



Sun GlassFish Enterprise Server 2.1 高可用性 (HA) 管理ガイド



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

Part No: 820-7434
2009 年 1 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ないものとします。

目次

はじめに	15
1 Enterprise Server の高可用性	21
高可用性の概要	21
HTTP ロードバランサプラグイン	22
高可用性 JMS (Java Message Service)	22
RMI-IIOP 負荷分散とフェイルオーバー	23
詳細情報	24
Enterprise Server による高可用性の提供方法	24
セッション状態データ用ストレージ	25
高可用性クラスタ	26
障害からの回復	28
Sun Cluster の使用	28
手動での復旧	28
Netbackup の使用	31
ドメイン管理サーバーの再作成	32
2 Enterprise Server でのクラスタの設定	35
クラスタの概要	35
グループ管理サービス	36
▼ クラスタに対して GMS を有効または無効にする	36
GMS の設定	37
クラスタの操作	38
▼ クラスタを作成する	39
▼ クラスタのサーバーインスタンスを作成するには	40
▼ クラスタを設定する	41
▼ クラスタ化されたインスタンスを起動、停止、および削除する	42

▼ クラスタ内のサーバーインスタンスを設定するには	42
▼ クラスタ用のアプリケーションを設定する	43
▼ クラスタ用のリソースを設定する	44
▼ クラスタを削除する	44
▼ EJB タイマーを移行する	45
▼ サービスを停止せずにコンポーネントをアップグレードする	46
3 ノードエージェントの設定	49
ノードエージェントとは	49
ノードエージェントの障害発生後のサーバーインスタンスの動作	51
ノードエージェントの配備	51
▼ ノードエージェントをオンラインで配備する	51
▼ ノードエージェントをオフラインで配備する	52
ノードエージェントとドメイン管理サーバーとの同期化	54
ノードエージェントの同期化	54
サーバーインスタンスの同期化	55
ライブラリファイルの同期化	56
固有の設定と設定管理	57
大きなアプリケーションの同期化	58
ノードエージェントログの表示	59
ノードエージェントの操作	59
ノードエージェントタスクの実行方法	59
ノードエージェントのプレースホルダ	60
▼ ノードエージェントのプレースホルダを作成する	61
ノードエージェントの作成	61
ノードエージェントの起動	64
ノードエージェントの停止	64
ノードエージェントの削除	65
▼ ノードエージェントの一般情報を表示する	65
▼ ノードエージェントの設定を削除する	66
▼ ノードエージェントの設定を編集する	67
▼ ノードエージェントのレルムを編集する	67
▼ ノードエージェントの JMX 対応リスナーを編集するには	68
▼ スタンドアロンのサーバーインスタンスを作成する	69

4 HTTP 負荷分散の設定	71
HTTP ロードバランサの動作	72
HTTP 負荷分散アルゴリズム	72
HTTP ロードバランサプラグインの機能	73
HTTP 負荷分散の設定	74
HTTP 負荷分散を設定するための前提条件	74
HTTP 負荷分散を設定するための手順	75
HTTP ロードバランサの配備	78
HTTP ロードバランサの設定	79
DAS 上での HTTP ロードバランサの設定	79
HTTP ロードバランサ参照の作成	81
HTTP 負荷分散のためのサーバーインスタンスの有効化	81
HTTP 負荷分散のためのアプリケーションの有効化	82
HTTP 健全性検査の作成	82
HTTP ロードバランサ設定ファイルのエクスポート	84
HTTP ロードバランサ設定の変更	85
動的再設定を有効にする	86
サーバーインスタンスまたはクラスタの無効化(休止)	87
アプリケーションの無効化(休止)	87
HTTP および HTTPS のフェイルオーバーの設定	88
HTTP ロードバランサによるリダイレクトの使用	90
べき等 URL の設定	93
複数の Web サーバーインスタンスの設定	94
▼ 複数の Web サーバーインスタンスを設定する	94
可用性を低下させないアプリケーションのアップグレード	95
アプリケーションの互換性	95
単一クラスタでのアップグレード	96
複数のクラスタでのアップグレード	98
互換性のないアプリケーションのアップグレード	99
HTTP ロードバランサプラグインの監視	102
ログメッセージの設定	102
ログメッセージのタイプ	102
HTTP ロードバランサのログの有効化	104
メッセージの監視について	105

5 HTTP 負荷分散のための Web サーバーの設定	107
Sun Web サーバー の設定	108
▼ Sun Web サーバー を設定する	108
自動適用を使用するための Sun Java System Web Server の設定	109
▼ Sun Web サーバー 6.1 用に SSL モードで HTTP ロードバランサを設定する	110
▼ Sun Web サーバー 6.1 用の DAS 証明書をエクスポートおよびインポートする	112
▼ Sun Web サーバー 7 用に SSL モードで HTTP ロードバランサを設定する	113
▼ Sun Web サーバー 7 用の DAS 証明書をエクスポートおよびインポートする	114
Apache Web Server の使用	116
Apache Web Server を使用するための要件	116
HTTP ロードバランサプラグインをインストールする前の Apache の設定	117
DAS 証明書のエクスポート	120
インストーラによる Apache Web Server 設定に対する変更	120
HTTP ロードバランサプラグインをインストールしたあとの Apache の設定	123
Solaris および Linux 上での Apache の起動	126
設定の確認	126
Microsoft IIS の使用	127
▼ HTTP ロードバランサプラグインを使用するように Microsoft IIS を設定する	127
自動的に設定される sun-passthrough プロパティ	129
6 設定の管理	131
設定の使用	131
設定	131
default-config 設定	132
インスタンスまたはクラスタの作成時に作成された設定	132
一意のポート番号と設定	133
名前付き設定に関連した作業	134
▼ 名前付き設定を作成する	134
名前付き設定のプロパティの編集	135
▼ 設定を参照するインスタンスのポート番号を編集する	136
▼ 名前付き設定のターゲットを表示する	137
▼ 名前付き設定を削除する	137
7 高可用性 (HA) セッション持続性とフェイルオーバーの設定	139
セッション持続性とフェイルオーバーの概要	139

要件	139
制限事項	140
高可用性セッション持続性の設定	141
▼ 高可用性セッション持続性を設定する	142
セッション可用性の有効化	143
HTTPセッションフェイルオーバー	145
Web コンテナの可用性の設定	145
個々の Web アプリケーションの可用性の設定	148
セッションフェイルオーバーでのシングルサインオンの使用	149
ステートフルセッション Bean のフェイルオーバー	150
EJB コンテナの可用性の設定	151
個々のアプリケーションまたはEJB モジュールの可用性の設定	153
個々の Bean の可用性の設定	153
チェックポイントを設定するメソッドの指定	154
8 Java Message Service 負荷分散とフェイルオーバー	157
Java Message Service の概要	157
詳細情報	158
Java Message Service の設定	158
Java Message Service の統合	159
JMS ホストリスト	160
接続プールとフェイルオーバー	161
負荷分散されたメッセージのインフロー	162
JMS サービスの高可用性	163
MQ クラスタと Enterprise Server の併用	164
高可用性 MQ クラスタ	164
ローカルモードでの高可用性ブローカクラスタの設定	164
リモートモードでの高可用性ブローカクラスタの設定	165
非 HA クラスタの自動クラスタ化	166
▼ MQ クラスタ と Enterprise Server クラスタの併用を有効にする	167
9 RMI-IIOP 負荷分散とフェイルオーバー	171
概要	171
要件	172
アルゴリズム	172

RMI-IIOP 負荷分散とフェイルオーバーの設定	173
▼アプリケーションクライアントコンテナ用に RMI-IIOP 負荷分散を設定する ...	173
10 高可用性 (HA) データベースのインストールと設定	177
HADB の設定の準備	177
前提条件と制限	178
ネットワーク冗長性の設定	178
共有メモリーとセマフォの設定	181
システムクロックの同期	184
インストール	185
HADB のインストール	185
ノードスーパーバイザープロセスの権限	186
高可用性の設定	187
▼高可用性に対応するシステムを用意する	187
HADB 管理エージェントの起動	188
高可用性のためのクラスタの設定	188
高可用性のためのアプリケーションの設定	189
クラスタの再起動	189
Web Server の再起動	189
▼ロードバランサとして機能している Web Server インスタンスをクリーンアップする	190
HADB のアップグレード	190
▼HADB をより新しいバージョンにアップグレードする	190
HADB パッケージの登録	191
HADB パッケージの登録の解除	192
管理エージェントの起動スクリプトの置き換え	193
▼HADB アップグレードの確認	193
11 高可用性データベースの管理	195
HADB 管理エージェントの使用法	196
管理エージェントの起動	196
管理エージェントコマンドの構文	201
管理エージェント設定のカスタマイズ	202
hadbm 管理コマンドの使用法	204
コマンド構文	205

セキュリティオプション	205
一般的なオプション	207
環境変数	208
HADB の設定	209
管理ドメインの作成	209
データベースの作成	210
設定属性の表示と変更	216
JDBC 接続プールの設定	222
HADB の管理	225
ドメインの管理	225
ノードの管理	226
データベースの管理	229
データセッション破損からの回復	233
HADB の拡張	234
既存ノードへの記憶スペースの追加	235
マシンの追加	235
ノードの追加	236
データベースの再断片化	238
データベースの再作成によるノードの追加	239
HADB の監視	240
HADB の状態の取得	241
デバイス情報の取得	243
ランタイムリソース情報の取得	245
HADB マシンの管理	248
▼単一のマシンに対して保守を実行する	248
▼すべての HADB マシンに対して計画的な保守を実行する	248
▼すべての HADB マシンに対して計画的な保守を実行する	249
▼障害発生時に予定外の保守を実行する	249
履歴ファイルの消去と保存	250
索引	253

表目次

表 3-1	リモートサーバーインスタンス間で同期化されるファイルとディレクトリ	55
表 3-2	ノードエージェントタスクの実行方法	60
表 4-1	ロードバランサ設定のパラメータ	80
表 4-2	健全性検査のパラメータ	83
表 4-3	健全性検査のプロパティ	84
表 10-1	hadbm registerpackage のオプション	192
表 11-1	管理エージェント共通オプション	202
表 11-2	管理エージェントサービスオプション (Windows のみ)	202
表 11-3	設定ファイルの設定値	203
表 11-4	hadbm セキュリティーオプション	206
表 11-5	hadbm 汎用オプション	207
表 11-6	HADB オプションと環境変数	208
表 11-7	hadbm create オプション	212
表 11-8	設定属性	218
表 11-9	HADB 接続プール設定	223
表 11-10	HADB 接続プールプロパティ	223
表 11-11	HADB JDBC リソース設定	224
表 11-12	hadbm clear オプション	232
表 11-13	hadbm addnodes オプション	237
表 11-14	HADB の状態	241
表 11-15	hadbm resourceinfo コマンドオプション	245

例目次

例 2-1	asadmin get および set コマンドによる GMS 設定の変更	37
例 3-1	ノードエージェントの作成	63
例 7-1	可用性が有効になっている EJB 配備記述子の例	154
例 7-2	メソッドのチェックポイント設定を指定する EJB 配備記述子の例	155
例 9-1	RMI-IIOP 重み付きラウンドロビン負荷分散に使用する負荷分散の重みの設定	175
例 10-1	マルチパスの設定	179
例 10-2	HADB の登録解除の例	193
例 11-1	hadbm コマンドの例	205
例 11-2	HADB 管理ドメインの作成	210
例 11-3	データベースの作成例	214
例 11-4	hadbm get の使用例	216
例 11-5	接続プールの作成	224
例 11-6	ノードを起動する例	228
例 11-7	ノードを停止する例	228
例 11-8	ノードを再起動する例	229
例 11-9	データベースを起動する例	230
例 11-10	データベースを停止する例	230
例 11-11	データベースを削除する例	233
例 11-12	データデバイスサイズを設定する例	235
例 11-13	ノードを追加する例	237
例 11-14	データベースを再断片化する例	239
例 11-15	HADB 状態を取得する例	241
例 11-16	デバイス情報を取得する例	244
例 11-17	データバッファプール情報の例	246
例 11-18	ロック情報の例	247
例 11-19	ログバッファ情報の例	247
例 11-20	内部ログバッファ情報の例	247

はじめに

このドキュメントでは、融合負荷分散、HTTP 負荷分散、クラスタ、セッション持続性とフェイルオーバー、および高可用性データベース (HADB) を含む、Enterprise Server での高可用性の諸機能について説明します。

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

Enterprise Server のドキュメントセット

Enterprise Server ドキュメントの URL (Uniform Resource Locator)

は、<http://docs.sun.com/coll/1343.8> です。Enterprise Server への導入としては、次の表に示されている順序でドキュメントを参照してください。

表 P-1 Enterprise Server のドキュメントセットの内容

ドキュメント名	説明
『Documentation Center』	タスクや主題ごとに整理された Enterprise 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』	Enterprise 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 Enterprise 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』	Enterprise 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 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。

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

ドキュメント名	説明
『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。
『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 Enterprise 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使用、およびロードバランサの使用。
『Administration Reference』	Enterprise 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 Enterprise Server の調整。
『Reference Manual』	Enterprise 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。

関連ドキュメント

その他のスタンドアロンの Sun GlassFish サーバー製品については、次のドキュメントを参照してください。

- [メッセージキュー ドキュメント \(http://docs.sun.com/coll/1343.4\)](http://docs.sun.com/coll/1343.4)
- [Identity Server documentation \(http://docs.sun.com/app/docs/prod/ident.mgmt#hic\)](http://docs.sun.com/app/docs/prod/ident.mgmt#hic)
- [Directory Server ドキュメント \(http://docs.sun.com/coll/1224.1\)](http://docs.sun.com/coll/1224.1)
- [Web サーバードキュメント \(http://docs.sun.com/coll/1308.3\)](http://docs.sun.com/coll/1308.3)

Enterprise Server とともに提供されるパッケージの Javadoc™ ツールリファレンスの場所は <http://glassfish.dev.java.net/nonav/javaee5/api/index.html> です。さらに、次のリソースが役立つことがあります。

- [Java EE 5 Specifications \(http://java.sun.com/javaee/5/javatech.html\)](http://java.sun.com/javaee/5/javatech.html)
- [Java EE Blueprints \(http://java.sun.com/reference/blueprints/index.html\)](http://java.sun.com/reference/blueprints/index.html)

NetBeans™ 統合開発環境 (IDE) でのエンタープライズアプリケーション開発については、<http://www.netbeans.org/kb/55/index.html> を参照してください。

Enterprise Server に含まれる Java DB データベースの詳細は、<http://developers.sun.com/javadb/> を参照してください。

GlassFish Samples プロジェクトは、さまざまな Java EE テクノロジーの使用法の例を示すサンプルアプリケーションの集合です。GlassFish Samples は Java EE Software Development Kit (SDK) に付属しています。また、<https://glassfish-samples.dev.java.net/> の GlassFish Samples プロジェクトページからも入手できます。

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

ブレースホルダ	説明	デフォルト値
<i>as-install</i>	Enterprise 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 プラットフォームへのインストールで、ルートユーザーでない場合: <i>user's-home-directory/SUNWappserver</i> Solaris および Linux プラットフォームへのインストールで、ルート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の場合: <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>
<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>

文字の表記ルール

この表は、このドキュメントで使用される文字体裁の種類について説明したものです。

表P-3 文字の表記ルール

スタイル	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、および画面上的コンピュータ出力	.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 (画面上的のコンピュータ出力と対照的に表示)	machine_name% su パスワード:
AaBbCc123	実際の名前または値に置換されるプレースホルダ	ファイルを削除するコマンドは、rm ファイル名です。
AaBbCc123	書名、新しい用語、および強調する用語 (オンラインでは、一部の強調項目は太字で表示される)	ユーザーズガイドの第6章を参照してください。 キャッシュとは、ローカルに格納されているコピーです。 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。

表 P-4 記号の表記ルール (続き)

記号	説明	例	意味
→	グラフィカルユーザーインターフェースでのメニュー項目の選択を示します。	「ファイル」→「新規」→「テンプレート」	「ファイル」メニューから「新規」を選択します。「新規」サブメニューから、「テンプレート」を選択します。

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>を参照し、「フィードバック」をクリックしてください。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。

Enterprise Server の高可用性

この章では、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利用できる Sun GlassFish Enterprise Server の高可用性機能について説明します。

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

この章の内容は、次のとおりです。

- 21 ページの「高可用性の概要」
- 24 ページの「Enterprise Server による高可用性の提供方法」
- 28 ページの「障害からの回復」

高可用性の概要

高可用性アプリケーションおよびサービスは、ハードウェアやソフトウェアの障害には関係なく、機能を継続的に提供します。このようなアプリケーションは、99.999%の時間利用可能であることから、ファイブナインの信頼性を実現していると言われることがあります。

Enterprise Server では、次の高可用性機能を提供します。

- 22 ページの「HTTP ロードバランサプラグイン」
- 22 ページの「高可用性 JMS (Java Message Service)」
- 23 ページの「RMI-IIOP 負荷分散とフェイルオーバー」

HTTP ロードバランサプラグイン

HTTP ロードバランサプラグインは、HTTP および HTTPS 要求を受け付け、それをクラスタ内のアプリケーションサーバーインスタンスに転送します。ネットワーク障害のためにインスタンスが失敗して使用不可になるか、または応答なくなると、ロードバランサは要求を既存の使用可能なマシンにリダイレクトします。ロードバランサはまた、障害が起きたインスタンスが復旧したことを認識し、それに応じて負荷を再配分することもできます。GlassFish Communications Server は、Sun Java System Web Server と Apache Web Server 用、および Microsoft Internet Information Server 用のロードバランサプラグインを提供しています。

ロードバランサによって、ワークロードが複数の物理マシンに分散されるため、全体的なシステムスループットが向上します。HTTP 要求のフェイルオーバーを通して、より高い可用性も提供されます。HTTP セッションの情報を持続させるには、HTTP セッションの持続性を設定する必要があります。

状態を持たない単純なアプリケーションであれば、負荷分散されたクラスタで十分なこともあります。しかし、セッション状態を持ったミッションクリティカルなアプリケーションの場合は、負荷分散されたクラスタを複製された持続性ととも使用します。

負荷分散に関わるサーバーインスタンスとクラスタは、同種の環境を確保しています。これは、通常、サーバーインスタンスが同じサーバー設定を参照し、同じ物理リソースにアクセスでき、さらに配備された同じアプリケーションを持っていることを意味します。この均質性によって、障害の前後に、ロードバランサが常に負荷を均等にクラスタ内のアクティブなインスタンスに分散することが保証されます。

HTTP ロードバランサプラグインの負荷分散およびフェイルオーバーの設定については、[第 4 章「HTTP 負荷分散の設定」](#)を参照してください。

高可用性 JMS (Java Message Service)

Java Message Service (JMS) API は、Java EE アプリケーションおよびコンポーネントに対して、メッセージの作成、送信、受信、および読み取りを可能にするメッセージング標準です。この API によって、緩やかに結合され、信頼性が高く、非同期の分散通信が可能となります。Sun Java System Message Queue (MQ) は JMS を実装し、Enterprise Server と密接に統合されているため、MQ を使用してメッセージ駆動型 Bean (MDB) などの JMS に依存するコンポーネントを作成できます。

接続のプールおよびフェイルオーバーと、MQ クラスタを通じて、JMS の高可用性が実現されます。詳細については、[第 8 章「Java Message Service 負荷分散とフェイルオーバー」](#)を参照してください。

接続プールとフェイルオーバー

Enterprise Server は JMS 接続プールとフェイルオーバーをサポートします。Enterprise Server は JMS 接続を自動的にプールします。デフォルトでは、Enterprise Server は、指定されたホストリストから主 MQ ブローカをランダムに選択します。フェイルオーバーが発生すると、MQ は負荷を別のブローカに透過的に転送し、JMS セマンティクスを保持します。

JMS 接続のプールおよびフェイルオーバーの詳細については、161 ページの「[接続プールとフェイルオーバー](#)」を参照してください。

MQ クラスタ

MQ Enterprise Edition は、ブローカクラスタと呼ばれる、相互に接続した複数のブローカインスタンスをサポートします。ブローカクラスタによって、クライアント接続はクラスタ内のすべてのブローカに分散されます。クラスタ化することで、水平方向のスケラビリティが提供され、可用性が向上します。

MQ クラスタの詳細については、164 ページの「[MQ クラスタと Enterprise Server の併用](#)」を参照してください。

RMI-IIOP 負荷分散とフェイルオーバー

RMI-IIOP 負荷分散では、IIOP クライアント要求が別のサーバーインスタンスまたはネームサーバーに分散されます。目標は、負荷をクラスタ間に均等に拡散して、スケラビリティを実現することです。また、IIOP 負荷分散を EJB のクラスタリングおよび可用性と結合すれば、EJB フェイルオーバーも実現されます。

クライアントがオブジェクトに対して JNDI 検索を実行すると、ネームサービスは、原則的に要求を特定のサーバーインスタンスにバインドします。それ以降、そのクライアントからの検索要求はすべて、同じサーバーインスタンスに送信されます。こうして、すべての EJBHome オブジェクトは、同じターゲットサーバーにホストされます。また、それ以降に取得された Bean 参照もすべて、同じターゲットホスト上に作成されます。JNDI 検索の実行時に、すべてのクライアントがターゲットサーバーのリストをランダムに選択するため、これにより負荷分散が効果的に実現されます。ターゲットサーバーインスタンスが停止すると、検索または EJB メソッド呼び出しは、別のサーバーインスタンスに処理が引き継がれます。

RMI-IIOP 負荷分散とフェイルオーバーは、透過的に発生します。アプリケーションの配備中に、特別な手順は必要ありません。アプリケーションクライアントが配備される Enterprise Server インスタンスがクラスタに参加する場合、Enterprise Server は、クラスタ内で現在アクティブなすべての IIOP 端点を自動的に検出します。ただし、端点の1つで障害が発生した場合に備えて、クライアントにはブートストラップ目的で少なくとも2つの端点を指定しておくことをお勧めします。

RMI-IIOP 負荷分散およびフェイルオーバーの詳細については、第9章「[RMI-IIOP 負荷分散とフェイルオーバー](#)」を参照してください。

詳細情報

ハードウェア要件の評価、ネットワーク構成の計画、およびトポロジの選択を含む、高可用性配備の計画については、『Sun GlassFish Enterprise Server 2.1 配備計画ガイド』を参照してください。また、このドキュメントでは、次に示すような概念への高レベルな導入も提供しています。

- ノードエージェント、ドメイン、クラスタなどの Enterprise Server コンポーネント
- クラスタ内の IIOP 負荷分散
- HADB のアーキテクチャー
- メッセージキューのフェイルオーバー

高可用性機能を利用するアプリケーションの開発の詳細については、『Sun GlassFish Enterprise Server 2.1 Developer's Guide』を参照してください。

高可用性サーバーおよびアプリケーションの調整

高可用性とともに最適なパフォーマンスを得るためにアプリケーションや Enterprise Server を設定および調整する方法については、docs.sun.com で『パフォーマンスチューニングガイド』を参照してください。このドキュメントでは、次のようなトピックが説明されています。

- 持続性の頻度および持続性のスコープの調整
- ステートフルセッション Bean のチェックポイントの設定
- JDBC 接続プールの設定
- セッションサイズ
- HADB のディスク使用、記憶域割り当て、パフォーマンス、およびオペレーティングシステム設定の調整
- 最適なパフォーマンスを得るためのロードバランサの設定

Enterprise Server による高可用性の提供方法

Enterprise Server は、次のサブコンポーネントと機能を通して高可用性を提供します。

- 25 ページの「セッション状態データ用ストレージ」
- 26 ページの「高可用性クラスタ」

セッション状態データ用ストレージ

セッション状態データを格納することにより、クラスタ内のサーバーインスタンスのフェイルオーバー後にセッション状態を復元できるようになります。セッション状態の復元により、情報を失うことなくセッションを継続できます。Enterprise Server では、HTTPセッションおよびステートフルセッション Bean のデータを格納するための、次のタイプの高可用性ストレージが用意されています。

- クラスタ内の別のサーバーにおけるインメモリーレプリケーション
- 高可用性データベース

クラスタ内の別のサーバーにおけるインメモリーレプリケーション

ほかのサーバー上でインメモリーレプリケーションを実行することにより、HADB などの別個のデータベースを入手しなくてもセッション状態データの軽量ストレージを用意できます。このタイプのレプリケーションは、ほかのサーバー上のメモリーを使用して HTTPセッションとステートフルセッション Bean データの高可用性ストレージを実現します。クラスタ化されたサーバーインスタンスはセッション状態をリングトポロジで複製します。各バックアップインスタンスは複製されたデータをメモリーに格納します。セッション状態データをほかのサーバー上のメモリーに複製することによって、セッションを分散することが可能になります。

インメモリーレプリケーションを使用するには、「グループ管理サービス (GMS)」を有効にする必要があります。GMS の詳細については、[36 ページの「グループ管理サービス」](#)を参照してください。

クラスタ内の複数のサーバーインスタンスが異なるマシンに配置されている場合は、次の前提条件が満たされていることを確認してください。

- GMS およびインメモリーレプリケーションが正常に機能することを保証するには、すべてのマシンが同じサブネット上に存在する必要があります。
- インメモリーレプリケーションが正常に機能することを保証するには、クラスタ内のすべてのマシンのシステムクロックができるだけ厳密に同期している必要があります。

高可用性データベース

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

GlassFish Communications Server は、HTTP セッションデータおよびステートフルセッション Bean データの高可用性ストレージのための高可用性データベース (HADB) を提供します。HADB は、負荷分散、フェイルオーバー、および状態復元により、最大 99.999% のサービスおよびデータの可用性をサポートするように設計されています。一般に、HADB は、Enterprise Server とは独立に設定および管理する必要があります。

状態管理の機能を GlassFish Communications Server と切り離しておくことには、大きな利点があります。GlassFish Communications Server インスタンスは、状態レプリケーションを外部の高可用性状態サービスに委任した、スケーラブルで高性能なアプリケーションコンテナとしての動作に CPU サイクルを消費します。この疎結合のアーキテクチャーのために、GlassFish Communications Server インスタンスを容易にクラスタに追加したり、クラスタから削除したりできます。HADB の状態レプリケーションサービスを独立に拡張して、最適な可用性とパフォーマンスを得ることができます。GlassFish Communications Server インスタンスがレプリケーションも実行していると、Java EE アプリケーションのパフォーマンスが低下したり、ガベージコレクションの一時停止時間が長くなったりすることがあります。

ハードウェアの設定、サイズ、およびトポロジの決定など、インストールで HADB による高可用性を計画および設定する方法については、『[Sun GlassFish Enterprise Server 2.1 配備計画ガイド](#)』を参照してください。

高可用性クラスタ

「クラスタ」とは、1つの論理エンティティとして一体となって動作するインスタンスの集まりです。クラスタは、1つ以上の Java EE アプリケーションに対して実行時環境を提供します。「高可用性クラスタ」では、状態レプリケーションサービスと、クラスタおよびロードバランサが統合されています。

クラスタの使用には、次の利点があります。

- 高可用性: クラスタ内のサーバーインスタンスに対するフェイルオーバーを可能にすることで実現します。1つのサーバーインスタンスが停止すると、別のサーバーインスタンスが、利用できないサーバーインスタンスが処理していた要求を引き継ぎます。

- スケーラビリティ: クラスタにサーバーインスタンスを追加できるようにし、それによってシステムの能力が増強されることによって実現します。ロードバランサプラグインは、要求をクラスタ内の使用可能なサーバーインスタンスに分配します。管理者はより多くのサーバーインスタンスをクラスタに追加しているので、処理の中断の必要はありません。

クラスタ内のすべてのインスタンスが次のように動作します。

- 同じ設定を参照します。
- Java EE アプリケーションの EAR ファイル、Web モジュールの WAR ファイル、EJB JAR ファイルなど、配備されたアプリケーションの同じセットを所有します。
- 同じ一連のリソースを所有しているため、同じ JNDI 名前空間が構成されます。

ドメイン内のすべてのクラスタが一意の名前を持ちます。また、この名前は、すべてのノードエージェント名、サーバーインスタンス名、クラスタ名、および設定名の間でも一意である必要があります。この名前を `domain` に使用してはいけません。アプリケーションの配備やリソースの作成など、クラスタ化されていないサーバーインスタンスで実行する操作と同じ操作をクラスタ上で実行します。

クラスタと設定

クラスタの設定は、ほかのクラスタで共有される可能性のある、名前を付けられている設定から派生されます。設定をほかのサーバーインスタンスまたはクラスタと共有していないクラスタは、スタンドアロン設定を持っていると言われます。デフォルトで、この設定の名前は `cluster_name -config` です。 `cluster_name` はクラスタの名前です。

設定をほかのクラスタまたはインスタンスと共有しているクラスタは、共有設定を持っていると言われます。

クラスタ、インスタンス、セッション、および負荷分散

クラスタ、サーバーインスタンス、ロードバランサ、およびセッションの関係は次のとおりです。

- サーバーインスタンスがクラスタの一部である必要はありません。ただし、クラスタの一部でないインスタンスは、1つのインスタンスから別のインスタンスへとセッション状態を移すことによって得られる高可用性を利用することはできません。
- クラスタ内のサーバーインスタンスを1つまたは複数のマシンでホストすることができます。異なるマシンにまたがるサーバーインスタンスを1つのクラスタにグループ化できます。

- 特定のロードバランサは、複数のクラスタにあるサーバーインスタンスに要求を転送できます。ロードバランサのこの機能を使って、サービスを中断することなく、オンラインアップグレードを実行できます。詳細については、「クラスタの設定」の章の「複数のクラスタを使用してのサービス中断のないオンラインアップグレードサービス」を参照してください。
- 単一のクラスタは、複数のロードバランサから要求を受信できます。クラスタが、2つ以上のロードバランサからサービスを受ける場合、各ロードバランサで、まったく同一に、クラスタを設定する必要があります。
- 各セッションは、特定のクラスタに関連づけられます。そのため、1つのアプリケーションを複数のクラスタに配備することは可能ですが、セッションフェイルオーバーは単一のクラスタ内でのみ発生します。

したがってクラスタは、そのクラスタ内のサーバーインスタンスがフェイルオーバーしたときには、安全境界として機能します。ロードバランサを使って、サービスを停止することなく、Enterprise Server 内のコンポーネントをアップグレードすることができます。

障害からの回復

- 28 ページの「Sun Cluster の使用」
- 28 ページの「手動での復旧」
- 31 ページの「Netbackup の使用」
- 32 ページの「ドメイン管理サーバーの再作成」

Sun Cluster の使用

Sun Cluster では、ドメイン管理サーバー、ノードエージェント、Enterprise Server インスタンスおよびメッセージキューの自動フェイルオーバーを提供しています。詳細については、docs.sun.com で『Sun Cluster Data Service for Sun Java System Application Server Guide for Solaris OS』を参照してください。

標準の Ethernet 相互接続および Sun Cluster 製品のサブセットを使用します。この機能は、Java ES に含まれています。

手動での復旧

さまざまな方法を使用して、個別のサブコンポーネントを手動で復旧できます。

- 29 ページの「ドメイン管理サーバーの回復」
- 29 ページの「ノードエージェントおよびサーバーインスタンスの回復」
- 30 ページの「HTTP ロードバランサおよび Web Server の回復」

- 30 ページの「メッセージキューの回復」
- 30 ページの「HADBの復元」

ドメイン管理サーバーの回復

ドメイン管理サーバー (DAS) が失われて影響があるのは、管理だけです。たとえ DAS が到達不可能であっても、Enterprise Server クラスタおよびアプリケーションは、それまでどおり稼働し続けます。

DAS を復旧するには、次のいずれかの方法を使用します。

- `asadmin` でバックアップコマンドを定期的に行って、定期的なスナップショットを作成します。ハードウェア障害後に、新しいマシンに同じネットワーク ID で Application Server をインストールし、以前に作成されたバックアップから `asadmin` で復元を実行します。詳細については、32 ページの「ドメイン管理サーバーの再作成」を参照してください。
- 共有されている強固なファイルシステム (NFS など) 上で、ドメインのインストールと設定をします。主 DAS マシンで問題が発見されると、2 番目のマシンが同じ IP アドレスで稼働し、手動による介入、またはユーザーが指定した自動化によって引き継がれます。Sun クラスタでは、DAS フォールトトレラントを実現する場合と同様の手法を使用します。
- Enterprise Server のインストールおよびドメインルートディレクトリを zip 形式で圧縮します。それを新しいマシンで復元し、同じネットワーク ID を割り当てます。ファイルベースのインストールを使用している場合は、これがもっとも簡単な方法である可能性があります。
- DAS バックアップから復元します。AS8.1 UR2 パッチ 4 の指示を参照してください。

ノードエージェントおよびサーバーインスタンスの回復

ノードエージェントおよびサーバーインスタンスを回復する方法は 2 つあります。

バックアップの zip ファイルを保持する。ノードエージェントおよびサーバーインスタンスをバックアップする明確なコマンドはありません。ノードエージェントディレクトリの内容を持つ zip ファイルを単に作成します。障害発生後に、同じホスト名と IP アドレスを持つ新しいマシンで保存済みのバックアップを解凍します。インストールディレクトリの場所、OS などと同じものを使用します。ファイルベースのインストール、パッケージベースのインストール、または復元されたバックアップイメージがマシン上に存在する必要があります。

手動での回復。同じ IP アドレスを持つ新しいホストを使用する必要があります。

1. Enterprise Server をノードエージェントとともにマシンにインストールします。
2. ノードエージェントを再作成します。サーバーインスタンスを作成する必要はありません。

3. 同期によって、設定およびデータが DAS からコピーされ更新されます。

HTTP ロードバランサおよび Web Server の回復

Web Server の設定だけをバックアップする明確なコマンドはありません。Web Server のインストールディレクトリを単に圧縮します。障害発生後に、同じネットワーク ID を持つ新しいマシンで保存済みのバックアップを解凍します。新しいマシンの IP アドレスが異なる場合は、DNS サーバーまたはルーターを更新します。

注-これは、あらかじめ Web Server にイメージが再インストールまたは復元されていることを前提としています。

ロードバランサプラグイン (plugins ディレクトリ) および設定は、Web Server のインストールディレクトリ (通常は /opt/SUNWwbsvr) にあります。web-install/web-instance/config ディレクトリには loadbalancer.xml ファイルがあります。

メッセージキューの回復

メッセージキュー (MQ) の設定およびリソースは、DAS に格納され、インスタンスに同期できます。その他のデータおよび設定情報は、すべて MQ ディレクトリ (通常は /var/imq の下) にあるため、必要に応じてこれらのディレクトリをバックアップおよび復元します。新しいマシンには、あらかじめ MQ インストールが含まれている必要があります。マシンを復元するときは、それまでどおり MQ ブローカが起動していることを必ず確認してください。

