します。

始める前に このサブコマンドを実行する前に、削除する JMS リソースへの参照をすべて削除する必要があります。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-jms-resources(1) サブコマンドを使用して、既存の JMS リソースを一覧表示します。

3 delete-jms-resource(1) サブコマンドを使用して、JMS リソースを削除します。

例 19–9 JMS リソースの削除

この例では、jms/Queue リソースを削除します。

```
asadmin> delete-jms-resource jms/Queue
Command delete-jms-resource executed successfully
```

参照 コマンド行に asadmin help delete-jms-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

JMS ホストの管理

「*JMS* ホスト」は、Message Queue ブローカを表します。JMS には「*JMS* ホストリスト」 (AddressList プロパティー) が含まれ、このリストには Enterprise Server が使用するすべての JMS ホストが含まれます。JMS ホストリストには、指定したメッセージキュー ブローカのホストとポートが取り込まれ、JMS ホスト構成が変更されるたびに更新されます。JMS リソースを作成するとき、またはメッセージ駆動型 Bean を配備するときに、リソースまたは Bean が JMS ホストリストを継承します。

JMS ホストリスト内のホストの 1 つが、デフォルト JMS ホストに指定されます。Message Queue ブローカモードが LOCAL に設定されている場合は、Enterprise Server がデフォルト JMS ホストを起動します。

ここでは、次のテーマを取り上げます。

- 326 ページの「JMS ホストを作成する」
- 327 ページの「JMS ホストを一覧表示する」
- 328 ページの「JMS ホストを更新する」
- 328 ページの「JMS ホストを削除する」

▼ JMS ホストを作成する

Enterprise Server では、デフォルト JMS ホストの default_JMS_host が提供されます。デフォルト JMS ホストは、JMS 送信先の作成や削除など、メッセージキュー ブローカのすべての管理操作を実行するために、Enterprise Server によって使用されます。

通常は、新しい JMS ホストを作成する必要はありません。このタスクは上級ユーザー向けのタスクです。追加の JMS ホストを作成するには、リモートモードで create-jms-host サブコマンドを使用します。

JMS は、実際にはサーバーオブジェクトというよりも Message Queue オブジェクトであるため、プロパティーを更新する場合は Message Queue ブローカのコマンドを使用

します。Message Queue プロパティーの詳細は、『Sun GlassFish Message Queue 4.4 Administration Guide』を参照してください。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 create-jms-host(1) サブコマンドを使用して、JMS ホストを作成します。
このサブコマンドのプロパティーについては、サブコマンドのマニュアルページを参照してください。

例 19–10 JMS ホストの作成

この例では、MyNewHost という名前の JMS ホストを作成します。

```
asadmin> create-jms-host --mqhost pigeon --mqport 7677 MyNewHost
Command create-jms-host executed successfully.
```

参照 コマンド行に asadmin help create-jms-host と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JMS ホストを一覧表示する

既存の JMS ホストを一覧表示するには、リモートモードで list-jms-hosts サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-jms-hosts(1) サブコマンドを使用して、JMS ホストを一覧表示します。

例 19–11 JMS ホストの一覧表示

次のサブコマンドは、既存の JMS ホストを一覧表示します。

```
asadmin> list-jms-hosts
default_JMS_host
MyNewHost
Command list-jmsdest executed successfully
```

▼ JMS ホストを更新する

1 list-jms-hosts(1) サブコマンドを使用して、JMS ホストを一覧表示します。

2 set(1) サブコマンドを使用して、JMS ホストを変更します。

例 19–12 JMS ホストの更新

この例では、デフォルト JMS ホストの host 属性の値を変更します。この属性のデフォルト値は localhost です。

```
asadmin>  set server-config.jms-service.jms-host.default_JMS_host.host=
"archie.india.sun.com"
```

参照 コマンド行に asadmin help set と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ JMS ホストを削除する

JMS ホストを JMS サービスから削除するには、リモートモードで delete-jms-host サブコマンドを使用します。JMS ホストだけを削除すると、新しい JMS ホストを作成するまで、Message Queue ブローカを起動できなくなります。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-jms-hosts(1) サブコマンドを使用して、JMS ホストを一覧表示します。

3 delete-jms-host(1) サブコマンドを使用して、JMS ホストを削除します。

例 19–13 JMS ホストの削除

この例では、MyNewHost という名前の JMS ホストを削除します。

```
asadmin> delete-jms-host MyNewHost
Command delete-jms-host executed successfully.
```

参照 コマンド行に asadmin help delete-jms-host と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

接続のアドレス指定の管理

一部の JMS リソースは、JMS ホストリスト (AddressList) の構成を使用します。このリストは、Enterprise Server で定義されている JMS ホストのホスト名とポートを指定します。JMS ホスト構成が変更されるたびに、JMS ホストリストは更新されます。JMS ホストリストは、JMS リソースの作成時およびメッセージ駆動型 Bean の配備時に、リソースまたは Bean によって継承されます。

メッセージキュー ソフトウェアでは、AddressList プロパティーは imqAddressList と呼ばれます。

ここでは、次のテーマを取り上げます。

JMS 接続プールの設定

Enterprise Server は JMS 接続を自動的にプールします。JMS 接続プールが作成されるときに、ManagedConnectionFactory のインスタンスが 1 つ関連付けられます。AddressList プロパティーを ManagedConnectionFactory プロパティーとして設定すると、ManagedConnectionFactory 値の AddressList の構成が、Enterprise Server で定義された値よりも優先されます。

create-connector-connection-pool サブコマンドを使用して、既存のプールを管理します。手順については、269 ページの「コネクタ接続プールの管理」を参照してください。

デフォルトでは、JMS サービス属性の addresslist-behavior が random に設定されます。この場合、ManagedConnectionFactory から作成された各物理送信先 (ManagedConnection) は、主ブローカを AddressList プロパティーからランダムに選択します。

接続が失われたときに Enterprise Server が主ブローカに再接続を試みるかどうかを指定するには、set(1) サブコマンドを使用して、JMS サービスの reconnect-enabled 属性を設定します。再試行の回数と間隔を指定するには、それぞれ reconnect-attempts 属性と reconnect-interval-in-seconds 属性で設定します。

再接続が有効な場合に主ブローカで障害が発生すると、Enterprise Server は JMS ホストリスト (AddressList) の別のブローカに再接続を試みます。スキャンのロジックは、addresslist-behavior と addresslist-iterations の 2 つの JMS サービス属性で決定されます。これらの設定は、JMS 接続ファクトリ設定を使用してオーバーライドできます。Sun GlassFish メッセージキュー ソフトウェアは、フェイルオーバーが発生したときに、負荷を別のブローカに透過的に転送します。JMS のセマンティクスはフェイルオーバー中も維持されます。

リモートサーバーへのアクセス

プロバイダとホストをリモートシステムに変更すると、すべての JMS アプリケーションがリモートサーバーで実行するようになります。ローカルサーバーと、1 つまたは複数のリモートサーバーの両方を使用するには、`AddressList` プロパティーを使用して接続ファクトリリソースを作成します。これにより、リモートサーバーにアクセスする接続が作成されます。

JMS のリソースアダプタの設定

Enterprise Server は、`jmsra` という名前のシステムリソースアダプタを使用して JMS を実装します。JMS リソースを作成するときに、Enterprise Server は自動的にコネクタリソースを作成します。リソースアダプタの設定により、JMS プロバイダが XA をサポートするかどうかを指定できます。JMS プロバイダで可能な統合のモードを指定することもできます。

リソースアダプタでは、2 つの統合のモードをサポートしています。最初のモードは、統合の手段として JNDI を使用します。この場合、管理対象オブジェクトが JMS プロバイダの JNDI ツリーに設定され、汎用リソースアダプタがそれらを検索して使用します。このモードが統合に適切でない場合は、JMS 管理対象オブジェクト JavaBean クラスの Java リフレクションを統合のモードとして使用することもできます。

