

Sun GlassFish Communications Server 2.0 管理ガイド



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

Part No: 821-1046-10
2009年9月

Sun Microsystems, Inc. (以下米国 Sun Microsystems 社とします) は、本書に記述されている製品に含まれる技術に関連する知的財産権を所有します。特に、この知的財産権はひとつかそれ以上の米国における特許、あるいは米国およびその他の国において申請中の特許を含んでいることがありますが、それらに限定されるものではありません。

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

この配布には、第三者によって開発された素材を含んでいることがあります。

本製品の一部は、カリフォルニア大学からライセンスされている Berkeley BSD システムに基づいていることがあります。UNIX は、X/Open Company, Ltd. が独占的にライセンスしている米国ならびに他の国における登録商標です。

un、Sun Microsystems、Sun のロゴマーク、Solaris のロゴマーク、Java Coffee Cup のロゴマーク、docs.sun.com、Java および Solaris は、米国およびその他の国における米国 Sun Microsystems 社の商標、登録商標もしくは、サービスマークです。すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

OPEN LOOK および SunTM Graphical User Interface は、米国 Sun Microsystems 社が自社のユーザおよびライセンス実施権者向けに開発しました。米国 Sun Microsystems 社は、コンピュータ産業用のビジュアルまたはグラフィカル・ユーザインタフェースの概念の研究開発における米国 Xerox 社の先駆者としての成果を認めるものです。米国 Sun Microsystems 社は米国 Xerox 社から Xerox Graphical User Interface の非独占的ライセンスを取得しており、このライセンスは、OPEN LOOK のグラフィカル・ユーザインタフェースを実装するか、またはその他の方法で米国 Sun Microsystems 社との書面によるライセンス契約を遵守する、米国 Sun Microsystems 社のライセンス実施権者にも適用されます。

本書で言及されている製品や含まれている情報は、米国輸出規制法で規制されるものであり、その他の国の輸出入に関する法律の対象となることがあります。核、ミサイル、化学あるいは生物兵器、原子力の海洋輸送手段への使用は、直接および間接を問わず厳しく禁止されています。米国が禁輸の対象としている国や、限定はされませんが、取引禁止顧客や特別指定国民のリストを含む米国輸出排除リストで指定されているものへの輸出および再輸出は厳しく禁止されています。

本書は、「現状のまま」をベースとして提供され、商品性、特定目的への適合性または第三者の権利の非侵害の黙示の保証を含みそれに限定されない、明示的であるか黙示的であるかを問わない、なんらの保証も行われぬものとします。

目次

はじめに	21
1 Communications Serverの概要	27
Communications Server の概要と概念	27
Communications Serverの概要	27
管理用ツール	28
Communications Server の概念	30
ドメイン	31
ドメイン管理サーバー (DAS)	31
プロファイル	32
クラスタ	33
ノードエージェント	33
サーバーインスタンス	34
Communications Server の基本コマンド	37
ドメインの作成	38
ドメインの削除	38
ドメインの一覧表示	39
ドメインの起動	39
Windows でのデフォルトドメインの起動	39
ドメインの停止	39
Windows でのデフォルトドメインの停止	40
ドメインの再起動	40
クラスタの作成	40
クラスタの起動	40
クラスタの停止	41
ノードエージェントの作成	41
ノードエージェントの起動	41
ノードエージェントの停止	42

インスタンスの起動	42
インスタンスの停止	42
インスタンスの再起動	42
ドメイン管理サーバーの再作成	43
2 Java Business Integration	47
JBI 環境	47
JBI コンポーネント	47
サービスアセンブリ	49
共用ライブラリ	50
JBI 記述子	50
3 JDBC リソース	51
JDBC リソース	51
JDBC 接続プール	52
JDBC リソースと接続プールの協調動作について	52
データベースアクセスの設定	53
JDBC 接続プールの操作	54
JDBC 接続プールの作成	54
JDBC 接続プールの編集	56
JDBC 接続プールの詳細属性の編集	59
各 JDBC ドライバの設定	61
Java DB Type 4 ドライバ	62
DB2 データベース用の Sun GlassFish JDBC ドライバ	63
Oracle 8.1.7 および 9.x データベース用の Sun GlassFish JDBC ドライバ	63
Microsoft SQL Server データベース用の Sun GlassFish JDBC ドライバ	64
Sybase データベース用の Sun GlassFish JDBC ドライバ	64
IBM DB2 8.1 Type 2 ドライバ	65
Sybase ASE 12.5 データベース用の JConnect Type 4 ドライバ	65
MySQL Type 4 Driver	66
Oracle 8.1.7 および 9.x データベース用の Inet Oraxo JDBC ドライバ	67
Microsoft SQL Server データベース用の Inet Merlia JDBC ドライバ	68
Sybase データベース用の Inet Sybelux JDBC ドライバ	68
Oracle 8.1.7 および 9.x 用の Oracle Thin Type 4 ドライバ	69
Oracle 8.1.7 および 9.x データベース用の OCI Oracle Type 2 ドライバ	70

IBM Informix Type 4 ドライバ	70
CloudScape 5.1 Type 4 ドライバ	71
4 Java Message Service (JMS) リソースの設定	73
JMS リソース	73
JMS リソースとコネクタリソースの関係	74
JMS 接続ファクトリ	75
JMS 送信先リソース	75
JMS 物理送信先	76
JMS プロバイダのプロパティの設定	76
リモートサーバーへのアクセス	77
外部 JMS プロバイダ	77
JMS の汎用リソースアダプタの設定	78
リソースアダプタのプロパティ	78
ManagedConnectionFactory プロパティ	82
管理対象オブジェクトリソースのプロパティ	83
有効化仕様プロパティ	84
5 JavaMail リソースの設定	87
JavaMail セッションの作成	87
6 JNDI リソース	91
Java EE ネームサービス	91
ネーミング参照とバインディング情報	92
カスタムリソースの使用	93
外部 JNDI リポジトリとリソースの使用	93
7 コネクタリソース	95
コネクタの概要	95
コネクタ接続プールの管理	96
▼コネクタ接続プールを作成する	96
▼コネクタ接続プールを編集する	97
▼コネクタ接続プールの詳細属性を編集する	100
▼コネクタ接続プールのプロパティを編集する	102

セキュリティマップの管理	102
▼コネクタ接続プールを削除する	103
▼EISアクセスをセットアップする	103
コネクタリソースの管理	103
▼コネクタリソースを作成する	103
▼コネクタリソースを編集する	104
▼コネクタリソースを削除する	105
▼コネクタサービスを設定する	105
管理対象オブジェクトリソースの管理	106
▼管理対象オブジェクトリソースを作成する	106
▼管理対象オブジェクトリソースを編集する	108
▼管理対象オブジェクトリソースを削除する	108
8 SIP、Web、およびEJB コンテナ	109
SIP Servlet コンテナ	109
SIP コンテナのプロパティの編集	110
Web コンテナ	111
EJB コンテナ	112
9 セキュリティーの設定	113
アプリケーションおよびシステムセキュリティについて	113
セキュリティ管理用ツール	114
パスワードのセキュリティ管理	115
domain.xml ファイル内のパスワードの暗号化	115
エンコード化されたパスワードを含むファイルの保護	116
マスターパスワードの変更	116
マスターパスワードとキーストアの操作	117
管理パスワードの変更	118
認証と承認について	118
エンティティーの認証	118
ユーザーの承認	119
JACC プロバイダの指定	120
認証および承認の決定の監査	120
メッセージセキュリティの設定	120
ユーザー、グループ、ロール、およびレルムについて	121

ユーザー	122
グループ	122
ロール	122
レルム	123
信頼設定およびエンティティ	125
信頼設定およびエンティティの作成	125
信頼設定およびエンティティの編集	126
証明書および SSL の概要	126
デジタル証明書について	126
SSL (Secure Sockets Layer) について	128
ファイアウォールについて	130
証明書ファイルについて	130
証明書ファイルの場所の変更	131
JSSE (Java Secure Socket Extension) ツールの使用	131
keytool ユーティリティの使用	132
keytool ユーティリティを使って証明書を生成する	133
keytool ユーティリティを使ってデジタル証明書に署名する	134
keytool ユーティリティを使って証明書を削除する	135
10 メッセージセキュリティの設定	137
メッセージセキュリティの概要	137
Communications Server のメッセージセキュリティの理解	138
メッセージセキュリティの責任の割り当て	138
セキュリティトークンとセキュリティメカニズムについて	139
メッセージセキュリティ用語の解説	141
Web サービスのセキュリティ保護	142
アプリケーション固有の Web サービスセキュリティの設定	143
サンプルアプリケーションのセキュリティ保護	144
メッセージセキュリティのための Communications Server の設定	144
要求および応答ポリシー設定のアクション	144
その他のセキュリティ機能の設定	146
JCE プロバイダの設定	146
メッセージセキュリティの設定	148
メッセージセキュリティのためのプロバイダの有効化	149
メッセージセキュリティプロバイダの設定	149

メッセージセキュリティープロバイダの作成	150
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	150
アプリケーションクライアント設定の要求および応答ポリシーの設定	151
詳細情報	152
11 診断サービスの設定	153
診断フレームワークとは	153
診断サービスフレームワーク	153
診断レポートの生成	154
12 トランザクション	155
トランザクションについて	155
トランザクションとは	155
Java EE テクノロジーのトランザクション	156
特定のデータベースに関する問題の回避方法	157
トランザクションに関する管理コンソールタスク	157
トランザクションの設定	157
13 HTTP サービスの設定	163
仮想サーバー	163
HTTP リスナー	164
14 SIP サービスの設定	169
SIP サービス	169
SIP サービスのアクセスログ属性の設定	170
SIP サービスのキープアライブ属性の設定	171
SIP サービスの接続プール属性の設定	171
SIP サービスのプロトコル属性の設定	172
SIP リスナー	173
SIP リスナーの作成と編集	174
▼ SSL を使用するように SIP リスナーを設定する	175
15 過負荷保護	177
CPU 過負荷保護	178

メモリ過負荷保護	179
16 Web サービスの管理	181
Web サービスの概要	181
Web サービスの規格	182
Java EE Web サービスの規格	183
Web サービスの配備とテスト	184
Web サービスの配備	184
配備済み Web サービスの表示	184
Web サービスのテスト	185
Web サービスのセキュリティー	185
Web サービスレジストリの使用	185
レジストリの追加	185
レジストリへの Web サービスの発行	186
XSLT フィルタによるメッセージの変換	186
Web サービスの監視	187
Web サービス統計の表示	187
Web サービスメッセージの監視	188
17 ORB (Object Request Broker) の設定	189
Object Request Broker の概要	189
CORBA	189
ORB とは	190
IIOP リスナー	190
ORB の設定	190
IIOP リスナーの管理	190
18 スレッドプール	191
スレッドプールの操作	192
SIP コンテナのスレッドプールの設定	193
19 ロギングの設定	195
ロギングについて	195
ログレコード	195

モジュールのログレベル	196
ロギングの設定	197
ログの一般設定	197
ログレベルの設定	198
SIP メッセージ検査	198
サーバーログの表示	199
20 コンポーネントとサービスの監視	203
監視について	203
Communications Server での監視	203
監視の概要	204
監視可能なオブジェクトのツリー構造について	204
監視対象のコンポーネントとサービスの統計について	208
監視の有効化と無効化	230
管理コンソールを使用した監視レベルの設定	230
▼ asadmin を使用して監視レベルを設定する	231
監視データの表示	231
管理コンソールでの監視データの表示	231
asadmin ツールによる監視データの表示	232
JConsole の使用	249
JConsole から Application Server への接続のセキュリティーを有効にする	249
JConsole を Application Server に接続する前提条件	250
▼ JConsole を Application Server に接続する	251
▼ 安全に JConsole を Application Server に接続する	251
21 管理ルールの設定	253
管理ルールについて	253
管理ルールの設定	254
22 Java 仮想マシンと詳細設定	257
JVM 設定の調整	257
詳細設定	258

A	ドメインまたはノードエージェントの自動再起動	261
	Solaris 10 での自動再起動	261
	Solaris 9 および Linux プラットフォーム上での inittab による自動再起動	263
	Microsoft Windows プラットフォーム上での自動再起動	264
	Windows サービスの作成	264
	ユーザーの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	265
	自動再起動時のセキュリティー	266
B	domain.xml のドット表記名属性	269
	トップレベル要素	269
	別名を使用しない要素	271
C	asadmin コーティリティー	273
	asadmin コーティリティー	274
	リモートコマンドの共通オプション	276
	multimode コマンド	278
	get 、 set 、 list コマンド	278
	サーバーのライフサイクルコマンド	280
	リストおよびステータスコマンド	281
	配備コマンド	282
	バージョンコマンド	283
	Message Queue 管理コマンド	283
	リソース管理コマンド	284
	設定コマンド	286
	SIP リスナーコマンド	286
	HTTP および IIOP リスナーコマンド	287
	ライフサイクルおよび監査モジュールコマンド	287
	プロファイラおよび SSL コマンド	288
	JVM オプションおよび仮想サーバーコマンド	288
	スレッドプールおよび認証レルムコマンド	289
	トランザクションおよびタイマーコマンド	290
	レジストリコマンド	290
	ユーザー管理コマンド	291
	ルールおよび監視コマンド	291
	データベースコマンド	292

診断およびロギングコマンド	292
Web サービスコマンド	293
セキュリティーサービスコマンド	293
パスワードコマンド	295
検証コマンド	295
カスタム MBean コマンド	296
サービスコマンド	296
プロパティコマンド	296
索引	299

図目次

図 1-1	Communications Server インスタンス	35
図 9-1	ロールマッピング	122

表目次

表 1-1	各プロファイルで使用できる機能	32
表 1-2	ポートを使用する Communications Server リスナー	36
表 6-1	JNDI ルックアップと関連する参照	93
表 9-1	Communications Server の認証方法	119
表 10-1	メッセージ保護ポリシーと WS-Security SOAP メッセージセキュリティ ティー処理との対応づけ	145
表 20-1	ネットワークマネージャーと SIP リスナーの統計	209
表 20-2	(ネットワークマネージャー) 受信した要求の統計	210
表 20-3	(ネットワークマネージャー) 受信した応答の統計	211
表 20-4	過負荷保護マネージャーの統計	211
表 20-5	セッションマネージャーの統計	212
表 20-6	トランザクションマネージャーの統計	212
表 20-7	融合ロードバランサのフロントエンドの統計	213
表 20-8	融合ロードバランサのフロントエンドの統計	215
表 20-9	EJB 統計	216
表 20-10	EJB メソッドの統計	217
表 20-11	EJB プールの統計	217
表 20-12	EJB キャッシュの統計	218
表 20-13	タイマーの統計	219
表 20-14	Web コンテナ (サーブレット) の統計	219
表 20-15	Web コンテナ (Web モジュール) の統計	219
表 20-16	HTTP サービスの統計 (開発者プロファイル)	221
表 20-17	JDBC 接続プールの統計	222
表 20-18	コネクタ接続プールの統計	223
表 20-19	コネクタ作業管理の統計	224
表 20-20	ORB の接続マネージャーの統計	224
表 20-21	スレッドプールの統計	225
表 20-22	トランザクションサービスの統計	225
表 20-23	JVM の統計	226

表 20-24	Java SE の JVM 統計 - クラス読み込み	226
表 20-25	Java SE の JVM 統計 - コンパイル	226
表 20-26	Java SE の JVM 統計 - ガベージコレクション	227
表 20-27	Java SE の JVM 統計 - メモリー	227
表 20-28	Java SE の JVM 統計 - オペレーティングシステム	228
表 20-29	Java SE の JVM 統計 - ランタイム	228
表 20-30	Java SE の JVM 統計 - ThreadInfo	229
表 20-31	Java SE の JVM 統計 - スレッド	229
表 20-32	トップレベル	242
表 20-33	アプリケーションレベル	242
表 20-34	アプリケーション - エンタープライズアプリケーションとスタンドアロンモジュール	243
表 20-35	HTTP サービスレベル	246
表 20-36	スレッドプールレベル	246
表 20-37	リソースレベル	247
表 20-38	トランザクションサービスレベル	247
表 20-39	ORB レベル	248
表 20-40	JVM レベル	248
表 C-1	リモートコマンドの必須オプション	276
表 C-2	サーバーのライフサイクルコマンド	280
表 C-3	リストおよびステータスコマンド	281
表 C-4	配備コマンド	282
表 C-5	バージョンコマンド	283
表 C-6	Message Queue コマンド	283
表 C-7	リソース管理コマンド	284
表 C-8	SIP リスナーコマンド	286
表 C-9	IIOP リスナーコマンド	287
表 C-10	ライフサイクルモジュールコマンド	287
表 C-11	プロファイラおよび SSL コマンド	288
表 C-12	JVM オプションおよび仮想サーバーコマンド	289
表 C-13	スレッドプールおよび認証レルムコマンド	289
表 C-14	レジストリコマンド	290
表 C-15	レジストリコマンド	290
表 C-16	ユーザー管理コマンド	291
表 C-17	ルールおよび監視コマンド	291
表 C-18	データベースコマンド	292

表 C-19	診断およびロギングコマンド	292
表 C-20	Web サービスコマンド	293
表 C-21	セキュリティーコマンド	294
表 C-22	パスワードコマンド	295
表 C-23	検証コマンド	295
表 C-24	カスタム MBean コマンド	296
表 C-25	サービスコマンド	296
表 C-26	プロパティコマンド	297

例目次

例 20-1	アプリケーションノードのツリー構造	205
例 20-2	SIPサービスのツリー構造	206
例 20-3	融合ロードバランサのツリー構造	206
例 20-4	HTTPサービスの図(開発者プロファイル版)	206
例 20-5	HTTPサービスの図(クラスタおよびエンタープライズプロファイル版)	206
例 20-6	リソースの図	207
例 20-7	コネクタサービスの図	207
例 20-8	JMSサービスの図	207
例 20-9	ORBの図	208
例 20-10	スレッドプールの図	208
例 C-1	パスワードファイルの内容	276

はじめに

管理ガイドでは、Communications Server の管理タスクについて説明します。

ここでは、Sun GlassFish™ Communications 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。

Sun GlassFish Communications Server ドキュメントセット

表 P-1 Communications Server のドキュメントセットの内容

ドキュメント名	説明
『Documentation Center』	タスクや主題ごとに整理された Communications Server のドキュメントのトピック。
『リリースノート』	ソフトウェアとドキュメントに関する最新情報。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™ Development Kit (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。
『クイックスタートガイド』	Communications Server 製品の使用を開始するための手順。
『Installation Guide』	ソフトウェアとそのコンポーネントのインストール。
『アプリケーション配備ガイド』	アプリケーションおよびアプリケーション コンポーネントの Communications Server への配備。配備記述子に関する情報を含みます。
『開発者ガイド』	Communications Server 上で動作することを目的とし、Java EE コンポーネントおよび API のオープン Java スタンダードモデルに準拠した、Java 2 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。開発者ツール、セキュリティ、デバッグ、ライフサイクルモジュールの作成に関する情報を含みます。
Java EE 5 Tutorial	Java EE 5 プラットフォームテクノロジーと API を使用した Java EE アプリケーションの開発。
『Java WSIT Tutorial』	Web サービス相互運用性テクノロジー (WSIT) を使用した Web アプリケーションの開発。WSIT テクノロジーを使用する方法、時期、および理由と、各テクノロジーがサポートする機能およびオプションについて説明します。

表 P-1 Communications 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
『管理ガイド』	設定、監視、セキュリティー、資源管理、および Web サービス管理を含む Communications Server のシステム管理。
『高可用性 (HA) 管理ガイド』	クラスタの設定、ノードエージェントの操作、およびロードバランサの使用法。
『Administration Reference』	Communications Server 設定ファイル <code>domain.xml</code> の編集。
『パフォーマンスチューニングガイド』	パフォーマンスを向上させるための Communications Server の調整。
『Reference Manual』	Communications Server で使用できるユーティリティーコマンド。マニュアルページのスタイルで記述されています。 <code>asadmin</code> コマンド行インタフェースも含まれます。

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-2 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
<i>as-install</i>	Communications Server のベースインストールディレクトリを表します。	Solaris™ および Linux へのインストールで、root ユーザーでない場合: <i>user's-home-directory/SUNWappserver</i> Solaris および Linux へのインストールで、root ユーザーである場合: <i>/opt/SUNWappserver</i> Windows のすべてのインストールの場合: <i>SystemDrive:\Sun\AppServer</i>
<i>domain-root-dir</i>	すべてのドメインを含むディレクトリを表します。	すべてのインストール: <i>as-install/domains/</i>
<i>domain-dir</i>	ドメインのディレクトリを表します。 設定ファイルには、次のように表される <i>domain-dir</i> があります。 <code>\${com.sun.aas.instanceRoot}</code>	<i>domain-root-dir/domain-dir</i>
<i>instance-dir</i>	サーバーインスタンスのディレクトリを表します。	<i>domain-dir/instance-dir</i>

表 P-2 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
<i>samples-dir</i>	サンプルアプリケーションを含むディレクトリを表します。	<i>as-install/samples</i>
<i>docs-dir</i>	ドキュメントを含むディレクトリを表します。	<i>as-install/docs</i>

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-3 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:
<i>AaBbCc123</i>	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 <i>filename</i> と入力します。
<i>AaBbCc123</i>	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。 <i>cache</i> 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-4 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ーフェイ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。

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/>)

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せん。

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よって発生した、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。

このドキュメントに関するコメント

弊社では、ドキュメントの改善に努めており、お客様からのコメントおよびご忠告をお受けしております。

コメントを共有するには、<http://docs.sun.com> にアクセスして、「Feedback」をクリックします。このオンラインフォームでは、ドキュメントのタイトルと Part No. もご記入ください。Part No. は、7桁か9桁の番号で、ドキュメントのタイトルページまたは最初のページに記載されています。

Communications Serverの概要

Sun GlassFish Communications Server の管理には、融合アプリケーションと SIP アプリケーションの配備、ドメイン、サーバーインスタンス、およびリソースの作成と設定、ドメインとサーバーインスタンスの制御 (起動と停止)、プロファイルとクラスターの管理、パフォーマンスの監視と管理、問題の診断とトラブルシューティングなどの多くの作業が含まれます。

Communications Server は、GlassFish 上にセッション起動プロトコル (SIP) に関連する機能を追加します。

Communications Server は Java EE 融合、Web、および SIP アプリケーションと Java Web サービスの開発と配備用の Java EE 互換サーバーを提供します。主な機能には、スケーラブルなトランザクション管理、コンテナ管理による持続性ランタイム、パフォーマンス Web サービス、クラスタリング、高可用性、セキュリティー、統合機能などがあります。

Communications Server の概要と概念

ここでは、次の内容について説明します。

- [27 ページの「Communications Serverの概要」](#)
- [28 ページの「管理用ツール」](#)

Communications Serverの概要

Communications Server は、Java EE 5 仕様と SIP Servlets 1.1 API をサポートします。Communications Server は、Sun GlassFish Enterprise Server 2.1 をベースとしています。Communications Server は、GlassFish のコアアーキテクチャーと管理インフラストラクチャー上に構築されています。Communications Server は、SIP Servlets テクノロ

ジと EJB、Java Persistence API、JDBC、Web サービス、再利用コンテナサービス (JNDI、JMS、依存性注入、セキュリティ、トランザクション管理など) などのテクノロジーを融合します。

Communications Server の SIP プロトコルスタックは、Grizzly (<http://grizzly.dev.java.net>) を使用して実装されました。受信要求は、それぞれが特定の機能を実行する一連のハンドラに順に渡されます。たとえば、Overload Handler が有効な場合、サーバーが過負荷状態になると、このハンドラは適切なエラー応答を送信します。オプションのロードバランサが有効な場合は、セッション維持型の負荷分散ロジックが適用され、要求はクラスタ内の適切なインスタンスに転送されるか、スタックが無視されます。初期要求は配備されたアプリケーションに照合され、適切に経路指定されます。また、デフォルトまたはユーザー配備の Application Router に照会して、初期経路決定を行うこともできます。多くの SIP リスナーは、複数のネットワークインターフェースで TCP および UDP トラフィックを受け付けるように設定できます。SIP over TLS はサポートされます。

Web ベースの集中管理コンソールを使用して、Communications Server のインスタンスとクラスタを安全にリモートから設定および監視できます。キャリア配備環境の管理は、さまざまな方法で自動化できます。コマンド行インタフェース (CLI) を使用して、プロセスをスクリプト化および自動化できます。安定した JMX API を利用して、サーバーをプログラムによって監視したり、設定の照会や設定データの変更を行えます。監視は、JMX および SNMP インタフェースを通してサポートされません。監視レベルは、OFF から「低」および「高」まで動的に変化し、これにより収集する情報の量が変化します。

アプリケーションサーバードメインを作成するときに適切なプロファイルを指定することで、Communications Server を特定の使用状況に対して設定および最適化することができます。

Communications Server には、Metro Web サービススタック (<http://metro.dev.java.net>) が含まれます。Metro は、Java と .NET Web サービス間の相互運用性を保証するために、重要な WS-* の標準と WS-I の標準化された相互運用性プロファイルを実装します。

Communications Server には、Java Business Integration (JBI) 仕様の実装が含まれます。JBI は、Server-Oriented Architecture (SOA) に従ってビジネスシステムを構築するための Java 標準です。

管理用ツール

Communications Server では、次の管理ツールおよび API を使用できます。

- 29 ページの「管理コンソール」
- 29 ページの「コマンド行インタフェース (asadmin ユーティリティ)」
- 30 ページの「JConsole」

管理コンソール

管理コンソールは、ナビゲートしやすいインタフェースとオンラインヘルプを装備したブラウザベースのツールです。管理コンソールを使用するには、管理サーバー (ドメイン管理サーバーまたは DAS と呼ばれる) が稼動している必要があります。管理コンソールを起動するには、管理サーバーのホスト名とポート番号がわかっている必要があります。サーバーのポート番号は、Communications 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 を使用します。ユーザー名とマスターパスワードも、インストール時に指定しています。

管理コンソールを起動するには、Web ブラウザで次のように入力します。

```
http://hostname:port
```

次に例を示します。

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

管理コンソールを Communications Server がインストールされたマシンで実行する場合は、ホスト名として localhost を指定します。

Windows 環境で管理コンソールを起動するには、「スタート」メニューから「管理コンソール」を起動します。

インストールプログラムにより、domain1 という名前のデフォルト管理ドメインがデフォルトポート番号 4848 で生成されます。また、ドメイン管理サーバー (DAS) とは分離したインスタンスも作成されます。インストール後は、管理ドメインを作成して追加できます。各ドメインには、一意のポート番号を持ったドメイン管理サーバーがそれぞれ必要です。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も指定する必要があります。

DAS とは異なるサーバー上で稼動するリモートサーバーインスタンスを管理する場合は、ノードエージェントを作成し、リモートサーバーインスタンスを容易に管理できるようにします。サーバーインスタンスの作成、起動、停止、および削除は、ノードエージェントの役割です。ノードエージェントを設定するには、コマンド行インタフェース (CLI) のコマンドを使用します。

コマンド行インタフェース (asadmin ユーティリティー)

asadmin ユーティリティーは Sun GlassFish Communications Server のコマンド行インタフェースです。asadmin ユーティリティーと、このユーティリティーに関連するコマンドを使用して、管理コンソールで提供されている一連の同じ管理タスクを実行します。Solaris でのデフォルトのインストールルートディレクトリは /opt/SUNWappserver です。

asadmin ユーティリティーを起動するには、*as-install/bin* ディレクトリに移動し、次のように入力します。

```
$ ./asadmin
```

asadmin 内で使用可能なコマンドをリスト表示するには、次のように入力します。

```
asadmin> help
```

シェルのコマンドプロンプトで、asadmin コマンドを次のように実行することもできます。

```
$ asadmin help
```

コマンドの構文と例を表示するには、help のあとにコマンド名を入力します。次に例を示します。

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

指定したコマンドの asadmin help 情報が、UNIX のマニュアルページの形式で表示されます。これらのマニュアルページは、HTML 形式および PDF 形式の『[Sun GlassFish Communications Server 2.0 Reference Manual](#)』でも利用できます。

JConsole

JConsole は Sun GlassFish Communications Server の監視に使用します。JConsole の「リモート」タブまたは「詳細」タブを使用して、Communications Server に接続できます。

- 「リモート」タブ: ユーザー名、パスワード、管理サーバーホスト、および JMS ポート番号 (デフォルトで 8686) を指定し、「接続」を選択します。
- 「詳細」タブ: JMXServiceURL のサービス URL (jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi) を指定し、「接続」を選択します。JMXServerURL は server.log ファイルに書き込まれるほか、ドメイン作成コマンドのコマンドウィンドウに出力されます。

Communications Server の概念

Communications Server は、1 つまたは複数のドメインから構成されます。ドメインは管理上の境界であり、コンテキストです。各ドメインには管理サーバー (ドメイン管理サーバーまたは DAS と呼ばれる) が関連付けられ、0 またはそれ以上のスタンドアロンインスタンスまたはクラスタ、あるいはその両方から構成されています。各クラスタには、1 つ以上の同機種サーバーインスタンスが含まれます。サーバーインスタンスは、単一の物理マシンで Application Server を実行する単一の Java 仮想マシン (JVM) です。ドメイン内のサーバーインスタンス (スタンドアロンでもクラスタ構成でも) は異なる物理ホストで実行できます。

ここでは、次の内容について説明します。

- 31 ページの「ドメイン」
- 31 ページの「ドメイン管理サーバー (DAS)」
- 32 ページの「プロファイル」
- 33 ページの「クラスタ」
- 33 ページの「ノードエージェント」
- 34 ページの「サーバーインスタンス」

ドメイン

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。ただし、アプリケーションサーバーインスタンスは1つのドメインにのみ属することができます。管理上の境界線であることに加え、ドメインは基本的なセキュリティー構造を提供し、これによってさまざまな管理者がアプリケーションサーバーインスタンスの特定のグループ(ドメイン)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 Communications Server インストールを共有できます。各ドメインには、固有の設定、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つのドメインの設定が変更されても、ほかのドメインの設定は影響を受けません。

Sun GlassFish Communications 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 (server という名前) も作成されます。インストール時には管理サーバーポート番号を指定する必要があります。デフォルトの管理サーバーポートは 4848 です。インストーラは、管理ユーザー名とマスターパスワードの入力も要求します。インストール後は、管理ドメインを作成して追加できます。

ドメイン管理サーバー (DAS)

各ドメインは、一意のポート番号を持ったドメイン管理サーバー (DAS) を持っています。管理コンソールは特定の DAS と通信し、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。

ドメイン管理サーバー (DAS) は管理対象アプリケーションの制御専用設計されたアプリケーションサーバーインスタンスです。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します。DAS は管理サーバーまた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、Sun GlassFish Communications 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。DAS は単に追加の管理機能を備えたサーバーインスタンスです。

管理コンソールの各セッションでは、単一のドメインを設定し、管理できます。複数のドメインを作成している場合は、追加の管理コンソールセッションを起動し

て、ほかのドメインを管理する必要があります。管理コンソールの URL を指定する場合は、管理するドメインに関連付けられた DAS のポート番号を使用してください。

プロファイル

すべての管理ドメインは、そのドメインの機能を特定するプロファイルに関連付けられます。Communications Server には次のプロファイルが用意されています。

- **開発者:** このプロファイルは、ドメインが開発環境で実行されており、開発中のアプリケーションに NSS キーストア機能やクラスタ化機能 (負荷分散やセッション持続など) が必要でない場合に使用します。
- **クラスタ:** このプロファイルは、クラスタを作成する必要があるが、高可用性データベース (HADB) や NSS キーストアは必要としない場合に使用します。

注 - クラスタプロファイルおよび開発者プロファイルでは、DAS はデフォルトで非セキュリティー保護モードに設定されます。asadmin コマンドを実行するときに、必要な場合は常に `--secure=false` を使用してください。DAS をセキュリティー保護モードに設定するために変更を行なっている場合は、このフラグを使用しないでください。

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。プロファイルにより、インストールされた Application Server 自体の実行バイナリと、実行環境の設定を区別しやすくなります。つまり、プロファイルにより、Communications Server の 1 つの実行バイナリを使用して、特定のニーズに合った異なるドメインを作成できます。たとえば、場合によっては最新の Java EE 仕様を理解するために Communications Server を使用する開発者もいます。そのような開発者には、厳格なセキュリティー設定は必要ありません。一方、本稼動環境でアプリケーションを配備するユーザーには、当然ながらセキュリティー保護された環境が必要です。

表 1-1 に、各プロファイルで使用できる機能の一覧を示します。

表 1-1 各プロファイルで使用できる機能

機能	開発者プロファイル	クラスタプロファイル	エンタープライズプロファイル (Sun GlassFish Communications Server では使用不可)
セキュリティーストア	JKS	JKS	NSS
クラスタ化/スタンドアロンインスタンス	使用不可	利用可能	利用可能

表 1-1 各プロファイルで使用できる機能 (続き)

機能	開発者プロファイル	クラスタプロファイル	エンタープライズプロファイル (Sun GlassFish Communications Server では使用不可)
セキュリティーマネージャー	無効	有効	有効
HADB	使用不可	使用不可	利用可能
負荷分散	使用不可	利用可能	利用可能
ノードエージェント	使用不可	利用可能	利用可能

クラスタ

クラスタは、一連の同じアプリケーション、リソース、および設定情報を共有するサーバーインスタンスの集まりに名前を付けたものです。1つのサーバーインスタンスは1つのクラスタにのみ属することが可能です。クラスタを使用すると、複数のマシン間で負荷が分散されることによって、サーバーインスタンスのロードバランスが容易になります。また、インスタンスレベルのフェイルオーバーによって、高可用性を実現します。管理上の観点では、クラスタは仮想エンティティーを表し、クラスタへの操作(アプリケーションの配備など)は、そのクラスタを構成するすべてのインスタンスに反映されます。

クラスタに Communications Server インスタンスを追加することで水平方向への拡張が実現され、それによってシステムの処理能力が向上します。サービスを中断せずに、クラスタに Communications Server インスタンスを追加することができます。HTTP、RMI/IIOP、および JMS のロードバランスにより、クラスタ内の正常な Communications Server インスタンスに要求を分散させます。

高可用性 - 可用性を有効にすると、クラスタ内の Communications Server インスタンスをフェイルオーバーによって保護できます。1つのアプリケーションサーバーインスタンスが停止すると、利用できなくなったサーバーに割り当てられていたセッションは別の Communications Server インスタンスに引き継がれます。セッション情報は、セッションレプリケーション機能を使用するか、高可用性データベース (HADB) を使用して保存されます。HADB は、持続的な HTTP セッションとステートフルセッション Beans をサポートします。

ノードエージェント

インスタンスのリモートライフサイクル管理を容易にするには、ドメインの各ノードに、軽量エージェント (JMX ランタイムのみで稼働できるなど) が必要です。この主な目的は、DAS の指示どおりに、サーバーインスタンスを起動、停止、作成することです。さらに、ノードエージェントはウォッチドッグとして機能し、障害の発生したプロセスを再起動します。DAS と同様に、ノードエージェント

は特定の管理操作にのみ必要で、高可用性を期待するべきではありません。ただし、ノードエージェントは「常時稼働」コンポーネントであるため、ネイティブ O/S ノードブートストラップ (Solaris/Linux inetd または Windows サービスとしてなど) によって起動するように設定する必要があります。ノードエージェントは DAS には必要ありません。

サーバーインスタンス

サーバーインスタンスは、単一のノードの Communications Server 上で稼動する単一の Java EE 互換 Java 仮想マシン (JVM) です。ドメインの各サーバーインスタンスは一意の名前を持ちます。クラスタ化されたサーバーインスタンスはクラスタのメンバーであり、親クラスタからすべてのアプリケーション、リソース、および設定を受け取るため、クラスタのすべてのインスタンスは均一になります。クラスタ化されていないサーバーインスタンスはクラスタに属さないため、アプリケーション、リソース、および設定で独立したセットを使用します。次の図は、アプリケーションサーバーインスタンスの詳細を示しています。アプリケーションサーバーインスタンスは、Communications Server のクラスタリング、ロードバランス、セッションの持続性といった各機能の基本を構成するものです。

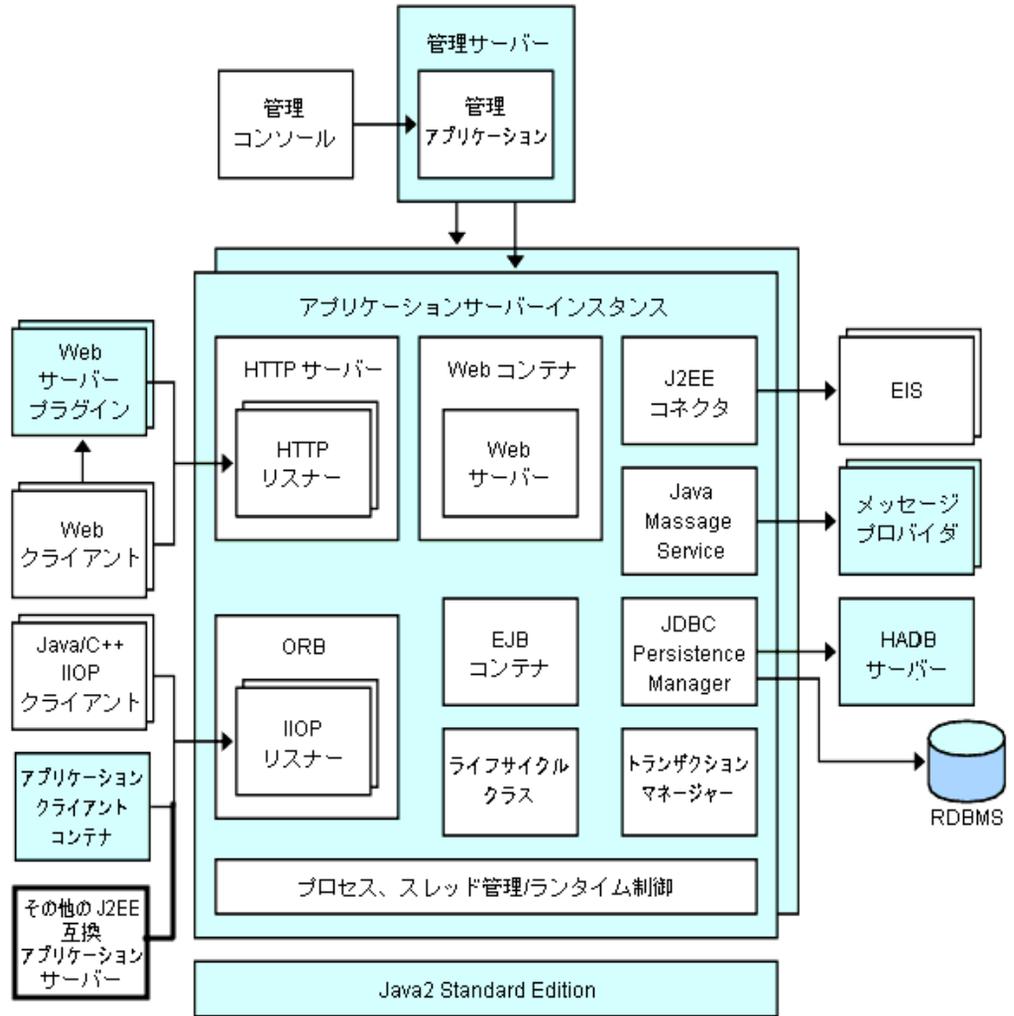


図 1-1 Communications Server インスタンス

Sun GlassFish Communications Server は、インストール時に `server` という名前のアプリケーションサーバーインスタンスを作成します。多くのユーザーは、1つのアプリケーションサーバーインスタンスがあれば、要件は満たされるでしょう。ただし、環境によっては、1つ以上の追加のアプリケーションサーバーインスタンスを作成する場合があります。たとえば、開発環境で、異なるアプリケーションサーバーインスタンスがあれば、異なる Communications Server 設定でテストしたり、異なるアプリケーション配備を比較およびテストしたりすることができま

す。アプリケーションサーバーインスタンスは簡単に追加または削除できるため、これらを使用して、一時的にサンドボックス領域を作成して試用することができます。

さらに、各アプリケーションサーバーインスタンスに対して、仮想サーバーを作成することもできます。単一のインストールされているアプリケーションサーバーインスタンス内で、企業または個人のドメイン名、IP アドレス、いくつかの管理機能を提供できます。ユーザーにとっては、ハードウェアを持つことも、サーバーの基本的な保守を行うこともなく、自分の Web サーバーを所有しているのと同様です。このような仮想サーバーは、複数のアプリケーションサーバーインスタンスにまたがりません。仮想サーバーの詳細は、第 13 章「HTTP サービスの設定」を参照してください。

運用上において、複数のアプリケーションサーバーインスタンスの代わりに仮想サーバーをさまざまな用途に応じて使用できます。ただし、仮想サーバーがニーズを満たさない場合、複数のアプリケーションサーバーインスタンスを使用することも可能です。アプリケーションサーバーインスタンスを停止すると、そのアプリケーションサーバーインスタンス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。マシンがクラッシュしたり、オフラインになったりすると、サーバーは終了し、処理中の要求が失われる可能性があります。

Communications Server のポート

次の表に、Communications Server のポートリスナーを示します。

表 1-2 ポートを使用する Communications Server リスナー

リスナー	デフォルトのポート番号	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 <code>asadmin</code> コマンドラインユーティリティを使用してアクセスします。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 <code>asadmin</code> コマンドを実行する場合は、 <code>--port</code> オプションを使用してポート番号を指定します。
HTTP	8080	サーバーはポート上で HTTP 要求を待機します。配備された 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。
IIOP	3700	EJB コンポーネントであるエンタープライズ Beans のリモートクライアントは IIOP リスナー経由で Beans にアクセスします。
IIOP_SSL	3820	セキュリティー保護された通信用に設定された IIOP リスナーは、ほかのポートを使用します。

表 1-2 ポートを使用する Communications Server リスナー (続き)

リスナー	デフォルトの ポート番号	説明
IOP_MUTUALAUTH	13920	相互(クライアントおよびサーバー)認証用に設定された IOP リスナーは、もう一方のポートを使用します。
SIP	5060	サーバーはポート上で SIP 要求を待機します。
SIPS	5061	セキュリティー保護された通信用に設定された SIP および融合アプリケーションは、個別のポートで待機します。
JMX_ADMIN	8686	
JMS	7676	

Communications Server の基本コマンド

Communications Server の管理には、ドメイン、クラスタ、ノードエージェント、およびサーバーインスタンスの作成、設定、制御、管理などのタスクが含まれます。ここでは、次の内容について説明します。

- 38 ページの「ドメインの作成」
- 38 ページの「ドメインの削除」
- 39 ページの「ドメインの一覧表示」
- 39 ページの「ドメインの起動」
- 39 ページの「Windows でのデフォルトドメインの起動」
- 39 ページの「ドメインの停止」
- 40 ページの「Windows でのデフォルトドメインの停止」
- 40 ページの「ドメインの再起動」
- 40 ページの「クラスタの作成」
- 40 ページの「クラスタの起動」
- 41 ページの「クラスタの停止」
- 41 ページの「ノードエージェントの作成」
- 41 ページの「ノードエージェントの起動」
- 42 ページの「ノードエージェントの停止」
- 42 ページの「インスタンスの起動」
- 42 ページの「インスタンスの停止」
- 42 ページの「インスタンスの再起動」
- 43 ページの「ドメイン管理サーバーの再作成」

ドメインの作成

ドメインは、`create-domain` コマンドを使用して作成します。次のコマンド例では、`mydomain` というドメインを作成します。管理サーバーが待機するポートは 5000 で、管理ユーザー名は `admin` です。このコマンドは、管理パスワードおよびマスターパスワードの入力を求めます。

```
$ asadmin create-domain --adminport 5000 --adminuser admin mydomain
```

`mydomain` ドメインの管理コンソールをブラウザ内で起動するには、次の URL を入力します。

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

Communications Server 9.1 では、各ドメインはそれぞれがプロファイルを持ちます。プロファイルについては、[32 ページの「プロファイル」](#)を参照してください。ドメインのプロファイルは、作成時に選択することができます。また、「一般」タブの「クラスタサポートを追加」をクリックすると、開発者プロファイルからクラスタプロファイルに変更できます。ドメインのプロファイルを指定するには、`create-domain` コマンドで `--profile` オプションを使用します。`--profile` オプションを使用してプロファイルを明示的に指定しない場合、そのドメインにはデフォルトのプロファイルが関連付けられます。デフォルトのプロファイルは、`asadminenv.conf` ファイル内の `AS_ADMIN_PROFILE` で定義されます。



注意 - HADB および ネットワークセキュリティーサービス (NSS) キーストアがない場合は、エンタープライズドメインを作成しないでください。HADB および NSS がないと、エンタープライズドメインを起動できません。

前述の `create-domain` の例の場合、ドメインのログファイル、設定ファイル、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
```

ドメインのディレクトリを別の位置に作成するには、`--domaindir` オプションを指定します。コマンドの完全な構文については、`asadmin help create-domain` と入力するか、[create-domain\(1\)](#) を参照してください。

ドメインの削除

ドメインは、`asadmin delete-domain` コマンドによって削除されます。ドメインを管理できる OS ユーザー (またはルート) だけが、このコマンドを正常に実行できます。たとえば、`mydomain` というドメインを削除するには、次のコマンドを入力します。

```
$ asadmin delete-domain mydomain
```

ドメインの一覧表示

マシン上に作成されているドメインを `asadmin list-domains` コマンドを使用して参照できます。デフォルトの `domain-root-dir` ディレクトリ内のドメインを一覧表示するには、次のコマンドを入力します。

```
$ asadmin list-domains
```

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。

ドメインの起動

ドメインの起動時に、管理サーバーとアプリケーションサーバーインスタンスが起動されます。アプリケーションサーバーインスタンスは、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。各ドメインは、別々に起動する必要があります。

ドメインを起動するには、`asadmin start-domain` コマンドを入力し、ドメイン名を指定します。たとえば、デフォルトのドメイン (`domain1`) を起動するには、次のように入力します。

```
$ asadmin start-domain --user admin domain1
```

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。コマンドの完全な構文を確認するには、`asadmin help start-domain` と入力してください。パスワードオプションを省略した場合は、入力するように要求されます。

Windows でのデフォルトドメインの起動

Windows の「スタート」メニューで、「プログラム」>「Sun Microsystems」>「Communications Server 9.1」>「デフォルトサーバーを起動」を選択します。

ドメインの停止

ドメインを停止すると、そのドメインの管理サーバーとアプリケーションサーバーインスタンスがシャットダウンします。ドメインを停止すると、そのサーバーインスタンス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。サーバーインスタンスはシャットダウンプロセスを完了しなければならないため、これ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` コマンドが使用できません。

ドメインを停止するには、`asadmin stop-domain` コマンドを入力し、ドメイン名を指定します。たとえば、デフォルトのドメイン (`domain1`) を停止するには、次のように入力します。

```
$ asadmin stop-domain domain1
```

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。コマンドの完全な構文を確認するには、`asadmin help stop-domain` と入力してください。

管理コンソールからドメインを停止するには、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。

Windows でのデフォルトドメインの停止

「スタート」メニューで、「プログラム」>「Sun Microsystems」>「Communications Server 9.1」>「デフォルトサーバー停止」を選択します。

ドメインの再起動

サーバーの再起動の手順はドメインの再起動と同じです。ドメインまたはサーバーを再起動するには、ドメインをいったん停止してから起動します。

クラスタの作成

クラスタを作成するには `create-cluster` コマンドを使用します。次の例では、`mycluster` という名前のクラスタを作成します。管理サーバーホストは `myhost`、サーバーポートは `1234`、管理ユーザー名は `admin` です。このコマンドは、管理パスワードの入力を求めます。

```
$ asadmin create-cluster --host myhost --port 1234 --user admin mycluster
```

コマンドの完全な構文を確認するには、`asadmin help create-cluster` と入力してください。

クラスタの起動

クラスタを起動するには `start-cluster` コマンドを使用します。次の例では `mycluster` という名前のクラスタを起動します。このコマンドは、管理パスワードの入力を求めます。

```
$ asadmin start-cluster --host myhost --port 1234 --user admin mycluster
```

コマンドの完全な構文を確認するには、`asadmin help start-cluster` と入力してください。

クラスタの停止

クラスタを停止するには `stop-cluster` コマンドを使用します。次の例では `mycluster` という名前のクラスタを停止します。このコマンドは、管理パスワードの入力を求めます。

```
$ asadmin stop-cluster --host myhost --port 1234 --user admin mycluster
```

`myhost` は管理サーバーホスト、`1234` は管理ポート、`admin` は管理ユーザー名です。

コマンドの完全な構文を確認するには、`asadmin help stop-cluster` と入力してください。クラスタを停止すると、クラスタのすべてのサーバーインスタンスが停止します。インスタンスを含まないクラスタは停止できません。

ノードエージェントの作成

ノードエージェントを作成するには `create-node-agent` コマンドを使用します。次の例では `mynodeagent` という名前のノードエージェントを作成します。管理サーバーホストは `myhost`、管理サーバーポートは `1234`、管理ユーザー名は `admin` です。このコマンドは、通常は管理パスワードの入力を求めます。ただし、`--savemasterpassword` オプションが指定されていないか、または `false` の場合、このコマンドはマスターパスワードの入力を求めません。

```
$ asadmin create-node-agent --host myhost --port 1234 --user admin mynodeagent
```

コマンドの完全な構文を確認するには、`asadmin help create-node-agent` と入力してください。

ノードエージェントの起動

ノードエージェントを起動するには `start-node-agent` コマンドを使用し、ノードエージェント名を指定します。たとえば、ノードエージェント `mynodeagent` を起動するには、次のように入力します。