HADB の復元

注-HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

アクティブな HADB ノードが 2 つある場合は、障害時に引き継ぐことができる 2 つのスペアノードを (別々のマシン上に) 設定できます。HADB のバックアップおよび復元では不整合セッションが復元される可能性があるため、この方法のほうがクリーンです。

スペアノードのあるデータベースを作成する方法については、[210 ページ](#)の「[データベースの作成](#)」を参照してください。スペアノードをデータベースに追加する方法

については、236 ページの「ノードの追加」を参照してください。復旧および自己修復に失敗すると、スベアノードによって自動的に引き継がれます。

Netbackup の使用

注- この手順は、Sun QA によって確認されていません。

各マシンのイメージを保存するには、Veritas Netbackup を使用します。BPIP の場合は、Web サーバーおよび Application Server とともに 4 つのマシンをバックアップします。

復元されたそれぞれのマシンについて、元のマシンと同じ設定 (同じホスト名、IP アドレスなど) を使用します。

Enterprise Server のようなファイルベース製品では、関連するディレクトリだけをバックアップおよび復元します。ただし、Web サーバーイメージのようなパッケージベースのインストールでは、マシン全体をバックアップおよび復元する必要があります。パッケージは、Solaris パッケージデータベースにインストールされます。そのため、ディレクトリだけをバックアップし、続いて新しいシステムに復元すると、パッケージデータベースには情報のない、「展開された」Web サーバーになります。このため、今後のパッチ適用やアップグレードで問題が発生する可能性があります。

Solaris パッケージデータベースを手動でコピーおよび復元しないでください。代替方法は、Web サーバーなどのコンポーネントのインストール後にマシンのイメージをバックアップすることです。これをベースライン tar ファイルと呼びます。Web サーバーに変更を加えた場合は、たとえば /opt/SUNWwbsvr の下にあるこれらのディレクトリをバックアップします。復元するには、ベースライン tar ファイルから開始し、次に変更した Web サーバーディレクトリを上書きコピーします。同様に、MQ でもこの手順を使用できます (BPIP のパッケージベースのインストール)。元のマシンをアップグレードまたはパッチ適用する場合、新しいベースライン tar ファイルを作成する必要があります。

DAS のあるマシンがダウンすると、復元するまで利用できない時間が発生します。

DAS は、中央リポジトリです。サーバーインスタンスを復元して再起動すると、DAS にある情報とのみ同期されます。そのため、すべての変更は asadmin または管理コンソールを使用して実行する必要があります。

イメージにアプリケーションセッションの古い状態が含まれる可能性があるため、HADB の日次バックアップは機能しないことがあります。

ドメイン管理サーバーの再作成

ドメイン管理サーバー (DAS) をホストしているマシンで障害が発生した場合、以前に DAS をバックアップしたことがあれば DAS を再作成できます。DAS の作業コピーを再作成するには、次の環境が必要です。

- 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 台目のマシンと同様に、**Enterprise Server** を 3 台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、**Enterprise Server** 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 ZIP ファイルを 3 番目のマシンに復元します。

```
asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip
--clienthostname machine3 domain1
```

注---clienthostname オプションを指定することによって、domain.xml ファイル内の jmx-connector 要素の client-hostname プロパティーを変更する必要性を回避します。

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次のとおりです。drwx----- (または 700)。

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も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 番目のマシン上の domain-root-dir/domain1/config/domain.xml ファイルで、jms-service 要素の host 属性の値を更新します。

この属性の元の設定は次のとおりです。

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

この属性の設定を次のように変更します。

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

- 6 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
```

DAS は稼働中のすべてのノードエージェントと通信し、DAS と通信するための情報をノードエージェントに提供します。ノードエージェントはこの情報を使用して DAS と通信します。

- 7 DAS の再起動時に稼働していないすべてのノードエージェントについて、machine2 で as-install/nodeagents/nodeagent/agent/config/das.properties の agent.das.host プロパティー値を変更します。

この手順は、DAS の再起動時に稼働しているノードエージェントについては不要です。

- 8 **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タイ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。

Enterprise Server でのクラスタの設定

この章では、Enterprise Server クラスタの使用法について説明します。次の節で構成されています。

- 35 ページの「クラスタの概要」
- 36 ページの「グループ管理サービス」
- 38 ページの「クラスタの操作」

クラスタの概要

クラスタは、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。異なるマシン上のサーバーインスタンスを1つの論理クラスタにまとめ、それらのインスタンスを1つの単位として管理できます。マルチマシンクラスタのライフサイクルは、DASを使用して容易に制御できます。

インスタンスはクラスタにグループ化できます。アプリケーションを単一配備のクラスタのすべてのインスタンスに分散できます。クラスタは動的です。インスタンスが追加または削除されると、変更は自動的に処理されます。

クラスタにより、水平方向のスケーラビリティ、負荷分散、およびフェイルオーバー保護が使用可能になります。定義により、クラスタ内のすべてのインスタンスに対してリソースとアプリケーションの設定は同じになります。あるサーバーインスタンスまたはクラスタ内のあるマシンに障害が起きると、ロードバランサは障害を検出し、障害の起きたインスタンスからクラスタ内のほかのインスタンスにトラフィックをリダイレクトし、ユーザーセッションの状態を回復します。クラスタ内のすべてのインスタンス上には同一のアプリケーションとリソースがあるため、インスタンスはクラスタ内のほかのどのインスタンスにも処理を継続させることができます。

クラスタインスタンスはリングトポロジの形式に構成されています。リング内の各メンバーは、リング内の次のメンバー(レプリカパートナー)にインメモリー状態

データを送信し、前のメンバーから状態データを受信します。いずれかのメンバーで状態データが更新されると、リング全体でレプリケートされます。リングトポロジでメンバーに障害が発生すると、リングは切断されます。グループ管理サービス (GMS) で、メンバーの障害を認識できます。その場合、レプリケーションフレームワークはクラスタのトポロジを作り直して、メンバーにその変更を通知します。メンバーはそのレプリカパートナーが消えたことを認識すると、稼働中のメンバーから新しいパートナーを選択します。

グループ管理サービス

グループ管理サービス (GMS) は、クラスタ内のインスタンスに対して有効になっている基盤コンポーネントです。GMS を有効にすると、クラスタインスタンスで障害が発生した場合にクラスタおよびドメイン管理サーバー (DAS) が障害を認識し、障害発生時に必要な処理を実行することができます。Enterprise Server の多くの機能は、GMS に依存します。たとえば、GMS は IIOP フェイルオーバー、インメモリーレプリケーション、トランザクションサービス、およびタイマーサービスの各機能によって使用されます。

クラスタ内の複数のサーバーインスタンスが異なるマシンに配置されている場合は、それらのマシンが同じサブネット上にあることを確認してください。

注 - GMS 機能は開発者プロファイルでは利用できません。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では、デフォルトで GMS が有効です。

GMS は Shoal フレームワークの中心的なサービスです。Shoal の詳細については、[Project Shoal ホームページ \(https://shoal.dev.java.net/\)](https://shoal.dev.java.net/) を参照してください。

▼ クラスタに対して **GMS** を有効または無効にする

- 1 ツリーコンポーネントで、「クラスタ」を選択します。
- 2 クラスタの名前をクリックします。
- 3 「一般情報」で、必要に応じて「ハートビート有効」チェックボックスが選択または選択解除されていることを確認します。
 - **GMS** を有効にするには、「ハートビート有効」チェックボックスが選択されていることを確認します。
 - **GMS** を無効にするには、「ハートビート有効」チェックボックスが選択解除されていることを確認します。

- 4 **GMS** を有効にしており、**GMS** で使用するデフォルトのポートおよび IP アドレスを変更する必要がある場合は、これらの設定値を変更します。
- 5 「保存」をクリックします。

GMS の設定

使用環境に合わせて GMS を設定します。これは、GMS が障害を確認する頻度を決定する設定値を変更することによって行います。たとえば、障害検出試行の間のタイムアウト、障害が疑われるメンバーの再試行の回数、または、クラスタのメンバーを確認するときのタイムアウトを変更できます。

次の例では、get コマンドを使用して、*cluster-config-name* に関連付けられているすべてのプロパティを取得します。

```
asadmin get cluster2-config.group-management-service.*

cluster2-config.group-management-service.fd-protocol-max-tries = 3
cluster2-config.group-management-service.fd-protocol-timeout-in-millis = 2000

cluster2-config.group-management-service.merge-protocol-max-interval-in-millis
= 10000

cluster2-config.group-management-service.merge-protocol-min-interval-in-millis
= 5000

cluster2-config.group-management-service.ping-protocol-timeout-in-millis = 5000

cluster2-config.group-management-service.vs-protocol-timeout-in-millis = 1500
```

▼ 管理コンソールを使用して **GMS** 設定を設定する

- 1 管理コンソールで、**Enterprise Server** ノードに移動します。
- 2 「設定」 -> *cluster_name* -config -> 「グループ管理サービス」の順にクリックします。

例 2-1 asadmin get および set コマンドによる GMS 設定の変更

管理コンソールの代わりに、asadmin get および set コマンドを使用できます。

```
asadmin> list cluster2-config.*
cluster2-config.admin-service
cluster2-config.admin-service.das-config
cluster2-config.admin-service.jmx-connector.system
```

```

cluster2-config.admin-service.jmx-connector.system.ssl
cluster2-config.availability-service
cluster2-config.availability-service.jms-availability
cluster2-config.availability-service.sip-container-availability
cluster2-config.diagnostic-service
cluster2-config.ejb-container
cluster2-config.ejb-container-availability
cluster2-config.ejb-container.ejb-timer-service
...
...
...
...
cluster2-config.web-container-availability

asadmin> get cluster2-config.group-management-service.*
cluster2-config.group-management-service.fd-protocol-max-tries = 3
cluster2-config.group-management-service.fd-protocol-timeout-in-millis = 2000
cluster2-config.group-management-service.merge-protocol-max-interval-in-millis = 10000
cluster2-config.group-management-service.merge-protocol-min-interval-in-millis = 5000
cluster2-config.group-management-service.ping-protocol-timeout-in-millis = 5000
cluster2-config.group-management-service.vs-protocol-timeout-in-millis = 1500

asadmin>set cluster2-config.group-management-service.fd-protocol-max-tries=4
cluster2-config.group-management-service.fd-protocol-max-tries = 4

asadmin> get cluster2-config.group-management-service.*
cluster2-config.group-management-service.fd-protocol-max-tries = 4
cluster2-config.group-management-service.fd-protocol-timeout-in-millis = 2000
cluster2-config.group-management-service.merge-protocol-max-interval-in-millis = 10000
cluster2-config.group-management-service.merge-protocol-min-interval-in-millis = 5000
cluster2-config.group-management-service.ping-protocol-timeout-in-millis = 5000
cluster2-config.group-management-service.vs-protocol-timeout-in-millis = 1500

```

ロードバランサを作成したときにクラスタがすでに起動している場合、ロードバランサを起動するには、クラスタを再起動する必要があります。

クラスタの操作

- 39 ページの「クラスタを作成する」
- 40 ページの「クラスタのサーバーインスタンスを作成するには」
- 41 ページの「クラスタを設定する」
- 42 ページの「クラスタ化されたインスタンスを起動、停止、および削除する」
- 42 ページの「クラスタ内のサーバーインスタンスを設定するには」
- 43 ページの「クラスタ用のアプリケーションを設定する」
- 44 ページの「クラスタ用のリソースを設定する」
- 44 ページの「クラスタを削除する」

- 45 ページの「EJB タイマーを移行する」
- 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」

▼ クラスタを作成する

- 1 ツリーコンポーネントで、「クラスタ」ノードを選択します。
- 2 「クラスタ」ページで、「新規」をクリックします。
「クラスタの作成」ページが表示されます。または「新しいクラスタ」ページが表示されます。
- 3 「名前」フィールドで、クラスタの名前を入力します。
この名前は、次の条件を満たす必要があります。
 - 大文字と小文字、数字、下線、ハイフン、およびピリオド(.)だけで構成される
 - すべてのノードエージェント名、サーバーインスタンス名、クラスタ名、および設定名の間で一意である
 - 「domain」ではない
- 4 「設定」フィールドで、ドロップダウンリストから設定を選択します。
 - 共用設定を使用しないクラスタを作成するには、`default-config`を選択します。
「選択している設定のコピーを作成します」ラジオボタンを選択済みにしておきます。デフォルト設定のコピーは、`cluster_name-config`という名前になります。
 - 共用設定を使用するクラスタを作成するには、ドロップダウンリストから設定を選択します。
「選択している設定を参照します」ラジオボタンを選択して、指定した既存の共用設定を使用するクラスタを作成します。
- 5 オプションとして、サーバーインスタンスを追加できます。
クラスタ作成後にサーバーインスタンスを追加することも可能です。
サーバーインスタンスは複数のマシンに常駐できます。すべてのサーバーインスタンスは、DAS と通信可能なノードエージェントに関連付ける必要があります。クラスタのサーバーインスタンスを作成する前に、まず1つまたは複数のノードエージェントまたはノードエージェントのプレースホルダを作成します。[61 ページの「ノードエージェントのプレースホルダを作成する」](#)を参照してください。
サーバーインスタンスを作成するには、次のようにします。
 - a. 「作成するサーバーインスタンス」セクションで、「追加」をクリックします。

- b. 「インスタンス名」フィールドにインスタンスの名前を入力します。
 - c. 「ノードエージェント」ドロップダウンリストからノードエージェントを選択します。
- 6 「了解」をクリックします。
 - 7 表示される「クラスタを正常に作成」ページで「了解」をクリックします。

参考 同機能を持つ `asadmin` コマンド

`create-cluster`

- 参照
- 41 ページの「クラスタを設定する」
 - 40 ページの「クラスタのサーバーインスタンスを作成するには」
 - 43 ページの「クラスタ用のアプリケーションを設定する」
 - 44 ページの「クラスタ用のリソースを設定する」
 - 44 ページの「クラスタを削除する」
 - 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」

クラスタ、サーバーインスタンス、およびノードエージェントを管理する方法の詳細については、51 ページの「ノードエージェントの配備」を参照してください。

▼ クラスタのサーバーインスタンスを作成するには

始める前に クラスタのサーバーインスタンスを作成する前に、まずノードエージェントまたはノードエージェントのプレースホルダを作成します。61 ページの「ノードエージェントのプレースホルダを作成する」を参照してください。

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 クラスタのノードを選択します。
- 3 「インスタンス」タブをクリックして、「クラスタ化されたサーバーインスタンス」ページを表示します。
- 4 「新規」をクリックして、「クラスタ化されたサーバーインスタンスの作成」ページまたは「新しいクラスタ化されたサーバーインスタンス」ページを表示します。
- 5 「名前」フィールドで、サーバーインスタンスの名前を入力します。
- 6 「ノードエージェント」ドロップダウンリストからノードエージェントを選択します。

- 7 「了解」をクリックします。

参考 同機能を持つ asadmin コマンド

create-instance

- 参照
- 49 ページの「ノードエージェントとは」
 - 39 ページの「クラスタを作成する」
 - 41 ページの「クラスタを設定する」
 - 43 ページの「クラスタ用のアプリケーションを設定する」
 - 44 ページの「クラスタ用のリソースを設定する」
 - 44 ページの「クラスタを削除する」
 - 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」
 - 42 ページの「クラスタ内のサーバーインスタンスを設定するには」

▼ クラスタを設定する

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 クラスタのノードを選択します。
「一般情報」ページで、次のタスクを実行できます。
 - 「クラスタの起動」をクリックして、クラスタ化されたサーバーインスタンスを起動します。
 - 「クラスタの停止」をクリックして、クラスタ化されたサーバーインスタンスを停止します。
 - 「EJB タイマーを移行」をクリックして、停止されたサーバーインスタンスからクラスタ内の別のサーバーインスタンスに EJB タイマーを移行します。

参考 同機能を持つ asadmin コマンド

start-cluster、stop-cluster、migrate-timers

- 参照
- 39 ページの「クラスタを作成する」
 - 40 ページの「クラスタのサーバーインスタンスを作成するには」
 - 43 ページの「クラスタ用のアプリケーションを設定する」
 - 44 ページの「クラスタ用のリソースを設定する」
 - 44 ページの「クラスタを削除する」
 - 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」
 - 45 ページの「EJB タイマーを移行する」

▼ クラスタ化されたインスタンスを起動、停止、および削除する

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 サーバーインスタンスを含むクラスタのノードを開きます。
- 3 「インスタンス」タブをクリックして、「クラスタ化されたサーバーインスタンス」ページを表示します。
このページで、次のタスクを実行できます。
 - インスタンスのチェックボックスを選択して「削除」、「起動」、または「停止」をクリックし、指定したすべてのサーバーインスタンスに対して選択したアクションを実行します。
 - インスタンスの名前をクリックして、「一般情報」ページを表示します。

▼ クラスタ内のサーバーインスタンスを設定するには

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 サーバーインスタンスを含むクラスタのノードを開きます。
- 3 サーバーインスタンスノードを選択します。
- 4 「一般情報」ページでは、次の操作を行えます。
 - 「インスタンスを起動」をクリックして、インスタンスを起動します。
 - 「インスタンスの停止」をクリックして、実行するインスタンスを停止します。
 - 「JNDI ブラウズ」をクリックして、実行中のインスタンスの JNDI ツリーをブラウズします。
 - 「ログファイルを表示」をクリックして、サーバーのログビューアを開きます。
 - 「ログをローテーション」をクリックして、インスタンスのログファイルをローテーションします。このアクションは、ログファイルのローテーションをスケジュールします。実際のローテーションは、次にログファイルがエントリに書き込まれたときに行われます。
 - 「トランザクションの回復」をクリックして、未完了のトランザクションを回復します。
 - 「プロパティ」タブをクリックして、インスタンスのポート番号を変更します。

- 「監視」タブをクリックして、監視プロパティを変更します。

- 参照
- 39 ページの「クラスタを作成する」
 - 41 ページの「クラスタを設定する」
 - 40 ページの「クラスタのサーバーインスタンスを作成するには」
 - 43 ページの「クラスタ用のアプリケーションを設定する」
 - 44 ページの「クラスタ用のリソースを設定する」
 - 44 ページの「クラスタを削除する」
 - 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」
 - 『Sun GlassFish Enterprise Server 2.1 管理ガイド』の「Enterprise Server のトランザクションからの回復方法を設定する」

▼ クラスタ用のアプリケーションを設定する

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 クラスタのノードを選択します。
- 3 「アプリケーション」タブをクリックして、「アプリケーション」ページを表示します。

このページで、次のタスクを実行できます。

- 「配備」ドロップダウンリストから、配備するアプリケーションのタイプを選択します。表示される「配備」ページで、アプリケーションを指定します。
- 「フィルタ」ドロップダウンリストから、リストに表示するアプリケーションのタイプを選択します。
- アプリケーションを編集するには、アプリケーション名をクリックします。
- アプリケーションの横にあるチェックボックスを選択して、「有効」または「無効」を選択し、クラスタのアプリケーションを有効または無効にします。

- 参照
- 39 ページの「クラスタを作成する」
 - 41 ページの「クラスタを設定する」
 - 40 ページの「クラスタのサーバーインスタンスを作成するには」
 - 44 ページの「クラスタ用のリソースを設定する」
 - 44 ページの「クラスタを削除する」
 - 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」

▼ クラスタ用のリソースを設定する

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 クラスタのノードを選択します。
- 3 「リソース」タブをクリックして、「リソース」ページを表示します。
このページで、次のタスクを実行できます。
 - クラスタ用の新規リソースを作成します。「新規」ドロップダウンリストから、作成するリソースのタイプを選択します。リソースを作成するときには、必ずクラスタをターゲットとして指定します。
 - リソースをグローバルに有効または無効にします。リソースの横にあるチェックボックスを選択して、「有効」または「無効」をクリックします。このアクションはリソースを削除しません。
 - 特定のタイプのリソースのみを表示します。「フィルタ」ドロップダウンリストから、リストに表示するリソースのタイプを選択します。
 - リソースを編集します。リソース名をクリックします。

- 参照
- [39 ページの「クラスタを作成する」](#)
 - [41 ページの「クラスタを設定する」](#)
 - [40 ページの「クラスタのサーバーインスタンスを作成するには」](#)
 - [43 ページの「クラスタ用のアプリケーションを設定する」](#)
 - [44 ページの「クラスタを削除する」](#)

▼ クラスタを削除する

- 1 ツリーコンポーネントで、「クラスタ」ノードを選択します。
- 2 「クラスタ」ページで、クラスタ名の横にあるチェックボックスを選択します。
- 3 「削除」をクリックします。

参考 同機能を持つ asadmin コマンド

```
delete-cluster
```

- 参照
- [39 ページの「クラスタを作成する」](#)
 - [41 ページの「クラスタを設定する」](#)
 - [40 ページの「クラスタのサーバーインスタンスを作成するには」](#)

- 43 ページの「クラスタ用のアプリケーションを設定する」
- 44 ページの「クラスタ用のリソースを設定する」
- 46 ページの「サービスを停止せずにコンポーネントをアップグレードする」

▼ EJB タイマーを移行する

サーバーインスタンスが異常に、または突然実行を停止した場合、そのサーバーインスタンス上にインストールされた EJB タイマーを、クラスタ内の実行中サーバーインスタンスに移動する必要があります。このためには、次の手順を実行します。

- 1 ツリーコンポーネントで、「クラスタ」ノードを開きます。
- 2 クラスタのノードを選択します。
- 3 「一般情報」ページで、「EJB タイマーを移行」をクリックします。
- 4 「EJB タイマーを移行」ページで、次の操作を行います。
 - a. 「ソース」ドロップダウンリストから、タイマーの移行元である停止されたサーバーインスタンスを選択します。
 - b. (省略可能) 「送信先」ドロップダウンリストから、タイマーを移行する先の実行中サーバーインスタンスを選択します。
このフィールドを空のままにした場合、実行中のサーバーインスタンスがランダムに選択されます。
 - c. 「了解」をクリックします。
- 5 送信先サーバーインスタンスを停止して再起動します。
ソースサーバーインスタンスが実行中の場合、または送信先サーバーインスタンスが停止中の場合は、管理コンソールにエラーメッセージが表示されます。

参考 同機能を持つ asadmin コマンド

```
migrate-timers
```

- 参照
- 41 ページの「クラスタを設定する」
 - EJB タイマーサービスの設定に関する管理コンソールオンラインヘルプ

▼ サービスを停止せずにコンポーネントをアップグレードする

ロードバランサと複数のクラスタを使用して、サービスを停止することなく、Enterprise Server内のコンポーネントをアップグレードできます。たとえば、コンポーネントとして、JVM、Enterprise Server、またはWebアプリケーションが可能です。

次の場合、この方法は使えません。

- 高可用性データベース (HADB) のスキーマを変更する場合。詳細については、第 11 章「高可用性データベースの管理」を参照してください。

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『Sun GlassFish Enterprise Server 2.1 Installation Guide』の「Distribution Types and Their Components」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『Sun GlassFish Enterprise Server 2.1 管理ガイド』の「プロファイル」を参照してください。

- アプリケーションデータベーススキーマに対する変更を含むアプリケーションアップグレードを実行する場合。



注意 - クラスタ内のすべてのサーバーインスタンスは一緒にアップグレードします。そうでないと、1つのインスタンスから異なるバージョンのコンポーネントを実行するインスタンスに処理を継続するセッションによって、バージョンミスマッチが発生するリスクがあります。

- 1 クラスタの「一般情報」ページで「クラスタの停止」ボタンを使って、クラスタの1つを停止します。
- 2 そのクラスタでコンポーネントをアップグレードします。
- 3 クラスタの「一般情報」ページで「クラスタの起動」ボタンを使って、クラスタを起動します。
- 4 ほかのクラスタで、1つずつプロセスを繰り返します。

1つのクラスタ内のセッションから別のクラスタ内のセッションに処理を引き継ぐことはないので、1つのバージョンのコンポーネントを実行しているサーバーインスタンスから、異なるバージョンのコンポーネントを実行している(別のクラスタ内の)別のサーバーインスタンスへのセッションへ処理が継続されることに

よって、バージョンのミスマッチが発生する危険はありません。クラスタは、このようにそのクラスタ内のサーバーインスタンスのセッションフェイルオーバーに対する安全境界として機能します。

ノードエージェントの設定

この章では、Enterprise Server のノードエージェントについて説明します。次の節で構成されています。

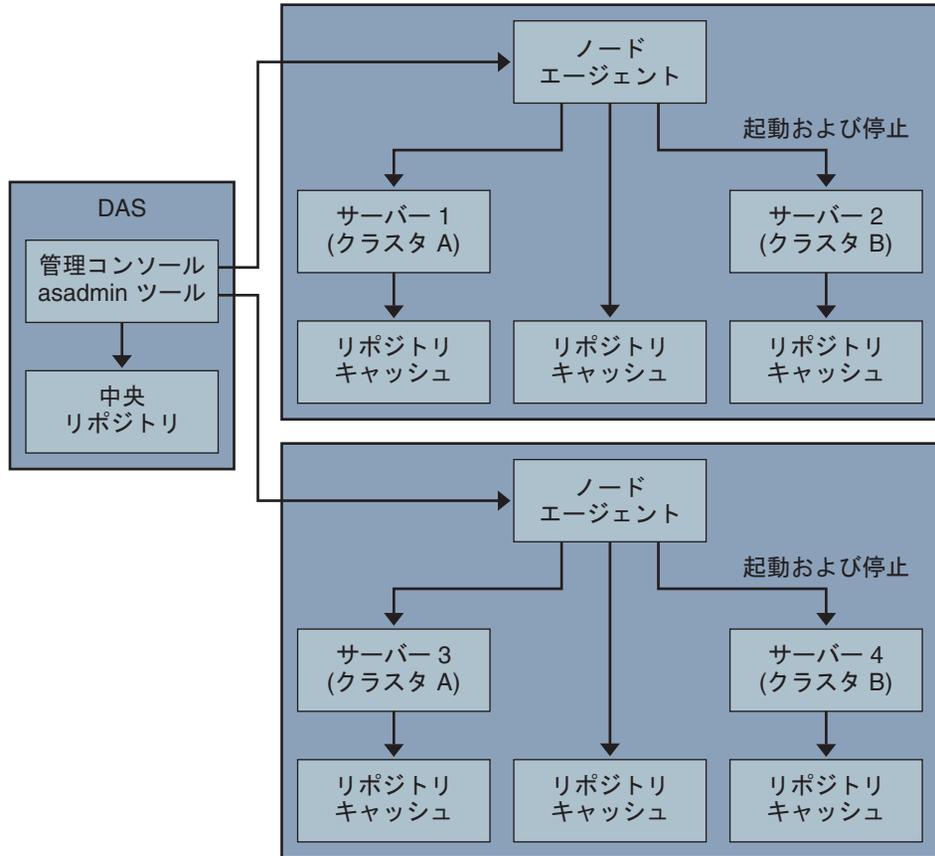
- 49 ページの「ノードエージェントとは」
- 51 ページの「ノードエージェントの障害発生後のサーバーインスタンスの動作」
- 51 ページの「ノードエージェントの配備」
- 54 ページの「ノードエージェントとドメイン管理サーバーとの同期化」
- 59 ページの「ノードエージェントログの表示」
- 59 ページの「ノードエージェントの操作」

ノードエージェントとは

ノードエージェントは、ドメイン管理サーバー (DAS) をホストするマシンを含む、サーバーインスタンスをホストするすべてのマシンに必要な軽量プロセスです。ノードエージェントは次の機能を実行します。

- ドメイン管理サーバーの指示により、サーバーインスタンスの起動、停止、作成、または削除を行います。
- 障害の発生したサーバーインスタンスを再起動します。
- 障害の発生したサーバーのログファイルを表示します。
- 各サーバーインスタンスのローカル設定リポジトリとドメイン管理サーバーの中央リポジトリを同期化します。各ローカルリポジトリには、そのサーバーインスタンスまたはノードエージェントに関する情報のみが含まれます。

次の図は、ノードエージェントの全体的なアーキテクチャーを示しています。



Application Server をインストールすると、マシンのホスト名を持つノードエージェントがデフォルトで作成されます。このノードエージェントは、実行する前に、ローカルマシン上で手動で起動する必要があります。

ノードエージェントを実行していない場合でも、サーバーインスタンスを作成および管理できます。ただし、ノードエージェントを使用してサーバーインスタンスを起動および停止するには、ノードエージェントが実行中である必要があります。

ノードエージェントは1つのドメインを処理します。マシンが複数のドメインで実行されるインスタンスをホストする場合は、複数のノードエージェントを実行する必要があります。

ノードエージェントの障害発生後のサーバーインスタンスの動作

ソフトウェアの障害またはその他のエラーによって、ノードエージェントが予期せず停止する場合があります。この状況では、ノードエージェントが管理していたすべてのサーバーインスタンスは管理されなくなります。ただし、そのようなサーバーインスタンスは稼働を続けており、DASからもアクセス可能なままです。それらのサーバーインスタンスについての情報はまだ Enterprise Server の管理インタフェースから入手可能であり、それらのサーバーインスタンスに配備されたアプリケーションへのアクセスもまだ可能です。

ノードエージェントを再起動しても、管理対象外となったサーバーインスタンスは管理されていない状態のままです。ノードエージェントはこれらのサーバーインスタンスの管理を再開しません。ソフトウェアの障害やその他のエラーによって、管理対象外のサーバーインスタンスが予期せず停止した場合、ノードエージェントはそのサーバーインスタンスを再起動できません。

管理対象外のサーバーインスタンスが稼働を続ける必要がある場合、ノードエージェントによるそのサーバーインスタンスの管理を再開することはできません。管理対象外となったサーバーインスタンスの管理を再開するただ1つの方法は、ノードエージェントを再起動したあとでサーバーインスタンスを停止し、再起動することです。

ノードエージェントの配備

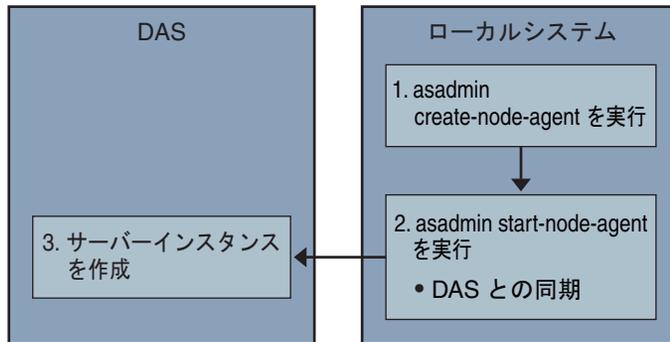
次の2とおりの方法で、ノードエージェントの設定および配備ができます。

- オンライン配備: 用いるトポロジがわかっている、すでにドメイン用のハードウェアが設置されている場合。
- オフライン配備: 完全な環境を設定する前に、ドメインとサーバーインスタンスを設定する場合。

▼ ノードエージェントをオンラインで配備する

すでにドメインのトポロジがわかっている、ドメイン用のハードウェアが設置されている場合は、オンライン配備を使用します。

次の図は、ノードエージェントのオンライン配備の概要を示しています。



始める前に ドメイン管理サーバーをインストールして起動します。ドメイン管理サーバーが起動し、実行中になったら、オンラインまたはオフライン配備を開始します。

- 1 サーバーインスタンスをホストするすべてのマシンにノードエージェントをインストールします。

インストーラまたは `asadmin create-node-agent` コマンドを使用します。マシンに複数のエージェントが必要な場合は、`asadmin create-node-agent` を使用してエージェントを作成します。

詳細については、[61 ページの「ノードエージェントの作成」](#)を参照してください。

- 2 `asadmin start-node-agent` コマンドを使用して、ノードエージェントを起動します。起動すると、ノードエージェントはドメイン管理サーバー (DAS) と通信します。それが DAS に到達すると、DAS にノードエージェントに対する設定が作成されます。設定が作成されると、管理コンソールでノードエージェントを表示できます。

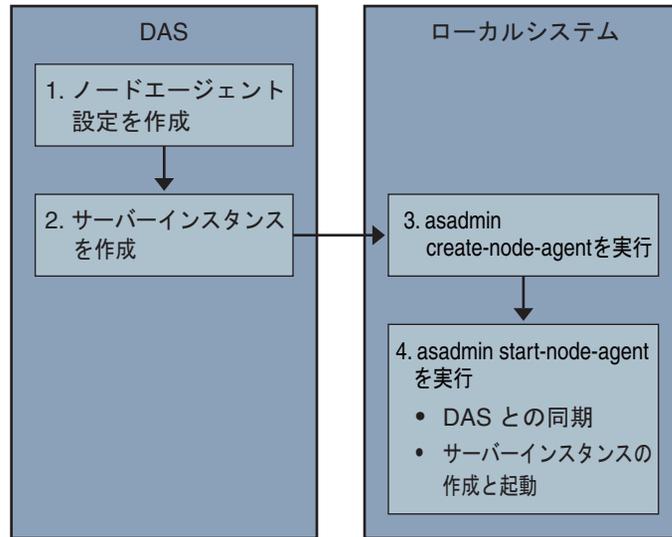
詳細については、[64 ページの「ノードエージェントの起動」](#)を参照してください。

- 3 ドメインを設定します。サーバーインスタンスを作成し、クラスタを作成して、アプリケーションを配備します。

▼ ノードエージェントをオフラインで配備する

個々のローカルマシンを設定する前に、オフライン配備を使用してドメイン内にノードエージェントを配備します。

次の図は、オフライン配備の概要を示しています。



始める前に ドメイン管理サーバーをインストールして起動します。ドメイン管理サーバーが起動し、実行中になったら、オンラインまたはオフライン配備を開始します。

- 1 ドメイン管理サーバーにプレースホルダノードエージェントを作成します。
詳細については、[61 ページの「ノードエージェントのプレースホルダを作成する」](#)を参照してください。
- 2 サーバーインスタンスとクラスタを作成して、アプリケーションを配備します。
サーバーインスタンスを作成するときは、まだ使用されていないポート番号を割り当てるようにしてください。設定がオフラインで実行されるため、作成時にはドメインでポートの競合をチェックすることができません。
- 3 サーバーインスタンスをホストするすべてのマシンにノードエージェントをインストールします。
インストーラまたは `asadmin create-node-agent` コマンドを使用します。ノードエージェントには、以前に作成したプレースホルダノードエージェントと同じ名前を付ける必要があります。
詳細については、[61 ページの「ノードエージェントの作成」](#)を参照してください。
- 4 `asadmin start-node-agent` コマンドを使用して、ノードエージェントを起動します。
ノードエージェントが起動すると、ドメイン管理サーバーにバインドされ、以前にノードエージェントに関連付けられたサーバーインスタンスを作成します。
詳細については、[64 ページの「ノードエージェントの起動」](#)を参照してください。

ノードエージェントとドメイン管理サーバーとの同期化

設定データは、ドメイン管理サーバーのリポジトリ(中央リポジトリ)に格納されると同時に、ノードエージェントのローカルマシンにもキャッシュされるため、これらの2つは同期化する必要があります。キャッシュの同期化は、常に管理ツールでの明示的なユーザーのアクションによって実行されます。

ここでは、次の内容について説明します。

- 54 ページの「ノードエージェントの同期化」
- 55 ページの「サーバーインスタンスの同期化」
- 56 ページの「ライブラリファイルの同期化」
- 57 ページの「固有の設定と設定管理」
- 58 ページの「大きなアプリケーションの同期化」

ノードエージェントの同期化

はじめてノードエージェントが起動すると、中央リポジトリの最新情報の要求をドメイン管理サーバー(DAS)に送信します。ノードエージェントがDASに正常に接続され、設定情報を取得すると、ノードエージェントはDASにバインドされます。

注-デフォルトでは、`asadmin start-node-agent` コマンドを使用すると、DAS と同期化せずに、リモートサーバーインスタンスが自動的に起動します。DAS によって管理されている中央リポジトリと同期化しているリモートサーバーインスタンスを起動する場合は、`asadmin start-node-agent` コマンドの `--startinstances=false` オプションを指定します。次に、`asadmin start-instance` コマンドを使用してリモートサーバーインスタンスを起動します。

DAS にプレースホルダノードエージェントを作成した場合、ノードエージェントがはじめて起動するときに、ノードエージェントはDASの中央リポジトリから設定を取得します。最初の起動時に、DAS が実行されていないため、ノードエージェントがDASに到達できない場合、ノードエージェントは停止し、バインドされないままの状態になります。

ドメインのノードエージェントの設定が変更された場合、ノードエージェントを実行するローカルマシンのノードエージェントと自動的に通信します。

DAS のノードエージェント設定を削除すると、次に同期するときにノードエージェント自体が停止し、削除待ちとしてマーク付けされます。ローカルの `asadmin delete-node-agent` コマンドを使用して、ノードエージェントを手動で削除します。

サーバーインスタンスの同期化

管理コンソールまたは `asadmin` ツールを使用してサーバーインスタンスを明示的に起動する場合、サーバーインスタンスは中央リポジトリと同期化されます。この同期が失敗すると、サーバーインスタンスは起動しません。

ノードエージェントが、管理コンソールまたは `asadmin` ツールによる明示的な要求なしにサーバーインスタンスを起動する場合、サーバーインスタンスのリポジトリキャッシュは同期しません。サーバーインスタンスは、キャッシュに格納された設定によって実行されます。リモートサーバーインスタンスのキャッシュ内にファイルを追加または削除してはいけません。

リモートサーバーインスタンスの設定は、キャッシュとして扱われ (`nodeagents/na1/server1` の下にあるすべてのファイル)、Application Server によって所有されます。極端な例を挙げれば、ユーザーがリモートサーバーインスタンスのすべてのファイルを削除し、ノードエージェントを再起動すると、リモートサーバーインスタンス (`server1` など) は再作成され、すべての必要なファイルは同期化されます。

次のファイルおよびディレクトリは Application Server によって同期が保たれます。

表 3-1 リモートサーバーインスタンス間で同期化されるファイルとディレクトリ

ファイルまたはディレクトリ	説明
<code>applications</code>	配備されているすべてのアプリケーション。このディレクトリ (およびサブディレクトリ) は、サーバーインスタンスから参照されるアプリケーションに基づいて同期化されます。ノードエージェントはどのアプリケーションも参照しないので、アプリケーションを同期化しません。
<code>config</code>	ドメイン全体に対する設定ファイルを格納します。実行時の一時ファイル (<code>admch</code> 、 <code>admsn</code> 、 <code>secure.seed</code> 、 <code>.timestamp</code> 、 <code>__timer_service_shutdown__.dat</code> など) を除いたこのディレクトリ内のすべてのファイルは、同期化されます。
<code>config/config_name</code>	<code>config_name</code> という名前の設定を使用してすべてのインスタンスによって共有されるファイルを格納するためのディレクトリ。domain.xml で定義されるすべての設定に対して、このようなディレクトリが1つずつ存在することになります。このディレクトリ内のすべてのファイルが、 <code>config_name</code> を使用しているサーバーインスタンスと同期化されます。

表 3-1 リモートサーバーインスタンス間で同期化されるファイルとディレクトリ (続き)

ファイルまたはディレクトリ	説明
<code>config/config_name/lib/ext</code>	Java 拡張クラスを (zip または jar アーカイブとして) 置くことができるフォルダ。これは、 <code>config_name</code> という名前の設定を使用してサーバーインスタンスに配備されたアプリケーションによって使用されます。これらの jar ファイルは、Java 拡張メカニズムを使用してロードされます。
<code>docroot</code>	HTTP ドキュメントルート。既定の設定では、ドメイン内のすべてのサーバーインスタンスが同じ <code>docroot</code> を使用します。それらのサーバーインスタンスに異なる <code>docroot</code> を使用させるためには、仮想サーバーの <code>docroot</code> プロパティを設定する必要があります。
<code>generated</code>	Java EE アプリケーションやモジュール用に生成されたファイル。たとえば、EJB スタブ、コンパイル済みの JSP クラス、セキュリティポリシーファイル。このディレクトリは、 <code>applications</code> ディレクトリと同様に同期化されます。したがって、サーバーインスタンスによって参照されるアプリケーションに対応するディレクトリのみが同期化されます。
<code>lib, lib/classes</code>	ドメイン全体に配備されたアプリケーションに使用される共通の Java クラスファイルまたは jar および zip アーカイブを置くことができるフォルダ。これらのクラスは、Application Server のクラスローダーを使用してロードされます。クラスローダーでのロード順序は次のとおりです。 <code>lib/classes</code> 、 <code>lib/*.jar</code> 、 <code>lib/*.zip</code>
<code>lib/ext</code>	ドメイン全体に配備されたアプリケーションによって使用される Java 拡張クラスを (jar または zip アーカイブとして) 置くことができるフォルダ。これらの jar ファイルは、Java 拡張メカニズムを使用してロードされます。
<code>lib/applibs</code>	依存する jar ファイルを <code>domains/<domain_name>lib/applibs</code> の下に配置し、 <code>libraries</code> オプションを使用して jar ファイルへの相対パスを指定します。 たとえば、 <code>asadmin deploy --libraries commons-coll.jar,X1.jar foo.ear</code> のようにします。
<code>java-web-start</code>	このディレクトリ (およびサブディレクトリ) の各部分が、サーバーインスタンスから参照されるアプリケーションに基づいて同期化されません。

ライブラリファイルの同期化

アプリケーションの `--libraries` 配備時属性を使用して、アプリケーションの実行時の依存関係を指定することができます。相対パス (jar の名前のみ) を指定すると、Application Server は指定したライブラリを `domain-dir /lib/applibs` 内で見つけようとします。

ライブラリをドメイン全体で使用できるようにするには、JAR ファイルを `domain-dir/lib` または `domain-dir/lib/classes` に配置することができます。(詳細については、『[Sun GlassFish Enterprise Server 2.1 Developer's Guide](#)』の「[Using the Common Class Loader](#)」を参照してください。)通常この方法は、JDBC ドライバや、ドメイン内のすべてのアプリケーションによって共有されているその他のユーティリティライブラリに対してあてはまりません。

クラスタ全体またはスタンドアロンのサーバー全体で使用する場合は、jar ファイルを `domain-dir/domain1/config/xyz-config/lib` ディレクトリにコピーします。次に、それらの jar ファイルを、`xyz-config` の `classpath-suffix` または `classpath-prefix` 要素に追加します。これによって、`xyz-config` を使用するすべてのサーバーインスタンスの jar ファイルが同期化されます。

要約すると、次のようになります。

- `domains/domain1/lib` - ドメイン全体スコープ、共通のクラスローダー、jar ファイルを自動的に追加。
- `domains/domain1/config/cluster1,config/lib` - 設定全体、`classpath-prefix` または `classpath-suffix` を更新。
- `domains/domain1/lib/applibs` - アプリケーションスコープ、アプリケーションのクラスローダーに自動的に追加。
- `domains/domain1/config/cluster1,config/lib/ext` - <http://java.sun.com/j2se/1.5.0/docs/guide/extensions/extensions.html> に自動的に追加。

固有の設定と設定管理

設定ファイル (`domains/domain1/config` の下) は、ドメイン全体にわたって同期化されます。スタンドアロンのサーバーインスタンス (`server1`) によって使用される `server1-config` 用に `server.policy` をカスタマイズする場合は、変更後の `server.policy` ファイルを `domains/domain1/config/server1-config` ディレクトリの下に配置します。

変更済みの `server.policy` ファイルは、スタンドアロンのサーバーインスタンス `server1` とのみ同期化されます。 `jvm-option` を更新することも忘れないでください。次に例を示します。