JMS の汎用リソースアダプタ 1.6 は、外部 JMS プロバイダ (IBM WebSphere MQ、Tibco EMS、Sonic MQ など) の JMS クライアントライブラリをラップできる Java EE コネクタ 2.0 リソースアダプタです。これにより、すべての JMS プロバイダは Sun GlassFish Enterprise Server などの Java EE 6 アプリケーションサーバーに統合されます。アダプタは、Java EE 6 アプリケーションサーバーの管理ツールを使用して配備および設定可能な `.rar` アーカイブです。

▼ 汎用リソースアダプタを設定する

汎用リソースアダプタを配備する前に、Enterprise Server が JMS クライアントライブラリを使用できるようにする必要があります。一部の JMS プロバイダでは、クライアントライブラリにネイティブライブラリが含まれている場合もあります。この場合は、これらのネイティブライブラリをすべての Enterprise Server JVM から使用できるようにする必要があります。

1. コネクタモジュールを配備する場合と同じように、汎用リソースアダプタを配備します。

2. コネクタ接続プールを作成します。
269 ページの「コネクタ接続プールを作成する」を参照してください。

3 コネクタリソースを作成します。
273 ページの「コネクタリソースを作成する」参照してください。

4 管理対象オブジェクトリソースを作成します。
285 ページの「管理対象オブジェクトリソースを作成する」を参照してください。

5 セキュリティーの **Enterprise Server** ポリシーファイルに次の変更を行います。

- `sjsas_home/domains/domain1/config/server.policy` ファイルに、次の行を追加します。

```
java.util.logging.LoggingPermission "control"
```

- `sjsas_home/lib/appclient/client.policy` ファイルに、次のアクセス権を追加します。

```
javax.security.auth.PrivateCredentialPermission
"javax.resource.spi.security.PasswordCredential ^ \"^\"","read":
```

JMS に関するトラブルシューティング

Enterprise Sever の起動時に、JMS サービスは使用可能になっていますが、必要になる (たとえば、JMS リソースを作成するとき) まで読み込まれません。`jms-ping(1)` サブコマンドを使用して、JMS サービスが動作しているか確認し、動作していない場合はサービスを起動します。`jms-ping` サブコマンドが組み込みの JMS サービスと通信できない場合は、エラーメッセージが表示されます。

問題が起きた場合は、次の点を考慮してください。

- Enterprise Server のログファイルを確認します。通常、ログファイルは *domain-dir*`/logs/server.log` にあります。

  ログファイルで Message Queue ブローカがメッセージに応答していないことを確認できた場合は、ブローカを停止して再起動します。
- ブローカのログファイルを確認します。通常、このファイルは *as-install*`/domains/domain1/imq/instances/imqbroker/log/log.txt` にあります。
- JMS の REMOTE モードでは、Message Queue を起動したあとに Enterprise Server を起動する必要があります。
- すべての Message Queue ブローカが停止している場合、JMS のデフォルト値を使用しているときは、Enterprise Server の停止または起動に 30 分かかります。このタイムアウトのデフォルト値は変更できます。次に例を示します。

```
asadmin set domain1.jms-service.reconnect-interval-in-seconds=5
```

◆ ◆ ◆ 第 20 章

Java Naming and Directory Interface (JNDI) サービスの管理

Java Naming and Directory Interface (JNDI) API は、別の種類のネームサービスおよびディレクトリサービスにアクセスするために使用されます。Java EE コンポーネントは、JNDI ルックアップメソッドを起動することによってオブジェクトを検出します。

ここでは、次のテーマを取り上げます。

- 333 ページの「JNDI について」
- 335 ページの「JNDI リソースの管理」

JNDI について

アプリケーションは JNDI API を呼び出すことにより、リソースとほかのプログラムオブジェクトを検出します。「リソース」は、データベースサーバーやメッセージングシステムなどのシステムへの接続を提供するプログラムオブジェクトです。JDBC リソースはデータソースと呼ばれる場合もあります。それぞれのリソースオブジェクトは人間が理解しやすい「*JNDI* 名」という一意の名前で識別されます。リソースオブジェクトと JNDI 名は、Enterprise Server に含まれているネーミングおよびディレクトリサービスによって相互にバインドされています。

新しい名前とオブジェクトのバインドが JNDI に入力されると、新しいリソースが作成されます。

ここでは、次のテーマを取り上げます。

- 334 ページの「Java EE ネーミング環境」
- 334 ページの「ネーミング環境とコンテナの動作」
- 335 ページの「ネーミング参照とバインディング情報」

Java EE ネーミング環境

JNDI 名は、Java EE サーバーが提供するネームサービスとディレクトリサービスによってオブジェクトにバインドされます。Java EE コンポーネントは JNDI API を介してこのサービスにアクセスするので、通常オブジェクトはその JNDI 名を使用します。たとえば、PointBase データベースの JNDI 名は `jdbc/Pointbase` となります。Enterprise Server は、起動時に構成ファイルから情報を読み取り、自動的に JNDI データベース名を名前空間に追加します。そのうちの 1 つが `jdbc/Pointbase` です。

Java EE アプリケーションクライアント、Enterprise JavaBeans、および Web コンポーネントは、JNDI ネーミング環境にアクセスする必要があります。

アプリケーションコンポーネントのネーミング環境は、配備またはアセンブリの際に、アプリケーションコンポーネントのビジネスロジックのカスタマイズを可能にするメカニズムです。この環境では、コンポーネントからソースコードにアクセスしたりソースコードを変更することなく、アプリケーションコンポーネントをカスタマイズすることができます。Java EE コンテナは、環境を「*JNDI* ネーミングコンテキスト」として実装し、アプリケーションコンポーネントのインスタンスに提供します。

ネーミング環境とコンテナの動作

アプリケーションコンポーネントの環境は次のとおり使用されます。

- アプリケーションコンポーネントのビジネスメソッドは JNDI インタフェースを使用して環境にアクセスします。アプリケーションコンポーネントプロバイダは、アプリケーションコンポーネントが実行時に環境内で提供されると期待しているすべての環境エントリを、配備記述子で宣言します。
- コンテナは、アプリケーションコンポーネントの環境を格納する JNDI ネーミングコンテキストの実装を提供します。また、コンテナは、配備担当者が各アプリケーションコンポーネントの環境を作成し、管理するツールも提供します。
- 配備担当者は、コンテナが提供するツールを使用して、アプリケーションコンポーネントの配備記述子で宣言された環境エントリを初期化します。配備担当者は環境エントリの値を設定し、変更します。
- コンテナは、実行時にアプリケーションコンポーネントのインスタンスで JNDI コンテキストを利用できるようにします。これらのインスタンスは、JNDI インタフェースを使用して環境エントリの値を取得します。

各アプリケーションコンポーネントは、自身の一連の環境エントリを定義します。同じコンテナ内のすべてのアプリケーションコンポーネントのインスタンスは、同じ環境エントリを共有します。アプリケーションコンポーネントのインスタンスは、実行時に環境を変更することはできません。

ネーミング参照とバインディング情報

「リソース参照」は、リソース用にコード化されたコンポーネントの名前を識別する配備記述子の要素です。たとえば、`jdbc/SavingsAccountDB` です。具体的には、コード化された名前はリソースの接続ファクトリを参照します。

リソースの JNDI 名とリソース参照名は同じではありません。このネーミングへのアプローチでは、配備前に 2 つの名前をマップする必要がありますが、同時にコンポーネントをリソースから分離します。この分離により、あとでコンポーネントが別のリソースにアクセスする必要があっても、名前を変更する必要がなくなります。この柔軟性により、既存のコンポーネントから Java EE アプリケーションを簡単にアセンブルすることが可能になります。