```
$ asadmin start-node-agent --user admin mynodeagent
```

コマンドの完全な構文を確認するには、`asadmin help start-node-agent` と入力してください。

ノードエージェントの停止

ノードエージェントを停止するには `stop-node-agent` コマンドを使用し、ノードエージェント名を指定します。たとえば、ノードエージェント `mynodeagent` を停止するには、次のように入力します。

```
$ asadmin stop-node-agent mynodeagent
```

コマンドの完全な構文を確認するには、`asadmin help stop-node-agent` と入力してください。

インスタンスの起動

サーバーインスタンスを起動するには `start-instance` コマンドを使用します。次の例では、`myinstance` という名前のサーバーインスタンスを起動します。このコマンドは、管理パスワードの入力を求めます。

```
$ asadmin start-instance --host myhost --port 1234 --user admin myinstance
```

管理サーバーホストは `myhost`、管理ポートは `1234`、管理ユーザー名は `admin` です。サーバーインスタンス `myinstance` はクラスタ化することもスタンドアロンにすることもできます。

コマンドの完全な構文を確認するには、`asadmin help start-instance` と入力してください。

インスタンスの停止

サーバーインスタンスを停止するには `stop-instance` コマンドを使用します。次の例では、`myinstance` という名前のサーバーインスタンスを停止します。このコマンドは、管理パスワードの入力を求めます。

```
$ asadmin stop-instance --host myhost --port 1234 --user admin myinstance
```

管理サーバーホストは `myhost`、管理ポートは `1234`、管理ユーザー名は `admin` です。サーバーインスタンス `myinstance` はクラスタ化することもスタンドアロンにすることもできます。

コマンドの完全な構文を確認するには、`asadmin help stop-instance` と入力してください。

インスタンスの再起動

サーバーインスタンスを再起動するには、インスタンスを停止してから、再起動します。

ドメイン管理サーバーの再作成

ミラーリングを行うため、および、ドメイン管理サーバー (DAS) の有効なコピーを提供するためには、次のものを用意する必要があります。

- 元の DAS を含む、1 番目のマシン (*machine1*)。
- アプリケーションを実行していてクライアントに提供しているサーバーインスタンスとクラスタを含む、2 番目のマシン (*machine2*)。1 番目のマシン上の DAS を使用して、クラスタが設定されています。
- 1 番目のマシンがクラッシュしたときには、3 番目のバックアップマシン (*machine3*) に DAS が再作成される必要があります。

注-1 番目のマシンからの DAS のバックアップを保持している必要があります。現在のドメインをバックアップするには、`asadmin backup-domain` を使用します。

▼ DAS を移行する

ドメイン管理サーバーを 1 台目のマシン (*machine1*) から 3 台目のマシン (*machine3*) に移行するには、次の手順が必要です。

- 1 1 番目のマシンにインストールしたときと同様に、アプリケーションサーバーを 3 番目のマシンにインストールします。
これは、DAS が 3 番目のマシンに正しく復元され、パスの競合が起きないようにするために必要です。
 - a. コマンド行(対話型)モードを使用して、アプリケーションサーバー管理パッケージをインストールします。対話型コマンド行モードを有効にするには、次のように `console` オプションを使用してインストールプログラムを呼び出します。

```
./bundle-filename -console
```


コマンド行インタフェースを使用してインストールするには、ルート権限が必要です。
 - b. デフォルトのドメインをインストールするオプションを選択解除します。
バックアップされたドメインの復元は、同じアーキテクチャーおよびまったく同じインストールパスを持つ 2 台のマシンでのみサポートされます (すなわち両方のマシンが同じ `as-install` と `domain-root-dir` を使用する)。
- 2 バックアップの ZIP ファイルを 1 番目のマシンから 3 番目のマシンの `domain-root-dir` にコピーします。ファイルを FTP 転送することもできます。

- 3 `asadmin restore-domain` コマンドを実行して、ZIP ファイルを 3 台目のマシンに復元します。

```
asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip domain1
```

任意のドメインをバックアップできます。ただし、ドメインの再作成時に、ドメイン名を元のドメインと同じ名前にしてください。

- 4 3 番目のマシンの `domain-root-dir/domain1/generated/tmp` ディレクトリのアクセス権を変更し、1 番目のマシンの同じディレクトリのアクセス権と一致させます。

このディレクトリのデフォルトのアクセス権は、`?drwx-----?`(または 700) です。

次に例を示します。

```
chmod 700 domain-root-dir/domain1/generated/tmp
```

この例では、`domain1` をバックアップしていることとします。ドメインを別の名前でバックアップしている場合は、上記の `domain1` をバックアップしているドメインの名前で置き換えるようにしてください。

- 5 3 台目のマシンの `domain.xml` で、プロパティのホスト値を変更します。
- 6 3 番目のマシンの `domain-root-dir/domain1/config/domain.xml` を更新します。
たとえば、`machine1` を検索し、`machine3` に置き換えます。次の箇所を変更できます。

```
<jmx-connector><property name=client-hostname value=machine1/>...
```

次のように変更します。

```
<jmx-connector><property name=client-hostname value=machine3/>...
```

- 7 変更前:

```
<jms-service... host=machine1.../>
```

次のように変更します。

```
<jms-service... host=machine3.../>
```

- 8 `machine3` で復元されたドメインを起動します。

```
asadmin start-domain --user admin-user --password admin-password domain1
```
- 9 `machine2` で、ノードエージェントの下にあるプロパティの `DAS` ホスト値を変更します。
- 10 `machine2` の `as-install/nodeagents/nodeagent/agent/config/das.properties` で、`agent.das.host` プロパティ値を変更します。

11 machine2でノードエージェントを再起動します。

注 - `asadmin start-instance` コマンドを使用してクラスティンスタンスを起動し、復元されたドメインと同期できるようにします。

Java Business Integration

Java Business Integration (JBI) は、[JSR 208 仕様](http://www.jcp.org/en/jsr/detail?id=208) (<http://www.jcp.org/en/jsr/detail?id=208>) の実装です。これは、Java Community Process (JCP) の下でサービス指向アーキテクチャー (SOA) を実装する方法として開発された標準です。

JBI は、Web Services Description Language (WSDL) 2.0 に直接基づいたサービスモデルを使用して相互に作用するプラグインコンポーネントのための環境を定義します。これらのプラグインコンポーネントは、サービスプロバイダ、サービスコンシューマ、またはその両方として機能します。

JBI 実行環境の主要コンポーネントおよびそれらのライフサイクル状態の管理の詳細については、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。JBI コマンドの使用方法については、『[Sun GlassFish Communications Server 2.0 Reference Manual](#)』を参照してください。

JBI 環境

次の各節で、JBI 環境の主要コンポーネントについて説明します。

- 47 ページの「JBI コンポーネント」
- 49 ページの「サービスアセンブリ」
- 50 ページの「共用ライブラリ」
- 50 ページの「JBI 記述子」

JBI コンポーネント

サービスエンジン

サービスエンジンは、ローカルサービス (つまり、JBI 環境内部のサービス) を提供し、ローカルまたはリモートサービスを消費するコンポーネントです。

バイディングコンポーネント

バイディングコンポーネントは、JBI環境の外部にあるコンシューマまたはプロバイダのプロキシです。バイディングコンポーネントは、通常、FTP、JMS、SMTPなどの標準的な通信プロトコル、またはSAPやwebSphereMQなどの外部サービスの呼び出しに基づいています。

JBIコンポーネントには、次のライフサイクル状態があります。

- 起動済み
- 停止
- シャットダウン

JBIランタイムは、JBIコンポーネントのライフサイクル状態を持続します。サーバーをシャットダウンしてから再起動すると、JBIコンポーネントはCommunications Serverがシャットダウンした時点の状態に戻ります。

注-JBIランタイムは、JBIコンポーネントの「適切な」状態に戻ろうとします。たとえば、JBIコンポーネントを起動しようとしたが、コンポーネント内のエラーのために起動しなかったとします。Communications Serverを再起動すると、JBIランタイムはコンポーネントの起動を再試行します。

JBIコンポーネントに対して次の操作を実行できます。詳細な手順については、管理コンソールにログオンし、JBIノードに移動して、「コンポーネント」をクリックし、次に「ヘルプ」をクリックしてください。

- 特定のライフサイクル状態ごとにJBIコンポーネントを表示する。
- JBIコンポーネントをインストールする。
- JBIコンポーネントをアンインストールする。
- JBIコンポーネントのライフサイクル状態を管理する。
- JBIコンポーネントの一般プロパティを表示する。
- JBIコンポーネントの設定情報を表示する。
- JBIコンポーネントの記述子を表示する。
- JBIコンポーネントのロガーを管理する。

JBIコンポーネントロガー

管理コンソールを使用して、JBIコンポーネントのログレベルを管理できます。JBIコンポーネントの中には、複数のロガーを提供するものや、ロガーを提供しないものもありますが、コンポーネント全体では常に1つのロガーレベルが表示されます。ただし、ロガーレベルの設定が有効になるのは、コンポーネントがデフォルトの名前に基づいてロガーを実装している場合だけです。ログレベルの指定に関しては、JBIコンポーネントのプロバイダから提供されるドキュメントもあわせて参照してください。

注-JBI コンポーネントのログレベルは、多くの場合、JBI ロガーなどの親ロガーから継承されます。親のログレベルを表示および設定するには、開発者プロファイルを持つドメインの場合、管理コンソールで「アプリケーションサーバー」パネルを選択します。次に、Communications Server パネルの「ログ」タブを選択し、「ログレベル」タブを選択します。JBI モジュールのドロップダウンリストで親の JBI ログレベルの表示および設定を行います。

サービスアセンブリ

サービスアセンブリは、アプリケーションのために特定のサービスを共同で提供または消費するターゲットコンポーネントをプロビジョニングするサービスユニットの集合です。通常、サービスアセンブリは開発ツール環境 (NetBeans Enterprise Pack で提供される環境など) で作成されます。

サービスアセンブリには、次のライフサイクル状態があります。

- 起動済み
- シャットダウン
- 停止

JBI ランタイムは、サービスアセンブリのライフサイクル状態を持続しません。Communications Server をシャットダウンしてから再起動すると、サービスアセンブリは Communications Server がシャットダウンした時点の状態に戻ります。

注-JBI ランタイムは、サービスアセンブリの「適切な」状態に戻ろうとします。たとえば、サービスアセンブリを起動しようとしたが、サービスアセンブリ内のエラーのために起動しなかったとします。Application Server を再起動すると、JBI ランタイムはサービスアセンブリの起動を再試行します。

サービスアセンブリに対して次の操作を実行できます。詳細な手順については、管理コンソールにログオンし、JBI ノードに移動して、「サービスアセンブリ」をクリックし、次に「ヘルプ」をクリックしてください。

- サービスアセンブリを表示する。ライフサイクル状態によるソートとフィルタリングがサポートされます。
- サービスアセンブリを配備する。
- サービスアセンブリの配備を取り消す。
- サービスアセンブリのライフサイクルを管理する。
- サービスアセンブリの一般プロパティを表示する。
- サービスアセンブリの記述子を表示する。

共用ライブラリ

共用ライブラリは、1つのコンポーネントに占有されない Java クラスを提供し、通常は複数の JBI コンポーネントによって共有されます。たとえば、Java EE サービスエンジンには WSDL 共用ライブラリが必要です。

共用ライブラリに対して次の操作を実行できます。詳細な手順については、管理コンソールにログオンし、JBI ノードに移動して、「共用ライブラリ」をクリックし、次に「ヘルプ」をクリックしてください。

- すべての共用ライブラリを表示する。
- 共用ライブラリをインストールする。
- 共用ライブラリの一般プロパティを表示する。
- 共用ライブラリの記述子を表示する。
- 共用ライブラリをアンインストールする。

JBI 記述子

サービスアセンブリ、JBI コンポーネント、および共用ライブラリの記述子ファイル (`jbi.xml`) は、次の情報を提供します。

- サービスアセンブリ: サービスアセンブリに含まれるサービスユニットと各サービスユニットのターゲットを示すリスト。サービスユニットによっては、接続エンドポイントに関する情報が示される場合もあります。
- JBI コンポーネント: JBI コンポーネントのタイプ (バインディングコンポーネントまたはサービスエンジン)、コンポーネントの説明、コンポーネントに関連するクラスパスの情報、およびコンポーネントが依存する共用ライブラリの名前を示すリスト。
- 共用ライブラリ: 共用ライブラリの名前と、共用ライブラリを格納するアーカイブファイル (.jar ファイル) またはクラスファイルサブディレクトリの名前を示すリスト。

JDBC リソース

この章では、データベースにアクセスするアプリケーションに必要な、JDBC リソースの設定方法について説明します。この章で説明する内容は次のとおりです。

- 51 ページの「JDBC リソース」
- 52 ページの「JDBC 接続プール」
- 52 ページの「JDBC リソースと接続プールの協調動作について」
- 53 ページの「データベースアクセスの設定」
- 54 ページの「JDBC 接続プールの操作」
- 61 ページの「各 JDBC ドライバの設定」

JDBC リソース

データを格納、編成、取得するために、ほとんど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ます。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。

JDBC リソース (データソース) は、アプリケーションにデータベースへ接続する手段を提供します。通常、管理者は、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各データベースの JDBC リソースを作成します。ただし、1つのデータベースに複数の JDBC リソースを作成できます。

JDBC リソースを作成するには、リソースを識別する一意の JNDI 名を指定します。「JNDI 名とリソース」の節を参照してください。通常、JDBC リソースの JNDI 名は `java:comp/env/jdbc` サブコンテキストにあります。たとえば、給与データベースのリソースの JNDI 名は `java:comp/env/jdbc/payrolldb` と名付けることができます。すべてのリソース JNDI 名は `java:comp/env` サブコンテキストにあるので、管理コンソールの JDBC リソースにある JNDI 名を指定するときは、`jdbc/name` だけを入力します。たとえば、給与データベースには `jdbc/payrolldb` を指定します。

JDBC リソース (データソース) は、アプリケーションにデータベースへ接続する手段を提供します。

JDBC リソースを作成する前に、まず JDBC 接続プールを作成します。

JDBC リソースを作成するときには、次の項目を指定してください。

1. JNDI 名。規則により、名前は jdbc/ 文字列で始まります。次に例を示します。
jdbc/payrolldb。スラッシュを忘れないでください。
2. 新しい JDBC リソースに関連付ける接続プールを選択します。
3. リソースの設定を指定します。
4. クラスタプロファイルまたはエンタープライズプロファイルの場合、リソースが利用できるターゲット (クラスタおよびスタンドアロンサーバーインスタンス) を指定します。

JDBC 接続プール

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 接続プールとは、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。新しい物理接続をそれぞれ作成するには時間がかかるので、パフォーマンスの向上のためにサーバーは利用可能な接続のプールを保持しています。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、接続はプールに戻されます。

接続プールのプロパティは、データベースベンダーによっては異なる場合があります。共通のプロパティには、データベースの名前 (URL)、ユーザー名、およびパスワードがあります。

関連項目

- [51 ページの「JDBC リソース」](#)
- [52 ページの「JDBC リソースと接続プールの協調動作について」](#)
- [56 ページの「JDBC 接続プールの編集」](#)

JDBC リソースと接続プールの協調動作について

データを格納、編成、取得するために、ほとんど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ます。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。アプリケーションがデータベースにアクセスするには、接続を取得する必要があります。

実行時に、アプリケーションがデータベース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データベースに関連した JDBC リソース (データソース) を取得します。

リソースの JNDI 名を取得すると、ネーミングおよび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. JDBC リソースを経由して、アプリケーションはデータベース接続を取得します。

バックグラウンドで、アプリケーションサーバーはデータベ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、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. データベースに接続すると、アプリケーションはデータベース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 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終了します。

アプリケーションサーバーは接続を接続プールに戻します。接続がプールに戻れると、次のアプリケーションがその接続を利用できるようになります。

データベースアクセスの設定

データベースアクセスを設定するには、次の手順に従います。

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。
Communications Server がサポートするデータベース製品のリストについては、リリースノートを参照してください。
2. データベース製品の JDBC ドライバをインストールします。
3. ドメインのサーバーインスタンスにアクセスできるドライバの JAR ファイルを作成します。
4. データベースを作成します。
通常、アプリケーションプロバイダがデータベースを作成し移行するためのスクリプトを配信します。
5. データベースの接続プールを作成します。
6. 接続プールを示す JDBC リソースを作成します。

次に、JDBC ドライバを管理ドメインに統合するため、次のどれかを実行します。

1. 共通クラスローダーにアクセスできるドライバを作成します。
ドライバの JAR および ZIP ファイルを *domain-dir/lib* ディレクトリにコピーするか、または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。

2. ドメインを再起動します。
3. 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 接続プールの操作

JDBC 接続プールとは、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。管理コンソールでプールを作成すると、管理者は実際には特定のデータベースへの接続の項目を定義していることとなります。

プールを作成するには、まず JDBC ドライバをインストールして統合する必要があります。「接続プールを作成」ページを構築する際は、JDBC ドライバおよびデータベースベンダーに固有の特定のデータを入力する必要があります。処理を開始する前に、次の情報を集めます。

- データベースベンダー名
- `javax.sql.DataSource` (ローカルランザクションのみ) や `javax.sql.XADataSource` (グローバルランザクション) 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

管理コンソールを使用して、JDBC 接続プールを作成できます。

▼ 管理コンソールを使用した JDBC 接続プールおよび JDBC リソースの作成

- 1 データベース接続用の JDBC ドライバを `as-install/domains/domain_name/lib/ext` にコピーします。または、JDBC ドライバのパスを `classpath-prefix` に追加して、サーバーを再起動します。
- 2 管理コンソールで、「共通操作」ページの「JDBC 接続プールを作成」をクリックします。
- 3 接続プールの名前を入力して、リソースのタイプを選択し、データベースベンダーを選択します。「次へ」をクリックします。
- 4 管理コンソールに、デフォルトのプール設定と、ベンダーから提供されるデータソースクラスのプロパティーが表示されます。
- 5 「Ping」 ボタンをクリックして、JDBC 接続プールをテストします。

- 6 管理コンソールの左側の区画で、「リソース」、「JDBC リソース」の順にクリックします。
- 7 「新規」をクリックします。
- 8 「新しい JDBC リソース」ページで、リソースの JNDI 名と、JDBC リソースを関連付ける必要がある JDBC 接続プールを指定します。
- 9 「了解」をクリックします。

▼ CLI を使用した JDBC 接続プールおよび JDBC リソースの作成

- 1 `asadmin create-jdbc-connection-pool` コマンドを使用して、JDBC 接続プールを作成します。

JDBC 接続プールを作成するサンプルコマンドを、次に示します。

```
asadmin create-jdbc-connection-pool --datasourceclassname
oracle.jdbc.pool.OracleDataSource --restype javax.sql.DataSource --property
user=dbuser:password=dbpassword:url="jdbc:oracle:thin\:@localhost\:1521\:ORCL"
myjdbc_oracle-pool
```

- 2 `asadmin create-jdbc-resource` コマンドを使用して、この接続プールに関連付ける JDBC リソースを作成します。

JDBC リソースを作成するサンプルコマンドを、次に示します。

```
asadmin create-jdbc-connection-pool --connectionpoolid myjdbc_oracle-pool
jdbc/oracle-resource
```

- 3 `asadmin ping-connection-pool` コマンドを使用して接続プールに **ping** を送信し、接続設定が正しいことをテストします。

```
asadmin ping-connection-pool myjdbc_oracle-pool
```

- 4 JDBC 接続プールのプロパティを編集するには、`asadmin list`、`get`、および `set` コマンドを使用します。

JDBC 接続プールのリストを表示する、サンプルの `list` コマンドを示します。

```
asadmin list myjdbc_oracle-pool
```

このコマンドの出力例は次のようになります。

```
domain.resources.jdbc-connection-pool.oracle-pool
```

JDBC 接続プールのすべてのプロパティを表示する、サンプルの `get` コマンドを示します。

```
asadmin list myjdbc_oracle-pool
```

JDBC 接続プールのすべてのプロパティを取得する、サンプルの `get` コマンドを示します。

```
asadmin get domain.resources.jdbc-connection-pool.myjdbc_oracle-pool
```

JDBC 接続プールのプロパティを設定する、サンプルの `get` コマンドを示します。

```
asadmin set
```

```
domain.resources.jdbc-connection-pool.myjdbc_oracle-pool.steady-pool-size=20
```

JDBC 接続プールの編集

「JDBC 接続プールを編集」ページは、名前を除く既存プールのすべての設定を変更する手段を提供します。

1. 一般設定を変更します。

一般設定の値は、インストールした固有の JDBC ドライバにより異なります。これらの設定は、Java プログラミング言語で記述されたクラスやインタフェースの名前です。

パラメータ	説明
データソースクラス名	DataSource API または XDataSource API あるいはその両方を実装するベンダー固有のクラス名。このクラスは JDBC ドライバにあります。
リソースタイプ	選択肢には <code>javax.sql.DataSource</code> (ローカルトランザクションに限る)、 <code>javax.sql.XDataSource</code> (グローバルトランザクション)、および <code>java.sql.ConnectionPoolDataSource</code> (ローカルトランザクション、パフォーマンス向上の可能性あり)があります。

2. プール設定を変更します。

一連の物理データベース接続はプール内にあります。アプリケーションが接続を要求すると、接続はプールから削除され、アプリケーションが接続を解放すると、接続はプールに戻されます。

パラメータ	説明
初期および最小プールサイズ	プール内の接続の最小数。この値は、プールを最初に作成するときの、プールに含まれる接続の数も指定します。
最大プールサイズ	プールに含まれる接続の最大数。

パラメータ	説明
プールサイズ変更量	プールのサイズを最大プールサイズおよび最小プールサイズに対して拡大および縮小すると、一括処理でプールサイズが変更されます。この値は一括処理での接続の数を指定します。この値を過大に設定すると接続の作成と再利用が遅れ、過小に設定すると効率が落ちます。
アイドルタイムアウト	プールで接続がアイドル状態のままでいられる最長時間を指定します。この時間を過ぎると、接続はプールから削除されます。
最大待ち時間	接続を要求するアプリケーションが接続タイムアウトになるまでに待つ時間。デフォルトの待ち時間は長いので、アプリケーションがハングアップしているように見える可能性があります。

3. 接続検証設定を変更します。

オプションで、アプリケーションサーバーは接続が渡される前にそれを検証することができます。この検証により、ネットワークやデータベースサーバーに障害が発生してデータベースが利用できなくなった場合でも、アプリケーションサーバーが自動的にデータベース接続を再確立できます。接続の検証は追加オーバーヘッドとなるため、パフォーマンスに若干の影響が生じます。

パラメータ	説明
接続検証	「必須」チェックボックスにチェックを付けて、接続検証を有効にします。

パラメータ	説明
検証方法	<p>アプリケーションサーバーは、次の3つの方法でデータベース接続を検証できます。 auto-commit、 metadata、 および table。</p> <p>auto-commit と meta-data - アプリケーションサーバーは、 <code>con.getAutoCommit()</code> メソッドと <code>con.getMetaData()</code> メソッドを呼び出して接続を検証します。</p> <p>auto-commit の検証は、2つのメソッドを使用して接続を検証します。 <code>getAutoCommit()</code> メソッドは auto-commit の現在の状態を取得し、 <code>setAutoCommit()</code> は auto-commit の状態を変更します。これにより、データベースへの問い合わせが許可されます。 <code>getAutomCommit()</code> がデータベースに問い合わせできるかどうかは、実装に基づきます。実際の物理的な接続は、接続プールなどの別の目的でラップされます。</p> <p>注-多くのJDDBCドライバでは、これらの呼び出しの結果をキャッシュしているため、常に信頼のある検証が行われるとは限りません。呼び出しがキャッシュされるかどうかについては、ドライバベンダーに問い合わせる必要があります。</p> <p>table - アプリケーションは指定したデータベース表に問い合わせます。表は実在し、アクセス可能である必要がありますが、行は必要ありません。多くの行を持つ既存の表や、すでに頻繁にアクセスされている表を使用しないでください。</p>
テーブル名	「検証方法」コンボボックスで表を選択した場合は、ここでデータベースのテーブル名を指定します。
すべての障害で	「すべての接続を閉じる」のチェックボックスを選択してある場合、1つの接続が失敗すると、アプリケーションサーバーはプールに含まれるすべての接続を閉じ、それらを再確立します。そのチェックボックスを選択していない場合は、使用されるときにだけ個々の接続が再確立されます。
コンポーネントの呼び出し側以外を許可	サブレットフィルタやライフサイクルモジュールなど、コンポーネント以外の呼び出し側が使用するためのプールを有効にする場合は、このチェックボックスにチェックマークを付けます。

4. トランザクション設定を変更します。

データベースは通常多くのユーザーが同時にアクセスするため、あるトランザクションがデータを読み込もうとするとときに別のトランザクションが同じデータを更新する可能性があります。トランザクションの遮断レベルは、更新されるデータがほかのトランザクションに見える度合いを定義します。遮断レベルの詳細については、データベースベンダーのドキュメントを参照してください。

パラメータ	説明
非トランザクション接続	Application Server がすべての非トランザクション接続を返すようにする場合は、このチェックボックスにチェックマークを付けます。
トランザクション遮断	プールの接続のトランザクション遮断レベルを選択できるようにします。指定しない場合、接続には JDBC ドライバによって設定されるデフォルトの遮断レベルが適用されます。
遮断レベルを保証	遮断レベルを指定した場合にだけ適用されます。「保証」チェックボックスを選択すると、プールから取得されるすべての接続が同じ遮断レベルになります。たとえば、最後の使用時に <code>con.setTransactionIsolation</code> を使って接続の遮断レベルをプログラマ的に変更した場合、このメカニズムによって状態が指定の遮断レベルに戻されます。

5. プロパティを変更します。

「追加プロパティ」テーブルで、データベース名 (URL)、ユーザー名、およびパスワードなど、必要なプロパティを指定できます。データベースベンダーによってプロパティが異なるため、詳細については、ベンダーのドキュメントを調べてください。

JDBC 接続プールの詳細属性の編集

接続リークを診断しやすくし、使い勝手を向上させるため、Application Server 9.1 には接続プールをその作成時点で設定するための新しい属性がいくつか用意されています。

1. 「詳細」タブを開いて、次の属性を指定します。

属性	説明
名前	プロパティを編集する JDBC 接続プールの名前。ただし、プール名は変更できません。
文のタイムアウト	異常に長い時間実行されているクエリーを終了させるまでの時間 (秒数)。Application Server は、作成された文に対して「QueryTimeout」を設定します。デフォルト値の -1 は、この属性が無効であることを示します。

JDDBC オブジェクトをラップ	true に設定すると、アプリケーションは Statement、PreparedStatement、CallableStatement、ResultSet、DatabaseMetaData に対してラップされた JDDBC オブジェクトを取得します。デフォルト値は false です。
------------------	--

2. 次の表の説明に従って、接続設定を指定します。

属性	説明
最大で 1 回検証	経過するまでに 1 回しか接続が検証されない時間 (秒)。これは、1 つの接続による検証要求の数を減らすのに役立ちます。デフォルト値の 0 は、接続の検証が無効であることを示します。
リークタイムアウト	接続プール内の接続リークを追跡する時間 (秒)。デフォルト値の 0 は、接続リーク追跡が無効であることを示します。接続リーク追跡を有効にすると、「リソースの監視」タブに接続リークの数に関する統計情報が表示されます。このタブを表示するには、「Application Server」→「監視」→「リソース」の順に選択します。
リーク再要求	このオプションを有効にすると、リーク接続の追跡が完了したあとで、リークした接続がプールにリストアされます。
作成再試行回数	新しい接続の作成に失敗した場合に行われる試行の回数。デフォルト値の 0 は、接続の再試行が行われないことを示します。
再試行間隔	接続作成の試行の間隔 (秒数) を指定します。デフォルト値は 10 秒です。この属性は、「作成再試行回数」が 1 以上の場合にのみ使用されます。
使用時にのみ接続を登録	リソースが実際にメソッドで使用される場合にのみリソースをトランザクションに登録するには、このオプションを有効にします。
使用時にのみ関連付け	接続に対して操作が実行された時点ではじめて接続が関連付けられます。また、トランザクションが完了してコンポーネントメソッドが終了したときに関連付けが解除されるため、物理的な接続を再利用しやすくなります。デフォルト値は false です。

スレッドとの関連付け	接続をスレッドに関連付けるには、このオプションを有効にします。これにより、同じスレッドが接続を必要とするときは、そのスレッドにすでに関連付けられている接続を再利用できるため、プールから接続を取得するオーバーヘッドが発生しません。デフォルト値は false です。
接続のマッチング	プールに対する接続マッチングのオン/オフを切り替えるには、このオプションを使用します。プール内の接続が常に同じ種類であり、プールから選択した接続とリソースアダプタを照合する必要がない場合は、false に設定できます。デフォルト値は false です。
最大接続使用数	接続をプールで再利用する回数を指定します。指定された回数だけ再利用されると、その接続は閉じられます。これは、たとえば、メモリを回避するのに役立ちます。デフォルト値の 0 は、接続が再利用されないことを示します。

各 JDBC ドライバの設定

Communications Server 2.0 は、対応する JDBC ドライバを持つ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。次の JDBC ドライバとデータベースの組み合わせがサポートされます。これらの組み合わせは、Communications Server 2.0 でテスト済みであり、Java EE と互換性があることが確認されています。

- 62 ページの「Java DB Type 4 ドライバ」
- 63 ページの「DB2 データベース用の Sun GlassFish JDBC ドライバ」
- 63 ページの「Oracle 8.1.7 および 9.x データベース用の Sun GlassFish JDBC ドライバ」
- 64 ページの「Microsoft SQL Server データベース用の Sun GlassFish JDBC ドライバ」
- 64 ページの「Sybase データベース用の Sun GlassFish JDBC ドライバ」
- 65 ページの「IBM DB2 8.1 Type 2 ドライバ」
- 65 ページの「Sybase ASE 12.5 データベース用の JConnect Type 4 ドライバ」
- 66 ページの「MySQL Type 4 Driver」

その他の JDBC ドライバは、Communications Server 2.0 で使用可能ですが、Java EE との互換性テストはまだ完了していません。Sun では、それらのドライバの製品サポートは提供していませんが、Communications Server 2.0 での使用についての限定サポートを提供しています。

- 67 ページの「Oracle 8.1.7 および 9.x データベース用の Inet Oraxo JDBC ドライバ」
- 68 ページの「Microsoft SQL Server データベース用の Inet Merlia JDBC ドライバ」

- 68 ページの「Sybase データベース用の Inet Sybelux JDBC ドライバ」
- 69 ページの「Oracle 8.1.7 および 9.x 用の Oracle Thin Type 4 ドライバ」
- 70 ページの「Oracle 8.1.7 および 9.x データベース用の OCI Oracle Type 2 ドライバ」
- 70 ページの「IBM Informix Type 4 ドライバ」
- 71 ページの「CloudScape 5.1 Type 4 ドライバ」

JDBC ドライバを統合する方法、および管理コンソールまたはコマンド行インタフェースを使用して設定を実装する方法については、『Sun GlassFish Communications Server 2.0 管理ガイド』を参照してください。

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 Communications Server 2.0 Reference Manual』を参照してください。

Java DB Type 4 ドライバ

Java DB JDBC ドライバは、デフォルトで Communications Server に含まれています。ただし、Solaris バンドル版のインストールには Java DB は含まれていません。したがって、Solaris バンドル版のインストール以外では、この JDBC ドライバを Communications Server に組み込む必要はありません。

Java DB ドライバの JAR ファイルは `derbyclient.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」: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」:
 - **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、デフォルトでは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。

- **databaseName** - データベースの名前を指定します。
- **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のみ)。
- 「URL」 : `jdbc:jdbc:// serverName:portNumber /databaseName;create=true`
;`create=true` の部分は、データベースが存在していない場合にそのデータベースを作成する場合にのみ含めます。

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 `smbase.jar`、`smdb2.jar`、および `smutil.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」 :
 - **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。
- 「URL」 : `jdbc:sun:db2:// serverName: portNumber;databaseName=databaseName`

Oracle 8.1.7 および 9.x 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 `smbase.jar`、`smoracle.jar`、および `smutil.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」 :

- **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。
 - **SID** - 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。
- 「URL」 : jdbc:sun:oracle:// *serverName*[: *portNumber*][;SID=*databaseName*]

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 `smbase.jar`、`smsqlserver.jar`、および `smutil.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」 :
 - **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。
- 「URL」 : jdbc:sun:sqlserver:// *serverName*[: *portNumber*]

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 `smbase.jar`、`smsybase.jar`、および `smutil.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」 :

- **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。
- 「URL」 : jdbc:sun:sybase:// *serverName*[: *portNumber*]

IBM DB2 8.1 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。次のように環境変数を設定します。

```
LD_LIBRARY_PATH=/usr/db2user/sqlllib/lib:${Java EE.home}/lib
DB2DIR=/opt/IBM/db2/V8.1
DB2INSTANCE=db2user
INSTHOME=/usr/db2user
VWSPATH=/usr/db2user/sqlllib
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」 :
 - **user** - 必要に応じて設定します。
 - **password** - 必要に応じて設定します。
 - **databaseName** - 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。

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 jconn2.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.jdbc2.jdbc.SybDataSource  
com.sybase.jdbc2.jdbc.SybXADataSource
```

- 「プロパティ」:
 - **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。
 - **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。
 - **BE_AS_JDBC_COMPLIANT_AS_POSSIBLE** - true に設定します。
 - **FAKE_METADATA** - true に設定します。

MySQL Type 4 Driver

この MySQL ドライバの JAR ファイルは `mysql-connector-java-version-bin-g.jar` です。たとえば、`mysql-connector-java-5.0.5-bin-g.jar` などです。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」: MySQL
- 「データソースクラス名」: 次のいずれかを指定します。

```
com.mysql.jdbc.jdbc2.optional.MysqlDataSource
```

- 「プロパティ」:
 - **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。
 - **port** - 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。
 - **databaseName** - 必要に応じて設定します。
 - **URL** - グローバルトランザクションを使用する場合は、`serverName`、`port`、および `databaseName` の代わりにこのプロパティを設定できます。

```
jdbc:mysql://host:port/database
```

Oracle 8.1.7 および 9.x 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」:
 - **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 です。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。

- **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。
- **port** - データベースサーバーのポート番号を指定します。
- **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。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」:
 - **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。
 - **port** - 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」:
 - **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。
 - **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。

Oracle 8.1.7 および 9.x 用の Oracle Thin Type 4 ドライバ

この Oracle ドライバの JAR ファイルは `ojdbc14.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、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」:
 - **user** - 必要に応じて設定します。
 - **password** - 必要に応じて設定します。
 - **URL** - 次の構文を使用して完全なデータベース URL を指定します。

```
jdbc:oracle:thin:[user/password]@host[:port]/service
```

次に例を示します。

```
jdbc:oracle:thin:@localhost:1521:customer_db
```

- **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 - グローバルトランザクションの復旧が正しく動作するためには、トランザクションサービスで `oracle-xa-recovery-workaround` プロパティを設定する必要があります。詳細については、[157 ページの「特定のデータベースに関する問題の回避方法」](#)を参照してください。

このドライバを使用する場合、2000 バイトを超えるデータを列に挿入することはできません。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。

Oracle 8.1.7 および 9.x 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 を介して共有ライブラリが利用可能であること、および ORACLE_HOME プロパティ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」:
 - **user** - 必要に応じて設定します。
 - **password** - 必要に応じて設定します。
 - **URL** - 次の構文を使用して完全なデータベース URL を指定します。

```
jdbc:oracle:oci:[user/password]@host[:port]/service
```

次に例を示します。

```
jdbc:oracle:oci:@localhost:1521:customer_db
```

- **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。

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.IfxDDataSource  
com.informix.jdbcx.IfXXADataSource
```

- 「プロパティ」:
 - **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。
 - **user** - 必要に応じて設定します。
 - **password** - 必要に応じて設定します。
 - **databaseName** - 必要に応じて設定します。これは任意指定です。
 - **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。

CloudScape 5.1 Type 4 ドライバ

この CloudScape ドライバの JAR ファイルは db2j.jar、db2jtools.jar、db2jcvview.jar、jh.jar、db2jcc.jar、および db2jnet.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」:Cloudscape
- 「データソースクラス名」:com.ibm.db2.jcc.DB2DataSource
- 「プロパティ」:
 - **user** - 必要に応じて設定します。
 - **password** - 必要に応じて設定します。
 - **databaseName** - 必要に応じて設定します。

Java Message Service (JMS) リソースの設定

Communications Server は、Message Queue ソフトウェアを統合して、Communications Server に Java Message Service (JMS) API を実装します。基本的な JMS API 管理タスクには、Communications Server の管理コンソールを使用します。Message Queue クラスターの管理などの高度なタスクには、*MQ-as-install/imq/bin* ディレクトリに用意されたツールを使用します。Message Queue の管理の詳細については、『Message Queue Administration Guide』を参照してください。

この章では、Java Message Service (JMS) API を使用するアプリケーションのリソースを設定する方法について説明します。次の節で構成されています。

JMS リソース

JMS (Java Message Service) API は、次の 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。アプリケーションがプログラムでほかの JMS オブジェクトを作成できるようにするオブジェクトです。
- 送信先。メッセージのリポジトリとして機能します。

オブジェクトは管理された上で作成され、その作成方法は JMS の実装に固有になります。Communications Server で、次のタスクを実行します。

- 接続ファクトリリソースを作成することによって、接続ファクトリを作成します
- 次の 2 つのオブジェクトを作成して、送信先を作成します
 - 物理的送信先
 - 物理的送信先を参照する送信先リソース

JMS アプリケーションは、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。作成するリソースを確認するには、アプリケーションを理解したり、アプリケーションの開発者の意見を確認したりすることをお勧めします。

接続ファクトリには次の3つのタイプがあります。

- 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の使用をお勧めします。

送信先には次の2種類があります。

- 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

『Java EE 5 Tutorial』のJMSに関する章では、これら2つの通信タイプと、JMSのほかの側面が説明されています (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html> を参照)。

リソースを作成する順序は重要ではありません。

Java EE アプリケーションでは、次の手順に従って `Communications Server` の配備記述子に接続ファクトリリソースと送信先リソースを指定します。

- 接続ファクトリ JNDI 名は `resource-ref` または `mdb-connection-factory` 要素に指定します。
- 送信先リソース JNDI 名は、メッセージ駆動型 Bean の `ejb` 要素と `message-destination` 要素に指定します。
- 物理送信先名は、Enterprise JavaBean 配備記述子の `message-driven` 要素または `message-destination-ref` 要素のいずれかにある `message-destination-link` 要素に指定します。さらに、`message-destination` 要素にも指定します。 `message-destination-ref` 要素は、新しいアプリケーションで推奨されない `resource-env-ref` 要素から置き換わります。 `Communications Server` 配備記述子の `message-destination` 要素で、物理送信先名と送信先リソース名をリンクします。

JMS リソースとコネクタリソースの関係

`Communications 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、`Communications Server` がコネクタリソースも自動的に作成します。コネクタリソースは、管理コンソールのツリービューで「コネクタ」ノードの下に表示されます。

ユーザーが作成する各 JMS 接続ファクトリに対して、`Communications 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。ユーザーが作成する個々の JMS 送信先に対して、`Communications Server` は管理オブジェクトリソースを作成します。ユーザーが JMS リソースを削除するときに、`Communications Server` はコネクタリソースを自動的に削除します。

「JMS リソース」ノードの代わりに管理コンソールの「コネクタ」ノードを使用して、JMS システムリソースアダプタ用のコネクタリソースを作成できます。詳細は、第7章「コネクタリソース」を参照してください。

JMS 接続ファクトリ

JMS 接続ファクトリは、アプリケーションがプログラムでほかの JMS オブジェクトを作成できるようにするオブジェクトです。これらの管理対象オブジェクトは、ConnectionFactory、QueueConnectionFactory、および TopicConnectionFactory インタフェースを実装します。Communications Server 管理コンソールを使用して、JMS 接続ファクトリを作成、編集、または削除できます。新しい JMS 接続ファクトリの作成では、ファクトリのコネクタ接続プールとコネクタリソースも作成します。

コマンド行ユーティリティを使用して、JMS 接続ファクトリを管理するには、`create-jms-resource`、`list-jms-resources`、または `delete-jms-resource` コマンドを使用します。

JMS 送信先リソース

JMS 送信先は、メッセージのリポジトリとして機能します。管理コンソールを使用して、JMS 送信先リソースを作成、変更、または削除できます。新しい JMS 送信先リソースを作成するには、「リソース」>「JMS リソース」>「送信先リソース」の順に選択します。「送信先リソース」ページで、次を指定できます。

- リソースの JNDI 名。JMS 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 `jms/` を使用することをお勧めします。次に例を示します。 `jms/Queue`
- リソースタイプ。 `javax.jms.Topic` または `javax.jms.Queue` です。
- 送信先リソースの追加プロパティ。これらのすべての設定と追加のプロパティの詳細については、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。

コマンド行ユーティリティを使用して、JMS 送信先を管理するには、`create-jms-resource` または `delete-jms-resource` コマンドを使用します。

ヒント - `asadmin create-jms-resource` コマンドの `addresslist` プロパティを (`host:mqport,host2:mqport,host3:mqport` の形式で) 指定するには、`\\` を使用して `:` をエスケープします。たとえば、`host1\\:mqport,host2\\:mqport,host3\\:mqport` のようになります。

エスケープ文字の使用の詳細については、`asadmin(8)` のマニュアルページを参照してください。

JMS 物理送信先

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。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 の設定プロパティで指定した期限が切れると効力を失います。

管理コンソールから物理送信先を作成するには、「設定」>「Java メッセージサービス」>「物理送信先」の順に選択します。「新規」をクリックし、「新しい物理送信先」ページで、物理送信先の名前を指定し、送信先のタイプを `topic` または `queue` から選択します。「新しい物理送信先」ページのフィールドとプロパティの詳細については、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。

コマンド行ユーティリティを使用して、JMS 物理送信先を管理するには、`create-jmsdest`、`flush-jmsdest`、または `delete-jmsdest` コマンドを使用します。

JMS プロバイダのプロパティの設定

管理コンソールの「Java メッセージサービス」ページを使用して、すべての JMS 接続で使用するプロパティを設定します。管理コンソールで、「設定」>「Java メッセージサービス」の順に選択します。「Java メッセージサービス」ページで、次の JMS サービスの一般設定を制御できます。

- 「起動タイムアウト」の間隔を選択します。これは、起動が中止されないように JMS サービスが開始するのを Communications Server が待機する時間を示します。
- 「JMS サービス」のタイプを選択します。これは、JMS サービスをローカルホストで管理するか、リモートホストで管理するかを指定します。
- 「起動引数」を指定して、JMS サービスの起動をカスタマイズします。
- 「再接続」チェックボックスにチェックマークを付けて、接続が失われたときに JMS サービスがメッセージサーバーまたは AddressList で指定したアドレスのリストに再接続を試みるように指定します。

- 「再接続間隔」を秒数で指定します。これは、AddressList で指定した各アドレスおよびリストの次のアドレスへの試行に適用されます。間隔が短すぎると、ブローカーにリカバリする時間が与えられません。間隔が長すぎると、再接続が許容できない遅延を示す場合があります。
- 再接続の試行回数を指定します。このフィールドに、クライアントランタイムがリストの次のアドレスを試行する前に、AddressList に指定した各アドレスへの接続(または再接続)を試行する回数を入力します。
- デフォルトの JMS ホストを選択します。
- 「アドレスリストの動作」ドロップダウンリストで、接続の試行を AddressList で指定したアドレスの順序(priority)で行うか、またはランダムな順序(random)で行うかを選択します。
- 「アドレスリストの繰り返し」フィールドで、接続の確立または再確立のために、AddressList を介して JMS サービスが反復する回数を入力します。
- デフォルト以外のスキームまたはサービスを使用する場合は、「MQ スキーム」および「MQ サービス」フィールドに、Message Queue アドレススキーム名と Message Queue 接続サービス名を入力します。

これらのすべてのプロパティの値は実行時にも更新できます。ただし、更新された値を取得するのは、プロパティの更新後に作成された接続ファクトリのみです。既存の接続ファクトリは元のプロパティ値のままになります。

コマンド行ユーティリティを使用して JMS プロバイダを管理するには、set または jms-ping コマンドを使用します。

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を使用するには、リモートサーバーにアクセスする接続を作成する AddressList プロパティを使用して、接続ファクトリリソースを作成します。

外部 JMS プロバイダ

JMS の汎用リソースアダプタは Java EE Connector 1.5 リソースアダプタで、IBM Websphere MQ、Tibco EMS、Sonic MQ などの外部 JMS プロバイダの JMS クライアントライブラリをラップできるため、任意の JMS プロバイダを Java EE アプリケーションサーバーに統合します。アダプタは、Communications Server 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。

JMSの汎用リソースアダプタの設定

Communications Server の管理ツールを使用して、JMS の汎用リソースアダプタを配備および設定できます。この節では、Communications Server で JMS の汎用リソースアダプタを設定する方法を説明します。

全体として、JMS プロバイダが XA をサポートするかどうかを、リソースアダプタの設定により選択できます。さらに、JMS プロバイダで可能な統合のモードを選択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を JMS プロバイダの JNDI ツリーに設定し、汎用リソースアダプタがそれらを検索し、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。

管理コンソールまたはコマンド行を使用して、リソースアダプタを設定できます。これは、他のリソースアダプタの設定と変わりありません。

汎用リソースアダプタの設定

リソースアダプタを配備する前に、Communications Server で JMS クライアントライブラリを使用できるようにします。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も含まれている場合があります。このような場合は、これらのネイティブライブラリも JVM から使用できるようにします。

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。
3. コネクタリソースを作成します。
4. 管理対象オブジェクトリソースを作成します。
5. Communications Server のセキュリティーポリシーを次のように変更します。
 - `sjsas_home/domains/domain1/config/server.policy` を変更して、`java.util.logging.LoggingPermission "control"` を追加します。
 - `sjsas_home/lib/appclient/client.policy` を変更して、`permission javax.security.auth.PrivateCredentialPermission "javax.resource.spi.security.PasswordCredential ^\\\"^\"";"read":` を追加します。

リソースアダプタのプロパティー

次の表に、リソースアダプタの作成時に使用するプロパティーを示します。

プロパティ名	有効な値	デフォルト値	説明
ProviderIntegration Mode	javabean/jndi	javabean	リソースアダプタとJMSクライアントの統合のモードを指定します。
ConnectionFactory ClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging. ConnectionFactory	なし	JMSクライアントの javax.jms.ConnectionFactory 実装のクラス名。ProviderIntegrationModeがjavabeanの場合に使用します。
QueueConnectionFactoryClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging. QueueConnectionFactory	なし	JMSクライアントの javax.jms.QueueConnectionFactory 実装のクラス名。ProviderIntegrationModeがjavabeanの場合に使用します。
TopicConnectionFactoryClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging. TopicConnectionFactory	なし	JMSクライアントの javax.jms.TopicConnectionFactory 実装のクラス名。ProviderIntegrationModeがjavabeanと指定されている場合に使用します。
XAConnectionFactory ClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging. XAConnectionFactory	なし	JMSクライアントの javax.jms.ConnectionFactory 実装のクラス名。ProviderIntegrationModeがjavabeanと指定されている場合に使用します。
XAQueueConnectionFactoryClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging. XAQueueConnectionFactory	なし	JMSクライアントの javax.jms.XAQueueConnectionFactory 実装のクラス名。ProviderIntegrationModeがjavabeanと指定されている場合に使用します。

プロパティ名	有効な値	デフォルト値	説明
XATopicConnectionFactoryClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging. XATopicConnectionFactory	なし	JMSクライアントの javax.jms.XATopicConnectionFactory 実装のクラス名。ProviderIntegrationMode が javabean の場合に使用します。
TopicClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging.Topic	なし	JMSクライアントの javax.jms.Topic 実装のクラス名。ProviderIntegrationMode が javabean の場合に使用します。
QueueClassName	アプリケーション サーパークラスパスで使用可能なクラス名。たとえば次のようになります。 com.sun.messaging.Queue	なし	JMSクライアントの javax.jms.Queue 実装のクラス名。ProviderIntegrationMode が javabean と指定されている場合に使用します。
SupportsXA	True/false	FALSE	JMSクライアントがXAをサポートするかどうかを指定します。
ConnectionFactoryProperties	コンマで区切られた名前と値のペア	なし	javabean プロパティ名とJMSクライアントのConnectionFactoryの値を指定します。ProviderIntegrationMode が javabean である場合にのみ必要です。
JndiProperties	コンマで区切られた名前と値のペア	なし	JMSプロバイダのJNDIへの接続に使用するJNDIプロバイダのプロパティを指定します。ProviderIntegrationMode が jndi である場合にのみ使用します。

プロパティ名	有効な値	デフォルト値	説明
CommonSetter MethodName	メソッド名	なし	一部のJMSベンダーが管理対象オブジェクトにプロパティを設定するために使用する一般的なsetterメソッド名を指定します。ProviderIntegrationModeがjavabeanである場合にのみ使用します。Sun Java System Message Queueの場合、このプロパティの名前はsetPropertyになります。
UserName	JMSユーザーの名前	なし	JMSプロバイダに接続するためのユーザー名。
Password	JMSユーザーのパスワード	なし	JMSプロバイダに接続するためのパスワード。