```
<java-config>
...
<jvm-options>-Djava.security.policy=${com.sun.aas.instanceRoot}/config
/server1-config/server.policy</jvm-options>
</java-config>
```

大きなアプリケーションの同期化

同期化の必要な大きなアプリケーションが使用環境に含まれる場合、または使用できるメモリーが制限されている場合は、JVM オプションを調整してメモリーの使用を制限できます。この調整によって、メモリー不足によるエラーを受信する可能性は低くなります。インスタンス同期化 JVM ではデフォルトの設定が使用されますが、JVM オプションを設定してそれらを変更することもできます。

INSTANCE-SYNC-JVM-OPTIONS プロパティを使用して、JVM オプションを設定します。このプロパティを設定するコマンドは次のとおりです。

```
asadmin set
domain.node-agent.node_agent_name.property.INSTANCE-SYNC-JVM-OPTIONS="JVM_options"
```

次に例を示します。

```
asadmin set
domain.node-agent.node0.property.INSTANCE-SYNC-JVM-OPTIONS="-Xmx32m -Xss2m"
```

この例では、ノードエージェントは node0、JVM オプションは -Xmx32m -Xss2m です。

詳細は、<http://java.sun.com/docs/hotspot/VMOptions.html> を参照してください。

注-ノードエージェントの設定にプロパティが追加されたり変更されてもノードエージェントは自動的に同期化されないため、INSTANCE-SYNC-JVM-OPTIONS プロパティの変更後、ノードエージェントを再起動してください。

doNotRemoveList フラグの使用

Application Server によって同期化されたディレクトリ (applications、generated、docroot、config、lib、java-web-start) 内のファイルを、アプリケーションによって保存して読み取る必要がある場合は、doNotRemoveList フラグを使用します。この属性は、ファイルまたはディレクトリのコンマ区切りのリストをとります。アプリケーション依存ファイルは、DAS によって管理される中央リポジトリに存在していない場合でも、サーバーの起動時には削除されません。中央リポジトリに同じファイルが存在する場合は、同期化中に上書きされます。

INSTANCE-SYNC-JVM-OPTIONS プロパティを使用して、doNotRemoveList 属性に渡します。

次に例を示します。

```
<node-agent name="na1" ...>
```

```
...
```

```
<property name="INSTANCE-SYNC-JVM-OPTIONS"
value="-Dcom.sun.appserv.doNotRemoveList=applications/j2ee-modules
/<webapp_context>/logs,generated/mylogdir"/>

</node-agent>
```

ノードエージェントログの表示

各ノードエージェントには、固有のログファイルがあります。ノードエージェント関連の問題がある場合、次の場所にあるログファイルを参照します。

`node_agent_dir/node_agent_name/agent/logs/server.log` です。

ノードエージェントログにより、サーバーのログを参照して問題に関する詳細なメッセージを調べるように指示される場合もあります。

サーバーログの場所は以下のとおりです。

`node_agent_dir/node_agent_name/server_name/logs/server.log`

`node_agent_dir` のデフォルトの位置は `install_dir/nodeagents` です。

ノードエージェントの操作

- [59 ページの「ノードエージェントタスクの実行方法」](#)
- [60 ページの「ノードエージェントのプレースホルダ」](#)
- [61 ページの「ノードエージェントのプレースホルダを作成する」](#)
- [61 ページの「ノードエージェントの作成」](#)
- [64 ページの「ノードエージェントの起動」](#)
- [64 ページの「ノードエージェントの停止」](#)
- [65 ページの「ノードエージェントの削除」](#)
- [65 ページの「ノードエージェントの一般情報を表示する」](#)
- [66 ページの「ノードエージェントの設定を削除する」](#)
- [67 ページの「ノードエージェントの設定を編集する」](#)
- [67 ページの「ノードエージェントのレルムを編集する」](#)
- [68 ページの「ノードエージェントの JMX 対応リスナーを編集するには」](#)
- [69 ページの「スタンドアロンのサーバーインスタンスを作成する」](#)

ノードエージェントタスクの実行方法

一部のノードエージェントタスクについては、ノードエージェントを実行するシステムでローカルに `asadmin` ツールを使用する必要があります。その他のタスクは、管理コンソールまたは `asadmin` を使用してリモートで実行できます。

次の表は、タスクとそれを実行する場所の概要です。

表 3-2 ノードエージェントタスクの実行方法

作業	管理コンソール	asadmin コマンド
ノードエージェントのプレースホルダをドメイン管理サーバーに作成します。	「新しいノードエージェントのプレースホルダ」 ページ。	create-node-agent-config
ノードエージェントを作成します。	使用不可	create-node-agent
ノードエージェントを起動します。	使用不可	start-node-agent
ノードエージェントを停止します。	使用不可	stop-node-agent
ドメイン管理サーバーからノードエージェント設定を削除します。	「ノードエージェント」 ページ。	delete-node-agent-config
ローカルマシンからノードエージェントを削除します。	使用不可	delete-node-agent
ノードエージェント設定を編集します。	「ノードエージェント」 ページ。	set
ノードエージェントを一覧表示します。	「ノードエージェント」 ページ。	list-node-agents

ノードエージェントのプレースホルダ

既存のノードエージェントが存在しなくても、ノードエージェントのプレースホルダを使用して、サーバーインスタンスを作成および削除することができます。ノードエージェントのプレースホルダは、ノードエージェント自体がノードエージェントのローカルシステムに作成される前に、ドメイン管理サーバー (DAS) 上に作成されます。

ノードエージェントのプレースホルダの作成については、[61 ページの「ノードエージェントのプレースホルダを作成する」](#)を参照してください。

注-プレースホルダノードエージェントを作成すると、それを使用してドメインにインスタンスを作成できます。ただし、インスタンスを起動する前に、asadmin コマンドを使用して、インスタンスが配置されるマシン上に実際のノードエージェントをローカルに作成し、起動する必要があります。詳細については、[61 ページの「ノードエージェントの作成」](#)および[64 ページの「ノードエージェントの起動」](#)を参照してください。

▼ ノードエージェントのプレースホルダを作成する

ノードエージェントは、リモートマシンで実行されているサーバーインスタンスのローカルウォッチドッグです。このため、ノードエージェントはサーバーインスタンスをホストしているマシン上に作成する必要があります。この要件の結果、管理コンソールを使用して、ノードエージェントのプレースホルダのみを作成できません。このプレースホルダは、ノードエージェントが存在しない場合のノードエージェントの設定です。

プレースホルダを作成したら、ノードエージェントをホスティングするマシン上で、`asadmin` コマンドの `create-node-agent` を使用して作成を完了します。詳細については、[61 ページの「ノードエージェントの作成」](#) を参照してください。

ノードエージェントを作成および使用するために必要な手順のリストについては、[51 ページの「ノードエージェントの配備」](#) を参照してください。

- 1 ツリーコンポーネントで、「ノードエージェント」ノードを選択します。
- 2 「ノードエージェント」ページで、「新規」をクリックします。
- 3 「新しいノードエージェントのプレースホルダ」ページで、新規ノードエージェントの名前を入力します。

名前は、ドメインのすべてのノードエージェント名、サーバーインスタンス名、クラスタ名、および設定名の間で一意である必要があります。

- 4 「了解」をクリックします。
新規ノードエージェントのプレースホルダが「ノードエージェント」ページにリスト表示されます。

参考 同機能を持つ `asadmin` コマンド

```
create-node-agent-config
```

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、ノードエージェントを実行するマシンで、`asadmin` コマンドの `create-node-agent` をローカルに実行します。

ノードエージェントのデフォルト名は、ノードエージェントを作成するホストの名前です。

ノードエージェントのプレースホルダをすでに作成している場合は、ノードエージェントプレースホルダと同じ名前を使用して、関連したノードエージェント

を作成します。ノードエージェントのプレースホルダをまだ作成しておらず、DAS が起動していて到達可能である場合、`create-node-agent` コマンドは DAS 上にノードエージェント設定(プレースホルダ)も作成します。

コマンド構文の詳しい説明については、コマンドに関するオンラインヘルプを参照してください。

セキュリティ保護された通信を行うように、DAS およびノードエージェントが設定される場合があります。この状況では、ノードエージェントが起動されるとき、DAS がノードエージェントに送信する証明書をノードエージェントの側で検証する必要があります。証明書を検証するために、ノードエージェントはそのローカルトラストストアから証明書を検索します。このトラストストアはマスターパスワードによって保護されています。ユーザーにパスワードの入力を求めることなくノードエージェントを起動できるようにするには、ノードエージェントの作成時に、ノードエージェントのマスターパスワードをファイルに保存します。ノードエージェントのマスターパスワードをファイルに保存しない場合、ユーザーはノードエージェントを起動するたびにマスターパスワードの入力を求められます。

注-状況によっては、DNS 経由で到達可能なホストの名前を指定する必要があります。詳細については、[63 ページの「DNS に到達可能なホストに対してノードエージェントを作成する」](#)を参照してください。

▼ ノードエージェントを作成する

- 次のコマンドを入力します。

```
asadmin create-node-agent --host das-host --port port-no --user das-user  
[--savemasterpassword=true] nodeagent
```

ユーザーにパスワードの入力を求めることなくノードエージェントを起動できるようにするには、ノードエージェントのマスターパスワードをファイルに保存します。ノードエージェントのマスターパスワードをファイルに保存するには、ノードエージェントを作成するためのコマンドで、`--savemasterpassword` オプションを `true` に設定します。

`--savemasterpassword` を `true` に設定すると、マスターパスワードの入力を求められます。それ以外の場合、パスワードの入力を求められることはありません。

`--host das-host` ドメイン管理サーバー (DAS) が稼働しているホストの名前を指定します。

`-port port-no` ドメインを管理するための HTTP/HTTPS ポート番号を指定します。

`--user das-user` DAS ユーザーを指定します。

`nodeagent` 作成するノードエージェントの名前を指定します。この名前はドメイン内で一意である必要があります。

例 3-1 ノードエージェントの作成

```
asadmin create-node-agent --host myhost --port 4848 --user admin nodeagent1
```

このコマンドは、`nodeagent1` という名前のノードエージェントを作成します。ノードエージェントが通信する DAS は、マシン `myhost` 上で稼働しています。エージェントのドメインを管理するための HTTP ポートは 4848 です。DAS ユーザーの名前は `admin` です。

▼ DNS に到達可能なホストに対してノードエージェントを作成する 次の状況では、DAS が稼働しているホストが DNS 経由で到達可能である必要があります。

- ドメインがサブネット境界にまたがっている。すなわち、ノードエージェントと DAS が異なるドメイン (例: `sun.com` と `java.com`) 内にある。
- ホスト名が DNS に登録されていない DHCP マシンが使用されている。

- 1 ドメインを作成するための `create-domain` コマンドで、`--domainproperties domain.hostName=das-host-name` オプションを指定します。
`das-host-name` は、DAS が稼働しているマシンの名前です。
- 2 ノードエージェントを作成するための `create-node-agent` コマンドで、次のオプションを指定します。
 - `--host das-host-name`。`das-host-name` は、手順 1 で指定した DAS ホスト名です。このオプションは、ファイル `as-install/nodeagents/nodeagentname/agent/config/das.properties` 内の `agent.das.host` プロパティに対応します。
 - `--agentproperties remoteclientaddress=node-agent-host-name`。`node-agent-host-name` は、DAS がノードエージェントへの接続に使用するホスト名です。このオプションは、ファイル `as-install/nodeagents/nodeagentname/agent/config/nodeagent.properties` 内の `agent.client.host` プロパティに対応します。

参考 hosts ファイルの更新によるホストの指定

別の解決法は、プラットフォームに特定のホスト名およびアドレス解決を定義する、`hosts` ファイルを更新し、ホスト名を正しい IP アドレスに解決することです。ただし、DHCP 使用して再接続する時に、異なる IP アドレスを割り当てられる可能性があります。その場合、各サーバーでホスト解決ファイルを更新する必要があります。

ノードエージェントの起動

ノードエージェントがサーバーインスタンスを管理できるようにするには、ノードエージェントを実行している必要があります。ノードエージェントを起動するには、ノードエージェントが存在するシステムで `asadmin` コマンドの `start-node-agent` をローカルに実行します。

コマンド構文の詳しい説明については、コマンドに関するオンラインヘルプを参照してください。

次に例を示します。

```
asadmin start-node-agent --user admin --startinstances=false nodeagent1
```

ここで、`admin` は管理ユーザーであり、`nodeagent1` は起動しているノードエージェントです。

デフォルトでは、ノードエージェントインスタンスのキャッシュリポジトリは、ノードエージェントの再起動時に中央リポジトリから同期されません。インスタンスのキャッシュリポジトリを中央リポジトリと強制的に同期するには、`asadmin start-node-agent` コマンドで `--syncinstances` オプションを `true` に設定します。

注 `--syncinstances` オプションを `true` に設定すると、すべてのインスタンスのリポジトリがノードエージェントの再起動時に同期されます。

ノードエージェントを再起動したあとで、`asadmin start-instance` コマンドを使用してサーバーインスタンスを起動します。

ノードエージェントの停止

実行中のノードエージェントを停止するには、ノードエージェントが存在するシステムで、`asadmin` コマンドの `stop-node-agent` を実行します。`stop-node-agent` は、ノードエージェントが管理するすべてのサーバーインスタンスを停止します。

コマンド構文の詳しい説明については、コマンドに関するオンラインヘルプを参照してください。

次に例を示します。

```
asadmin stop-node-agent nodeagent1
```

ここで、`nodeagent1` はノードエージェントの名前です。

ノードエージェントの削除

ノードエージェントを削除する前に、ノードエージェントを停止する必要があります。ノードエージェントが起動しない場合、またはドメイン管理サーバーに正常に接続できない(バインドされない)場合も、ノードエージェントを削除できます。

ノードエージェントのファイルを削除するには、ノードエージェントが存在するシステムで、`asadmin delete-node-agent` を実行します。

コマンド構文の詳細な説明については、コマンドに関するオンラインヘルプを参照してください。

次に例を示します。

```
asadmin delete-node-agent nodeagent1
```

ここで、`nodeagent1` はノードエージェントです。

ノードエージェントを削除する場合は、管理コンソールまたは `asadmin delete-node-agent-config` コマンドのいずれかを使用して、ドメイン管理サーバーからノードエージェントの設定も削除する必要があります。

▼ ノードエージェントの一般情報を表示する

- 1 ツリーコンポーネントで、「ノードエージェント」ノードを選択します。
- 2 ノードエージェントの名前をクリックします。
ノードエージェントがすでに存在するのに、ここに表示されない場合は、ノードエージェントのホストマシンで、`asadmin start-node-agent` を使用して、ノードエージェントを起動します。[64 ページの「ノードエージェントの起動」](#)を参照してください。
- 3 ノードエージェントのホスト名をチェックします。
ホスト名が「不明なホスト」の場合、ノードエージェントはドメイン管理サーバー(DAS)と初期接続をしていません。
- 4 ノードエージェントの状態をチェックします。
この状態は次のいずれかです
 - 稼動中: ノードエージェントが正常に作成され、現在実行中です。
 - 停止中: 「ノードエージェントはローカルマシンで作成されているが、起動していない」、または「ノードエージェントは起動したが、その後停止した」のどちらかです。

- ランデブーを待機しています: ノードエージェントは、ローカルマシンで作成されていないプレースホルダです。

詳細については、[61 ページの「ノードエージェントの作成」](#) および [64 ページの「ノードエージェントの起動」](#) を参照してください。

- 5 起動時にインスタンスを起動するかどうかを選択します。
ノードエージェントが起動するときに、ノードエージェントに関連するサーバーインスタンスが自動的に起動するようにするには「Yes」を選択します。インスタンスを手動で起動するには、「No」を選択します。
- 6 ノードエージェントがドメイン管理サーバーと接続したかどうかを確認します。
ノードエージェントがドメイン管理サーバーと接続していない場合、正常に起動していません。
- 7 ノードエージェントに関連するサーバーインスタンスを管理します。
ノードエージェントが実行中の場合、インスタンス名の横にあるチェックボックスをクリックし、「起動」または「停止」をクリックしてインスタンスを起動または停止します。

▼ ノードエージェントの設定を削除する

管理コンソールを使用すると、ドメインからノードエージェントの設定を削除することができます。設定は削除できますが、実際のノードエージェントは削除できません。ノードエージェント自体を削除するには、ノードエージェントのローカルマシンで `asadmin` コマンドの `delete-node-agent` を実行します。詳細については、[65 ページの「ノードエージェントの削除」](#) を参照してください。

ノードエージェントの設定を削除する前に、ノードエージェントの実行を停止し、関連するインスタンスを削除する必要があります。ノードエージェントを停止するには、`asadmin` コマンドの `stop-node-agent` を使用します。詳細については、[64 ページの「ノードエージェントの停止」](#) を参照してください。

- 1 ツリーコンポーネントで、「ノードエージェント」ノードを選択します。
- 2 「ノードエージェント」ページで、削除するノードエージェントの横にあるチェックボックスを選択します。
- 3 「削除」をクリックします。

参考 同機能を持つ asadmin コマンド