次の表に、Enterprise Server で使用される J2EE リソースの JNDI 検索と関連するリソース参照 を示します。

表 20–1 JNDI 検索名と関連する参照

JNDI ルックアップ名	関連するリソース参照
`java:comp/env`	アプリケーション環境エントリ
`java:comp/env/jdbc`	JDBC データソースリソースマネージャー接続ファクトリ
`java:comp/env/ejb`	EJB 参照
`java:comp/UserTransaction`	UserTransaction 参照
`java:comp/env/mail`	JavaMail セッション接続ファクトリ
`java:comp/env/url`	URL 接続ファクトリ
`java:comp/env/jms`	JMS 接続ファクトリと送信先
`java:comp/ORB`	アプリケーションコンポーネント間で共有された ORB インスタンス

JNDI リソースの管理

Enterprise Server 内で、カスタムリソースおよび外部 JNDI リソース用に環境を設定できます。カスタムリソースはローカル JNDI リポジトリにアクセスし、外部リソースは外部 JNDI リポジトリにアクセスします。どちらのタイプのリソースにも、ユーザーが指定したファクトリクラスの要素や JNDI 名前属性などが必要です。

- 336 ページの「カスタム JNDI リソースの管理」
- 338 ページの「外部 JNDI リソースの管理」

カスタム JNDI リソースの管理

カスタムリソースは、javax.naming.spi.ObjectFactory インタフェースを実装するサーバー全体のカスタムリソースオブジェクトファクトリを指定します。ここでは、次のテーマを取り上げます。

▼ カスタム JNDI リソースを作成する

カスタムリソースを作成するには、リモートモードで create-custom-resource サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. create-custom-resource(1) サブコマンドを使用して、カスタムリソースを作成します。
   サブコマンドのプロパティーについては、サブコマンドのマニュアルページを参照してください。

3. Enterprise Server を再起動します。
   94 ページの「ドメインの再起動」を参照してください。

例 20–1 カスタムリソースの作成

この例では、sample-custom-resource という名前のカスタムリソースを作成します。

```
asadmin> create-custom-resource --restype topic --factoryclass com.imq.topic
sample_custom_resource
Command create-custom-resource executed successfully.
```

参照 コマンド行に asadmin help create-custom-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ カスタム JNDI リソースを一覧表示する

既存のカスタムリソースを一覧表示するには、リモートモードで list-custom-resources サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 list-custom-resources(1) サブコマンドを使用して、カスタムリソースを一覧表示します。

例 20–2 カスタムリソースの一覧表示

この例では、既存のカスタムリソースを一覧表示します。

```
asadmin> list-custom-resources
sample_custom_resource01
sample_custom_resource02
Command list-custom-resources executed successfully
```

参照 コマンド行に asadmin help list-custom-resources と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ カスタム JNDI リソースを更新する

1 list-custom-resources(1) サブコマンドを使用して、カスタムリソースを一覧表示します。

2 set(1) サブコマンドを使用して、カスタム JNDI リソースを変更します。

例 20–3 カスタム JNDI リソースの更新

この例では、カスタムリソースを変更します。

```
asadmin> set server.resources.custom-resource.custom
/my-custom-resource.property.value=2010server.resources.custom-resource.custom
/my-custom-resource.property.value=2010
```

▼ カスタム JNDI リソースを削除する

カスタムリソースを削除するには、リモートモードで delete-custom-resource サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-custom-resources(1) サブコマンドを使用して、カスタムリソースを一覧表示します。

3 delete-custom-resource(1) サブコマンドを使用して、カスタムリソースを削除します。

例 **20–4** カスタムリソースの削除

この例では、sample-custom-resource という名前のカスタムリソースを削除します。

```
asadmin> delete-custom-resource sample_custom_resource
Command delete-custom-resource executed successfully.
```

参照 コマンド行に asadmin help delete-custom-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

外部 JNDI リソースの管理

Enterprise Server 上で動作しているアプリケーションでは、外部 JNDI リポジトリに保存されているリソースへのアクセスが必要となる場合があります。たとえば、汎用の Java オブジェクトを Java スキーマに従って LDAP サーバーに格納する場合があります。外部 JNDI リソース要素を使用すると、このような外部リソースリポジトリを設定できます。

ここでは、次のテーマを取り上げます。

▼ 外部 JNDI リソースを登録する

外部 JNDI リソースを登録するには、リモートモードで create-jndi-resource サブコマンドを使用します。

始める前に 外部 JNDI ファクトリは、javax.naming.spi.InitialContextFactory インタフェースを実装する必要があります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。
2. create-jndi-resource(1) サブコマンドを使用して、外部 JNDI リソースを登録します。
   サブコマンドのプロパティーについては、サブコマンドのマニュアルページを参照してください。

3 **Enterprise Server**を再起動します。
94 ページの「ドメインの再起動」を参照してください。

例 **20–5** 外部 JNDI リソースの登録

この例では、sample_jndi_resource が登録されています。

```
asadmin> create-jndi-resource --jndilookupname sample_jndi
--restype queue --factoryclass sampleClass --description "this is a sample jndi
resource" sample_jndi_resource
Command create-jndi-resource executed successfully
```

参照 コマンド行に asadmin help create-jndi-resource と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 外部 JNDI リソースを一覧表示する

既存の JNDI リソースをすべて表示するには、リモートモードで list-jndi-resources サブコマンドを使用します。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 list-jndi-resources(1) サブコマンドを使用して、既存の **JNDI** リソースを一覧表示します。

例 **20–6** JNDI リソースの一覧表示

この例では、JNDI リソースを一覧表示します。

```
asadmin> list-jndi-resources
jndi_resource1
jndi_resource2
jndi_resource3
Command list-jndi-resources executed successfully
```

参照 コマンド行に asadmin help list-jndi-resources と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 外部 JNDI エントリを一覧表示する

JNDI ツリーでエントリを参照および一覧表示するには、リモートモードで list-jndi-entries サブコマンドを使用します。すべてのエントリを表示するか、JNDI コンテキストまたはサブコンテキストを指定して特定のエントリだけを一覧表示できます。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. list-jndi-entries(1) サブコマンドを使用して、構成の JNDI エントリを一覧表示します。

例 20–7 JNDI エントリの一覧表示

この例では、ネームサービスの JNDI エントリをすべて表示します。

```
asadmin> list-jndi-entries
jndi_entry03
jndi_entry72
jndi_entry76
Command list-jndi-resources executed successfully
```

参照 コマンド行に asadmin help list-jndi-entries と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ 外部 JNDI リソースを更新する

1. list-jndi-resources(1) サブコマンドを使用して、既存の JNDI リソースを一覧表示します。

2. set(1) サブコマンドを使用して、外部 JNDI リソースを変更します。

例 20–8 外部 JNDI リソースの更新

この例では、外部リソースを変更します。

```
asadmin> set server.resources.external-jndi-resource.my-jndi-resource.
jndi-lookup-name=bar server.resources.external-jndi-resource.my-jndi-resource.jndi-lookup-name=bar
```

▼ 外部 JNDI リソースを削除する

JNDI リソースを削除するには、リモートモードで delete-jndi-resource サブコマンドを使用します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. delete-jndi-resource(1) サブコマンドを使用して、外部 JNDI エントリを削除します。

例 20–9 外部 JNDI リソースの削除

この例では、外部 JNDI リソースを削除します。

```
asadmin> delete-jndi-resource jndi_resource2
Command delete-jndi-resource executed successfully.
```

参照 コマンド行に `asadmin help delete-jndi-resource` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