プロパティ名	有効な値	デフォルト値	説明
RMPolicy	ProviderManagedまたはOnePerPhysicalConnection	Provider Managed	<p>XAResourceのisSameRMメソッドは、トランザクションマネージャーで、2つのXAResourcesによって表されたリソースマネージャーインスタンスが同じであるかどうかを判断するために使用されます。RMPolicyをProviderManaged(デフォルト値)に設定すると、JMSプロバイダが、汎用リソースアダプタのRMPolicyラッパーおよびXAResourceラッパーがisSameRM呼び出しをメッセージキュープロバイダのXAリソースの実装に単に委任するかどうかをJMSプロバイダが決定します。これは、大半のメッセージキュー製品で最適に機能するべきです。</p> <p>IBM MQ シリーズなどのXAResourceの一部の実装では、物理接続ごとに1つのリソースマネージャーを使用します。これにより、単一のトランザクションで同じキューマネージャーに対するインバウンド通信とアウトバウンド通信がある(たとえばMDBが送信先に応答を送信するなど)場合に問題が発生します。RMPolicyがOnePerPhysicalConnectionに設定されている場合、汎用リソースアダプタのXAResourceラッパーの実装のisSameRMは、ラッパー対象オブジェクトに委任する前に、両方のXAResourcesが同じ物理接続を使用するかどうかを確認します。</p>

ManagedConnectionFactory プロパティ

ManagedConnectionFactory プロパティは connector-connection-pool の作成時に指定します。リソースアダプタの作成時に指定されたすべてのプロパティは、ManagedConnectionFactory でオーバーライドできます。ManagedConnectionFactory でのみ使用可能な追加のプロパティを次に示します。

プロパティ名	有効な値	デフォルト値	説明
ClientId	有効なクライアント ID	なし	JMS 1.1 仕様に指定されている ClientID
ConnectionFactory JndiName	JNDI 名	なし	JMS プロバイダの JNDI ツリーにバインドされた接続ファクトリの JNDI 名。管理者は、JMS プロバイダ自体にすべての接続ファクトリプロパティ (clientId を除く) を指定するようにしてください。このプロパティ名は ProviderIntegrationMode が jndi の場合にのみ使用されます。
ConnectionValidation Enabled	true/false	FALSE	true に設定した場合、リソースアダプタは例外リスナーを使用して、接続の例外をキャッチし、CONNECTION_ERROR_OCCURED イベントをアプリケーションサーバーに送信します。

管理対象オブジェクトリソースのプロパティ

このプロパティは、管理対象オブジェクトリソースの作成時に指定します。リソースアダプタのすべてのプロパティは、管理対象リソースオブジェクトでオーバーライドできます。管理対象オブジェクトでのみ使用可能な追加のプロパティを次に示します。

プロパティ名	有効な値	デフォルト値	説明
DestinationJndiName	JNDI 名	なし	JMS プロバイダの JNDI ツリーにバインドされた送信先の JNDI 名。管理者は JMS プロバイダ自体にすべてのプロパティを指定するようにしてください。このプロパティ名は ProviderIntegrationMode が jndi の場合にのみ使用されます。
DestinationProperties	コンマで区切られた名前と値のペア	なし	JMS クライアントの送信先の javabeen プロパティ名と値を指定します。ProviderIntegrationMode が javabeen である場合にのみ必要です。

有効化仕様プロパティ

このプロパティは、`activation-config-properties`としてMDBのSun固有の配備記述子に指定されています。リソースアダプタのすべてのプロパティは、有効化仕様でオーバーライドできます。有効化仕様のみで使用できるプロパティを、次の表に示します。

プロパティ名	有効な値	デフォルト値	説明
MaxPoolSize	整数	8	同時メッセージ配信用のリソースアダプタによって、内部で作成されるサーバーセッションプールの最大サイズ。これはMDBオブジェクトの最大プールサイズに等しくなるべきです。
MaxWaitTime	整数	3	リソースアダプタは、その内部プールからサーバーセッションを取得するために、このプロパティに指定された秒単位の時間を待機します。この制限を超えると、メッセージ配信が失敗します。
SubscriptionDurability	持続性または非持続性	非持続性	JMS 1.1仕様に指定されているSubscriptionDurability
SubscriptionName		なし	JMS 1.1仕様に指定されているSubscriptionName
MessageSelector	有効なメッセージセクタ	なし	JMS 1.1仕様に指定されているMessageSelector
ClientID	有効なクライアントID	なし	JMS 1.1仕様に指定されているClientID
ConnectionFactoryJndiName	有効なJNDI名	なし	JMSプロバイダで作成された接続ファクトリのJNDI名。この接続ファクトリはリソースアダプタが接続を作成し、メッセージを受け取るために使用します。ProviderIntegrationModeがjndiと設定されている場合にのみ使用します。

プロパティ名	有効な値	デフォルト値	説明
DestinationJndiName	有効な JNDI 名	なし	JMS プロバイダで作成された送信先の JNDI 名。この送信先は、リソースアダプタが接続を作成し、メッセージを受け取るために使用します。ProviderIntegrationMode が jndi と設定されている場合にのみ使用します。
DestinationType	javax.jms.Queue または javax.jms.Topic	NULL	MDB が待機する送信先のタイプ。
Destination Properties	コマンドで区切られた名前と値のペア	なし	JMS クライアントの送信先の javabean プロパティ名と値を指定します。ProviderIntegrationMode が javabean である場合にのみ必要です。
RedeliveryAttempts	整数		MDB でメッセージによって実行時例外が発生した場合には、メッセージが配信される回数。
RedeliveryInterval	秒単位での時間		MDB でメッセージによって実行時例外が発生した場合には、配信を繰り返す間隔。
SendBadMessages ToDMD	true/false	False	配信の試行回数を超えた場合に、リソースアダプタがデッドメッセージ送信先にメッセージを送信すべきかどうかを示します。
DeadMessage Destination JndiName	有効な JNDI 名。	なし	JMS プロバイダによって作成された送信先の JNDI 名。これは、デッドメッセージのターゲット送信先です。これは ProviderIntegrationMode が jndi の場合にのみ使用します。
DeadMessage Destination ClassName	送信先オブジェクトのクラス名。	なし	ProviderIntegrationMode が javabean の場合に使用します。
DeadMessage Destination Properties	コマンドで区切られた名前と値のペア	なし	JMS クライアントの送信先の javabean プロパティ名と値を指定します。これは ProviderIntegrationMode が javabean の場合にのみ必要です。

プロパティ名	有効な値	デフォルト値	説明
ReconnectAttempts	整数		例外リスナーが接続時のエラーをキャッチした場合に試行される再接続の回数。
ReconnectInterval	秒単位での時間		再接続の間隔。

JavaMail リソースの設定

Communications Server には JavaMail API が含まれています。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。この API は、メールとメッセージングアプリケーションを構築するための、プラットフォームにもプロトコルにも依存しないフレームワークを提供します。JavaMail API には、電子メッセージの読み取りと送信のための機能が備えられています。サービスプロバイダは特定のプロトコルを実装します。JavaMail 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。メールサーバー機能はないため、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、J2EE プラットフォームの一部としても使用できます。

Communications Server には、JavaMail API とともに、アプリケーションコンポーネントがインターネットを介して電子メール通知を送信したり IMAP や POP3 メールサーバーからの電子メールを読んだりするための JavaMail サービスプロバイダが含まれています。

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。

ここでは、次の内容について説明します。

JavaMail セッションの作成

Communications Server で JavaMail を設定して使用するには、Communications Server 管理コンソールでメールセッションを作成します。これにより、サーバー側コンポーネントとアプリケーションは、JavaMail サービスに割り当てられたセッションプロパティを使用して、JNDI を使用して JavaMail サービスにアクセスできます。メールセッションを作成する際に、管理コンソールで、メールホスト、トラン

サ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ため、JavaMail を使用するコンポーネントはこれらのプロパティを設定する必要がありません。Application Server は単一のセッションオブジェクトを作成して、JNDI を介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に使用できるようにするため、電子メールを大量に使用するアプリケーションで役立ちます。

管理コンソールを使用して JavaMail セッションを作成するには、「リソース」、「JavaMail セッション」の順に選択します。次のように JavaMail 設定を指定します。

- 「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。次に例を示します。 `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が提供されていない場合、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` プロパティが提供されていない場合、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。次の形式で入力します。 `username@host.domain`
- 「説明」: コンポーネントの説明を入力します。
- 「状態」: このときメールセッションを有効にしない場合は、「有効」チェックボックスを選択解除します。

さらに、メールプロバイダが、デフォルト以外のストアやトランスポートプロトコルを使用するように設定し直した場合にのみ、次の詳細設定を定義します。

- 「ストアプロトコル」: 使用するストアオブジェクト通信の方法を定義します。デフォルトでは、ストアプロトコルは `imap` です。
- 「ストアプロトコルクラス」: 目的のストアプロトコルを実装するストア通信メソッドクラスを指定します。デフォルトでは、ストアプロトコルクラスは `com.sun.mail.imap.IMAPStore` です。
- 「転送プロトコル」: トランスポート通信の方法を指定します。デフォルトでは、トランスポートプロトコルは `smtp` です。
- 「転送プロトコルクラス」: トランスポートクラスの通信メソッドを定義します。デフォルトでは、トランスポートプロトコルクラスは `com.sun.mail.smtp.SMTPTransport` です。

- 「デバッグ」: このメールセッションのプロトコルトレースなど、追加のデバッグ出力を有効にするには、このチェックボックスにチェックマークを付けます。JavaMail のログレベルが「FINE」以上に設定されている場合は、デバッグ出力が生成され、それがシステムログファイルに記録されます。
- 「追加プロパティ」: プロトコル固有のホストやユーザー名のプロパティなどアプリケーションに必要なプロパティを作成します。定義するプロパティ名の先頭には、`mail-`プレフィックスを必ず使用してください。プロパティ名の先頭が `mail-` でない場合、プロパティは無視されます。

JNDI リソース

Java Naming and Directory Interface (JNDI) は、さまざまな種類のネーミングおよびディレクトリサービスにアクセスするための Application Programming Interface (API) です。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。

JNDI は、Java Naming and Directory Interface API の略語です。API を呼び出すことにより、アプリケーションは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。リソースとは、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、Communications 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。新しいリソースを作成すると、JNDI に新しい名前とオブジェクトのバインドが入力されます。

この節の内容は、次のとおりです。

- 91 ページの「Java EE ネームサービス」
- 92 ページの「ネーミング参照とバインディング情報」
- 93 ページの「カスタムリソースの使用」
- 93 ページの「外部 JNDI リポジトリとリソースの使用」

Java EE ネームサービス

JNDI 名は人間が理解しやすいオブジェクトの名前です。これらの名前は、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。Communications Server は、起動時に設定ファイルから情報を読み込み、JNDI データベース名を自動的に名前空間に追加します。

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。このアプリケーションコンポーネントの環境を使用することにより、アプリケーションコンポーネントのソースコードにアクセスしたり、このソースコードを変更したりせずに、アプリケーションコンポーネントをカスタマイズできます。

Java EE コンテナはアプリケーションコンポーネントの環境を実装し、この環境をアプリケーションコンポーネントのインスタンスに JNDI ネーミングコンテキストとして提供します。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せん。

- 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、実行時にその環境に用意されるとアプリケーションコンポーネントが想定する、環境エントリのすべて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、環境ネーミングコンテキストを利用できるようにします。アプリケーションコンポーネントのインスタンスは、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。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。次の節で説明する例では、リソース参照名は `jdbc/SavingsAccountDB` です。

リソースの JNDI 名とリソース参照名と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SEMBL することが可能になります。

次の表には、Communications Server が使用する Java EE リソースの JNDI 検索と関連する参照が一覧表示されています。

表 6-1 JNDI ルックアップと関連する参照

JNDI ルックアップ名	関連する参照
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

カスタムリソースの使用

カスタムリソースはローカルの 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。両方のリソースのタイプが、ユーザー指定のファクトリクラス要素、JNDI 名属性などを必要とします。この節では、Java EE リソースの JNDI 接続ファクトリリソースを設定し、これらのリソースにアクセスする方法を説明します。

Communications Server では、list-jndi-entries はもとより、リソースを作成、削除、一覧表示することができます。

外部 JNDI リポジトリとリソースの使用

通常、Communications Server で実行中のアプリケーションは、外部 JNDI リポジトリに格納されているリソースにアクセスする必要があります。たとえば、Java スキーマのように一般的な Java オブジェクトを LDAP サーバーに格納できます。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。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。

外部JNDIリソースの使用例を示します。

```
<resources>
  <!-- external-jndi-resource element specifies how to access J2EE resources
  -- stored in an external JNDI repository. The following 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>
```

コネクタリソース

この章では、エンタープライズ情報システム (EIS) へのアクセスに使用されるコネクタの設定方法について説明します。この章で説明する内容は次のとおりです。

- 95 ページの「コネクタの概要」
- 96 ページの「コネクタ接続プールの管理」
- 103 ページの「コネクタリソースの管理」
- 106 ページの「管理対象オブジェクトリソースの管理」

コネクタの概要

コネクタモジュールとは、アプリケーションがエンタープライズ情報システム (EIS) と対話することを可能にする Java EE コンポーネントであり、リソースアダプタとも呼ばれます。EIS ソフトウェアには、さまざまな種類のシステムが含まれています。ERP (Enterprise Resource Planning)、メインフレームトランザクション処理、および非リレーショナルデータベースなどです。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 (リソースアダプタ) を指定します。

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。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。アプリケーションはリソースの JNDI 名を検索してその位置を確定します。JNDI の詳細については、「JNDI 名とリソース」の節を参照してください。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。

Communications Server は、コネクタモジュール (リソースアダプタ) を使って JMS を実装します。「JMS リソースとコネクタリソースの関係」を参照してください。

コネクタ接続プールの管理

- 96 ページの「コネクタ接続プールを作成する」
- 97 ページの「コネクタ接続プールを編集する」
- 100 ページの「コネクタ接続プールの詳細属性を編集する」
- 102 ページの「コネクタ接続プールのプロパティを編集する」

▼ コネクタ接続プールを作成する

始める前に プールを作成する前に、プールに関連するコネクタモジュール(リソースアダプタ)を配備します。新しいプールに指定する値は、配備されるコネクタモジュールによって異なります。

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。「コネクタ接続プール」ノードを選択します。「コネクタ接続プール」ページで、「新規」をクリックします。
- 2 最初の「新しいコネクタ接続プール」ページで、次の設定を指定します。
 - a. 「名前」フィールドに、プールの論理名を入力します。
コネクタリソースを作成するときに、この名前を指定します。
 - b. 「リソースアダプタ」コンボボックスでエントリを選択します。
コンボボックスには、配備されたリソースアダプタ(コネクタモジュール)が表示されます。
- 3 「接続定義」コンボボックスで値を選択します。
コンボボックスの選択肢は、選択したリソースアダプタによって異なります。「接続定義」の属性は、リソースアダプタの ra.xml ファイル中の connection-definition 要素を表します。
- 4 「次へ」をクリックします。
- 5 「新しいコネクタ接続プール」ページで、次のタスクを実行します。
 - a. 「一般設定」領域で、値が正しいことを確認します。
 - b. 「プール設定」領域のフィールドは、デフォルト値をそのまま使用できます。
これらの設定は、あとで変更できます。97 ページの「コネクタ接続プールを編集する」を参照してください。

- c. 「追加のプロパティ」テーブルで、必要なプロパティを追加します。
 前の「新しいコネクタ接続プール」ページで、「接続定義」コンボボックスのクラスを選択しました。このクラスがサーバーのクラスパスにある場合、「追加のプロパティ」テーブルにはデフォルトのプロパティが表示されます。
- 6 「完了」をクリックします。

参考 同機能を持つ asadmin コマンド

```
create-connector-connection-pool
```

▼ コネクタ接続プールを編集する

- ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。「コネクタ接続プール」ノードを展開し、編集するコネクタ接続プールを選択します。
- 「コネクタ接続プールを編集」ページで、この接続プールの設定を変更できます。「一般設定」では、接続プールの説明を変更することもできます(省略可能)。
- 「プール設定」で、次の表に示すようにパラメータ値を指定できます。

パラメータ	説明
初期および最小プールサイズ	プール内の接続の最小数。この値は、プールを最初に作成したり、アプリケーションサーバーを起動したりするときの、プールに含まれる接続の数も指定します。
最大プールサイズ	プールに含まれる接続の最大数。
プールサイズ変更量	プールのサイズを、最大プールサイズおよび最小プールサイズに向けて拡大および縮小すると、一括処理でプールサイズが変更されます。この値は一括処理での接続の数を指定します。この値を過大に設定すると接続の作成と再利用が遅れ、過小に設定すると効率が落ちます。
アイドルタイムアウト	プールで接続がアイドル状態のままでいられる最長時間を指定します。この時間を過ぎると、接続はプールから削除されます。

最大待ち時間	接続を要求するアプリケーションが接続タイムアウトになるまでに待つ時間。
すべての障害で	「すべての接続を閉じる」のチェックボックスを選択している場合、1つの接続が失敗すると、アプリケーションサーバーはプールに含まれるすべての接続を閉じ、それらを再確立します。チェックボックスを選択していない場合は、使用されるときにだけ個々の接続が再確立されます。

トランザクションサポート	<p>「トランザクションサポート」リストを使用して、接続プールのトランザクションサポートのタイプを選択できます。選択したトランザクションサポートは、接続プールに関連付けられたリソースアダプタのトランザクションサポートを上書きし、下位互換性を維持します。つまり、リソースアダプタで指定されたレベルより低いトランザクションレベルや、リソースアダプタで指定された同じレベルのトランザクションレベルをサポートできますが、こより高いレベルは指定できません。トランザクションサポートには次のオプションがあります。「トランザクションサポート」メニューで「なし」を選択すると、リソースアダプタはリソースマネージャーのローカルトランザクションやJTA トランザクションをサポートせず、XAResource または LocalTransaction インタフェースを実装しません。JAXR リソースアダプタの場合は、「トランザクションサポート」メニューで「なし」を選択する必要があります。JAXR リソースアダプタは、ローカルまたはJTA のトランザクションをサポートしません。ローカルトランザクションサポートとは、リソースアダプタが LocalTransaction インタフェースを実装することでローカルトランザクションをサポートすることを意味します。ローカルトランザクションはリソースマネージャーの内部で管理され、外部トランザクションマネージャーを必要としません。XA トランザクションサポートとは、リソースアダプタが LocalTransaction と XAResource インタフェースを実装することで、ローカルおよびJTA トランザクションをサポートすることを意味します。XA トランザクションは、リソースマネージャーの外部にあるトランザクションマネージャーで管理および調整されます。ローカルトランザクションはリソースマネージャーの内部で管理され、外部トランザクションマネージャーを必要としません。</p>
接続検証	<p>「必須」チェックボックスにチェックを付けて、接続検証を有効にします。</p>

- 4 「保存」をクリックします。
- 5 すべての設定をデフォルト値に戻す場合は、「デフォルトを読み込み」をクリックします。

参考 `asadmin` コマンドを使用して、接続プールのプロパティを変更します。

`asadmin get` および `asadmin set` コマンドを使用して、接続プールのプロパティ値を表示および変更できます。

サーバーのすべてのコネクタ接続プールを表示するには、次のコマンドを使用します。

```
asadmin list domain.resources.connector-connection-pool.*
```

コネクタ接続プールのプロパティを表示するには、次のコマンドを使用します。

```
asadmin get domain.resources.connector-connection-pool.conectionpoolname.*
```

コネクタ接続プールのプロパティを設定するには、次のコマンドを使用します。

```
asadmin set domain.resources.connector-connection-pool.conectionpoolname
.pool-resize-quantity=3
```

▼ コネクタ接続プールの詳細属性を編集する

- 1 「詳細」タブで、コネクタ接続プールの詳細属性を編集します。

パラメータ	説明
最大で1回検証	経過するまでに1回しか接続が検証されない時間(秒)。これは、1つの接続による検証要求の数を減らすのに役立ちます。デフォルト値の0は、接続の検証が無効であることを示します。
リークタイムアウト	接続プール内の接続リークを追跡する時間(秒)。デフォルト値の0は、接続リーク追跡が無効であることを示します。接続リーク追跡を有効にすると、「リソースの監視」タブに接続リークの数に関する統計情報が表示されます。このタブを表示するには、「Application Server」→「監視」→「リソース」の順に選択します。
リーク再要求	このオプションを有効にすると、リーク接続の追跡が完了したあとで、リークした接続がプールにリストアされます。
作成再試行回数	新しい接続の作成に失敗した場合には行われる試行の回数。デフォルト値の0は、接続の再試行が行われないことを示します。
再試行間隔	接続作成の試行の間隔(秒数)を指定します。デフォルト値は10秒です。この属性は、「作成再試行回数」が1以上の場合にのみ使用されません。

使用時のみ関連付け	接続に対して操作が実行された時点ではじめて接続が関連付けられません。また、トランザクションが完了してコンポーネントメソッドが終了したときに関連付けが解除されるため、物理的な接続を再利用しやすくなります。デフォルト値は false です。
使用時のみ接続を登録	リソースが実際にメソッドで使用される場合にのみリソースをトランザクションに登録するには、このオプションを有効にします。
スレッドとの関連付け	接続をスレッドに関連付けるには、このオプションを有効にします。これにより、同じスレッドが接続を必要とするときは、そのスレッドにすでに関連付けられている接続を再利用できるため、プールから接続を取得するオーバーヘッドが発生しません。デフォルト値は false です。
接続のマッチング	プールに対する接続マッチングのオン/オフを切り替えるには、このオプションを使用します。プール内の接続が常に同じ種類であり、プールから選択した接続とリソースアダプタを照合する必要がない場合は、false に設定できます。デフォルト値は false です。
最大接続使用数	接続をプールで再利用する回数を指定します。指定された回数だけ再利用されると、その接続は閉じられます。これは、たとえば、文リークを回避するのに役立ちます。デフォルト値の 0 は、この属性が無効であることを示します。

- 2 「保存」をクリックして、変更内容を有効にします。
- 3 すべての設定をデフォルト値に戻す場合は、「デフォルトを読み込み」をクリックします。

参考 **asadmin** コマンドを使用して、接続プールのプロパティを変更します。

asadmin get および **asadmin set** コマンドを使用して、接続プールのプロパティ値を表示および変更できます。

サーバーのすべてのコネクタ接続プールを表示するには、次のコマンドを使用します。

```
asadmin list domain.resources.connector-connection-pool.*
```

コネクタ接続プールのプロパティを表示するには、次のコマンドを使用します。

```
asadmin get domain.resources.connector-connection-pool.conectionpoolname.*
```

コネクタ接続プールのプロパティを設定するには、次のコマンドを使用します。

```
asadmin set domain.resources.connector-connection-pool.conectionpoolname
.validate-atmost-once-period-in-seconds=3
```

▼ コネクタ接続プールのプロパティを編集する

- 1 「追加のプロパティ」タブで、既存のプールのプロパティを変更します。
指定するプロパティは、このプールで使用するリソースアダプタによって異なります。配備担当者がこのテーブルで指定する名前と値のペアを使用して、リソースアダプタのベンダーが定義したプロパティのデフォルト値を上書きすることができます。
- 2 「保存」をクリックして、変更内容を有効にします。

セキュリティマップの管理

コネクタ接続プールに関連するセキュリティマップを作成、編集、および削除できます。

▼ コネクタ接続プールのセキュリティマップを作成する

- 1 「コネクタ接続プールを編集」ページで、「セキュリティマップ」タブ付き区画をクリックします。
- 2 「新規」をクリックして、新しいセキュリティマップを作成します。
- 3 「新しいセキュリティマップ」ページで、セキュリティマップの名前、ユーザー名、およびパスワードを入力します。
- 4 「了解」をクリックします。

▼ コネクタ接続プールのセキュリティマップを編集する

- 1 「コネクタ接続プールを編集」ページで、「セキュリティマップ」タブ付き区画をクリックします。
- 2 セキュリティマップのリストで、編集するセキュリティマップの名前をクリックします。
- 3 必要に応じて値を変更し、「保存」をクリックします。

▼ コネクタ接続プールを削除する

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。
- 2 「コネクタ接続プール」ノードを選択します。
- 3 「コネクタ接続プール」ページで、削除するプールのチェックボックスを選択します。
- 4 「削除」をクリックします。

参考 同機能を持つ asadmin コマンド

```
delete-connector-connection-pool
```

▼ EIS アクセスをセットアップする

- 1 コネクタを配備(インストール)します。
- 2 コネクタの接続プールを作成します。
- 3 接続プールに関連付けるコネクタリソースを作成します。

コネクタリソースの管理

- [103 ページの「コネクタリソースを作成する」](#)
- [104 ページの「コネクタリソースを編集する」](#)
- [105 ページの「コネクタリソースを削除する」](#)
- [105 ページの「コネクタサービスを設定する」](#)

▼ コネクタリソースを作成する

コネクタリソース(データソース)は、アプリケーションに EIS への接続を提供します。

始める前に コネクタリソースを作成する前に、まずコネクタ接続プールを作成します。

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。

- 2 「コネクタリソース」ノードを展開します。
- 3 「コネクタリソース」ページで、「新規」をクリックします。
- 4 「新しいコネクタリソース」ページで、リソースの設定を指定します。
 - a. 「JNDI 名」フィールドに一意の名前を入力します (たとえば、eis/myERP)。
 - b. 「プール名」コンボボックスで、新しいコネクタリソースを作成する接続プールを選択します。
 - c. リソースを使用不可に変更するには、「ステータス」チェックボックスのチェックマークを外します。
デフォルトでは、リソースは使用可能 (有効) で作成されます。
 - d. 次のいずれかの操作を行います。
 - クラスタプロファイルを使用している場合は、ページに「ターゲット」領域が表示されます。「ターゲット」領域の「利用可能」フィールドで、コネクタリソースを作成するドメイン、クラスタ、またはサーバーインスタンスを選択し、「追加」をクリックします。コネクタリソースを、「選択」フィールドに表示されているドメイン、クラスタ、またはサーバーインスタンスに配備しない場合は、フィールドでいずれかの対象を選択して「削除」をクリックします。
 - ドメインを開発者プロファイルで実行している場合は、手順5に進みます。
- 5 「了解」をクリックします。

参考 同機能を持つ asadmin コマンド

```
create-connector-resource
```

▼ コネクタリソースを編集する

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。
- 2 「コネクタリソース」ノードを展開します。
- 3 編集するコネクタリソースのノードを選択します。「コネクタリソースを編集」ページで、接続の設定を変更できます。

- 4 次のいずれかの操作を行います。
 - クラスタプロファイルまたはエンタープライズプロファイルを使用している場合は、「ターゲット」タブ付き区画が表示されます。「ターゲット」タブ付き区画では、各ターゲットのコネクタリソースを有効または無効にすることができます。「ターゲットの管理」をクリックして、コネクタリソースを配備するターゲットを変更できます。
 - 開発者プロファイルを使用している場合は、次の手順に進みます。
- 5 「保存」をクリックして、編集内容を適用します。

▼ コネクタリソースを削除する

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。
- 2 「コネクタリソース」ノードを選択します。
- 3 「コネクタリソース」ページで、削除するリソースのチェックボックスを選択します。
- 4 「削除」をクリックします。

参考 同機能を持つ `asadmin` コマンド

```
delete-connector-resource
```

▼ コネクタサービスを設定する

「コネクタサービス」画面を使用して、このクラスタまたはサーバーインスタンスに配備されたすべてのリソースアダプタに対して、コネクタコンテナを設定できます。

- 1 ツリーで「設定」を選択します。
- 2 クラスタプロファイルを使用している場合は、設定するインスタンスを選択します。
 - 特定のインスタンスを設定するには、そのインスタンスの設定ノードを選択します。たとえば、デフォルトインスタンス `server` の場合は、`server-config` ノードを選択します。

- 今後作成するインスタンスのデフォルトの設定を変更するには、**default-config** ノードを選択します。
- 3 「コネクタサービス」ノードを選択します。
 - 4 「シャットダウンタイムアウト」フィールドに、シャットダウンのタイムアウトを秒単位で指定します。

コネクタモジュールのインスタンスで `ResourceAdapter.stop` メソッドが完了するまでに、アプリケーションサーバーが待機する最大の時間(秒単位)を、整数値で入力します。指定したシャットダウンタイムアウトより時間がかかるリソースアダプタは、アプリケーションサーバーから無視され、シャットダウンの手順が継続されます。デフォルトのシャットダウンタイムアウトは 30 秒です。クラスタまたはサーバーインスタンスに配備されたリソースアダプタの、デフォルトのシャットダウンタイムアウトを選択する場合は、「デフォルトを読み込み」をクリックします。

管理対象オブジェクトリソースの管理

リソースアダプタ(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、リソースアダプタ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。オブジェクトは管理対象に指定することができます。つまり、管理者により設定可能です。オブジェクトを設定するには、「新しい管理オブジェクトリソース」または「管理オブジェクトリソースを編集」ページで、名前と値のプロパティーペアを追加します。管理対象オブジェクトリソースを作成するときに、その管理対象オブジェクトを JNDI 名に関連付けます。

管理対象オブジェクトリソースを作成、編集、および削除するには、管理コンソールで「リソース」、「管理オブジェクトリソース」の順にクリックします。管理対象オブジェクトリソースの管理手順の詳細については、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。

- [106 ページの「管理対象オブジェクトリソースを作成する」](#)
- [108 ページの「管理対象オブジェクトリソースを編集する」](#)
- [108 ページの「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リソースを作成する

リソースアダプタ(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、リソースアダプタ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。オブジェクトは管理対象に指定することができます。つまり、管理者により設定可能です。オブジェクトを設定するには、「新しい管理オブジェクトリソース」または「管理オブジェクトリソースを編

集」 ページで、名前と値のプロパティペアを追加します。管理対象オブジェクトリソースを作成するときに、その管理対象オブジェクトを JNDI 名に関連付けます。

Communications Server は、リソースアダプタを使用して JMS を実装します。作成された JMS の送信先ごとに、Communications Server は自動的に管理対象オブジェクトリソースを作成します。

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。
- 2 「管理オブジェクトリソース」ノードを展開します。
- 3 「管理オブジェクトリソース」ページで、「新規」をクリックします。
- 4 「管理オブジェクトリソース」ページで、次の設定を指定します。
 - a. 「JNDI 名」フィールドに、リソースを識別する一意の名前を入力します。
 - b. 「リソースタイプ」フィールドに、リソースの Java タイプを入力します。
 - c. 「リソースアダプタ」コンボボックスで、管理対象オブジェクトを含むリソースアダプタを選択します。
 - d. 「ステータス」チェックボックスを選択または選択解除して、リソースを有効または無効にします。
 - e. 名前と値のプロパティのペアを使用して管理対象オブジェクトを設定するには、「プロパティを追加します」をクリックします。
 - f. クラスタプロファイルを使用している場合は、ページ内の「ターゲット」領域で、「利用可能」フィールドから管理対象オブジェクトを作成するドメイン、クラスタ、またはサーバーインスタンスを選択し、「追加」をクリックします。
「選択」フィールドのドメイン、クラスタ、またはサーバーインスタンスで管理対象オブジェクトの配備を取り消すには、フィールドでドメイン、クラスタ、またはサーバーインスタンスを選択し、「削除」をクリックします。
 - g. 「完了」をクリックします。

参考 同機能を持つ asadmin コマンド

```
create-admin-object
```

▼ 管理対象オブジェクトリソースを編集する

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。
- 2 「管理オブジェクトリソース」ノードを展開します。
- 3 編集する管理対象オブジェクトリソースのノードを選択します。
- 4 「管理オブジェクトリソースの編集」ページで、「管理オブジェクトリソースを作成する」の手順で指定した値を変更します。
- 5 クラスタプロファイルを使用している場合は、「ターゲット」タブ付き区画で「ターゲットの管理」をクリックして、管理対象オブジェクトを配備するターゲットを編集します。
- 6 「保存」をクリックして、編集内容を適用します。

▼ 管理対象オブジェクトリソースを削除する

- 1 ツリーコンポーネントで、「リソース」ノードを展開し、次に「コネクタ」ノードを展開します。
- 2 「管理オブジェクトリソース」ノードを選択します。
- 3 「管理オブジェクトリソース」ページで、削除するリソースのチェックボックスを選択します。
- 4 「削除」をクリックします。

参考 同機能を持つ `asadmin` コマンド

`delete-admin-object`

SIP、Web、およびEJB コンテナ

コンテナは、アプリケーションコンポーネントに実行時サポートを提供します。アプリケーションコンポーネントは、コンテナのプロトコルとメソッドを使用して、サーバーが提供するほかのアプリケーションコンポーネントとサービスにアクセスします。Communications Server は、アプリケーションクライアントコンテナ、アプレットコンテナ、Web コンテナ、およびEJB コンテナを提供します。コンテナを示す図については、27 ページの「Communications Serverの概要」の節を参照してください。

この章では、次のコンテナについて説明します。

- 109 ページの「SIP Servlet コンテナ」
- 111 ページの「Web コンテナ」
- 112 ページの「EJB コンテナ」

SIP Servlet コンテナ

Communications Server は、SIP Servlet コンテナを提供し、SIP 準拠のアプリケーションをホストします。このコンテナには次の機能があります。

- SIP 要求を待機するネットワークエンドポイントを提供します。
- SIP Servlet のライフサイクルをホストおよび管理する環境を提供します。
- ホストするアプリケーションと、それらをホストする順序を決定します。
- Secure SIP プロトコル(トランスポートレイヤープロトコル TLS 上の SIPS)をサポートします。
- サーバー側ソケットリスナーに Grizzly NIO フレームワークを使用します。
- セキュリティにダイジェスト認証を実装します。
- 呼び出しフローをサポートします。呼び出しフローは、アプリケーション開発者および Application Server 管理者が、配備したアプリケーションの動作を監視するための機能です。

SIP コンテナのプロパティの編集

デフォルトドメインの SIP コンテナの設定を表示するには、次の `asadmin` コマンドを使用します。

```
get server.sip-container.*
```

一般的な出力は次のとおりです。

```
server.sip-container.external-address =
server.sip-container.external-sip-port = 5060
server.sip-container.external-sips-port = 5061p
```

デフォルトでは、外部アドレスは空です。SIP コンテナは、マシンのすべてのネットワークインタフェースを検査して適切なアドレスを見つけようとするヒューリスティックを備えます。クラスタ設定では、外部アドレスの変更が必須です。SIP コンテナの外部 SIP および SIPS ポート設定は、SIP リスナーの外部 SIP および SIPS ポート設定によりオーバーライドされます。

SIP コンテナのポート設定を変更するには、次の `asadmin` コマンドを使用します。

```
set server.sip-container.external-sip-port=5065
```

SIP コンテナのサブ要素は、`session-manager` と `session-properties` です。`store-properties` と `manager-properties` は、`session-manager` のサブ要素です。

管理コンソールを使用して SIP コンテナのプロパティを変更するには、「設定」ノードを選択し、設定名を選択して、「SIP コンテナ」ノードを選択します。「一般」、「セッションプロパティ」、「マネージャプロパティ」、「ストアプロパティ」のタブを使用して、SIP コンテナのプロパティを表示および変更できます。

次の CLI コマンドを使用して、SIP コンテナのサブ要素と属性を表示することもできます。

SIP コンテナのサブ要素を表示するには、`list server.sip-container.*` コマンドを使用します。

SIP コンテナの属性を表示するには、`get server.sip-container.*` コマンドを使用します。

SIP コンテナのセッションプロパティの編集

SIP コンテナのセッションプロパティの属性を表示するには、`get server.sip-container.session-config.session-properties.*` コマンドを使用します。

SIP コンテナのセッションプロパティの属性を設定するには、`get` および `set` コマンドを使用します。

```
get server.sip-container.session-config.session-properties.*
```

```
set
```

```
server.sip-container.session-config.session-properties.timeout-in-seconds=0
```

SIP コンテナのセッションマネージャプロパティの編集

SIP コンテナのセッションマネージャプロパティのサブ要素を表示するには、`list server.sip-container.session-config.session-manager.*` コマンドを使用します。

サブ要素は `store-properties` と `manager-properties` の 2 つです。

`store-properties` の属性を表示するには、`get`

```
server.sip-container.session-config.session-manager.store-properties.*
```

 コマンドを使用します。

`manager-properties` の属性を表示するには、`get`

```
server.sip-container.session-config.session-manager.manager-properties.*
```

 コマンドを使用します。

SIP コンテナのセッションマネージャプロパティの属性を設定するには、`get` および `set` コマンドを使用します。

```
get
```

```
server.sip-container.session-manager.manager-properties.reap-interval-in-seconds
```

```
set
```

```
server.sip-container.session-manager.manager-properties.reap-interval-in-seconds=2
```

SIP コンテナのすべてのプロパティについては、[TBDlink](#) を参照してください。

Web コンテナ

Web コンテナは、Web アプリケーションを稼働させる Java EE コンテナです。Web コンテナは、サーブレットと JSP (JavaServer Pages) の実行環境を開発者に提供することにより、Web サーバーの機能を拡張します。

EJB コンテナ

エンタープライズ Beans (EJB コンポーネント) は、ビジネスロジックを含む Java プログラミング言語サーバーコンポーネントです。EJB コンテナは、エンタープライズ Beans へのローカルアクセスとリモートアクセスを提供します。

エンタープライズ Beans には、次の 3 つのタイプがあります。セッション Beans、エンティティ Beans、およびメッセージ駆動型 Beans です。セッション Beans は一時的なオブジェクトやプロセスを表し、通常は 1 つのクライアントが使用します。エンティティ Beans は通常データベースに保持されている持続性データを表します。メッセージ駆動型 Beans は、メッセージを非同期でアプリケーションモジュールやサービスに渡すために使われます。

コンテナの機能は、Enterprise JavaBean を作成したり、ほかのアプリケーションコンポーネントが Enterprise JavaBean にアクセスできるように Enterprise JavaBean をネームサービスにバインドしたり、承認されたクライアントだけが Enterprise JavaBean メソッドにアクセスできるようにしたり、Bean の状態を持続的記憶領域に保存したり、Bean の状態をキャッシュしたり、必要に応じて Bean を活性化したり非活性化したりすることです。

セキュリティの設定

セキュリティとはデータの保護に関することであり、ストレージ内または伝送中のデータへの不正アクセスやそうしたデータの破損をどのようにして防止するか、ということです。Communications Server には、Java EE 標準に基づく動的で拡張可能なセキュリティアーキテクチャがあります。標準実装されているセキュリティ機能には、暗号化、認証と承認、および公開鍵インフラストラクチャーがあります。Communications Server は Java セキュリティ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。この章では、次の内容について説明します。

- 113 ページの「アプリケーションおよびシステムセキュリティについて」
- 114 ページの「セキュリティ管理用ツール」
- 115 ページの「パスワードのセキュリティ管理」
- 118 ページの「認証と承認について」
- 121 ページの「ユーザー、グループ、ロール、およびレルムについて」
- 125 ページの「信頼設定およびエンティティ」
- 126 ページの「証明書および SSL の概要」
- 130 ページの「ファイアウォールについて」
- 130 ページの「証明書ファイルについて」
- 131 ページの「JSSE (Java Secure Socket Extension) ツールの使用」

アプリケーションおよびシステムセキュリティについて

概して、2 種類のアプリケーションセキュリティがあります。

- 「プログラムによるセキュリティ」。開発者が記述したアプリケーションコードがセキュリティ動作を処理します。管理者が、このメカニズムを操作する必要はまったくありません。一般的に、プログラムによるセキュリ

ティアーは、Java EE コンテナで管理するのではなく、アプリケーションのセキュリティ設定をハードコード化するのでお勧めできません。

- 「宣言によるセキュリティ」。Communications Server のコンテナがアプリケーションの配備記述子によりセキュリティを処理します。宣言によるセキュリティは、配備記述子を直接または `deploytool` などのツールで編集することによって操作できます。配備記述子はアプリケーションの開発後に変更可能なので、宣言によるセキュリティの方が柔軟性に富んでいます。

アプリケーションによるセキュリティのほかに、Communications Server システムのアプリケーション全体に影響するシステムセキュリティもあります。

プログラムによるセキュリティはアプリケーション開発者により制御されるため、このドキュメントでは説明していません。宣言によるセキュリティについては、このドキュメントである程度説明しています。このドキュメントは、主にシステム管理者を対象としているため、システムセキュリティを中心に説明していません。

セキュリティ管理用ツール

Communications Server には次のセキュリティ管理用ツールがあります。

- 管理コンソール。サーバー全体のセキュリティ設定、ユーザー、グループ、レルムの管理、およびほかのシステム全体のセキュリティタスクの実行に使用するブラウザベースのツール。管理コンソールの概要については、[28 ページの「管理用ツール」](#)を参照してください。セキュリティタスクの概要については、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。
- `asadmin`。管理コンソールと同じタスクの多くを行うコマンド行ツール。管理コンソールからできない操作でも、`asadmin` を使用して操作できる場合があります。コマンドプロンプトまたはスクリプトのいずれかから `asadmin` コマンドを実行して、繰り返しのタスクを自動化します。`asadmin` の概要については、[28 ページの「管理用ツール」](#)を参照してください。

Java Platform, Standard Edition (Java SE) には、次の2つのセキュリティ管理用ツールがあります。

- `keytool`。デジタル証明書および鍵のペアの管理に使用するコマンド行ユーティリティ。 `keytool` は、`certificate` レルムのユーザー管理に使用します。
- `policytool`。システム全体の Java セキュリティポリシー管理に使用するグラフィカルユーティリティ。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 セキュリティツールの使用方法は、<http://java.sun.com/j2se/1.5.0/docs/tooldocs/#security> の『JDK Tools and Utilities』を参照してください。

エンタープライズプロファイルでは、NSS (Network Security Services) を実装した2つのセキュリティー管理ツールも利用可能です。NSSの詳細は、<http://www.mozilla.org/projects/security/pki/nss/> を参照してください。それらのセキュリティー管理ツールは次のとおりです。

- `certutil`。証明書および鍵データベースの管理に使用されるコマンド行ユーティリティー。
- `pk12util`。証明書または鍵データベースと PKCS12 形式のファイル間における鍵と証明書のインポートおよびエクスポートに使用されるコマンド行ユーティリティー。

パスワードのセキュリティー管理

Communications Server では、特定のドメインの仕様を収めるファイル `domain.xml` 内に、Message Queue ブローカのパスワードが、あらかじめ平文で格納されています。このパスワードを収めた `domain.xml` ファイルの要素は、`json-host` 要素の `admin-password` 属性です。このパスワードはインストール時には変更不可能なので、セキュリティーに重大な影響は与えません。

ただし、管理コンソールを使用してユーザーやリソースを追加し、そのユーザーやリソースにパスワードを割り当てた場合、パスワードの中には、たとえばデータベースにアクセスするためのパスワードなど、平文で `domain.xml` ファイルに記述されるものもあります。このようなパスワードを平文で `domain.xml` ファイルに保持すると、セキュリティー上の危険を引き起こす可能性があります。`admin-password` 属性やデータベースパスワードを含む `domain.xml` のすべてのパスワードを暗号化できます。セキュリティーパスワードを管理する手順は、次のトピックを参照してください。

- 115 ページの「[domain.xml ファイル内のパスワードの暗号化](#)」
- 116 ページの「[エンコード化されたパスワードを含むファイルの保護](#)」
- 116 ページの「[マスターパスワードの変更](#)」
- 117 ページの「[マスターパスワードとキーストアの操作](#)」
- 118 ページの「[管理パスワードの変更](#)」

domain.xml ファイル内のパスワードの暗号化

`domain.xml` ファイル内のパスワードを暗号化するには、次の手順を実行します。

1. `domain.xml` ファイルが格納されているディレクトリ (デフォルトでは `domain-dir/config`) から、次の `asadmin` コマンドを実行します。

```
asadmin create-password-alias --user admin alias-name
```

次に例を示します。

```
asadmin create-password-alias --user admin jms-password
```

パスワードプロンプトが表示されます。この場合は `admin` です。詳細については、`create-password-alias`、`list-password-aliases`、`delete-password-alias` コマンドのマニュアルページを参照してください。

2. `domain.xml` のパスワードの削除および置き換えを行います。これは、`asadmin set` コマンドを使用して行います。このような目的での `set` コマンドの使用例は、次のとおりです。

```
asadmin set --user admin server.jms-service.jms-host.  
default_JMS_host.admin-password='${ALIAS=jms-password}'
```

注-エイリアスのパスワードは、例に示すように単一引用符で囲みます。

3. 該当するドメインの Communications Server を再起動します。

エンコード化されたパスワードを含むファイルの保護

ファイルの中にはエンコード化されたパスワードを含むものがあり、ファイルシステムのアクセス権を使用しての保護が必要になります。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` への `--passwordfile` 引数を使用して、引数と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。

マスターパスワードの変更

マスターパスワード (MP) とは、全体で共有するパスワードです。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。このパスワードはセキュリティー全体の要なので、ユーザーが必要に応じて手動で入力したり、またはファイルに隠蔽したりすることができます。これは、システムで最高の機密データです。ユーザーは、このファイルを削除することで、強制的にマスターパスワードの入力を要求できます。マスターパスワードが変更されると、Java JCEKS タイプのキーストアであるマスターパスワードキーストアに再保存されません。

マスターパスワードの変更手順は次のとおりです。

1. ドメインの Communications Server を停止します。新旧のパスワードの入力を促す `asadmin change-master-password` コマンドを使用して、依存するすべての項目を再暗号化してください。次に例を示します。

```
asadmin change-master-password>
Please enter the master password>
Please enter the new master password>
Please enter the the new master password again>
```

2. Communications Server を再起動します。



注意 - この時点で、実行中のサーバーインスタンスを再起動してはいけません。対応するノードエージェントの SMP が変更されるまでは、決して実行中のサーバーインスタンスを再起動しないでください。SMP が変更される前にサーバーインスタンスを再起動すると、起動に失敗します。

3. 各ノードエージェントおよび関連するサーバーを 1 つずつ停止します。 `asadmin change-master-password` コマンドをもう一度実行してから、ノードエージェントおよび関連するサーバーを再起動してください。
4. すべてのノードエージェントで対応が終了するまで、次のノードエージェントで同様の作業を続けます。このようにして、継続的な変更作業が完了します。

マスターパスワードとキーストアの操作

マスターパスワードは、セキュリティ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でロックされます。マスターパスワードがデフォルトではない場合、`start-domain` コマンドにより、マスターパスワードが要求されます。正しいマスターパスワードが入力されると、ドメインが起動します。

ドメインに関連付けられたノードエージェントが作成されると、ノードエージェントはデータをドメインと同期化させます。その間に、キーストアも同期化されます。このノードエージェントによって制御されるサーバーインスタンスは、キーストアを開く必要があります。このストアは、ドメイン作成処理で作成されたストアと基本的に同一であるため、同一のマスターパスワードでのみ開くことができます。マスターパスワード自体は、同期化されることはなく、同期中にノードエージェントには転送されません。しかし、ローカルでノードエージェントが利用できるようにする必要があります。このため、ノードエージェントの作成または起動、あるいはその両方の場合に、マスターパスワードが要求され、かつドメインの作成や起動のときに入力したパスワードと同じものを入力する必要があります。マスターパスワードがドメインで変更された場合は、ドメインと関連付けられている各ノードエージェントでも同じ手順で変更する必要があります。

管理パスワードの変更

管理パスワードの暗号化については、115 ページの「パスワードのセキュリティー管理」を参照してください。管理パスワードの暗号化は強く推奨されています。管理パスワードを暗号化する前に変更する場合は、`change-admin-password` コマンドを使用してください。

管理コンソールを使用して管理パスワードを変更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。

認証と承認について

認証と承認は、アプリケーションサーバーセキュリティーの中心的な概念です。ここでは、認証と承認に関連する次の項目について説明します。

- 118 ページの「エンティティーの認証」
- 119 ページの「ユーザーの承認」
- 120 ページの「JACC プロバイダの指定」
- 120 ページの「認証および承認の決定の監査」
- 120 ページの「メッセージセキュリティーの設定」

エンティティーの認証

「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、認証はユーザー名とパスワードでユーザーがアプリケーションにログインすることを意味していますが、アプリケーションがサーバーのリソースを要求するとき、セキュリティー資格を提供する EJB を指す場合もあります。普通、サーバーやアプリケーションはクライアントに認証を要求しますが、さらにクライアントもサーバーに自らの認証を要求できます。双方向で認証する場合、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、Communications 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。

Communications Server は、4 種類の認証をサポートします。アプリケーションは、配備記述子で使用する認証タイプを指定します。

表 9-1 Communications Server の認証方法

認証方法	通信プロトコル	説明	ユーザー資格の暗号化
BASIC	HTTP (オプションで SSL)	サーバーのビルトインポップアップログインダイアログボックスを使用します。	SSL を使用しないかぎりありません。
FORM	HTTP (オプションで SSL)	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。	SSL を使用しないかぎりありません。
CLIENT-CERT	HTTPS (HTTP over SSL)	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。	SSL
DIGEST	HTTP および SIP	サーバーは暗号化された応答に基づいてクライアントを認証します。	SSL および TLS

シングルサインオンの確認

シングルサインオンとは、1つの仮想サーバーインスタンスの複数のアプリケーションがユーザー認証状態を共有することを可能にするものです。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、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。