```
delete-node-agent-config
```

▼ ノードエージェントの設定を編集する

- 1 ツリーコンポーネントで、「ノードエージェント」ノードを開きます。
- 2 編集するノードエージェントの設定を選択します。
- 3 「起動時にインスタンスを起動」を「Yes」に設定し、エージェントの起動時にエージェントのサーバーインスタンスが起動されるようにします。
このページから、手動でのインスタンスの起動または停止もできます。

この設定がプレースホルダノードエージェント用である場合は、`asadmin create-node-agent` を使用して実際のノードエージェントを作成するときに、この設定が引き継がれます。ノードエージェントの作成については、[61 ページの「ノードエージェントの作成」](#)を参照してください。

この設定が既存のノードエージェント用である場合、ノードエージェントの設定情報が自動的に同期されます。

▼ ノードエージェントのレルムを編集する

ノードエージェントに接続しているユーザーの認証レルムを設定する必要があります。管理ユーザーだけがノードエージェントにアクセスできます。

- 1 ツリーコンポーネントで、「ノードエージェント」ノードを開きます。
- 2 編集するノードエージェントの設定を選択します。
- 3 「認証レルム」タブをクリックします。

- 4 「レルムの編集」ページで、レルムを入力します。

デフォルトは、ノードエージェントの作成時に作成された `admin-realm` です。別のレルムを使用するには、ドメインによって制御されるすべてのコンポーネントまたは正常に通信しないコンポーネントのレルムを置き換えます。

- 5 「クラス名」フィールドで、レルムを実装する **Java** クラスを指定します。

- 6 必要なプロパティを追加します。

認証レルムは、特定の実装によって必要とするものが異なるプロバイダ固有のプロパティが必要です。

▼ ノードエージェントのJMX対応リスナーを編集するには

ノードエージェントは、JMXを使用してドメイン管理サーバーと通信します。このため、JMX要求を待機するポートとその他のリスナー情報が必要です。

- 1 ツリーコンポーネントで、「ノードエージェント」ノードを開きます。
- 2 編集するノードエージェントの設定を選択します。
- 3 「JMX」タブをクリックします。
- 4 「アドレス」フィールドに、IPアドレスまたはホスト名を入力します。
リスナーが一意のポート値を使用して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。
- 5 「ポート」フィールドに、ノードエージェントのJMXコネクタが待機するポート番号を入力します。
IPアドレスが「0.0.0.0」の場合、ポート番号は一意のものである必要があります。
- 6 「JMXプロトコル」フィールドで、JMXコネクタがサポートするプロトコルを入力します。
デフォルトはrmi_jrmpです。
- 7 「すべてのアドレスを受け付ける」の横にあるチェックボックスをクリックして、すべてのIPアドレスに接続できるようにします。
ノードエージェントは、ネットワークカードに関連付けられた特定のIPアドレスを待機するか、またはすべてのIPアドレスを待機します。すべてのアドレスを許可すると、「待機するホストアドレス」プロパティに値「0.0.0.0」が設定されます。
- 8 「レルム名」フィールドで、リスナーの認証を処理するレルムの名前を入力します。
このページの「セキュリティ」セクションで、リスナーがSSL、TLS、あるいはこの両方のセキュリティーを使用するように設定します。
安全なリスナーを設定するには、次の手順を実行します。

- 9 「セキュリティ」フィールドの「有効」ボックスにチェックマークを付けます。
デフォルトで、セキュリティが有効になります。
- 10 クライアント認証を設定します。
このリスナーを使っている個々のクライアントにサーバーへの認証を要求する場合は、「クライアント認証」フィールドの「有効」ボックスにチェックマークを付けます。
- 11 証明書のニックネームを入力します。
「証明書のニックネーム」フィールドに、既存サーバーの鍵ペアと証明書の名前を入力します。
証明書およびSSLの操作の詳細については、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。
- 12 SSL3/TLS セクションでは次の手順を実行します。
 - a. リスナーで有効にするセキュリティプロトコルにチェックマークを付けます。
SSL3とTLSのどちらか、または両方のプロトコルにチェックマークを付ける必要があります。
 - b. プロトコルが使用する暗号化方式群にチェックマークを付けます。
すべての暗号化方式群を有効にするには、「サポートされるすべての暗号化方式群」にチェックマークを付けます。
- 13 「保存」をクリックします。

▼ スタンドアロンのサーバーインスタンスを作成する

- 1 ツリーコンポーネントで、「スタンドアロンインスタンス」ノードを選択します。
- 2 「新規」をクリックします。
- 3 「新しいスタンドアロンサーバーインスタンス」ページで、インスタンスの名前を入力します。
- 4 このインスタンスが作成されるノードエージェントを選択し、「OK」をクリックします。

HTTP 負荷分散の設定

この章では、HTTP ロードバランサプラグインについて説明します。ここで説明する内容は次のとおりです。

- 72 ページの「HTTP ロードバランサの動作」
- 74 ページの「HTTP 負荷分散の設定」
- 79 ページの「HTTP ロードバランサの設定」
- 94 ページの「複数の Web サーバーインスタンスの設定」
- 95 ページの「可用性を低下させないアプリケーションのアップグレード」
- 102 ページの「HTTP ロードバランサプラグインの監視」

ロードバランサは、Sun Java Web Server 6.1 および 7.0、Apache 2.0.x および 2.2.x、IIS Server (32 ビットモードのみ) で動作します。サポートされる Web サーバーの一覧については、『Sun GlassFish Enterprise Server 2.1 リリースノート』の「Web サーバー」を参照してください。

その他の負荷分散については、第 8 章「Java Message Service 負荷分散とフェイルオーバー」および第 9 章「RMI-IIOP 負荷分散とフェイルオーバー」を参照してください。

ここでは、Enterprise Server に付属している HTTP ロードバランサプラグインの使用方法について説明します。もう 1 つの HTTP 負荷分散オプションは、Enterprise Server で Sun Secure Application Switch を使用し、ハードウェアベースの負荷分散ソリューションを構築するというものです。このソリューションの設定については、Clustering and Securing Web Applications: A Tutorial (<http://developers.sun.com/prodtech/appserver/reference/techart/load-balancing.html>) を参照してください。

HTTP ロードバランサの動作

ロードバランサの目的は、スタンドアロンまたはクラスタ化された複数のインスタンスの間でワークロードを均等に分散させ、それにより、システムの全体的なスループットを向上させることです。

HTTP ロードバランサにより、Java EE アプリケーションサーバーに配備されるサービスの高可用性を実現できます。負荷分散処理の間、それまでサービスを提供していたインスタンスが稼働していないか、または正常な状態でないために要求を処理できないことが検出された場合、HTTP ロードバランサはセッション要求を別のサーバーインスタンスにフェイルオーバーします。HTTP セッション情報を持続させるには、クラスタプロファイルを使用していることと、インメモリーレプリケーションまたは HADB を使用するように HTTP セッション持続性を設定することが必要です。詳細については、[第7章「高可用性 \(HA\) セッション持続性とフェイルオーバーの設定」](#)を参照してください。

注-ロードバランサは、8K バイトを超える URI や URL を処理しません。

HTTP 負荷分散アルゴリズム

デフォルトでは、HTTP ロードバランサは「スティッキラウンドロビンアルゴリズム」を使用して、着信 HTTP および HTTPS 要求を負荷分散します。

新しい HTTP 要求がロードバランサプラグインに送信されると、単純なラウンドロビンスキーマに基づいて、アプリケーションサーバーインスタンスに転送されます。要求がセッションベースのアプリケーションに対するものである場合、これには新しいセッションに対する要求も含まれます。同じセッションベースのアプリケーションに対する同じクライアントからの後続の要求は、割り当て済み要求 (スティッキ要求) と見なされ、ロードバランサによって同じインスタンスにルーティングされます。スティッキ (sticky: 粘着性の) ラウンドロビンという名前が付いているのはそのような理由からです。セッションベースでないアプリケーションへの要求や、セッションベースのアプリケーションに対する最初の要求は未割り当て要求と呼ばれます。スティッキー性は Cookie を使用して、または明示的 URL 書き換えによって実現されます。ロードバランサは、スティッキ度を判断する方法を自動的に決定します。

ロードバランサプラグインは次の方法を使ってセッションのスティッキ度を判断します。

- **cookie** の使用: ロードバランサプラグインは、個別の cookie を使用してルート情報を記録します。Cookie に基づいた方法を使用するには、HTTP クライアント (通常は Web ブラウザ) が Cookie をサポートしている必要があります。HTTP クライアントが Cookie を受け入れることができない場合、プラグインは次の方法を使用します。
- 明示的な **URL** 書き換え: スティッキ情報が URL に追加されます。この方法は、HTTP クライアントが Cookie をサポートしない場合でも機能します。

スティッキ情報から、ロードバランサプラグインは、まず、以前に要求が転送されたインスタンスを判断します。そのインスタンスが正常であるとわかると、ロードバランサプラグインは、要求をその特定のインスタンスに転送します。したがって、特定のセッションに対するすべての要求が同じインスタンスに送信されます。

HTTP ロードバランサプラグインの機能

ロードバランサの次の機能により、柔軟性と使いやすさが向上します。

自動適用

Enterprise Server では、管理コンソールで行なったロードバランサ設定の変更を、ネットワーク経由で自動的に Web サーバーの構成ディレクトリに送信できます。Enterprise Server の以前のバージョンでは、ロードバランサ設定をエクスポートしてから、Web サーバーの構成ディレクトリにコピーする必要がありました。

重み付きラウンドロビン

ロードバランサでは、HTTP 要求の配信の改良を実現しています。管理者は「重み」と呼ばれる属性を使用して、重みに比例した形で要求をインスタンスにルーティングする方式を指定できます。たとえば、クラスタに2つのインスタンスがあり、管理者はインスタンス x に対して 100、インスタンス y に対して 400 の重みをそれぞれ設定しているとします。この場合、100 の要求があると、20 の要求がインスタンス x に、80 の要求がインスタンス y に送られます。

ユーザー-定義の負荷分散

Enterprise Server では、HTTP 要求の分散に関するカスタムポリシーを管理者が定義できます。カスタムポリシーでは、ロードバランサプラグインが使用しなければならない負荷分散アルゴリズムを定義します。言い換えると、どのインスタンスが HTTP 要求を処理するかを管理者が定義できます。この機能を使用するには、管理者は共有ライブラリを作成する必要があります。共有ライブラリでは、loadbalancer.h

というインタフェースを実装する必要があります。たとえば、共有ライブラリを使用すると、受け取った着信要求のヘッダーを評価し、その要求を処理できるインスタンスを何らかの基準に従って選択することができます。この共有ライブラリはロードバランサによって読み込まれます。

インタフェース `loadbalancer.h` は、`webserver_install_dir/plugins/lib/install` にあります。Enterprise Server には、共有ライブラリの `roundrobin.c` も付属しています。管理者はこのライブラリをテンプレートに使用して、共有ライブラリを構築できます。

HTTP 負荷分散の設定

この節では、ロードバランサプラグインを設定する方法について説明します。次の項目が含まれています。

- 74 ページの「HTTP 負荷分散を設定するための前提条件」
- 75 ページの「HTTP 負荷分散を設定するための手順」
- 78 ページの「HTTP ロードバランサの配備」

HTTP 負荷分散を設定するための前提条件

ロードバランサを設定する前に、次の手順を実行する必要があります。

- サポートされている Web サーバーをインストールし、必要な設定を行います。サポートされている Web サーバーの設定については、第 5 章「HTTP 負荷分散のための Web サーバーの設定」を参照してください。
- ロードバランサプラグインをインストールします。
インストールの手順については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』を参照してください。
- 負荷分散に参加する Enterprise Server クラスタまたはサーバーインスタンスを作成します。
- これらのクラスタまたはインスタンスに対してアプリケーションを配備します。

注- インスタンスとロードバランサが異なるネットワークドメインにインストールされる配備状況では、ノードエージェントの作成時に、オプション `--agentproperties` を使用して完全指定ドメイン名を指定する必要があります。たとえば、`asadmin create-node-agent --agentproperties remoteclientaddress=myhost.country.example.com test-na` のようになります。

HTTP 負荷分散を設定するための手順

ユーザーの環境で負荷分散を設定するには、管理コンソールの GUI または `asadmin` ツールを使用します。以降の節では、さらに詳しい情報を示します。

▼ 管理コンソールを使用して負荷分散を設定する

- 1 ロードバランサ設定を作成します。

管理コンソールでは、左側の区画で「HTTP ロードバランサ」をクリックし、「新規」をクリックします。「新しい HTTP ロードバランサ」ページで、デバイスの詳細を指定し、ターゲットのクラスタまたはインスタンスも選択します。

- 2 ロードバランサが管理するクラスタまたはスタンドアロンサーバーインスタンスへの参照を追加します。

管理コンソールを使用してこれを行うには、左側の区画で「HTTP ロードバランサ」ノードをクリックし、ノードの下に表示されるリストから目的のロードバランサをクリックします。「ターゲット」タブを開き、「ターゲットを管理」をクリックします。「ロードバランサのターゲットを管理」ページで、必要なターゲットを選択します。

ターゲットを指定してロードバランサ設定を作成しており、そのターゲットが、ロードバランサが参照する唯一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である場合は、この手順を飛ばしてください。

- 3 ロードバランサによって参照されるクラスタまたはスタンドアロンサーバーインスタンスを有効にします。

管理コンソールを使用してスタンドアロンサーバーインスタンスを有効にするには、左側の区画で「HTTP ロードバランサ」ノードをクリックし、ノードの下に表示されるリストから目的のロードバランサをクリックします。「ターゲット」タブを開き、「ターゲット」テーブルで、有効にするインスタンスの隣のチェックボックスにチェックマークを付け、「有効」をクリックします。

クラスタ内のサーバーインスタンスを有効にするには、上で説明した手順でロードバランサを選択し、「ターゲット」タブで目的のクラスタをクリックします。次に、「インスタンス」タブを開き、目的のインスタンスを選択し、「ロードバランサの操作」ドロップダウンリストから、「負荷分散の有効化」を選択します。

クラスタまたはスタンドアロンインスタンスを有効にするための同様のコマンドは、`asadmin enable-http-lb-server` です。

- 4 アプリケーションの負荷分散を有効にします。

これを管理コンソールで行うには、上で説明した手順で「ターゲット」タブを開き、必要なクラスタをクリックします。ここで「アプリケーション」タブを開

き、必要なアプリケーションを選択し、「その他の操作」ドロップダウンリストから、「ロードバランサ有効」を選択します。

これらのアプリケーションは、ロードバランサが参照するクラスタまたはスタンドアロンインスタンスで使用するために、事前に配備および有効にしておく必要があります。負荷分散のためにアプリケーションを有効にする手順は、アプリケーションを使用可能にする手順とは別です。

5 健全性検査を作成します。

これを管理コンソールを使用して行うには、前に説明した手順でロードバランサの「ターゲット」タブを開き、「ターゲット」テーブルで「健全性チェックを編集」をクリックします。

健全性チェックは、不健全なサーバーインスタンスを監視し、それらの健全性が戻ったときにロードバランサが新しい要求を送信できるようにします。

注 - Sun Java System Web Server (6.1 または 7.0) を使用している場合は、手順 6 および 7 を実行する代わりに、ロードバランサ設定ファイルを生成してネットワーク経由でデータを Web Server に送信すれば、処理を 1 つの手順で完了できます。

これを管理コンソールを使用して行うには、目的のロードバランサをクリックし、「エクスポート」タブを開きます。このタブで、「今すぐ適用」をクリックします。これにより、データは Web サーバーの構成ディレクトリに送信されます。

6 ロードバランサ設定ファイルを生成します。

これを管理コンソールを使用して行うには、ロードバランサをクリックし、「エクスポート」タブを開きます。このタブで、「今すぐエクスポート」をクリックします。

このコマンドは、Sun GlassFish Enterprise Server に同梱されているロードバランサプラグインとともに使用する設定ファイルを生成します。

7 ロードバランサ設定ファイルを、ロードバランサプラグイン設定ファイルが格納されている Web サーバーの config ディレクトリにコピーします。

注 - ロードバランサ設定ファイルを自動生成し、ネットワーク経由で Web サーバーにデータを送信する処理を 1 つの手順で行うには、SSL 設定用に Web サーバーを設定し、DAS 証明書をインポートする必要があります。Sun Web Server の設定については、108 ページの「[Sun Web サーバーの設定](#)」を参照してください。

▼ asadmin ツールを使用して負荷分散を設定する

1 ロードバランサ設定を作成します。

これを行うには、コマンド `asadmin create-http-lb-config` を使用します。

注 - 次のすべての手順(手順2~7)を、単一の `asadmin` コマンド `create-http-lb` とそのオプションを使用して実行できます。このコマンドの詳細については、[create-http-lb\(1\)](#) を参照してください。

- ロードバランサが管理するクラスタまたはスタンドアロンサーバーインスタンスへの参照を追加します。
これを行うには、コマンド `asadmin create-http-lb-ref` を使用します。このコマンドの詳細については、[create-http-lb-ref\(1\)](#) を参照してください。
ターゲットを指定してロードバランサ設定を作成しており、そのターゲットが、ロードバランサが参照する唯一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である場合は、この手順を飛ばしてください。
- ロードバランサによって参照されるクラスタまたはスタンドアロンサーバーインスタンスを有効にします。
これを行うには、コマンド `asadmin enable-http-lb-server` を使用します。このコマンドの詳細については、[enable-http-lb-server\(1\)](#) を参照してください。
- アプリケーションの負荷分散を有効にします。
これを行うには、コマンド `asadmin enable-http-lb-application` を使用します。このコマンドの詳細については、[enable-http-lb-application\(1\)](#) を参照してください。
これらのアプリケーションは、ロードバランサが参照するクラスタまたはスタンドアロンインスタンスで使用するために、事前に配備および有効にしておく必要があります。負荷分散のためにアプリケーションを有効にする手順は、アプリケーションを使用可能にする手順とは別です。
- 健全性検査を作成します。
これを行うには、コマンド `asadmin create-http-health-checker` を使用します。このコマンドの詳細については、[create-http-health-checker\(1\)](#) を参照してください。
健全性チェッカーは、不健全なサーバーインスタンスを監視し、それらの健全性が戻ったときにロードバランサが新しい要求を送信できるようにします。

注 - Sun Java System Web Server (6.1 または 7.0) を使用している場合は、手順6および7を実行する代わりに、ロードバランサ設定ファイルを生成してネットワーク経由でデータを Web Server に送信すれば、処理を1つの手順で完了できます。

`asadmin` ツールを使用してこれを行うには、`create-http-lb` コマンドの `--autoapplyenabled` オプションを `true` に設定します。このコマンドの詳細については、[create-http-lb\(1\)](#) を参照してください。

- 6 ロードバランサ設定ファイルを生成します。
これを行うには、コマンド `asadmin export-http-lb-config` を使用します。
このコマンドは、Sun GlassFish Enterprise Server に同梱されているロードバランサプラグインとともに使用する設定ファイルを生成します。
- 7 ロードバランサ設定ファイルを、ロードバランサプラグイン設定ファイルが格納されている **Web** サーバーの `config` ディレクトリにコピーします。

注-ロードバランサ設定ファイルを自動生成し、ネットワーク経由で Web サーバーにデータを送信する処理を 1 つの手順で行うには、SSL 設定用に Web サーバーを設定し、DAS 証明書をインポートする必要があります。Sun Web Server の設定については、108 ページの「[Sun Web サーバー の設定](#)」を参照してください。

HTTP ロードバランサの配備

ロードバランサは、目標や環境に応じて、以下の節で説明している各種の方法で設定できます。

- 78 ページの「[クラスタ化されたサーバーインスタンスの使用](#)」
- 79 ページの「[複数のスタンドアロンインスタンスの使用](#)」

クラスタ化されたサーバーインスタンスの使用

ロードバランサを配備するためのもっとも一般的な方法は、サーバーインスタンスのクラスタ (1 つまたは複数) を使用する方法です。デフォルトでは、クラスタ内のすべてのインスタンスは同じ設定を持ち、同じアプリケーションが配備されています。ロードバランサは、サーバーインスタンスの間でワークロードを分散させ、正常でないインスタンスから正常なインスタンスへのフェイルオーバーを要求します。HTTP セッション持続性を設定している場合は、要求が処理を引き継がれるとセッション情報は保持されます。

複数のクラスタがある場合、要求はクラスタ間で負荷分散されますが、要求のフェイルオーバーは単一クラスタ内のインスタンス間でのみ行われます。ロードバランサで複数のクラスタを使用すると、アプリケーションの順次アップグレードが容易に可能になります。詳細については、95 ページの「[可用性を低下させないアプリケーションのアップグレード](#)」を参照してください。

注-クラスタ間およびスタンドアロンインスタンス間で要求を負荷分散することはできません。

複数のスタンドアロンインスタンスの使用

複数のスタンドアロンインスタンスを使用するようにロードバランサを設定し、要求をそれらのインスタンス間で負荷分散したりフェイルオーバーしたりすることも可能です。ただし、この設定では、それぞれのスタンドアロンインスタンスに同種の環境が確保され、同じアプリケーションが配備されていることを手動で確認する必要があります。クラスタでは自動的に同種の環境が維持されるため、ほとんどの状況では、クラスタの使用がより適切で、より容易な方法です。

HTTP ロードバランサの設定

ロードバランサ設定は `domain.xml` ファイルに保持されます。ロードバランサの設定は非常に柔軟性があります。

- ロードバランサがサービスを提供するドメインは1つだけですが、ドメインは関連する複数のロードバランサを持つことができます。
- 各ロードバランサ設定は、関連する複数のロードバランサを持つことができます。ただし、1つのロードバランサには、1つのロードバランサ設定しかできません。

以下の節では、ロードバランサ設定を作成、変更、および使用方法についてさらに詳しく説明します。

- 79 ページの「DAS 上での HTTP ロードバランサの設定」
- 81 ページの「HTTP ロードバランサ参照の作成」
- 81 ページの「HTTP 負荷分散のためのサーバーインスタンスの有効化」
- 82 ページの「HTTP 負荷分散のためのアプリケーションの有効化」
- 82 ページの「HTTP 健全性検査の作成」
- 84 ページの「HTTP ロードバランサ設定ファイルのエクスポート」
- 85 ページの「HTTP ロードバランサ設定の変更」
- 86 ページの「動的再設定を有効にする」
- 87 ページの「サーバーインスタンスまたはクラスタの無効化 (休止)」
- 87 ページの「アプリケーションの無効化 (休止)」
- 88 ページの「HTTP および HTTPS のフェイルオーバーの設定」
- 90 ページの「HTTP ロードバランサによるリダイレクトの使用」
- 93 ページの「べき等 URL の設定」

DAS 上での HTTP ロードバランサの設定

管理コンソールまたは `asadmin` コマンドの `create-http-lb` を使用して、DAS 上のロードバランサ設定を作成できます。以降の手順ではその方法を説明します。`asadmin` コマンドの詳細については、`man` ページまたは『Reference Manual』で `create-http-lb`、`delete-http-lb`、および `list-http-lbs` を参照してください。

管理コンソールの左区画で下にスクロールして「HTTP ロードバランサ」ノードをクリックし、右側の「HTTP ロードバランサ」ページで「新規」をクリックします。「新しいHTTP ロードバランサ」ページで、ロードバランサのホストになるマシンについて次の情報を指定します。

フィールド	説明
名前	ロードバランサ設定の名前。
有効	「有効」チェックボックスにチェックを入れると、ロードバランサ設定の変更が、Web サーバー構成ディレクトリ内の物理ロードバランサに自動的に送信されます。
ホスト	Web サーバーインスタンスがインストールされているサーバー。
管理ポート	保護された HTTP リスナーポート。
プロキシホスト	プロキシサーバーインスタンスがインストールされているサーバー。
プロキシポート	プロキシサーバーによって使用されるポート番号。

ロードバランサ設定は、`asadmin` コマンド `create-http-lb-config` を使用して作成することもできます。表 4-1 は、パラメータについての説明です。詳細については、`create-http-lb-config`、`delete-http-lb-config`、および `list-http-lb-configs` についてのドキュメントを参照してください。

表 4-1 ロードバランサ設定のパラメータ

パラメータ	説明
応答タイムアウト	サーバーインスタンスが応答を返すまでの秒数。タイムアウト時間内に応答が着信しない場合、サーバーが正常でないとして判断されます。デフォルトは 60 です。
HTTPS ルーティング	ロードバランサに対する HTTPS 要求の結果が、サーバーインスタンスに対する HTTPS または HTTP 要求となるかどうかを指定します。詳細については、89 ページの「HTTPS ルーティングの設定」を参照してください。
再読み込み間隔	ロードバランサ設定ファイル <code>loadbalancer.xml</code> に対する変更をチェックする間隔。チェックによって変更が検出されると、設定ファイルが再読み込みされます。この値が 0 の場合は、再読み込みが無効になります。詳細については、86 ページの「動的再設定を有効にする」を参照してください。
監視	ロードバランサで監視が有効かどうかを指定します。

表 4-1 ロードバランサ設定のパラメータ (続き)

パラメータ	説明
ルート Cookie	ロードバランサプラグインがルート情報を記録するために使用する Cookie の名前を指定します。HTTP クライアントは Cookie をサポートする必要があります。Cookie を格納する前に確認するようにブラウザが設定されている場合は、その Cookie の名前は「JROUTE」です。
ターゲット	ロードバランサ設定のターゲットを指定します。ターゲットを指定すると、設定に参照を追加した場合と同じ結果になります。ターゲットは、クラスタまたはスタンドアロンインスタンスです。

HTTP ロードバランサ参照の作成

ロードバランサでスタンドアロンのサーバーまたはクラスタへの参照を作成すると、ロードバランサが制御するターゲットサーバーおよびクラスタの一覧に、参照先のサーバーまたはクラスタが追加されます。この場合でも、参照先のサーバーまたはクラスタに対する要求を負荷分散する前に、そのサーバーまたはクラスタを有効化する必要があります。ターゲットを指定してロードバランサ設定を作成した場合、そのターゲットはすでに参照として追加されています。

管理コンソールを使用して参照を作成するには、左側の区画で「HTTP ロードバランサ」ノードをクリックし、ノードの下に表示されるリストから目的のロードバランサをクリックします。「ターゲット」タブを開き、「ターゲットを管理」をクリックします。「ロードバランサのターゲットを管理」ページで、必要なターゲットを選択します。別の方法として、`create-http-lb-ref` を使用して参照を作成することもできます。ロードバランサ設定名と、ターゲットサーバーインスタンスまたはクラスタを指定する必要があります。

参照を削除するには、`delete-http-lb-ref` を使用します。参照を削除する前に、`disable-http-lb-server` を使用して参照先のサーバーまたはクラスタを無効にする必要があります。

詳細については、`create-http-lb-ref` および `delete-http-lb-ref` についてのドキュメントを参照してください。

HTTP 負荷分散のためのサーバーインスタンスの有効化

サーバーインスタンスまたはクラスタへの参照を作成したら、`enable-http-lb-server` を使用してサーバーインスタンスまたはクラスタを有効にします。ロードバランサ設定の作成時にサーバーインスタンスまたはクラスタをターゲットとして使用した場合は、それを有効にする必要があります。管理コンソールを使用してこれを行うには、左側の区画で「HTTP ロードバランサ」ノードを

クリックし、ノードの下に表示されるリストから目的のロードバランサをクリックします。ここで「ターゲット」タブを開き、「ターゲット」テーブルで、有効にするインスタンスの隣のチェックボックスにチェックマークを付け、「有効」をクリックします。

詳細については、`enable-http-lb-server` についてのドキュメントを参照してください。

HTTP 負荷分散のためのアプリケーションの有効化

ロードバランサによって管理されるすべてのサーバーは、アプリケーションの同じセットが配備されていることを含め、同じように設定されている必要があります。アプリケーションが配備されてアクセス可能になると (配備手順の実行中または完了後)、負荷分散を有効にする必要があります。アプリケーションで負荷分散が有効化されていない場合、そのアプリケーションが配備されているサーバーへの要求が負荷分散およびフェイルオーバーされていても、アプリケーションへの要求は負荷分散およびフェイルオーバーされません。

アプリケーションを有効にする際に、アプリケーション名とターゲットを指定します。ロードバランサが複数のターゲット (2つのクラスタなど) を管理している場合は、すべてのターゲットでアプリケーションを有効にしてください。

管理コンソールを使用してアプリケーションを有効化するには、左側の区画で「HTTP ロードバランサ」ノードをクリックし、ノードの下に表示されるリストから目的のロードバランサをクリックします。前の手順の説明に従って「ターゲット」タブを開き、必要なクラスタをクリックします。ここで「アプリケーション」タブを開き、必要なアプリケーションを選択し、「その他の操作」ドロップダウンリストから、「ロードバランサ有効」を選択します。コマンド行からこれを行う場合、コマンド `asadmin enable-http-lb-application` を使用できます。コマンドの詳細については、`man` ページを参照してください。

新しいアプリケーションを配備する場合にも、アプリケーションで負荷分散を有効にして、再度ロードバランサ設定をエクスポートする必要があります。

HTTP 健全性検査の作成

ロードバランサの健全性検査は、設定されている Enterprise Server インスタンスの中で、正常ではないとしてマークされているすべてのインスタンスを定期的にチェックします。健全性検査は必須ではありませんが、このプログラムが存在しない場合、または無効になっている場合は、正常でないインスタンスの定期的な健全性検査は実行されません。ロードバランサは、正常でないインスタンスが正常になるタイミングを判断することはできません。

ロードバランサの健全性検査メカニズムは、HTTP を使用してインスタンスと通信します。健全性検査は、指定された URL に HTTP 要求を送信し、応答を待ちます。HTTP 応答ヘッダー内の状態コードが 100 ~ 500 の間であれば、インスタンスが正常であることを示します。

注-ロードバランサがクラスタに対するフロントエンドであり、クライアント証明書認証を有効にしてセキュリティー保護ポートを使用しているインスタンスがそのクラスタに含まれる配備状況では、健全性検査はインスタンスの診断を実行できません。したがって、そのようなインスタンスは常に正常でないと認識され、それらのインスタンスには要求は送られません。

健全性検査の作成

健全性検査のプロパティを指定するために、管理コンソールまたは `asadmin create-http-health-checker` コマンドを使用できます。管理コンソールで行うには、「HTTP ロードバランサ」ノードに移動し、ノードを展開してロードバランサを選択します。次に「ターゲット」タブを開き、「ターゲット」テーブルで目的のターゲットの「健全性検査を編集」リンクをクリックします。次のパラメータを指定します。

表 4-2 健全性検査のパラメータ

パラメータ	説明	デフォルト
ロードバランサ	選択したサーバーで負荷分散を使用できるようにするには、「有効」チェックボックスにチェックを入れます。	False/無効
無効タイムアウト	このサーバーが無効にされてから休止状態に入るまでの時間(分単位)。	30 分
url	ロードバランサが健康状態を判断するためにチェックするリスナーの URL を指定します。	"/"
interval	インスタンスの健全性検査を実行する間隔を秒単位で指定します。0 を指定すると、健全性検査が無効になります。	30 秒
timeout	正常だと見なされるリスナーが応答を受け取るまでのタイムアウト間隔を秒単位で指定します。	10 秒

インスタンスが正常でないとマークされている場合、健全性検査が正常ではないインスタンスをポーリングして、インスタンスが正常になったかどうかを判断します。健全性検査は、指定された URL を使用して正常でないインスタンスをすべてチェックし、それらが正常な状態に戻っているかどうかを判断します。

健全性検査により、正常ではないインスタンスが正常になったことが確認されると、そのインスタンスが正常なインスタンスのリストに加えられます。

詳細については、`create-http-health-checker` および `delete-http-health-checker` のドキュメントを参照してください。

正常なインスタンス用健全性検査の追加プロパティ

`create-http-health-checker` によって作成された健全性検査は、正常ではないインスタンスのみをチェックします。正常なインスタンスを定期的にチェックするには、エクスポートした `loadbalancer.xml` ファイルに追加のプロパティをいくつか設定します。

正常なインスタンスをチェックするには、次のプロパティを設定します。

表 4-3 健全性検査のプロパティ

プロパティ	定義
<code>active-healthcheck-enabled</code>	サーバーインスタンスが正常であるかどうかを調べるために、それらに対して Ping を実行するかどうかを示す <code>true/false</code> フラグ。サーバーインスタンスに対して Ping を実行するには、このフラグを <code>true</code> に設定します。
<code>number-healthcheck-retries</code>	ロードバランサの健全性検査が、応答しないサーバーインスタンスを正常でないとしてマークするまでに、それらに対して Ping を実行する回数を指定します。有効な値は 1～1000 です。デフォルトでは 3 に設定されています。

`asadmin set` コマンドを使用して、プロパティを設定します。次に例を示します。

```
asadmin set
domain.lb-configs.load-balancer-config.property.active-healthcheck-enabled=true

asadmin set
domain.lb-configs.load-balancer-config.property.number-healthcheck-retries=5
```

これらのプロパティを追加したあと、`loadbalancer.xml` ファイルをふたたび編集およびエクスポートする場合、新しくエクスポートした設定にはこれらのプロパティが含まれます。

HTTP ロードバランサ設定ファイルのエクスポート

Sun GlassFish Enterprise Server で提供されるロードバランサプラグインは、`loadbalancer.xml` という設定ファイルを使用します。ロードバランサを設定したあとで、設定の詳細を `domain.xml` から `loadbalancer.xml` ファイルにエクスポートできます。これは、管理コンソールまたは `asadmin` ユーティリティを使用することができます。

▼ 管理コンソールを使用してロードバランサ設定をエクスポートする

- 1 「HTTP ロードバランサ」ノードに移動し、ノードを展開します。
- 2 目的のロードバランサをクリックします。
すべてのロードバランサ設定の詳細が「一般」、「設定」、および「ターゲット」の各タブに表示されます。
- 3 「エクスポート」タブを開き、「今すぐエクスポート」をクリックします。
- 4 エクスポートしたロードバランサ設定ファイルを、**Web** サーバーの構成ディレクトリにコピーします。

▼ asadmin ツールを使用してロードバランサ設定をエクスポートする

- 1 asadmin コマンドの `export-http-lb-config` を使用して、`loadbalancer.xml` ファイルをエクスポートします。

特定のロードバランサ設定の `loadbalancer.xml` ファイルをエクスポートします。パスまたは別のファイル名を指定できます。ファイル名を指定しない場合、ファイルには `loadbalancer.xml`、`load-balancer-config-name` という名前が付けられます。パスを指定しない場合、ファイルは `domain-dir/generated` ディレクトリに作成されます。

Windows でパスを指定する場合は、パスを引用符で囲みます。たとえば、`"C:\Sun\AppServer\loadbalancer.xml"` のように指定します。

- 2 エクスポートしたロードバランサ設定ファイルを、**Web** サーバーの構成ディレクトリにコピーします。

たとえば、Sun Java System Web Server の場合、通常のコピー先は `web-server-root/config` となります。

Web サーバーの構成ディレクトリ内のロードバランサ設定ファイルには、`loadbalancer.xml` という名前を付ける必要があります。ファイルの名前は `loadbalancer.xml`、`load-balancer-config-name` などの別の名前になっている場合は、名前を変更する必要があります。

HTTP ロードバランサ設定の変更

サーバーへの参照の作成または削除、新しいアプリケーションの配備、サーバーまたはアプリケーションの有効化/無効化などによってロードバランサ設定を変更した場合は、ロードバランサ設定ファイルをふたたびエクスポートして、Web

サーバーの config ディレクトリにコピーします。詳細については、[84 ページ](#)の「[HTTP ロードバランサ設定ファイルのエクスポート](#)」を参照してください。

ロードバランサプラグインは、ロードバランサ設定で指定した再読み込み間隔に従って、更新された設定を定期的にチェックします。指定した時間が経過して、ロードバランサが新しい設定ファイルを検出した場合は、その設定を使用して再読み込みが開始されます。

動的再設定を有効にする

動的再設定で、ロードバランサプラグインは更新された設定がないかどうかを定期的にチェックします。

動的再設定を有効にするには、次の手順に従います。

- ロードバランサ設定を作成するときに、`asadmin create-http-lb` で `--reloadinterval` オプションを使用します。
このオプションは、ロードバランサ設定ファイル `loadbalancer.xml` に対する変更のチェックの間隔を設定します。この値が 0 の場合は、動的再設定が無効になります。デフォルトでは、動的再設定が有効になり、再読み込み間隔は 60 秒に設定されます。
- 以前に動的再設定が無効にしていた場合、または再読み込み間隔を変更する場合は、`asadmin set` コマンドを使用します。
再読み込み間隔を変更したら、ロードバランサ設定ファイルをふたたびエクスポートして、Web サーバーの config ディレクトリにコピーしたあと、Web サーバーを再起動します。

注-ロードバランサがそれ自体の再設定を試みているときにハードディスク読み込みエラーが発生した場合、ロードバランサは現在メモリーに格納されている設定を使用します。ロードバランサはまた、既存の設定を上書きする前に、変更された設定データが必ず DTD に適合するようにします。

ディスク読み込みエラーが発生すると、Web サーバーのエラーログファイルに警告メッセージが記録されます。

Sun Java System Web Server のエラーログは、次の場所にあります。
`web-server-install-dir/web-server-instance/logs/`。

サーバーインスタンスまたはクラスタの無効化(休止)

何らかの理由でサーバーを停止する場合は、停止の前にインスタンスで要求の処理を完了する必要があります。サーバーインスタンスまたはクラスタを正常に無効にするプロセスは、休止と呼ばれます。

ロードバランサはインスタンスを休止するために、次のポリシーを使用します。

- あるインスタンス(スタンドアロンまたはクラスタの一部)が無効化され、タイムアウトが経過していない場合、スティッキ要求はインスタンスに配信され続けます。ただし、新しい要求は無効化されたインスタンスに送信されません。
- タイムアウトを経過すると、インスタンスは無効化されます。ロードバランサからインスタンスへのすべてのオープン接続が閉じられます。このインスタンスに固定されている一部のセッションが無効化されなかった場合でも、ロードバランサはこのインスタンスに要求を送りません。ロードバランサはスティッキ要求を別の正常なインスタンスにフェイルオーバーします。

▼ サーバーインスタンスまたはクラスタを無効にする

- 1 `asadmin disable-http-lb-server` を実行して、タイムアウトを分単位で設定します。
- 2 `asadmin export-http-lb-config` を使用して、ロードバランサ設定ファイルをエクスポートします。
- 3 エクスポートした設定を **Web** サーバーの `config` ディレクトリにコピーします。
- 4 サーバーインスタンスを停止します。

アプリケーションの無効化(休止)

Web アプリケーションの配備を取り消す前に、アプリケーションで要求の処理が完了する必要があります。アプリケーションを正常に無効にするプロセスは、休止と呼ばれます。アプリケーションを休止する場合は、タイムアウト期間を指定します。ロードバランサは、指定されたタイムアウト期間に基づいて、アプリケーションを休止するために次のポリシーを使用します。

- タイムアウト期間が経過していない場合、ロードバランサは新しい要求をアプリケーションには転送せずに、Web サーバーに戻します。ただし、タイムアウト期間が経過するまで、スティッキ要求の転送は引き続き行います。
- タイムアウト期間を経過すると、ロードバランサは、スティッキ要求を含むアプリケーションへのすべての要求を受け付けなくなります。

ロードバランサが参照するすべてのサーバーインスタンスまたはクラスタから、あるアプリケーションを無効にする場合、無効化されたアプリケーションのユーザーは、アプリケーションが再度有効化されるまでサービスを受けられません。1つのサーバーインスタンスまたはクラスタからアプリケーションを無効にして、別のサーバーインスタンスまたはクラスタでは有効にする場合、ユーザーは引き続きアプリケーションにアクセスできます。詳細については、[95 ページの「可用性を低下させないアプリケーションのアップグレード」](#)を参照してください。

▼ アプリケーションを無効にする

- 1 `asadmin disable-http-lb-application` を使用して、次のパラメータを指定します。
 - タイムアウト (分単位)
 - 無効にするアプリケーションの名前
 - 無効化を実行するターゲットクラスタまたはインスタンス
- 2 `asadmin export-http-lb-config` を使用して、ロードバランサ設定ファイルをエクスポートします。
- 3 エクスポートした設定を **Web** サーバーの `config` ディレクトリにコピーします。

HTTP および HTTPS のフェイルオーバーの設定

HTTP/HTTPS セッションが接続されていた元のインスタンスを利用できなくなった場合、ロードバランサプラグインは、そのセッションを別のアプリケーションサーバーインスタンスにフェイルオーバーします。この節では、HTTP/HTTPS ルーティングとセッションフェイルオーバーを有効にするようにロードバランサプラグインを設定する方法について説明します。

HTTPS ルーティング

ロードバランサプラグインは、すべての着信 HTTP または HTTPS 要求をインスタンスにルーティングします。ただし、HTTPS ルーティングが有効になっている場合、ロードバランサプラグインは HTTPS ポートのみを使用して HTTPS 要求をサーバーに転送します。HTTPS ルーティングは、新しい要求とスティッキ要求の両方について実行されます。

HTTPS 要求が受信され、処理中のセッションがない場合、ロードバランサプラグインは設定されている HTTPS ポートを使用して使用可能なインスタンスを選択し、要求をそのインスタンスに転送します。

継続中の HTTP セッションで、同じセッションに対して新しい HTTPS 要求が受信された場合、HTTP セッション中に保存されたセッションおよびスティッキ情報を使用

してHTTPS 要求がルーティングされます。新しいHTTPS 要求は、最後のHTTP 要求が処理された同じサーバーにルーティングされます。ただし、HTTPS ポートが使用されます。

HTTPS ルーティングの設定

`create-http-lb-config` コマンドの `httpsrouting` オプションは、負荷分散に関わるすべてのアプリケーションサーバーに対してHTTPS ルーティングが有効か無効かを制御します。このオプションが `false` に設定されている場合、すべてのHTTP およびHTTPS 要求はHTTP として転送されます。`true` に設定されている場合、HTTPS はHTTPS 要求として転送されます。新しいロードバランサ設定を作成する場合、または、作成後に `asadmin set` コマンドを使用して変更する場合には、HTTPS ルーティングを設定してください。

注 -

- HTTPS ルーティングを動作させるには、1つまたは複数のHTTPS リスナーを設定する必要があります。
- `https-routing` が `true` に設定されていて、クラスタ内に正常なHTTPS リスナーが存在していない状態で新しい要求またはスティッキ要求が着信した場合、その要求はエラーを生成します。

既知の問題

ロードバランサには、HTTP/HTTPS 要求の処理に関する次の制限事項があります。

- あるセッションがHTTP 要求とHTTPS 要求を組み合わせで使用する場合、最初の要求は必ずHTTP 要求にする必要があります。最初の要求がHTTPS 要求の場合、そのあとHTTP 要求を続けられません。これは、HTTPS セッションに関連付けられているCookie がブラウザによって返されないからです。ブラウザは、異なる2つのプロトコルを異なる2つのサーバーと解釈し、新しいセッションを開始します。この制限は、`httpsrouting` が `true` に設定されている場合のみ有効です。
- あるセッションにHTTP 要求とHTTPS 要求の組み合わせが含まれる場合、インスタンスはHTTP リスナーとHTTPS リスナーの両方を使用して設定される必要があります。この制限は、`httpsrouting` が `true` に設定されている場合のみ有効です。
- あるセッションにHTTP 要求とHTTPS 要求の組み合わせが含まれる場合、インスタンスは、標準ポート番号 (HTTP では80、HTTPS では443) を使用するHTTP およびHTTPS リスナーによって設定される必要があります。この制限は、`httpsrouting` に設定された値に関係なく適用されます。

HTTP ロードバランサによるリダイレクトの使用

リダイレクトを使用して、ある URL から別の URL へ要求をリダイレクトします。たとえば、リダイレクトを使用して、ユーザーを別の Web サイトに送信したり(旧バージョンのアプリケーションから新しいバージョンへリダイレクトする場合など)、HTTP から HTTPS へ、または HTTPS から HTTP へ送信したりします。アプリケーション内でリダイレクトを有効にする方法はいくつもあります(たとえば、servlet ベースのリダイレクト、web.xml リダイレクト)。ただし、ロードバランサを通してリダイレクト URL を送信するには、Enterprise Server またはロードバランサに対していくつか追加設定が必要な場合があります。リダイレクトは HTTPS ルーティングを使用して転送される要求とは異なるので注意してください。リダイレクトを使用するときには、`httpsrouting` を `false` に設定します。HTTPS 要求を HTTP に転送するように設定する場合は、88 ページの「HTTPS ルーティング」を使用します。

リダイレクトに影響するプロパティは次のとおりです。HTTP サービスまたは HTTP リスナーの `authPassthroughEnabled` および `proxyHandler` プロパティと、`loadbalancer.xml` ファイル内の `rewrite-location` プロパティ。

authPassthroughEnabled プロパティ

Enterprise Server `authPassthroughEnabled` プロパティを `true` に設定した場合、カスタムな要求ヘッダーを使用して、元のクライアント要求に関する情報(クライアント IP アドレス、SSL キーサイズ、認証されたクライアント証明書チェーンなど)が HTTP リスナーへ送信されます。`authPassthroughEnabled` プロパティを使用すると、ハードウェアアクセラレータ(インストールされている場合)を利用して SSL 認証をより高速にできます。ハードウェアアクセラレータの設定は、クラスタ化されたそれぞれの Enterprise Server インスタンス上よりも、ロードバランサ上で行う方が簡単です。



注意 - `authPassthroughEnabled` は、Enterprise Server がファイアウォールの後ろにある場合のみ `true` に設定します。

`asadmin set` コマンドを使用して、HTTP サービスまたは個別の HTTP リスナー上に `authPassthroughEnabled` プロパティを設定します。個別の HTTP リスナーに対する設定は、HTTP サービスに対する設定よりも優先されます。

すべての HTTP および HTTPS リスナー上に `authPassthroughEnabled` 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  
cluster-name-config.http-service.property.authPassthroughEnabled=true
```