外部 JNDI リソースの使用例

```
<resources>
 <!-- external-jndi-resource element specifies how to access J2EE resources
 -- stored in an external JNDI repository. This example
 -- illustrates how to access a java object stored in LDAP.
 -- factory-class element specifies the JNDI InitialContext factory that
 -- needs to be used to access the resource factory. property element
 -- corresponds to the environment applicable to the external JNDI context
 -- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
 -- designated (in this case the java) object.
 -->
  <external-jndi-resource jndi-name="test/myBean"
      jndi-lookup-name="cn=myBean"
      res-type="test.myBean"
      factory-class="com.sun.jndi.ldap.LdapCtxFactory">
    <property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
    <property name="SECURITY_AUTHENTICATION" value="simple" />
    <property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
    <property name="SECURITY_CREDENTIALS" value="changeit" />
  </external-jndi-resource>
</resources>
```

◆ ◆ ◆ 第 21 章

トランザクションの管理

この章では、asadmin コマンド行ユーティリティーを使用して、Sun GlassFish™ Enterprise Server 環境でトランザクションサービスを管理する方法について説明します。トランザクションを手動で回復する手順についても説明します。

ここでは、次のテーマを取り上げます。

本章で説明するタスクを 管理コンソール から実行する手順については、管理コンソール オンラインヘルプを参照してください。トランザクションサービス、トランザクションログ、および分散トランザクションの回復の設定については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 15 章「Using the Transaction Service」を参照してください。

トランザクションについて

「トランザクション」は、アプリケーション内の独立したアクションをひと続きにまとめたもので、一連のアクションのすべてが正常に完了する必要があります。トランザクションでは、1 つ以上のアクションをそれ以上分割不可能な作業単位にまとめることで、データの完全性と整合性を保証しています。すべてのアクションが完了しなかった場合、変更はロールバックされます。

たとえば、当座預金口座から普通預金口座に資金を移動する場合は、一般的に次のような手順が発生します。

1. 当座預金口座にその移動をカバーするだけの金額があるかどうかを確認します。
2. その金額を当座預金口座の借方に記帳します。
3. その金額を普通預金口座の貸方に記帳します。
4. その移動を当座預金口座ログに記録します。

5. その移動を普通預金口座ログに記録します。

これらの手順をまとめて、1 つのトランザクションと見なします。

すべての手順が正常に完了すれば、トランザクションは「コミット」されます。いずれかの手順が失敗した場合は、それ以前の手順で行われた変更がすべてロールバックされ、当座預金口座と普通預金口座はトランザクションが開始される前の状態に戻されます。この種類のイベントは「ロールバック」と呼ばれます。通常のトランザクションは、コミットされた状態かロールバックされた状態で終了します。

次の要素は、さまざまな API と機能を実装することで、信頼性の高いトランザクション処理に貢献します。

- 「トランザクションマネージャー」。トランザクション境界、トランザクションリソース管理、同期化、およびトランザクションコンテキスト伝達のサポートに必要なサービスと管理機能を提供します。
- 「**Enterprise Server**」。トランザクション状態管理を含むアプリケーション実行環境のサポートに必要なインフラストラクチャーを提供します。
- 「リソースマネージャー」。リソースアダプタを通して、アプリケーションがリソースにアクセスできるようにします。リソースマネージャーは、トランザクションマネージャーがトランザクションの関連付け、トランザクションの完了、および回復作業の伝達に使用するトランザクションリソースインタフェースを実装することで、分散トランザクションに参加します。このようなリソースマネージャーの例としては、リレーショナルデータベースサーバーがあります。
- 「リソースアダプタ」。Enterprise Server またはクライアントがリソースマネージャーへの接続に使用する、システムレベルのソフトウェアライブラリです。通常、リソースアダプタはリソースマネージャーに固有です。リソースアダプタはライブラリとして使用可能で、クライアントのアドレス空間内で使用されます。リソースアダプタの例として、Java™ Database Connectivity (JDBC) ドライバがあります。サポートされている JDBC ドライバについては、256 ページの「JDBC ドライバに固有の構成」を参照してください。
- 「トランザクションユーザーアプリケーション」。Enterprise Server 環境では、トランザクションユーザーアプリケーションは Java Naming and Directory Interface (JNDI) を使用して、トランザクションデータソースおよびオプションのユーザートランザクションを検索します。アプリケーションは、Enterprise Bean の宣言的なトランザクション属性設定や、プログラムによる明示的なトランザクション境界設定を使用する場合があります。

トランザクションサービスの管理

この節で説明する asadmin サブコマンドを使用して、単一のトランザクションをロールバックできます。ロールバックを行うには、アクティブなトランザクションを表示して、ロールバックが必要なトランザクションを正しく識別できるように、トランザクションサービスを停止する必要があります。また、これらの操作のあと、トランザクションサービスを再開する必要があります。

トランザクションサービスの設定と自動回復の設定の手順については、『Sun GlassFish Enterprise Server v3 Application Development Guide』の第 15 章「Using the Transaction Service」を参照してください。

ここでは、次のテーマを取り上げます。

- 345 ページの「トランザクションサービスを停止する」
- 346 ページの「トランザクションをロールバックする」
- 346 ページの「トランザクションサービスを再開する」

▼ トランザクションサービスを停止する

トランザクションサービスを停止するには、リモートモードで freeze-transaction-service サブコマンドを使用します。トランザクションサービスを停止すると、実行中のすべてのトランザクションがただちに中断されます。実行中のトランザクションをロールバックする前に、トランザクションサービスを停止する必要があります。

停止されたトランザクションサブシステムでこのサブコマンドを実行しても、影響はありません。トランザクションサービスは、unfreeze-transaction-service サブコマンドを使用して再開するまで、中断されたままになります。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. freeze-transaction-service(1) サブコマンドを使用して、トランザクションサービスを停止します。

例 21–1 トランザクションサービスの停止

この例では、トランザクションサービスを停止します。

```
asadmin> freeze-transaction-service
Command freeze-transaction-service executed successfully
```

参照 コマンド行に asadmin help freeze-transaction-service と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ トランザクションをロールバックする

状況に応じて、特定のトランザクションをロールバックできます。トランザクションをロールバックする前に、トランザクション処理を中断するためにトランザクションサービスを停止する必要があります。特定のトランザクションをロールバックするには、リモートモードで `rollback-transaction` サブコマンドを使用します。

始める前に　実行中のトランザクションをロールバックする前に、トランザクションサービスを停止します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2. ロールバックするトランザクションの **ID** を確認します。
   アクティブなトランザクションの ID を一覧表示するには、`get` サブコマンドを使用して `activeids` 統計の監視データを取得します。168 ページの「トランザクションサービスの統計」を参照してください。

3. `rollback-transaction`**(1)** サブコマンドを使用して、トランザクションをロールバックします。

例 21–2 トランザクションのロールバック

この例では、トランザクション ID が `0000000000000001_00` のトランザクションをロールバックします。

```
asadmin> rollback-transaction 0000000000000001_00
Command rollback-transaction executed successfully
```

参照　コマンド行に `asadmin help rollback-transaction` と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

▼ トランザクションサービスを再開する

中断されている実行中のトランザクションをすべて再開するには、リモートモードで `unfreeze-transaction-service` サブコマンドを使用します。トランザクションサービスを凍結したあとは、このサブコマンドを実行してサービスを再開します。

1. サーバーが実行されていることを確認します。
   リモートサブコマンドには、実行中のサーバーが必要です。

2 unfreeze-transaction-service(1) サブコマンドを使用して、中断されているトランザクションサービスを再開します。

例 21–3 トランザクションサービスの再開

この例では、凍結されているトランザクションサービスを再開します。

```
asadmin> unfreeze-transaction-service
Command unfreeze-transaction-service executed successfully
```

参照 コマンド行に asadmin help unfreeze-transaction-service と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