デフォルトでシングルサインオンは、Communications Server に定義された仮想サーバーで有効です。

ユーザーの承認

いったんユーザーが認証されると、「承認」のレベルによってどのような操作が可能かが決まります。ユーザーの承認は、ユーザーの「ロール」に基づいています。たとえば、人事管理アプリケーションでは、管理者には社員全員の個人情報を見ることを承認し、社員には自身の個人情報だけを見ることを承認します。ロールの詳細については、121 ページの「ユーザー、グループ、ロール、およびレルムについて」を参照してください。

JACC プロバイダの指定

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ています。これによって、管理者は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せん。

デフォルトで、Communications 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。その他のサードパーティー製の JACC プロバイダを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。これは、標準 PAM (Pluggable Authentication Module) フレームワークの Java テクノロジーバージョンを実装しています。

認証および承認の決定の監査

Communications Server は、「監査モジュール」によってすべての認証および承認の決定の監査トレールを提供します。Communications 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。

メッセージセキュリティの設定

「メッセージセキュリティ」によって、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および応答をエンドツーエンドで認証できません。Communications Server は、SOAP レイヤーのメッセージセキュリティプロバイダを使用して、メッセージセキュリティを実装します。メッセージセキュリティプロバイダは、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。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。

このリリースには、2つのメッセージセキュリティプロバイダが付属しています。メッセージセキュリティプロバイダは、SOAP レイヤーの認証用に設定されます。設定可能なプロバイダには、ClientProvider と ServerProvider があります。

メッセージレイヤーセキュリティのサポートは、プラグイン可能な認証モジュールの形式で Communications Server とそのクライアントコンテナに統合されています。Communications Server では、メッセージレイヤーセキュリティはデフォルトで無効になっています。

メッセージレベルのセキュリティーは、Communications Server 全体または特定のアプリケーションあるいはメソッドに対して設定できます。Communications Server レベルのメッセージセキュリティーの設定については、[第10章「メッセージセキュリティーの設定」](#)を参照してください。アプリケーションレベルのメッセージセキュリティーの設定については、『Developer's Guide』を参照してください。

ユーザー、グループ、ロール、およびレルムについて

Communications Server は次のエンティティーに対して認証および承認ポリシーを実施します。

- [122 ページの「ユーザー」](#)：「Communications Server で定義される」個別の ID。一般に、ユーザーとは、人物、Enterprise JavaBeans などのソフトウェアコンポーネント、またはサービスを意味します。認証されたユーザーを「主体」と呼ぶ場合もあります。また、ユーザーが「被認証者」と呼ばれる場合もあります。
- [122 ページの「グループ」](#)：「Communications Server で定義される」一連のユーザー。共通の特性に基づいて分類されます。
- [122 ページの「ロール」](#)：アプリケーションによって定義される指定した承認レベル。ロールは錠を開ける鍵に例えられます。多くの人が鍵のコピーを所持している場合があります。錠は、だれがアクセスを求めるとにかかわらず、適切な鍵が使用される場合だけ対応します。
- [123 ページの「レルム」](#)：ユーザーとグループの情報、および関連するセキュリティー資格を含むリポジトリ。レルムは、「セキュリティーポリシードメイン」とも呼ばれます。

注-ユーザーおよびグループは Communications Server 全体で指定されますが、各アプリケーションは独自のロールを定義します。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。

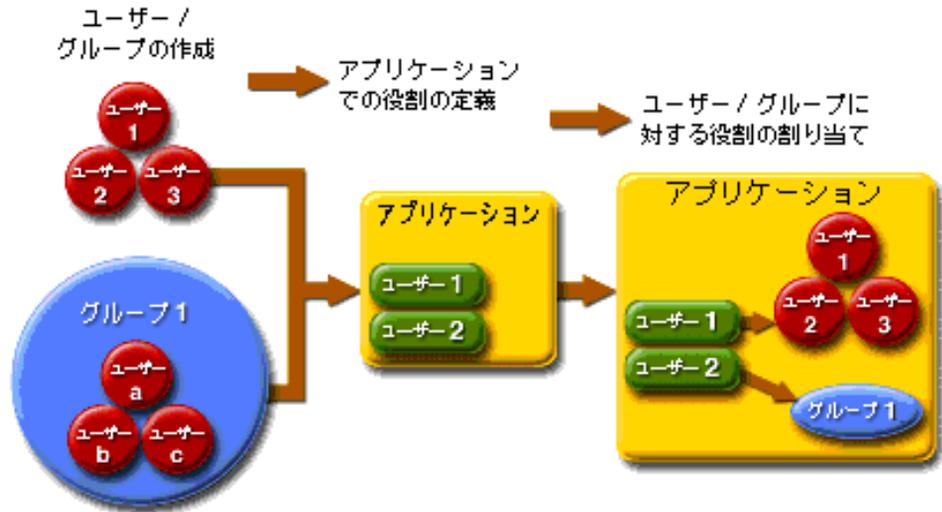


図9-1 ロールマッピング

ユーザー

「ユーザー」とは、Communications Serverによって定義された個人またはアプリケーションプログラムのIDです。ユーザーは1つのグループと関連付けできます。Communications 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。

グループ

「Java EEグループ」(または単にグループ)は、肩書きや顧客のプロファイルなど、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グループに属しますが、お得意様はPREFERREDグループに属します。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。

ロール

「ロール」は、ユーザーが、どのアプリケーションのどの部分にアクセスできるかと、何を実行できるかを定義します。つまり、ロールによってユーザーの承認レベルが決まります。

たとえば、人事アプリケーションの場合、電話番号とメールアドレスにはすべての社員がアクセスできますが、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少なくとも2つのロールが定義されます。employeeとmanagerです。そして、managerロールのユーザーだけに給与情報の表示が許可されます。

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、ロールとユーザーグループは異なります。たとえば、人事アプリケーションには、full-time、part-time、およびon-leaveなどのグループがありますが、これらすべてのグループのユーザーはemployeeロールに所属します。

ロールはアプリケーション配備記述子で定義されます。反対に、グループはサーバーおよびレルム全体に対して定義されます。アプリケーション開発者または配備担当者は、配備記述子により各アプリケーション内で、ロールを1つまたは複数のグループに割り当てます。

レルム

「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」とも呼ばれている「レルム」とは、サーバー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のことです。実際には、レルムとはサーバーがユーザーおよびグループの情報を格納するリポジトリです。

Communications Serverには、次の3つのレルムが事前に設定されています。file(初期のデフォルトレルム)、certificate、およびadmin-realmです。さらに、ldapレルム、JDBCレルム、solarisレルム、またはカスタムレルムも設定できます。アプリケーションは、その配備記述子でレルムを指定して使用できます。レルムを指定しない場合、Communications Serverはデフォルトレルムを使用します。

fileレルムでは、サーバーはユーザー資格をkeyfileという名前のファイルにローカルで格納します。管理コンソールを使用してfileレルムのユーザーを管理できます。

certificateレルムでは、サーバーはユーザー資格を証明書データベースに格納します。certificateレルムを使用する際、サーバーはHTTPプロトコルを使う証明書を使用してWebクライアントを認証します。証明書の詳細については、[126 ページの「証明書およびSSLの概要」](#)を参照してください。

admin-realmはFileRealmでもあり、管理者ユーザーの資格をadmin-keyfileという名前のファイルにローカルで格納します。fileレルムでユーザーを管理するのと同じ方法で、管理コンソールを使用してこのレルムのユーザーを管理してください。

ldapレルムでは、サーバーはDirectory ServerなどのLDAP (Lightweight Directory Access Protocol) サーバーからユーザー資格を取得します。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レルムのユーザーおよび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 レルムでは、サーバーはデータベースからユーザー資格を取得します。Communications Server は、データベース情報および設定ファイル内の有効化された JDBC レルムオプションを使用します。ダイジェスト認証では、`jdbcDigestRealm` を使用して JDBC レルムを JAAS コンテキストとして作成します。

`solaris` レルムでは、サーバーは Solaris オペレーティングシステムからユーザー資格を取得します。このレルムは Solaris 9 OS 以降でサポートされています。`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。

カスタムレルムとは、リレーショナルデータベースやサードパーティー製のコンポーネントなどその他のユーザー資格のリポジトリです。詳細については、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。

▼ Web、EJB、または SIP アプリケーションに対して JDBC レルムを設定する

Communications Server では、接続プールの代わりに JDBC レルムにユーザーの資格を指定できます。接続プールの代わりに JDBC 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。ユーザーの資格とは、ユーザーの名前とパスワードです。

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。通常の場合では、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 **手順 1** で作成したデータベース表にユーザーの資格を追加します。
データベース表にユーザーの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 JDBC レルムを作成します。
この作業には管理コンソール GUI を使用します。JDBC レルムの作成手順については、管理コンソール GUI のオンラインヘルプを参照してください。
- 4 **手順 3** で作成したレルムをアプリケーションのレルムとして指定します。
レルムを指定するには、アプリケーションの該当す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。
- **SIP** アプリケーションの場合は、sun-sip.xml ファイルを変更します。
詳細は、TBDlink を参照してください。
- **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 Communications Server 2.0 Developer's Guide](#)』の「[How to Set a Realm for an Application or Module](#)」を参照してください。

- 5 レルム内のユーザーにセキュリティロールを割り当てます。
ユーザーにセキュリティロールを割り当てるには、[手順 4](#) で変更した配備記述子に security-role-mapping 要素を追加します。

次の例は、ユーザー Calvin にセキュリティロール Employee を割り当てる security-role-mapping 要素を示しています。

```
<security-role-mapping>
  <role-name>Employee</role-name>
  <principal-name>Calvin</principal-name>
</security-role-mapping>
```

信頼設定およびエンティティ

ドメインとホストの間に信頼関係を構成するように設定を作成できます。信頼のタイプ、信頼されるホストまたはドメイン、およびメッセージの送受信に対して信頼関係を作成するかどうかを指定できます。

信頼設定およびエンティティの作成

管理コンソールまたは CLI を使用して、アイデンティティアサーションの信頼を作成できます。

管理コンソールで、「設定」ノードを展開し、設定を選択します。「セキュリティ」ノードを展開して、「トラスト設定」をクリックし、「新規」をクリックします。

または、`asadmin create-trusted-entity` および `asadmin create-trust-config` コマンドを使用することもできます。このコマンドの詳細は、TBDlink を参照してください。

信頼できるエンティティ (信頼できるホストまたはドメイン) を信頼設定に関連付けたり、Trust Handler を選択してカスタム実装で信頼を決定することができます。

信頼設定およびエンティティの編集

信頼設定のプロパティを編集するには、管理コンソールを使用するか、list、get、およびset コマンドを次のように使用します。

ターゲットの信頼設定を一覧表示するには、list *config-name.security-service.identity-assertion-trust.** コマンドを使用します。

信頼設定の属性を取得するには、get *config-name.security-service.identity-assertion-trust.trust-config-name.** コマンドを使用します。

信頼設定のすべての属性については、TBDlink を参照してください。

設定中の信頼できるエンティティを一覧表示するには、list *config-name.security-service.identity-assertion-trust.trust-config-name.** コマンドを使用します。

信頼できるエンティティのプロパティを編集するには、管理コンソールを使用するか、list、get、およびset コマンドを次のように使用します。

信頼できるエンティティの属性を取得するには、get *config-name.security-service.identity-assertion-trust.trust-config-name.trusted-entity.trusted-entity-name.** コマンドを使用します。

信頼できるエンティティの属性を設定するには、set *config-name.security-service.identity-assertion-trust.trust-config-name.trusted-entity.trusted-entity-name.ip-address=121.x.x.x* コマンドを使用します。

信頼できるエンティティのすべての属性については、TBDlink を参照してください。

証明書およびSSLの概要

この節では、次の項目について説明します。

- 126 ページの「デジタル証明書について」
- 128 ページの「SSL (Secure Sockets Layer) について」

デジタル証明書について

「デジタル証明書」(または単に証明書)とは、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つのエンティティ間の安全で機密保護された通信を可能にします。

個人により使用される個人証明書やSSL (Secure Sockets Layer) テクノロジーでサーバーとクライアント間の安全なセッションを確立するために使用されるサーバー証明書など、さまざまな種類の証明書があります。SSLの詳細については、128 ページの「[SSL \(Secure Sockets Layer\) について](#)」を参照してください。

証明書は「公開鍵暗号化」に基づき、意図した受信者だけが解読できるようデジタルの「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。鍵は数学的に関連しているので、1つの鍵で暗号化されたデータは、そのペアのもう1つの鍵でだけ復号化ができます。

証明書とはパスポートのようなものです。所有者を識別し、その他の重要な情報を提供します。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。いったんCAが証明書に署名すると、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。パスポートが写真とその所有者についての個人情報を結び付けるように、証明書は公開鍵とその所有者についての情報を結び付けます。

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。
- 証明書が発行された CA の名前。
- 有効期限の日付。

デジタル証明書は、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。

証明書チェーンについて

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。

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)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。

SSL (Secure Sockets Layer) について

「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。WebアプリケーションはHTTPS(HTTP over SSL)を使用します。HTTPSは、サーバーとクライアント間のセキュアで機密保護された通信を確保するため、デジタル証明書を使用します。SSL接続では、クライアントとサーバーの両方が送信前にデータを暗号化し、受信するとそれを復号化します。

クライアントのWebブラウザがセキュアなサイトに接続する場合、次のように「SSLハンドシェイク」が行われます。

- 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。
- 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。
- 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。
- 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。
- 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。

ハンドシェイクの後、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。Communications Server は、SSL (Secure Sockets Layer) 3.0 および TLS (Transport Layer Security) 1.0 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、セキュアな接続を受け付ける各外部インタフェースまたはIPアドレスの証明書を、Communications Server が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、デジタル証明書がインストールされるまで実行されません。[133 ページの「keytool コーティリティーを使って証明書を生成する」](#)の手順に従って、WebサーバーがSSL用に使用できるデジタル証明書を設定してください。

暗号化方式について

「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。SSL および TLS プロトコルから暗号化方式を選択してください。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であるもっとも強力な暗号化方式を使用します。そのため、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の使用

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 ハンドシェイクは、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。

ファイアウォールについて

「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。この節では、一般的ないくつかのファイアウォールアーキテクチャーとその設定について説明します。ここでの情報は、主に Communications Server に関係するものです。特定のファイアウォールテクノロジーの詳細については、使用しているファイアウォールのベンダーのドキュメントを参照してください。

一般的には、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ファイアウォールを設定します。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 8181 に設定されている場合は、HTTPS 要求をポート 8181 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 リスナーポートを開きますが、これにはセキュリティ上のリスクが伴うので、使用しないことを強くお勧めします。

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 Communications 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。

証明書ファイルについて

Communications Server をインストールすると、内部テストに適した JSSE (Java Secure Socket Extension) または NSS (Network Security Services) 形式のデジタル証明書が生成されます。デフォルトでは、Communications Server は *domain-dir/config* ディレクトリの証明書データベースに、証明書情報を格納します。

- キーストアファイル。key3.db には、非公開鍵を含む Communications 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。パスワードを変更するには、`asadmin change-master-password` コマンドを使用します。

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 Communications Server キーストア内には、エイリアス `s1as` を持つ単一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 には、ほかのエンティティの公開鍵を含む Communications 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。信頼できる証明書には、通常、証明書発行局 (CA) の証明書も含まれています。

開発者プロファイルでは、サーバー側で、Communications Server は `keytool` を使用して証明書とキーストアを管理する JSSE 形式を使用します。クラスタおよびエンタープライズプロファイルでは、サーバー側で、Communications Server は `certutil` を使用して非公開鍵と証明書を格納する NSS データベースを管理する NSS を使用します。いずれのプロファイルでも、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) では、JSSE 形式を使用します。

デフォルトで、Communications 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。本稼動環境のために、証明書エイリアスを変更し、トラストストアにほかの証明書を追加し、キーストアおよびトラストストアファイルの名前と場所を変更する必要がある可能性があります。

証明書ファイルの場所の変更

開発用として提供されているキーストアファイルとトラストストアファイルは、`domain-dir/config` ディレクトリに格納されています。

証明書ファイルの新しい場所の値フィールドを追加または変更するには、管理コンソールを使用します。

```
-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/NSS-database-directory
```

ここで、`NSS-database-directory` は NSS データベースの場所です。

JSSE (Java Secure Socket Extension) ツールの使用

`keytool` を使用して、JSSE (Java Secure Socket Extension) デジタル証明書を設定および操作します。開発者プロファイルの場合、Communications Server はサーバー側で、JSSE 形式を使っ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) では、JSSE 形式を使用します。

J2SE SDK に同梱されている `keytool` を使用すれば、管理者は、公開鍵と非公開鍵のペアおよび関連する証明書を管理できます。さらに、ユーザーは、通信接続先の公開鍵を証明書の形式でキャッシュできます。

`keytool` を実行するには、J2SE の `/bin` ディレクトリがパスの中に設定されているか、またはツールへのフルパスがコマンド行に存在するように、シェルの環境を設

定する必要があります。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ユーティリティーの使用

次の例は、JSSEツールによる証明書処理に関する使用方法を示したものです。

- RSA鍵アルゴリズムを使ってタイプJKSのキーストア内に自己署名付き証明書を作成する。RSAはRSA Data Security, Inc.が開発した公開鍵暗号化技術です。この略語は、この技術の開発者であるRivest、Shamir、およびAdelmanを表していません。

```
keytool -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}
-dname ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

証明書を作成する別の例については、133ページの「[keytoolユーティリティーを使って証明書を生成する](#)」を参照してください。

- デフォルトの鍵アルゴリズムを使ってタイプJKSのキーストア内に自己署名付き証明書を作成する。

```
keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
${keystore.pass}
```

証明書に署名する例については、134ページの「[keytoolユーティリティーを使ってデジタル証明書に署名する](#)」を参照してください。

- タイプJKSのキーストアで利用可能な証明書を表示する。

```
keytool -list -v -keystore ${keystore.file} -storepass ${keystore.pass}
```

- タイプJKSのキーストア内の証明書情報を表示する。

```
keytool -list -v -alias ${cert.alias} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

- RFC/テキスト形式の証明書をJKSストア内にインポートする。証明書は、バイナリエンコーディングではなく、Internet RFC (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}
```

- タイプ JKS のキーストア内の証明書を PKCS7 形式でエクスポートする。「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}
```

- タイプ JKS のキーストア内の証明書を RFC/テキスト形式でエクスポートする。

```
keytool -export -noprompt -rfc -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

- タイプ JKS のキーストアから証明書を削除する。

```
keytool -delete -noprompt -alias ${cert.alias} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

キーストアから証明書を削除する別の例については、[135 ページの「keytool ユーティリティーを使って証明書を削除する」](#)を参照してください。

keytool ユーティリティーを使って証明書を生成する

keytool を使用して証明書の生成、インポート、およびエクスポートを行います。デフォルトでは、keytool 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。

1. 証明書を実行すべき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たディレクトリ (デフォルトでは *domain-dir/config*) 内で行います。これらのファイルの場所を変更する方法については、[131 ページの「証明書ファイルの場所の変更」](#)を参照してください。
2. 次の keytool コマンドを入力することで、キーストアファイル *keystore.jks* 内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以外の値に変更した場合には、前述のコマンドの *changeit* をその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「*slas*」です。

プロンプトが表示され、keytool が証明書の生成に使用するユーザーの名前、組織、およびその他の情報の入力を求められます。

3. 次の `keytool` コマンドを入力することで、生成された証明書をファイル `server.cer` (または `client.cer` でもよい) 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、[134 ページの「keytool ユーティリティを使ってデジタル証明書に署名する」](#)を参照してください。
5. トラストストアファイル `cacerts.jks` を作成し、そのトラストストアに証明書を追加するには、次の `keytool` コマンドを入力します。

```
keytool -import -v -trustcacerts
-alias keyAlias
-file server.cer
-keystore cacerts.jks
-keypass changeit
```

6. キーストアまたは非公開鍵のパスワードをデフォルト以外の値に変更した場合には、前述のコマンドの `changeit` をその新しいパスワードで置き換えてください。このツールは、証明書に関する情報を表示し、その証明書を信頼するかどうかをユーザーに尋ねます。
7. `yes` と入力し、続いて `Enter` キーを押します。すると、`keytool` から次のようなメッセージが表示されます。

```
Certificate was added to keystore
[Saving cacerts.jks]
```

8. `Communications Server` を再起動します。

keytool ユーティリティを使ってデジタル証明書に署名する

デジタル証明書の作成後、所有者はそれ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。認証に心配がない場合、たとえば、非公開のセキュアな通信だけがが必要な場合などは、CA 証明書の取得に必要な時間と費用を節約して、自己署名付き証明書を使用してください。

1. 証明書の鍵のペアを生成するため、CA の Web サイトの指示に従います。
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たディレクトリ (デフォルトでは `domain-dir/config` ディレクトリ) 内に、証明書を保存します。

[131 ページの「証明書ファイルの場所の変更」](#)を参照してください。

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. `keytool` を使用して、証明書をローカルのキーストア、および必要に応じてローカルの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には、前述のコマンドの `changeit` をその新しいパスワードで置き換えてください。

5. Communications Server を再起動します。

keytool ユーティリティーを使って証明書を削除する

既存の証明書を削除するには、`keytool -delete` コマンドを使用します。次に例を示します。

```
keytool -delete
-alias keyAlias
-keystore keystore-name
-storepass password
```


メッセージセキュリティの設定

この章の一部の内容は、セキュリティと Web サービスに関する基本概念の理解を前提としてます。この章では、Communications Server で Web サービスのメッセージレイヤーセキュリティを設定する方法について説明します。この章の内容は次のとおりです。

- 137 ページの「メッセージセキュリティの概要」
- 138 ページの「Communications Server のメッセージセキュリティの理解」
- 142 ページの「Web サービスのセキュリティ保護」
- 144 ページの「サンプルアプリケーションのセキュリティ保護」
- 144 ページの「メッセージセキュリティのための Communications Server の設定」
- 148 ページの「メッセージセキュリティの設定」

メッセージセキュリティの概要

「メッセージセキュリティ」を使用する場合、メッセージ内にセキュリティ情報が挿入され、その情報が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は、『Java EE 5.0 Tutorial』の「Security」の章で説明されているトランスポートレイヤーセキュリティとは異なり、メッセージトランスポートからメッセージ保護を分離して伝送後もメッセージを保護されたままにするために使用することができます。

「Web Services Security: SOAP Message Security (WS-Security)」は、米国 Sun Microsystems, Inc. を含むすべての主要な Web サービステクノロジプロバイダによって共同開発された、相互運用可能な Web サービスセキュリティを実現するための OASIS 国際標準です。WS-Security のメッセージセキュリティメカニズムは、SOAP 経由で送信される Web サービスメッセージを XML 暗号化と XML デジタル署名を使ってセキュリティ保護する、というものです。WS-Security 仕様には、X.509 証明書、SAML アサーション、ユーザー名/パスワードなどの各種セキュリティトークンを使って SOAP Web サービスメッセージの認証および暗号化を実現する方法が規定されています。

WS-Security 仕様は、<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf> で確認できます。

Communications Server のメッセージセキュリティーの理解

Communications Server は、Web サービスのクライアント側コンテナとサーバー側コンテナにおいて、WS-Security 標準に対する統合化されたサポートを提供します。この機能は統合化されているため、Communications Server のコンテナがアプリケーションに代わって Web サービスセキュリティーを適用します。また、そうしたセキュリティーで Web サービスアプリケーションを保護する際、アプリケーションの実装を変更する必要はありません。Communications Server は、これを実現する目的で、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、コンテナおよびコンテナ内に配備されたアプリケーションにバインドする機能を提供しています。

メッセージセキュリティーの責任の割り当て

Communications Server でメッセージセキュリティー設定の主要責任者として期待されるのは、138 ページの「システム管理者」ロールと139 ページの「アプリケーション配備担当者」ロールです。場合によっては、139 ページの「アプリケーション開発者」もその責任の一端を担うことがあります。通常は、システム管理者またはアプリケーション配備者のいずれかのロールが既存アプリケーションをセキュリティー保護し、開発者が関与することも、実装が変更されることもあります。次の各節では、各種ロールの責任を定義します。

- 138 ページの「システム管理者」
- 139 ページの「アプリケーション配備担当者」
- 139 ページの「アプリケーション開発者」

システム管理者

システム管理者は次の責任を負います。

- Communications Server 上のメッセージセキュリティープロバイダの設定。
- ユーザー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。
- 暗号化を使用し、バージョン 1.5.0 より前のバージョンの Java SDK を実行している場合の JCE (Java Cryptography Extension) プロバイダの設定。
- サンプルサーバーのインストール。ただし、これを行うのは、xms サンプルアプリケーションを使ってメッセージレイヤー Web サービスセキュリティーの使用方法を示す場合だけです。

システム管理者は、管理コンソールを使用してサーバーセキュリティの設定を管理し、コマンド行ツールを使用して証明書データベースを管理します。開発者プロファイルとクラスタプロファイルでは、証明書と非公開鍵はキーストア内に格納され、keytool を使って管理されます。一方、エンタープライズプロファイルでは、証明書と非公開鍵は NSS データベース内に格納され、certutil を使って管理されます。このドキュメントは主にシステム管理者を対象にしています。メッセージセキュリティタスクの概要については、144 ページの「メッセージセキュリティのための Communications Server の設定」を参照してください。

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責任を負います。

- 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アセンブリ時に指定(それらのポリシーが上流行程の役割(開発者またはプログラマ)によって指定されていなかった場合)。
- 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

アプリケーション開発者はメッセージセキュリティを有効にできますが、そのようにする責任はありません。メッセージセキュリティの設定をシステム管理者が行う場合、すべての Web サービスがセキュリティ保護されます。コンテナにバインドされている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、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の設定を行います。

アプリケーション開発者またはプログラマは次の責任を負います。

- アプリケーション固有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、その必要なポリシーがアプリケーションアセンブリで指定されているかどうかの確認。それにはアプリケーション配備担当者に連絡します。

セキュリティトークンとセキュリティメカニズムについて

WS-Security 仕様は、セキュリティトークンを使って SOAP Web サービスメッセージを認証および暗号化するための拡張可能なメカニズムを提供します。Communications Server とともにインストールされる SOAP レイヤーメッセージセキュリティプロバイダを使えば、ユーザー名/パスワードセキュリティトークンと X.509 証明書セキュリティトークンによる SOAP Web サービスメッセージの認証

と暗号化を行えます。Communications Server の今後のリリースでは、SAML サッションなどのほかのセキュリティトークンを採用したプロバイダも追加される予定です。

ユーザー名トークンについて

Communications Server は、SOAP メッセージ内で「ユーザー名トークン」を使ってメッセージ「送信者」の認証 ID を確立します。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、そのメッセージの送信者がそのトークンによって識別されるユーザーとして振る舞うことを許可されているかどうかを検証するために、その送信者がユーザーの秘密情報(パスワード)を知っているかどうかを確認します。

ユーザー名トークンを使用する場合、有効なユーザーデータベースを Communications Server 上に設定する必要があります。

デジタル署名について

Communications 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、呼び出し元 ID を確立します。トランスポートレイヤーセキュリティを使用する場合も、基本認証または SSL クライアント証明書認証を使う場合も、呼び出し元 ID を確立する方法はほぼ同様です。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、有効なキーストアおよびトラストストアファイルを Communications Server 上に設定する必要があります。このトピックの詳細については、[130 ページの「証明書ファイルについて」](#)を参照してください。

暗号化について

暗号化の目的は、対象読者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。公開鍵暗号方式に関して言えば、暗号化はメッセージを読み取ることができる関係者の 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 JCE プロバイダがインストールされている必要があります。このトピックの詳細については、[146 ページの「JCE プロバイダの設定」](#)を参照してください。

メッセージ保護ポリシーについて

メッセージ保護ポリシーは、要求メッセージ処理と応答メッセージ処理に対して定義され、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。ソース認証ポリシーは、メッセージを送信したエンティティ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は、メッセージを受信可能なエンティティの 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。プロバイダは、特定のメッセージセキュリティ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。要求と応答に対するメッセージ保護ポリシーが定義されるのは、特定のプロバイダがコンテナ内に設定される時です。また、アプリケーションまたは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操作の粒度でのポリシー) を設定することも可能です。いずれにせよ、メッセージ保護ポリシーを定義する場合、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は、サーバーのそれと一致する (等しい) 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方法の詳細は、『開発者ガイド』の「Securing Applications」の章を参照してください。

メッセージセキュリティ用語の解説

次に、このドキュメントで使用する用語について説明します。これらの概念については、144 ページの「メッセージセキュリティのための Communications Server の設定」も参照してください。

■ 認証レイヤー

「認証レイヤー」とは、認証処理を実行する必要があるメッセージレイヤーです。Communications Server は、SOAP レイヤーにおいて Web サービスメッセージセキュリティを適用します。

■ 認証プロバイダ

Communications Server のこのリリースでは、Communications 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を処理します。

- 「クライアント側プロバイダ」は、署名またはユーザー名/パスワードを使って要求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。Communications 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。
- 「サーバー側プロバイダ」は、受信した要求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、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
「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プロバイダを識別するために使用されます。「デフォルトサーバープロバイダ」は「デフォルトプロバイダ」とも呼ばれます。
- デフォルトクライアントプロバイダ
「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プロバイダを識別するために使用されます。
- 要求ポリシー
「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
「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。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。

Web サービスのセキュリティ保護

Communications Server 上に配備された Web サービスをセキュリティ保護するには、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。Communications 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。

Communications 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プロバイダが Communications 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、プロバイダにはある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。

Communications Server の管理インタフェースを使えば、既存のプロバイダをバインドして Communications Server のサーバー側コンテナから利用できるようにしたり、プロバイダが適用するメッセージ保護ポリシーを変更したり、別のメッセージ保護ポリシーを備えた新しいプロバイダ設定を作成したりできます。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設定でも、これと同様の管理操作を実行できます。それらについては、[150 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の有効化」](#) で定義しています。

Communications Server では、メッセージレイヤーセキュリティはデフォルトで無効になっています。Communications Server のメッセージレイヤーセキュリティを設定するには、[144 ページの「メッセージセキュリティのための Communications Server の設定」](#) の手順に従ってください。Communications Server 上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で保護するには、[149 ページの「メッセージセキュリティのためのプロバイダの有効化」](#) の手順に従ってください。

上記の手順 (Communications Server の再起動が必要な場合もあり) を実行し終わると、Communications Server 上に配備されたすべての Web サービスアプリケーションに Web サービスセキュリティが適用されるようになります。

アプリケーション固有の Web サービスセキュリティの設定

アプリケーション固有の Web サービスセキュリティ機能を (アプリケーション構築上で) 設定するには、その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 Communications Server 2.0 Developer's Guide](#)』の第 5 章「[Securing Applications](#)」を参照してください。

サンプルアプリケーションのセキュリティー保護

Communications Server には、`xms` という名前のサンプルアプリケーションが付属しています。`xms` アプリケーションは、J2EE 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。このサービスエンドポイントインタフェースには、単一の操作 `sayHello` が定義されています。この操作は、文字列引数を 1 つ受け取り、その呼び出し引数の前に `Hello` が付加された String を返します。

`xms` サンプルアプリケーションは、Communications 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目的で提供されています。サンプルに付属する手順では、Communications 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、WS-Security 機能をアプリケーションに直接バインドする方法 (143 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の設定」を参照) も示しています。

`xms` サンプルアプリケーションは次のディレクトリにインストールされます。
`as-install/samples/webservices/security/ejb/apps/xms/`。

`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、および実行に関する詳細については、『Developers' Guide』の「Securing Applications」の章を参照してください。

メッセージセキュリティーのための **Communications Server** の設定

- 144 ページの「要求および応答ポリシー設定のアクション」
- 146 ページの「その他のセキュリティー機能の設定」
- 146 ページの「JCE プロバイダの設定」

要求および応答ポリシー設定のアクション

次の表に、メッセージ保護ポリシーの設定と、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 10-1 メッセージ保護ポリシーと WS-Security SOAP メッセージセキュリティ処理との対応づけ

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 10-1 メッセージ保護ポリシーと WS-Security SOAP メッセージセキュリティ処理との対応づけ (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処理を一切行いません。

その他のセキュリティ機能の設定

Communications Server は、SOAP 処理レイヤー内に統合化されたメッセージセキュリティプロバイダを使用して、メッセージセキュリティを実装します。メッセージセキュリティプロバイダは、Communications Server のその他のセキュリティ機能に依存します。

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、暗号化技術を使用する場合は、JCE プロバイダを設定します。
2. JCE プロバイダの設定については、[146 ページの「JCE プロバイダの設定」](#)を参照してください。
3. ユーザー名トークンを使用する場合は、必要に応じてユーザーデータベースを設定します。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、このレルムに適切なユーザーデータベースを設定する必要があります。
4. 必要に応じて証明書と非公開鍵を管理します。

次の手順

Communications Server の機能の設定が完了し、メッセージセキュリティプロバイダがそれらの機能を使用できるようになると、Communications Server とともにインストールされたプロバイダを有効にできます。その手順については、[149 ページの「メッセージセキュリティのためのプロバイダの有効化」](#)を参照してください。

JCE プロバイダの設定

J2SE 1.4.x に付属している JCE (Java Cryptography Extension) プロバイダは RSA 暗号化をサポートしていません。通常、WS-Security で定義されている XML 暗号化は RSA 暗号化に基づいているため、WS-Security を使って SOAP メッセージを暗号化するには、RSA 暗号化をサポートする JCE プロバイダをダウンロードおよびインストールする必要があります。

注 - RSA は RSA Data Security, Inc. が開発した公開鍵暗号化技術です。この略語は、この技術の開発者である Rivest、Shamir、および Adelman を表しています。

Java SDK Version 1.5 以上で Communications Server を実行している場合は、JCE プロバイダは正しく設定されています。Java SDK Version 1.4.x で Communications Server を実行している場合は、次のように JCE プロバイダを JDK 環境の一部として静的に追加できます。

1. JCE プロバイダの JAR (Java ARchive) ファイルをダウンロードし、インストールします。
RSA 暗号化をサポートする JCE プロバイダのリストについては、http://java.sun.com/products/jce/javase_providers.html を参照してください。
2. JCE プロバイダの JAR ファイルを `java-home/jre/lib/ext/` にコピーします。
3. Communications Server を停止します。
Communications Server を停止せずにこの手順の最後で再起動した場合、JCE プロバイダは Communications Server に認識されません。
4. 任意のテキストエディタで `java-home/jre/lib/security/java.security` プロパティファイルを編集します。このファイルに、前述の手順でダウンロードした JCE プロバイダを追加します。

`java.security` ファイルに、このプロバイダを追加する詳細手順が含まれています。基本的には、類似のプロパティを持つ場所に次の形式の行を追加する必要があります。

```
security.provider.n=provider-class-name
```

この例では、*n* は、Communications Server がセキュリティプロバイダを評価する際に使用する優先順位を示します。追加した JCE プロバイダには、*n* を 2 に設定します。

たとえば、The Legion of the Bouncy Castle JCE プロバイダをダウンロードした場合は、次のような行を追加します。

```
security.provider.2=org.bouncycastle.jce.provider.  
    BouncyCastleProvider
```

Sun セキュリティプロバイダが、値 1 の最高の優先順位に設定されていることを確認してください。

```
security.provider.1=sun.security.provider.Sun
```

各レベルにセキュリティプロバイダがただ 1 つだけ設定されるように、ほかのセキュリティプロバイダのレベルを下位に調整します。

次に示す例は、必要な JCE プロバイダを提供し、既存のプロバイダを正しい位置に保持する `java.security` ファイルのサンプルです。

```
security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.
    BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.rsa.jca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
```

5. ファイルを保存して、閉じます。
6. Communications Server を再起動します。

メッセージセキュリティの設定

メッセージセキュリティを使用できるように Communications Server を設定する手順のほとんどは、管理コンソールまたは `asadmin` コマンド行ツールを使用するか、あるいはシステムファイルを手動で編集することで実現できます。一般に、システムファイルの編集はお勧めできません。なぜなら、Communications Server が適切に動作しなくなるような変更を間違えて施してしまう可能性があるからです。したがって、できるだけ、管理コンソールによる Communications Server の設定手順を最初に示し、その後に `asadmin` ツールコマンドによる手順を示しています。システムファイルを手動で編集する手順は、管理コンソールと `asadmin` に同等の方法が存在しない場合にだけ示しています。

メッセージレイヤーセキュリティのサポートは、プラグイン可能な認証モジュールの形式で Communications Server とそのクライアントコンテナに統合されています。Communications Server では、メッセージレイヤーセキュリティはデフォルトで無効になっています。次の各節では、メッセージセキュリティ設定とプロバイダを有効化、作成、編集、および削除する方法について、詳しく説明します。

- 149 ページの「メッセージセキュリティのためのプロバイダの有効化」
- 149 ページの「メッセージセキュリティプロバイダの設定」
- 150 ページの「メッセージセキュリティプロバイダの作成」
- 150 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の有効化」
- 151 ページの「アプリケーションクライアント設定の要求および応答ポリシーの設定」
- 152 ページの「詳細情報」

ほとんどの場合、上記の管理操作を実行したあとで Communications Server を再起動する必要があります。特に、操作実行時に Communications Server 上にすでに配備されていたアプリケーションに管理上の変更を適用する場合に Communications Server の再起動が必要となります。

メッセージセキュリティのためのプロバイダの有効化

Communications Server 上に配備された 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のデフォルトプロバイダを有効にする場合、Communications Server 上に配備された Web サービス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。クライアントが使用するプロバイダを有効にする方法については、150 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の有効化」を参照してください。

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。Communications 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、Communications Server 内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、メッセージレイヤーセキュリティ用に正しく設定されていることを確認する必要があります。

コマンド行ユーティリティを使用するには、次の手順に従います。

- デフォルトサーバープロバイダを指定するには、次のコマンドを実行します。

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

- デフォルトクライアントプロバイダを指定するには、次のコマンドを実行します。

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

メッセージセキュリティプロバイダの設定

プロバイダの再設定は、プロバイダタイプ、実装クラス、およびプロバイダ固有の設定プロパティを変更するために実行することもできますが、通常はメッセージ保護ポリシーを変更するために実行します。

コマンド行ユーティリティを使用して、応答ポリシーを設定する場合は、次のコマンドの request という単語を response に置き換えてください。

- 要求ポリシーをクライアントに追加して、認証元を設定します。

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_source=
sender | content
```

- 要求ポリシーをサーバーに追加して、認証元を設定します。

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_source=
sender | content
```

- 要求ポリシーをクライアントに追加して、認証受信者を設定します。

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=
before-content | after-content
```

- 要求ポリシーをサーバーに追加して、認証受信者を設定します。

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=
before-content | after-content
```

メッセージセキュリティープロバイダの作成

管理コンソールを使用して既存のプロバイダを設定するには、「設定」ノード > 設定するインスタンス > 「セキュリティー」ノード > 「メッセージセキュリティー」ノード > 「SOAP」ノード > 「プロバイダ」タブの順に選択します。

メッセージセキュリティープロバイダの作成方法については、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。

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。Communications Server がインストールされる時、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、有効化はされていない)。

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 Communications Server 固有の設定を変更します。

アプリケーションクライアント設定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設定の要求ポリシーを設定するには、[150 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 Communications Server 固有の設定を変更します。アプリケーションクライアント設定ファイル内で request-policy 要素と response-policy 要素を次のように追加することで、要求ポリシーを設定します。

その他のコードは参照用に用意されています。実際のインストールでは、その他のコードが若干異なっている可能性があります。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/applclient/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。これらの値をさまざまに組み合わせた結果を記述した表については、[144 ページ](#)の「[要求および応答ポリシー設定のアクション](#)」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/>
```

詳細情報

- Java 2 Standard Edition のセキュリティーに関する議論 (<http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html>)
- 『Java EE 5.0 Tutorial』の「Security」の章 (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>)
- 『管理ガイドの』の章
- 『開発者ガイド』の「Securing Applications」の章
- 『XML-Signature Syntax and Processing』のドキュメント (<http://www.w3.org/TR/xmlsig-core/>)
- 『XML Encryption Syntax and Processing』のドキュメント (<http://www.w3.org/TR/xmlenc-core/>)

診断サービスの設定

診断サービスを使用すると、サーバーやそのアプリケーションの実行時パフォーマンスをわかりやすい形式で表示および制御でき、障害の発生時にその障害を診断および特定できます。

この章で説明する内容は次のとおりです。

- 153 ページの「診断フレームワークとは」
- 153 ページの「診断サービスフレームワーク」

診断フレームワークとは

Application Server 診断フレームワークは、アプリケーションサーバーの標準ライフサイクル内で実行するサービスのセットを定義および実装するための監視フレームワークです。診断サービスでは、稼働しているサーバーやサーバーが配備するアプリケーションによって生成される診断データの定義、作成、収集、およびそれらのデータへのアクセスが可能です。

診断サービスフレームワーク

診断サービスは、アプリケーションサーバーインスタンスの設定の詳細を報告します。例外やパフォーマンスの問題、その他の予期しない結果が起きたときなど、アプリケーションサーバーの問題を診断するのに役立ちます。管理コンソールの「診断サービス」で、次の操作を実行できます。

- 「チェックサムの計算」：
appserver_install_dir/lib、appserver_install_dir/etc、および appserver_install_dir/bin ディレクトリにある Application Server のバイナリファイルのうち、選択したファイルのチェックサムを収集します。
- 「設定の確認」：domain.xml や server.policy などの設定ファイルを取り込みます。

- 「インストールログの取り込み」: Application Server のバージョン番号やパッチ ID、インストール時に生成されたログファイルの内容など、インストールに関する詳細情報。インストールディレクトリの絶対パスは、同じマシンに複数のインストール環境がある場合に、収集するインストールログファイルの判別に使用されます。config/asenv.conf の内容は、DAS のインストールフォルダとノードエージェントからコピーされます。

インストールに関する詳細情報は、ファイルベースでのインストールに関してのみ収集されます。

- 「システム情報の取り込み」: デフォルトでは、次のシステム情報が収集されません。
 - ネットワーク設定
 - OS の詳細
 - ハードウェア情報

ネイティブコードを使用して収集したデータは、Platform Edition の Application Server では利用できません。

- 「アプリケーション配備記述子の取り込み」:
ejb-jar.xml、sun-ejb-jar.xml、web.xml、sun-web.xml、sun-sip.xml などの配備記述子。
- 「ログレベル」:
- 「ログエントリ」:
生成される診断レポートに含めるログエントリの数。

診断レポートの生成

診断レポートは、管理コンソールの「診断」タブでの設定内容に基づいて生成されます。生成されたレポートでは、機密データは「機密プロパティ」テーブルにリストされているとおりに表示されます。

◆◆◆ 第 12 章

トランザクション

トランザクションを使用すると、1つ以上のステップがそれ以上分割不可能な作業単位 (Unit of Work) にまとめられるため、データの完全性と整合性が保証されます。この章で説明する内容は次のとおりです。

- 155 ページの「トランザクションについて」
- 157 ページの「トランザクションに関する管理コンソールタスク」

トランザクションについて

- 155 ページの「トランザクションとは」
- 156 ページの「Java EE テクノロジーのトランザクション」
- 157 ページの「特定のデータベースに関する問題の回避方法」

トランザクションとは

トランザクションは、すべて正常に完了することが必要なアプリケーションで、周到に用意された一連のアクションです。正常に完了しない場合、各アクションで行われたすべての変更が取り消されます。たとえば、当座預金から普通預金に資金を移動するのは次の手順を実行するトランザクションになります。

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. 当座預金に十分なお金が入っている場合は、当座預金の金額を借り方に記帳します。
3. その金額を普通預金口座の貸し方に記帳します。
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。

これらの手順のいずれかが失敗すると、先行する手順によって行われた変更がすべて取り消されます。当座預金口座と普通預金口座はこのトランザクションが始まる前と同じ状態になる必要があります。このイベントは「ロールバック」と呼ばれま

す。すべての手順が正常に完了すると、トランザクションは「コミット」状態になります。トランザクションはコミットかロールバックのどちらかで終了します。

関連項目

- 156 ページの「Java EE テクノロジーのトランザクション」
- 157 ページの「トランザクションの設定」

Java EE テクノロジーのトランザクション

トランザクション処理には、次の5つの処理が含まれます。

- トランザクションマネージャー
- Communications Server
- リソースマネージャー (複数可)
- リソースアダプタ (複数可)
- ユーザーアプリケーション

これらの各エンティティは、次に説明する API や機能を実装することにより、信頼性のあるトランザクション処理を実現しています。

- トランザクションマネージャーは、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。
- Communications Server は、トランザクション状態管理を含むアプリケーションランタイム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は、リソースアダプタを介して、リソースへのアプリケーションアクセスを提供します。リソースマネージャーは、特定の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。このインタフェースは、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、および回復作業を伝達する際に使用されます。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。
- リソースアダプタはシステムレベルのソフトウェアライブラリで、リソースマネージャーへ接続するためにアプリケーションサーバーまたはクライアントが使用します。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。そのようなリソースアダプタの一例として、JDBC ドライバが挙げられます。
- アプリケーションサーバー環境で動作するように開発されたトランザクションユーザーアプリケーションは、JNDI を使用してトランザクションデータソースおよびトランザクションマネージャー (オプション) を検索します。アプリケーションは、エンタープライズ Bean 用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を使用することがあります。

関連項目

- 155 ページの「トランザクションとは」
- 157 ページの「トランザクションの設定」

特定のデータベースに関する問題の回避方法

Communications Server には、次の JDBC ドライバの回復実装に関するいくつかの既知の問題に対する回避方法が用意されています。明示的に無効にしないかぎり、それらの回避方法が使用されます。

- Oracle thin ドライバ - XAResource.recover メソッドは、入力フラグに関係なく、繰り返し同じ未確定 Xid セットを返します。XA 仕様に従って、トランザクションマネージャは最初に 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。

Communications Server の回避方法を無効にするに

は、oracle-xa-recovery-workaround プロパティ値を false に設定します。プロパティの設定方法については、158 ページの「Communications Server のトランザクションからの回復方法を設定する」を参照してください。

注-これらの回避方法は、特定の JDBC ドライバに対応するものではありません。

トランザクションに関する管理コンソールタスク

Communications Server は、管理コンソールの設定に基づいてトランザクションを処理します。

トランザクションの設定

この節では、トランザクションの設定方法について説明します。

- 158 ページの「Communications Server のトランザクションからの回復方法を設定する」
- 159 ページの「トランザクションタイムアウト値を設定する」
- 159 ページの「トランザクションログの位置を設定する」
- 160 ページの「キーポイント間隔を設定する」

トランザクションに関する追加情報については、次の各節を参照してください。

- 155 ページの「トランザクションとは」
- 156 ページの「Java EE テクノロジーのトランザクション」

▼ **Communications Server** のトランザクションからの回復方法を設定する

トランザクションは、サーバークラッシュまたはリソースマネージャークラッシュのいずれかにより未完了になる可能性があります。これらの未完了トランザクションを完了させ、障害を回復させる必要があります。Communications 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。

復元を行なっている間に一部のリソースにアクセスできない場合は、トランザクションを回復しようとしてサーバーの再起動が遅れている可能性があります。

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、そのトランザクションは「特殊な結果判別」フィールドを使用してその結果を判別します。

- 1 ツリーコンポーネントで、「設定」ノードを選択します。
- 2 開発者プロファイルで非クラスタドメインを実行している場合は、次の手順に進みます。クラスタプロファイルでドメインを実行している場合は、設定するインスタンスを選択します。
 - 特定のインスタンスを設定するには、そのインスタンスの設定ノードを選択します。たとえば、デフォルトインスタンス `server` の場合は、`server-config` ノードを選択します。
 - すべてのインスタンスのデフォルト値を設定するには、`default-config` ノードを選択します。
- 3 「トランザクションサービス」ノードを選択します。
- 4 未完了なトランザクションの復元を有効にするには、「再起動時」フィールドで「回復」にチェックマークを付けます。
- 5 「再試行タイムアウト」フィールドに、**Communications Server** がアクセスできないサーバーに対して接続を試みる時間を秒単位で設定します。デフォルト値は **10 分 (600 秒)** です。
- 6 「特殊な結果判別」フィールドに、トランザクションでアクセスできないサーバーのポリシーを設定します。

このフィールドを「コミット」に設定する適切な理由がないかぎり、「特殊な結果判別」を「ロールバック」のままにしておきます。未確定なトランザクションのコミットは、アプリケーションのデータの整合性を損なう可能性があります。

- 7 必要なプロパティがあれば設定します。
「プロパティを追加」ボタンをクリックし、「名前」フィールドと「値」フィールドに名前と値を入力し、「名前」フィールドの横にあるボックスにチェックマークを付けて、そのプロパティを有効にします。
- 8 「保存」をクリックします。
- 9 **Communications Server** を再起動します。

▼ トランザクションタイムアウト値を設定する

デフォルトでは、サーバーはトランザクションをタイムアウトしないようになっています。つまり、サーバーはトランザクションの完了を待機し続けます。トランザクションのタイムアウト値を設定して、トランザクションが設定された時間内に完了しない場合、Communications Server はトランザクションをロールバックします。

- 1 ツリーコンポーネントで、「設定」ノードを選択します。
- 2 開発者プロファイルで非クラスタドメインを実行している場合は、次の手順に進みます。クラスタプロファイルでドメインを実行している場合は、設定するインスタンスを選択します。
 - 特定のインスタンスを設定するには、そのインスタンスの設定ノードを選択します。たとえば、デフォルトインスタンス `server` の場合は、`server-config` ノードを選択します。
 - すべてのインスタンスのデフォルト値を設定するには、`default-config` ノードを選択します。
- 3 「トランザクションサービス」ノードを選択します。
- 4 「トランザクションタイムアウト」フィールドに、トランザクションがタイムアウトする秒数を入力します。
トランザクションタイムアウトのデフォルト値は0秒です。これにより、トランザクションのタイムアウトは無効になります。
- 5 「保存」をクリックします。
- 6 **Communications Server** を再起動します。

▼ トランザクションログの位置を設定する

トランザクションログは、関連リソースのデータの整合性を維持して障害を回復するために、各トランザクションについての情報を記録します。トランザクションロ

グは、「トランザクションログの位置」フィールドで指定したディレクトリの tx サブディレクトリに保存されます。これらのログは人間が読み取れるものではありません。

- 1 ツリーコンポーネントで、「設定」ノードを選択します。
- 2 開発者プロファイルで非クラスタドメインを実行している場合は、次の手順に進みます。クラスタプロファイルでドメインを実行している場合は、設定するインスタンスを選択します。
 - 特定のインスタンスを設定するには、そのインスタンスの設定ノードを選択します。たとえば、デフォルトインスタンス server の場合は、server-config ノードを選択します。
 - すべてのインスタンスのデフォルト値を設定するには、default-config ノードを選択します。
- 3 「トランザクションサービス」ノードを選択します。
- 4 「トランザクションログの位置」フィールドに、トランザクションログの位置を入力します。

tx サブディレクトリが作成され、トランザクションログがそのディレクトリの下に保存されます。

デフォルト値は `${com.sun.aas.instanceRoot}/logs` です。`${com.sun.aas.instanceRoot}` 変数はインスタンスの名前であり、Communications Server インスタンスの起動時に設定されます。`${com.sun.aas.instanceRoot}` の値を表示するには、「実際の値」をクリックします。
- 5 「保存」をクリックします。
- 6 **Communications Server** を再起動します。

▼ キーポイント間隔を設定する

トランザクションログファイルは、キーポイント処理によって圧縮されます。キーポイント間隔とは、ログ上のキーポイント処理間のトランザクション数です。キーポイント処理によって、トランザクションログファイルのサイズを小さくすることができます。キーポイント間隔を大きくすると(例: 2048)、トランザクションログファイルが大きくなりますが、キーポイント処理が少なくなるのでパフォーマンスが向上する可能性があります。キーポイント間隔を小さくすると(例: 256)、ログファイルのサイズが小さくなりますが、キーポイント処理が多くなるので、パフォーマンスがわずかに低下します。

- 1 ツリーコンポーネントで、「設定」ノードを選択します。
- 2 開発者プロファイルで非クラスタドメインを実行している場合は、次の手順に進みます。クラスタプロファイルでドメインを実行している場合は、設定するインスタンスを選択します。
 - 特定のインスタンスを設定するには、そのインスタンスの設定ノードを選択します。たとえば、デフォルトインスタンス `server` の場合は、`server-config` ノードを選択します。
 - すべてのインスタンスのデフォルト値を設定するには、`default-config` ノードを選択します。
- 3 「トランザクションサービス」ノードを選択します。
- 4 「キーポイント間隔」フィールドに、キーポイント処理間のトランザクション数を入力します。
デフォルト値は 65536 です。
- 5 「保存」をクリックします。
- 6 **Communications Server** を再起動します。

HTTP サービスの設定

HTTP サービスは、Web アプリケーションの配備機能を提供する Communications Server のコンポーネントで、配備された Web アプリケーションに HTTP クライアントがアクセスできるようにします。これらの機能は、仮想サーバーと HTTP リスナーという 2 種類の関連オブジェクトによって提供されます。

この章では、次の内容について説明します。

- 163 ページの「仮想サーバー」
- 164 ページの「HTTP リスナー」

仮想サーバー

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ティングするためのオブジェクトで、仮想ホストとも呼ばれます。同一物理サーバーにホスティングされるすべての仮想サーバーは、その物理サーバーの IP (Internet Protocol) 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、Communications Server が稼動するサーバーを関連付けます。

注 - インターネットドメインと Communications Server の管理ドメインを混同しないでください。

たとえば、ある物理サーバーで次のドメインをホスティングすると仮定します。

www.aaa.com
www.bbb.com
www.ccc.com

また、www.aaa.com、www.bbb.com、www.ccc.com には、それぞれに関連付けられた Web モジュール web1、web2、web3 があるものとします。

つまり、その物理サーバーでは、次のすべての 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 は仮想ホスト `www.aaa.com`、2 番目の URL は仮想ホスト `www.bbb.com`、3 番目の URL は仮想ホスト `www.ccc.com` にそれぞれマッピングされます。

一方、`www.bbb.com` には `web3` が登録されていないため、次の URL は 404 リターンコードのエラーとなります。