このプロパティを個別のリスナー上に設定するには、次のコマンドを使用します。

```
asadmin set cluster-name-config.http-service.http-listener.listener-name.property.  
authPassthroughEnabled=true
```

proxyHandler プロパティ

Enterprise Server のプロキシハンドラは、プロキシサーバー（この場合はロードバランサ）によって遮断されてサーバーに転送された元のクライアント要求に関する情報を取得し、クライアント要求のターゲットである配備された Web アプリケーションでこの情報を使用できるようにする役目を担っています。遮断するプロキシサーバーが SSL の終端となる場合、プロキシハンドラは、元の要求が HTTPS 要求であったのか、SSL クライアント認証が有効なのかなど元の要求に関する追加の情報を取得して使用可能にします。proxyHandler プロパティは、authPassthroughEnabled が true に設定されている場合のみ使用します。

プロキシハンドラは、プロキシサーバーが元のクライアント要求に関する情報を伝達するのに使用するカスタムな要求ヘッダーについて着信要求を調べ、標準の ServletRequest API を使用してこの情報を Web アプリケーションで使用できるようにします。

プロキシハンドラの実装は、proxyHandler プロパティを使用して、HTTP サービスレベルでグローバルに設定することも、個別の HTTP リスナーに対して設定することもできます。このプロパティの値は、com.sun.appserv.ProxyHandler 抽象クラスの実装の完全修飾クラス名を指定します。設定可能なプロキシハンドラの実装が、HTTP 要求ヘッダー名について認識し、プロキシサーバーが元のクライアント要求に関する情報を伝達するのに使用する値の書式を理解しているかぎり、プロキシハンドラの実装によってサーバーは任意のプロキシサーバーで動作できます。

Enterprise Server のプロキシハンドラは、要求ヘッダーから SSL 証明書チェーンを読み込んで解析します。これによって、バックエンドのインスタンスは、SSL 終端となるプロキシサーバー（ここではロードバランサ）によって遮断された元のクライアント要求に関する情報を取得できるようになります。デフォルトのプロキシハンドラ設定を使用することも、HTTP サービスまたは HTTP/HTTPS リスナーの proxyHandler プロパティを使用してユーザー設定することもできます。proxyHandler プロパティでは、そのリスナーまたはすべてのリスナーによって使用される com.sun.appserv.ProxyHandler 抽象クラスのカスタム実装の完全修飾クラス名を指定します。

この抽象クラスの実装は、インスタンスへの元のクライアント要求に関する情報を伝達するためにプロキシサーバーが使用するカスタム要求ヘッダーに対して行われる特定の要求を調べ、その情報を呼び出し元に返します。デフォルトの実装では、Proxy-ip という名前の HTTP 要求ヘッダーからクライアント IP アドレスを、Proxy-keysize という名前の HTTP 要求ヘッダーから SSL キーサイズを、Proxy-auth-cert という名前の HTTP 要求ヘッダーから SSL クライアント証明書チェーンを読み取ります。Proxy-auth-cert 値には、BEGIN CERTIFICATE および END CERTIFICATE の境界がなく、\n が %d%a によって置き換えられた BASE-64 エンコードのクライアント証明書チェーンを含む必要があります。

このプロパティは、`authPassThroughEnabled` を `true` に設定した場合のみ使用できません。個別の HTTP または HTTPS リスナー上に `proxyHandler` プロパティを設定すると、すべてのリスナーのデフォルト設定が上書きされます。

`asadmin set` コマンドを使用して、HTTP サービスまたは個別の HTTP リスナー上に `proxyHandler` プロパティを設定します。

`proxyHandler` プロパティを、すべての HTTP および HTTPS リスナー上に設定するには、次のコマンドを使用します。

```
asadmin set cluster-name-config.http-service.property.proxyHandler=classname
```

このプロパティを個別のリスナー上に設定するには、次のコマンドを使用します。

```
asadmin set cluster-name-config.http-service.http-listener.listener-name.property.proxyHandler=classname
```

rewrite-location プロパティ

`rewrite-location` プロパティを `true` に設定すると、元の要求情報が書き換えられ、プロトコル (HTTP または HTTPS)、ホスト、およびポートの情報が追加されます。デフォルトでは、以前のリリースの Enterprise Server との後方互換性を保つために、`rewrite-location` プロパティは `true` に設定されます。

`rewrite-location` プロパティは、`asadmin create-http-lb-config` コマンドを介して使用することはできません。プロパティを使用するには、次のように `asadmin set` コマンドを使用します。

```
asadmin set domain.lb-configs.load-balancer-config.property.rewrite-location=false
```

rewrite-location プロパティを設定するときには、次の点に注意します。

- httpsrouting が false で、Enterprise Server 上で authPassthroughEnabled が有効でない場合は、rewrite-location プロパティを true に設定します。authPassthroughEnabled が有効でない場合、Enterprise Server は元の要求のプロトコル (HTTP または HTTPS) を認識しません。rewrite-location を true に設定することで、ロードバランサは書き換えの場所のプロトコル部分を適切に変更します。つまり、クライアントが HTTPS 要求を送信していれば、ロードバランサはクライアントをロードバランサ上の HTTPS が有効になっているリスナーポートにリダイレクトします。このプロセスは HTTP 要求の場合と同じです。
- httpsrouting が false で、authPassthroughEnabled が Enterprise Server 上で有効になっている場合、Enterprise Server はクライアント要求が HTTP と HTTPS のどちらであるのかを認識しているため、rewrite-location を true または false に設定することができます。authPassthroughEnabled が有効になっている場合、Enterprise Server は書き換えの場所のプロトコル部分を適切に変更します。rewrite-location が false に設定されている場合、ロードバランサはリダイレクトされた URL の場所を書き換えません。true に設定されている場合は、リダイレクトされた URL の場所を書き換えます。しかしこの書き換えは、Enterprise Server がクライアントからの HTTPS 接続を認識していれば必要ありません。また、アプリケーションが HTTP から HTTPS または HTTPS から HTTP にリダイレクトする必要のある場合、rewrite-location パラメータを false に設定する必要があります。

べき等 URL の設定

べき等要求とは、再試行時にアプリケーションに変更や不一致をもたらさないタイプの要求です。HTTP の場合、GET などの一部のメソッドはべき等のですが、POST などその他のメソッドはそうではありません。べき等 URL の再試行では、サーバーまたはデータベースの値が変更されてはいけません。ユーザーが受信する応答が変更されるだけです。

べき等要求の例としては、検索エンジンクエリーやデータベースクエリーがあります。基礎となる原則は、再試行によってデータの更新や変更が発生しないことです。

配備されたアプリケーションの可用性を向上させるには、ロードバランサによって処理されたすべてのインスタンスに、失敗したべき等の HTTP 要求を再試行する環境を設定します。このオプションは、検索要求の再試行など、読み取り専用の要求に使用されます。

sun-web.xml ファイルに、べき等 URL を設定します。ロードバランサ設定をエクスポートする場合、べき等 URL の情報は自動的に loadbalancer.xml ファイルに追加されます。

べき等 URL の詳細については、『[Sun GlassFish Enterprise Server 2.1 Developer's Guide](#)』の「[Configuring Idempotent URL Requests](#)」を参照してください。

複数の Web サーバーインスタンスの設定

インストーラでは、1 台のマシンに複数のロードバランサプラグインをインストールできません。1 つまたは複数のクラスタ内の 1 台のマシンに、ロードバランサプラグインとともに複数の Web サーバーを置くには、いくつかの手順を手動で実行してロードバランサプラグインを設定する必要があります。

▼ 複数の Web サーバーインスタンスを設定する

- 1 ロードバランサプラグインを使用するように新しい Web サーバーインスタンスを設定します。
詳細な手順については、[108 ページの「Sun Web サーバーの設定」](#)を参照してください。
- 2 既存の Web サーバーインスタンスの config ディレクトリから、DTD ファイル `sun-loadbalancer_1_1.dtd` を新しいインスタンスの config ディレクトリにコピーします。
- 3 ロードバランサ設定ファイルを設定します。次のいずれかを実行します。
 - 既存のロードバランサ設定をコピーします。
既存のロードバランサ設定を使用して、既存の Web サーバーインスタンスの config ディレクトリから、`loadbalancer.xml` ファイルを新しいインスタンスの config ディレクトリにコピーします。
 - 新しいロードバランサ設定を作成します。
 - a. `asadmin create-http-lb-config` を使用して、新しいロードバランサ設定を作成します。
 - b. `asadmin export http-lb-config` を使用して、新しい設定を `loadbalancer.xml` ファイルにエクスポートします。
 - c. `loadbalancer.xml` ファイルを、新しい Web サーバーの config ディレクトリにコピーします。
ロードバランサ設定を作成し、それを `loadbalancer.xml` ファイルにエクスポートする方法については、[79 ページの「DAS 上での HTTP ロードバランサの設定」](#)を参照してください。

可用性を低下させないアプリケーションのアップグレード

ユーザーへの可用性を低下させることなくアプリケーションを新しいバージョンにアップグレードする方法は、順次アップグレードと呼ばれます。アップグレードの前後で2つのバージョンのアプリケーションを慎重に管理することによって、アプリケーションの現在のユーザーが中断されることなくタスクを完了できる一方で、新しいユーザーが新しいバージョンのアプリケーションを透過的に取得できるようになります。順次アップグレードの場合、ユーザーはアップグレードが行われたことに気付きません。

アプリケーションの互換性

順次アップグレードでは、2つのアプリケーションバージョン間の変更の大きさに応じて、さまざまなレベルの困難が発生します。

変更が、たとえば、静的なテキストやイメージへの変更のような表面的なものであれば、2つのバージョンのアプリケーションには互換性があり、同じクラスタ内で両方のバージョンを一度に実行することができます。

互換性のあるアプリケーションは、次の条件を備えている必要があります。

- 同じセッション情報を使用している。
- 互換性のあるデータベーススキーマを使用している。
- 一般に互換性のあるアプリケーションレベルのビジネスロジックを採用している。
- 同じ物理データソースを使用している。

互換性のあるアプリケーションの順次アップグレードは、単一のクラスタまたは複数のクラスタのどちらでも実行できます。詳細については、[96 ページの「単一クラスタでのアップグレード」](#)を参照してください。

2つのバージョンのアプリケーションがこれらのすべての条件を満たしていない場合、アプリケーションは「互換性がない」と見なされます。互換性のないバージョンのアプリケーションを同じクラスタで実行すると、アプリケーションのデータが破壊され、セッションフェイルオーバーが正しく機能しない場合があります。発生する問題は、非互換性の種類や程度によって異なります。新しいバージョンを配備して古いクラスタやアプリケーションを徐々に休止する「シャドウクラスタ」を作成して、互換性のないアプリケーションをアップグレードすることをお勧めします。詳細については、[99 ページの「互換性のないアプリケーションのアップグレード」](#)を参照してください。

アプリケーション開発者および管理者は、アプリケーションのバージョンに互換性があるかどうかを判断できる最適な人びとです。不明な場合は、バージョンには互換性がないと仮定してください。これがもっとも安全な方法です。

単一クラスタでのアップグレード

単一のクラスタに配備されたアプリケーションの順次アップグレードは、そのクラスタの設定がほかのどのクラスタとも共有されていないと仮定して行うことができます。

▼ 単一のクラスタでアプリケーションをアップグレードする

- 1 旧バージョンのアプリケーションを保存するか、ドメインをバックアップします。ドメインをバックアップするには、`asadmin backup-domain` コマンドを使用します。

- 2 クラスタの動的再設定を無効にします (有効になっている場合)。管理コンソールを使用してこれを行うには、次の手順に従います。

- a. 「設定」ノードを開きます。
- b. クラスタの設定の名前をクリックします。
- c. 「システムプロパティの設定」ページで、「動的再設定を有効」ボックスのチェックをはずします。
- d. 「保存」をクリックします。

あるいは、次のコマンドを使用します。

```
asadmin set --user user --passwordfile password-file  
cluster-name-config.dynamic-reconfiguration-enabled=false
```

- 3 ターゲットの domain に対して、アップグレードしたアプリケーションを再配備します。管理コンソールを使って再配備する場合、ドメインが自動的にターゲットになります。asadmin を使用している場合は、ターゲットのドメインを指定します。動的再設定が無効なので、旧アプリケーションがクラスタで実行し続けます。
- 4 `asadmin enable-http-lb-application` を使用して、インスタンスに対して再配備したアプリケーションを有効にします。
- 5 ロードバランサから、クラスタ内の1つのサーバーインスタンスを休止します。次の手順を実行します。
 - a. `asadmin disable-http-lb-server` を使用して、サーバーインスタンスを無効にします。

複数のクラスタでのアップグレード

▼ 2つ以上のクラスタで、互換性のあるアプリケーションをアップグレードする

- 1 旧バージョンのアプリケーションを保存するか、ドメインをバックアップします。ドメインをバックアップするには、`asadmin backup-domain` コマンドを使用します。
- 2 すべてのクラスタの動的再設定を無効にします (有効になっている場合)。管理コンソールを使用してこれを行うには、次の手順に従います。

- a. 「設定」ノードを開きます。
- b. 1つのクラスタの設定の名前をクリックします。
- c. 「システムプロパティの設定」ページで、「動的再設定を有効」ボックスのチェックをはずします。
- d. 「保存」をクリックします。
- e. ほかのクラスタに対して上記手順を繰り返します
あるいは、次のコマンドを使用します。

```
asadmin set --user user --passwordfile password-file  
cluster-name-config.dynamic-reconfiguration-enabled=false
```

- 3 ターゲットの domain に対して、アップグレードしたアプリケーションを再配備します。
管理コンソールを使って再配備する場合、ドメインが自動的にターゲットになります。`asadmin` を使用している場合は、ターゲットのドメインを指定します。動的再設定が無効なので、旧アプリケーションがクラスタで実行し続けます。
- 4 `asadmin enable-http-lb-application` を使用して、クラスタに対して再配備したアプリケーションを有効にします。
- 5 ロードバランサから1つのクラスタを休止します
 - a. `asadmin disable-http-lb-server` を使用して、クラスタを無効にします。
 - b. `asadmin export-http-lb-config` を使用して、ロードバランサ設定ファイルをエクスポートします。

- c. エクスポートした設定ファイルを **Web** サーバーインスタンスの構成ディレクトリにコピーします。
たとえば、Sun Java System Web Server の場合、コピー先は `web-server-install-dir/https-host-name/config/loadbalancer.xml` となります。新しいロードバランサ設定ファイルが自動的にロードされるように、ロードバランサ設定の `reloadinterval` を設定して、ロードバランサの動的再設定を有効にする必要があります。
 - d. タイムアウトが経過するまで待機します。
ロードバランサのログファイルを監視して、インスタンスがオフラインであることを確認します。ユーザーに再試行 URL が表示される場合は、休止期間をスキップして、サーバーをただちに再起動します。
- 6 ほかのクラスタが実行中の間に、無効となっていたクラスタを再起動します。
再起動すると、クラスタはドメインと同期し、アプリケーションを更新します。
 - 7 再起動したクラスタ上でアプリケーションをテストし、正しく動作していることを確認します。
 - 8 ロードバランサでクラスタを有効にします。
 - a. `asadmin enable-http-lb-server` を使用して、クラスタを有効にします。
 - b. `asadmin export-http-lb-config` を使用して、ロードバランサ設定ファイルをエクスポートします。
 - c. 設定ファイルを **Web** サーバーの構成ディレクトリにコピーします。
 - 9 ほかのクラスタに対して、手順 5～8 を繰り返します。
 - 10 すべてのサーバーインスタンスに新しいアプリケーションがあり、それらのインスタンスが実行中である場合は、すべてのクラスタに対して動的再設定を再度有効にすることができます。

互換性のないアプリケーションのアップグレード

アプリケーションの新しいバージョンと古いバージョンとの間に互換性がない場合、次の手順を実行します。アプリケーションの互換性に必要な条件については、[95 ページの「アプリケーションの互換性」](#)を参照してください。互換性のないアプリケーションは、2つ以上のクラスタでアップグレードする必要があります。クラスタが1つしかない場合は、後述の説明に従って、アップグレードのための「シャドウクラスタ」を作成します。

互換性のないアプリケーションをアップグレードする場合は、次の手順を実行します。

- 新しいバージョンのアプリケーションに、古いバージョンのアプリケーションとは別の名前を付けます。このあとの手順は、アプリケーションの名前が変更されていることを前提にしています。
- データスキーマに互換性がない場合は、データの移行を計画したあとに、別の物理データソースを使用します。
- 新しいバージョンを、古いバージョンが配備されているクラスタとは別のクラスタに配備します。
- アプリケーションへの要求は新しいクラスタに処理を引き継がないため、クラスタをオフラインにする前に、古いアプリケーションを実行しているクラスタには適切な長いタイムアウトを設定します。これらのユーザーセッションは、単純に失敗します。

▼ 2番目のクラスタを作成することにより互換性のないアプリケーションをアップグレードする

- 1 旧バージョンのアプリケーションを保存するか、ドメインをバックアップします。ドメインをバックアップするには、`asadmin backup-domain` コマンドを使用します。
- 2 同じマシンセットまたは別のマシンセットに、既存のクラスタとして「シャドウクラスタ」を作成します。2番目のクラスタがすでに存在する場合、この手順はスキップします。
 - a. 管理コンソールを使用して、既存のクラスタで名前を付けられている設定から新しいクラスタと参照を作成します。
既存のアクティブポートとの競合を回避するために、各マシンで新しいインスタンスのポートをカスタマイズします。
 - b. `asadmin create-resource-ref` を使用して、クラスタに関連付けられたすべてのリソースについて、新しく作成されたクラスタにリソース参照を追加します。
 - c. `asadmin create-application-ref` を使用して、新しく作成されたクラスタから、クラスタに配備されているほかのすべてのアプリケーション(現在再配備されているアプリケーションを除く)への参照を作成します。
 - d. `asadmin configure-ha-cluster` を使用して、クラスタを高可用性に設定します。
 - e. `asadmin create-http-lb-ref` を使用して、ロードバランサ設定ファイル内の新しく作成されたクラスタへの参照を作成します。

- 3 新しいバージョンのアプリケーションに、古いバージョンとは別の名前を付けます。
- 4 新しいクラスタをターゲットとして、新しいアプリケーションを配備します。別のコンテキストルートを使用します。
- 5 `asadmin enable-http-lb-application` を使用して、クラスタに対して配備した新しいアプリケーションを有効にします。
- 6 ほかのクラスタが実行している間に、新しいクラスタを起動します。
起動すると、クラスタはドメインと同期し、新しいアプリケーションで更新されます。
- 7 新しいクラスタ上でアプリケーションをテストして、正しく動作していることを確認します。
- 8 `asadmin disable-http-lb-server` を使用して、ロードバランサから古いクラスタを無効にします。
- 9 無応答のセッションに対するタイムアウト時間を設定します。
- 10 `asadmin enable-http-lb-server` を使用して、ロードバランサから新しいクラスタを有効にします。
- 11 `asadmin export-http-lb-config` を使用して、ロードバランサ設定ファイルをエクスポートします。
- 12 エクスポートした設定ファイルを **Web** サーバーインスタンスの構成ディレクトリにコピーします。
たとえば、Sun Java System Web Server の場合、コピー先は `web-server-install-dir/https-host-name/config/loadbalancer.xml` となります。新しいロードバランサ設定ファイルが自動的にロードされるように、ロードバランサ設定の `reloadinterval` を設定して、ロードバランサの動的再設定を有効にする必要があります。
- 13 タイムアウト期間が経過するか、または古いアプリケーションのすべてのユーザーが終了したら、古いクラスタを停止し、古いアプリケーションを削除します。

HTTPロードバランサプラグインの監視

- 102 ページの「ログメッセージの設定」
- 102 ページの「ログメッセージのタイプ」
- 104 ページの「HTTPロードバランサのログの有効化」
- 105 ページの「メッセージの監視について」

ログメッセージの設定

ロードバランサプラグインは、Webサーバーのログメカニズムを使用してメッセージを書き込みます。デフォルトのログレベルは、Sun Java System Web Server (INFO)、Apache Web Server (WARN)、および Microsoft IIS (INFO) のデフォルトのログレベルに設定されています。ログレベルの FINE、FINER、および FINEST は、Webサーバーの DEBUG レベルに対応します。

これらのログメッセージは Web サーバーのログファイルに書き込まれます。これらは raw データ形式で、スクリプトを使用して解析されるかまたは表計算ドキュメントにインポートされて、必要なメトリックスを計算します。

ログメッセージのタイプ

ロードバランサプラグインは、次の種類のログメッセージを生成します。

- 102 ページの「ロードバランサコンフィギュレータログメッセージ」
- 103 ページの「要求ディスパッチおよび実行時ログメッセージ」
- 104 ページの「コンフィギュレータエラーメッセージ」

ロードバランサコンフィギュレータログメッセージ

これらのメッセージは、べき等 URL とエラーページ設定を使用している場合に記録されます。

べき等 URL のパターン設定の出力には、次の情報が含まれます。

- ログレベルが FINE に設定されている場合:
CONFxxxx: IdempotentUrlPattern configured <url-pattern> <no-of-retries> for web-module : <web-module> (意味) IdempotentUrlPattern によって、Web モジュール <web-module> に対する <url-pattern> <no-of-retries> が設定されました
- ログレベルが SEVERE に設定されている場合:
CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern> for webModule <web-module> in loadbalancer.xml. (意味) loadbalancer.xml の Web モジュール <web-module> に対するべき等 URL 要素 <url-pattern> のエントリが重複しています

- ログレベルが WARN に設定されている場合:

CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for web-module <web-module> (意味) Web モジュール <web-module> の IdempotentUrlPatternData <url-pattern> が無効です

エラーページの URL 設定の出力には、次の情報が含まれます (ログレベルが WARN に設定されている場合)。

CONFxxxx: Invalid error-url for web-module <web-module> (意味) Web モジュール <web-module> の error-url が無効です

要求ディスパッチおよび実行時ログメッセージ

これらのログメッセージは、要求が負荷分散およびディスパッチされている間に生成されます。

- 各メソッドの始まりの標準的なログの出力には、次の情報が含まれます (ログレベルが FINE に設定されている場合)。

ROUTxxxx: Executing Router method <method_name> (意味) ルーターメソッド <method_name> を実行しています

- 各メソッドの始まりのルーターログの出力には、次の情報が含まれます (ログレベルが INFO に設定されている場合)。

ROUTxxxx: Successfully Selected another ServerInstance for idempotent request <Request-URL> (意味) べき等要求 <Request-URL> に対する別の ServerInstance の選択に成功しました

- 実行時ログの出力には、次の情報が含まれます (ログレベルが INFO に設定されている場合)。

RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL> (意味) べき等の <GET/POST/HEAD> 要求 <Request-URL> を再試行しています

コンフィギュレータエラーメッセージ

これらのエラーは、参照先のカスタムエラーページがなくなっているなど、設定上の問題がある場合に表示されます。

- ログレベルが INFO に設定されている場合:

ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried (意味) 非べき等要求 <Request-URL> は、再試行されません

次に例を示します。 ROUTxxxx: Non Idempotent Request
http://sun.com/addToDB?x=11&abc=2 cannot be retried

- ログレベルが FINE に設定されている場合:

RNTMxxxx: Invalid / Missing Custom error-url / page: <error-url> for web-module: <web-module> (意味) Web モジュール <web-module> に対するカスタムエラー URL またはページ <error-url> が、無効または不明です

次に例を示します。 RNTMxxxx: Invalid / Missing Custom error-url / page:
myerror1xyz for web-module: test

HTTP ロードバランサのログの有効化

ロードバランサプラグインは、次の情報をログに記録します。

- すべての要求の開始/終了情報。
- 要求が正常ではないインスタンスから正常なインスタンスにフェイルオーバーされた際の、引き継がれた要求の情報。
- すべての健全性検査サイクルの最後にある正常ではないインスタンスのリスト。

注 - ロードバランサのログが有効になっていて、Web サーバーのログレベルが DEBUG かまたは verbose メッセージを出力するように設定されている場合、ロードバランサは Web サーバーのログファイルに HTTP セッション ID を記録します。したがって、ロードバランサプラグインをホストしている Web サーバーが DMZ 内にある場合、本稼動環境では DEBUG または同等のログレベルを使用しないでください。

ログレベル DEBUG を使用する必要がある場合は、loadbalancer.xml で require-monitor-data プロパティを false に設定して、ロードバランサのログを無効にしてください。

▼ HTTP ロードバランサのログを有効にする

- 1 Web サーバーのログオプションを設定します。この手順は、Web サーバーによって異なります。
 - **Sun Java System Web Server** の場合
サーバーの管理コンソールで、「Magnus Editor」タブを表示し、「Log Verbose」オプションを「On」に設定します。
 - **Apache Web Server** の場合は、ログレベルを DEBUG に設定します。
 - **Microsoft IIS** の場合は、sun-passthrough.properties ファイルのログレベルを FINE に設定します。
- 2 ロードバランサ設定の「監視」オプションを true に設定します。
asadmin create-http-lb-config コマンドを使用して最初にロードバランサ設定を作成する際に監視を true に設定するか、asadmin set コマンドを使用してあとから true に設定します。デフォルトでは、監視は無効になっています。

メッセージの監視について

ロードバランサプラグインのログメッセージの形式は、次のとおりです。

- HTTP 要求の開始時には、次の情報が含まれます。
RequestStart Sticky(New) <req-id> <time-stamp> <URL>
タイムスタンプの値は、1970 年 1 月 1 日からの時間がミリ秒単位で表されます。次に例を示します。
RequestStart New 123456 602983
http://austen.sun.com/Webapps-simple/servlet/Example1
- HTTP 要求の最後には、RequestExit メッセージが次のように表示されます。
RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>
<response-time> Failure-<reason for error>(incase of a failure)
次に例を示します。
RequestExit 新規 123456 603001
http://austen.sun.com/Webapps-simple/servlet/Example1 http://austen:2222 18

注 - RequestExit メッセージでは *response-time* は、要求の合計ターンアラウンドタイムをロードバランサプラグインの側からミリ秒単位で表します。

- 正常ではないインスタンスのリストは、次のとおりです。

UnhealthyInstances <cluster-id> <time-stamp> <listener-id>, <listener-id>...
次に例を示します。

UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010

- フェイルオーバーされた要求のリストは、次のとおりです。

FailedoverRequest <req-id> <time-stamp> <URL> <session-id>
<failed-over-listener-id> <unhealthy-listener-id>

次に例を示します。

FailedoverRequest 239496 705623
http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40
http://austen:4044 http://austen:4045

HTTP 負荷分散のための Web サーバーの設定

この章では、HTTP ロードバランサプラグインでサポートされる Web サーバーを設定する方法について説明します。HTTP ロードバランサプラグインは次の Web サーバーをサポートします。

- Sun Web サーバー 6.1 および 7.0 (32 ビット)
- Apache Web Server 2.2 および 2.0.x (32 ビット)
- Microsoft IIS 5.0 および 6.0 (32 ビット)

注 - HTTP ロードバランサプラグインは、64 ビットモードで動作する Web サーバーをサポートしません。

HTTP ロードバランサプラグインのインストールプログラムは、Enterprise Server のインストールプログラムの一部であり、Web サーバーの設定ファイルにいくつかの変更を行います。これらの変更内容は、使用している Web サーバーによって異なります。また、一部の Web サーバーでは、HTTP ロードバランサを適切に動作させるために手動で設定を行う必要があります。

注 - サポート対象の Web サーバーが動作しているマシン上で、ロードバランサプラグインを Sun GlassFish Enterprise Server とともに、または個別にインストールすることができます。インストール手順の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の第 1 章「[Installing Sun GlassFish Enterprise Server](#)」を参照してください。

- 108 ページの「[Sun Web サーバー の設定](#)」
- 116 ページの「[Apache Web Server の使用](#)」
- 127 ページの「[Microsoft IIS の使用](#)」

Sun Web サーバー の設定

Sun Web サーバー では、 のインストールウィザードを使用してロードバランサプラグインをインストールするとき、すべての必要な設定がインストールウィザードによって自動的に行われます。手動での設定は必要ありません。Enterprise Server に付属のロードバランサプラグインは、次のバージョンの Sun Web サーバー をサポートしています。

- Sun Web サーバー 6.1
- Sun Web サーバー 7.0

ただし、GlassFish v2.1 または HADB バンドルのない Enterprise Server を使用している場合は、http://download.java.net/javaee5/external/SunOS_X86/aslb/jars/ からロードバランサプラグインをダウンロードして、手動で設定を変更する必要があります。プラグインのインストールおよび設定方法については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』を参照してください。

▼ Sun Web サーバー を設定する

始める前に

注 - Enterprise Server では、次の手順はインストールプログラムによって自動的に実行されます。GlassFish v2.1 を使用している場合は、次の手順を手動で実行する必要があります。

- 1 **Web** サーバーインスタンスの `magnus.conf` ファイルに、次の行を追加します。

```
##BEGIN EE LB Plug-in Parameters
Init fn="load-modules"
shlib="web-server-install-dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough" Thread="no"
Init fn="init-passthrough"
##END EE LB Plug-in Parameters=
```

- 2 まだ存在しない場合、次の行を追加します。

```
Init fn="load-modules" shlib="../../../libj2eeplugin.so" shlib_flags="(global|now)"
```

- 3 ファイル `web-server-install-dir/config/obj.conf` で、文字列 `nametrans` が最初に表示される箇所の前に、次の内容を 1 行で挿入します。

```
Nametrans fn="name-trans-passthrough" name="lbplugin"
config-file="web-server-install-dir/config/loadbalancer.xml"
```

`obj.conf` ファイル内での `NameTrans` エントリの表示順序は非常に重要です。インストーラは `NameTrans` エントリを正しい場所に配置しますが、ユーザーが別の目的で `obj.conf` を編集している場合には、このエントリの順序を正しく保持する必要があります。特に、ロードバランサ情報は、`document-root` 関数の前に配置する必要があります。

ります。obj.conf ファイルの詳細については、docs.sun.com で『Sun Java System Web Server 7.0 Administrator's Configuration File Reference』を参照してください。

- 4 ファイルweb-server-install-dir/config/obj.confに次の行を追加します。

```
<Object name = "lbplugin">
ObjectType fn="force-type" type="magnus-internal/lbplugin"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/lbplugin" fn="service-passthrough"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

- 5 *web-server-install-dir/start* スクリプトを編集し、*app-server-install-dir/lib/lbplugin/lib* が含まれるように LD_LIBRARY_PATH の値を更新します。
app-server-install-dir/lib/lbplugin/lib ディレクトリには、ロードバランサプラグインに必要なバイナリが含まれています。
- 6 (省略可能) 新しいDAS ベースのロードバランサ管理を行う場合は、SSLを使用するように Web サーバーを設定します。
Web Server 6.1 での詳細な手順については、110 ページの「Sun Web サーバー 6.1 用に SSL モードで HTTP ロードバランサを設定する」を参照してください。
Web Server 7 での詳細な手順については113 ページの「Sun Web サーバー 7用に SSL モードで HTTP ロードバランサを設定する」を参照してください。
- 7 Web サーバーがまだ動作していない場合、Web サーバーを起動します。

自動適用を使用するための Sun Java System Web Server の設定

Enterprise Server 9.1 で提供される自動適用は、ロードバランサの設定を Web サーバーの構成ディレクトリにネットワーク経由で自動的に送信する機能です。以下の手順では、この機能を使用するために Sun Java System Web Server (バージョン 6 および 7) を設定する方法を説明します。

▼ SunWeb サーバー 6.1 用に SSL モードで HTTP ロードバランサを設定する

注-この節の手順は、ロードバランサプラグインの自動適用機能を使用する場合にのみ実行する必要があります。この機能は、ロードバランサプラグインの設定を Web サーバーの構成ディレクトリにネットワーク経由で自動的に送信するために役立ちます。

- 1 ブラウザを使用して **Web Server** の管理 **GUI** にアクセスし、ログインします。
- 2 サーバーインスタンスを選択して「**Manage**」をクリックします。
- 3 「**Security**」タブをクリックします。
- 4 ユーザー名とパスワードを入力して、信頼データベースを初期化します。これは、certutil コマンドまたは **GUI** のどちらかを使用して行うことができます。信頼データベースの初期化は、certutil コマンドの次のオプションを使用して実行できます。

```
certutil -N -P "https-instance-name-hostname-" -d .
```

- certutil から入力を求められたら、キーを暗号化するためのパスワードを入力します。キーの暗号化に使用するパスワードを入力します。このパスワードの長さは少なくとも 8 文字以上にし、アルファベット以外の文字を少なくとも 1 つ以上含めるようにしてください。
- 新しいパスワードを入力するように求められたら、パスワードを指定します。

- 5 次のコマンドを使用して、サンプルのローカル認証局 (**CA**) を作成します。