トランザクションの回復

一般的にサーバーまたはリソースマネージャーのクラッシュが原因で、コミットまたはロールバックの処理が中断される場合があります。クラッシュが発生すると、一部のトランザクションがステップの間に取り残される可能性があります。Enterprise Server は、これらの障害を回復し、サーバーの起動時にそのトランザクションを完了するように設計されています。失敗したトランザクションが複数のサーバーにわたっている場合は、トランザクションを開始したサーバーがトランザクションの結果を取得しようとしてほかのサーバーに問い合わせる場合があります。ほかのサーバーにアクセスできない場合、トランザクションは特殊な結果判別の情報を使用して、その結果を判別します。トランザクションはサーバーの起動時に解決されます。

クラッシュ以外の障害が発生している場合は、次の手順で手動回復を実行することができます。

▼ トランザクションを手動で回復する

サーバー上のリソースで障害が発生したときに保留中になったトランザクションを手動で回復するには、リモートモードで recover-transactions サブコマンドを使用します。サーバーがクラッシュしたあとに、手動のトランザクション回復を使用してトランザクションを回復することはできません。手動による操作は、リソースで予期しない障害が発生しても、まだサーバーが動作している状況での回復を目的としています。サーバーがクラッシュした場合、不確かなトランザクションを回復できるのは、完全な起動による回復処理だけです。

1 サーバーが実行されていることを確認します。
リモートサブコマンドには、実行中のサーバーが必要です。

2 recover-transactions(1) サブコマンドを使用して、トランザクションを手動で回復します。

例 21–4 手動によるトランザクションの回復

この例では、sampleserver でトランザクションの手動回復を実行します。

```
asadmin recover-transactions sampleserver
Transaction recovered.
```

参照 コマンド行に asadmin help recover-transactions と入力して、このサブコマンドの完全な構文とオプションを確認することもできます。

パート IV

付録

◆ ◆ ◆ 付 録 A

asadmin ユーティリティーのサブコマンド

この付録では、本リリースの Sun GlassFish™ Enterprise Server v3 ソフトウェアで提供される asadmin サブコマンドの一覧表を記載します。

asadmin のアプリケーション配備サブコマンドの説明と使い方については、『Sun GlassFish Enterprise Server v3 Application Deployment Guide』を参照してください。

asadmin サブコマンドのオンラインヘルプは、asadmin create-domain --help などのコマンド行で呼び出すことができます。『Sun GlassFish Enterprise Server v3 Reference Manual』にも、マニュアルページを集めたものが用意されています。

注 – リモートサブコマンドで使用する共通オプションは、asadmin(1M)マニュアルページに記載されています。

一般的な管理サブコマンド

add-resources(1)	指定した XML ファイル内に指定したリソースを作成します。リモートモードでのみサポートされています。手順については、本ガイドの59 ページの「XML ファイルからリソースを追加する」を参照してください。
asadmin(1M)	asadmin ユーティリティーの仕組みについて説明します。
create-service(1)	無人の自動起動によってドメイン管理サーバー (DAS) が起動されるように設定します。このサブコマンドは、Solaris 10 では Service Management Facility (SMF) を使用します。手順については、本ガイドの96 ページの「Solaris 10 でのドメイン自動的再起動」を参照してください。
create-system-properties(1)	システムプロパティーを作成または更新します。リモートモードでのみサポートされています。手順については、本ガイドの57 ページの「システムプロパティーを作成する」を参照してください。
delete-system-property(1)	ドメイン、構成、またはサーバーインスタンスのシステムプロパティーを削除します。リモートモードでのみサポートされています。手順については、本ガイドの58 ページの「システムプロパティーを削除する」を参照してください。
get(1)	domain.xml ファイル内の要素の属性を取得します。-m オプションを使用すると、監視可能な属性または設定可能な属性の名前と値を取得します。手順については、本ガイドの143 ページの「list および get サブコマンドを監視に使用する場合のガイドライン」を参照してください。
list(1)	設定可能な要素を一覧表示します。Solaris で、* をオプション値やオペランドとして使用してサブコマンドを実行する場合は、引用符が必要です。手順については、本ガイドの143 ページの「list および get サブコマンドを監視に使用する場合のガイドライン」を参照してください。
list-commands(1)	最初にローカルサブコマンド、次にリモートサブコマンドの順に、asadmin サブコマンドのすべてを一覧表示します。リモートサブコマンドだけ、またはローカルサブコマンドだけが表示されるように指定

	することもできます。リモートモードでのみサポートされています。手順については、本ガイドの63 ページの「サブコマンドを一覧表示する」を参照してください。
list-containers(1)	アプリケーションコンテナと、各コンテナのステータスを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの61 ページの「コンテナを一覧表示する」を参照してください。
list-modules(1)	Enterprise Server サブシステムにアクセス可能なモジュールを一覧表示します。各モジュールのステータスが記載されます。リモートモードでのみサポートされています。手順については、本ガイドの62 ページの「モジュールを一覧表示する」を参照してください。
list-system-properties(1)	ドメインまたは構成のシステムプロパティーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの58 ページの「システムプロパティーを一覧表示する」を参照してください。
list-timers(1)	特定のサーバーインスタンスに備えられたタイマーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの64 ページの「タイマーを一覧表示する」を参照してください。
multimode(1)	オプション設定と環境設定を残したままマルチサブコマンドを実行するための asadmin> プロンプトを表示します。ローカルモードでのみサポートされています。手順については、49 ページの「asadmin ユーティリティーの使用」を参照してください。
set(1)	1 つ以上の設定可能な属性の値を設定します。手順については、本ガイドの137 ページの「監視の設定」を参照してください。
show-component-status(1)	既存コンポーネントのステータスを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの64 ページの「コンポーネントの状態を表示する」を参照してください。

start-database(1)	Java DB サーバーを起動します。このコマンドは Enterprise Server に配備されたアプリケーションの操作に対してのみ使用します。手順については、本ガイドの245 ページの「データベースを起動する」を参照してください。
stop-database(1)	Java DB データベースサーバーのプロセスを停止します。手順については、本ガイドの246 ページの「データベースを停止する」を参照してください。
version(1)	アーカイブまたはフォルダフォーマットに指定されたオプションのバージョン情報を表示します。リモートモードでのみサポートされています。手順については、本ガイドの60 ページの「Enterprise Server のバージョンを表示する」を参照してください。

接続サブコマンド

create-admin-object(1)	管理対象オブジェクトを作成します。手順については、本ガイドの285 ページの「管理対象オブジェクトリソースを作成する」を参照してください。
create-connector-connection-pool(1)	指定した接続プール名で新しいコネクタ接続プールを追加します。手順については、本ガイドの269 ページの「コネクタ接続プールを作成する」を参照してください。
create-connector-resource(1)	コネクタリソースを作成します。手順については、本ガイドの273 ページの「コネクタリソースを作成する」を参照してください。
create-connector-security-map(1)	指定したコネクタ接続プールのコネクタセキュリティーマップを作成します。手順については、278 ページの「コネクタセキュリティーマップを作成する」を参照してください。
create-connector-work-security-map(1)	指定したリソースアダプタのコネクタ作業セキュリティーマップを作成 します。リモートモードでのみサポートされています。手順については、本ガイドの