```
http://www.bbb.com:8080/web3
```

このマッピングが機能するには、`www.aaa.com`、`www.bbb.com`、`www.ccc.com` のすべてを物理サーバーの IP アドレスとして解決する必要があります。これをネットワークの DNS サーバーに登録しなければなりません。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)。

Communications Server を起動すると、次の 2 つの仮想サーバーが自動的に起動されません。

- `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
- `__asadmin`: すべての管理関連 Web モジュール (具体的には管理コンソール) 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、ユーザーが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。

本稼動環境以外での Web サービスの開発、テスト、配備が必要となる仮想サーバーは、通常、`server` だけです。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。

HTTP リスナー

各仮想サーバーは、1 つまたは複数の HTTP 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 HTTP リスナーは、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待機ソケットです。

HTTP 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 アドレスをマシンの特定のポートで待機できます。また、各リスナーに一意の IP アドレスを指定した上で、同一ポートを使用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であるため、IP アドレスが同じでポート番号が異なる HTTP リスナーや (例: `1.1.1.1:8081` および `1.1.1.1:8082`)、IP

アドレスが異なっていてポート番号が同じ HTTP リスナー (例: 1.1.1.1: 8081 および 1.2.3.4: 8081)。ただし、マシンがこれら両方のアドレスに応答するように設定されている場合) を複数使用することができます。

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。

通常、Communications Server が稼動するシステムでアクセスできる IP アドレスは 1 つだけであるため、HTTP リスナーは、ポートが異なる 0.0.0.0 IP アドレスを通常使用し、役割ごとに異なるポート番号を使用します。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。

デフォルトでは、Communications Server を起動すると、次の HTTP リスナーが準備されます。

- `server` という仮想サーバーに関連付けられた `http-listener-1` および `http-listener-2` という 2 つの HTTP リスナー。
- 仮想サーバー `__asadmin` に関連付けられた HTTP リスナー `admin-listener`。

これらのリスナーはすべて、Communications Server のインストールの間に HTTP サーバーポート番号として指定された IP アドレス 0.0.0.0 とポート番号を使用します。Communications Server がデフォルトのポート番号を使用する場合、`http-listener-1` はポート 8080、`http-listener-2` はポート 8181、`admin-listener` はポート 48489 を使用します。

各 HTTP リスナーはデフォルトの仮想サーバーを持ちます。デフォルトの仮想サーバーは、HTTP リスナーがその HTTP リスナーに関連付けられたどの仮想サーバーともホストコンポーネントが一致しないすべての要求 URL をルーティングする宛先のサーバーです。仮想サーバーと HTTP リスナーの関連付けは、仮想サーバーの `http-listeners` 属性に HTTP リスナーを指定することで行われます。

さらに、HTTP リスナー内のアクセプタスレッドの数を指定します。アクセプタスレッドは、新しい接続を受け付けて接続キューに入れます。続いて、セッションスレッドがキューから接続を選択し、要求を処理します。要求の最後でセッションスレッドが要求された場合、サーバーはさらにセッションスレッドを送信します。

一連の要求処理スレッドが、接続キューから受信 HTTP 要求を取り出し、それらの要求を処理します。これらのスレッドは、HTTP ヘッダーを解析し、適切な仮想サーバーを選択し、要求処理エンジンを実行して要求を処理します。処理すべき要求がなくなったあと、その接続が HTTP/1.1 を使用するか `Connection: keep-alive` ヘッダーを送信することで持続可能になっていた場合、要求処理スレッドは、その接続がアイドル状態にあると判断し、その接続をキープアライブ接続管理サブシステムに渡します。

キープアライブサブシステムは、そうしたアイドル状態の接続を定期的にポーリングし、活動中の接続が見つかる则ちそれらを接続キュー内に格納し、さらに処理できるようにします。要求処理スレッドは、そのキューからふたたび接続を取り出し、その要求を処理します。キープアライブサブシステムはマルチスレッド化されています。なぜなら、このサブシステムは数万個の接続を管理する可能性があるからです。効率的なポーリングテクニックに基づいて多数の接続がより少数の接続を含むサブセットへと分割され、どの接続で要求の準備が整ったか、あるいはどの接続のアイドル時間が閉じてもよいほど十分長い時間になったか (最大許容キープアライブタイムアウトを超えたか) が判断されます。

HTTP リスナーのサーバー名は、サーバーがクライアントに送信する URL にリダイレクトの一部として表示されるホスト名です。この属性は、サーバーが自動的に生成する URL には影響しますが、サーバーに格納されているディレクトリやファイルの URL には影響しません。サーバーがエイリアスを使っている場合、普通、この名前はエイリアス名です。クライアントが `Host:` ヘッダーを送信する場合、HTTP リスナーのサーバー名の代わりにホスト名がリダイレクトに指定されます。

リダイレクトポートを指定して、元の要求に指定されているポート番号とは異なるポート番号を使用します。リダイレクトは、次のいずれかの状況で行われます。

- リソースが別の位置に移動され、クライアントのアクセス対象のリソースが指定の URL に存在しない場合、サーバーは 404 を返す代わりに指定の応答コードを返し、応答のロケーションヘッダーに新しい位置を含めることで、クライアントを新しい位置にリダイレクトします。
- SSL などによって保護されているリソースにクライアントが通常の HTTP ポートからアクセスを試みる場合、サーバーは要求を SSL 有効ポートにリダイレクトします。この場合、サーバーは、元のセキュリティ保護されていないポートを SSL 有効ポートに置き換えた新しい URL がロケーション応答ヘッダーに指定された応答を返します。クライアントは、この新しい URL に接続します。

また、HTTP リスナーのセキュリティを有効にするかどうか、あるいは、どのセキュリティの種類を使用するか (例: SSL プロトコルや暗号化方式の種類) も指定します。

Communications 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/` (または、セキュリティ保護されたアプリケーションでは `https://localhost:8181/`) 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/` の URL か、そのデフォルトコンテキストルートの `http://localhost:4848/asadmin/` を使用します。

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、ほかの仮想サーバーによってすでに使用されている HTTP リスナーを指定できないことから、新しい仮想サーバーを作成するときは、事前に 1 つの HTTP リスナーを作成しま

す。この制限は、新しい仮想サーバーのホストまたはエイリアス名が、既存の HTTP リスナーにすでにバインドされている仮想サーバーのいずれかと同じである場合にのみ適用されます。

SIP サービスの設定

SIP サービスは、SIP アプリケーションの配備機能を提供する Communications Server のコンポーネントで、配備された SIP アプリケーションに SIP クライアントがアクセスできるようにします。

HTTP サービスと異なり、SIP サービスは仮想サーバーを使用しません。

SIP サービス

次の SIP サービスの属性を、管理コンソールを使用するか、`asadmin get` および `set` コマンドを使用して変更できます。SIP サービスの属性は、SIP サービスのすべてのリスナーに適用されます。

SIP サービスには、アクセスログ、要求処理、キープアライブ、SIP プロトコル、SIP リスナー、および接続プールのサブ要素があります。

SIP サービスのサブ要素および属性を管理コンソールを使用して編集するには、「設定」>`config-name`>「SIP サービス」の順にクリックします。

SIP サービスのすべてのサブ要素を表示するには、次のコマンドを使用できます。

```
list server-config.sip-service.*
```

一般的な出力は次のとおりです。

```
server-config.sip-service.access-log
server-config.sip-service.connection-pool
server-config.sip-service.keep-alive
server-config.sip-service.request-processing
server-config.sip-service.sip-listener.sip-listener-1
server-config.sip-service.sip-listener.sip-listener-2
server-config.sip-service.sip-listener.sip-listener-2.ssl
server-config.sip-service.sip-listener.sip-listener-3
```

```
server-config.sip-service.sip-protocol
server-config.sip-service.sip-protocol.sip-link
server-config.sip-service.sip-protocol.sip-timers
```

SIP サービスのアクセスログ属性の設定

SIP サービスでアクセスログのローテーションを有効化および設定できます。これらのログは `as-install/domains/domain_dir/logs/sipaccess` ディレクトリに保存され、ファイル名は `sip_access_log.yyyy-mm-dd` のようになります。この場所は、`sip-service` 要素以下の `accesslog` プロパティを編集することで変更できません。

`sip-service` 以下の `accessLoggingEnabled` プロパティは、SIP アクセスログの有効化に使用します。アクセスログを有効または無効にしたら、サーバーを再起動します。次のコマンドを使用します。

```
asadmin set config.sip-service.property.accessLoggingEnabled=true
```

`sip-service` 以下の `accesslogsize` プロパティは、ファイルサイズに基づくログのローテーションに使用できます。このプロパティのデフォルト値は 2000000 です。

SIP サービスのアクセスログ属性を次に示します (括弧内は `domain.xml` ファイルで定義される実際の属性名)。

- ローテーション (`rotation-enabled`): 「ローテーション」ボックスにチェックマークを付けると、ファイルローテーションが有効になります。
- ローテーションポリシー (`rotation-policy`): 「ローテーションポリシー」ドロップダウンリストからポリシーを選択します (利用できるポリシーは `time` のみです)。
- ローテーション時間間隔 (`rotation-interval-in-minutes`): アクセスログのローテーションを実行する間隔 (分単位) を指定する数値を入力します。このフィールドは、「ローテーションポリシー」が「`time`」の場合にのみ有効です。デフォルトは 1440 分です。

`rotation-interval` を指定した場合、ローテーション間隔が経過したときに、新しいアクセスログファイルに付加されるローテーションサフィックスが、既存のアクセスログファイルのローテーションサフィックスと異なる場合のみ、アクセスログがローテーションされます。したがって、アクセスログファイルが予定した間隔でローテーションされたことを確認するには、ローテーション間隔とローテーションサフィックスを同時に指定する必要があります。

- ローテーションサフィックス (`rotation-suffix`): ローテーションのあとログファイルの名前に付加されるサフィックスを表す文字列を入力します。デフォルトは `YYYY-MM-DD` です。

- 書式 (format): アクセスログの書式を指定する文字列値を入力します。次の表に示す書式を使用します。デフォルトの書式は、`%client.name% %auth-user-name% %datetime% %request% %status% %response.length%` です。

SIP ヘッダーの

`client.name`、`auth-user-name`、`datetime`、`request`、`status`、`response.length`、`max.forwards`、`from`、`cseq`、`contact`、`to`、`content.type`、`via`、`call.id` もサポートされます。

すべてのヘッダーを有効にするには、次のコマンドを実行します。

```
asadmin set server-config.sip-service.access-log.format="\%client.name%
%auth-user-name% %datetime% %request% %status% %response.length%
%max.forwards% %from% %cseq% %contact% %to% %content.type% %via% %call.id%"
```

注-書式の文字列をエスケープする必要があります。

これらのプロパティは、管理コンソールか、次の `get` および `set` コマンドを使用して編集できます。

SIP サービスのアクセスログのプロパティ値をすべて取得するには、`asadmin get server-config.sip-service.access-log.*` コマンドを使用します。

SIP サービスのアクセスログの属性を設定するには、`set` コマンドを `asadmin set server-config.sip-service.access-log.rotation-enabled=false` のように実行します。

SIP サービスのキープアライブ属性の設定

これらの属性は、管理コンソールか、次の `get` および `set` コマンドを使用して編集できます。

SIP サービスのキープアライブ属性およびプロパティをすべて取得するには、`asadmin get server-config.sip-service.keep-alive.*` コマンドを使用します。

SIP サービスのキープアライブの `timeout-in-seconds` プロパティを設定できます。デフォルトでは、このプロパティは `-1` に設定されています。0 以下を設定すると、キープアライブ接続は開いたままになります。

SIP サービスのキープアライブの属性を設定するには、`set` コマンドを `asadmin set server-config.sip-service.keep-alive.timeout-in-seconds=0` のように使用します。

SIP サービスの接続プール属性の設定

これらの属性は、管理コンソールか、次の `get` および `set` コマンドを使用して編集できます。

SIP サービスの接続プールの属性値をすべて取得するには、`asadmin get server-config.sip-service.connection-pool.*` コマンドを使用します。

- キューサイズ (`queue-size-in-bytes`): SIP リスナーでスレッドを処理できるようになるまでに、キューに入れることができるメッセージの最大数を指定します。デフォルトは -1 です。この場合、メッセージの数は制限されません。
- 受信バッファサイズ (`receive-buffer-size-in-bytes`): ソケットチャネルからの読み取りに使用するバイトバッファのサイズを指定します。デフォルトでは、このプロパティはソケットチャネルの受信バッファサイズに合わせて調整されます。
- 送信バッファサイズ (`send-buffer-size-in-bytes`): 書き出し可能な SIP メッセージの最大サイズを指定します。デフォルトでは、このプロパティは 8192 に設定されます。

SIP サービスの接続プールの属性を設定するには、`set` コマンドを `asadmin set server-config.sip-service.connection-pool.receive-buffer-size-in-bytes=10` のように使用します。

SIP サービスの Protokol 属性の設定

これらの属性は、管理コンソールか、次の `get` および `set` コマンドを使用して編集できます。

SIP サービス Protokol のすべての属性とプロパティの値を取得するには、`asadmin get server-config.sip-service.sip-protocol.*` コマンドを使用します。

- エラー応答の有効化 (`error-response-enabled`): エラー応答を送信するかどうかを指定します。デフォルトでは `false` に設定されます。
- デフォルトトランスポート (`default-tcp-transport`): `true` に設定すると、`contact` および `record-route` ヘッダーの URI に、`transport=tcp` が挿入されます。

SIP サービス Protokol の属性を設定するには、`set` コマンドを `asadmin set server-config.sip-service.sip-protocol.default-tcp-transport=false` のように使用します。

SIP サービス Protokol には、SIP タイマーというサブ要素があります。SIP サービス Protokol のサブ要素は、`asadmin list server-config.sip-service.access-log.*` コマンドを使用して一覧表示できます。

SIP サービスタイマーのすべての属性とプロパティの値を取得するには、`asadmin get server-config.sip-service.protocol.sip-timers.*` コマンドを使用します。

SIP リスナー

管理コンソールまたは `get` および `set` コマンドを使用して、SIP リスナーを設定できません。SIP コンテナが待機するサーバーポートは、SIP リスナーを使用して設定できません。

デフォルトドメインのすべての SIP リスナーを表示するには、次の `asadmin` コマンドを入力します。

```
list server.sip-service.sip-listener.*
```

一般的な出力は次のとおりです。

```
server.sip-service.sip-listener.sip-listener-1
server.sip-service.sip-listener.sip-listener-2
server.sip-service.sip-listener.sip-listener-2.ssl
```

デフォルトでは、ドメインに `sip-listener-1` と `sip-listener-2` の2つのリスナーが作成され、それぞれ SIP ポートと SIPS ポートに対応します。デフォルトでは、`sip-listener-2` は TLS トランスポートを使用するように設定されます。

デフォルトドメインの `sip-listener-1` のすべての属性を表示するには、次の `asadmin` コマンドを入力します。

```
get server.sip-service.sip-listener.sip-listener-1.*
```

一般的な出力は次のとおりです。

```
server.sip-service.sip-listener.sip-listener-1.address = 0.0.0.0
server.sip-service.sip-listener.sip-listener-1.enabled = true
server.sip-service.sip-listener.sip-listener-1.external-sip-address =
server.sip-service.sip-listener.sip-listener-1.external-sip-port =
server.sip-service.sip-listener.sip-listener-1.id = sip-listener-1
server.sip-service.sip-listener.sip-listener-1.port = 5060
server.sip-service.sip-listener.sip-listener-1.transport = udp_tcp
server.sip-service.sip-listener.sip-listener-1.type = default
```

デフォルトでは、ポートは 5060 に設定され、アドレスは 0.0.0.0 に設定されません。0.0.0.0 のアドレスは、任意にバインド可能なアドレスで、サーバーソケットをシステム上の任意のネットワークインタフェースにバインドできます。

`sip-listener-1` のデフォルトポートを変更するには、次のコマンドを使用します。

```
set server.sip-service.sip-listener.sip-listener-1.port=5065
```

Communications Server では、SIP リスナーを実行時でも変更できます。現在のところ、Communications Server はこれらの変更による影響を考慮しません。SIP リスナーの設定を変更する場合は、あらかじめ計画することをお勧めします。

SIP リスナーの作成と編集

SIP リスナーを作成するには、管理コンソールで「設定」ノードを展開し、設定する設定を選択します。次に、SIP サービスを選択し、「SIP リスナー」ノードを選択して、「新規」をクリックします。SIP リスナーの設定を定義して、「保存」をクリックします。

または、`asadmin create-sip-listener` コマンドを使用することもできます。サンプルコマンドを次に示します。

```
asadmin create-sip-listener --siplisteneraddress=127.0.0.1
--siplistenerport=5066 --transport=udp_tcp mysiplistener
```

次に、SIP リスナーの属性を示します。

- 名前 (`id`): 新しいリスナーの一意の ID を指定します。
- ネットワークアドレス (`address`): 一意のポート値を使用して、リスナーでサーバーのすべての IP アドレスを待機する場合は、`0.0.0.0` を入力します。それ以外の場合は、サーバーの有効な IP アドレスを入力します。
- リスナーポート (`port`): 「ネットワークアドレス」フィールドが `0.0.0.0` の場合は、一意のポート値を入力します。別の IP アドレスを使用する場合は、目的のポート値を入力します。
- 有効 (`enabled`): リスナーを有効にするかどうかを決定します。 `false` に設定した場合は、リスナーに接続しようとする、ソケット例外 (`java.net.ConnectException`) が発生します。
- トランスポート (`transport`): トランスポートレイヤープロトコルの種類を指定します。設定できる値は `udp_tcp` と `tls` です。
- 外部 SIP ポート (`external-sip-port`): 外部 SIP ポートを指定します。この値を指定した場合は、SIP コンテナの外部 SIP ポートの値よりも優先されます。
- 外部 SIPS ポート (`external-sips-port`): 外部 SIPS ポートを指定します。この値を指定した場合は、SIP コンテナの外部 SIPS ポートの値よりも優先されます。
- トランスポート (`transport`): トランスポートレイヤープロトコルの種類を指定します。設定できる値は `udp_tcp` と `tls` です。
- タイプ (`type`): このオプションに指定できる値は、`internal`、`external`、および `default` です。 `internal` に設定すると、このリスナーは融合ロードバランサによるプロキシのみで使用されます。 `external` に設定すると、このリスナーは SIP ユーザーエージェントのみで使用されます。 `default` に設定すると、このリスナーは SIP ユーザーエージェントと融合ロードバランサで使用されます。デフォルトでは、このオプションの値は `default` に設定されます。

SIP リスナーの属性を変更するには、`asadmin set` コマンドを使用するか、管理コンソールを使用します。SIP リスナーを編集するには、管理コンソールで「設定」ノードを展開し、設定を選択します。次に、SIP サービスを選択し、「SIP リス

ナー」ノードを選択して、編集する SIP リスナーを選択します。SIP リスナーの設定を編集して、「保存」をクリックします。

たとえば、SIP リスナーの SIP ポートを変更する場合は、`asadmin set server.sip-service.sip-listener.sip-listener-1.port=5065` のコマンドを使用します。

▼ SSL を使用するように SIP リスナーを設定する

- 1 開発者プロファイルで、「設定」、SIP サービス、「SIP リスナー」ノードの順に展開します。
- 2 クラスタプロファイルおよびエンタープライズプロファイルでは、「設定」ノードを展開し、変更する設定を選択します。次に、SIP サービスを選択し、「SIP リスナー」ノードを選択します。
- 3 設定する SIP リスナーを選択します。
- 4 「SSL」タブを選択して、次のように SSL を使用するリスナーを設定します。
 - 「クライアント認証」: このフィールドをチェックすると、クライアントはこのリスナーを使用するときにサーバーに対して自身を認証します。
 - 「証明書のニックネーム」: 既存のサーバーキーペアの名前と証明書を入力します。詳細は、セキュリティに関する章を参照してください。
 - 「SSL3」、「SSL2」、「TLS」: 「有効」チェックボックスにチェックマークを付けると、リスナーでセキュリティプロトコルが有効になります。SSL3、SSL2、TLS のいずれか、または3つすべてにチェックマークを付けます。
 - 「暗号化方式群」: プロトコルで使用される利用可能な暗号化方式群を選択し、「追加」ボタンをクリックします。すべての暗号化方式群を有効にするには、「すべてを追加」ボタンを選択します。
- 5 「保存」をクリックします。

参考 同機能を持つ `asadmin` コマンド

```
create-ssl
```


過負荷保護

Communications Server は、システムが過負荷状態になるのを防ぐ、過負荷保護機能を備えています。一般的な過負荷保護および負荷規制の方式は、負荷が増大しているときに、特定の種類の要求を拒否することです。過負荷の状況に応じて、特定の要求に優先順位が付けられ、システムの負荷を減少させるために特定の要求が拒否されます。たとえば、過負荷の状況では、Communications Server は新規の SIP セッションを拒否して、実行中のセッションを優先します。

過負荷保護サービスは、管理コンソールで「設定」→「過負荷保護」ノードを使用するか、`asadmin get` と `set` コマンドを使用して表示や設定ができます。

Communications Server で過負荷保護を有効にするには、次の操作を行います。

1. `enabled` 属性を `true` に設定します。デフォルトでは、この値は `false` に設定されています。次のコマンドを使用して、このプロパティを設定します。

```
asadmin set config.overload-protection-service.enabled=true
```

注 - 次のプロパティの使用は、Communications Server 2.0 では推奨されません。
`config.sip-container.property.olpInserted`、
`config.sip-container.property.CPUOverLoadRegulation`、および
`config.sip-container.property.MemOverLoadRegulation`。これらのプロパティの代わりに、`overload-protection-service` 要素の属性を使用してください。

2. この属性の設定に加えて、次のいずれかのタイプの過負荷保護を有効にする必要があります。
 - 178 ページの「CPU 過負荷保護」
 - 179 ページの「メモリー過負荷保護」

CPU 過負荷保護

Communications Server では、CPU の負荷に基づいて、受信 HTTP および SIP 要求を規制できます。

CPU 過負荷保護を有効にするには、`cpu-overload-protection` 属性を `true` に設定します。デフォルトでは、この値は `false` に設定されています。次のコマンドを使用して、この属性を設定します。

```
asadmin set config.overload-protection-service.cpu-overload-protection=true
```

過負荷保護の設定を表示するには、次のコマンドを使用します。

```
asadmin get config.overload-protection-service.*
```

```
config1.overload-protection-service.cpu-http-threshold = 70
config1.overload-protection-service.cpu-ir-threshold = 70
config1.overload-protection-service.cpu-mm-threshold = 99
config1.overload-protection-service.cpu-overload-activation-algorithm = MEDIAN
config1.overload-protection-service.cpu-overload-deactivation-algorithm = CONSECUTIVE
config1.overload-protection-service.cpu-overload-protection = true
config1.overload-protection-service.cpu-sr-threshold = 90
config1.overload-protection-service.number-of-samples = 5
config1.overload-protection-service.retry-after-interval = 10
config1.overload-protection-service.sample-rate = 2
```

CPU 過負荷保護の属性は次のとおりです。

- `cpu-http-threshold` - このしきい値を超えると、すべての受信 HTTP 要求が拒否され、503 応答が送信されます。
- `cpu-ir-threshold` - このしきい値を超えると、初期受信 SIP 要求が拒否され、503 応答が送信されます。
- `cpu-sr-threshold` - このしきい値を超えると、以降のすべての受信 SIP 要求が拒否され、503 応答が送信されます。
- `cpu-mm-threshold` - このしきい値を超えると、すべての受信 HTTP 要求および SIP 要求が拒否されます。
- `cpu-overload-protection` - CPU 過負荷保護が有効かどうかを指定します。
- `cpu-overload-activation-algorithm` - CPU の過負荷状態を検出するアルゴリズムを指定します。有効な値は、`CONSECUTIVE` と `MEDIAN` です。
- `cpu-overload-deactivation-algorithm` - CPU の過負荷状態の終わりを検出するアルゴリズムを指定します。有効な値は、`CONSECUTIVE` と `MEDIAN` です。
- `mm-threshold-http-wait-time` - スレッドをプールに解放するまでに保持する期間を示します。この属性は、HTTP 要求の最大過負荷に関して使用されます。過負荷状態では、要求の処理にスレッドをすぐに利用できないため、システムの過剰な要求を減らすことができます。デフォルトは2秒です。

- `sample-rate` - CPU 使用状況を測定する間隔。
- `number-of-samples` - 要求の拒否と 503 応答の送信がトリガーされるまでに、CPU 使用状況がしきい値レベルを超える必要があるサンプル数。
- `retry-after-interval` - CPU 使用状況を測定するまでの間隔。これは、503 応答の `retry-after` ヘッダーに設定される値です。

これらの値を表示および設定するには、管理コンソールで「設定」→「過負荷保護」ノードを使用するか、`asadmin get` と `set` コマンドを使用します。

たとえば、`cpu-mm-threshold` プロパティの値は、次のコマンドを使用して設定できます。

```
asadmin set config.overload-protection-service.cpu-mm-threshold =80
```

メモリー過負荷保護

Communications Server では、メモリーの使用状況に基づいて、受信 HTTP および SIP 要求を規制できます。

メモリー過負荷保護を有効にするには、`memory-overload-protection` 属性を `true` に設定します。デフォルトでは、この値は `false` に設定されています。次のコマンドを使用して、この属性を設定します。

```
asadmin set config.overload-protection-service.memory-overload-protection=true
```

過負荷保護サービスのメモリー過負荷保護の属性は、次のとおりです。

```
config1.overload-protection-service.mem-http-threshold = 85
config1.overload-protection-service.mem-ir-threshold = 85
config1.overload-protection-service.mem-mm-threshold = 99
config1.overload-protection-service.mem-overload-activation-algorithm = MEDIAN
config1.overload-protection-service.mem-overload-deactivation-algorithm = CONSECUTIVE
config1.overload-protection-service.mem-sr-threshold = 85
config1.overload-protection-service.memory-overload-protection = false
config1.overload-protection-service.mm-threshold-http-wait-time = 2
config1.overload-protection-service.number-of-samples = 5
config1.overload-protection-service.retry-after-interval = 10
config1.overload-protection-service.sample-rate = 2
```

次のしきい値レベルを設定できます。

- `mem-http-threshold` - このしきい値を超えると、すべての受信 HTTP 要求が拒否され、503 応答が送信されます。
- `mem-ir-threshold` - このしきい値を超えると、初期受信要求が拒否され、503 応答が送信されます。

- `mem-mm-threshold` - このしきい値を超えると、すべての受信 HTTP 要求および SIP 要求が拒否されます。
- `mem-overload-protection` - メモリー過負荷保護が有効かどうかを指定します。
- `mem-overload-activation-algorithm` - メモリーの過負荷状態を検出するアルゴリズムを指定します。有効な値は、`CONSECUTIVE` と `MEDIAN` です。
- `mem-overload-deactivation-algorithm` - メモリーの過負荷状態の終わりを検出するアルゴリズムを指定します。有効な値は、`CONSECUTIVE` と `MEDIAN` です。
- `mm-threshold-http-wait-time` - スレッドをプールに解放するまでに保持する期間を示します。この属性は、HTTP 要求の最大過負荷に関して使用されます。過負荷状態では、要求の処理にスレッドをすぐに利用できないため、システムの過剰な要求を減らすことができます。デフォルトは2秒です。
- `sample-rate` - メモリー使用状況を測定する間隔。
- `number-of-samples` - 要求の拒否と 503 応答の送信がトリガーされるまでに、メモリー使用状況がしきい値レベルを超える必要があるサンプル数。
- `retry-after-interval` - メモリー使用状況を測定するまでの間隔。これは、503 応答の `retry-after` ヘッダーに設定される値です。

Web サービスの管理

この章では、Communications Server による Web サービス管理について説明します。管理コンソールおよび asadmin ツールで、Web サービスの配備、テスト、および管理を行うことができます。複雑な Web サービスをすばやく視覚化して理解し、監視および管理することができます。ドメインに配備されたすべての Web サービスを、Java EE アプリケーションや、EJB のようなアプリケーションコンポーネントと同じように表示できます。

さらに、次の操作も可能です。

- Web サービスの応答時間や呼び出し回数をリアルタイムで追跡してグラフに表示する。
- 応答時間やスループットの障害などの境界条件に対してアラートを生成する。
- Web サービスの呼び出し内容を XML で表示する。
- XSLT を使用して実行時にメッセージを変換する。

この章の内容は次のとおりです。

- 181 ページの「Web サービスの概要」
- 184 ページの「Web サービスの配備とテスト」
- 185 ページの「Web サービスレジストリの使用」
- 186 ページの「XSLT フィルタによるメッセージの変換」
- 187 ページの「Web サービスの監視」

Web サービスの概要

Web サービスは、クライアントが Simple Object Access Protocol (SOAP) などの XML ベースプロトコルを使用してアクセスし、HTTP などのインターネットプロトコルを介して送信されるアプリケーションです。クライアントは、Web Services Definition Language (WSDL) ファイルなどの XML アーティファクトを使って定義されたインタフェースとバインディングを通して Web サービスアプリケーションにアクセスします。

eXtensible Markup Language (XML) は、World Wide Web Consortium (W3C) によって開発された規格であり、Web サービス構築の基盤の1つです。XMLによって、Web サービスとクライアントが共通の言語で互いに通信できます。XMLは、シンプルで柔軟な、テキストベースのマークアップ言語です。XMLデータは、山括弧で囲まれたタグを使用してマーク付けされます。タグ内には、そのタグでマーク付けするデータの意味が含まれます。このようなマークアップにより、異なるシステム間で容易にデータを交換できます。

文書型定義 (Document Type Definitions、DTD) または XML スキーマ定義 (XML Schema Definition、XSD) には、XML ドキュメントの構造が記述されます。これには、対応する XML ドキュメントで使用できるタグや、それらのタグの順序などの情報が含まれます。

XSLT (eXtensible Stylesheet Language Transformation の略) は、XML ドキュメントをある形式から別の形式に変換するために使用されます。

Web サービスの規格

Simple Object Access Protocol (SOAP) は、Web サービスに共通のメッセージング形式を提供します。SOAPによって、互いに未知のオブジェクトがメッセージを交換できます。SOAPでは、XML ベースのデータコード化形式と HTTP を使用してメッセージを転送します。SOAPは、プログラミング言語にも動作プラットフォームにも依存せず、エンドポイントで特定のテクノロジーを一切必要としません。

Universal Description, Discovery, and Integration (UDDI) は、Web サービスの登録、登録解除、および検索を行うための標準的な方法を提供します。電話システムのイエローページと同様に、UDDI レジストリによって、提供者はサービスを登録し、要求者はサービスを検索できます。要求者がサービスを見つけたあとは、要求者と提供者の間でレジストリが果たす役割はもうありません。

Web Services Description Language (WSDL) は、Web サービスの詳細を指定する標準的な方法を定義します。これは汎用 XML スキーマであり、Web サービスのインタフェース、バインディング、および配備に関するその他の詳細を指定できます。このようなサービスの詳細を指定する標準的な方法があることで、クライアントは予備知識がなくても Web サービスを使用できます。

ebXML (Electronic Business using eXtensible Markup Language) は、企業がインターネットを介してビジネスを行うための一連の仕様です。ebXML 仕様は、OASIS (Organization for the Advancement of Structured Information Standards) により管理されています。

Java EE Web サービスの規格

Java APIs for XML Processing (JAXP) は、XML ドキュメントの解析や処理を行うための、ベンダー中立の一連の軽量 API です。JAXP によって、規格に準拠するすべての XML パーサーを Web サービスに「プラグイン」できます。外部パーサーが「プラグイン」されていない場合、JAXP は自身に実装された XML パーサーを使用します。

Java API for XML-based Remote Procedure Calls (JAX-RPC) は、クライアントサーバー遠隔手続き呼び出しに XML ベースプロトコルを使用します。JAX-RPC は、SOAP ベースの相互運用可能で移植性のある Web サービスを可能にします。開発者は、JAX-RPC プログラミングモデルを使用して、SOAP ベースの Web サービスエンドポイントとそれに対応する WSDL 記述、およびクライアントを開発します。JAX-RPC ベースの Web サービスは、Java ベース以外のクライアントと対話できます。同様に、JAX-RPC ベースのクライアントは、Java ベース以外の Web サービスと対話できます。

Java API for XML Registries (JAXR) は、ビジネスレジストリにアクセスするための Java API です。JAXR には、UDDI およびその他のレジストリ仕様 (ebXML など) をサポートする柔軟なアーキテクチャーが備わっています。JAXR クライアント (スタンドアロン Java アプリケーションや J2EE コンポーネントなど) は、JAXR プロバイダが提供する JAXR API の実装を使用して、ビジネスレジストリにアクセスします。JAXR プロバイダは2つの部分から構成されます。レジストリ固有の API 実装を提供するレジストリ固有の JAXR プロバイダと、レジストリのタイプに依存しない API のレジストリ機能を実装するプラグイン可能な JAXR プロバイダです。プラグイン可能なプロバイダは、レジストリ固有プロバイダの詳細をクライアントから見えないようにします。

SOAP with Attachments API for Java (SAAJ) により、開発者は、SOAP 1.1 仕様と SOAP with Attachments note に準拠するメッセージを作成および処理できます。SAAJ プロバイダは、添付ファイル付きの SOAP メッセージを処理するための抽象化オブジェクトを提供します。上級開発者は、SAAJ を使用して、SOAP メッセージで直接アプリケーションを動作させることができます。添付ファイルには、完全な XML ドキュメント、XML フラグメント、または MIME タイプの添付ファイルを使用できます。また、SAAJ により、開発者は、ほかの MIME タイプのサポートを有効にできます。JAX-RPC などの JAX テクノロジーでは、内部的に SAAJ を使用することで、SOAP の複雑さを開発者に感じさせないようにしています。SAAJ には次の機能があります。

- 要求/応答型の同期メッセージング: クライアントはメッセージを送信し、応答を待ちます。
- 一方向の非同期メッセージング: クライアントはメッセージを送信し、応答を待たずにその処理を続けます。

Web サービスの配備とテスト

Communications Server では、Web サービスの配備とテストを簡単に行えます。

Web サービスの配備

エンタープライズアプリケーションと同じように、エンタープライズアーカイブ (EAR) で Web サービスを配備します。

また、POJO (Plain Old Java Object) によって Web サービスを実装することもできます。POJO Web サービスを配備するには、自動配備機能を使用します。これを行うには、サービスを自動配備ディレクトリにドラッグ&ドロップします。Communications Server によって、自動的に適切な Web XML ファイルが生成され、Web サービスが配備されます。

管理コンソールで、「Application Server」>「Web サービス」|「一般」の順に選択すると、配備済みの Web サービスの一覧を表示できます。

配備済み Web サービスの表示

管理コンソールで Web サービスをテストするには、「アプリケーション」>「Web サービス」>「web-service-name」|「一般」の順に選択します。管理コンソールに、次のような Web サービスの属性が表示されます。

- 名前: Web サービスの名前。
- エンドポイントアドレス URI: Web サービスのエンドポイントの URI。
- アプリケーション: リンクをクリックすると、Web アプリケーションまたはエンタープライズアプリケーションのプロパティが表示されます。
- WSDL: リンクをクリックすると、Web サービスの WSDL ファイルが表示されません。
- モジュール名: Web サービスの WAR ファイルまたは EAR ファイルの名前。
- マッピングファイル: リンクをクリックすると、Java WSDL マッピングファイルが表示されます。
- Webservices.xml: リンクをクリックすると、webservices.xml ファイルが表示されません。
- 実装タイプ: SERVLET または EJB。
- 実装クラス名:
- 配備記述子:

Web サービスのテスト

管理コンソールで、Web サービスをテストし、問題を診断できます。汎用テストサーブレットで、配備済み Web サービスに対して ping を実行できます。メソッドの呼び出しごとに SOAP メッセージが表示されます。

管理コンソールで Web サービスをテストするには、「アプリケーション」>「Web サービス」>「web-service-name」|「一般」の順に選択し、「テスト」ボタンをクリックします。

Web サービスのセキュリティー

SOAP メッセージ層セキュリティーのサポートは、WS-Security の SAML トークンプロファイルに基づきます。Web サービスの改ざん防止監査機能も提供されます。

Web サービスレジストリの使用

注 - Communications Server には、内部レジストリが付属していません。Web サービスを内部レジストリに発行するには、レジストリを Application Server にダウンロードし、インストールする必要があります。Web サービスを外部レジストリに発行するには、外部レジストリのアドレスを指定します。

レジストリの追加

管理コンソールで Web サービスレジストリを追加または削除するには、「Application Server」>「Web サービス」|「レジストリ」の順に選択します。このページで、レジストリアクセスポイント (RAP) を作成します。レジストリを追加する場合、次のパラメータを指定します。

- JNDI 名: レジストリの接続リソースプール (JNDI) 名。このコネクタリソースの JNDI 名は、レジストリの JNDI 名です。
- 追加するレジストリの種類を選択します。UDDI 3.0、ebXML のいずれかです。
- 「発行 URL」および「照会 URL」: レジストリの発行用アドレスと照会用アドレスをそれぞれ指定します。形式は `http://<hostname>/<path of registry installation>` です。
- レジストリのユーザー名とパスワード。

次の手順によって、レジストリ JNDI 名が作成されます。

- レジストリと通信できるリソースアダプタが作成されます。

- アプリケーションサーバーのコンテキストでは、JAXR リソースアダプタは UDDI レジストリと通信するように事前設定されています。また、SOA レジストリリソースアダプタモジュールをダウンロードすることもできます。SOA レジストリは、Sun 独自の ebXML レジストリです。
- リソースアダプタを使用して、接続リソースプールを作成します。
- 作成した接続プールを使用して、コネクタリソースを作成します。

レジストリへの Web サービスの発行

管理コンソールで Web サービスを発行するには、「アプリケーション」>「Web サービス」>「*web-service-name*」|「発行」の順に選択します。

「Web サービスの発行」画面で、Web サービスの発行対象のレジストリを1つ以上選択し、「発行」をクリックします。利用可能なレジストリをすべて発行するには、「すべて追加」ボタンをクリックします。

レジストリ内でこの Web サービスを表示するカテゴリを入力します。各カテゴリを区切るには、コンマを使用します。カテゴリは、使用中のレジストリに定義されます。この Web サービスの説明を入力します。UDDI レジストリに発行する場合は、Organization の名前を入力します。

ロードバランサを使用する場合は、ロードバランサのホスト名、ポート番号、および SSL ポート番号を入力します。Web サービスを外部レジストリに発行する場合は、WSDL はインターネットを通じて検出されますが、これらのオプションによって WSDL ファイルに指定されたホスト名とポート名はロードバランサのホスト名とポート名に置き換えられます。

Web サービスを発行解除するには、「Web サービスの発行」画面で、Web サービスを発行解除するレジストリを選択し、「発行解除」をクリックします。

XSLT フィルタによるメッセージの変換

Web サービスのエンドポイントに XSLT 変換規則を適用できます。これによって、Web サービスの要求と応答に関して詳細な制御を行うことができます。1つの Web サービスエンドポイントメソッドに複数の XSLT 規則を適用できます。また、各変換の適用順序も設定できます。XSLT ファイルはすべて、中央リポジトリの `generated/xml/appOrModule` ディレクトリに格納されます。これらの変換規則は、リモートサーバーインスタンスと同期されます。

変換規則を SOAP 要求や SOAP 応答に適用できます。

管理コンソールで Web サービス操作に適用する変換規則を追加するには、「アプリケーション」>「Web サービス」>「*web-service-name*」|「変換」の順に選択します。「追加」をクリックします。

この Web サービスのエンドポイントに利用可能な変換規則のリストが表示されません。

変換規則が定義されている XSLT ファイルの場所を参照します。生成された XSLT ファイルはすべて `generated/xml/application or module name/` ディレクトリに格納されます。

Web サービスのエンドポイントに複数の変換規則を追加する場合、変換規則は追加した順序で適用されます。

変換規則を有効にするには、「変換規則」ページで、その規則のチェックボックスにチェックマークを付けてから、「有効」をクリックします。規則を無効にするには、「無効」をクリックします。

変換規則を削除するには、「変換規則」ページで、その規則のチェックボックスにチェックマークを付けてから、「削除」をクリックします。これによって、変換規則がリストから削除されます。変換規則が Web サービスエンドポイントに適用されている場合、その変換規則は自動的に無効になります。ただし、XSLT ファイルはファイルパスの示す場所に残ります。ほかの Web サービスエンドポイントでは、この XSLT ファイルを使用できます。

Web サービスの監視

管理コンソールで、Web サービス処理の統計情報を追跡してグラフィック表示したり、Web サービスで送受信されたメッセージを表示したりできます。

Web サービスの監視を有効にするには、管理コンソールで、「アプリケーション」>「Web サービス」>「*web-service-name*」|「監視」|「設定」の順に選択します。

「監視の設定」ページで、監視レベルを設定します。

- LOW - Web サービスの応答時間、スループット、要求の合計数、および障害を監視します。メソッドレベルの監視は行いません。
- HIGH - 1 秒あたりの要求数、平均応答時間、およびスループット属性を追跡および監視するメッセージを追加します。
- OFF - 監視を無効にします。

「メッセージ履歴」の値を入力します。デフォルトは 25 秒です。「リセット」ボタンをクリックして、すべての統計情報を消去します。移動平均が再開されます。

Web サービス統計の表示

Communications Server 2.0 は、Web サービス処理の統計情報を追跡し、グラフィック表示する機能を提供します。

監視統計を表示するには、「アプリケーション」>「Web サービス」>「*web-service-name*」|「監視」|「統計」の順にクリックします。利用できる統計情報は次のとおりです。

- すべての正常処理または異常処理のミリ秒単位の応答時間(最大、最小、平均)。
- スループット
- 要求の合計数
- 障害の合計数および障害発生元エンドポイントの URI
- 認証失敗の合計数
- 承認成功の合計数

Web サービスメッセージの監視

Web サービスのエンドポイントに関するメッセージ(デフォルトは25)を表示するように Web サービスを設定することもできます。これらのメッセージは、リモートサーバーインスタンスのメモリーに格納されます。SOAP 要求、応答、HTTP ヘッダー情報の詳細が表示されます。

Web サービスメッセージを監視するには、「アプリケーション」>「Web サービス」>「*web-service-name*」|「監視」|「メッセージ」の順にクリックします。

有効にすると、Web サービスエンドポイントに関する最新のメッセージ(デフォルトでは25件)を表示できます。これらのメッセージはリモートサーバーインスタンスのメモリーに保持されます。これには、SOAP 要求と応答の詳細および HTTP ヘッダー情報が含まれます。

Web サービスに関して受信したメッセージのリストが表示されます。表示されるメッセージの数は、監視の設定によって決まります。

フィルタを選択して、正常処理のメッセージまたは異常処理のメッセージだけを表示することもできます。

ORB (Object Request Broker) の設定

この章では、ORB (Object Request Broker) と IIOP リスナーの設定方法について説明します。内容は次のとおりです。

- 189 ページの「Object Request Broker の概要」
- 190 ページの「ORB の設定」
- 190 ページの「IIOP リスナーの管理」

Object Request Broker の概要

- 189 ページの「CORBA」
- 190 ページの「ORB とは」
- 190 ページの「IIOP リスナー」

CORBA

Communications Server は、相互運用性を確実にする一連の標準的なプロトコルおよび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。

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、実行する必要が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 とは

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。

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。その代わりに、リモートスタブを介して、ローカルマシンで実行されている ORB に要求を送ります。次に、ローカルの ORB が、IIOP (Internet Inter-Orb Protocol) を使ってほかのマシン上の ORB へ要求を転送します。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 を介して Communications Server と通信します。

IIOP リスナー

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。Communications 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、ネットワークアドレス、およびオプションでセキュリティ属性を指定してください。

ORB の設定

ORB を設定するには、管理コンソールで「設定」タブをクリックします。設定するインスタンスに対応する ORB タブをクリックします。

IIOP リスナーの管理

IIOP リスナーを作成、編集、および削除するには、管理コンソールで「設定」タブをクリックします。設定するインスタンスに対応する ORB タブをクリックします。「IIOP リスナー」タブを選択します。手順の詳細については、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。

◆◆◆ 第 18 章

スレッドプール

Java 仮想マシン (JVM) は、1 回の実行で多数のスレッドをサポートできます。パフォーマンスに役立つように、Communications 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と ORB 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。サーバーは要求を受け取ると、スレッドプールから使用可能なスレッドにその要求を割り当てます。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。

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。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だけが Communications Server のスレッドをすべて占有している場合、Communications Server のスレッドを異なるスレッドプールに分割して、スレッド不足を防止してください。

この章の内容は次のとおりです。

- 192 ページの「スレッドプールの操作」

スレッドプールの操作

管理コンソールを使用してスレッドプールを作成するには、「設定」>「スレッドプール」>「現在のプール」>「新規」の順に選択します。

- 「スレッドプールID」フィールドに、スレッドプールの名前を入力します。
- 「最小スレッドプールサイズ」フィールドに、キューで要求を処理するスレッドプール内のスレッドの最小数を入力します。
スレッドプールがインスタンス化されると、これらのスレッドが前もって作成されます。
- 「最大スレッドプールサイズ」フィールドに、キューで要求を処理するスレッドプール内のスレッドの最大数を入力します。
これがそのスレッドプールに存在するスレッドの数の上限になります。
- 「アイドルタイムアウト」フィールドに、プールからアイドルスレッドが削除されるアイドル時間の制限秒数を入力します。
- 「作業キューの数」フィールドに、このスレッドプールが処理するワークキューの総数を入力します。
- **Communications Server** を再起動します。

スレッドプールの作成に関する詳細については、管理コンソールで「ヘルプ」をクリックしてください。

また、`asadmin` コマンドの `create-threadpool` を使用して、コマンド行からスレッドプールを作成することもできます。

管理コンソールを使用してスレッドプールの設定を編集するには、「設定」>「スレッドプール」>「現在のプール」の順に選択し、設定するプールを選択します。選択したスレッドプールの値を変更して保存し、**Communications Server** を再起動します。

スレッドプールの編集の詳細については、管理コンソールで「ヘルプ」をクリックしてください。

管理コンソールを使用してスレッドプールを削除するには、「設定」>「スレッドプール」>「現在のプール」の順に選択します。削除するスレッドプール名を確認し、「削除」をクリックします。

また、`asadmin` コマンドの `delete-threadpool` を使用して、コマンド行からスレッドプールを削除することもできます。

SIP コンテナのスレッドプールの設定

SIP コンテナはスレッドを使用して、受信要求の処理、応答の送信、タイマーの実行、クライアントでの新しい要求の開始などのタスクを実行します。これらのタスクを実行するために必要なスレッドは、SIP サービス専用で作成および設定されたスレッドプールから取得されます。

Grizzly フレームワークは、スレッドを取得するスレッドプール(パイプライン)を提供します。

SIP サービス要素には、スレッドプールを設定するための要素として `initial-thread-count`、`thread-count`、および `thread-increment` が用意されています。これらの属性の詳細は、『[Sun GlassFish Communications Server 2.0 Administration Reference](#)』を参照してください。

ロギングの設定

この章では、ロギングの設定方法とサーバーログの表示方法について簡単に説明します。次の節で構成されています。

- 195 ページの「ロギングについて」
- 197 ページの「ロギングの設定」

ロギングについて

- 195 ページの「ログレコード」
- 196 ページの「モジュールのログレベル」

ログレコード

Communications Server は、JSR 047 に記述された Java EE プラットフォームのロギング API を使用します。Communications Server のログメッセージはサーバーログ (通常は `&DomainDir/logs/server.log`) に記録されます。ログをローテーションするときに、Communications Server は `server.log` という名前で新しい空のファイルを作成し、古いファイルの名前を `server.log_date` に変更します。`date` はファイルがローテーションされた日付と時刻になります。

Communications Server のコンポーネントがログ出力を生成します。アプリケーションコンポーネントもログ出力を生成できます。

アプリケーションコンポーネントは、Apache Commons ロギングライブラリを使ってメッセージをロギングしてもかまいません。ただし、ログ設定を効率的に行いたい場合は、プラットフォーム標準の JSR 047 API を使用することをお勧めします。

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|#]
```