```
certutil -S -P "https-boqueron.virkki.com-boqueron-"  
-d .  
-n SelfCA -s "CN=Self CA,OU=virkki.com,C=US"  
-x -t "CT,CT,CT"  
-m 101 -v 99 -5
```

- a. 証明書の種類 (0 ~ 7) を入力するように求められたら、**SSL CA** を表す「5」を入力します。プロンプトがもう一度表示されたら、「9」を指定します。
- b. 「Is this a critical extension [y/n]?」 (これは重要な拡張ですか) という質問が表示されたら、「y」を指定します。

- 6 前の手順で作成したサンプルCAを使用して、証明書を生成します。

```
certutil -S -P "https-instance-name-hostname-"
-d . -n MyServerCert -s "CN=boqueron.virkki.com,C=US"
-c SelfCA -t "u,u,u" -m 102 -v 99 -5
```

- a. 証明書の種類(0～7)を入力するように求められたら、SSLサーバーを表す「1」を入力します。プロンプトがもう一度表示されたら、「9」を指定します。
- b. 「Is this a critical extension [y/n]?(これは重要な拡張ですか)」という質問が表示されたら、「y」を指定します。

- 7 「設定の変更」>「リスナーソケットを編集」をクリックして、現在のHTTPリスナーソケットを編集します。セキュリティーを有効にして、前の手順で作成して証明書を選択します。

GUIを使用しない場合は、エントリを「タグを変更して、securityの値をtrueに設定します」と読み替えてください。追加する本体の内容と終了タグを含むように、タグを変更する必要があります。タグを追加するときは、キャリッジリターンを必ず削除してください。

```
LS id="ls1" port="80" servername="$DEPLOY-INSTANCE"
defaulttvs="https-$DEPLOY-INSTANCE" ip="any" security="true"
acceptorthreads="1" blocking="false">
<SSLPARAMS servercertnickname="$HOST-DOMAIN" ssl2="off"
ssl2ciphers="-rc4,-rc 4export,-rc2,-rc2export,-desede3,-des"
ssl3="on"
tls="on"
ssl3tlsciphers="-rsa_rc4_128_sha,+rsa_rc4_128_md5,-rsa_rc4_56_sha,-rsa_rc4_40_md5
+rsa_3des_sha,+rsa_des_sha,-rsa_des_56_sha,-rsa_rc2_40_md5,
-rsa_null_md5,-fortezza,-fortezza_rc4_128_sha,-fortezza_null,
+fips_3des_sha,-fips_des_sha" tlsrollback="on"
clientauth="off"/>
</LS>
```

▼ Sun Web サーバー 6.1 用の DAS 証明書をエクスポートおよびインポートする

- 1 **Enterprise Server with HADB** バンドルを使用している場合は、次のコマンドを実行して **DAS** 証明書をエクスポートします。

```
<as home>/lib/upgrade/pk12util -d <domain root>/config -o sjsas.p12-W
<file password> -K <master password> -n slas
```

- **GlassFish v2.1** または **HADB** バンドルのない **Enterprise Server** を使用している場合は、次のコマンドを使用して **DAS** 証明書をエクスポートする必要があります。

```
<JAVA_HOME>/bin/keytool -export -rfc -alias slas -keystore
<GLASSFISH_HOME>/domains/<DOMAIN_NAME>/config/keystore.jks-file slas.rfc
```

ここで、<GLASSFISH_HOME> は Application Server のインストールディレクトリ、<DOMAIN_NAME> は証明書のエクスポート元のドメインを示します。

- 証明書ファイルを **Web** サーバーの構成ディレクトリにコピーします。

- 2 **Enterprise Server with HADB** バンドルを使用している場合、次のコマンドを使用して、**DAS** 証明書を **Web Server** インスタンスにインポートします。

```
<webserver home>/bin/https/admin/bin/pk12util-i sjsas.p12-d <webserver
home>/alias -W<file password> -K <webserver security db password> -P
<instance-name>-<hostname>-
```

```
<webserver home>/bin/https/admin/bin/certutil -M -n slas -t "TCu,Cu,Tuw"
-d alias -P <instance-name>-<hostname>-
```

このコマンドにより、Application Server CA が、クライアント証明書およびサーバー証明書の両方に署名するための信頼済み CA になります。

- **GlassFish v2.1** または **HADB** バンドルのない **Enterprise Server** を使用している場合は、**NSS** セキュリティーツールの **certutil** を使用して作成された rfc ファイルから **DAS** 証明書をインポートします。

```
<webserver_home>/bin/certutil -A -a -n slas -t "TCu,Cu,Tuw" -i slas.rfc -d alias -P <instance-name>-<hostname>-
```

<webserver_home> は、Web サーバーのインストールディレクトリです。

次のコマンドを使用することで、この証明書の存在を確認できます。このコマンドは、デフォルトのサーバー証明書を含むその他の CA 証明書とともに slas 証明書を一覧表示します。コマンドは必ず 1 行で入力してください。

```
<WS_INSTALL_ROOT>/bin/certutil -L -d
<WS_INSTALL_ROOT>/admin-server/config-store/
<DEFAULT_CONFIG_NAME>/config
```

- obj.conf に次の行が含まれていない場合は、ファイルの末尾に追加してください。**Enterprise Server with HADB**バンドルを使用している場合、この手順はインストールプログラムによって自動的に実行されます。

```
<Object ppath="*lbconfigupdate*">
PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
<Object ppath="*lbgetmonitordata*">
PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
```

- 126 ページの「設定の確認」で説明されている手順を使用することにより、上記の設定を **DAS** から確認できます。ローカル **CA** を使用する代わりに、ほかの任意の **CA** およびサーバー証明書を使用できます。その場合、前の節の手順 5 および 6 は省略できますが、ほかの **CA** から入手したサーバー証明書をインポートする必要があります。

▼ Sun Web サーバー 7 用に SSL モードで HTTP ロード バランサを設定する

- (省略可能) 次のコマンドを使用して、**NSS** データベースを作成します。**NSS** データベースが存在している場合は、この手順は必要ありません。コマンドは必ず 1 行で入力してください。

```
websERVER-install-dir/bin/certutil -N -d
/websERVER-install-dir/admin-server/config-store/config-name/config
```

指示に従って、**NSS** データベースのパスワードを入力します。

- 次のコマンドを使用して、管理サーバーを起動します。

```
websERVER-install-dir/admin-server/bin/startserv.bat
```

- 次のコマンドを使用して、自己署名証付き明書を作成します。コマンドは必ず 1 行で入力してください。

```
websERVER-install-dir/bin/wadm create-selfsigned-cert --user=
admin-user --server-name=host-name
--nickname=ServerCert --token=internal --config=config-name
```

- 次のコマンドを使用して、**HTTP** リスナーを作成します。コマンドは必ず 1 行で入力してください。

```
websERVER-install-dir/bin/wadm create-http-listener
--user=admin-user --server-name=host-name
-default-virtual-server-name=default-virtual-server-name
--listener-port=8090 --config=config-name http-listener-ssl
```

- 5 次のコマンドを使用して、SSL を有効にして証明書を割り当てます。コマンドは必ず 1 行で入力してください。

```
webserver-install-dir/bin/wadm set-ssl-prop
--user=admin-user --http-listener=http-listener-ssl
--config=config-name enabled=true server-cert-nickname=ServerCert
```

▼ Sun Web サーバー 7 用の DAS 証明書をエクスポートおよびインポートする

- 1 Enterprise Server with HADB バンドルを使用している場合は、次のコマンドを実行して、DAS 証明書をエクスポートします。

```
<as home>/lib/upgrade/pk12util -d <domain root>/config -o sjsa.p12 -W
<file password> -K <master password> -n slas
```

- GlassFish v2.1 または HADB バンドルのない Enterprise Server を使用している場合は、Java SE 5.0 セキュリティツールの **keytool** を使用し、別名「s1as」を名前に指定して DAS 証明書をエクスポートします。この作業の間、Internet RFC 1421 標準で定義されている印刷可能なエンコーディング形式で証明書をエクスポートするには、**-rfc** オプションを選択します。

コマンド行からは、次のコマンドを使用して DAS 証明書をエクスポートできません。

```
<JAVA_HOME>/bin/keytool -export -rfc -alias slas -keystore
<GLASSFISH_HOME>/domains/<DOMAIN_NAME>/config/keystore.jks-file slas.rfc
```

ここで、<GLASSFISH_HOME> は Application Server のインストールディレクトリ、<DOMAIN_NAME> は証明書のエクスポート元のドメインを示します。

- 証明書ファイルを Web サーバーの構成ディレクトリにコピーします。

- 2 Enterprise Server with HADB バンドルを使用している場合、次のコマンドを使用して、DAS 証明書を Web Server インスタンスにインポートします。

```
<webserver home>/bin/https/admin/bin/pk12util-i sjsas.p12 -d <webserver
home>/alias -W<file password> -K <webserver security db password> -P
<instance-name>-<hostname>-
```

```
<webserver home>/bin/https/admin/bin/certutil -M -n slas -t "TCu,Cu,Tuw"
-d alias -P <instance-name>-<hostname>-
```

このコマンドにより、Application Server CA が、クライアント証明書およびサーバー証明書の両方に署名するための信頼済み CA になります。

- **GlassFish v2.1** または **HADB** バンドルのない **Enterprise Server** を使用している場合は、**NSS** セキュリティーツールの **certutil** を使用して作成された rfc ファイルから **DAS** 証明書をインポートします。

```
<webserver_home>/bin/certutil -A -a -n slas -t "TC" -i slas.rfc -d
<WS_INSTALL_ROOT>/admin-server/config-store/<CONFIG_NAME>/config
```

ここで、<webserver_home> は Web サーバーのインストールディレクトリ、<CONFIG_NAME> はデフォルトの Web サーバーインスタンスに対して作成された構成名を指します。

次のコマンドを使用することで、この証明書の存在を確認できます。このコマンドは、デフォルトのサーバー証明書を含むその他の CA 証明書とともに slas 証明書を一覧表示します。必ず、コマンド全体を 1 行で入力してください。

```
<WS_INSTALL_ROOT>/bin/certutil -L -d
<WS_INSTALL_ROOT>/admin-server/config-store/
<DEFAULT_CONFIG_NAME>/config
```

この表示は、Web Server の管理コンソールを使用しても行うこともできます。証明書をインポートした先の構成(この場合はデフォルト構成)を選択してから、「証明書」タブを選択します。利用可能なすべての証明書を見るには、「認証局」サブタブを選択します。

3 Web Server 7.0 で次のように設定を変更します。

- a. <WS_INSTALL_ROOT>/admin-server/config-store/<DEFAULT_CONFIG_NAME>/config/ にある obj.conf ファイルに、次の行を追加します。

```
<Object ppath="*lbconfigupdate*">
  PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
<Object ppath="*lbgetmonitordata*">
  PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
```

- 4 構成を配備します。前の手順で示した一連の変更を行なっている間、管理コンソールによりこの構成が配備対象としてマークされます。

- a. **Web Server** の管理コンソールで、「配備保留中」のアイコンを選択します。次のように、**CLI** ユーティリティー **wadm** を使用してこの構成を配備することもできます。

```
<WS_INSTALL_ROOT>/bin/wadm deploy-config-user=<admin><DEFAULT_CONFIG_NAME>
```

<admin> は、管理者のユーザー名です。

- 5 **GlassFish DAS** からこの設定をテストして、設定された HTTP ロードバランサとの通信が SSL 経由で行われているかどうかを確認します。詳細については、[126 ページ](#)の「[設定の確認](#)」を参照してください。

Apache Web Server の使用

ロードバランサプラグインは、Apache Web Server 2.2.x および 2.0.x をサポートします。Apache Web Server を使用するには、ロードバランサプラグインをインストールする前とインストールしたあとに、特定の設定手順を実行する必要があります。また、ロードバランサプラグインのインストールによっても、Apache Web Server に追加の変更が加えられます。プラグインをインストールしてから、追加の設定手順を実行する必要があります。ロードバランサプラグインは、Apache Web Server の 32 ビットバージョンのみをサポートします。

- [116 ページ](#)の「[Apache Web Server を使用するための要件](#)」
- [117 ページ](#)の「[HTTP ロードバランサプラグインをインストールする前の Apache の設定](#)」
- [120 ページ](#)の「[インストーラによる Apache Web Server 設定に対する変更](#)」
- [120 ページ](#)の「[DAS 証明書のエクスポート](#)」
- [126 ページ](#)の「[Solaris および Linux 上での Apache の起動](#)」

Apache Web Server を使用するための要件

Apache Web Server を使用するには、インストール環境が最小要件を満たしている必要があります。

Apache では、ロードバランサプラグインに次のものがが必要です。

- openssl-0.9.7e (ソース)
- Apache 2.2.x では httpd-2.2.10 (ソース)、Apache 2.0.x では httpd-2.0.59 (ソース)
- gcc-3.3-sol9-sparc-local パッケージ (Solaris 9 SPARC の場合)
- gcc-3.3-sol9-intel-local パッケージ (Solaris 9 x86 の場合)
- プリインストールされた gcc (Solaris 10 の場合)
- flex-2.5.4a-sol9-sparc-local パッケージ (Solaris 9 SPARC の場合)
- flex-2.5.4a-sol9-intel-local パッケージ (Solaris 9 x86 の場合)
- プリインストールされた flex (Solaris 10 の場合)

ソフトウェアのソースは、<http://www.sunfreeware.com>から入手できます。

さらに、Apache をコンパイルする前に、次の操作をしてください。

- Linux プラットフォームでは、同じマシンに Sun GlassFish Enterprise Server をインストールします。
- Solaris 9 オペレーティングシステムでは、pkgadd を使用して gcc と flex をインストールします。pkgadd を使用するには、ルートアクセスが必要です。
- Solaris 9 オペレーティングシステムでは、gcc バージョン 3.3 と make が PATH に含まれており、flex がインストールされていることを確認してください。
- Solaris 10 オペレーティングシステムでは、OpenSSL 用の make を実行する前に、Solaris SPARC の場合は
/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools に、Solaris x86 の場合は /usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools に格納されている mkheaders を実行します。
- Red Hat Enterprise Unix Advanced Server 2.1 で gcc を使用している場合は、gcc 3.0 より新しいバージョンを使用する必要があります。

注 - gcc 以外の C 言語のコンパイラを使用するには、PATH 環境変数内でその C 言語のコンパイラと make ユーティリティのパスを設定します。

Apache 2.0.x への Apache Web Server Patch の適用

Apache 2.0.x にロードバランサプラグインをインストールする前に、Apache Web Server の問題 12355 に関するパッチを適用します。この問題の詳細については、http://issues.apache.org/bugzilla/show_bug.cgi?id=12355 を参照してください。このパッチは、Apache 2.0.x で自動適用機能を使用するために必要です。パッチを適用するには、次の手順に従います。

1. http-2.0.59.tar を展開し、httpd-2.0.59 ディレクトリに移動します。
2. <http://issues.apache.org/bugzilla/attachment.cgi?id=16495> からパッチをダウンロードして、12355.diff などのファイル名で保存します。
3. httpd-2.0.59/modules/ssl ディレクトリから、次のコマンドを実行します。

```
patch < 12355.diff
```

HTTP ロードバランサプラグインをインストールする前の Apache の設定

Apache ソースをコンパイルし、SSL で動作するようにビルドする必要があります。この節では、ロードバランサプラグインが実行されるように Apache Web Server を正常にコンパイルするために必要な最小要件と手順の概要について説明しま

す。これらの要件と手順は、ソフトウェアの Solaris および Linux バージョンにのみ適用されます。Apache の Windows バージョンについては、Apache の Web サイトを参照してください。

注-ここで説明する手順は、<http://httpd.apache.org/docs> に示されている手順からの抜粋です。SSL 対応の Apache の詳細なインストール方法については、この Web サイトを参照してください。

▼ SSL 対応の Apache をインストールする

始める前に Apache ソフトウェアがすでにダウンロードされ、圧縮解除されている必要があります。

1 **OpenSSL** のソースをダウンロードし、展開します。

2 **OpenSSL** をコンパイルしてビルドします。

完全なインストール手順については、OpenSSL を圧縮解除したディレクトリにある `INSTALL` という名前のファイルを参照してください。このファイルには、OpenSSL をユーザー指定の場所にインストールする方法に関する情報が含まれています。

OpenSSL の詳細については、<http://www.openssl.org/> を参照してください。

3 **Apache** をダウンロードし、展開します。

Apache は <http://httpd.apache.org> から入手できます。

4 **Apache** をコンパイルしてビルドします。ソースツリーを設定します。

a. **Apache 2.0.x** では、コマンド `cd http-2.0_x` を使用します。**Apache 2.2.x** では、コマンド `cd http-2.2_x` を使用します。

b. 次のコマンドを実行します。

```
./configure --with-ssl= OpenSSL-install-path --prefix= Apache-install-path
--enable-ssl --enable-so
```

このコマンドの中で、*x* は Apache のバージョン番号、*open-ssl-install-path* は OpenSSL がインストールされているディレクトリへの絶対パス、および *Apache-install-path* は Apache をインストールするディレクトリです。

Apache 2 サーバーが HTTPS 要求を受け入れる場合、`--enable-ssl --enable-so` オプションを使う必要があるだけです。

`--with-mpm=worker` オプションを使用してコンパイルした場合、Apache 2.0.x は動作をマルチスレッド化します。

注 - Apache 2.2 では、`--with-included-apr` オプションを使用して、バンドル版の Apache Portable Runtime (APR) を構築します。

5 Linux 2.1 上の Apache の場合は、コンパイルの前に次の手順を実行します。

- a. `src/Makefile` を開き、自動的に生成されるセクションの最後を見つけます。
- b. 自動的に生成されるセクションのあとの最初の 4 行のあとに、次の行を追加します。

```
LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c
-lsupport -lnsprwrap -lns-httpd40
LDFLAGS+= -L/application-server-install-dir/lib -L/opt/sun/private/lib
```

`-L/opt/sun/private/lib` は、Application Server を Java Enterprise System インストールの一部としてインストールした場合にのみ必要であることに注意してください。

次に例を示します。

```
## (End of automatically generated section)
##
CFLAGS=$(OPTIM) $(CFLAGS1) $(EXTRA_CFLAGS)
LIBS=$(EXTRA_LIBS) $(LIBS1)
INCLUDES=$(INCLUDES1) $(INCLUDES0) $(EXTRA_INCLUDES)
LDFLAGS=$(LDFLAGS1) $(EXTRA_LDFLAGS)
"LIBS+= -licuuc -licui18n -lnspr4 -lpthread
-lxerces-c -lsupport -lnsprwrap -lns-httpd40
LDFLAGS+= -L/application-server-install-dir /lib -L/opt/sun/private/lib
```

- c. 環境変数 `LD_LIBRARY_PATH` を設定します。
スタンドアロンのインストールでは、`as-install/lib` を追加します。

Java Enterprise System インストールでは、これを Enterprise Server に次のように設定します。`as-install/lib:opt/sun/private/lib`。

Solaris 9 を使用している場合、`LD_LIBRARY_PATH` に `/usr/local/lib` を追加します。

6 使用しているバージョンのインストール手順で説明されている方法で、Apache をコンパイルします。

詳細については、<http://httpd.apache.org/> を参照してください。

一般的な手順は次のとおりです。

- a. `make`
- b. `make install`

注 – Apache 2.2.x では、*apache-install-location/conf/httpd.conf* ファイルで、`Include conf/extra/httpd-vhosts.conf` の行をコメント解除します。

DAS 証明書のエクスポート

次のコマンドを使用して、DAS 証明書を手動でエクスポートする必要があります。

```
appserver-install-dir/lib/upgrade/certutil -L -d appserver-instance-dir/config -n slas -a -o sjsas.crt
```

この証明書は、ロードバランサプラグインのインストール時に必要になります。ロードバランサプラグインをインストールする前に、必ずこのタスクを実行してください。

インストーラによる Apache Web Server 設定に対する変更

Enterprise Server のインストールプログラムは、ロードバランサプラグインのインストール中に、Apache 設定に対して次の変更を行います。ロードバランサプラグインを手動でインストールするように選択した場合は、これらの手順を手動で実行する必要があります。インストールプログラムは、Web サーバーのルートディレクトリにある `modules` ディレクトリに、必要なファイルを展開します。

注 – ロードバランサプラグインをインストールする前に、必ず DAS 証明書をエクスポートしてください。

Apache 2.0.x の場合、インストーラは Web サーバーインスタンスの `httpd.conf` ファイルに次のエントリを追加します。

```
##BEGIN EE LB Plugin Parameters
LoadModule apachelbplugin_module modules/mod_loadbalancer.so
#AddModule mod_apache2lbplugin.cpp
<IfModule mod_apache2lbplugin.cpp>
  config-file webserver-instance/httpd/conf/loadbalancer.xml
  locale en
</IfModule>
<VirtualHost machine-ip-address>
  DocumentRoot "webserver-instance/httpd/htdocs"
  ServerName server-name
</VirtualHost>
##END EE LB Plugin Parameters
```

Apache 2.2.x の場合、インストーラは Web サーバーインスタンスの `httpd.conf` ファイルに次のエントリを追加します。

```
##BEGIN EE LB Plugin Parameters
LoadFile /usr/lib/libCstd.so.1 (For Solaris SPARC only)
LoadModule apachelbplugin_module modules/mod_loadbalancer.so
#AddModule apachelbplugin_module
<IfModule apachelbplugin_module>
    config-file Apache-install-location/conf/loadbalancer.xml
    locale en
</IfModule>
##END EE LB Plugin Parameters
```

Apache 2.2.x の場合、インストーラは Web サーバーインスタンスの `httpd-vhosts.conf` ファイルに次のエントリを追加します。

```
##BEGIN EE LB Plugin Parameters
<VirtualHost machine-ip-address>
    ServerName host-name
    DocumentRoot Apache-install-location/htdocs
</VirtualHost>
##END EE LB Plugin Parameters
```

インストーラによって行われるその他の変更により、使用中の環境で Apache の `config-file` および `ssl-config` が正しい値に設定されます。 `ssl-config` ファイルは、Apache 2.0.x の `Apache-install-location/conf/ssl.conf` または `Apache-install-location/conf/extras/httpd-ssl.conf` に格納されます。 `config` ファイルは、Apache 2.0.x および Apache 2.2.x の `Apache-install-location/conf/httpd.conf` に格納されます。変更内容の概要は次のとおりです。

- `ssl-config` では、`VirtualHost default:port` のデフォルトのホスト名とポートが、Apache がインストールされているローカルシステムのホスト名とサーバーのポート番号に置き換えられます。この変更を行わないと、ロードバランサは機能しません。Solaris Apache が起動しない可能性があり、Linux では HTTPS 要求が機能しない可能性があります。
- `ssl-config` では、`ServerName www.example.com:443` の `www.example.com` が、Apache がインストールされているローカルシステムのホスト名で置き換えられます。この変更を行わないと、セキュリティ証明書がインストールされている場合に Apache を起動すると次の警告が表示されます。

```
[warn] RSA server certificate CommonName (CN)
hostname does NOT match server name!
```

Apache に対する証明書のインストールについては、[124 ページの「Apache のセキュリティ証明書を作成する」](#)を参照してください。

- *config* では、*ServerName www.example.com:80* の *www.example.com* の部分を、Apache がインストールされているローカルシステムのホスト名で置き換えます。
この変更を行わないと、Apache を起動したときに、システムがサーバーの完全修飾ドメイン名を特定できなかったことと、重複する *VirtualHost* エントリが存在することを示す警告が表示されます。
- *apache-install-location/conf/extra/httpd-vhosts.conf* では、*ServerName www.example.com:80* の *www.example.com* が、Apache がインストールされているローカルシステムのホスト名で置き換えられます。
この変更を行わないと、Apache を起動したときに、システムがサーバーの完全修飾ドメイン名を特定できなかったことと、重複する *VirtualHost* エントリが存在することを示す警告が表示されます。

DAS 証明書のインポート

Enterprise Server のインストールプログラムは、次のタスクをユーザーに代わって実行します。

- *sjsas.crt* を *apache-install-dir/conf/ssl.crt* ディレクトリにコピーすることにより、DAS 証明書をインポートします。
- Apache 2.2.x の *apache-install-dir/conf/extra/httpd-ssl.conf* または Apache 2.0.x の *apache-install-dir/conf/httpd.conf* に、次の行を付加します。

```
<Location /lbconfigupdate>
SSLVerifyClient require
SSLVerifyDepth 1
SSLRequireSSL
SSLCACertificateFile apache-install-dir/conf/ssl.crt/sjsas.crt
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Sun Microsystems" \
and %{SSL_CLIENT_S_DN_OU} eq "Sun GlassFish Enterprise Server" \
and %{SSL_CLIENT_M_SERIAL} eq "<serial_number>" )
</Location>
<Location /getmonitordata>
SSLVerifyClient require
SSLVerifyDepth 1
SSLRequireSSL
SSLCACertificateFile apache-install-dir/conf/ssl.crt/sjsas.crt
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Sun Microsystems" \
and %{SSL_CLIENT_S_DN_OU} eq "Sun GlassFish Enterprise Server" \
and %{SSL_CLIENT_M_SERIAL} eq <serial_number> )
</Location>
```

注 - Apache 2.2.x では、*apache-install-dir* /conf/httpd.conf ファイルの Include conf/extra/httpd-ssl.conf の行が、コメント解除されていることを確認します。

serial-number の値は、DAS 証明書ファイルから生成する必要があります。 *serial-number* を生成するには、コマンド `keytool -printcert -file sjsas.crt` を使用します。このコマンドの出力の小文字をすべて大文字に変更して、*serial-number* として使用します。このコマンドは、使用しているアプリケーションサーバーの名前も表示します。

HTTP ロードバランサプラグインをインストールしたあとの Apache の設定

この節では、Apache Web Server のインストール後に必要な変更を説明します。

スティッキラウンドロビンを有効にするための httpd.conf パラメータの変更

スティッキラウンドロビン機能を使用するには、Apache 2.2.x の *apache-install-location* /conf/extra/ httpd-mpm.conf ファイル、または Apache 2.0.x の *apache-install-location* /conf/httpd.conf ファイルに次の変更を行います。

prefork MPM セクションで、パラメータ `StartServers` および `maxclients` の値が 1 に設定されていることを確認します。ほかの値に設定されている場合は、新しいセッション要求ごとに新しい Apache プロセスが生成され、ロードバランサプラグインが初期化されて、要求が同じインスタンスで受信されます。

Apache 2.2.x では、*apache-install-location* /conf/httpd.conf ファイルの次の行のコメントを解除します。