	282 ページの「コネクタ作業セキュリティーマップを作成する」を参照してください。
create-jdbc-resource(1)	JDBC リソースを新規作成します。リモートモードでのみサポートされています。手順については、本ガイドの253 ページの「JDBC リソースを作成する」を参照してください。
create-jdbc-connection-pool(1)	新しい JDBC 接続プールを、指定した JDBC 接続プール名で登録します。リモートモードでのみサポートされています。手順については、本ガイドの248 ページの「JDBC 接続プールを作成する」を参照してください。
create-resource-adapter-config(1)	コネクタモジュールの構成情報を作成します。リモートモードでのみサポートされています。手順については、本ガイドの276 ページの「リソースアダプタの構成情報を作成する」を参照してください。
delete-admin-object(1)	管理対象オブジェクトを削除します。手順については、本ガイドの287 ページの「管理対象オブジェクトリソースを削除する」を参照してください。
delete-connector-connection-pool(1)	オペランド connector_connection_pool_name を使用して指定したコネクタ接続プールを削除します。手順については、本ガイドの272 ページの「コネクタ接続プールを削除する」を参照してください。
delete-connector-resource(1)	コネクタリソースを削除します。手順については、本ガイドの275 ページの「コネクタリソースを削除する」を参照してください。
delete-connector-security-map(1)	指定したコネクタセキュリティーマップを削除します。リモートモードでのみサポートされています。手順については、本ガイドの281 ページの「コネクタセキュリティーマップを削除する」を参照してください。

delete-connector-work-security-map(1)	指定したコネクタ作業セキュリティーマップを削除します。リモートモードでのみサポートされています。手順については、本ガイドの284 ページの「コネクタ作業セキュリティーマップを削除する」を参照してください。
delete-jdbc-connection-pool(1)	指定した JDBC 接続プールを削除します。リモートモードでのみサポートされています。手順については、本ガイドの252 ページの「JDBC 接続プールを削除する」を参照してください。
delete-jdbc-resource(1)	JDBC リソースを削除します。指定した JNDI 名が、削除されるリソースになります。リモートモードでのみサポートされています。手順については、本ガイドの255 ページの「JDBC リソースを削除する」を参照してください。
delete-resource-adapter-config(1)	コネクタモジュールの構成情報を削除します。リモートモードでのみサポートされています。手順については、本ガイドの277 ページの「リソースアダプタ構成を削除する」を参照してください。
flush-connection-pool(1)	指定した接続内に確立した接続をすべて再初期化します。手順については、本ガイドの251 ページの「接続プールをリセット (フラッシュ) する」を参照してください。
list-admin-objects(1)	管理対象オブジェクトを一覧表示します。手順については、本ガイドの286 ページの「管理対象オブジェクトを一覧表示する」を参照してください。
list-connector-connection-pools(1)	作成済みのコネクタ接続プールを一覧表示します。手順については、本ガイドの271 ページの「コネクタ接続プールを一覧表示する」を参照してください。
list-connector-resources(1)	コネクタリソースを作成します。手順については、本ガイドの274 ページの「コネクタリソースを一覧表示する」を参照してください。

list-connector-security-maps(1)	指定したコネクタ接続プールに属するコネクタセキュリティーマップを一覧表示します。手順については、本ガイドの279 ページの「コネクタセキュリティーマップの一覧表示」を参照してください。
list-connector-work-security-maps(1)	リソースアダプタの既存のコネクタ作業セキュリティーマップを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの283 ページの「コネクタ作業セキュリティーマップを一覧表示する」を参照してください。
list-jdbc-connection-pools(1)	既存の JDBC 接続プールを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの250 ページの「JDBC 接続プールを一覧表示する」を参照してください。
list-jdbc-resources(1)	既存の JDBC リソースを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの254 ページの「JDBC リソースを一覧表示する」を参照してください。
list-resource-adapter-configs(1)	コネクタモジュールの構成情報を一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの276 ページの「リソースアダプタ構成の一覧表示」を参照してください。
ping-connection-pool(1)	JDBC 接続プールが使用可能かどうかをテストします。リモートモードでのみサポートされています。手順については、本ガイドの250 ページの「接続プールと通信する (ping を実行する)」を参照してください。
update-connector-security-map(1)	指定したコネクタ接続プールのセキュリティーマップを変更します。手順については、263 ページの「コネクタセキュリティーマップを更新する」を参照してください。

update-connector-work-security-map(1)	指定したリソースアダプタ (コネクタモジュール) に属する作業セキュリティーマップを変更します。手順については、本ガイドの284 ページの「コネクタ作業セキュリティーマップを更新する」を参照してください。

ドメインサブコマンド

create-domain(1)	ドメインの構成を作成します。ドメインは、1 つずつ独立して存在しています。所定のホストの asadmin ユーティリティーに対してアクセス権を持つユーザーは、ドメインを作成し、自分の選択する場所にその構成を格納することができます。手順については、本ガイドの88 ページの「ドメインの作成」を参照してください。
delete-domain(1)	指定したドメインを削除します。削除する前に、ドメインがすでに停止している必要があります。手順については、本ガイドの92 ページの「ドメインの削除」を参照してください。
list-domains(1)	既存のドメインとそのステータスを一覧表示します。ドメインのディレクトリが指定されていない場合は、デフォルトの *as-install*/domains ディレクトリにあるドメインが表示されます。手順については、本ガイドの90 ページの「ドメインの一覧表示」を参照してください。
login(1)	ユーザーをドメインにログインさせます。手順については、本ガイドの90 ページの「ドメインへのログイン」を参照してください。
restart-domain(1)	指定したドメインのドメイン管理サーバー (DAS) を再起動します。リモートモードでのみサポートされています。手順については、本ガイドの94 ページの「ドメインの再起動」を参照してください。
start-domain(1)	ドメインを起動します。ドメインのディレクトリが指定されていない場合は、デフォルトの *as-install*/domains ディレクトリにある domain1 が起動します。複数のドメインが存在する場合、*domain_name* オペランドを指定する必要があります。手順については、本ガイドの93 ページの「ドメインの起動」を参照してください。

stop-domain(1)	指定したドメインのドメイン管理サーバー (DAS) を停止します。リモートモードでのみサポートされています。手順については、本ガイドの94 ページの「ドメインの停止」を参照してください。
uptime(1)	ドメイン管理サーバー (DAS) を最後に再起動してから今までの稼働時間を表示します。リモートモードでのみサポートされています。手順については、本ガイドの98 ページの「ドメインの稼働時間の表示」を参照してください。

インターネット接続サブコマンド

create-http(1)	プロトコルの HTTP パラメータ一式を作成し、1 つ以上のネットワークリスナーを設定します。リモートモードでのみサポートされています。手順については、本ガイドの295 ページの「HTTP 構成を作成する」を参照してください。
create-http-listener(1)	新しい HTTP 待機ソケットを作成します。リモートモードでのみサポートされています。手順については、本ガイドの292 ページの「インターネット接続を作成する」を参照してください。
create-network-listener(1)	新しい HTTP 待機ソケットを作成します。リモートモードでのみサポートされています。手順については、本ガイドの292 ページの「インターネット接続を作成する」を参照してください。
create-protocol(1)	リスナーのプロトコルを作成します。リモートモードでのみサポートされています。手順については、本ガイドの293 ページの「プロトコルを作成する」を参照してください。
create-transport(1)	リスナーのトランスポートを作成します。リモートモードでのみサポートされています。手順については、本ガイドの296 ページの「トランスポートを作成する」を参照してください。
create-virtual-server(1)	指定した仮想サーバー要素を作成します。リモートモードでのみサポートされています。手順については、本ガイドの303 ページの「仮想サーバーを作成する」を参照してください。
create-ssl(1)	選択した HTTP リスナー内で SSL 要素を作成および設定し、そのリスナーまたはサービス上でセキュリティー保護された通信ができるようにします。リ