次に例を示します。

```
[#|2006-10-21T13:25:53.852-0400|INFO|sun-appserver9.1|javax.enterprise.  
system.core|_ThreadID=13;|CORE5004: Resource Deployed:  
[cr:jms/DurableConnectionFactory].|#]
```

この例について説明します。

- [#と#] はレコードの開始と終了をマーク付けします。
- 垂直バー (|) はレコードのフィールドを区切ります。
- 2006-10-21T13:25:53.852-0400 は日付と時刻を指定します。
- *Log Level* は INFO です。このレベルは次のいずれかの値を取ることができます。SEVERE、WARNING、INFO、CONFIG、FINE、FINER、および FINEST。
- *ProductName-Version* は sun-glassfish-comms-server2.0 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、この場合は javax.enterprise.system.core です。
- *Key Value Pairs* はキー名と値で、通常は _ThreadID=14; のようなスレッド ID です。
- *Message* は、ログメッセージのテキストです。Communications Server のすべての SEVERE メッセージと WARNING メッセージ、および多くの INFO メッセージは、モジュールコードと数値から構成されるメッセージ ID (この場合は CORE5004) で始まります。

このログレコード形式は、将来のリリースでは変更または拡張される可能性があります。

モジュールのログレベル

Communications Server は各モジュールのロガーを提供します。

```
asadmin> get server-config.log-service.module-log-levels.*  
server-config.log-service.module-log-levels.admin = INFO  
server-config.log-service.module-log-levels.classloader = INFO  
server-config.log-service.module-log-levels.cmp = INFO  
server-config.log-service.module-log-levels.cmp-container = INFO  
server-config.log-service.module-log-levels.configuration = INFO  
server-config.log-service.module-log-levels.connector = INFO  
server-config.log-service.module-log-levels.corba = INFO  
server-config.log-service.module-log-levels.deployment = INFO  
server-config.log-service.module-log-levels.ejb-container = INFO  
server-config.log-service.module-log-levels.group-management-service = INFO  
server-config.log-service.module-log-levels.javamail = INFO  
server-config.log-service.module-log-levels.jaxr = INFO
```

```
server-config.log-service.module-log-levels.jaxrpc = INFO
server-config.log-service.module-log-levels.jdo = INFO
server-config.log-service.module-log-levels.jms = INFO
server-config.log-service.module-log-levels.jta = INFO
server-config.log-service.module-log-levels.jts = INFO
server-config.log-service.module-log-levels.management-event = INFO
server-config.log-service.module-log-levels.mdb-container = INFO
server-config.log-service.module-log-levels.naming = INFO
server-config.log-service.module-log-levels.node-agent = INFO
server-config.log-service.module-log-levels.property.ar = INFO
server-config.log-service.module-log-levels.property.clb = INFO
server-config.log-service.module-log-levels.property.sip = INFO
server-config.log-service.module-log-levels.property.smi = INFO
server-config.log-service.module-log-levels.property.ssr = INFO
server-config.log-service.module-log-levels.resource-adapter = INFO
server-config.log-service.module-log-levels.root = INFO
server-config.log-service.module-log-levels.saaj = INFO
server-config.log-service.module-log-levels.security = INFO
server-config.log-service.module-log-levels.self-management = INFO
server-config.log-service.module-log-levels.server = INFO
server-config.log-service.module-log-levels.synchronization = INFO
server-config.log-service.module-log-levels.util = INFO
server-config.log-service.module-log-levels.verifier = INFO
server-config.log-service.module-log-levels.web-container = INFO
```

ロギングの設定

ここでは、次の内容について説明します。

- [197 ページの「ログの一般設定」](#)
- [198 ページの「ログレベルの設定」](#)
- [198 ページの「SIP メッセージ検査」](#)
- [199 ページの「サーバーログの表示」](#)

ログの一般設定

管理コンソールを使用してログの一般設定を設定するには、次の手順に従います。

- 開発者プロファイルの場合は、「Application Server」→「ログ」→「一般」の順に選択します。
- クラスタプロファイルの場合は、「設定」→「設定」→「ログ設定」→「一般」の順に選択します。

「一般」ページで適切な値を入力し、必要に応じてログをカスタマイズします。Communications Server を停止して再起動します。

各設定パラメータの設定の詳細については、管理コンソールで「ヘルプ」をクリックしてください。

これらのログ設定を `asadmin` で設定するには、`get` および `set` コマンドを使用します。

ログレベルの設定

管理コンソールを使用してログレベルを設定するには、次の手順に従います。

- 開発者プロファイルの場合は、「Application Server」→「ログ」→「ログレベル」の順に選択します。
- クラスタおよびエンタープライズプロファイルの場合は、「設定」→「設定」→「ログ」→「ログ設定」→「ログレベル」の順に選択します。

このページで、一覧表示されたモジュールのログレベルを設定します。アプリケーションロガーのログレベルを設定するには、「追加プロパティ」領域を使用します。モジュールのロガーのリストについては、[196 ページの「モジュールのログレベル」](#)を参照してください。

各設定パラメータの設定の詳細については、管理コンソールで「ヘルプ」をクリックしてください。

これらのログ設定を `asadmin` で設定するには、`get` および `set` コマンドを使用します。

たとえば、SIP コンテナのログレベルを「FINE」に設定する場合は、次の `set` コマンドを使用します。

```
asadmin set server-config.log-service.module-log-levels.property.sip=FINE
```

SIP メッセージ検査

SIP メッセージ検査機能は、Communications Server での SIP トラフィックのトラブルシューティングに役立ちます。この機能は、SMI ログドメインのレベルを「FINE」に設定することで有効になります。

▼ 管理コンソールを使用して SIP メッセージ検査を有効にする

- 1 ブラウザで管理コンソールを開きます。
- 2 開発者プロファイルの場合は、「Application Server」→「ログ」→「ログレベル」の順に選択します。クラスタプロファイルの場合は、「設定」→「設定」→「ログ」→「ログ設定」→「ログレベル」の順に選択します。

- 3 「ログレベル」ページの「追加のプロパティ」領域で、**smi** プロパティを「**FINE**」に設定します。

▼ 管理 CLI を使用して SIP メッセージ検査を有効にする

- 1 端末ウィンドウで `asadmin` ユーティリティを開きます。
- 2 次のコマンドを使用します。

```
asadmin set server-config.log-service.module-log-levels.property.smi=FINE
```

サーバーログの表示

ログファイルを表示するには、次の手順に従います。

- 開発者プロファイルの場合は、「Application Server」→「ログ」→「ログファイルを表示」の順に選択します。
- クラスタおよびエンタープライズプロファイルの場合は、「設定」→「設定」→「ロガーの設定」→「一般」の順に選択し、「ログファイルを表示」をクリックします。

設定内容に基づいたログ結果を表示するには、「検索基準」領域に示されているオプションを使用します。

- インスタンス名: ドロップダウンリストからインスタンス名を選択して、そのサーバーインスタンスのログを表示します。デフォルトは現在のサーバーインスタンスです。
- ログファイル: ドロップダウンリストからログファイル名を選択して、そのログのコンテンツを表示します。デフォルトは `server.log` です。
- タイムスタンプ: 最新のメッセージを表示するには、「最新」(デフォルト)を選択します。特定期間内のメッセージだけを表示するには、「特定範囲」を選択して、表示される「開始」と「終了」フィールドに日付と時刻を入力します。時刻の値の構文は次の形式を必ず使用してください (SSS はミリ秒)。

hh:mm:ss.SSS

次に例を示します。

17:10:00.000

「開始」フィールドの値が「終了」フィールドの値よりも遅い場合は、エラーメッセージが表示されます。

- ログレベル: ログレベルでメッセージをフィルタリングするには、ドロップダウンリストからログレベルを選択します。デフォルトでは、サーバーログにある選択したログレベルおよびさらに重大なレベルのすべてのメッセージが表示に含ま

れます。選択したレベルのメッセージだけを表示するには、「より重度の高いメッセージをこれ以上含めない」というラベルの付いたチェックボックスを選択します。

表示したいメッセージを確実にサーバーログに表示するには、最初に「ログレベル」ページで適切なログレベルを設定してください。198 ページの「[ログレベルの設定](#)」を参照してください。

ログレベルに基づくログメッセージのフィルタリングを選択する場合、指定したフィルタリング基準と一致するメッセージだけが表示されます。しかし、このフィルタリングは、どのメッセージがサーバーログに記録されるかには影響しません。

最新の 40 エントリが「ログ設定」と「ログレベル」ページで指定した設定で表示されます。

最新のメッセージが最後に表示されるようにメッセージを並べ替えるには、タイムスタンプヘッダーのとなりの矢印をクリックします。

形式設定済みのメッセージを表示するには、次の印が付いたリンクをクリックします。

(details)

「ログエントリの詳細」というウィンドウが現れて、形式設定済みメッセージを表示します。

エントリのリストの末尾で、ボタンをクリックしてログファイルの古いエントリまたは新しいエントリを表示します。

「検索基準」領域で「詳細検索」をクリックして、ログビューアの詳細設定を行います。「詳細オプション」フィールドは次のように使用します。

- ログガー — モジュール別にフィルタリングするには、ドロップダウンリストから 1 つ以上の名前空間を選択します。shift-click または control-click を使い、複数の名前空間を選択します。

上位レベルの名前空間を 1 つ選ぶと、その下のすべての名前空間が選択されます。たとえば、`javax.enterprise.system` を選択すると、その名前空間の下にあるすべてのモジュールのログガーも選択されます。

`javax.enterprise.system.core`、`javax.enterprise.system.tools.admin` などです。

- カスタムログガー — 特定のアプリケーションに固有のログガーのメッセージを表示するには、テキストフィールドで 1 行に 1 つずつログガー名を入力します。アプリケーションに複数のモジュールがある場合は、そのいずれかまたはすべてを表示できます。たとえば、使用しているアプリケーションに次の名前のあるログガーがあるとしたします。

```
com.mycompany.myapp.module1
com.mycompany.myapp.module2
com.mycompany.myapp.module3
```

アプリケーション内のすべてのモジュールのメッセージを表示するには、`com.mycompany.myapp` と入力します。`module2` のメッセージだけを表示するには、`com.mycompany.myapp.module2` と入力します。

1つ以上のカスタムロガーを指定した場合、Communications Server モジュールのメッセージは、「ロガー」領域で明示的に指定されるときだけ表示されます。

- 名前と値のペア — 特定のスレッドの出力を表示するには、テキストフィールドにスレッドのキー名と値を入力します。キー名は `_ThreadID` です。次に例を示します。

```
_ThreadID=13
```

`com.mycompany.myapp.module2` がいくつかのスレッドで実行されるとします。1つのスレッドの出力だけを表示するようにログビューアを修正するには、「カスタムロガー」フィールドでモジュールのロガーを指定してからこのフィールドにスレッド ID を指定します。

- 表示 — 一度に 40 メッセージ (デフォルト) 以上を表示するには、ドロップダウンリストからほかの可能な値 (100、250、または 1000) を選択します。

スタックトレースを表示するには、「過度に長いメッセージを制限」チェックボックスのチェックマークを外します。デフォルトでは、スタックトレースはビューアに表示されませんが、メッセージの「(詳細)」リンクをクリックすると表示できます。

「基本検索」をクリックして、「詳細オプション」領域を非表示にします。

コンポーネントとサービスの監視

この章では、Communications Server の管理コンソールを使ってコンポーネントを監視する方法について説明します。この章で説明する内容は次のとおりです。

- 203 ページの「監視について」
- 230 ページの「監視の有効化と無効化」
- 231 ページの「監視データの表示」
- 249 ページの「JConsole の使用」

監視について

- 203 ページの「Communications Server での監視」
- 204 ページの「監視の概要」
- 204 ページの「監視可能なオブジェクトのツリー構造について」
- 208 ページの「監視対象のコンポーネントとサービスの統計について」

Communications Server での監視

監視機能を使用して Communications Server のサーバーインスタンスに配備されている各種コンポーネントおよびサービスの実行時状態を把握します。実行時コンポーネントとプロセスに関する情報を使用して、チューニングに関わるパフォーマンスボトルネックを識別し、処理能力を計画し、障害を見積もり、障害の場合の原因を分析して、期待どおりの機能性を確保できます。

監視をオンにすると、オーバーヘッドの増大によりパフォーマンスが低下します。

管理ルールを使用して、サーバーに関する潜在的な問題を通知することもできます。さらにオプションで、サーバーが一定のパフォーマンスしきい値に達したときに措置を講じることもできます。詳細については、[第 21 章「管理ルールの設定」](#)を参照してください。

監視の概要

Communications Server を監視するには、次の手順を実行します。

1. 管理コンソールまたは `asadmin` ツールを使用して、特定のサービスおよびコンポーネントの監視を有効にします。

この手順の詳細については、[230 ページの「監視の有効化と無効化」](#)を参照してください。

2. 管理コンソールまたは `asadmin` ツールを使用して、特定のサービスおよびコンポーネントの監視データを表示します。

この手順の詳細については、[231 ページの「監視データの表示」](#)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

Communications 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。監視オブジェクトのツリーは動的であり、インスタンス内におけるコンポーネントの追加、更新、削除に応じて変更されます。ツリー内のルートオブジェクトは、`server` などのサーバーインスタンス名です。開発者プロファイルでは、1つのサーバーインスタンスしか使用できません。

次のコマンドを実行すると、ツリーのトップレベルが表示されます。

```
asadmin> list --user adminuser --monitor server
server.applications
server.http-service
server.sip-service
server.connector-service
server.converged-load-balancer
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

次の各節では、これらのサブツリーについて説明します。

- [205 ページの「アプリケーションのツリー」](#)
- [205 ページの「SIP サービスのツリー」](#)
- [206 ページの「融合ロードバランサのツリー」](#)
- [206 ページの「HTTP サービスのツリー」](#)
- [207 ページの「リソースのツリー」](#)
- [207 ページの「コネクタサービスのツリー」](#)
- [207 ページの「JMS サービスのツリー」](#)
- [208 ページの「ORB のツリー」](#)

- 208 ページの「スレッドプールのツリー」

アプリケーションのツリー

次の図に、エンタープライズアプリケーションの各種コンポーネントのトップノードと子ノードを示します。監視統計が利用可能なノードには、アスタリスク(*)を付けています。詳細については、215 ページの「EJB コンテナの統計」を参照してください。

例 20-1 アプリケーションノードのツリー構造

```

applications
|--- application1
|   |--- ejb-module-1
|   |   |--- ejb1 *
|   |       |--- cache (for entity/sfsb) *
|   |       |--- pool (for slsb/mdb/entity) *
|   |       |--- methods
|   |           |---method1 *
|   |           |---method2 *
|   |           |--- stateful-session-store (for sfsb)*
|   |           |--- timers (for slsb/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 *
|   |       |--- cache (for entity/sfsb) *
|   |       |--- pool (for slsb/mdb/entity) *
|   |       |--- methods
|   |           |--- method1 *
|   |           |--- method2 *
|--- application2

```

SIP サービスのツリー

SIP サービスのノードを、次の図に示します。監視情報が利用可能なノードには、アスタリスク(*)を付けています。209 ページの「SIP サービスの統計」を参照してください。

例 20-2 SIP サービスのツリー構造

```

sip-service
  |-- network-manager*
      |-- request-received*
      |-- request-sent*
      |-- request-received*
      |-- request-sent*
      |-- sip-listener*
  |-- overload-protection-manager*
  |-- session-manager*
  |-- transaction-manager*

```

融合ロードバランサのツリー

融合ロードバランサのノードを、次の図に示します。監視情報が利用可能なノードには、アスタリスク (*) を付けています。213 ページの「融合ロードバランサの統計」を参照してください。

例 20-3 融合ロードバランサのツリー構造

```

converged-load-balancer

```

HTTP サービスのツリー

HTTP サービスのノードを、次の図に示します。監視情報が利用可能なノードには、アスタリスク (*) を付けています。221 ページの「HTTP サービスの統計」を参照してください。

例 20-4 HTTP サービスの図(開発者プロファイル版)

```

http-service
  |-- virtual-server-1
      |-- http-listener-1 *
      |-- http-listener-2 *
  |-- virtual-server-2
      |-- http-listener-1 *
      |-- http-listener-2 *

```

例 20-5 HTTP サービスの図(クラスタおよびエンタープライズプロファイル版)

```

http-service *
  |-- connection-queue *
  |-- dns *
  |-- file-cache *
  |-- keep-alive *

```

例 20-5 HTTP サービスの図(クラスタおよびエンタープライズプロファイル版) (続き)

```
|---virtual-server-1*
|         |--- request *
|---virtual-server-2*
|         |--- request *
```

リソースのツリー

リソースノードには、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。次の図に、各種リソースコンポーネントのトップノードと子ノードを示します。監視統計が利用可能なノードには、アスタリスク (*) を付けています。221 ページの「[JDBC 接続プールの統計](#)」を参照してください。

例 20-6 リソースの図

```
resources
  |---connection-pool1(either connector-connection-pool or jdbc)*
  |---connection-pool2(either connector-connection-pool or jdbc)*
```

コネクタサービスのツリー

コネクタサービスノードには、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。次の図に、各種コネクタサービスコンポーネントのトップノードと子ノードを示します。監視統計が利用可能なノードには、アスタリスク (*) を付けています。223 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。

例 20-7 コネクタサービスの図

```
connector-service
  |--- resource-adapter-1
  |         |-- connection-pools
  |         |         |-- pool-1 (All pool stats for this pool)
  |         |-- work-management (All work mgmt stats for this RA)
```

JMS サービスのツリー

JMS サービスノードには、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。次の図に、各種 JMS サービスコンポーネントのトップノードと子ノードを示します。監視統計が利用可能なノードには、アスタリスク (*) を付けています。

例 20-8 JMS サービスの図

```
jms-service
  |-- connection-factories [AKA conn. pools in the RA world]
```

例 20-8 JMS サービスの図 (続き)

```

|          |-- connection-factory-1 (All CF stats for this CF)
|-- work-management (All work mgmt stats for the MQ-RA)

```

ORB のツリー

ORB ノードには、接続マネージャーの監視可能な属性が格納されます。次の図に、各種 ORB コンポーネントのトップノードと子ノードを示します。監視統計が利用可能なノードには、アスタリスク (*) を付けています。224 ページの「ORB の接続マネージャーの統計」を参照してください。

例 20-9 ORB の図

```

orb
  |-- connection-managers
  |   |-- connection-manager-1 *
  |   |-- connection-manager-1 *

```

スレッドプールのツリー

スレッドプールノードには、接続マネージャーの監視可能な属性が格納されます。次の図に、各種 ORB コンポーネントのトップノードと子ノードを示します。監視統計が利用可能なノードには、アスタリスク (*) を付けています。224 ページの「スレッドプールの統計」を参照してください。

例 20-10 スレッドプールの図

```

thread-pools
  |-- thread-pool-1 *
  |-- thread-pool-2 *

```

監視対象のコンポーネントとサービスの統計について

この節では、利用可能な監視統計について説明します。

- 209 ページの「SIP サービスの統計」
- 213 ページの「融合ロードバランサの統計」
- 215 ページの「EJB コンテナの統計」
- 219 ページの「Web コンテナの統計」
- 221 ページの「HTTP サービスの統計」
- 221 ページの「JDBC 接続プールの統計」
- 223 ページの「JMS サービスおよびコネクタサービスの統計」
- 224 ページの「ORB の接続マネージャーの統計」

- 224 ページの「スレッドプールの統計」
- 225 ページの「トランザクションサービスの統計」
- 225 ページの「Java 仮想マシン (JVM) の統計」
- 226 ページの「Java SE の JVM 統計」

SIP サービスの統計

SIP サービスの統計を次の表に示します。

- 表 20-1
- 表 20-2

注 - 送信した要求で収集される統計は、表 20-2 に示す受信した要求の統計に似ています。統計は、ACK や BYE など、複数の SIP メッセージのタイプについて収集されます。

- 表 20-3

注 - 送信した応答で収集される統計は、表 20-3 に示す受信した応答の統計に似ています。統計は、1xx や 3xx など、複数の SIP メッセージのタイプについて収集されます。

- 表 20-4
- 表 20-5
- 表 20-6

表 20-1 ネットワークマネージャーと SIP リスナーの統計

属性名	データ型	説明
receivedsiprequests	CountStatistic	受信した SIP 要求の数 (再送を含む)。
sentsiprequests	CountStatistic	発信および中継で送信した SIP 要求の数 (再送とフォークを含む)。
receivedsipresponses	CountStatistic	受信した SIP 応答の数 (再送を含む)。
sentsipresponses	CountStatistic	発信および中継で送信した SIP 応答の数 (再送を含む)。

表 20-1 ネットワークマネージャーと SIP リスナーの統計 (続き)

属性名	データ型	説明
invalidsipmessages	CountStatistic	SIP メッセージとして認識されなかった受信メッセージの数 (再送を含む)。これらのメッセージは要求または応答として認識されないか、無効な応答コードによる応答として認識されます。

表 20-2 (ネットワークマネージャー) 受信した要求の統計

属性名	データ型	説明
ackreceivedsiprequests	CountStatistic	受信した ACK SIP 要求の数。
byereceivedsiprequests	CountStatistic	受信した BYE SIP 要求の数。
cancelreceivedsiprequests	CountStatistic	受信した CANCEL SIP 要求の数。
inforeceivedsiprequests	CountStatistic	受信した INFO SIP 要求の数。
invitereceivedsiprequests	CountStatistic	受信した INVITE SIP 要求の数。
joinreceivedsiprequests	CountStatistic	受信した JOIN SIP 要求の数。
messagereceivedsiprequests	CountStatistic	受信した MESSAGE SIP 要求の数。
notifyreceivedsiprequests	CountStatistic	受信した NOTIFY SIP 要求の数。
optionsreceivedsiprequests	CountStatistic	受信した OPTIONS SIP 要求の数。
otherreceivedsiprequests	CountStatistic	受信した OTHER SIP 要求の数。
prackreceivedsiprequests	CountStatistic	受信した PRACK SIP 要求の数。
publishreceivedsiprequests	CountStatistic	受信した PUBLISH SIP 要求の数。

表 20-2 (ネットワークマネージャー)受信した要求の統計 (続き)

属性名	データ型	説明
referreceivedsiprequests	CountStatistic	受信した REFER SIP 要求の数。
registerreceivedsiprequests	CountStatistic	受信した REGISTER SIP 要求の数。
subscribereceivedsiprequests	CountStatistic	受信した SUBSCRIBE SIP 要求の数。
updatereceivedsiprequests	CountStatistic	受信した UPDATE SIP 要求の数。

表 20-3 (ネットワークマネージャー)受信した応答の統計

属性名	データ型	説明
1xxreceivedsipresponses	CountStatistic	受信した 1xx SIP 応答の数。
2xxreceivedsipresponses	CountStatistic	受信した 2xx SIP 応答の数。
3xxreceivedsipresponses	CountStatistic	受信した 3xx SIP 応答の数。
420receivedsipresponses	CountStatistic	受信した 420 SIP 応答の数。
4xxreceivedsipresponses	CountStatistic	受信した 4xx SIP 応答の数。
5xxreceivedsipresponses	CountStatistic	受信した 5xx SIP 応答の数。
6xxreceivedsipresponses	CountStatistic	受信した 6xx SIP 応答の数。

表 20-4 過負荷保護マネージャーの統計

属性名	データ型	説明
overloadrejectedsiprequests	CountStatistic	過負荷状態のため拒否された SIP 要求の数。
overloadrejectedhttprequests	CountStatistic	過負荷状態のため拒否された HTTP 要求の数。

表 20-5 セッションマネージャーの統計

属性名	データ型	説明
concurrentsipdialogs	CountStatistic	現在ノードに保存されている SIP ダイアログの数。
failedsipdialogs	CountStatistic	失敗した SIP ダイアログ設定の数。アプリケーションまたは信号パスの1つあとのノードが、4xx-6xx 応答コードで応答しています。
expiredsipdialogs	CountStatistic	セッションの有効期限切れにより削除された SIP ダイアログの数。
sucessfulsipdialogs	CountStatistic	正常に確立された SIP ダイアログの数。
totalsipdialogcount	CountStatistic	SIP ダイアログの合計数。この数は、sucessfulsipdialogs + expiredsipdialogs + failedsipdialogs - concurrentsipdialogs として計算されます。
totalsipdialoglifetime	CountStatistic	SIP ダイアログのライフタイムの合計 (ミリ秒単位)。

表 20-6 トランザクションマネージャーの統計

属性名	データ型	説明
sipservertransactions	CountStatistic	処理中または終了状態に達した SIP サーバートランザクションの数。
sipclienttransactions	CountStatistic	処理中または終了状態に達した SIP クライアントトランザクションの数。
totalsiptransactioncount	CountStatistic	SIP トランザクションの合計数。
totalsiptransactiontime	CountStatistic	すべての SIP トランザクションの応答時間の合計 (ミリ秒単位)。

融合ロードバランサの統計

表 20-7 融合ロードバランサのフロントエンドの統計

属性名	データ型	説明
TotalIncomingSipRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサが受信した SIP 要求の合計数。
TotalIncomingHttpRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサが受信した HTTP 要求の合計数。
TotalProxiedHttpRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、プロキシのリモートバックエンドで処理される HTTP 要求の合計数。
TotalLocalSipResponsesFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、同じインスタンスのスタックに転送されて処理された SIP 応答の合計数。
TotalOutgoingSipResponsesFE	CountStatistic	バックエンドインスタンスから (フロントエンドとして動作している) 融合ロードバランサで受信され、クライアントに送信された SIP 要求の合計数。
TotalIncomingSipResponsesFE	CountStatistic	フロントエンドとして動作している融合ロードバランサが受信した SIP 応答の合計数。
TotalProxiedSipRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、プロキシのリモートバックエンドで処理された SIP 要求の合計数。

表 20-7 融合ロードバランサのフロントエンドの統計 (続き)

属性名	データ型	説明
TotalDiscardedSipResponsesFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、コンシューマにルーティングできないために破棄される SIP 要求の合計数。
TotalErrorHttpResponsesAtCLBLayerFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、クライアントにエラー応答を返す HTTP 要求の合計数。
TotalFailedOverSipRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、この要求を処理するインスタンスが使用できないために、別のインスタンスにフェイルオーバーされた SIP 応答の合計数。
TotalFailedOverHttpRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、この要求を処理するインスタンスが使用できないために、別のインスタンスにフェイルオーバーされた HTTP 要求の合計数。
TotalLocalHttpRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、同じインスタンスのスタックに転送されて処理された HTTP 要求の合計数。
TotalProxiedSipResponsesFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、リモートバックエンドにプロキシされて処理された SIP 応答の合計数。

表 20-7 融合ロードバランサのフロントエンドの統計 (続き)

属性名	データ型	説明
TotalErrorSipResponsesAtCLBLayerFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、クライアントにエラー応答を返した SIP 要求の合計数。
TotalLocalSipRequestsFE	CountStatistic	フロントエンドとして動作している融合ロードバランサで受信され、同じインスタンスのスタックに転送されて処理された SIP 要求の合計数。

表 20-8 融合ロードバランサのフロントエンドの統計

属性名	データ型	説明
TotalIncomingSipRequestsBE	CountStatistic	バックエンドとして動作している融合ロードバランサで処理された受信 SIP 要求の合計数。
TotalIncomingHttpRequestsBE	CountStatistic	バックエンドとして動作している融合ロードバランサで処理された受信 SIP 要求の合計数。
TotalIncomingSipResponsesBE	CountStatistic	バックエンドとして動作している融合ロードバランサで処理された受信 SIP 応答の合計数。
TotalOutgoingSipResponsesBE	CountStatistic	バックエンドとして動作している融合ロードバランサで処理された送信 SIP 応答の合計数。
TotalOutgoingSipRequestsBE	CountStatistic	バックエンドとして動作している融合ロードバランサで処理された送信 SIP 要求の合計数。

EJB コンテナの統計

EJB コンテナの統計を次の表に示します。

- 表 20-9
- 表 20-10

- 表 20-11
- 表 20-12
- 表 20-13

EJB 統計を次の表に示します。

表 20-9 EJB 統計

属性名	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。
removecount	CountStatistic	特定の EJB に対する remove メソッドの呼び出し回数。
pooledcount	RangeStatistic	プールされた状態にあるエンティティ Bean の数。
readycount	RangeStatistic	実行可能状態にあるエンティティ Bean の数。
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。
activationErrorCount	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 Bean に対して収集されます。

EJB メソッド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 20-10 EJB メソッドの統計

属性名	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。
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。

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 20-11 EJB プールの統計

属性名	データ型	説明
numbeansinpool	BoundedRangeStatistic	関連付けられたプール内の EJB 数。これにより、プールがどのように変化しているかがわかります。
numthreadswaiting	BoundedRangeStatistic	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。

表 20-11 EJB プールの統計 (続き)

属性名	データ型	説明
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。
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。

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 20-12 EJB キャッシュの統計

属性名	データ型	説明
cachemisses	BoundedRangeStatistic	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	BoundedRangeStatistic	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	BoundedRangeStatistic	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。

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 20-13 タイマーの統計

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。

Web コンテナの統計

Web コンテナは、[205 ページの「アプリケーションのツリー」](#)に示したオブジェクトツリー内に含まれます。Web コンテナの統計は、個々の Web アプリケーションごとに表示されます。Web コンテナのサーブレットに関して利用可能な統計を[表 20-14](#)に、Web モジュールに関して利用可能な統計を[表 20-15](#)に示します。

表 20-14 Web コンテナ (サーブレット) の統計

Statistic	単位	データ型	コメント
errorcount	番号	CountStatistic	応答コードが 400 以上になった場合の累計件数。
maxtime	ミリ秒	CountStatistic	Web コンテナの要求待ち状態の最大継続時間。
processingtime	ミリ秒	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	番号	CountStatistic	その時点までに処理された要求の合計数。

Web モジュールに関して利用可能な統計を、[219 ページの「Web コンテナの統計」](#)に示します。

表 20-15 Web コンテナ (Web モジュール) の統計

Statistic	データ型	コメント
jspcount	CountStatistic	この Web モジュール内に読み込まれた JSP ページの数。

表 20-15 Web コンテナ (Web モジュール) の統計 (続き)

Statistic	データ型	コメント
jspreloadcount	CountStatistic	この Web モジュール内に再読み込みされた JSP ページの数。
sessionstotal	CountStatistic	この Web モジュールに対して作成されたセッションの合計数。
activesessionscurrent	CountStatistic	この Web モジュールで現在アクティブになっているセッションの数。
activesessionshigh	CountStatistic	この Web モジュールで同時にアクティブになれるセッションの最大数。
rejectedsessiontotal	CountStatistic	この Web モジュールで拒否されたセッションの合計数。これは、最大許可セッション数がすでにアクティブになっていたために作成されなかったセッションの数です。
expiredsessiontotal	CountStatistic	この Web モジュールで期限切れになったセッションの合計数。
sessionsize	AverageRangeStatistic	この Web モジュールのセッションのサイズ。値は high、low、average のいずれかです。ただし、直列化されたセッションの場合はバイト値になります。
sessionpersisttime	AverageRangeStatistic	この Web モジュールの HTTP セッション状態のバックエンドストアへの持続化に要した時間 (ミリ秒値、low、high、average のいずれか)。
cachedsessionscurrent	CountStatistic	この Web モジュールで現在メモリー内にキャッシュされているセッションの数。
passivatedsessionscurrent	CountStatistic	この Web モジュールで現在非活性化されているセッションの数。

HTTP サービスの統計

開発者プロファイルの 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 20-16 HTTP サービスの統計 (開発者プロファイル)

Statistic	単位	データ型	コメント
bytesreceived	バイト	CountStatistic	各要求プロセッサが受信したバイトの累計値。
bytessent	バイト	CountStatistic	各要求プロセッサが送信したバイトの累計値。
currentthreadcount	番号	CountStatistic	リスナースレッドプール内に現在存在している処理スレッドの数。
currentthreadsbusy	番号	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。
errorcount	番号	CountStatistic	エラー回数の累計値。これは、応答コードが 400 以上になった場合の回数を表します。
maxsparethreads	番号	CountStatistic	存在可能な未使用応答処理スレッドの最大数。
minsparethreads	番号	CountStatistic	存在可能な未使用応答処理スレッドの最小数。
maxthreads	番号	CountStatistic	リスナーが作成する要求処理スレッドの最大数。
maxtime	ミリ秒	CountStatistic	スレッド処理時間の最大値。
processing-time	ミリ秒	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。
request-count	番号	CountStatistic	その時点までに処理された要求の合計数。

JDBC 接続プールの統計

JDBC リソースを監視することで、パフォーマンスを測定するとともに、実行時のリソースの使用状況を把握します。JDBC 接続の作成はコストのかかる処理であり、アプリケーションのパフォーマンス上のボトルネックになることが多いため、JDBC 接

続プールで新しい接続がどのように解放/作成されているかや、特定のプールから接続を取得するために待機しているスレッドがどれくらい存在するかを監視することが不可欠です。

JDBC 接続プールに関して利用可能な統計を、次の表に示します。

表 20-17 JDBC 接続プールの統計

Statistic	単位	データ型	説明
numconnfailedvalidation	番号	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	番号	RangeStatistic	接続の使用状況に関する統計を提供します。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	番号	CountStatistic	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。
numconntimeout	番号	BoundedRangeStatistic	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。
averageconnwaittime	番号	CountStatistic	コネクタ接続プールに対する接続要求が成功した場合の平均接続待ち時間を示します。
waitqueuelength	番号	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。
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。
numconncreated	ミリ秒	CountStatistic	前回のリセット後に作成された物理接続の数。
numconndestroyed	番号	CountStatistic	前回のリセット後に破棄された物理接続の数。

表 20-17 JDBC 接続プールの統計 (続き)

Statistic	単位	データ型	説明
numconnacquired	番号	CountStatistic	プールから取得された論理接続の数。
numconnreleased	番号	CountStatistic	プールに解放された論理接続の数。

JMS サービスおよびコネクタサービスの統計

コネクタ接続プールに関して利用可能な統計を、表 20-18 に示します。コネクタ作業管理に関する統計を、表 20-19 に示します。

表 20-18 コネクタ接続プールの統計

Statistic	単位	データ型	説明
numconnfailedvalidation	番号	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	番号	RangeStatistic	接続の使用状況に関する統計を提供します。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	番号	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。
numconntimedout	番号	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。
averageconnwaittime	番号	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。
waitqueuelenght	番号	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。
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。
numconncreated	ミリ秒	CountStatistic	前回のリセット後に作成された物理接続の数。
numconndestroyed	番号	CountStatistic	前回のリセット後に破棄された物理接続の数。

表 20-18 コネクタ接続プールの統計 (続き)

Statistic	単位	データ型	説明
numconnacquired	番号	CountStatistic	プールから取得された論理接続の数。
numconnreleased	番号	CountStatistic	プールに解放された論理接続の数。

コネクタ作業管理に関して利用可能な統計を、次の図に示します。

表 20-19 コネクタ作業管理の統計

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。
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。
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。
rejectedworkcount	CountStatistic	Communications Server によって拒否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。

ORB の接続マネージャーの統計

ORB の接続マネージャーに関して利用可能な統計を、次の図に示します。

表 20-20 ORB の接続マネージャーの統計

Statistic	単位	データ型	説明
connectionsidle	番号	CountStatistic	ORB への接続のうち、アイドル状態のもの合計数を提供します。
connectionsinuse	番号	CountStatistic	ORB への接続のうち、使用中のもの合計数を提供します。
totalconnections	番号	BoundedRangeStatistic	ORB への接続の合計数。

スレッドプールの統計

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 20-21 スレッドプールの統計

Statistic	単位	データ型	説明
averagetimeinqueue	ミリ秒	RangeStatistic	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)。
averageworkcompletion-time	ミリ秒	RangeStatistic	1つの作業の平均完了時間(ミリ秒)。
currentnumberofthreads	番号	BoundedRangeStatistic	要求処理スレッドの現在の数。
numberofavailablethreads	番号	CountStatistic	利用可能なスレッドの数。
numberofbusythreads	番号	CountStatistic	ビジー状態のスレッドの数。
totalworkitemsadded	番号	CountStatistic	その時点までに作業キューに追加された作業項目の合計数。

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えば、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できます。フリーズすると、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したりできます。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 20-22 トランザクションサービスの統計

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。

Java 仮想マシン (JVM) の統計

JVM の監視可能な属性は、常に有効になっています。JVM に関して利用可能な統計を、次の表に示します。

表 20-23 JVM の統計

Statistic	データ型	説明
heapsize	BoundedRangeStatistic	JVM のメモリーヒープサイズの上 限と下限の間にある常駐メモ リーフットプリント。
uptime	CountStatistic	JVM の稼働時間。

Java SE の JVM 統計

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については、『Monitoring and Management for the Java Platform』というタイトルの文書を参照してください。この文書は、<http://java.sun.com/javase/6/docs/technotes/guides/management/> にあります。

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、次の図に示します。

表 20-24 Java SE の JVM 統計 - クラス読み込み

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。
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 で利用可能なコンパイル関連の統計を、次の図に示します。

表 20-25 Java SE の JVM 統計 - コンパイル

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)。

Java SE の JVM で利用可能なガベージコレクション関連の統計を、次の図に示し
ます。

表 20-26 Java SE の JVM 統計 - ガベージコレクション

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	コレクション時間の累計値(ミリ秒)。

Java SE の JVM で利用可能なメモリー関連の統計を、次の図に示します。

表 20-27 Java SE の JVM 統計 - メモリー

Statistic	データ型	説明
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。
initheapsize	CountStatistic	JVM が最初に要求したヒープのサイズ。
usedheapsize	CountStatistic	現在使用されているヒープのサイズ。
maxheapsize	CountStatistic	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(バイト)。
committedheapsize	CountStatistic	JVM 用としてコミットされたメモリーのサイズ(バイト)。
initnonheapsize	CountStatistic	JVM が最初に要求した非ヒープ領域のサイズ。
usednonheapsize	CountStatistic	現在使用されている非ヒープ領域のサイズ。
maxnonheapsize	CountStatistic	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(バイト)。
committednonheapsize	CountStatistic	JVM 用としてコミットされたメモリーのサイズ(バイト)。

Java SE の JVM で利用可能なオペレーティングシステム関連の統計を、次の図に示します。

表 20-28 Java SE の JVM 統計 - オペレーティングシステム

Statistic	データ型	説明
arch	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	StringStatistic	オペレーティングシステムの名前。
version	StringStatistic	オペレーティングシステムのバージョン。

Java SE の JVM で利用可能なランタイム関連の統計を、次の図に示します。

表 20-29 Java SE の JVM 統計 - ランタイム

Statistic	データ型	説明
name	StringStatistic	実行中の JVM を表す名前
vmname	StringStatistic	JVM 実装の名前。
vmvendor	StringStatistic	JVM 実装のベンダー。
vmversion	StringStatistic	JVM 実装のバージョン。
specname	StringStatistic	JVM 仕様の名前。
specvendor	StringStatistic	JVM 仕様のベンダー。
specversion	StringStatistic	JVM 仕様のバージョン。
managementspecversion	StringStatistic	JVM が実装している管理仕様のバージョン。
classpath	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。
librarypath	StringStatistic	Java のライブラリパス。
bootclasspath	StringStatistic	ブートストラップ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。
inputarguments	StringStatistic	JVM に渡された入力引数。main メソッドに対する引数は含みません。
uptime	CountStatistic	JVM の稼働時間(ミリ秒)。

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 20-30 Java SE の JVM 統計 - ThreadInfo

Statistic	データ型	説明
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。
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。
blockedcount	CountStatistic	このスレッドが BLOCKED 状態に入った合計回数。
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。
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 <code>Object.wait</code> 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。

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 20-31 Java SE の JVM 統計 - スレッド

Statistic	データ型	説明
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。

表 20-31 Java SE の JVM 統計 - スレッド (続き)

Statistic	データ型	説明
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。
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。
daemonthreadcount	CountStatistic	ライブデーモンスレッドの現在の数。
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。
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。

監視の有効化と無効化

ここでは、次の内容について説明します。

- 230 ページの「管理コンソールを使用した監視レベルの設定」
- 231 ページの「asadmin を使用して監視レベルを設定する」

管理コンソールを使用した監視レベルの設定

管理コンソールを使用して監視を設定するには、次の手順に従います。

- 開発者プロファイルの場合は、「設定」→「監視」の順に選択します。
- クラスタおよびエンタープライズプロファイルの場合は、「設定」→「設定」→「監視」の順に選択します。

デフォルトでは、すべてのコンポーネントおよびサービスについて監視はオフになっています。監視をオンにするには、コンボボックスから「LOW」または「HIGH」を選択します。監視をオフにするには、コンボボックスから「OFF」を選択します。

監視の設定の詳細については、管理コンソールで利用可能なオンラインヘルプを参照してください。

▼ **asadmin** を使用して監視レベルを設定する

- 1 get コマンドを使って監視が現在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 --user admin-user server.monitoring-service.module-monitoring-levels.*
```

次の結果が返されます。

```
server.monitoring-service.module-monitoring-levels.  
connector-connection-pool = OFF  
server.monitoring-service.module-monitoring-levels.  
connector-service = OFF  
server.monitoring-service.module-monitoring-levels.ejb-container = OFF  
server.monitoring-service.module-monitoring-levels.http-service = OFF  
server.monitoring-service.module-monitoring-levels.sip-service = OFF  
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool = OFF  
server.monitoring-service.module-monitoring-levels.jms-service = OFF  
server.monitoring-service.module-monitoring-levels.jvm = OFF  
server.monitoring-service.module-monitoring-levels.orb = OFF  
server.monitoring-service.module-monitoring-levels.thread-pool = OFF  
server.monitoring-service.module-monitoring-levels.transaction-service = OFF  
server.monitoring-service.module-monitoring-levels.web-container = OFF
```

- 2 set コマンドを使って監視を有効にします。

たとえば、HTTP サービスの監視を有効にするには、次のようにします。

```
asadmin> set --user admin-user
```

```
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

監視を無効にするには、set コマンドを使って監視レベルに OFF を指定します。

監視データの表示

- [231 ページの「管理コンソールでの監視データの表示」](#)
- [232 ページの「asadmin ツールによる監視データの表示」](#)

管理コンソールでの監視データの表示

開発者プロファイルの場合、監視データを表示するには、「Application Server」→「監視」の順に選択します。

クラスタおよびエンタープライズプロファイルの場合、スタンドアロンインスタンスの監視データを表示するには、「スタンドアロンインスタンス」→「インスタンス」→「監視」の順に選択します。クラスタ化されたインスタンスの監視データを表示するには、「クラスタ」→「クラスタ」→「インスタンス」→「監視」の順に選択します。

JVM、サーバー、アプリケーション、スレッドプール、SIP サービス、HTTP サービス、トランザクションサービス、ログ統計、および呼び出しフロー統計の監視データを選択して表示できます。204 ページの「監視可能なオブジェクトのツリー構造について」を参照してください。

監視の表示または設定の詳細については、管理コンソールで利用可能なオンラインヘルプを参照してください。

各コンポーネントやサービスの属性の詳細については、208 ページの「監視対象のコンポーネントとサービスの統計について」を参照してください。

asadmin ツールによる監視データの表示

ここでは、次の内容について説明します。

- 232 ページの「`asadmin monitor` コマンドを使用して監視データを表示する」
- 233 ページの「`asadmin get` および `asadmin list` コマンドを使用して監視データを表示する」
- 234 ページの「ドット表記名とその指定方法について」
- 236 ページの「`list` コマンドと `get` コマンドの例」
- 236 ページの「`list --user admin-user --monitor` コマンドの例」
- 237 ページの「`get --user admin-user --monitor` コマンドの例」
- 238 ページの「PetStore サンプルを使用する」
- 241 ページの「すべてのレベルにおける `list` コマンドと `get` コマンドの予想出力」

▼ `asadmin monitor` コマンドを使用して監視データを表示する

`asadmin` で監視データを表示する方法は2とおりあります。最初の方法では、`monitor` コマンドを使用します。このコマンドは、一般的な監視統計を出力します。統計情報を絞り込むためのオプションと、出力をコンマ区切り (CSV) ファイルに取り込むためのオプションがあります。

- 1 監視データを表示するには、`monitor` コマンドを使用して、監視データのタイプを指定します。監視データのタイプには、`httplistener`、`siplistener`、`keepalive`、`filecache`、`connectionqueue`、`jdbcpool`、

jvm、threadpool、servlet、connection、connectorpool、endpoint、entitybean、messagedriven、statefulsession、statelesssession、httpservice、またはwebmoduleがあります。

たとえば、server上のjvmのデータを表示するには、次のように入力します。

```
asadmin>monitor --type jvm --user adminuser server
```

```

                                JVM Monitoring
UpTime(ms)                                HeapSize(bytes)
current          min          max          low          high          count
327142979        0          531628032  0          45940736  45940736

```

- 2 監視データを表示してCSVファイルに出力を送信するには、**filename** オプションを使用します。次に例を示します。

```
asadmin> monitor --type jvm --filename myoutputfile --user adminuser server
```

▼ **asadmin get** および **asadmin list** コマンドを使用して監視データを表示する

ほとんどの状況で **monitor** コマンドを使用できます。ただし、**monitor** コマンドでは、監視可能なすべてのオブジェクトの完全なリストは表示できません。**asadmin** ツールを使用して監視可能なすべてのデータを表示するには、**asadmin list** および **asadmin get** コマンドに続けて監視可能なオブジェクトのドット表記名を使用します。手順を次に示します。

- 1 監視可能なオブジェクトの名前を表示するには、**asadmin list** コマンドを使用します。

たとえば、サーバーインスタンスで監視が有効なアプリケーションコンポーネントおよびサブシステムのリストを表示するには、次のコマンドを端末ウィンドウに入力します。

```
asadmin> list --user adminuser --monitor server
```

前述のコマンドは、監視が有効なアプリケーションコンポーネントおよびサブシステムの次のようなリストを返します。

```

server.resources
server.connector-service
server.orb
server.jms-service
server.jvm
server.applications
server.http-service
server.sip-service
server.thread-pools

```

list コマンドのその他の使用例については、236 ページの「list コマンドと get コマンドの例」を参照してください。list コマンドで使用できるドット表記名の詳細については、234 ページの「ドット表記名とその指定方法について」を参照してください。

- 2 監視が有効なアプリケーションコンポーネントまたはサブシステムの監視統計を表示するには、`asadmin get` コマンドを使用します。

統計を取得するには、`asadmin get` コマンドを端末ウィンドウに入力して、前述の手順の list コマンドで表示された名前を指定します。次の例では、特定のオブジェクトのサブシステムからすべての属性を取得します。