```
Include conf/extra/httpd-mpm.conf
```

ロードバランサを使用するためのセキュリティーファイルの設定

Apache Web Server は、ロードバランサプラグインと連動するために正しいセキュリティーファイルを保持している必要があります。ロードバランサは、これらのセキュリティーデータベースファイルを必要とする NSS (Network Security Service) ライブラリに依存しています。これらのセキュリティーデータベースファイルを Enterprise Server から取得する必要があるため、Enterprise Server のインストールは、Web Server によってアクセス可能な場所で実行される必要があります。

ロードバランサが機能するようにセキュリティーファイルを設定する

- Apache 2.0.x では、*Apache-install-dir* /bin/apachectl スクリプトの LD_LIBRARY_PATH に /usr/lib/mps を追加します。
- Apache 2.2.x では、*Apache-install-dir* /bin/envvars ファイルの LD_LIBRARY_PATH に /usr/lib/mps を追加します。

Apache ユーザーへのアクセス権の付与

Apache ユーザーが、*apache-install-location*/conf/ ディレクトリとそのディレクトリ内のファイルに対する必要なアクセス許可を持っていることを確認します。Apache ユーザーとは、その配下で Apache サーバーが要求に応答する UNIX ユーザーのことです。このユーザーはファイル *httpd.conf* で定義されます。root ユーザーとして Apache をインストールした場合は、*apache-install-location*/conf/*httpd.conf* 内の、Apache ユーザーおよびグループの設定に関する注意事項に目を通します。

注-ユーザーおよびグループの設定が、このディレクトリに対するセキュリティー要件を満たしていることを確認します。たとえば、このディレクトリへのアクセスを制限するには、同じユーザーグループにディレクトリの所有者として Apache ユーザーを追加します。

ロードバランサプラグインの初期化

Apache の起動時にロードバランサプラグインが初期化されることを保証するには、次のファイルに対する読み取りおよび書き込みアクセス権を Apache ユーザーに付与します。

- *apache-install-location*/conf/loadbalancer.xml
- *apache-install-location*/conf/sun-loadbalancer_1_2.dtd

自動適用を有効にするディレクトリアクセス権の変更

自動適用機能が正常に動作することを保証するには、*apache-install-location*/conf/ ディレクトリに対する読み取り、書き込み、および実行のアクセス権を Apache ユーザーに付与します。

Apache ユーザーがこのディレクトリの所有者と同じグループに属している場合は、モードを 775 に変更します。Apache ユーザーがこのディレクトリの所有者と異なるグループに属している場合は、モードを 777 に変更します。

▼ Apache のセキュリティー証明書を作成する

次の手順は、Apache で HTTPS 要求をサポートするために必要となるものです。

Apache でセキュリティー証明書を設定する方法については、http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html および http://www.modssl.org/docs/2.8/ssl_faq.html の手順を参照してください。次の手順は、これらの Web サイトから抜粋したものです。

- 1 次の環境変数を設定します。
`OPENSSL_CONF=OpenSSL-installation-directory/apps/openssl.cnf`
- 2 次のコマンドを実行して、サーバー証明書とキーを作成します。
`openssl req -new -x509 -keyout newreq.pem -out newreq.pem -days 365`
共通名を求められたら、Apache を実行する予定のホスト名を入力します。その他のすべてのプロンプトに対しては、環境ごとの特定の要件を満たす値を入力してください。
このコマンドによって `newreq.pem` が作成されます。
- 3 `openssl` コマンドを実行した場所から、新しく作成した `newreq.pem` を開きます。
- 4 **BEGIN CERTIFICATE** で始まり **END CERTIFICATE** で終わる部分の行をすべてコピーして、`Apache-install-dir /conf/ssl.crt/server.crt` に貼り付けます。次に例を示します。
-----BEGIN CERTIFICATE-----
....
...
-----END CERTIFICATE-----
- 5 **BEGIN RSA PRIVATE KEY** で始まる行から **END RSA PRIVATE KEY** で終わる行までをコピーして、`Apache-install-dir /conf/ssl.key/server.key` に貼り付けます。次に例を示します。
-----BEGIN RSA PRIVATE KEY-----
...
...
...
-----END RSA PRIVATE KEY-----
- 6 Apache 2.0.x の `Apache-install-dir /conf/ssl.conf` または Apache 2.2.x の `Apache-install-dir /conf/extra/httpd-ssl.conf` で、変数 `SSLCertificateKeyFile` と `SSLCertificateFile` が正しい値に設定されていることを確認します。
- 7 `ServerName` が `www.example.com` でないことを確認します。`ServerName` を Apache を実行する実際のホスト名にして、サーバー証明書とキーを作成するときに入力した **Common Name** と一致させます。

Solaris および Linux 上での Apache の起動

一般的に、Apache は、Enterprise Server をインストールしたユーザーと同じユーザーで起動します。次の条件にあてはまる場合は、Apache をルートとして起動する必要があります。

- Java Enterprise System ユーザーである場合。
- 1024 より小さいポート番号を使用した場合。
- Apache を起動したユーザーとは異なるユーザーとして実行する場合。

Apache を SSL モードで起動するには、次のコマンドのいずれかを使用します。

Apache 2.0.x では **apachectl startssl** または **apachectl -k start -DSSL**。Apache 2.2.x では、**apachectl start** を使用します。

必要に応じて、Apache サーバーの起動に関する最新情報を Apache の Web サイトで確認してください。

設定の確認

1. ロードバランサプラグインをインストールします。プラグインの詳細なインストール手順については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』を参照してください。インストールの間に、DAS 証明書へのパスを指定します。
2. Application Server 管理コンソールにログインし、新しいクラスタを作成します。新しいクラスタを作成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。
3. 新しい HTTP ロードバランサを作成します。ロードバランサの作成中、デバイスのホストとして Web サーバーホストを、デバイスポートとして Web サーバーの SSL ポートを指定し、前の手順で作成したクラスタをターゲットとして選択します。新しい HTTP ロードバランサを作成するための詳細な手順については、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。
4. DAS と Web サーバーの間の通信が正常に機能していることを確認するには、管理コンソールで「HTTP ロードバランサ」ノードに移動して「HTTP ロードバランサ」をクリックします。表示された「ロードバランサデバイス設定」ページで、「テスト接続」ボタンをクリックします。

ロードバランサの作成中に「変更を自動的に適用」オプションを有効にしなかった場合は、「エクスポート」タブに移動して「今すぐ適用」をクリックすることによって、ロードバランサの設定を手動でエクスポートする必要があります。
5. テスト接続に失敗した場合は、Application Server ドメインログおよび Web サーバーのログを確認して問題に対処します。また、すべての接続手順が正しく実行されたかどうかを確認します。

Microsoft IIS の使用

ロードバランサプラグインとともに Microsoft Internet Information Services (IIS) を使用するには、以降の各節の手順に従います。

▼ HTTP ロードバランサプラグインを使用するように Microsoft IIS を設定する

- 1 **Internet Services Manager** を開きます。
- 2 プラグインを有効にする **Web** サイトを選択します。
この Web サイトは通常、デフォルトの Web サイトと名付けられます。
- 3 この **Web** サイト上で右クリックして「プロパティ」を選択し、「プロパティ」ノートブックを開きます。
- 4 次の手順に従って、新しい **ISAPI** フィルタを追加します。
 - a. 「**ISAPI** フィルタ」タブを開きます。
 - b. 「追加」をクリックします。
 - c. 「フィルタ名」フィールドに、「**Enterprise Server**」と入力します。
 - d. 「実行ファイル」フィールドに、「**C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll**」と入力します。
 - e. 「了解」をクリックして、「プロパティ」ノートブックを閉じます。
- 5 新しい仮想ディレクトリを作成および設定します。
 - a. デフォルトの **Web** サイト上で右クリックして「新規」を選択し、「仮想ディレクトリ」を選択します。
「仮想ディレクトリの作成ウィザード」が開きます。
 - b. 「エイリアス」フィールドに、「**sun-passthrough**」と入力します。
 - c. 「ディレクトリ」フィールドに、「**C:\Inetpub\wwwroot\sun-passthrough**」と入力します。

- d. 「実行パーミッション」チェックボックスにチェックマークを付けます。
ほかのすべてのパーミッション関連のチェックボックスは、チェックしないでおきます。
 - e. 「完了」をクリックします。
- 6 システムの PATH 環境変数に、sun-passthrough.dll ファイルのパス、**Enterprise Server as-install/bin** および **Enterprise Server as-install/lib** を追加します。
 - 7 IIS 6.0 ユーザーである場合は、次の手順を実行して、ロードバランサの Web サービス拡張が IIS 6 で実行されるように設定します。
 - a. IIS マネージャーで、ローカルコンピュータを展開し、「Web サービス拡張」をクリックします。
 - b. 「タスク」ペインで、「新しい Web サービス拡張を追加」を選択します。
 - c. 拡張機能の名前に **Sun-Passthrough** と入力して、「追加」をクリックします。
 - d. sun-passthrough.dll へのパスを **C:\Inetpub\wwwroot\sun-passthrough** と入力します。
 - e. 「了解」をクリックします。
 - f. 「拡張機能の状態を許可に設定」を選択します。
 - 8 IIS 6.0 ユーザーの場合、ファイル **C:\inetpub\wwwroot\sun-passthrough\lb.log** を作成し、そのファイル上のグループ **IIS_WPG** に対して、**NTFS** の書き込み権および変更権を与えます。
IIS 6.0 は Worker Process Isolation モードで実行されるため、IIS サーバーはグループ **IIS_WPG** のセキュリティー権限で実行されます。
 - 9 IIS ユーザーはすべて、コンピュータを再起動します。
 - 10 Web サーバー、ロードバランサプラグイン、および **Enterprise Server** が正常に動作していることを確認します。
Web ブラウザに **http://web-server-name/web-application** と入力して、Web アプリケーションのコンテキストルートにアクセスします。ここで、*web-server-name* は Web サーバーのホスト名または IP アドレスであり、*web-application* は **C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties** ファイルに一覧表示したコンテキストルートです。

ヒント-ISAPI フィルタの状態は緑色になっているはずですが、フィルタの状態を確認するには、Web サイトの「プロパティ」ノートブックを開き、「ISAPI フィルタ」タブをクリックします。状態が緑色になっていない場合は、何らかの HTTP 要求を IIS HTTP ポートに送信してみてください。要求が失敗すれば、正常です。ISAPI フィルタの状態を再確認します。

自動的に設定される sun-passthrough プロパティ

インストーラは、`sun-passthrough.properties` 内の次のプロパティを自動的に設定します。デフォルト値は変更可能です。

プロパティ	定義	デフォルト値
<code>lb-config-file</code>	ロードバランサ設定ファイルへのパス	<code>IIS-www-root\sun-passthrough\loadbalan</code>
<code>log-file</code>	ロードバランサログファイルへのパス	<code>IIS-www-root\sun-passthrough\lb.log</code>
<code>log-level</code>	Web サーバーのログレベル	INFO

注 - 自動適用機能は現時点で、IIS との組み合わせではサポートされていません。

設定の管理

この章では、Enterprise Server における名前付きサーバー設定の追加、変更、および使用について説明します。次の節で構成されています。

- 131 ページの「設定の使用」
- 134 ページの「名前付き設定に関連した作業」

設定の使用

- 131 ページの「設定」
- 132 ページの「default-config 設定」
- 132 ページの「インスタンスまたはクラスタの作成時に作成された設定」
- 133 ページの「一意のポート番号と設定」

設定

設定とは一連のサーバー設定情報で、HTTP リスナー、ORB/IIOP リスナー、JMS ブローカ、EJB コンテナ、セキュリティー、ロギング、および監視などの設定が含まれます。アプリケーションやリソースは、名前付き設定では定義されません。

設定は管理ドメインに配置されます。ドメイン内の複数のサーバーインスタンスが同じ設定を参照したり、別個の設定を使用したりできます。

クラスタでは、クラスタのインスタンスで均質の環境が確保されるように、クラスタ内のすべてのサーバーインスタンスがクラスタの設定を継承します。

設定には数多くの必須設定情報が含まれるため、既存の名前付き設定をコピーして新しい設定を作成します。設定情報を変更しないかぎり、新規に作成された設定はコピーした設定と同じです。

クラスタまたはインスタンスが設定を使用するには3つの方法があります。

- 「スタンドアロン:」 スタンドアロンのサーバーインスタンスまたはクラスタは、ほかのサーバーインスタンスまたはクラスタと設定を共有しません。つまり、ほかのサーバーインスタンスまたはクラスタはその名前付き設定を参照しません。スタンドアロンのインスタンスまたはクラスタは、既存の設定をコピーして名前を変更することにより作成します。
- 共有: 共有サーバーインスタンスまたはクラスタは、ほかのサーバーインスタンスまたはクラスタと設定を共有します。つまり、複数のインスタンスまたはクラスタが同じ名前付き設定を参照します。共有サーバーインスタンスまたはクラスタは、既存の設定をコピーするのではなく参照することにより作成します。
- クラスタ化: クラスタ化されたサーバーインスタンスはクラスタの設定を継承します。

関連項目

- 132 ページの「default-config 設定」
- 132 ページの「インスタンスまたはクラスタの作成時に作成された設定」
- 133 ページの「一意のポート番号と設定」
- 134 ページの「名前付き設定を作成する」
- 135 ページの「名前付き設定のプロパティの編集」

default-config 設定

default-config 設定は、スタンドアロンサーバーインスタンスまたはスタンドアロンクラスタの設定を作成するテンプレートとして機能する特殊な設定です。クラスタとサーバーインスタンスは、default-config 設定を参照できません。この設定は、新しい設定を作成するためにコピーできるだけです。デフォルト設定を編集して、コピーした新しい設定が正しく初期設定されているかどうか確認します。

詳細は、次の項目を参照してください。

- 132 ページの「インスタンスまたはクラスタの作成時に作成された設定」
- 131 ページの「設定」
- 134 ページの「名前付き設定を作成する」
- 135 ページの「名前付き設定のプロパティの編集」
- 136 ページの「設定を参照するインスタンスのポート番号を編集する」

インスタンスまたはクラスタの作成時に作成された設定

新しいサーバーインスタンスまたは新しいクラスタを作成する場合は、次のどちらかを実行します。

- 既存の設定を参照します。新しい設定は追加されません。

- 既存の設定のコピーを作成します。サーバーインスタンスまたはクラスタを追加すると、新しい設定が追加されます。

デフォルトでは、`default-config` 設定からコピーした設定を使用して新しいクラスタまたはインスタンスが作成されます。別の設定からコピーするには、新規インスタンスまたはクラスタの作成時に設定を指定します。

サーバーインスタンスの場合、新しい設定には `instance_name-config` という名前が付けられます。クラスタの場合、新しい設定には `cluster-name-config` という名前が付けられます。

詳細は、次の項目を参照してください。

- 132 ページの「`default-config` 設定」
- 131 ページの「設定」
- 134 ページの「名前付き設定を作成する」
- 135 ページの「名前付き設定のプロパティの編集」

クラスタ化された設定の同期

クラスタ化された設定を作成すると、Enterprise Server によってクラスタ設定ディレクトリがドメイン管理サーバーの `domain-root/domain-dir/config/cluster-config` に作成されます。このディレクトリは、クラスタ内ですべてのインスタンスの設定を同期するために使われます。

一意のポート番号と設定

同じホストマシン上の複数のインスタンスが同じ設定を参照する場合、各インスタンスは固有のポート番号で待機する必要があります。たとえば、ポート 80 の HTTP/SIP リスナーを使用する名前付き設定を 2 つのサーバーインスタンスが参照する場合、ポートの競合により、どちらかのサーバーインスタンスが起動できなくなります。一意のポートが使用されるように、個々のサーバーインスタンスが待機するポート番号を定義するプロパティを変更します。

ポート番号に次の原則を適用します。

- 個々のサーバーインスタンスのポート番号は、最初に設定から継承されます。
- サーバーインスタンスの作成時にポートがすでに使用されている場合は、継承されたデフォルト値をインスタンスレベルでオーバーライドして、ポートの競合を防止します。
- インスタンスが設定を共有しているものと仮定します。設定にはポート番号 n があります。同じ設定を使用するマシンで新しいインスタンスを作成する場合、新しいインスタンスにはポート番号 $n+1$ が割り当てられます (使用可能な場合)。この番号が使用できない場合は、 $n+1$ の次に使用可能なポートが選択されます。
- 設定のポート番号を変更する場合、そのポート番号を継承するサーバーインスタンスは変更されたポート番号を自動的に継承します。

- インスタンスのポート番号を変更し、続いて設定のポート番号を変更する場合、インスタンスのポート番号は変更されません。
詳細は、次の項目を参照してください。
- [136 ページ](#)の「設定を参照するインスタンスのポート番号を編集する」
- [135 ページ](#)の「名前付き設定のプロパティの編集」
- [131 ページ](#)の「設定」

名前付き設定に関連した作業

- [134 ページ](#)の「名前付き設定を作成する」
- [135 ページ](#)の「名前付き設定のプロパティの編集」
- [136 ページ](#)の「設定を参照するインスタンスのポート番号を編集する」
- [137 ページ](#)の「名前付き設定のターゲットを表示する」
- [137 ページ](#)の「名前付き設定を削除する」

▼ 名前付き設定を作成する

- 1 ツリーコンポーネントで、「設定」ノードを選択します。
- 2 「設定」ページで、「新規」をクリックします。
- 3 「設定の作成」ページで、一意の設定の名前を入力します。または「新しい設定」ページで、一意の設定の名前を入力します。
- 4 設定を選択して、コピーします。

default-config 設定は、スタンドアロンサーバーインスタンスまたはスタンドアロンクラスタを作成するときに使用するデフォルトの設定です。

参考 同機能を持つ asadmin コマンド

copy-config

- 参照
- [131 ページ](#)の「設定」
 - [132 ページ](#)の「default-config 設定」
 - [135 ページ](#)の「名前付き設定のプロパティの編集」
 - [136 ページ](#)の「設定を参照するインスタンスのポート番号を編集する」
 - [137 ページ](#)の「名前付き設定のターゲットを表示する」
 - [137 ページ](#)の「名前付き設定を削除する」

名前付き設定のプロパティの編集

次の表は、設定に対して定義済みのプロパティを示しています。

あらかじめ定義されたプロパティはポート番号です。有効なポートの値は1～65535です。UNIXでは、ポート1～1024で待機するソケットを作成するには、スーパーユーザー権限が必要です。複数のサーバーインスタンスがある場合、ポート番号は一意にする必要があります。

プロパティ名	説明
HTTP_LISTENER_PORT	http-listener-1 のポート番号。
HTTP_SSL_LISTENER_PORT	http-listener-2 のポート番号。
IIOP_SSL_LISTENER_PORT	IIOP リスナー SSL が待機する IIOP 接続用の ORB リスナーポート。
IIOP_LISTENER_PORT	orb-listener-1 が待機する IIOP 接続用の ORB リスナーポート。
JMX_SYSTEM_CONNECTOR_PORT	JMX コネクタが待機するポート番号。
IIOP_SSL_MUTUALAUTH_PORT	IIOP リスナー SSL_MUTUALAUTH が待機する IIOP 接続の ORB リスナーポート。

▼ 名前付き設定のプロパティを編集する

- 1 ツリーコンポーネントで、「設定」ノードを開きます。
- 2 名前付き設定のノードを選択します。
- 3 「システムプロパティ」を選択します。
- 4 「システムプロパティの設定」ページで、動的再設定を有効にするかどうかを選択します。
有効な場合は、設定に対する変更は、サーバーを再起動することなくサーバーインスタンスに適用されます。
- 5 必要に応じて、プロパティを追加、削除、または変更します。
- 6 設定に関連するすべてのインスタンスの現在のプロパティの値を編集するには、「インスタンス値」をクリックします。

参考 同機能を持つ asadmin コマンド

set

- 参照
- 131 ページの「設定」
 - 134 ページの「名前付き設定を作成する」
 - 137 ページの「名前付き設定のターゲットを表示する」
 - 137 ページの「名前付き設定を削除する」

▼ 設定を参照するインスタンスのポート番号を編集する

名前付き設定を参照する各インスタンスは、最初にその設定からポート番号を継承します。ポート番号はシステムで一意である必要があるため、継承されたポート番号をオーバーライドする必要があります。

- 1 ツリーコンポーネントで、「設定」ノードを開きます。
- 2 名前付き設定のノードを選択します。
- 3 「システムプロパティー」を選択します。
管理コンソールに「システムプロパティーの設定」ページが表示されます。
- 4 編集するインスタンス変数の横にある「インスタンス値」をクリックします。
たとえば、HTTP-LISTENER-PORT インスタンス変数の横にある「インスタンス値」をクリックすると、その設定を参照するすべてのサーバーインスタンスの HTTP-LISTENER-PORT の値が表示されます。
- 5 必要に応じて値を変更して、「保存」をクリックします。

参考 同機能を持つ asadmin コマンド

set

- 参照
- 133 ページの「一意のポート番号と設定」
 - 131 ページの「設定」
 - 135 ページの「名前付き設定のプロパティーの編集」

▼ 名前付き設定のターゲットを表示する

「システムプロパティの設定」ページに、設定を使用するすべてのターゲットのリストが表示されます。クラスタ設定の場合、ターゲットはクラスタです。インスタンス設定の場合、ターゲットはインスタンスです。

- 1 ツリーコンポーネントで、「設定」ノードを開きます。
- 2 名前付き設定のノードを選択します。
- 3 「システムプロパティ」を選択します。
管理コンソールに「システムプロパティの設定」ページが表示されます。

- 参照
- 133 ページの「一意のポート番号と設定」
 - 131 ページの「設定」
 - 134 ページの「名前付き設定を作成する」
 - 135 ページの「名前付き設定のプロパティの編集」
 - 137 ページの「名前付き設定を削除する」

▼ 名前付き設定を削除する

- 1 ツリーコンポーネントで、「設定」ノードを選択します。
- 2 「設定」ページで、削除する名前付き設定のチェックボックスにチェックマークを付けます。
default-config 設定は削除できません。
- 3 「削除」をクリックします。

参考 同機能を持つ asadmin コマンド

```
delete-config
```

- 参照
- 131 ページの「設定」
 - 134 ページの「名前付き設定を作成する」
 - 135 ページの「名前付き設定のプロパティの編集」
 - 137 ページの「名前付き設定のターゲットを表示する」

高可用性 (HA) セッション持続性とフェイルオーバーの設定

この章では、高可用性セッション持続性の有効化と設定を行う方法について説明します。

- 139 ページの「セッション持続性とフェイルオーバーの概要」
- 141 ページの「高可用性セッション持続性の設定」
- 145 ページの「HTTP セッションフェイルオーバー」
- 150 ページの「ステートフルセッション Bean のフェイルオーバー」

セッション持続性とフェイルオーバーの概要

Enterprise Server は、HTTP セッションデータおよびステートフルセッション Bean (SFSB) セッションデータのフェイルオーバーを通して、高可用性セッション持続性を提供します。フェイルオーバーとは、サーバーインスタンスまたはハードウェアに障害が発生しても、別のサーバーインスタンスが分散セッションを引き継ぐことを意味します。

要件

分散セッションは、次の条件が満たされた場合に、複数の Sun GlassFish Enterprise Server インスタンスで動作できます。

- 各サーバーインスタンスが、同じセッション状態データにアクセスする。Enterprise Server では、HTTP セッションおよびステートフルセッション Bean のデータを格納するための、次のタイプの高可用性ストレージが用意されています。
 - クラスタ内の別のサーバーにおけるインメモリーレプリケーション。インメモリーレプリケーションは、クラスタプロファイルではデフォルトで有効です。

インメモリーレプリケーションを使用するには、「グループ管理サービス (GMS)」を有効にする必要があります。GMS の詳細については、[36 ページ](#)の「[グループ管理サービス](#)」を参照してください。

クラスタ内の複数のサーバーインスタンスが異なるマシンに配置されている場合は、次の前提条件が満たされていることを確認してください。

- GMS およびインメモリーレプリケーションが正常に機能することを保証するには、すべてのマシンが同じサブネット上に存在する必要があります。
- インメモリーレプリケーションが正常に機能することを保証するには、クラスタ内のすべてのマシンのシステムクロックができるだけ厳密に同期している必要があります。
- 高可用性データベース (HADB)。このデータベースを使用可能にする方法については、[configure-ha-cluster\(1\)](#)を参照してください。

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

- 各サーバーインスタンスに、同じ分散可能な Web アプリケーションが配備されていること。web.xml 配備記述子ファイルの web-app 要素に、distributable 要素が含まれている必要があります。
- Web アプリケーションが、高可用性セッション持続性を使用していること。分散可能でない Web アプリケーションが、高可用性セッション持続性を使用するように設定されていると、サーバーはログファイルにエラーを書き込みます。
- Web アプリケーションは、`--availabilityenabled` オプションが `true` に設定された `deploy` または `deploydir` コマンドを使用して配備されている必要があります。これらのコマンドの詳細については、[deploy\(1\)](#) および [deploydir\(1\)](#) を参照してください。

制限事項

セッションが処理を継続すると、ファイルを開くための参照やネットワーク接続はすべて失われます。アプリケーションは、この制限を念頭においてコード化する必要があります。

フェイルオーバーをサポートする分散セッションには、特定のオブジェクトしかバインドできません。サーブレット 2.4 仕様とは異なり、Sun GlassFish Enterprise Server は、フェイルオーバーがサポートされていないオブジェクト型が分散セッションにバインドされると `IllegalArgumentException` をスローしません。

フェイルオーバーをサポートする分散セッションには、次のオブジェクトをバインドできます。

- すべての EJB コンポーネントに対するローカルホームおよびオブジェクト参照。
- 共存ステートレスセッション Bean、ステートフルセッション Bean、またはエンティティ Bean の参照。
- 分散ステートレスセッション Bean、ステートフルセッション Bean、またはエンティティ Bean の参照。
- `InitialContext` および `java:comp/env` に対する JNDI コンテキスト。
- `UserTransaction` オブジェクト。ただし、失敗したインスタンスが再起動されない場合は、準備されたグローバルトランザクションはすべて失われ、正しくロールバックまたはコミットされない可能性もあります。
- 直列化可能な Java 型。

フェイルオーバーをサポートする分散セッションには、次のオブジェクト型をバインドできません。

- JDBC データソース
- Java Message Service (JMS) の `ConnectionFactory` および `Destination` オブジェクト
- JavaMail™ セッション
- 接続ファクトリ
- 管理対象オブジェクト
- Web サービス参照

一般に、これらのオブジェクトに対して、フェイルオーバーは機能しません。ただし、オブジェクトが直列化可能な場合など、フェイルオーバーが機能する場合もあります。

高可用性セッション持続性の設定

この節では、高可用性セッション持続性を設定する方法について、次のトピックとともに説明します。

- [142 ページの「高可用性セッション持続性を設定する」](#)
- [143 ページの「セッション可用性の有効化」](#)

▼ 高可用性セッション持続性を設定する

始める前に 高可用性セッション持続性は、動的配備、動的再読み込み、および自動配備とは互換性がありません。これらの機能は、本稼働環境ではなく開発環境を対象としているため、HAセッション持続性を有効にする前に無効にする必要があります。これらの機能を無効にする方法については、『[Sun GlassFish Enterprise Server 2.1 Application Deployment Guide](#)』を参照してください。

- 1 **Enterprise Server** クラスタを作成します。
詳細については、[39 ページ](#)の「**クラスタを作成する**」を参照してください。
- 2 セッション状態データの格納に **HADB** を使用している場合は、クラスタの **HADB** データベースを作成します。

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布 で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

セッション状態データの格納にクラスタ内の別のサーバーへのインメモリーレプリケーションを使用している場合は、この手順を省略します。

HADB データベースの作成の詳細については、[configure-ha-cluster\(1\)](#)を参照してください。

- 3 クラスタの負荷分散を設定します。
詳細については、[74 ページ](#)の「**HTTP 負荷分散の設定**」を参照してください。
- 4 目的のアプリケーションサーバーインスタンス、および **Web** または **EJB** コンテナの可用性を有効にします。
次に、セッション持続性の設定を行います。次の方法のうち1つを選択します。
 - 管理コンソールを使用します。[144 ページ](#)の「**サーバーインスタンスの可用性の有効化**」を参照してください。
 - `asadmin` コマンド行ユーティリティを使用します。[set\(1\)](#)および[configure-ha-persistence\(1\)](#)を参照してください。
- 5 クラスタ内の各サーバーインスタンスを再起動します。
インスタンスが現在要求を処理中の場合、インスタンスをいったん休止してから再起動して、インスタンスが処理中の要求を完了するまでの時間が十分に取れるようにします。詳細については、「**サーバーインスタンスまたはクラスタの無効化(休止)**」を参照してください。

- 6 可用性を必要とする特定の **SFSB** の可用性を有効にします。
セッション状態にチェックポイントを設定する必要のあるメソッドを選択します。153 ページの「個々の **Bean** の可用性の設定」を参照してください。
- 7 高可用性を必要とする各 **Web** モジュールを分散可能 (**distributable**) にします。
- 8 配備中に、個々のアプリケーション、**Web** モジュール、または **EJB** モジュールの可用性を有効にします。
153 ページの「個々のアプリケーションまたは **EJB** モジュールの可用性の設定」を参照してください。
管理コンソールで、可用性を有効にするチェックボックスをチェックするか、または `--availabilityenabled` オプションを `true` にして `asadmin deploy` コマンドを実行します。

セッション可用性の有効化

セッション可用性は、次の5つの異なるスコープ(高いレベルから低いレベルへの順)で有効にすることができます。

1. サーバーインスタンス。デフォルトでは有効になっています。サーバーインスタンスのセッション可用性を有効にすると、サーバーインスタンスで実行されているすべてのアプリケーションが高可用性セッション持続性を持つことができますようになります。手順については、次の節の144 ページの「サーバーインスタンスの可用性の有効化」を参照してください。
2. コンテナ (**Web** または **EJB**)。デフォルトでは有効になっています。コンテナレベルでの可用性の有効化については、次の節を参照してください。
 - 145 ページの「**Web** コンテナの可用性の設定」
 - 151 ページの「**EJB** コンテナの可用性の設定」
3. アプリケーション。デフォルトでは無効になっています。
4. スタンドアロンの **Web** または **EJB** モジュール。デフォルトでは無効になっています。
5. 個々の **SFSB**。デフォルトでは無効になっています。

可用性を指定されたスコープで有効にするには、それより上のすべてのレベルでも有効にする必要があります。たとえば、アプリケーションレベルで可用性を有効にするには、サーバーインスタンスレベルおよびコンテナレベルでも有効にする必要があります。

ある特定のレベルの可用性は、デフォルトでは1つ上のレベルに設定されます。たとえば、可用性がコンテナレベルで有効になっている場合、デフォルトではアプリケーションレベルで有効になります。

可用性がサーバーインスタンスレベルで無効になっている場合、ほかのすべてのレベルで有効にしても反映されません。可用性がサーバーインスタンスレベルで有効になっている場合、明示的に無効化しないかぎり、すべてのレベルで有効になります。

サーバーインスタンスの可用性の有効化

サーバーインスタンスの可用性を有効にするには、`asadmin set` コマンドを使用して、設定の `availability-service.availability-enabled` プロパティを `true` に設定します。

たとえば、設定の名前が `config1` の場合は、次のように指定します。

```
asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.availability-enabled="true"
```

▼ 管理コンソールを使用してサーバーインスタンスの可用性を有効にする

- 1 ツリーコンポーネントで、「設定」ノードを開きます。
- 2 編集する設定のノードを展開します。
- 3 「可用性サービス」ノードを選択します。
- 4 「可用性サービス」ページで、「可用性サービス」ボックスにチェックマークを付けて、インスタンスレベルの可用性を有効にします。

無効にするには、このボックスのチェックマークを外します。

さらに、セッションの持続性のために HADB への接続に使用する JDBC リソースを変更した場合は、ストアプール名を変更できます。詳細については、[configure-ha-cluster\(1\)](#)を参照してください。

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

- 5 「保存」ボタンをクリックします。

- 6 サーバーインスタンスを停止し、再起動します。

HTTPセッションフェイルオーバー

Java EE アプリケーションは一般に、大量のセッション状態データを保持しています。Web ショッピングカートは、セッション状態の古典的な例です。アプリケーションはまた、頻繁に必要なデータをセッションオブジェクトにキャッシュすることもできます。実際、ユーザーとの対話が多いほぼすべてのアプリケーションには、セッション状態の保持が必要になります。

Web コンテナの可用性の設定

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

HADB を使用している場合は、`asadmin configure-ha-persistence` を使用して、Web コンテナの可用性を有効にして設定します。このコマンドの詳細については、[configure-ha-persistence\(1\)](#) を参照してください。

あるいは、`asadmin set` コマンドを使用して、設定の `availability-service.web-container-availability.availability-enabled` プロパティを `true` に設定し、次に `configure-ha-persistence` を使用して必要に応じてプロパティを設定します。

注 - セッション状態データの格納にインメモリーレプリケーションを使用している場合、Web コンテナ可用性の有効化およびプロパティの設定は `asadmin set` コマンドを使用して行う必要があります。`configure-ha-persistence` コマンドは HADB に対してのみ使用できます。

たとえば、`set` コマンドを使用して次のように指定します。ここで、`config1` は設定の名前です。

```
asadmin set
config1.availability-service.web-container-availability.availability-enabled="true"
```

```
asadmin set
config1.availability-service.web-container-availability.persistence-frequency="time-based"
```

または、set および configure-ha-persistence コマンドを次のように使用します。config1 は設定名を表します。

```
asadmin set
config1.availability-service.web-container-availability.availability-enabled="true"
asadmin configure-ha-persistence --user admin --passwordfile secret.txt
--type ha
--frequency web-method
--scope modified-session
--store jdbc/hastore
--property maxSessions=1000:reapIntervalSeconds=60 cluster1
```

▼ 管理コンソールを使用して **Web** コンテナの可用性を有効にする

- 1 ツリーコンポーネントで、目的の設定を選択します。
- 2 「可用性サービス」をクリックします。
- 3 「**Web** コンテナの可用性」タブを選択します。
「可用性サービス」ボックスにチェックマークを付けて、可用性を有効にします。無効にするには、このボックスのチェックマークを外します。
- 4 次の節の [146 ページ](#) の「**Web** コンテナの可用性の設定」の説明に従って、ほかの設定を変更します。
- 5 サーバーインスタンスを再起動します。

Web コンテナの可用性の設定

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

「可用性サービス」の「Web コンテナの可用性」タブを使用すると、次の可用性設定を変更できます。

持続型: 可用性が有効化されている Web アプリケーションのセッションの持続性メカニズムを指定します。使用できる値は memory (持続性なし)、file (ファイルシステム)、ha (HADB)、および replicated (ほかのサーバー上のメモリー) です。

haセッション持続性を使用するには、HADBを設定し、有効にしておく必要があります。設定の詳細については、[configure-ha-cluster\(1\)](#)を参照してください。

Web コンテナの可用性が有効にされている場合、次の表に示されているように、デフォルトの持続型はプロファイルに応じて異なります。

プロファイル	持続型
開発者	memory
クラスタ	replicated
エンタープライズ	ha

セッションの持続性が必要となる本稼動環境では、haまたはreplicatedを使用します。memoryおよびfileの持続型は、高可用性セッション持続性を提供しません。

Web コンテナの可用性を無効にする場合、デフォルトの持続型はmemoryです。

持続性の頻度: セッション状態を格納する頻度を指定します。持続性のタイプがhaまたはreplicatedの場合にのみ適用できます。指定できる値は次のとおりです。

- web-method - セッション状態は、各 Web 要求の終了時に、クライアントに応答を返信する前に格納されます。このモードでは、障害発生時にセッション状態を完全に更新するための最良の保証が得られます。これはデフォルトの設定です。
- time-based - セッション状態が、reapIntervalSeconds ストアプロパティによって設定された頻度でバックグラウンドに格納されます。このモードでは、セッション状態が必ずしも完全に更新される保証は得られません。ただし、各要求後に状態が格納されないため、パフォーマンスが大幅に向上します。

持続性の範囲: 格納するセッションオブジェクトの範囲と、セッション状態を格納する頻度を指定します。持続性のタイプがhaまたはreplicatedの場合にのみ適用できます。使用できる値は次のとおりです。

- session - 常にすべてのセッション状態が格納されます。このモードでは、セッションデータを分散可能な Web アプリケーションに正しく格納するための最良の保証が得られます。これはデフォルトの設定です。
- modified-session - セッション状態が変更された場合、すべてのセッション状態が格納されます。HttpSession.setAttribute() または HttpSession.removeAttribute() が呼び出された場合に、セッションが変更されたと見なします。属性が変更されるたびに、必ず setAttribute() を呼び出す必要があります。これは Java EE 仕様の要件ではありませんが、このモードを正しく動作させるために必要になります。
- modified-attribute - 変更されたセッション属性だけが格納されます。このモードを正しく動作させるには、次のガイドラインに従う必要があります。
 - セッション状態が変更されるたびに、setAttribute() を呼び出します。

- 属性間で相互参照しないようにします。別個の各属性キーにあるオブジェクトグラフを直列化し、別々に格納します。別個の各キーにあるオブジェクト間に相互参照がある場合は、正常な直列化および直列化復元は行われません。
- 複数の属性間、または少なくとも読み取り専用属性と変更可能な属性間でセッション状態を分散します。

シングルサインオン状態: シングルサインオン状態の持続性を有効にするには、このボックスにチェックマークを付けます。無効にするには、このボックスのチェックマークを外します。詳細については、[149 ページの「セッションフェイルオーバーでのシングルサインオンの使用」](#)を参照してください。

HTTP セッションストア: セッションの持続性のために HADB への接続に使用する JDBC リソースを変更した場合、HTTP セッションストアを変更できます。詳細については、[configure-ha-cluster\(1\)](#)を参照してください。

個々の Web アプリケーションの可用性の設定

個々の Web アプリケーションの可用性の有効化と設定を行うには、アプリケーション配備記述子ファイル `sun-web.xml` を編集します。アプリケーションの配備記述子の設定は、Web コンテナの可用性の設定より優先されます。

`session-manager` 要素の `persistence-type` 属性によって、アプリケーションが使用するセッション持続性のタイプが決定されます。高可用性セッション持続性を有効にするには、この要素を `ha` または `replicated` に設定する必要があります。

`sun-web.xml` ファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 Application Deployment Guide](#)』の「[The sun-web.xml File](#)」を参照してください。

例

```
<sun-web-app> ...
  <session-config>
    <session-manager persistence-type="replicated">
      <manager-properties>
        <property name="persistenceFrequency" value="web-method" />
      </manager-properties>
      <store-properties>
        <property name="persistenceScope" value="session" />
      </store-properties>
    </session-manager> ...
  </session-config> ...
```

セッションフェイルオーバーでのシングルサインオンの使用

単一のアプリケーションサーバーインスタンスにおいて、ユーザーがあるアプリケーションによって一度認証されると、同じインスタンス上で動作しているほかのアプリケーションに対する個別の再認証は必要ありません。これをシングルサインオンといいます。詳細については、『[Sun GlassFish Enterprise Server 2.1 Developer's Guide](#)』の「[User Authentication for Single Sign-on](#)」を参照してください。

HTTPセッションがクラスタ内のほかのインスタンスにフェイルオーバーした場合でも、シングルサインオンが機能し続けるようにするには、インメモリーレプリケーションまたはHADBを使用して、シングルサインオン情報が持続される必要があります。シングルサインオン情報を持続させるには、最初にサーバーインスタンスとWebコンテナの可用性を有効にし、次にシングルサインオン状態のフェイルオーバーを有効にします。

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

145 ページの「[Web コンテナの可用性の設定](#)」で説明するように、シングルサインオン状態のフェイルオーバーは、管理コンソールの「可用性サービス」の「Web コンテナの可用性」タブで有効にすることができます。asadmin set コマンドを使用して、設定の

availability-service.web-container-availability.sso-failover-enabled プロパティを true に設定することもできます。

たとえば、set コマンドを使用して次のように指定します。ここで、config1 は設定の名前です。

```
asadmin set --user admin --passwordfile password.txt
--host localhost --port 4849
config1.availability-service.web-container-availability.
sso-failover-enabled="true"
```

シングルサインオングループ

単一の名前とパスワードの組み合わせによってアクセス可能なアプリケーションは、シングルサインオングループを構成します。シングルサインオングループに属するアプリケーションに対応する HTTP セッションでは、1 つのセッションがタイム

アウトになった場合でも、ほかのセッションは無効化されず、引き続き有効となります。これは、1つのセッションがタイムアウトしてもほかのセッションの可用性に影響を与えるべきではないからです。

この動作の当然の結果として、あるセッションがタイムアウトして、セッションを実行していた同じブラウザウィンドウから対応するアプリケーションにアクセスを試みる場合、再度認証を行う必要はありません。ただし、新しいセッションが作成されます。

シングルサインオングループに属するショッピングカートアプリケーションの例を挙げます。このグループにはほかに2つのアプリケーションが含まれます。ほかの2つのアプリケーションのセッションタイムアウト値は、ショッピングカートアプリケーションのセッションタイムアウト値を上回るものと仮定します。ショッピングカートアプリケーションのセッションがタイムアウトして、セッションを実行していた同じブラウザウィンドウからショッピングカートアプリケーションの実行を試みる場合、再度認証を行う必要はありません。ただし、以前のショッピングカートは失われていて、新しいショッピングカートを作成する必要があります。ほかの2つのアプリケーションは、ショッピングカートアプリケーションを実行していたセッションのタイムアウト後も変わらず動作し続けます。

同様に、ほかの2つのアプリケーションのどちらかに対応するセッションがタイムアウトしたとします。セッションを実行していた同じブラウザウィンドウからアプリケーションに接続している間は、再度認証を行う必要はありません。

注- この動作は、セッションがタイムアウトした場合にのみ当てはまります。シングルサインオンが有効になっていて、`HttpSession.invalidate()` を使用してセッションの1つを無効にする場合、シングルサインオングループに属するすべてのアプリケーションのセッションが無効になります。シングルサインオングループに属する任意のアプリケーションへのアクセスを試みる場合、再認証が必要であり、アプリケーションにアクセスするクライアントに対して新しいセッションが作成されます。

ステートフルセッション Bean のフェイルオーバー

ステートフルセッション Bean (SFSB) には、クライアント固有の状態が含まれています。クライアントとステートフルセッション Bean の間には、一対一の関係が存在します。作成時、EJB コンテナは各 SFSB に、クライアントにバインドするための一意のセッション ID を割り当てます。

サーバーインスタンスが失敗した場合に備えて、SFSB の状態を持続的なストアに保存することができます。SFSB の状態は、そのライフサイクル内のあらかじめ定義された時点で、持続性ストアに保存されます。これを、チェックポイント設定と呼び

ます。有効になっている場合、チェックポイント設定は一般に、トランザクションがロールバックする場合でも、Bean がトランザクションを完了したあとに実行されます。

ただし、SFSB が Bean 管理によるトランザクションに参加している場合、そのトランザクションは Bean メソッドの実行の途中でコミットされる可能性があります。このメソッド呼び出しの結果、Bean の状態は遷移している途中である可能性があるため、これは Bean の状態にチェックポイントを設定するのに適切なタイミングではありません。この場合、EJB コンテナは、対応するメソッドの終了時に Bean の状態にチェックポイントを設定します。ただし、メソッドの終了時に、その Bean が別のトランザクションの範囲に入っていないことが前提です。Bean 管理によるトランザクションが複数のメソッドにまたがっている場合は、後続のメソッドの終了時にアクティブなトランザクションが存在しなくなるまで、チェックポイント設定が遅延されます。

SFSB の状態は必ずしもトランザクションではなく、非トランザクションビジネスメソッドの結果として大幅に変更される可能性もあります。SFSB がこれに当てはまる場合は、[154 ページの「チェックポイントを設定するメソッドの指定」](#)で説明しているように、チェックポイントを設定するメソッドのリストを指定することができます。

分散可能 Web アプリケーションが SFSB を参照しており、その Web アプリケーションのセッションがフェイルオーバーする場合は、EJB 参照もフェイルオーバーされます。

Enterprise Server インスタンスの停止中に、セッション持続性を使用している SFSB の配備が取り消されると、持続性ストア内のセッションデータがクリアされない可能性があります。これを回避するには、Enterprise Server インスタンスが動作している間、SFSB の配備を取り消します。

EJB コンテナの可用性の設定

▼ EJB コンテナの可用性を設定する

- 1 「EJB コンテナの可用性」タブを選択します。
- 2 「可用性サービス」ボックスにチェックマークを付けます。
可用性を無効にするには、このボックスのチェックマークを外します。
- 3 [152 ページの「可用性の設定」](#)の説明に従って、ほかの設定を変更します。
- 4 「保存」ボタンをクリックします。
- 5 サーバーインスタンスを再起動します。

参考 同機能を持つ asadmin コマンド

EJB コンテナの可用性を有効にするには、`asadmin set` コマンドを使用して、設定に次の3つのプロパティを設定します。

- `availability-service.ejb-container-availability.availability-enabled`
- `availability-service.ejb-container-availability.sfsb-persistence-type`
- `availability-service.ejb-container-availability.sfsb-ha-persistence-type`

たとえば、設定の名前が `config1` の場合は、次のコマンドを使用します。