	モートモードでのみサポートされています。手順については、本ガイドの300 ページの「SSL の HTTP リスナーを構成する」を参照してください。
delete-http(1)	既存の HTTP 構成を削除します。リモートモードでのみサポートされています。手順については、本ガイドの295 ページの「HTTP 構成を削除する」を参照してください。
delete-http-listener(1)	指定した HTTP リスナーを削除します。リモートモードでのみサポートされています。手順については、本ガイドの300 ページの「HTTP ネットワークリスナーを削除する」を参照してください。
delete-network-listener(1)	指定した HTTP リスナーを削除します。リモートモードでのみサポートされています。手順については、本ガイドの300 ページの「HTTP ネットワークリスナーを削除する」を参照してください。
delete-protocol(1)	既存の HTTP プロトコルを削除します。リモートモードでのみサポートされています。手順については、本ガイドの294 ページの「プロトコルを削除する」を参照してください。
delete-ssl(1)	選択した HTTP リスナー内の SSL 要素を削除します。リモートモードでのみサポートされています。手順については、本ガイドの301 ページの「HTTP リスナーから SSL を削除する」を参照してください。
delete-transport(1)	既存の HTTP トランスポートを削除します。リモートモードでのみサポートされています。手順については、本ガイドの297 ページの「トランスポートを削除する」を参照してください。
delete-virtual-server(1)	指定した仮想サーバー要素を削除します。リモートモードでのみサポートされています。手順については、本ガイドの304 ページの「仮想サーバーを削除する」を参照してください。
list-http-listeners(1)	既存の HTTP リスナーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの299 ページの「HTTP ネットワークリスナーを一覧表示する」を参照してください。
list-network-listeners(1)	既存の HTTP リスナーを一覧表示します。リモートモードでのみサポートされています。手順について

	は、本ガイドの299 ページの「HTTP ネットワークリスナーを一覧表示する」を参照してください。
list-protocols(1)	既存の HTTP プロトコルを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの293 ページの「プロトコルを一覧表示する」を参照してください。
list-transports(1)	既存の HTTP トランスポートを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの296 ページの「トランスポートを一覧表示する」を参照してください。
list-virtual-servers(1)	既存の仮想サーバーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの304 ページの「仮想サーバーを一覧表示する」を参照してください。

JavaMail サブコマンド

create-javamail-resource(1)	JavaMail セッションリソースを作成します。リモートモードでのみサポートされています。手順については、本ガイドの312 ページの「JavaMail リソースを作成する」を参照してください。
delete-javamail-resource(1)	JavaMail セッションリソースを削除します。リモートモードでのみサポートされています。手順については、本ガイドの314 ページの「JavaMail リソースを削除する」を参照してください。
list-javamail-resources(1)	JavaMail セッションリソースを作成します。リモートモードでのみサポートされています。手順については、本ガイドの313 ページの「JavaMail リソースを一覧表示する」を参照してください。

JMS サブコマンド

create-jmsdest(1)	JMS 物理送信先を作成します。物理送信先とともに、create-jms-resource サブコマンドを使用して、物理送信先を指定する Name プロパティーを持つ JMS 送信先リソースを作成します。リモートモードでのみサポートされています。手順については、本ガイドの320 ページの「JMS 物理送信先を作成する」を参照してください。

create-jms-host(1)	JMS サービス内の JMS ホストを作成します。リモートモードでのみサポートされています。手順については、本ガイドの326 ページの「JMS ホストを作成する」を参照してください。
create-jms-resource(1)	JMS 接続ファクトリリソースまたは JMS 送信先リソースを作成します。リモートモードでのみサポートされています。リモートモードでのみサポートされています。手順については、本ガイドの323 ページの「接続ファクトリまたは送信先リソースを作成する」を参照してください。
delete-jmsdest(1)	指定した JMS 送信先を削除します。リモートモードでのみサポートされています。手順については、本ガイドの322 ページの「JMS 物理送信先を削除する」を参照してください。
delete-jms-host(1)	JMS サービス内の JMS ホストを削除します。リモートモードでのみサポートされています。手順については、本ガイドの328 ページの「JMS ホストを削除する」を参照してください。
delete-jms-resource(1)	JMS 接続ファクトリリソースまたは JMS 送信先リソースを削除します。リモートモードでのみサポートされています。手順については、本ガイドの325 ページの「接続ファクトリまたは送信先リソースを削除する」を参照してください。
flush-jmsdest(1)	指定したターゲットの指定 JMS サービス構成にある物理送信先から、メッセージをパージします。リモートモードでのみサポートされています。手順については、本ガイドの321 ページの「物理送信先からメッセージを消去する」を参照してください。
jms-ping(1)	JMS サービス (JMS プロバイダとも呼ばれる) が起動して稼働中かどうかを確認します。リモートモードでのみサポートされています。手順については、本ガイドの「JMS に関するトラブルシューティング」を参照してください。
list-jmsdest(1)	JMS 物理送信先を一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの320 ページの「JMS 物理送信先を一覧表示する」を参照してください。

list-jms-hosts(1)	既存の JMS ホストを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの327 ページの「JMS ホストを一覧表示する」を参照してください。
list-jms-resources(1)	既存の JMS 接続ファクトリまたは送信先リソースを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの324 ページの「JMS リソースを一覧表示する」を参照してください。

JNDI サブコマンド

create-custom-resource(1)	カスタム JNDI リソースを作成します。リモートモードでのみサポートされています。手順については、本ガイドの336 ページの「カスタム JNDI リソースを作成する」を参照してください。
create-jndi-resource(1)	外部 JNDI リソースを作成します。リモートモードでのみサポートされています。手順については、本ガイドの338 ページの「外部 JNDI リソースを登録する」を参照してください。
delete-custom-resource(1)	カスタム JNDI リソースを削除します。リモートモードでのみサポートされています。手順については、本ガイドの337 ページの「カスタム JNDI リソースを削除する」を参照してください。
delete-jndi-resource(1)	外部 JNDI リソースを削除します。リモートモードでのみサポートされています。手順については、本ガイドの340 ページの「外部 JNDI リソースを削除する」を参照してください。
list-custom-resources(1)	既存のカスタム JNDI リソースを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの336 ページの「カスタム JNDI リソースを一覧表示する」を参照してください。
list-jndi-entries(1)	JNDI ツリー内のエントリを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの339 ページの「外部 JNDI エントリを一覧表示する」を参照してください。
list-jndi-resources(1)	既存の外部 JNDI リソースを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの339 ページの「外部 JNDI リソースを一覧表示する」を参照してください。

JVM サブコマンド

create-jvm-options(1)	Java 構成または domain.xml ファイルのプロファイラ要素に、JVM オプションを作成します。リモートモードでのみサポートされています。手順については、本ガイドの102 ページの「JVM オプションを作成する」を参照してください。
create-profiler(1)	プロファイラ要素を作成します。リモートモードでのみサポートされています。手順については、本ガイドの105 ページの「プロファイラを作成する」を参照してください。
delete-jvm-options(1)	Java 構成または domain.xml ファイルのプロファイラ要素から、指定した JVM オプションを削除します。リモートモードでのみサポートされています。手順については、本ガイドの103 ページの「JVM オプションを削除する」を参照してください。
delete-profiler(1)	指定したプロファイラ要素を削除します。リモートモードでのみサポートされています。手順については、本ガイドの106 ページの「プロファイラを削除する」を参照してください。
generate-jvm-report(1)	Enterprise Server を実行する仮想マシンのスレッド、クラス、およびメモリーを表示したレポートを生成します。手順については、本ガイドの104 ページの「JVM レポートを生成する」を参照してください。
list-jvm-options(1)	Enterprise Server 起動時に、Java アプリケーションランチャーに渡されるコマンド行オプションを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの102 ページの「JVM オプションを一覧表示する」を参照してください。

ライフサイクルモジュール用サブコマンド

create-lifecycle-module(1)	ライフサイクルモジュールを新規作成します。リモートモードでのみサポートされています。手順については、本ガイドの176 ページの「ライフサイクルモジュールを作成する」を参照してください。
list-lifecycle-modules(1)	ライフサイクルモジュールを一覧表示します。リモートモードでのみサポートされています。手順に

ついては、本ガイドの177 ページの「ライフサイクルモジュールを一覧表示する」を参照してください。

delete-lifecycle-module(1)	既存のライフサイクルモジュールを削除します。リモートモードでのみサポートされています。手順については、本ガイドの178 ページの「ライフサイクルモジュールを削除する」を参照してください。