```
asadmin> get --user adminuser --monitor server.jvm.*
```

このコマンドは次の属性およびデータを返します。

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
    the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
    been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

get コマンドのその他の使用例については、236 ページの「list コマンドと get コマンドの例」を参照してください。get コマンドで使用できるドット表記名の詳細については、234 ページの「ドット表記名とその指定方法について」を参照してください。

ドット表記名とその指定方法について

`asadmin list` コマンドと `asadmin get` コマンドでは、監視可能オブジェクトのドット表記名を指定します。すべての子オブジェクトのアドレス指定にはドット (.) 文字が区切り文字として使用され、それらの名前は「ドット表記名」と呼ばれます。子ノードが単独タイプの場合、その監視オブジェクトタイプを指定するだけで、そのオブジェクトを指定できます。それ以外の場合は、`type.name` 形式の名前を指定する必要があります。

たとえば、`http-service` は、有効な監視可能オブジェクトタイプの1つであり、単独タイプです。インスタンス `server` の `http-service` を表す単独タイプの子ノードを指定する場合、ドット表記名は次のようになります。

```
server.http-service
```

もう1つ例を挙げます。`applications` は、有効な監視可能オブジェクトタイプですが、単独タイプではありません。たとえば、アプリケーション `PetStore` を表す、単独タイプでない子ノードを指定するには、ドット表記名は次のようになります。

```
server.applications.petstore
```

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、`http-service` には、`bytesreceived-laststampltime` という名前の監視可能な属性があります。次の名前は、`bytesreceived` 属性を指定していることとなります。

```
server.http-service.server.http-listener-1.  
bytesreceived-laststampltime
```

管理者は、`asadmin list` コマンドと `asadmin get` コマンドの有効なドット表記名を覚えておく必要はありません。`list` コマンドを使えば、利用可能な監視可能オブジェクトが表示され、ワイルドカードパラメータ付きの `get` コマンドを使えば、任意の監視可能オブジェクトで利用可能なすべての属性を確認することができます。

`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、根本的に次のことを前提としています。

- `list` コマンドでドット表記名の後にワイルドカード (*) が指定されていなかった場合、現在のノードの直接の子ノードが結果として返される。たとえば、`list --user adminuser --monitor server` を実行した場合、`server` ノードに属するすべての直接の子ノードが一覧表示される。
- `list` コマンドでドット表記名の後に `.*` 形式のワイルドカードが指定されていた場合、現在のノードの子ノード階層ツリーが結果として返される。たとえば、`list --user adminuser --monitor server.applications.*` を実行した場合、`applications` のすべての子ノードに加え、それらの配下にある子ノードなども一覧表示される。
- `list` コマンドで `*dottedname`、`dotted * name`、`dotted name *` のいずれかの形式でドット表記名の前後にワイルドカードが指定されていた場合、そのマッチングパターンによって生成された正規表現にマッチするすべてのノードとそれらの子ノードが、結果として返される。
- `get` コマンドの末尾に `「.*」`、`「*」` のいずれかが指定されていた場合、マッチング対象の現在のノードに属する属性とその値のセットが、結果として返される。

詳細については、241 ページの「すべてのレベルにおける `list` コマンドと `get` コマンドの予想出力」を参照してください。

list コマンドと get コマンドの例

ここでは、次の内容について説明します。

- [236 ページの「list --user admin-user --monitor コマンドの例」](#)
- [237 ページの「get --user admin-user --monitor コマンドの例」](#)

list --user admin-user --monitor コマンドの例

list コマンドは、指定されたサーバーインスタンス名で現在監視されているアプリケーションコンポーネントやサブシステムに関する情報を提供します。このコマンドを使えば、特定のサーバーインスタンスの監視可能なコンポーネントやそのサブコンポーネントを表示できます。list のより詳しい例については、[241 ページの「すべてのレベルにおける list コマンドと get コマンドの予想出力」](#)を参照してください。

例 1

```
asadmin> list --user admin-user --monitor server
```

前述のコマンドは、監視が有効なアプリケーションコンポーネントおよびサブシステムの次のようなリストを返します。

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

また、指定されたサーバーインスタンス内で現在監視されているアプリケーションを一覧表示することも可能です。これは、get コマンドを使って特定のアプリケーションの特定の監視統計を取得する場合に便利です。

例 2

```
asadmin> list --user admin-user --monitor server.applications
```

次の結果が返されます。

```
server.applications.adminapp
  server.applications.admingui
server.applications.myApp
```

より包括的な例については、[238 ページの「PetStore サンプルを使用する」](#)を参照してください。

get --user admin-user --monitor コマンドの例

このコマンドは、次の監視対象情報を取得します。

- 特定のコンポーネントまたはサブシステム内で監視されているすべての属性
 - 特定のコンポーネントまたはサブシステム内で監視されている特定の属性
- 特定のコンポーネントまたはサブシステムに存在しない属性が要求された場合、エラーが返されます。同様に、特定のコンポーネントまたはサブシステムのアクティブでない属性が要求された場合も、エラーが返されます。
- get コマンドの使用方法の詳細については、[241 ページの「すべてのレベルにおける list コマンドと get コマンドの予想出力」](#)を参照してください。

例 1

特定のオブジェクトのすべての属性をサブシステムから取得します。

```
asadmin> get --user admin-user --monitor server.jvm.*
```

次の結果が返されます。

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

例 2

特定の Java EE アプリケーションからすべての属性を取得します。

```
asadmin> get --user admin-user --monitor server.applications.myJavaEEApp.*
```

次の結果が返されます。

```
No matches resulted from the wildcard expression.  
CLI137 Command get failed.
```

Java EE アプリケーションレベルで公開されている監視可能な属性が存在しないため、このような応答が表示されました。

例 3

特定のサブシステムから特定の属性を取得します。

```
asadmin> get --user admin-user --monitor server.jvm.uptime-lastsampletime
```

次の結果が返されます。

```
server.jvm.uptime-lastsampletime = 1093215374813
```

例 4

特定のサブシステム属性内から未知の属性を取得します。

```
asadmin> get --user admin-user --monitor server.jvm.badname
```

次の結果が返されます。

```
No such attribute found from reflecting the corresponding Stats  
interface: [badname]  
CLI137 Command get failed.
```

▼ PetStore サンプルを使用する

次の例は、asadmin ツールを監視目的でどのように使えばよいかを示したものです。

あるユーザーが、Communications Server 上に配備済みのサンプル Petstore アプリケーションに含まれる特定のメソッドの呼び出し回数を調査しようとしています。その配備先インスタンスの名前は、server です。list コマンドと get コマンドを併用することで、そのメソッドの目的の統計情報にアクセスします。

- 1 **Communications Server** と asadmin ツールを起動します。
- 2 いくつかの有用な環境変数を設定することで、それらの値をコマンドごとに入力しないで済むようにします。

```
asadmin> export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123  
asadmin> export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4848
```

- 3 インスタンス server の監視可能なコンポーネントを一覧表示します。

```
asadmin> list --user adminuser --monitor server*
```

次のような出力結果が返されます。

```
server
server.applications
server.applications.CometEJB
server.applications.ConverterApp
server.applications.petstore
server.http-service
server.resources
server.thread-pools
```

この監視可能なコンポーネントの一覧には、thread-pools、http-service、resources、および配備済みで有効化されているすべての applications が含まれています。

- 4 PetStore アプリケーションの監視可能なサブコンポーネントを一覧表示します (--monitor の代わりに -m を使用可能)。

```
asadmin> list -m server.applications.petstore
```

次の結果が返されます。

```
server.applications.petstore.signon-ejb_jar
server.applications.petstore.catalog-ejb_jar
server.applications.petstore.uidgen-ejb_jar
server.applications.petstore.customer-ejb_jar
server.applications.petstore.petstore-ejb_jar
server.applications.petstore.petstore\war
server.applications.petstore.AsyncSenderJAR_jar
server.applications.petstore.cart-ejb_jar
```

- 5 Petstore アプリケーションの EJB モジュール signon-ejb_jar の監視可能なサブコンポーネントを一覧表示します。

```
asadmin> list -m server.applications.petstore.signon-ejb_jar
```

次の結果が返されます。

```
server.applications.petstore.signon-ejb_jar.SignOnEJB
server.applications.petstore.signon-ejb_jar.UserEJB
```

- 6 Petstore アプリケーションの EJB モジュール signon-ejb_jar のエンティティ Bean UserEJB に含まれる監視可能なサブコンポーネントを一覧表示します。

```
asadmin> list -m server.applications.petstore.signon-ejb_jar.UserEJB
```

次の結果が返されます (ドット表記名はスペースの関係で削除してある)。

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
```

- 7 PetStore アプリケーションの EJB モジュール `signon-ejb_jar` のエンティティ `Bean UserEJB` に含まれるメソッド `getUserName` 内の監視可能なサブコンポーネントを一覧表示します。

```
asadmin> list -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName
```

次の結果が返されます。

```
Nothing to list at server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName. To get the valid names beginning with a
string, use the wildcard "*" character. For example, to list all names
that begin with "server", use "list server*".
```

- 8 メソッドに対する監視可能なサブコンポーネントは存在しません。メソッド `getUserName` の監視可能なすべての統計を取得します。

```
asadmin> get -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName.*
```

次の結果が返されます。

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-description = Provides the time in milliseconds
    spent during the last successful/unsuccessful attempt to execute the
    operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = Provides the number of times an
    operation was called, the total time that was spent during the
    invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
```

```

server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
  server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of errors
  that ocured during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
  successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count

```

- 9 また、実行回数など、特定の統計を取得するには、次のようなコマンドを使用します。

```

asadmin> get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count

```

次の結果が返されます。

```

server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1

```

すべてのレベルにおける **list** コマンドと **get** コマンドの予想出力

次の各表は、ツリーの各レベルにおけるコマンド、ドット表記名、および対応する出力を示したものです。

表 20-32 トップレベル

コマンド	ドット表記名	出力
list -m	server	server.applicationsserver.thread-poolserver. resourceeserver.http-serviceserver.transaction- serviceserver.orb.connection-managersserver.orb. connection-managers.orb\Connections\Inbound\ AcceptedConnectionsserver.jvm
list -m	server.*	このノードから下の子ノード階層。
get -m	server.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。

次の表に、アプリケーションレベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-33 アプリケーションレベル

コマンド	ドット表記名	出力
list -m	server.applications または *applications	appl1app2web-module1_warejb-module2_jar...
list -m	server.applications.* または *applications.*	このノードから下の子ノード階層。
get -m	server.applications.* または *applications.*	このノードに属性が存在しないことを示す メッセージだけが表示されます。

次の表に、アプリケーションレベルのスタンドアロンモジュールとエンタープライズアプリケーションのコマンド、ドット表記名、および対応する出力を示します。

表 20-34 アプリケーション-エンタープライズアプリケーションとスタンドアロンモジュール

コマンド	ドット表記名	出力
list -m	server.applications.app1 または *app1 注: このレベルが適用可能なのは、エンタープライズアプリケーションが配備されている場合だけです。スタンドアロンモジュールが配備されている場合には適用できません。	ejb-module1_jarweb-module2_warejb-module3_jarweb-module3_war...
list -m	server.applications.app1.* または *app1.*	このノードから下の子ノード階層。
get -m	server.applications.app1.* または *app1.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.applications.app1.ejb-module1_jar または *ejb-module1_jar または server.applications.ejb-module1_jar	bean1bean2bean3...
list -m	server.applications.app1.ejb-module1_jar または *ejb-module1_jar または server.applications.ejb-module1_jar	このノードから下の子ノード階層。
get -m	server.applications.app1.ejb-module1_jar.* または *ejb-module1_jar.* または server.applications.ejb-module1_jar.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。

表 20-34 アプリケーション-エンタープライズアプリケーションとスタンドアロンモジュール (続き)

コマンド	ドット表記名	出力
list -m	server.applications.app1. ejb-module1_jar.bean1 注: スタンドアロンモジュールでは、アプリケーション名を含むノード (この例では app1) は表示されません。	次の子ノード一覧が表示されます。 bean-poolbean-cachebean-method
list -m	server.applications.app1. ejb-module1_jar.bean1 注: スタンドアロンモジュールでは、アプリケーション名を含むノード (この例では app1) は表示されません。	子ノードの階層とこのノードとそれより下のすべての子ノードの全属性の一覧。
get -m	server.applications.app1. ejb-module1_jar.bean1.* 注: スタンドアロンモジュールでは、アプリケーション名を含むノード (この例では app1) は表示されません。	次の属性とそれらの関連付けられた値が表示されます。 CreateCount_CountCreateCount_ DescriptionCreateCount_ LastSampleTimeCreateCount_ NameCreateCount_ StartTimeCreateCount_ UnitMethodReadyCount_ CurrentMethodReadyCount_ DescriptionMethodReadyCount_ HighWaterMarkMethodReadyCount_ LastSampleTimeMethodReadyCount_ LowWaterMarkMethodReadyCount_ NameMethodReadyCount_ StartTimeMethodReadyCount_ UnitRemoveCount_CountRemoveCount_ DescriptionRemoveCount_ LastSampleTimeRemoveCount_ NameRemoveCount_StartTimeAttribute RemoveCount_Unit
list -m	server.applications.app1. ejb-module1_jar.bean1.bean-pool 注: スタンドアロンモジュールでは、アプリケーション名を含むノード (この例では app1) は表示されません。	属性は表示されず、server.applications.app1.ejb-module1_jar.bean1-cache には表示すべき情報がないことを示すメッセージが表示されます。特定の文字列で始まる有効な名前を取得するには、ワイルドカード (*) 文字を使用します。たとえば、server で始まるすべての名前を一覧表示するには、list server* を使用します。

表 20-34 アプリケーション-エンタープライズアプリケーションとスタンドアロンモジュール (続き)

コマンド	ドット表記名	出力
get -m	server.applications.app1. ejb-module1_jar.bean1.bean-pool.* 注: スタンドアロンモジュールでは、アプリケーション名を含むノード(この例では app1)は表示されません。	表 20-11 で説明した EJB プール属性に対応する属性と値の一覧。
list -m	server.applications.app1. ejb-module1_jar.bean1.bean-cache 注: スタンドアロンモジュールでは、アプリケーション名を含むノード(この例では app1)は表示されません。	属性は表示されず、代わりに「get --monitor 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 -m	server.applications.app1. ejb-module1_jar.bean1.bean-cache.* 注: スタンドアロンモジュールでは、アプリケーション名を含むノード(この例では app1)は表示されません。	表 20-12 で説明した EJB キャッシュ属性に対応する属性と値の一覧。
list -m	server.applications.app1. ejb-module1_jar.bean1.bean-method.method1 注: スタンドアロンモジュールでは、アプリケーション名を含むノード(この例では app1)は表示されません。	属性は表示されず、代わりに「get --monitor 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 -m	server.applications.app1. ejb-module1_jar.bean1.bean-method.method1.* 注: スタンドアロンモジュールでは、アプリケーション名を含むノード(この例では app1)は表示されません。	表 20-10 で説明した EJB メソッド属性に対応する属性と値の一覧。
list -m	server.applications.app1.web-module1_war	このモジュールに割り当てられた 1 つまたは複数の仮想サーバーが表示されます。
get -m	server.applications.app1.web-module1_war.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.applications.app1.web-module1_war. virtual_server	登録されているサーブレットの一覧が表示されます。
get -m	server.applications.app1.web-module1_war. virtual_server.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.applications.app1.web-module1_war. virtual_server.servlet1	属性は表示されず、代わりに「get --monitor 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。

表 20-34 アプリケーション-エンタープライズアプリケーションとスタンドアロンモジュール (続き)

コマンド	ドット表記名	出力
get -m	server.applications.app1.web-module1_war.virtual_server.servlet1.*	表 20-14 で説明した Web コンテナ (サーブレット) 属性に対応する属性と値の一覧。

次の表に、HTTP サービスレベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-35 HTTP サービスレベル

コマンド	ドット表記名	出力
list -m	server.http-service	仮想サーバーの一覧。
get -m	server.http-service.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.http-service.server	HTTP リスナーの一覧。
get -m	server.http-service.server.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.http-service.server.http-listener1	属性は表示されず、代わりに「get --monitor 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 -m	server.http-service.server.*	表 20-16 で説明した HTTP サービス属性に対応する属性と値の一覧。

次の表に、スレッドプールレベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-36 スレッドプールレベル

コマンド	ドット表記名	出力
list -m	server.thread-pools	スレッドプール名の一覧。
get -m	server.thread-pools.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.thread-pools.orb.threadpool.thread-pool-1	属性は表示されず、代わりに「get --monitor 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。

表 20-36 スレッドプールレベル (続き)

コマンド	ドット表記名	出力
get -m	server.thread-pools.orb\ .threadpool\ .thread-pool-1.*	表 20-21 で説明したスレッドプール属性に対応する属性と値の一覧。

次の表に、リソースレベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-37 リソースレベル

コマンド	ドット表記名	出力
list -m	server.resources	プール名の一覧。
get -m	server.resources.*	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server.resources.jdbc-connection-pool-pool. connection-pool1	属性は表示されず、代わりに「get --monitor 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 -m	server.resources.jdbc-connection-pool-pool. connection-pool1.*	表 20-17 で説明した接続プール属性に対応する属性と値の一覧。

次の表に、トランザクションサービスレベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-38 トランザクションサービスレベル

コマンド	ドット表記名	出力
list -m	server.transaction-service	属性は表示されず、代わりに「get --monitor 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 -m	server.transaction-service.*	表 20-22 で説明したトランザクションサービス属性に対応する属性と値の一覧。

次の表に、ORB レベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-39 ORB レベル

コマンド	ドット表記名	出力
list -m	server. orb	server- orb. connection- managers
get -m	server. orb. *	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server. orb. connection- managers	1つまたは複数の ORB 接続マネージャー名。
get -m	server. orb. connection- managers. *	このノードに属性が存在しないことを示すメッセージだけが表示されます。
list -m	server. orb. connection- managers. orb\ .Connections\ .Inbound\ .AcceptedConnections	属性は表示されず、代わりに「get --monitor 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 -m	server. orb. connection- managers. orb\ .Connections\ .Inbound\ .AcceptedConnections. *	表 20-20 で説明した ORB 接続マネージャー属性に対応する属性と値の一覧。

次の表に、JVM レベルに対するコマンド、ドット表記名、および対応する出力を示します。

表 20-40 JVM レベル

コマンド	ドット表記名	出力
list -m	server. jvm	属性は表示されず、代わりに「get --monitor 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 -m	server. jvm. *	表 20-23 で説明した JVM 属性に対応する属性と値の一覧。

JConsole の使用

ここでは、次の内容について説明します。

- 249 ページの「JConsole から Application Server への接続のセキュリティーを有効にする」
- 250 ページの「JConsole を Application Server に接続する前提条件」
- 251 ページの「JConsole を Application Server に接続する」
- 251 ページの「安全に JConsole を Application Server に接続する」

Communications Server の管理と監視は、JMX テクノロジをベースにしています。つまり、Communications Server の JVM で実行している MBeanServer では、管理対象コンポーネントは MBean で表されます。

Java SE 5 では、Platform MBean Server を含めること、および JVM を設定するための MBean を含めることにより、JVM の管理と監視を拡張します。Communications Server は、これらの拡張機能を利用して、MBean を Platform MBean Server に登録します。JMX コネクタクライアントには、JVM MBean と Communications Server MBean が統合表示されます。

すべての MBean を表示するために、Communications 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設定が用意されています。Communications 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、このコネクタサーバーを使用してサーバーに接続できます。

Java SE には、MBean Server に接続し、そこに登録されている MBean を表示するためのツールも用意されています。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。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。

Communications Server で使用できるように JConsole を設定すると、Communications Server は JMX コネクタのサーバー側となり、JConsole は JMX コネクタの優先クライアント側となります。251 ページの「JConsole を Application Server に接続する」に、正常な接続を作成する方法が示されています。

JConsole から Application Server への接続のセキュリティーを有効にする

Communications Server、つまり JMX コネクタサーバー側への接続方法は、接続のトランスポート層のセキュリティーによって若干異なります。サーバー側がセキュリティー保護されている(トランスポート層のセキュリティーが保証されている)場合、クライアント側で実行する設定があります。

- Communications Server の開発者プロファイルの場合、デフォルトでは、セキュリティ保護されていないシステム JMX コネクタサーバーが設定されます。
- Communications Server のクラスタおよびエンタープライズプロファイルの場合、デフォルトでは、セキュリティ保護されたシステム JMX コネクタサーバーが設定されます。
- 通信に使用されるプロトコルは、RMI/JRMP です。JMX コネクタのセキュリティが有効な場合、使用されるプロトコルは SSL 上の RMI/JRMP です。

注 - SSL 上の RMI では、クライアントが目的のサーバーと通信できるようにするための追加チェックは行われません。そのため、JConsole の使用時は、悪意のあるホストにユーザー名とパスワードを送信している可能性が常にあります。セキュリティが安全であるかどうかの確認は、管理者に完全に委ねられています。

開発者プロファイルドメインを `appserver.sun.com` のようなマシンにインストールすると、ドメイン管理サーバー (DAS) の `domain.xml` ファイルに次のようなエントリが含まれます。

```
<!-- The JSR 160 "system-jmx-connector" -->
<jmx-connector accept-all="false" address="0.0.0.0"
auth-realm-name="admin-realm" enabled="true" name="system" port="8686"
protocol="rmi_jrmp" security-enabled="false"/>
<!-- The JSR 160 "system-jmx-connector" -->
```

JMX コネクタの `security-enabled` フラグは `false` です。クラスタまたはエンタープライズプロファイルが稼働している場合、または開発者プロファイルの JMX コネクタのセキュリティを有効にした場合、このフラグは `true` に設定されます。

```
<!-- The JSR 160 "system-jmx-connector" -->
<jmx-connector accept-all="false" address="0.0.0.0"
auth-realm-name="admin-realm" enabled="true" name="system" port="8686"
protocol="rmi_jrmp" security-enabled="true"/>
...
</jmx-connector>
<!-- The JSR 160 "system-jmx-connector" -->
```

JConsole を Application Server に接続する前提条件

JConsole の設定は、2 つに分かれます。サーバー側とクライアント側です。この例では、Communications Server ドメインは、強力な Solaris サーバーである `appserver.sun.com` と呼ばれるマシンにインストールされます。これがサーバー側です。

クライアント側にも Communications Server のインストールがあります。ここでは、クライアント側は Windows マシンで、Java SE 6.0 と Communications Server がインストールされているものとします。

注-クライアント側で Communications Server のインストールが必要になるのは、Communications Server ドメインのリモートマシン上でセキュリティが有効な場合(クラスタおよびエンタープライズプロファイルのデフォルト)だけです。前述の Solaris マシンで Communications Server 開発者プロファイルドメインを管理する場合、このクライアントマシンに Communications Server のインストールは必要ありません。

同じマシン上にサーバー側とクライアント側がある場合、localhost を使用してホスト名を指定できます。

▼ JConsole を Application Server に接続する

この手順では、JMX コネクタでセキュリティを有効にしないで JConsole を Communications Server に接続する方法について説明します。デフォルトでは、開発者プロファイルの Communications Server のセキュリティは有効になっていません。

- 1 appserver.sun.com でドメインを起動します。
- 2 `JDK_HOME/bin/jconsole` を実行して JConsole を起動します。
- 3 JConsole の「エージェントに接続」タブで、ユーザー名、パスワード、ホスト名、およびポート(デフォルトは **8686**)を入力します。
ユーザー名は管理ユーザーの名前、パスワードはドメインの管理パスワードを参照します。
- 4 「接続」をクリックします。
JConsole ウィンドウの各種タブに、MBean、VM 情報などが表示されます。

▼ 安全に JConsole を Application Server に接続する

この手順では、JMX コネクタでセキュリティを有効にして JConsole を Communications Server に接続する方法について説明します。クラスタまたはエンタープライズプロファイルの Communications Server のセキュリティは、デフォルトで有効になっています。この手順は、開発者プロファイルの JMX コネクタでセキュリティを有効にした場合に使用してください。

- 1 クライアントマシン (JConsole がインストールされている) に **Communications Server** をインストールします。
この作業が必要になるのは、信頼するドメイン管理サーバーのサーバー証明書の場合を JConsole に対して通知するためです。この証明書を取得するには、リモートの `asadmin` コマンドを 1 回以上呼び出しますが、そのためには **Communications Server** のローカルインストールが必要です。
 - 2 `appserver.sun.com` で **Communications Server** を起動します。
これはクラスタまたはエンタープライズドメインであるため、システム JMX コネクタサーバーはセキュリティー保護されています。開発者プロファイルの JMX コネクタのセキュリティーを有効にするには、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。
 - 3 ローカル **Communications Server** インストールから `install-dir\bin\asadmin list --user admin --secure=true --host appserver.sun.com --port 4848` を実行します。**4848** はサーバーの管理ポートです。
この例では `asadmin list` コマンドを選択していますが、任意のリモート `asadmin` コマンドを実行できます。`appserver.sun.com` の DAS から送信される証明書を受け入れることを要求されます。
 - 4 `y` を押して、`appserver.sun.com` の **DAS** から送信される証明書を受け入れます。
サーバーの証明書は、クライアントマシンのホームディレクトリにある `.asadmintruststore` ファイルに格納されます。
-
- 注-サーバーマシンとクライアントマシンが同じである場合、この手順は必要ありません。つまり、JConsole も `appserver.sun.com` で稼働している場合です。
-
- 5 次の JConsole コマンドを使用して、トラストストアの場所を JConsole に通知します。
`JDK-dir\bin\jconsole.exe -J-Djavax.net.ssl.trustStore="C:\Documents and Settings\user\.asadmintruststore"`
 - 6 `JDK_HOME/bin/jconsole` を実行して **JConsole** を起動します。
 - 7 **JConsole** の「エージェントに接続」タブで、ユーザー名、パスワード、ホスト名、およびポート (デフォルトは **8686**) を入力します。
ユーザー名は管理ユーザーの名前、パスワードはドメインの管理パスワードを参照します。
 - 8 「接続」をクリックします。
JConsole ウィンドウの各種タブに、MBean、VM 情報などが表示されます。

管理ルールの設定

この章では、定期的な管理作業の自動化、実行時のさまざまな状況に対応したアプリケーションサーバーの自己調整、および障害の発生防止による可用性の向上を目的とした管理ポリシーの設定について説明します。また、カスタマイズ可能な定義済み管理ルールである自己管理テンプレートについても説明します。

ここでは、次の内容について説明します。

- 253 ページの「管理ルールについて」
- 254 ページの「管理ルールの設定」

管理ルールについて

管理ルールを設定すると、定期的な管理作業の自動化、実行時のさまざまな状況に対応したアプリケーションサーバーの自己調整、および障害の発生防止による可用性の向上を実現できます。管理ルールには、指定したイベントの発生時や設定したしきい値への到達時に行うアクションが含まれます。指定したイベントに基づいて修正アクションを自動的に実行できる管理ルールを設定できます。

管理ルールは、イベントとアクションの2つの部分から構成されます。

- イベントは、JMX 通知機構を使用して定義済みのアクションをトリガーします。
- アクションは、関連付けられたイベントが発生したときにトリガーされます。アクションは、`javax.management.NotificationListener` を実装した通知リスナーである MBean です。

たとえば、イベントが EJB ロガーによって記録された SEVERE メッセージであれば、アクションによってログメッセージの内容を含むアラートを管理者に送信できます。イベントが発生すると、イベントデータが `javax.management.Notification` の `userData` 部分の一部として渡されます。

ルールに指定したアクションは、カスタム MBean として実装する必要があります。そのため、管理ルールを設定する前に、イベント通知を受信したら適切なアク

ションを実行するように設計したカスタム MBean を配備してください。カスタム MBean の開発と配備については、『[Sun GlassFish Communications Server 2.0 Developer's Guide](#)』の第 14 章「[Developing Custom MBeans](#)」を参照してください。

Communications Server には、いくつかの便利なイベントが用意されており、通知を発行するカスタム MBean を記述することで、それらのイベントをさらに拡張できます。プロパティーを変更することにより、各イベントをさらにカスタマイズできます。

使用可能なイベントタイプは次のとおりです。

- 監視イベント: MBean の属性を監視します。監視イベントは、`javax.management.monitor` パッケージと同様の機能を持っています。監視イベントは、Java SE 5 の `javax.management.monitor` が行う単純な属性の監視に加えて、複雑な属性の監視もサポートしています。
- 通知イベント: カスタム MBean からのイベントを通知します。これらのイベントを使ってカスタムイベントを作成することにより、イベント辞書を拡張します。通知を発行できる MBeans は、すべてイベントとして指定できます。
- システムイベント:
 - ライフサイクル: サーバーの起動、シャットダウン、および終了を示すイベント。
 - ログ: 指定されたロガーがログエントリを書き込んだときにトリガーされるイベント。たとえば、EJB コンテナロガーが SEVERE ログエントリを記録したときに管理者にアラートを送信する管理ルールを作成できます。
 - タイマー: 指定された日時や間隔などでトリガーされるイベント。これらのイベントは、`javax.management.timer` パッケージと同様の機能を持っています。
 - トレース: HTTP/IIOP 要求メソッド、EJB メソッド、および Web メソッドの入口と出口でトリガーされるイベント。たとえば、サーブレットとの対話をログに記録するために使用するサーブレットフィルタを、Web メソッドの入口および出口イベントを使った管理ルールとして設計できます。
 - クラスタ: クラスタまたはインスタンスが起動、停止、または失敗したときにトリガーされるイベント。これらのイベントでは、グループ管理システムのクラスタ監視が使用されます。

管理ルールの設定

管理コンソールで管理ルールを設定するには、次の手順に従います。

- 開発者プロファイルの場合は、「設定」→「管理ルール」の順に選択します。
- クラスタおよびエンタープライズプロファイルの場合は、「設定」→「設定」→「管理ルール」の順に選択します。

注-管理ルールをグローバルに有効にするには、このページで「すべての規則」にチェックマークを付けます。管理ルールがグローバルに有効になっていない場合は、どの管理ルールも実行されません。

また、個別の管理ルールを有効にするには、このページで、有効にするルールの横にあるボックスをクリックし、「有効」をクリックします。

ルールの MBean も、ターゲットで有効にする必要があります。MBean を有効にするには、「カスタム MBean」→「MBean」の順に選択します。「カスタム MBean を編集」ページで、「ターゲット」タブをクリックして「カスタム MBean ターゲット」ページにアクセスします。このページで、一部またはすべてのターゲットで MBean を有効にできます。

詳細については、オンラインヘルプを参照してください。

コマンド行から管理ルールを作成するには、`create-management-rule` コマンドを使用します。管理ルールのプロパティを設定するには、`get` および `set` コマンドを使用します。管理ルールを表示および削除するには、`list-management-rules` および `delete-management-rule` コマンドを使用します。詳細は、これらのコマンドのオンラインヘルプまたは『[Sun GlassFish Communications Server 2.0 Reference Manual](#)』を参照してください。

Java 仮想マシンと詳細設定

Java 仮想マシン (JVM) は、コンパイル済みの Java プログラムでバイトコードを実行する、インタプリタ型の処理エンジンです。JVM 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 Communications Server には JVM が必要であり、JVM が Application Server を実行し、Application Server 上で稼働する Java アプリケーションをサポートします。JVM 設定は、アプリケーションサーバー設定の一部です。

この章では、Java 仮想マシン (JVM™) とその他の詳細設定の設定方法について説明します。次の節で構成されています。

- 257 ページの「JVM 設定の調整」
- 258 ページの「詳細設定」

JVM 設定の調整

アプリケーションサーバーを設定する一環として、Java 仮想マシンの使用を拡張する設定を定義します。管理コンソールを使用して JVM の設定を変更するには、「アプリケーションサーバー」>「JVM 設定」タブの順に選択し、次のように JVM の一般設定を定義します。

- 「Java ホーム」: Java ソフトウェアのインストールディレクトリの名前を入力します。Communications Server は Java SE ソフトウェアに依存します。

注-存在しないディレクトリ名を入力したり、サポートされないバージョンの Java EE ソフトウェアのインストールディレクトリを指定したりした場合、Communications Server は起動しません。

- 「Javac オプション」: Java プログラミング言語コンパイラのコマンド行オプションを入力します。EJB コンポーネントの配備時に、Communications Server はコンパイラを実行します。

- 「デバッグ」: JPDA (Java Platform Debugger Architecture) によるデバッグを設定するときは、「有効」チェックボックスにチェックマークを付けます。
JPDA はアプリケーション開発者によって使用されます。
- 「デバッグオプション」: デバッグを有効にしたときに JVM に渡される JPDA オプションを指定します。
- 「RMI コンパイルオプション」: `rmic` コンパイラのコマンド行オプションを入力します。EJB コンポーネントの配備時に Communications Server は `rmic` コンパイラを実行します。
- 「バイトコードプリプロセッサ」: クラス名のコンマ区切りリストを入力します。各クラスは、`com.sun.appserv.BytecodePreprocessor` インタフェースを実装する必要があります。クラスは指定の順序で呼び出されます。
プロファイラなどのツールは、「バイトコードプリプロセッサ」フィールドの入力を必要とすることがあります。プロファイラは、サーバーパフォーマンスの分析に使用される情報を生成します。

詳細設定

管理コンソールを使用して詳細なアプリケーション設定を行うには、「アプリケーションサーバー」>「詳細」タブ>「アプリケーション設定」タブの順に選択し、次のようにアプリケーション設定を行います。

- 「再読み込み」: このチェックボックスを選択して、アプリケーションの動的再読み込みを有効にします。
動的再読み込みが有効になっている場合は(デフォルトでは有効)、アプリケーションやモジュールのコードや配備記述子を変更したときにアプリケーションやモジュールを再配備する必要はありません。変更された JSP またはクラスファイルをアプリケーションまたはモジュールの配備ディレクトリにコピーするだけで十分です。サーバーは定期的に変更を確認し、変更が見つかったら、自動的にかつ動的にアプリケーションを再配備します。この機能は、変更したコードをすぐにテストできるため、開発環境で役に立ちます。しかし、本稼働環境では、動的再読み込みはパフォーマンスを低下させる可能性があります。また、再読み込みが行われているときは、その転送時のセッションが無効になります。クライアントのセッションを再起動する必要があります。
- 「再読み込みのポーリング間隔」: アプリケーションとモジュールにコードの変更がないかをチェックして動的に再読み込みする間隔を指定します。デフォルトは2秒です。
- 「管理セッションタイムアウト」: 管理セッションがタイムアウトするまでの非活動の分数を指定します。

また、配備設定を次のように定義します。

- 「自動配備」: このチェックボックスを選択して、アプリケーションの自動配備を有効にします。

自動配備では、アプリケーションやモジュールファイル (JAR、WAR、RAR、または EAR) が特別なディレクトリにコピーされ、そこで Application Server によって自動的に配備されます。

- 「自動配備のポーリング間隔」: アプリケーションとモジュールにコードの変更がないかをチェックして動的に再読み込みする間隔を指定します。デフォルトは 2 秒です。
- 「ベリファイア」: 配備記述子ファイルを検証するには、「ベリファイアを有効」ボックスにチェックマークを付けます。これは任意指定です。
- 「プリコンパイル」: JSP ファイルをプリコンパイルするには、「プリコンパイルを有効」ボックスにチェックマークを付けます。

ドメインまたはノードエージェントの自動再起動

マシンの再起動が必要になった場合など、ドメインまたはノードエージェントが予想外に停止される場合にそなえて、ドメインまたはノードエージェントが自動的に再起動されるようにシステムを設定することが可能です。

この付録では、次の項目について説明します。

- 261 ページの「Solaris 10 での自動再起動」
- 263 ページの「Solaris 9 および Linux プラットフォーム上での `inittab` による自動再起動」
- 264 ページの「Microsoft Windows プラットフォーム上での自動再起動」
- 266 ページの「自動再起動時のセキュリティ」

Solaris 10 での自動再起動

Solaris 10 ユーザーは、`asadmin create-service` コマンドを使用して、ノードエージェントまたはドメイン管理サーバー (DAS) を再起動するサービスを作成できます。作成したサービスでは、Solaris サービス管理機能 (SMF) が使用されます。

サービスが再起動するプロセスは、そのサービスが DAS またはノードエージェントのどちらを再起動するかによって異なります。

- DAS を再起動するサービスのプロセスは、`asadmin start-domain` です。
- ノードエージェントを再起動するサービスのプロセスは、`asadmin start-node-agent` です。

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。`asadmin create-service` コマンドを使用して 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 OS 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 OS 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。

`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。このコマンドの詳細は、[create-service\(1\)](#) のマニュアルページを参照してください。

構文は次のとおりです。

```
asadmin create-service [--name service-name] [--type das|node-agent]
--passwordfile password-file [--serviceproperties serviceproperties]
domain-or-node-agent-configuration-directory
```

たとえば、`domain1` に対して `domain1` という名前のサービスを作成するには、次の手順に従います。

1. 次のコマンドを実行します。

```
asadmin create-service --type das --passwordfile password.txt
/appserver/domains/domain1
```

これで、ドメイン `domain1` を自動的に再起動するサービスが作成されます。このコマンドにより、バックグラウンドで、テンプレートからマニフェストファイルが作成されて検証され、そのファイルがサービスとしてインポートされます。

注 - 特定の Communications 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。ユーザーの特権を調べるには、そのユーザーとしてログインし、`ppriv -l` コマンドを入力します。

2. サービスが作成されたら、次の `svacdm enable` コマンドを使用してサービスを有効にします。

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

3. 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。

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
- usermod

これらのコマンドの詳細については、各コマンドのマニュアルページを参照してください。

Solaris 9 および Linux プラットフォーム上での inittab による自動再起動

Solaris 9 または Linux プラットフォーム上でドメインを再起動するには、`/etc/inittab` ファイルにテキストを 1 行追加します。

`/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 `asadmin` コマンドを呼び出す行を追加します。

たとえば、`opt/SUNWappserver` ディレクトリにインストールされた Communications Server の `domain1` を、`password.txt` という名前のパスワードファイルを使って再起動するには、次のテキストを追加します。

```
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。

ノードエージェントを再起動する場合の構文も、これと似ています。たとえば、`opt/SUNWappserver` ディレクトリにインストールされた Communications Server の `agent1` を、`password.txt` という名前のパスワードファイルを使って再起動するには、次のテキストを追加します。

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-node-agent --user admin
--passwordfile /opt/SUNWappserver/password.txt agent1
```

Microsoft Windows プラットフォーム上での自動再起動

Microsoft Windows 上で自動的に再起動するには、Windows サービスを作成し、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されないようにします。

Windows サービスの作成

Sun GlassFish Communications Server に同梱されている実行可能ファイル `appservService.exe` と `appserverAgentService.exe` を、Microsoft が提供するサービス制御コマンド (`sc.exe`) と組み合わせて使用します。

`sc.exe` コマンドは Windows XP に含まれており、Windows インストールディレクトリの `system32` サブディレクトリ (通常は `C:\windows\system32` か `C:\winnt\system32` のいずれか) に格納されています。現時点では、Windows 2000 の `sc.exe` は <ftp://ftp.microsoft.com/reskit/win2000/sc.zip> でダウンロードできます。`sc.exe` の使用方法については、http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmslite.asp を参照してください。

`appservService.exe` と `appservAgentService.exe` の使用方法は次のとおりです。

```
C:\winnt\system32\sc.exe create service-name binPath= \"fully-qualified-path-to-appservService.exe
\"fully-qualified-path-to-asadmin.bat start-command\"
\"fully-qualified-path-to-asadmin.bat stop-command\"
start= auto DisplayName= \"display-name\"
```

注 - `binpath` と等号 (=) の間にスペースは入れません。等号とパスの間にはスペースが必要です。

たとえば、パスワードファイル `C:\Sun\AppServer\password.txt` を使ってドメイン `domain1` を開始および停止するサービス `SunJavaSystemAppServer DOMAIN1` を作成するには、次のようにします。

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin --passwordfile C:\Sun\AppServer\password.txt domain1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\" start= auto
DisplayName= \"SunJavaSystemAppServer DOMAIN1\"
```

ノードエージェント `agent1` を開始および停止するサービスを作成するには、次のようにします。

```
C:\windows\system32\sc.exe create agent1 binPath=
"C:\Sun\AppServer\lib\appservAgentService.exe \"C:\Sun\AppServer\bin\asadmin.bat
```

```
start-node-agent --user admin --passwordfile C:\Sun\AppServer\password.txt agent1\
"C:\Sun\AppServer\bin\asadmin.bat stop-node-agent agent1\" start= auto
DisplayName= "SunJavaSystemAppServer AGENT1"
```

注 - `binPath`= パラメータの一部として入力された開始コマンドと停止コマンドは、正しい構文で記述されている必要があります。確認するには、それらのコマンドをコマンドプロンプトから実行します。コマンドを実行してもドメインまたはノードエージェントが正常に開始または停止しない場合、そのサービスは正しく動作していません。

注 - `asadmin` の `start/stop` コマンドとサービスの開始/停止を混在させないでください。両者を混在させると、サーバーの状態の同期が取れなくなります。たとえば、サーバーのコンポーネントが実行されていないのに「コンポーネントが開始された」と表示されたりします。こうした状況を避けるには、サービス使用時には常に、`sc.exe` コマンドを使ってコンポーネントを開始および停止するようにしてください。

`sc.exe create` コマンドでサービスが正しく作成されなかった場合は、そのサービスを削除し、もう一度実行してみてください。サービスを削除するには、`sc.exe delete "service-name"` コマンドを使用します。

ユーザーの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 VM は、オペレーティングシステムのシャットダウンまたはユーザーの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、ユーザーが Windows からログアウトすると Communications 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。

`-Xrs Java VM オプション`を設定するには、`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>
```

Communications Server サービスが稼働している場合、変更を有効にするには、そのサービスを停止して再起動します。

注 - Windows 2003 Server インストールでは、`-Xrs` オプションを `domain.xml` ファイル追加しても、サービスのシャットダウンを防止できないことがあります。この場合は、次のように、このオプションを `as-install\lib\processLauncher.xml` ファイルに追加します。

```
<process name="as-service-name">
  ...
  <sysproperty key="-Xrs"/>
  ...
```

自動再起動時のセキュリティー

クラスタプロファイルまたはエンタープライズプロファイルを使用している場合は、Communications Server を自動的に再起動するときに、管理パスワードとマスターパスワードが必要です。開発者プロファイルを使用している場合は、これらのパスワードは必要ありません。

クラスタプロファイルおよびエンタープライズプロファイルに必要なパスワードとマスターパスワードは、次のいずれかの方法で処理します。

- Microsoft Windows 上で、ユーザーにパスワードを尋ねるようにサービスを設定します。

1. サービスコントロールパネルで、作成したサービスをダブルクリックします。
2. 「プロパティ」ウィンドウの「ログオン」タブをクリックします。
3. 「デスクトップとの対話をサービスに許可」にチェックマークを付け、必要なパスワードに対するプロンプトがコンポーネント起動時に表示されるようにします。

ログインしてプロンプトを表示させ、入力時にエントリがエコーバックされないことを確認する必要があります。これがサービスオプションを使用する際のもっとも安全な方法ですが、この方法の場合、ユーザーが関与しないとサービスが利用可能になりません。

デスクトップとの対話オプションを設定しなかった場合、サービスは「開始保留」状態のままになり、ハンガアップしたように見えます。この状態から抜け出すには、このサービスのプロセスを終了してください。

- Windows または UNIX 上で、`--savemasterpassword=true` オプションを使ってドメインを作成し、管理パスワード格納用のパスワードファイルを作成します。コンポーネント起動時に、`--passwordfile` オプションを使ってパスワードが格納されたファイルを指定します。

次に例を示します。

1. ドメイン作成時にマスターパスワードを保存します。次の構文では、ユーザーは管理パスワードとマスターパスワードの入力を求められます。

```
asadmin create-domain --adminport 4848 --adminuser admin
--savemasterpassword=true --instanceport 8080 domain1
```

- Windows の場合は、サービスを作成します。その際、パスワードファイルを使って管理パスワードを提供します。

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin
--passwordfile C:\Sun\AppServer\password.txt domain1\"
"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\" start= auto
DisplayName= "SunJavaSystemAppServer DOMAIN1"
```

パスワードファイル password.txt のパスは、C:\Sun\AppServer\password.txt です。このファイルには、パスワードが次の形式で格納されています。

```
AS_ADMIN_password=password
```

たとえば、パスワードが adminadmin の場合、次のようになります。

```
AS_ADMIN_password=adminadmin
```

- UNIX の場合、inittab ファイルに追加する行の中で、--passwordfile オプションを使用します。

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

パスワードファイル password.txt のパスは、/opt/SUNWappserver/password.txt です。このファイルには、パスワードが次の形式で格納されています。

```
AS_ADMIN_password=password
```

たとえば、パスワードが adminadmin の場合、次のようになります。

```
AS_ADMIN_password=adminadmin
```


domain.xml のドット表記名属性

この付録では、Mbean とその属性を指定するために使用可能なドット表記名属性について説明します。domain.xml ファイル内のすべての要素は対応する MBean を持ちます。これらの名前は、「個々の名前をピリオドで区切る」という構文規則に従うため、「ドット表記名」と呼ばれます。

この付録の内容は次のとおりです。

- [269 ページの「トップレベル要素」](#)
- [271 ページの「別名を使用しない要素」](#)

トップレベル要素

domain.xml ファイル内のすべてのトップレベル要素で、次の条件が満たされている必要があります。

- サーバー、設定、クラスタ、またはノードエージェントの名前はそれぞれ一意である必要があります。
- サーバー、設定、クラスタ、またはノードエージェントに「domain」という名前を付けることはできません。
- サーバーインスタンスに「agent」という名前を付けることはできません。

次の表に、トップレベル要素と対応するドット表記名プレフィックスを示します。

要素名	ドット表記名プレフィックス
applications	domain.applications
resources	domain.resources
configurations	domain.configs

要素名	ドット表記名プレフィックス
servers	domain.servers この要素に含まれるすべてのサーバーは、 <i>server-name</i> としてアクセス可能です。ここで、 <i>server-name</i> は、server サブ要素の name 属性の値です。
clusters	domain.clusters この要素に含まれるすべてのクラスタは、 <i>cluster-name</i> としてアクセス可能です。ここで、 <i>cluster-name</i> は、cluster サブ要素の name 属性の値です。
node-agents	domain.node-agents
lb-configs	domain.lb-configs Sun GlassFish Enterprise Server のみに適用可能
system-property	domain.system-property
converged-lb-configs	domain.converged-lb-configs Sun GlassFish Communications Server のみに適用可能

次の2つのレベルの別名が利用可能です。

1. 1つ目のレベルの別名を使えば、プレフィックス `domain.servers` または `domain.clusters` を使わずにサーバーインスタンスまたはクラスタの属性にアクセスできます。したがって、たとえば、`server1` という形式のドット表記名は、ドット表記名 `domain.servers.server1` (`server1` は特定のサーバーインスタンス) にマッピングされます。
2. 2つ目のレベルの別名を使えば、特定のクラスタまたはスタンドアロンサーバーインスタンス (ターゲット) の設定、アプリケーション、およびリソースを参照できます。

次の表に、サーバー名またはクラスタ名で始まるドット表記名と、それらの別名であるドメイン配下のトップレベル名を示します。

ドット表記名	別名	コメント
<code>target.applications.*</code>	<code>domain.applications.*</code>	この別名の解決結果は、 <i>target</i> のみによって参照されるアプリケーションになります。
<code>target.resources.*</code>	<code>domain.resources.*</code>	この別名の解決結果は、 <i>target</i> によって参照されるすべての <code>dbc-connection-pool</code> 、 <code>connector-connection-pool</code> 、 <code>resource-adapter-config</code> 、およびその他のすべてのリソースになります。

次の表に、サーバー名またはクラスタ名で始まるドット表記名と、それらの別名である、そのサーバーまたはクラスタによって参照されている設定内のトップレベル名を示します。すべてのリストを表示するには、`asadmin list target.*` コマンドを使用します。

ドット表記名	別名
<code>target.http-service</code>	<code>config-name.http-service</code>
<code>target.iiop-service</code>	<code>config-name.iiop-service</code>
<code>target.admin-service</code>	<code>config-name.admin-service</code>
<code>target.web-container</code>	<code>config-name.web-container</code>
<code>target.sip-container</code>	<code>config-name.sip-container</code> Sun GlassFish Communications Server のみに適用可能
<code>target.ejb-container</code>	<code>config-name.ejb-container</code>
<code>target.mdb-container</code>	<code>config-name.mdb-container</code>
<code>target.jms-service</code>	<code>config-name.jms-service</code>
<code>target.sip-service</code>	<code>config-name.sip-service</code> Sun GlassFish Communications Server のみに適用可能
<code>target.log-service</code>	<code>config-name.log-service</code>
<code>target.security-service</code>	<code>config-name.security-service</code>
<code>target.transaction-service</code>	<code>config-name.transaction-service</code>
<code>target.monitoring-service</code>	<code>config-name.monitoring-service</code>
<code>target.java-config</code>	<code>config-name.java-config</code>
<code>target.availability-service</code>	<code>config-name.availability-service</code>
<code>target.thread-pools</code>	<code>config-name.thread-pools</code>

別名を使用しない要素

クラスタ化されたインスタンスでは、別名を使用すべきではありません。クラスタ化されたインスタンスの特定のシステムプロパティを取得する際のドット表記名属性は、`domain.servers.clustered-instance-name.system-property` のように記述します。`clustered-instance-name.system-property` ではありません。

asadmin ユーティリティー

Application Server には、asadmin という名前のコマンド行管理ユーティリティーが含まれています。asadmin ユーティリティーは、Application Server の起動と停止のほか、ユーザー、リソース、およびアプリケーションの管理にも使用されます。