ログ作成および監視用サブコマンド

disable-monitoring(1)	監視サービスを無効にします。リモートモードでのみサポートされています。手順については、本ガイドの139 ページの「監視を無効にする」を参照してください。
enable-monitoring(1)	監視サービスを有効にします。リモートモードでのみサポートされています。手順については、本ガイドの138 ページの「監視を有効にする」を参照してください。
monitor(1)	共通な Enterprise Server リソースの監視情報を表示します。リモートモードでのみサポートされています。手順については、本ガイドの140 ページの「共通の監視データを表示する」を参照してください。
list-logger-levels(1)	既存のロガーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの126 ページの「モジュールのロガーレベルを設定する」を参照してください。
rotate-log(1)	server.logファイルをローテーションして、タイムスタンプ付きファイルに保存します。リモートモードでのみサポートされています。手順については、本ガイドの127 ページの「ログファイルを手動でローテーションする」を参照してください。
set-log-level(1)	モジュールのログレベルを設定します。リモートモードでのみサポートされています。手順については、本ガイドの126 ページの「モジュールのロガーレベルを設定する」を参照してください。

ORB サブコマンド

create-iiop-listener(1)	IIOP リスナーを作成します。リモートモードでのみサポートされています。手順については、本ガイドの308 ページの「IIOP リスナーを作成する」を参照してください。
delete-iiop-listener(1)	IIOP リスナーを削除します。リモートモードでのみサポートされています。手順については、本ガイドの310 ページの「IIOP リスナーを削除する」を参照してください。
list-iiop-listeners(1)	既存の IIOP リスナーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの309 ページの「IIOP リスナーを一覧表示する」を参照してください。

セキュリティー用サブコマンド

change-admin-password(1)	管理パスワードを変更します。旧パスワードと新 admin パスワードを入力し、確定するよう求められます。手順については、本ガイドの203 ページの「管理パスワードを変更する」を参照してください。
change-master-password(1)	管理マスターパスワードを変更します。このパスワードは、プライマリキーの保存先となるセキュリティー保護された保存先にアクセスする際に使用されます。サーバーが停止していないかぎり、このサブコマンドは機能しません。手順については、本ガイドの202 ページの「マスターパスワードを変更する」を参照してください。
configure-ldap-for-admin(1)	所定の LDAP 内に、認証レルム名付き admin-realm を設定します。リモートモードでのみサポートされています。
create-audit-module(1)	監査機能を実装するプラグイン用に、指定した監査モジュールを追加します。リモートモードでのみサポートされています。手順については、本ガイドの

	208 ページの「監査モジュールを作成する」を参照してください。
create-message-security-provider(1)	指定されたメッセージ層の provider-config サブ要素 (パラメータとプロパティーを指定するファイルである、domain.xml の message-security-config 要素) を作成します。リモートモードでのみサポートされています。手順については、本ガイドの 237 ページの「メッセージセキュリティープロバイダを作成する」を参照してください。
create-password-alias(1)	パスワードのエイリアスを作成します。エイリアスはdomain.xml ファイル内に平文として保存されません。リモートモードでのみサポートされています。手順については、本ガイドの205 ページの「パスワードエイリアスを作成する」を参照してください。
delete-audit-module(1)	指定した監査モジュールを削除します。リモートモードでのみサポートされています。手順については、本ガイドの 210 ページの「監査モジュールを削除する」を参照してください。
delete-message-security-provider(1)	指定したメッセージ層の provider-config サブ要素を削除します。リモートモードでのみサポートされています。手順については、本ガイドの239 ページの「メッセージセキュリティープロバイダを削除する」を参照してください。
delete-password-alias(1)	パスワードのエイリアスを削除します。リモートモードでのみサポートされています。手順については、本ガイドの 207 ページの「パスワードエイリアスを削除する」を参照してください。
list-audit-modules(1)	すべての監査モジュールを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの 209 ページの「監査モジュールを一覧表示する」を参照してください。

list-message-security-providers(1)	指定したメッセージ層のセキュリティーメッセージプロバイダを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの238 ページの「メッセージセキュリティープロバイダを一覧表示する」を参照してください。
list-password-aliases(1)	既存パスワードのエイリアスを一覧表示します。リモートモードでのみサポートされています。手順については、206 ページの「パスワードエイリアスを一覧表示する」を参照してください。
update-password-alias(1)	指定したパスワードのエイリアスを変更します。リモートモードでのみサポートされています。手順については、207 ページの「パスワードエイリアスを更新する」を参照してください。

スレッドプール用サブコマンド

create-threadpool(1)	スレッドプールを新規作成します。リモートモードでのみサポートされています。手順については、本ガイドの108 ページの「スレッドプールを作成する」を参照してください。
delete-threadpool(1)	指定したスレッドプールを削除します。リモートモードでのみサポートされています。手順については、本ガイドの110 ページの「スレッドプールを削除する」を参照してください。
list-threadpools(1)	既存のスレッドプールを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの109 ページの「スレッドプールを一覧表示する」を参照してください。

トランザクションサービス用サブコマンド

freeze-transaction-service(1)	実行中のすべてのトランザクションが中断している間、トランザクションサブシステムを凍結します。リモートモードでのみサポートされて

	います。手順については、345 ページの「トランザクションサービスを停止する」を参照してください。
recover-transactions(1)	保留中のトランザクションを手動で回復します。リモートモードでのみサポートされています。手順については、347 ページの「トランザクションを手動で回復する」を参照してください。
rollback-transaction(1)	指定したトランザクションをロールバックします。リモートモードでのみサポートされています。手順については、346 ページの「トランザクションをロールバックする」を参照してください。
unfreeze-transaction-service(1)	中断していた実行中のすべてのトランザクションを再開します。このサブコマンドは、すでに凍結しているトランザクションに対して呼び出します。リモートモードでのみサポートされています。手順については、346 ページの「トランザクションサービスを再開する」を参照してください。

ユーザー管理用サブコマンド

create-auth-realm(1)	指定した認証レルムを追加します。リモートモードでのみサポートされています。手順については、本ガイドの 217 ページの「認証レルムを作成する」を参照してください。
create-file-user(1)	指定したファイルベースの認証レルムにファイルユーザーを作成します。エントリは、指定したユーザー名、パスワード、およびグループでキーファイルに追加されます。それぞれをコロン (:) で区切ることによって、複数のグループを作成することもできます。リモートモードでのみサポートされています。手順については、本ガイドの221 ページの「ファイルユーザーを作成する」を参照してください。
delete-auth-realm(1)	指定した認証レルムを削除します。リモートモードでのみサポートされています。手順については、本ガイドの 218 ページの「認証レルムを削除する」を参照してください。

delete-file-user(1)	キーファイルに指定したユーザーエントリを削除します。リモートモードでのみサポートされています。手順については、本ガイドの224 ページの「ファイルユーザーを削除する」を参照してください。
list-auth-realms(1)	既存の認証レルムを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの217 ページの「認証レルムを一覧表示する」を参照してください。
list-file-users(1)	file レルム認証メソッドで利用できるファイルユーザーを一覧表示します。リモートモードでのみサポートされています。手順については、本ガイドの222 ページの「ファイルユーザーを一覧表示する」を参照してください。
list-file-groups(1)	ファイルユーザーのグループを一覧表示します。--name オプションが指定されていない場合は、すべてのグループを表示します。手順については、本ガイドの222 ページの「ファイルグループを一覧表示する」を参照してください。
update-file-user(1)	指定したユーザー名、パスワード、およびグループを使用して、キーファイル内の既存のエントリを更新します。リモートモードでのみサポートされています。手順については、本ガイドの223 ページの「ファイルユーザーを更新する」を参照してください。

索引

D

E

F

G

H

I

J

K

L

S

T

U

V

W

X

ア

イ

エ

オ

カ

管

起

形

元

更

構

作