この章で説明する内容は次のとおりです。

- 276 ページの「リモートコマンドの共通オプション」
- 278 ページの「multimode コマンド」
- 278 ページの「get、set、list コマンド」
- 280 ページの「サーバーのライフサイクルコマンド」
- 281 ページの「リストおよびステータスコマンド」
- 282 ページの「配備コマンド」
- 283 ページの「バージョンコマンド」
- 283 ページの「Message Queue 管理コマンド」
- 284 ページの「リソース管理コマンド」
- 286 ページの「設定コマンド」
- 291 ページの「ユーザー管理コマンド」
- 291 ページの「ルールおよび監視コマンド」
- 292 ページの「データベースコマンド」
- 292 ページの「診断およびロギングコマンド」
- 293 ページの「Web サービスコマンド」
- 293 ページの「セキュリティーサービスコマンド」
- 295 ページの「パスワードコマンド」
- 295 ページの「検証コマンド」
- 296 ページの「カスタム MBean コマンド」
- 296 ページの「サービスコマンド」
- 296 ページの「プロパティーコマンド」

asadmin ユーティリティ

asadmin ユーティリティを使用すると、Application Server の管理タスクを実行できます。この asadmin ユーティリティは、管理者インタフェースの代わりに使用できます。

asadmin ユーティリティは、ユーザーの実行する操作やタスクを特定するサブコマンドを呼び出します。サブコマンドは大文字と小文字を区別します。短形式のオプションの引数にはダッシュ (-) が、長形式のオプションの引数には二重ダッシュ (--) が付いています。オプションによって、ユーティリティによるサブコマンドの実行方法を制御します。オプションでも大文字と小文字を区別します。機能のオン/オフを切り替えるブール型のオプションを除いて、大部分のオプションには引数値が必要です。オペランドは引数値の後ろに表示され、空白、タブ、または二重ダッシュ (--) で区切られます。asadmin ユーティリティでは、オプションとその値の後ろに続くものをオペランドとして処理します。

asadmin は、コマンドシェルを呼び出して、またはマルチコマンドモード (multimode コマンドと呼ばれる) で使用できます。コマンドシェルの呼び出しを使用する場合は、コマンドシェルから asadmin ユーティリティを呼び出します。asadmin がコマンドを実行し、終了します。マルチコマンドモードでは、一度 asadmin を呼び出すと、asadmin が終了するまで複数のコマンドが受け入れられます。終了後は通常のコマンドシェルの呼び出しに戻ります。マルチコマンドモードで設定した環境変数は、multimode を終了するまで、あとに続くすべてのコマンドに使用されます。また、あらかじめ準備したコマンドの一覧をファイルまたは標準入力から渡す (パイプする) ことによって、コマンドを提供することもできます。また、マルチモードセッション内から multimode を呼び出すこともできます。2 つ目のマルチモード環境を終了すると、元のマルチモード環境に戻ります。

また、asadmin ユーティリティは、対話型オプションまたは非対話型オプションで実行できます。デフォルトでは、対話型オプションが有効になっています。対話型オプションでは、必要な引数を入力するように求められます。対話型オプションは、すべての環境のコマンドシェルの呼び出しで使用することができます。コマンドプロンプトから一度に 1 つのサブコマンドを実行したり、ファイルから multimode で実行したりするときは、multimode で対話型オプションを使用できます。multimode のサブコマンド (入力ストリームからパイプされた場合) や、別のプログラムから呼び出されたサブコマンドは、対話型オプションでは実行できません。

ローカルサブコマンドは、管理サーバーが存在しなくても実行できます。ただし、サブコマンドを実行したり、インストールディレクトリやドメインディレクトリに対するアクセス権を得たりするには、ドメインをホストしているマシンにユーザーがログインする必要があります。リモートサブコマンドの実行は、管理サーバーに接続してサブコマンドを実行することによって常に行われます。稼働中の管理サーバーが必要です。すべてのリモートサブコマンドで、次のオプションが必須になります。

- `-u --user` 認証されたドメインアプリケーションサーバーの管理ユーザー名。

- `--passwordfile` ドメインアプリケーションサーバーのパスワードを格納したファイル。形式は `AS_ADMIN_PASSWORD=password` です。 `password` には、実際の管理者パスワードを指定します。
- `-H --host` ドメインアプリケーションサーバーが稼働しているマシン名。
- `-p --port` 管理要求を待機しているドメインアプリケーションサーバーのポート番号。デフォルトのポート番号は 4848 です。
- `-s --secure true` の場合、SSL/TLS を使用してドメインアプリケーションサーバーと通信します。
- `-t --terse` 出力データを簡潔にすることを示します。通常、人間が読みやすい文を避けて、スクリプトで使用するために整形されたデータを優先します。デフォルトは `false` です。
- `-e --echo true` に設定すると、コマンド行の文が標準出力にエコーされます。デフォルトは `false` です。
- `-I --interactive true` (デフォルト) に設定した場合、必須パスワードオプションのみの入力が必要されます。
- `-h --help` コマンドに関するヘルプテキストが表示されます。

ローカルまたはリモートで実行できるサブコマンドの場合、環境内またはコマンド行のどちらかで、`--host`、`--port`、`--user`、および `--passwordfile` オプションのいずれか1つが設定されていれば、そのサブコマンドはリモートモードで実行されます。さらに、ローカルまたはリモートで実行できるサブコマンドの場合、`--local` オプションが `true` に設定されていると、そのサブコマンドはローカルで実行されます。また、コマンド行または環境内で、ローカルオプションが何も設定されていない場合でも、デフォルトでサブコマンドはローカルで実行されます。`--local` オプションが `true` に設定されていると、ローカルの `--host`、`--port`、`--user`、および `--passwordfile` の設定が指定されている場合でも、それらより優先されます。サブコマンドはローカルモードで実行されます。

ローカルで実行できるサブコマンドでは、対象のドメインを指定する `--domain` オプションを使用でき、ドメインが1つだけの場合は、そのドメインがデフォルトドメインとみなされます。複数のドメインが存在する場合、`--domain` オプションは必須オプションになります。ローカルまたはリモートで実行できるサブコマンドの場合、リモートで `--host`、`--port`、`--user`、および `--passwordfile` オプションを指定して実行すると、`--domain` オプションは無視されます。サブコマンドがリモートモードで実行される場合、`--domain` オプションは無視されます。ドメインごとに1つの管理インスタンスがあるため、1つのマシンに複数のドメインがある場合、ローカルでの実行するときはドメインを指定し、リモートで実行するときはそのドメインの管理インスタンスの `--host`、`--port`、`--user`、および `--passwordfile` オプションを指定してください。

セキュリティのため、コマンド行でパスワードを入力する代わりに、ファイルからサブコマンドのパスワードを設定することができます。--passwordfile オプションを使用すると、パスワードを格納したファイルを取得できます。このファイルの有効な内容は次のとおりです。

例 C-1 パスワードファイルの内容

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

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、--password オプションの使用に関する警告が表示されます。この警告が表示されないようにするには、**AS_ADMIN_PASSWORD** の設定を取り消します。マスターパスワードは、コマンド行または環境変数では伝達されませんが、passwordfile で指定できます。

--secure オプションを使用するには、set コマンドを使用して、domain.xml 内の admin http-listener でセキュリティの --enabled フラグを有効にします。asadmin のサブコマンドを使用して作成や削除を行った場合、新しく作成したコマンドを有効にするには、サーバーを再起動する必要があります。サーバーを再起動するには、start-domain コマンドを使用します。

Solaris プラットフォーム上の Application Server のコマンド行インタフェースサブコマンドのマニュアルページにアクセスするには、MANPATH 環境変数に \$AS_INSTALL/man を追加します。

asadmin ユーティリティサブコマンドの全体的な使用法の情報は、--help オプションを呼び出すことで取得できます。サブコマンドを指定すると、そのサブコマンドの使用法が表示されます。サブコマンドを指定せずに --help オプションを実行すると、使用可能なすべてのサブコマンドの一覧が表示されます。

リモートコマンドの共通オプション

すべてのリモートコマンドで、次の共通オプションが必須になります。

表 C-1 リモートコマンドの必須オプション

オプション	定義
--host	ドメイン管理サーバーの稼働しているマシン名。デフォルト値は、localhost です。

表C-1 リモートコマンドの必須オプション (続き)

オプション	定義
--port	管理用のHTTP/Sポート。これは、ドメインを管理するためにブラウザで指定するポートです。たとえば、http://localhost:4848などです。デフォルトのポート番号は4848です。
--user	認証されたドメイン管理サーバーの管理ユーザー名。asadmin login コマンドを使用してドメインに対して認証を行った場合、その後の操作では、この特定のドメインに対して--user オプションを指定する必要はありません。
--passwordfile	<p>--passwordfile オプションは、特定の形式でパスワードエントリを格納しているファイルの名前を指定します。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。</p> <p>たとえば、ドメイン管理サーバーのパスワードを指定するには、次の形式のエントリを使用します。AS_ADMIN_PASSWORD=<i>password</i> (<i>password</i> は実際の管理者パスワード)その他の指定できるパスワードには、AS_ADMIN_MAPPEDPASSWORD、AS_ADMIN_USERPASSWORD、AS_ADMIN_ALIASPASSWORD などがあります。</p> <p>すべてのリモートコマンドでは、--passwordfile または asadmin login を使用するか、コマンドプロンプトによる対話形式で、ドメイン管理サーバーに対して認証を行うための管理パスワードを指定する必要があります。asadmin login コマンドを使用するのは、管理パスワードを指定するときだけです。リモートコマンド用に指定する必要があるその他のパスワードについては、--passwordfile を使用するか、コマンドプロンプトで入力します。</p> <p>asadmin login コマンドを使用してドメインに対して認証を行った場合、その後の操作では、この特定のドメインに対して--passwordfile オプションを使用して管理パスワードを指定する必要はありません。ただし、これはAS_ADMIN_PASSWORD オプションにしか適用されません。なお、個別のコマンド(update-file-user など)が要求する場合は、その他のパスワード(AS_ADMIN_USERPASSWORD など)を指定する必要があります。</p> <p>セキュリティ上の理由により、環境変数として指定されたパスワードは、asadmin によって読み取られません。</p>
--secure	true に設定した場合、SSL/TLS を使用してドメイン管理サーバーと通信します。
--interactive	true (デフォルト) に設定した場合、必須パスワードオプションのみの入力が必要されます。
--terse	出力データを簡潔にすることを示します。通常、人間が読みやすい文を避けて、スクリプトで使用するために整形されたデータを優先します。デフォルトはfalse です。
--echo	true に設定すると、コマンド行の文が標準出力にエコーされます。デフォルトはfalse です。

表 C-1 リモートコマンドの必須オプション (続き)

オプション	定義
--help	コマンドに関するヘルプテキストが表示されます。

multimode コマンド

multimode コマンドを使用すると、asadmin コマンドを処理できます。コマンド行インタフェースによってコマンドの入力が求められます。入力されたコマンドが実行され、コマンドの結果が表示されたあと、次のコマンドの入力が求められます。さらに、このモードで設定されたすべての asadmin オプション名は、後続のすべてのコマンドで使用されます。exit または quit を入力して multimode を終了するまで、環境を設定したり、コマンドを実行することができます。また、あらかじめ準備したコマンドの一覧をファイルまたは標準入力から渡す(パイプする)ことによって、コマンドを提供することもできます。multimode セッション内から multimode を呼び出すことができます。2つ目の multimode 環境を終了すると、元の multimode 環境に戻ります。

get、set、list コマンド

asadmin get、set、および list コマンドは、Application Server の抽象階層に対するナビゲーションメカニズムを提供するために、連携して動作します。階層には、configuration と monitoring の 2つがあり、これらのコマンドはこの両方に対して機能します。list コマンドでは、読み取り専用または変更可能な属性を持つ管理コンポーネントの完全修飾のドット表記名で表示されます。

configuration 階層は、変更可能な属性を提供します。一方、monitoring 階層にある管理コンポーネントの属性は純粋に読み取り専用です。configuration 階層は、大まかにドメインのスキーマドキュメントに基づいていますが、monitoring 階層は少し異なっています。

list コマンドを使用すると、必要な階層内の特定の管理コンポーネントに到達できます。次に、get および set コマンドを呼び出すと、すぐに管理コンポーネントの属性の名前と値を取得したり、値を設定することができます。ワイルドカード(*)オプションを使用すると、指定した完全修飾のドット表記名の中から、一致するものをすべて取得できます。ナビゲーション可能な階層および管理コンポーネントの詳細な説明については、例を参照してください。

アプリケーションサーバーのドット表記名では、名前全体を複数部分に分けるための区切り文字として「.」(ピリオド)を使用します。これは、Unix ファイルシステムで、ファイルの絶対パス名のレベルを「/」を使用して区切る方法と同じです。get、set、および list コマンドによって受け入れられるドット表記名を形成する場合、次の規則が適用されます。特定のコマンドには追加のセマンティクスが適用されることに留意してください。

- . (ピリオド) は常に、名前を連続した2つの部分に区切ります。
- 名前の1つの部分は、通常、アプリケーションサーバーのサブシステムまたはその固有のインスタンス、あるいはその両方を特定します。次に例を示します。
web-container、log-service、thread-pool-1 など。
- 名前の一部に . (ピリオド) が含まれている場合は、. の前に \ (バックスラッシュ) を付けて、区切り文字として機能しないようにする必要があります。
- * (アスタリスク) は、ドット表記名の任意の場所で使用できます。これは正規表現におけるワイルドカード文字のような役割を果たします。また、* によって、ドット表記名のすべての部分を折りたたむことができます。「<classname>this.is.really.long.hierarchy </classname>」のような長形式のドット表記名を「<classname>th*.hierarchy</classname>」に短縮することができます。ただし、. は常に名前の区切りに使われることに注意してください。
- ドット表記名の最上位のスイッチは --monitor または -m であり、所定のコマンド行で個別に指定されます。このスイッチが存在するかしないかによって、アプリケーションサーバー管理の2つの階層 (monitoring と configuration) のどちらを選択するのかが示されます。
- ワイルドカード文字をまったく含まない完全なドット表記名を使用する場合は、list および get/set では、セマンティクスが少し異なります。
 - list コマンドは、この完全なドット表記名を、抽象階層内の親ノードの完全な名前として処理します。この名前を list コマンドに与えると、そのレベルの直接の子ノードの名前が単に返されます。たとえば、list server.applications.web-module では、ドメインまたはデフォルトのサーバーに配備されたすべての Web モジュールが一覧表示されます。
 - get および set コマンドは、この完全なドット表記名を、ノードの属性の完全修飾名 (ノードのドット表記名そのものが、このドット表記名の最後の部分を削除したときに取得する名前となる) として処理し、その属性の値を取得または設定します。これはこのような属性が存在する場合です。したがって、最初からこれを実行することはできません。まず、階層内の特定のノードの属性名を見つけるために、ワイルドカード文字の * を使用する必要があります。たとえば、server.applications.web-module.JSPWiki.context-root* では、ドメインまたはデフォルトサーバーに配備された Web アプリケーションのコンテキストルートが返されます。

list コマンドは、これら3つのコマンドのナビゲーション機能では、必ず最初に来るものです。特定のアプリケーションサーバーのサブシステムの属性を set (設定) または get (取得) する場合は、そのドット表記名を知っておく必要があります。list コマンドを使用すると、サブシステムのドット表記名を見つけることができます。たとえば、/ で始まる大規模なファイルシステム内の特定のファイルの変更日 (属性) を検索する場合を考えます。最初に、そのファイルのファイルシステム内での場所を検索し、その属性を確認する必要があります。したがって、appserver の階層を理解するための最初の2つのコマンドは、* list "*" と <command>* list * --monitor になります。これらのコマンドのソートされた出力を確認するには、get、set、または list コマンドのマニュアルページを参照してください。

サーバーのライフサイクルコマンド

サーバーのライフサイクルコマンドとは、ドメインまたはインスタンスを、作成、削除、起動、または停止するコマンドのことです。

表C-2 サーバーのライフサイクルコマンド

コマンド	定義
<code>create-domain</code>	ドメインの設定を作成します。ドメインとは管理用の名前空間のことです。どのドメインにも設定があり、その設定は一連のファイルに格納されます。アプリケーションサーバーの所定のインストールでは、任意の数のドメインを作成できます。それぞれのドメインには個別の管理アイデンティティが与えられます。ドメインは、1つずつ独立して存在しています。所定のシステムの <code>asadmin</code> スクリプト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設定を格納することができます。デフォルトでは、ドメイン設定は <code>install_dir/domains</code> 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、別の場所に設定を格納することもできます。
<code>delete-domain</code>	指定したドメインを削除します。ドメインはすでに存在して、停止している必要があります。
<code>start-domain</code>	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 <code>install_dir/domains</code> 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、 <code>domain_name</code> オペランドを指定する必要があります。
<code>stop-domain</code>	指定したドメインのドメイン管理サーバーを停止します。
<code>restore-domain</code>	ドメイン下のファイルをバックアップディレクトリから復元します。
<code>list-domains</code>	ドメインを一覧表示します。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 <code>install_dir/domains</code> ディレクトリにあるドメインが表示されます。複数のドメインが存在する場合、 <code>domain_name</code> オペランドを指定する必要があります。
<code>backup-domain</code>	指定したドメイン下のファイルをバックアップします。

表C-2 サーバーのライフサイクルコマンド (続き)

コマンド	定義
login	ユーザーをドメインにログインさせます。(ローカルの) 各種マシン上でさまざまなアプリケーションサーバードメインが作成されている場合、これらの中の任意のマシンから <code>asadmin</code> を呼び出すことによって、任意の場所にあるドメインを(リモートで)管理することができます。この機能は、特定のマシンが管理クライアントとして選択されており、そのマシンが複数のドメインやサーバーを管理しているような場合に特に役立ちます。任意の場所にあるドメインを管理するために使用される <code>asadmin</code> コマンドは、リモートコマンドと呼ばれます。 <code>asadmin login</code> コマンドを使用すると、このようなリモートドメインの管理が簡単になります。 <code>login</code> コマンドは対話型モードのみで実行されます。ここでは、管理ユーザー名とパスワードの入力が求められます。正常にログインしたら、ユーザーのホームディレクトリにファイル <code>.asadminpass</code> が作成されます。これは、 <code>--saveLogin</code> オプションの使用時に <code>create-domain</code> コマンドによって変更されるファイルと同じものです。このコマンドを実行するには、ドメインが実行されている必要があります。
create-instance	ローカルまたはリモートマシン上に新しいサーバーインスタンスを作成します。
delete-instance	サーバーインスタンスを削除します。このコマンドは、リモートまたはローカルで実行できます。ユーザーの認証には、管理サーバー用に指定されたパスワードを使用します。また、削除するインスタンスは、管理サーバーが処理するドメイン内にすでに存在していなければなりません。削除操作は取り消せないため、このコマンドは慎重に使用してください。

リストおよびステータスコマンド

リストおよびステータスコマンドは、配備されたコンポーネントのステータスを表示します。

表C-3 リストおよびステータスコマンド

コマンド	定義
show-component-status	配備されたコンポーネントのステータスを取得します。ステータスは、サーバーから返された文字列で表現されます。ステータスを表す文字列は、 <code>app-name</code> のステータスは <code>enabled</code> である、または <code>app-name</code> のステータスは <code>disabled</code> である、と表現されます。
list-components	配備されたすべての Java EE 5 コンポーネントを一覧表示します。 <code>--type</code> オプションが指定されていない場合は、すべてのコンポーネントが表示されます。

表C-3 リストおよびステータスコマンド (続き)

list-sub-components	配備されたモジュール内か、配備されたアプリケーションのモジュール内にあるEJBまたはサーブレットを一覧表示します。モジュールが指定されていない場合は、すべてのモジュールが表示されます。
enable	指定したコンポーネントを有効にします。コンポーネントがすでに有効になっている場合は、再有効化されます。有効にするには、コンポーネントが配備済みである必要があります。コンポーネントが配備済みでない場合は、エラーメッセージが返されます。
disable	指定したコンポーネントを即座に無効にします。コンポーネントが配備済みである必要があります。コンポーネントが配備済みでない場合は、エラーメッセージが返されます。
export	後続のコマンド環境に対して、自動エクスポートの変数名にマークを付けます。指定した変数名の値を設定解除するか、マルチモードを終了しないかぎり、後続のコマンドはすべてその変数名の値を使用します。
get	属性の名前と値を取得します。
set	1つ以上の設定可能な属性の値を設定します。
list	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コマンドを実行する場合は、引用符が必要です。
unset	マルチモード環境に対して設定した1つ以上の変数を削除します。変数と変数に関連付けられた値は、その環境内に存在しなくなります。

配備コマンド

配備コマンドは、アプリケーションを配備したり、クライアントスタブを取得したりします。

表C-4 配備コマンド

コマンド	定義
deploy	エンタープライズアプリケーション、Webアプリケーション、EJBモジュール、コネクタモジュール、またはアプリケーションクライアントモジュールを配備します。コンポーネントがすでに配備済みであるか、すでに存在している場合、--force オプションがtrueに設定されていれば、強制的に再配備されます。
deploydir	アプリケーションを配備ディレクトリから直接配備します。配備ディレクトリには、Java EE仕様に準拠する適切なディレクトリ階層と配備記述子が存在していなければなりません。

表 C-4 配備コマンド (続き)

コマンド	定義
get-client-stubs	AppClient スタンドアロンモジュールまたは AppClient モジュールを含むアプリケーション用のクライアントスタブ JAR ファイルを、サーバーマシンからローカルディレクトリに取得します。このコマンドを実行する前に、アプリケーションまたはモジュールを配備済みにしてください。

バージョンコマンド

バージョンコマンドを使用すると、バージョン文字列を返したり、すべての `asadmin` コマンドを一覧表示したり、ライセンスファイルをインストールしたりできます。

表 C-5 バージョンコマンド

コマンド	定義
version	バージョン情報を表示します。このコマンドによって、特定のユーザー/パスワード、およびホスト/ポートを使用して管理サーバーと通信できない場合は、ローカルでバージョンを取得し、警告メッセージを表示します。
help	すべての <code>asadmin</code> コーティリティーコマンドの一覧を表示します。コマンドを指定すると、そのコマンドの使用方法が表示されます。
install-license	Application Server の不正な使用を防止します。このコマンドを使用すると、ライセンスファイルをインストールできます。
shutdown	管理サーバーと実行中のすべてのインスタンスをシャットダウンします。再起動するには、管理サーバーを手動で起動させる必要があります。

Message Queue 管理コマンド

Message Queue 管理コマンドを使用すると、JMS 送信先を管理できます。

表 C-6 Message Queue コマンド

コマンド	定義
create-jmsdest	JMS 物理送信先を作成します。物理送信先とともに、 <code>create-jms-resource</code> コマンドを使用して、物理送信先を指定する Name プロパティを持つ JMS 送信先リソースを作成します。
delete-jmsdest	指定した JMS 送信先を削除します。

表 C-6 Message Queue コマンド (続き)

コマンド	定義
flush-jmsdest	指定したターゲットの JMS サービス設定の物理送信先から、メッセージをパージします。
list-jmsdest	JMS 物理送信先を一覧表示します。
jms-ping	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。JMS サービスは、デフォルトでは Application Server の起動時に起動します。また、このコマンドは JMS サービス内のデフォルトの JMS ホストのみを ping します。組み込まれている JMS サービスに ping できない場合には、エラーメッセージが表示されません。

リソース管理コマンド

リソースコマンドを使用すると、アプリケーション内で使用されているさまざまなリソースを管理できます。

表 C-7 リソース管理コマンド

コマンド	定義
create-jdbc-connection-pool	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。
delete-jdbc-connection-pool	JDBC 接続プールを削除します。削除する JDBC 接続プールは、オペランドによって特定されます。
list-jdbc-connection-pools	作成済みの JDBC 接続プールを取得します。
create-jdbc-resource	JDBC リソースを新規作成します。
delete-jdbc-resource	指定した JNDI 名の JDBC リソースを削除します。
list-jdbc-resources	作成済みの JDBC リソースの一覧を表示します。
create-jms-resource	Java Message Service (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。
delete-jms-resource	指定した JMS リソースを削除します。
list-jms-resources	既存の JMS リソース (送信先および接続ファクトリリソース) を一覧表示します。
create-jndi-resource	JNDI リソースを登録します。
delete-jndi-resource	指定した JNDI 名の JNDI リソースを削除します。
list-jndi-resources	既存のすべての JNDI リソースを特定します。

コマンド	定義
<code>list-jndi-entries</code>	JNDI ツリーを表示して照会します。
<code>create-javamail-resource</code>	JavaMail セッションリソースを作成します。
<code>delete-javamail-resource</code>	指定した JavaMail セッションリソースを削除します。
<code>list-javamail-resources</code>	既存の JavaMail セッションリソースを一覧表示します。
<code>create-persistence-resource</code>	持続性リソースを登録します。
<code>delete-persistence-resource</code>	持続性リソースを削除します。持続性リソースを削除すると、 <code>create-persistence-resource</code> コマンドを使用して作成された JDBC リソースも一緒に削除されます。
<code>list-persistence-resources</code>	すべての持続性リソースを表示します。
<code>create-custom-resource</code>	カスタムリソースを作成します。カスタムリソースは、 <code>javax.naming.spi.ObjectFactory</code> 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。
<code>delete-custom-resource</code>	カスタムリソースを削除します。
<code>list-custom-resources</code>	カスタムリソースを一覧表示します。
<code>create-connector-connection-pool</code>	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。
<code>delete-connector-connection-pool</code>	オペランド <code>connector_connection_pool_name</code> を使用して指定したコネクタ接続プールを削除します。
<code>list-connector-connection-pools</code>	作成済みのコネクタ接続プールを一覧表示します。
<code>create-connector-resource</code>	指定した JNDI 名でコネクタリソースを登録します。
<code>delete-connector-resource</code>	指定した JNDI 名のコネクタリソースを削除します。
<code>list-connector-resources</code>	すべてのコネクタリソースを取得します。
<code>create-admin-object</code>	指定した JNDI 名の管理対象オブジェクトを作成します。
<code>delete-admin-object</code>	指定した JNDI 名の管理対象オブジェクトを削除します。
<code>list-admin-objects</code>	すべての管理対象オブジェクトを一覧表示します。
<code>create-resource-adapter-config</code>	コネクタモジュールの設定情報を作成します。
<code>delete-resource-adapter-config</code>	<code>domain.xml</code> に作成されたコネクタモジュールの設定情報を削除します。
<code>list-resource-adapter-configs</code>	<code>domain.xml</code> 内のコネクタモジュールの設定情報を一覧表示します。

表 C-7 リソース管理コマンド (続き)

コマンド	定義
add-resources	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。 <i>xml_file_path</i> は、作成するリソースを格納する XML ファイルへのパスです。DOCTYPE は、resources.xml ファイル内で <i>install_dir/lib/dtds/sun-resources_1_2.dtd</i> と指定してください。
ping-connection-pool	JDBC 接続プールとコネクタ接続プールの両方に対して、接続プールが使用可能かどうかをテストします。たとえば、あとで配備する予定のアプリケーション用に JDBC 接続プールを新規作成した場合、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 プールをテストします。接続プールに ping する前に、認証された接続プールを作成し、エンタープライズサーバーまたはデータベースが起動していることを確認する必要があります。

設定コマンド

設定コマンドを使用すると、IIOP リスナー、ライフサイクルモジュール、HTTP および HTTP リスナー、プロファイラ、およびその他のサブシステムを構築できます。

ここでは、次の内容について説明します。

- 286 ページの「SIP リスナーコマンド」
- 287 ページの「HTTP および IIOP リスナーコマンド」
- 287 ページの「ライフサイクルおよび監査モジュールコマンド」
- 288 ページの「プロファイラおよび SSL コマンド」
- 288 ページの「JVM オプションおよび仮想サーバーコマンド」
- 289 ページの「スレッドプールおよび認証レルムコマンド」
- 290 ページの「トランザクションおよびタイマーコマンド」

SIP リスナーコマンド

SIP リスナーコマンドにより、リスナーを管理できます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-8 SIP リスナーコマンド

コマンド	定義
create-sip-listener	新しい SIP リスナーソケットを追加します。

表 C-8 SIP リスナーコマンド (続き)

コマンド	定義
<code>delete-sip-listener</code>	指定した SIP リスナーを削除します。
<code>list-sip-listeners</code>	既存の SIP リスナーを一覧表示します。

HTTP および IIOP リスナーコマンド

HTTP および IIOP リスナーコマンドを使用して、リスナーを管理することができます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-9 IIOP リスナーコマンド

コマンド	定義
<code>create-http-listener</code>	新しい HTTP 待機ソケットを追加します。
<code>delete-http-listener</code>	指定した HTTP リスナーを削除します。
<code>list-http-listeners</code>	既存の HTTP リスナーを一覧表示します。
<code>create-iiop-listener</code>	IIOP リスナーを作成します。
<code>delete-iiop-listener</code>	指定した IIOP リスナーを削除します。
<code>list-iiop-listeners</code>	既存の IIOP リスナーを一覧表示します。

ライフサイクルおよび監査モジュールコマンド

ライフサイクルおよび監査モジュールコマンドを使用すると、ライフサイクルモジュールや、監査機能を実装するオプションのプラグインモジュールを制御できるようになります。これらのコマンドは、リモートモードのみでサポートされています。

表 C-10 ライフサイクルモジュールコマンド

コマンド	定義
<code>create-lifecycle-module</code>	ライフサイクルモジュールを作成します。ライフサイクルモジュールによって、アプリケーションサーバー環境内で短期または長期の Java ベースのタスクを実行する手段が提供されます。
<code>delete-lifecycle-module</code>	指定したライフサイクルモジュールを削除します。
<code>list-lifecycle-modules</code>	既存のライフサイクルモジュールを一覧表示します。

表 C-10 ライフサイクルモジュールコマンド (続き)

コマンド	定義
<code>create-audit-module</code>	監査機能を実装するプラグインモジュール用に、指定した監査モジュールを追加します。
<code>delete-audit-module</code>	指定した監査モジュールを削除します。
<code>list-audit-modules</code>	すべての監査モジュールを一覧表示します。

プロファイラおよび SSL コマンド

プロファイラおよび SSL コマンドを使用すると、プロファイラおよび SSL クライアント設定を管理できます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-11 プロファイラおよび SSL コマンド

コマンド	定義
<code>create-profiler</code>	プロファイラ要素を作成します。サーバーインスタンスは、Java 設定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。プロファイラの変更時には、サーバーを再起動する必要があります。
<code>delete-profiler</code>	指定したプロファイラ要素を削除します。サーバーインスタンスは、Java 設定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。プロファイラの変更時には、サーバーを再起動する必要があります。
<code>create-ssl</code>	選択した SIP リスナー、HTTP リスナー、IIOP リスナー、または IIOP サービス内で 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。
<code>delete-ssl</code>	選択した SIP リスナー、HTTP リスナー、IIOP リスナー、または IIOP サービス内の SSL 要素を削除します。

JVM オプションおよび仮想サーバーコマンド

JVM オプションおよび仮想サーバーコマンドを使用すると、次のような要素を制御できます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-12 JVM オプションおよび仮想サーバーコマンド

コマンド	定義
create-jvm-option	Java 設定または 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。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実行に必要な設定を記録するために使用されます。新しく作成した JVM オプションを有効にするには、サーバーを再起動する必要があります。
delete-jvm-option	Java 設定または domain.xml ファイルのプロファイラ要素から、JVM オプションを削除します。
create-virtual-server	指定した仮想サーバーを作成します。Application Server で仮想化を行うことで、複数のホストアドレス上で待機している 1 つの HTTP サーバープロセスによって、複数の URL ドメインを処理できるようになります。アプリケーションを 2 つの仮想サーバーで使用できる場合は、同じ物理リソースプールを共有します。
delete-virtual-server	指定した仮想サーバー ID の仮想サーバーを削除します。

スレッドプールおよび認証レルムコマンド

スレッドプールおよび認証レルムコマンドを使用すると、次のような要素を制御できます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-13 スレッドプールおよび認証レルムコマンド

コマンド	定義
create-threadpool	指定した名前付きのスレッドプールを作成します。プール内のスレッドの最大数および最小数、作業キューの数、およびスレッドのアイドルタイムアウトを指定できます。作成したスレッドプールは、IIOP 要求やリソースアダプタの作業管理要求のサービスに使用できます。作成したスレッドプールは、複数のリソースアダプタで使用できます。
delete-threadpool	指定した ID のスレッドプールを削除します。
list-threadpools	すべてのスレッドプールを一覧表示します。
create-auth-realm	名前付き認証レルムを追加します。
delete-auth-realm	名前付き認証レルムを削除します。

トランザクションおよびタイマーコマンド

トランザクションおよびタイマーコマンドを使用すると、トランザクションおよびタイマーサブシステムを制御できます。これによって、実行中のトランザクションを中断できるようになります。これらのコマンドは、リモートモードのみでサポートされています。

表C-14 レジストリコマンド

コマンド	定義
<code>freeze-transaction</code>	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。このコマンドは、実行中のトランザクションをロールバックする前に呼び出します。すでに凍結しているトランザクションサブシステムに対してこのコマンドを呼び出しても、効果はありません。
<code>unfreeze-transaction</code>	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。
<code>recover-transactions</code>	保留中のトランザクションを手動で回復します。
<code>rollback-transaction</code>	指定したトランザクションをロールバックします。
<code>unpublish-from-registry</code>	
<code>list-timers</code>	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。

レジストリコマンド

レジストリコマンドを使用すると、Web サービスのアーティファクトを発行または発行解除できます。

表C-15 レジストリコマンド

コマンド	定義
<code>publish-to-registry</code>	レジストリに Web サービスのアーティファクトを発行します。
<code>unpublish-from-registry</code>	レジストリから Web サービスのアーティファクトの発行を解除します。
<code>list-registry-locations</code>	

ユーザー管理コマンド

ユーザー管理コマンドは、ファイルレلم認証によってサポートされているユーザーを管理します。これらのコマンドは、リモートモードのみでサポートされています。

表 C-16 ユーザー管理コマンド

コマンド	定義
<code>create-file-user</code>	指定したユーザー名、パスワード、およびグループで、キーファイル内にエントリを作成します。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。
<code>delete-file-user</code>	指定したユーザー名のエントリをキーファイル内から削除します。
<code>update-file-user</code>	指定した <code>user_name</code> 、 <code>user_password</code> 、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。コロン(:)で区切ることによって、複数のグループを入力することもできます。
<code>list-file-users</code>	ファイルレلم認証によってサポートされているファイルユーザーの一覧を作成します。
<code>list-file-groups</code>	ファイルレلم認証によってサポートされているファイルユーザーおよびグループを管理します。このコマンドでは、ファイルユーザー内の使用可能なグループが表示されます。

ルールおよび監視コマンド

ルールおよび監視コマンドを使用すると、規則を管理し、サーバーを監視できます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-17 ルールおよび監視コマンド

コマンド	定義
<code>create-management-rule</code>	アプリケーションサーバーインストールや配備済みのアプリケーションをインテリジェントに自己管理するために、新しい管理規則を作成します。
<code>delete-management-rule</code>	指定した管理規則を削除します。
<code>create-transformation-rule</code>	Web サービス操作に適用できる XSLT 変換規則を作成します。この規則は、要求または応答に適用できます。
<code>delete-transformation-rule</code>	指定した Web サービスの XSLT 変換規則を削除します。

表 C-17 ルールおよび監視コマンド (続き)

コマンド	定義
start-callflow-monitoring	Web コンテナ、EJB コンテナ、および JDBC からデータを収集して相互に関連付け、要求の完全な呼び出しフロー/パスを提示します。callflow-monitoring がオンの場合のみ、データは収集されません。
stop-callflow-monitoring	要求の呼び出しフロー情報の収集を無効にします。

データベースコマンド

データベースコマンドを使用すると、Java DB データベース (Apache Derby に基づく) を起動および停止することができます。これらのコマンドは、ローカルモードのみでサポートされています。

表 C-18 データベースコマンド

コマンド	定義
start-database	Application Server で使用可能な Java DB サーバーを起動します。このコマンドは、Application 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。
stop-database	Java DB サーバーのプロセスを停止します。Java DB サーバーは Application Server で使用できます。

診断およびロギングコマンド

診断およびロギングコマンドは、アプリケーションサーバーによる問題のトラブルシューティングに役立ちます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-19 診断およびロギングコマンド

コマンド	定義
generate-diagnostic-report	生成される HTML レポートには、アプリケーションサーバーインスタンスの設定詳細、ロギング詳細、またはプロセス固有の情報などの、アプリケーションサーバーのインストールの詳細情報へのポインタまたはナビゲーションリンクが含まれます。
display-error-statistics	前回のサーバーの再起動以降の server.log 内の重要なメッセージや警告を要約して一覧表示します。

表 C-19 診断およびロギングコマンド (続き)

コマンド	定義
<code>display-error-distribution</code>	モジュールレベルでインスタンスの <code>server.log</code> から配布されたエラーを表示します。
<code>display-log-records</code>	指定のタイムスタンプでの所定のモジュールに関するすべてのエラーメッセージを表示します。

Web サービスコマンド

Web サービスコマンドを使用すると、配備された Web サービスを監視し、変換規則を管理することができます。

表 C-20 Web サービスコマンド

コマンド	定義
<code>configure-webservice-management</code>	配備された Web サービスの監視属性または <code>maxhistory</code> 属性を設定します。
<code>create-transformation-rule</code>	Web サービス操作に適用できる XSLT 変換規則を作成します。この規則は、要求または応答に適用できます。
<code>delete-transformation-rule</code>	指定した Web サービスの XSLT 変換規則を削除します。
<code>list-transformation-rules</code>	指定した Web サービスのすべての変換規則を、適用された順に一覧表示します。
<code>publish-to-registry</code>	レジストリに Web サービスのアーティファクトを発行します。
<code>unpublish-from-registry</code>	レジストリから Web サービスのアーティファクトの発行を解除します。
<code>list-registry-locations</code>	設定済みの Web サービスレジストリのアクセスポイントの一覧を表示します。

セキュリティサービスコマンド

次のセキュリティコマンドを使用して、コネクタ接続プールのセキュリティマッピングを制御します。これらのコマンドは、リモートモードのみでサポートされています。

表C-21 セキュリティーコマンド

コマンド	定義
<code>create-connector-security-map</code>	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作成します。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。また、コンテナ管理のトランザクションベースのシナリオでは、このコマンドを使用して、アプリケーションの呼び出し側アイデンティティー (主体またはユーザーグループ) を適切なエンタープライズ情報システム (EIS) の主体にマップします。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。コネクタセキュリティーマップの設定では、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。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。
<code>delete-connector-security-map</code>	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削除します。
<code>update-connector-security-map</code>	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。
<code>list-connector-security-map</code>	指定したコネクタ接続プールに属するセキュリティーマップを一覧表示します。
<code>create-message-security-provider</code>	管理者は、特定のメッセージ層 (Application Server のパラメータおよびプロパティーを指定するファイル <code>domain.xml</code> の <code>message-security-config</code> 要素) の <code>provider-config</code> サブ要素を作成できます。
<code>delete-message-security-provider</code>	管理者は、特定のメッセージ層 (Application Server のパラメータおよびプロパティーを指定するファイル <code>domain.xml</code> の <code>message-security-config</code> 要素) の <code>provider-config</code> サブ要素を削除できます。
<code>list-message-security-providers</code>	管理者は、特定のメッセージ層 (<code>domain.xml</code> の <code>message-security-config</code> 要素) のすべてのセキュリティーメッセージプロバイダ (<code>provider-config</code> サブ要素) を一覧表示できます。

パスワードコマンド

パスワードコマンドを使用すると、パスワードを管理して、アプリケーションサーバーのセキュリティーを確保することができます。

表C-22 パスワードコマンド

コマンド	定義
create-password-alias	パスワードのエイリアスを作成し、これを domain.xml に格納します。エイリアスは、\${ALIAS=password-alias-password} という形式のトークンです。エイリアス名に対応するパスワードは、暗号化形式で格納されます。このコマンドでは、セキュリティー保護された対話型形式(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送される)の両方の形式が使用できます。
delete-password-alias	パスワードのエイリアスを削除します。
update-password-alias	名前付きターゲットにあるパスワードのエイリアスIDを更新します。
list-password-aliases	すべてのパスワードのエイリアスを一覧表示します。
change-admin-password	このリモートコマンドは、管理パスワードを変更します。このコマンドは対話型で、ユーザーは元の管理パスワードと新しい管理パスワードの両方の入力を求められます(確認入力も必要です)。
change-master-password	このローカルコマンドを使用して、マスターパスワードを変更します。このコマンドは対話型で、ユーザーは元の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求められます。サーバーが停止していないかぎり、このコマンドは機能しません。

検証コマンド

XML 検証コマンドは、domain.xml ファイルの内容を検証します。

表C-23 検証コマンド

コマンド	定義
verify-domain-xml	domain.xml ファイルの内容を検証します。

カスタム MBean コマンド

MBean コマンドを使用すると、カスタム MBean を管理および登録できます。これらのコマンドは、リモートモードのみでサポートされています。

表 C-24 カスタム MBean コマンド

コマンド	定義
create-mbean	カスタム MBean を作成および登録します。ターゲットの MBeanServer が実行されていない場合は、MBean は登録されません。
delete-mbean	カスタム MBean を削除します。ターゲットの MBeanServer が実行されていることを確認します。
list-mbeans	指定したターゲットのカスタム MBean を一覧表示します。

サービスコマンド

サービスコマンドを使用すると、ドメイン管理サーバー (DAS) の起動を設定できます。

表 C-25 サービスコマンド

コマンド	定義
create-service	無人の自動起動によって DAS が起動されるように設定します。このコマンドは、Solaris 10 では Service Management Facility (SMF) を使用します。これはローカルコマンドで、スーパーユーザー権限のある OS レベルのユーザーとして実行する必要があります。Solaris 10 のみで使用できます。サービスを作成するとき、ユーザーはサービスの起動、有効化、無効化、削除、または停止を行う必要があります。DAS は、スーパーユーザーがアクセス権を持つフォルダに格納する必要があります。設定をネットワークファイルシステムに格納することはできません。サービスは、DAS の設定の存在するフォルダを所有する OS レベルのユーザーによって制御されるように作成されません。このコマンドを実行するには、 <code>solaris.smf.*</code> の承認が必要です。

プロパティコマンド

共有サーバーインスタンスでは、参照される設定に定義された属性の上書きが頻繁に必要になります。サーバーインスタンスの任意の設定属性を、対応する名前のシステムプロパティによって上書きできます。システムプロパティコマンドを使用して、これらの共有サーバーインスタンスを管理します。

表C-26 プロパティコマンド

コマンド	定義
<code>create-system-property</code>	ドメイン、設定、またはサーバーインスタンスのシステムプロパティを一度に1つずつ作成します。
<code>delete-system-property</code>	ドメイン、設定、またはサーバーインスタンスのシステムプロパティを1つずつ削除します。
<code>list-system-properties</code>	ドメイン、設定、またはサーバーインスタンスのシステムプロパティを表示します。

索引

A

ACC

「コンテナ」を参照
アプリケーションクライアント, 109
asadminユーティリティ, 29

B

bean-cache, 属性名の監視, 218-219

C

cache-hits, 218
cache-misses, 218
CloudScape Type 4 JDBC ドライバ, 71
CORBA, 189
スレッド, 191
create-domain コマンド, 38

D

delete-domain コマンド, 38

E

Enterprise Java Beans, スレッド, 191
Enterprise JavaBeans
エンティティ, 112
キャッシュ, 112

Enterprise JavaBeans (続き)

セッション, 112
メッセージ駆動型, 112
活性化, 112
作成, 112
持続, 112
承認, 112
非活性化, 112
executiontime, 217

G

get コマンド, 監視データ, 237

H

HTTP サービス
HTTP リスナー, 164-167
キーアライブサブシステム, 166
仮想サーバー, 163-164
要求処理スレッド, 165
HTTP リスナー
アクセプタスレッド, 165
デフォルトの仮想サーバー, 165
概要, 164-167

I

IBM DB2 JDBC ドライバ, 63, 65
IIOP リスナー, 190

Inet MSSQL JDBC ドライバ, 68
Inet Oracle JDBC ドライバ, 67
Inet Sybase JDBC ドライバ, 68
Informix Type 4 JDBC ドライバ, 70-71

J

Java Business Integration (JBI), 「JBI 環境」を参照, 47
Java DB JDBC ドライバ, 62-63
Java ネーミングおよびディレクトリサービス, 「JNDI」を参照, 112
JavaServer Pages, 111
JCE プロバイダ
設定, 146
JDBC
サポートされるドライバ, 61
ドライバ, 156
JMS
リソースアダプタ, 汎用, 77-86
外部プロバイダ, 77-86
JMS リソース
キュー, 73-74
トピック, 73-74
概要, 73-74
接続ファクトリリソース, 73-74
送信先リソース, 73-74
物理的接続先, 73-74
jmsmaxmessagesload, 218
jmsra システムリソースアダプタ, 74-75
JNDI, 112
カスタムリソース, 使用, 93
外部リポジトリ, 93
検索と関連する参照, 93
名前, 91
JSP, 「JavaServer Pages」を参照, 111

K

keystore.jks ファイル, 131

L

list-domains コマンド, 39
list コマンド, 監視, 236

M

MSSQL Inet JDBC ドライバ, 68
MSSQL/SQL Server2000 Data Direct JDBC ドライバ, 64
MySQL Type 4 JDBC ドライバ, 非 XA, 66

N

numbeansinpool, 217
numexpiredsessionsremoved, 218
numpassivationerrors, 218
numpassivations, 218
numpassivationsuccess, 218
numthreadswaiting, 217

O

Object Request Broker, スレッド, 191
Oracle Data Direct JDBC ドライバ, 63-64
Oracle Inet JDBC ドライバ, 67
Oracle OCI JDBC ドライバ, 70
Oracle Thin Type 4 JDBC ドライバ, 69-70
Oracle Thin Type 4 ドライバ, 回避方法, 157
oracle-xa-recovery-workaround プロパティ, 157
ORB, 189
IIOP リスナー, 190
「Object Request Broker」を参照, 191
サービス, 監視, 224
概要, 190
ORB (Object Request Broker), 189
概要, 190

R

RSA 暗号化, 146

S

start-domain コマンド, 39
stop-domain コマンド, 40
Sybase Data Direct JDBC ドライバ, 64-65
Sybase Inet JDBC ドライバ, 68
Sybase JConnect Type 4 JDBC ドライバ, 65-66

T

total-beans-created, 218
totalbeansdestroyed, 218
totalnumerrors, 217
totalnumsuccess, 217
truststore.jks ファイル, 131

ア

アクセプタスレッド, HTTP リスナー, 165
アプレット, 109

カ

カスタムリソース, 使用, 93

キ

キーブアライブサブシステム, HTTP サービス, 166
キーポイント間隔, 160
キーポイント処理, 160
キュー, JMS, 73-74

コ

コネクタ, モジュール, 191
コネクタリソース, JMS リソース, 74-75
コネクタ接続プール, JMS リソース, 74-75
コンテナ
Enterprise JavaBeans, 109, 112
Web, 109, 111

コンテナ (続き)

アプリケーションクライアント, 109
アプレット, 109
サーブレット
Web, 111
「コンテナ」を参照, 111

サ

サーバーの再起動, 40
サーバーログ, 表示, 199-201
サービスエンジン, 47
サーブレット, 111

ス

スレッド, 「スレッドプール」を参照, 191
スレッドプール, 191
スレッド不足, 191
パフォーマンス, 191

デ

データベース
JNDI 名, 91
サポートされる, 61
リソース参照, 92

ト

トピック, JMS, 73-74
トランザクション, 155
コミット, 156
タイムアウト, 159
マネージャー, 156
ロールバック, 155
ログ, 159-160
回復, 156, 158-159
完了, 156
関連付け, 156
境界, 156

トランザクション (続き)

属性, 156

分散, 156

トランザクションサービス, 監視, 225

トランザクションマネージャー

「トランザクション」を参照

マネージャー, 156

ド

ドメイン, 作成, 38

ネ

ネーミング, JNDI とリソース参照, 92

バ

バインディングコンポーネント, 概要, 48

パ

パフォーマンス, スレッドプール, 191

ポ

ポートリスナー, 36

マ

マニュアルページ, 29

リ

リソースアダプタ, 156

jmsra, 74-75

リソースアダプタ, 汎用, JMS, 77-86

リソースマネージャー, 156

リソース参照, 92

レ

レルム, certificate, 127

ロ

ロールバック

「トランザクション」を参照

ロールバック, 155

ロギング

ロガー名前空間, 196-197

概要, 195-196

ログ

サーバーログの表示, 199-201

トランザクション, 159-160

レベルの設定, 198

一般設定の設定, 197-198

ログレコード, 195-196

ログレベル, 設定, 198

仮

仮想サーバー, 概要, 163-164

外

外部プロバイダ, JMS, 77-86

外部リポジトリ, アクセス, 93

監

監視

bean-cache 属性, 218-219

get コマンドの使用, 237

list コマンドの使用, 236

ORB サービス, 224

コンテナサブシステム, 205

トランザクションサービス, 225

管

管理コンソール, 29

高

高可用性, 33

接

接続ファクトリ, JMS, 概要, 73-74

接続先, JMS, 概要, 73-74

要

要求処理スレッド, HTTP サービス, 165