```
asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.
ejb-container-availability.availability-enabled="true"
```

```
asadmin set --user admin --passwordfile password.txt --host localhost --port
4849
config1.availability-service.
ejb-container-availability.sfsb-persistence-type="file"
```

```
asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.
ejb-container-availability.sfsb-ha-persistence-type="replicated"
```

可用性の設定

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

「可用性サービス」の「EJB コンテナの可用性」タブを使用すると、次の設定を変更できます。

HA 持続型: 可用性が有効になっている SFSB のセッション持続性と非活性化メカニズムを指定します。使用できる値は、`file` (ファイルシステム)、`ha` (HADB)、および `replicated` (ほかのサーバー上のメモリー) です。HADB がインストールされている場合、デフォルト値は `ha` です。それ以外の場合は、`replicated` です。セッションの持続性が必要となる本稼動環境では、`ha` または `replicated` を使用します。

SFSB 持続型: 可用性が有効になっていない SFSB の非活性化メカニズムを指定します。使用できる値は、file (デフォルト)、ha、および replicated です。

いずれかの持続型を file に設定すると、EJB コンテナによって非活性化されたセッション Bean が格納されるファイルシステムの場所が指定されます。ファイルシステムに対するチェックポイントはテストには有効ですが、本稼働環境には役立ちません。ストアプロパティの設定については、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。

HA 持続性によって、どのサーバーインスタンスが失敗した場合でも、サーバーインスタンスのクラスタは SFSB 状態を復元できます。HA ストアはまた、非活性化と活性化のストアとしても使用されます。SFSB 状態の持続性を必要とする本稼働環境では、このオプションを使用します。HADB の詳細については、[configure-ha-cluster\(1\)](#)を参照してください。

SFSB ストアプール名: セッションの持続性のために HADB への接続に使用する JDBC リソースを変更した場合は、SFSB ストアプール名を変更できます。詳細については、[configure-ha-cluster\(1\)](#)を参照してください。

可用性が無効の場合の **SFSB** セッションストアの設定

可用性が無効になっている場合、ローカルファイルシステムは SFSB 状態の非活性化に使用されますが、持続性には使用されません。SFSB 状態が格納される場所を変更するには、EJB コンテナのセッション格納位置の設定を変更します。ストアプロパティの設定については、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。

個々のアプリケーションまたは **EJB** モジュールの可用性の設定

配備中に、個々のアプリケーションまたは EJB モジュールの SFSB の可用性を有効にすることができます。

- 管理コンソールを使用して配備している場合は、可用性を有効にするチェックボックスをチェックします。
- `asadmin deploy` または `asadmin deploydir` コマンドを使用して配備している場合は、`--availabilityenabled` オプションを `true` に設定します。詳細については、[deploy\(1\)](#) および [deploydir\(1\)](#)を参照してください。

個々の **Bean** の可用性の設定

個々の SFSB について可用性を有効にし、チェックポイントを設定するメソッドを選択するには、`sun-ejb-jar.xml` 配備記述子ファイルを使用します。

高可用性セッション持続性を有効にするには、`ejb` 要素に `availability-enabled="true"` を設定します。SFSB キャッシュのサイズと動作を制御するには、次の要素を使用します。

- `max-cache-size`: キャッシュに保持されるセッション Bean の最大数を指定します。キャッシュがオーバーフローする (Bean の数が `max-cache-size` を超える) 場合、コンテナは一部の Bean を非活性化するか、または Bean の直列化された状態をファイルに書き出します。ファイルを作成するディレクトリは、設定 API を使用して EJB コンテナから取得されます。
- `resize-quantity`
- `cache-idle-timeout-in-seconds`
- `removal-timeout-in-seconds`
- `victim-selection-policy`

`sun-ejb-jar.xml` の詳細については、『[Sun GlassFish Enterprise Server 2.1 Application Deployment Guide](#)』の「[The sun-ejb-jar.xml File](#)」を参照してください。

例 7-1 可用性が有効になっている EJB 配備記述子の例

```
<sun-ejb-jar>
  ...
  <enterprise-beans>
    ...
    <ejb availability-enabled="true">
      <ejb-name>MySFSB</ejb-name>
    </ejb>
    ...
  </enterprise-beans>
</sun-ejb-jar>
```

チェックポイントを設定するメソッドの指定

有効になっている場合、チェックポイント設定は一般に、トランザクションがロールバックする場合でも、Bean がトランザクションを完了したあとに実行されます。Bean の状態に重要な変更をもたらす非トランザクションビジネスメソッドの終了時に、SFSB のオプションのチェックポイント設定を追加で指定するには、`sun-ejb-jar.xml` 配備記述子ファイルの `ejb` 要素にある `checkpoint-at-end-of-method` 要素を使用します。

`checkpoint-at-end-of-method` 要素内の非トランザクションメソッドは、次のいずれかになります。

- SFSB のホームインタフェースで定義された `create()` メソッド。作成の直後に、SFSB の初期状態にチェックポイントを設定する場合に使用します。

- コンテナ管理によるトランザクションのみを使用している SFSB の場合は、トランザクション属性 TX_NOT_SUPPORTED または TX_NEVER でマークされた Bean のリモートインタフェースのメソッド。
- Bean 管理によるトランザクションのみを使用している SFSB の場合は、Bean 管理によるトランザクションが起動もコミットもされないメソッド。

このリストに記述されているその他のメソッドはすべて無視されます。これらの各メソッドの呼び出しの終了時に、EJB コンテナは SFSB の状態を持続性ストアに保存します。

注 - SFSB がどのトランザクションにも参加しておらず、`checkpoint-at-end-of-method` 要素で明示的に指定されているメソッドがない場合は、この Bean に対して `availability-enabled="true"` が設定されていても、この Bean の状態にチェックポイントは設定されません。

パフォーマンスを向上させるには、メソッドの小さなサブセットを指定します。これらのメソッドは一般に、大量の処理を実行するか、または Bean の状態に重要な変更をもたらします。

例 7-2 メソッドのチェックポイント設定を指定する EJB 配備記述子の例

```
<sun-ejb-jar>
...
  <enterprise-beans>
    ...
    <ejb availability-enabled="true">
      <ejb-name>ShoppingCartEJB</ejb-name>
      <checkpoint-at-end-of-method>
        <method>
          <method-name>addToCart</method-name>
        </method>
      </checkpoint-at-end-of-method>
    </ejb>
    ...
  </enterprise-beans>
</sun-ejb-jar>
```


Java Message Service 負荷分散とフェイルオーバー

この章では、Enterprise Server で使用するために Java Message Service (JMS) の負荷分散とフェイルオーバーを設定する方法について説明します。ここで説明する内容は次のとおりです。

- 157 ページの「Java Message Service の概要」
- 158 ページの「Java Message Service の設定」
- 161 ページの「接続プールとフェイルオーバー」
- 164 ページの「MQ クラスと Enterprise Server の併用」

Java Message Service の概要

Java Message Service (JMS) API は、Java EE アプリケーションおよびコンポーネントに対して、メッセージの作成、送信、受信、および読み取りを可能にするメッセージング標準です。この API によって、緩やかに結合され、信頼性が高く、非同期の分散通信が可能となります。Sun Java System Message Queue (MQ) は JMS を実装し、Enterprise Server と密接に統合されているため、MQ を使用してメッセージ駆動型 Bean (MDB) などのコンポーネントを作成できます。

MQ は「コネクタモジュール」を使用して Enterprise Server と統合されます。コネクタモジュールはリソースアダプタとしても知られており、Java EE Connector Architecture Specification 1.5 によって定義されています。Enterprise Server に配備された Java EE コンポーネントは、コネクタモジュールを介して統合された JMS プロバイダを使用して、JMS メッセージをやり取りします。Enterprise Server で JMS リソースを作成すると、バックグラウンドでコネクタリソースが作成されます。そのようにして、JMS 操作のたびにコネクタランタイムが呼び出され、バックグラウンドで MQ リソースアダプタが使用されます。

Java Message Service は、管理コンソールまたは `asadmin` コマンド行ユーティリティから管理することができます。

詳細情報

JMS リソースの設定の詳細については、『Sun GlassFish Enterprise Server 2.1 管理ガイド』の第4章「Java Message Service (JMS) リソースの設定」を参照してください。JMSの詳細については、『Sun GlassFish Enterprise Server 2.1 Developer's Guide』の第18章「Using the Java Message Service」を参照してください。コネクタ(リソースアダプタ)の詳細については、『Sun GlassFish Enterprise Server 2.1 Developer's Guide』の第12章「Developing Connectors」を参照してください。

Sun GlassFish Message Queue の詳細については、メッセージキュードキュメント (<http://docs.sun.com/coll/1343.4>) を参照してください。JMS API の概要については、JMS Web ページ (<http://java.sun.com/products/jms/index.html>) を参照してください。

Java Message Service の設定

「Java Message Service」設定は、Sun GlassFish Enterprise Server クラスタまたはインスタンスへのすべてのインバウンドおよびアウトバウンド接続に使用できます。次にあげるものを使用して、Java Message Service を設定できます。

- 管理コンソール。関連する設定で「Java メッセージサービス」コンポーネントを開きます。詳細については、『Sun GlassFish Enterprise Server 2.1 管理ガイド』の第4章「Java Message Service (JMS) リソースの設定」を参照してください。
- `asadmin set` コマンド。次の属性を設定できます。

```
server.jms-service.init-timeout-in-seconds = 60
server.jms-service.type = LOCAL
server.jms-service.start-args =
server.jms-service.default-jms-host = default_JMS_host
server.jms-service.reconnect-interval-in-seconds = 60
server.jms-service.reconnect-attempts = 3
server.jms-service.reconnect-enabled = true
server.jms-service.addresslist-behavior = random
server.jms-service.addresslist-iterations = 3
server.jms-service.mq-scheme = mq
server.jms-service.mq-service = jms
```

次のようなプロパティーも設定できます。

```
server.jms-service.property.instance-name = imqbroker
server.jms-service.property.instance-name-suffix =
server.jms-service.property.append-version = false
```

Java Message Service のすべての属性とプロパティーを一覧表示するには、`asadmin get` コマンドを使用します。`asadmin get` の詳細については、[get\(1\)](#)を参照してください。`asadmin set` の詳細については、[set\(1\)](#)を参照してください。

JMS 接続ファクトリの設定を使用して、Java Message Service の設定をオーバーライドできます。詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[JMS 接続ファクトリ](#)」を参照してください。

注 - Java Message Service の設定を変更したあとは、Enterprise Server インスタンスを再起動する必要があります。

JMS 管理の詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の第 4 章「[Java Message Service \(JMS\) リソースの設定](#)」を参照してください。

Java Message Service の統合

MQ を Enterprise Server に統合する方法には、LOCAL、REMOTE、および EMBEDDED の 3 通りがあります。管理コンソールでは、これらのモードは Java Message Service の「タイプ」属性で表されます。

LOCAL Java Message Service

「タイプ」属性が LOCAL (クラスタインスタンスのデフォルト) の場合、Enterprise Server はデフォルト JMS ホストとして指定された MQ ブローカを起動および停止します。MQ プロセスはアウトプロセスで (別個の VM 内で) アプリケーション サーバプロセスから起動されます。Enterprise Server は、追加のポートをブローカに提供します。このポートはブローカによって、RMI レジストリを起動するために使用されます。このポート番号は、そのインスタンス用に設定された JMS ポートの番号に 100 を足した数値になります。たとえば、JMS ポート番号が 37676 である場合、この追加ポートの番号は 37776 になります。

Enterprise Server インスタンスと Message Queue ブローカの間に 1 対 1 の関係を作成するには、タイプを LOCAL に設定し、各 Enterprise Server インスタンスに異なるデフォルト JMS ホストを指定します。この作業は、クラスタが Enterprise Server と MQ のどちらかに定義されているかに関係なく行えます。

LOCAL タイプでは、「起動引数」属性を使用して MQ ブローカの起動パラメータを指定します。

REMOTE Java Message Service

「タイプ」属性が REMOTE の場合、MQ ブローカは別個に起動する必要があります。ブローカの起動については、『[Sun GlassFish Message Queue 管理ガイド](#)』を参照してください。

この場合、Enterprise Server は外部的に設定されたブローカまたはブローカクラスタを使用します。また、MQ ブローカの起動と停止は Enterprise Server とは別個に行い、MQ ツールを使用してブローカまたはブローカクラスタを設定および調整する必要があります。REMOTE タイプは Enterprise Server クラスタに最適です。

REMOTE タイプでは、MQ ツールを使用して MQ ブローカ起動パラメータを指定する必要があります。「起動引数」属性は無視されます。

EMBEDDED Java Message Service

JMS の「タイプ」属性が EMBEDDED の場合、Enterprise Server と JMS ブローカが同じ VM 内に共存し、JMS サービスはインプロセスで起動され、Enterprise Server によって管理されます。このモードでは、JMS 操作はネットワークスタックを通して行われ、パフォーマンスの最適化につながります。

JMS ホストリスト

JMS ホストは MQ ブローカを表します。Java Message Service には「JMS ホストリスト」(AddressList と呼ばれる)が含まれており、このリストには Enterprise Server が使用するすべての JMS ホストが含まれます。

JMS ホストリストには指定された MQ ブローカのホストとポートが取り込まれ、JMS ホスト設定が変更になるたびに更新されます。JMS リソースを作成するかまたは MDB を配備すると、JMS リソースや MDB は JMS ホストリストを継承します。

注 - Sun GlassFish Message Queue ソフトウェアでは、AddressList プロパティは `imqAddressList` と呼ばれています。

デフォルト JMS ホスト

JMS ホストリスト内のホストの 1 つが、`Default_JMS_host` という名前のデフォルト JMS ホストに指定されます。Enterprise Server インスタンスは、Java Message Service のタイプが LOCAL に設定されている場合に、デフォルト JMS ホストを起動します。

Sun GlassFish Message Queue ソフトウェア内にマルチブローカクラスタを作成してある場合は、デフォルト JMS ホストを削除してから、その Message Queue クラスタのブローカを JMS ホストとして追加します。この場合、デフォルト JMS ホストが JMS ホストリスト内の最初のホストになります。

Enterprise Server が Message Queue クラスタを使用する場合には、デフォルト JMS ホスト上で Message Queue 固有のコマンドが実行されます。たとえば、3 つのブローカを持つ Message Queue クラスタ用に物理送信先を作成する場合、物理送信先を作成するコマンドはデフォルトの JMS ホスト上で実行されますが、クラスタ内の 3 つのブローカすべてがその物理送信先を使用します。

JMS ホストの作成

追加の JMS ホストを、以下の方法で作成できます。

- 管理コンソールを使用します。関係する設定の「Java メッセージサービス」コンポーネントを開き、「JMS ホスト」コンポーネントを選択してから、「新規」をクリックします。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。
- `asadmin create-jms-host` コマンドを使用します。詳細については、`create-jms-host(1)`を参照してください。
JMS ホスト設定が変更されるたびに、JMS ホストリストは更新されます。

接続プールとフェイルオーバー

Enterprise Server は JMS 接続プールとフェイルオーバーをサポートします。Sun GlassFish Enterprise Server は JMS 接続を自動的にプールします。「アドレスリストの動作」属性が `random` (デフォルト) である場合、Enterprise Server は主ブローカを JMS ホストリストからランダムに選択します。フェイルオーバーが発生すると、MQ は負荷を別のブローカに透過的に転送し、JMS セマンティクスを保持します。JMS タイプが `LOCAL` タイプの場合、「アドレスリストの動作」属性のデフォルト値は `priority` です。

接続が失われたときに Enterprise Server が主ブローカへの再接続を試行するかどうかを指定するには、「再接続」チェックボックスを選択します。再接続を有効に設定した状態で、主ブローカが停止すると、Enterprise Server は JMS ホストリストにある別のブローカへの再接続を試みます。

「再接続」を有効にする場合には、以下の属性も指定します。

- アドレスリストの動作: 接続を、JMS ホストリスト内のアドレスの順序 (`priority`) とランダムな順序 (`random`) のどちらで行うかを指定します。Priority に設定すると、Java Message Service は JMS ホストリストの最初に指定された MQ ブローカに接続を試行し、そのブローカが利用できない場合にのみ別のブローカを使用します。Random に設定すると、Java Message Service は JMS ホストリストから MQ ブローカをランダムに選択します。多数のクライアントが同じ接続ファクトリを使用して接続を試行する場合は、すべてのクライアントが同じアドレスに接続しないようにこの設定を使用します。
- アドレスリストの繰り返し: 接続の確立または再確立のために、JMS ホストリストを介して Java Message Service が試行を繰り返す回数です。値 `-1` は試行回数が無制限であることを示します。
- 再接続試行: 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、JMS ホストリストに指定した各アドレスへの接続 (または再接続) を試行する回数を指定します。値 `-1` は、再試行回数が無制限であることを示します。クライアントランタイムは、接続が成功するまで最初のアドレスへの接続を試みます。

- 再接続間隔: 再接続を試行する間隔を秒数で指定します。これは、JMS ホストリストで指定した各アドレスおよびリストのそれ以降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。

これらの設定は、JMS 接続ファクトリ設定を使用してオーバーライドできます。詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[JMS 接続ファクトリ](#)」を参照してください。

負荷分散されたメッセージのインフロー

メッセージ駆動型 Bean の `sun-ejb-jar.xml` ファイル内の `activation-config-property` 要素を使用して、`jmsra` リソースアダプタの `ActivationSpec` プロパティを設定できます。メッセージ駆動型 Bean (EndPointFactory) が配備されるたびに、コネクタランタイムエンジンがこれらのプロパティを検出し、それに従ってリソースアダプタ内でそれらのプロパティを設定します。『[Sun GlassFish Enterprise Server 2.1 Application Deployment Guide](#)』の「[activation-config-property](#)」を参照してください。

Enterprise Server はメッセージが同じ ClientID を持つメッセージ駆動型 Bean にランダムに配信されることを透過的に有効にします。ClientID は永続的なサブスクライバには必須です。

ClientID が設定されない非永続サブスクライバに対しては、同じトピックをサブスクライブする特定のメッセージ駆動型 Bean のすべてのインスタンスは同等であると見なされます。メッセージ駆動型 Bean が Enterprise Server の複数のインスタンスに配備される場合、メッセージ駆動型 Bean のうちの 1 つだけがメッセージを受信します。複数の異なるメッセージ駆動型 Bean が同じトピックをサブスクライブすると、メッセージ駆動型 Bean ごとに 1 つのインスタンスがメッセージのコピーを受信します。

同じキューを使用する複数のコンシューマをサポートするには、物理送信先の `maxNumActiveConsumers` プロパティを大きい値に設定します。このプロパティを設定すると、Sun メッセージキューソフトウェアは、プロパティに設定した数までのメッセージ駆動型 Bean が同じキューからメッセージを消費することを許可します。メッセージはそれらのメッセージ駆動型 Bean にランダムに配信されます。`maxNumActiveConsumers` を -1 に設定した場合は、コンシューマの数に制限はありません。

ローカル配信が優先されることを保証するには、`addresslist-behavior` を `priority` に設定します。この設定は、`AddressList` 内の最初のブローカが最初に選択されることを指定します。この最初のブローカは、ローカルで共存するメッセージキューインスタンスです。このブローカが利用できない場合、`AddressList` 内で列挙されている順序でブローカへの接続試行が行われます。この設定は、クラスタに属する Enterprise Server インスタンスに対するデフォルトです。

注-クラスタ化機能は開発者プロファイルでは利用できません。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「プロファイル」を参照してください。

JMS サービスの高可用性

JMS コンポーネントには、次の2つのレベルの可用性があります。

- サービス可用性 - このレベルでは JMS サービスの可用性が問題になりますが、メッセージがしばらくの間利用できないかどうかは重要ではありません。サービスを提供している新規の利用可能なインスタンスに接続がフェイルオーバーされる限り、JMS コンポーネントは、そのサービスは利用可能であり正常に機能していると認識します。このレベルの可用性については、『[Sun GlassFish Enterprise Server 2.1 Developer's Guide](#)』の「[Connection Failover](#)」を参照してください。
- データ可用性 - このレベルでは、サービスの可用性と持続メッセージの両方が必須です。1回および1回限りの配信とメッセージ順序付けの JMS セマンティクスもこのレベルで扱われます。

データ可用性は、Java Message Service (JMS) に準拠する Sun GlassFish メッセージキュークラスタで有効にできます。メッセージは共通の持続性ストアに維持されます。高可用性データベース (HADB) がインストールされ、エンタープライズプロファイルが選択されている場合は、HADB からこれらのメッセージを利用できます。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「プロファイル」を参照してください。対応するブローカに対してデータ可用性を有効にする前に、Enterprise Server インスタンスに対して可用性を有効にする必要があります。

注-個別のアプリケーションおよびモジュールは、JMS の可用性を制御またはオーバーライドできません。

データ可用性を有効にするには、管理コンソールの関連する設定下で「可用性サービス」コンポーネントを選択します。「可用性サービス」ボックスにチェックマークを付けます。JMS サービスの可用性を有効にするには、「JMS の可用性」タブを選択して「可用性サービス」ボックスにチェックマークを付けます。動作の一貫性を保証するために、Enterprise Server クラスタ内のすべてのインスタンスで、インスタンス可用性および JMS 可用性の設定を統一してください。詳細については、『[Sun GlassFish Enterprise Server 2.1 高可用性 \(HA\) 管理ガイド](#)』を参照してください。

注-クラスタ化機能は開発者プロファイルでは利用できません。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「プロファイル」を参照してください。

MQ クラスタと Enterprise Server の併用

MQ Enterprise Edition は、ブローカクラスタと呼ばれる、相互に接続した複数のブローカインスタンスをサポートします。ブローカクラスタによって、クライアント接続はクラスタ内のすべてのブローカに分散されます。クラスタ化することで、水平方向のスケラビリティが提供され、可用性が向上します。

この節では、高可用性を備えた Sun Java System Message Queue クラスタを使用するために Enterprise Server を設定する方法を説明します。また、Message Queue クラスタを開始および設定する方法も解説します。

Enterprise Server および MQ 配備のトポロジの詳細については、『[Sun GlassFish Enterprise Server 2.1 配備計画ガイド](#)』の「Message Queue ブローカの配備の計画」を参照してください。

高可用性 MQ クラスタ

Sun Java System Message Queue 4.1 は、新しい「高可用性」クラスタタイプを通じて高可用性メッセージングサービスを提供します。このタイプの MQ クラスタでは、すべてのブローカインスタンスがピアツーピア関係を共有し、すべてのブローカインスタンスが共通の持続データストアを共有するため、データ可用性が提供されます。インスタンスでは、インスタンスで障害が発生したかどうかを自動的に検出し、障害が発生したブローカの持続メッセージのテイクオーバーを、テイクオーバー選出を通じて動的に実行できます。したがって、アプリケーションサーバーに配備されるアプリケーションコンポーネントはこれらの可用性機能を利用できます。

このクラスタタイプでは、キューまたは永続トピックサブスクリプションにトランザクション処理される持続メッセージの損失は一切ありません。永続的でないサブスクライバへの持続的でないメッセージまたは持続的メッセージは、クライアントランタイムが接続されているブローカが利用不能になったときに、失われる可能性があります。

ローカルモードでの高可用性ブローカクラスタの設定

1. HADB を起動します。

2. ドメインを作成して起動します。ドメインの作成には `asadmin` コマンドの `create-domain` を、そのドメインの起動には `start-domain` をそれぞれ使用します。これらのコマンドの詳細については、[create-domain\(1\)](#) および [start-domain\(1\)](#) を参照してください。
3. ノードエージェントを作成して起動します。ノードエージェントの作成には `asadmin` コマンドの `create-node-agent` を、そのエージェントの起動には `start-node-agent` をそれぞれ使用します。これらのコマンドの詳細については、[create-node-agent\(1\)](#) および [start-node-agent\(1\)](#) を参照してください。
4. クラスタを作成します。クラスタの作成は、`asadmin` コマンドの `create-cluster`、または管理コンソールを使用して行うことができます。`create-cluster` コマンドの詳細については、[create-cluster\(1\)](#) を参照してください。管理コンソールを使用してクラスタを作成する方法については、管理コンソールのオンラインヘルプを参照してください。
5. クラスタ内にインスタンスを作成します。インスタンスの作成中に、リモートブローカによって使用されている JMS プロバイダのポート番号を指定します。このポート番号を指定しない場合、デフォルトの JMS プロバイダポート番号が使用されます。インスタンスの作成は、管理コンソール、または `asadmin` コマンドの `create-instance` を使用して行うことができます。管理コンソールを使用してインスタンスを作成する方法については、管理コンソールのオンラインヘルプを参照してください。`create-instance` コマンドの詳細については、[create-instance\(1\)](#) を参照してください。
6. クラスタを起動します。これは管理コンソール、または `asadmin` コマンドの `start-cluster` を使用して行うことができます。管理コンソールを使用してクラスタを起動する方法については、管理コンソールのオンラインヘルプを参照してください。`start-cluster` コマンドの詳細については、[start-cluster\(1\)](#) を参照してください。
7. `asadmin` コマンドの `configure-ha-cluster` を使用して HA クラスタを設定します。コマンドの詳細については、[configure-ha-cluster\(1\)](#) を参照してください。

リモートモードでの高可用性ブローカクラスタの設定

1. HADB を起動します。
2. データベース表を作成します。
3. 高可用性ブローカクラスタを作成している場合、HA ドライバをコピーします。

```
cp $AS_HOME/hadb/4.4.3-6/lib/hadbjdbc4.jar $S1AS_HOME/imq/lib/ext
```

4. ドメインを作成して起動します。これを行うには、コマンド `asadmin create-domain` および `start-domain` を使用します。これらのコマンドの詳細については、[create-domain\(1\)](#) および [start-domain\(1\)](#) を参照してください。

5. ノードエージェントを作成して起動します。これを行うには、`asadmin` コマンド `create-domain` および `start-node-agent` を使用します。これらのコマンドの詳細については、[create-node-agent\(1\)](#) および [start-node-agent\(1\)](#) を参照してください。
6. クラスタを作成します。クラスタの作成は、`asadmin` コマンドの `create-cluster`、または管理コンソールを使用して行うことができます。詳細については、[create-cluster\(1\)](#) を参照してください。管理コンソールを使用してクラスタを作成する方法については、管理コンソールのオンラインヘルプを参照してください。
7. クラスタ内にインスタンスを作成します。インスタンスの作成中に、リモートブローカによって使用されている JMS プロバイダのポート番号を指定します。このポート番号を指定しない場合、デフォルトの JMS プロバイダポート番号が使用されます。
8. デフォルトの JMS ホストを削除し、インスタンスが接続できる JMS ホストを作成します。必ず、各ブローカは独立した JMS ホストとして追加してください。JMS ホストの詳細については、[160 ページの「JMS ホストリスト」](#) を参照してください。
9. JMS タイプを `Remote` に設定します。これは `asadmin` コマンドの `set` を使用して、または管理コンソールの「JMS サービス」ページから行うことができます。
10. 高可用性ブローカを設定している場合、「JMS の可用性」を `true` に設定します。これは `asadmin` コマンドの `set` を使用して、または管理コンソールの「JMS の可用性」ページから行うことができます。
11. ブローカインスタンスを起動します。
12. クラスタを起動します。詳細については、[start-cluster\(1\)](#) を参照してください。

非 HA クラスタの自動クラスタ化

これまでは、この節のあとの手順で説明されているように、「非高可用性」MQ クラスタ (マスターブローカを持つ MQ クラスタ) を管理者が個別に設定する必要がありました。このリリースでは、(REMOTE タイプの) 手動でのプロセスによる MQ クラスタの設定に加えて、Enterprise Server は「自動クラスタ化」を提供します。これは、ユーザーが Application Server クラスタを作成するときに、(LOCAL タイプの) 共存する非 HA クラスタが自動的に作成されることを意味します。これは MQ クラスタの作成のデフォルトモードになります。たとえば、3つの Application Server インスタンスを持つ Application Server クラスタを管理者が作成すると、各 Application Server インスタンスは共存するブローカと連動するように設定され、結果的に MQ クラスタが透過的に3つの MQ ブローカインスタンスを形成します。最初の Application Server インスタンスの MQ ブローカがマスターブローカに設定されます。ただし、自動クラスタ化には短所もあります。管理者がクラスタにインスタンスを追加する場

合、自動的に作成される MQ ブローカインスタンスはクラスタに参加できません。この動作は、インスタンスがクラスタから削除される場合にも適用されます。

▼ MQ クラスタと Enterprise Server クラスタの併用を有効にする

始める前に クラスタが REMOTE タイプの場合、次の手順を実行します。クラスタが LOCAL タイプの場合、手順 1～4 は該当しません。

- 1 クラスタがまだない場合、クラスタを作成します。
クラスタの作成については、39 ページの「クラスタを作成する」を参照してください。

- 2 MQ ブローカクラスタを作成します。
まず、ドメイン管理サーバーによって起動されるブローカを参照するデフォルト JMS ホストを削除してから、MQ ブローカクラスタに3つの外部ブローカ (JMS ホスト) を作成します。

JMS ホストの作成は、管理コンソールまたは `asadmin` コマンド行ユーティリティのいずれかを使用して行います。

`asadmin` を使用する場合は、たとえば次のコマンドを実行します。

```
asadmin delete-jms-host --target cluster1 default_JMS_host
asadmin create-jms-host --target cluster1
    --mqhost myhost1 --mqport 6769
    --mquser admin --mqpassword admin broker1
asadmin create-jms-host --target cluster1
    --mqhost myhost2 --mqport 6770
    --mquser admin --mqpassword admin broker2
asadmin create-jms-host --target cluster1
    --mqhost myhost3 --mqport 6771
    --mquser admin --mqpassword admin broker3
```

管理コンソールを使用してホストを作成するには、次のようにします。

- a. 「JMS ホスト」ノードに移動します(「設定」>*config-name*>「Java メッセージサービス」>「JMS ホスト」)。
- b. デフォルトのブローカ (`default_JMS_host`) を削除します。
そのブローカの横にあるチェックボックスを選択して、「削除」をクリックします。

- c. 「新規」をクリックして、各 JMS ホストを作成し、それぞれにプロパティ値を入力します。

ホスト名、DNS 名または IP アドレス、ポート番号、管理ユーザー名、パスワードの値を指定します。

- 3 マスター MQ ブローカとほかの MQ ブローカを起動します。

JMS ホストマシン上で起動する 3 つの外部ブローカに加えて、任意のマシン上で 1 つのマスターブローカを起動します。このマスターブローカは、ブローカクラスタの一部である必要はありません。次に例を示します。

```
/usr/bin/imqbrokerd -tty -name brokerm -port 6772
-cluster myhost1:6769,myhost2:6770,myhost2:6772,myhost3:6771
-D"imq.cluster.masterbroker=myhost2:6772"
```

- 4 クラスタ内のインスタンスを起動します。

- 5 クラスタ上に JMS リソースを作成します。

- a. JMS 物理送信先を作成します。

たとえば、次の `asadmin` を使用します。

```
asadmin create-jmsdest --desttype queue --target cluster1 MyQueue
asadmin create-jmsdest --desttype queue --target cluster1 MyQueue1
```

管理コンソールを使用する場合は、次のようにします。

- i. 「物理送信先」ページに移動します(「設定」>*config-name*>「Java メッセージサービス」>「物理送信先」)。

- ii. 「新規」をクリックして、各 JMS 物理送信先を作成します。

- iii. 各送信先に対して名前とタイプ(キュー)を入力します。

- b. JMS 接続ファクトリを作成します。

たとえば、次の `asadmin` を使用します。

```
asadmin create-jms-resource --target cluster1
--restype javax.jms.QueueConnectionFactory jms/MyQcf
asadmin create-jms-resource --target cluster1
--restype javax.jms.QueueConnectionFactory jms/MyQcf1
```

管理コンソールを使用する場合は、次のようにします。

- i. 「JMS 接続ファクトリ」ページに移動します(「リソース」>「JMS リソース」>「接続ファクトリ」)。

- ii. それぞれの接続ファクトリを作成するために、「新規」をクリックします。
「新しい JMS 接続ファクトリ」ページが開きます。
 - iii. 各接続ファクトリについて、「JNDI 名」(jms/MyQcf など)を入力し、「リソースタイプ」に `javax.jms.QueueConnectionFactory` を指定します。
 - iv. ページ最下部にリストされた利用可能なターゲットからクラスタを選択して、「追加」をクリックします。
 - v. 「了解」をクリックして、接続ファクトリを作成します。
- c. JMS 送信先リソースを作成します。
たとえば、次の `asadmin` を使用します。
- ```
asadmin create-jms-resource --target cluster1
--restype javax.jms.Queue
--property imqDestinationName=MyQueue jms/MyQueue
asadmin create-jms-resource --target cluster1
--restype javax.jms.Queue
--property imqDestinationName=MyQueue1 jms/MyQueue1
```
- 管理コンソールを使用する場合は、次のようにします。
- i. 「JMS 送信先リソース」ページに移動します(「リソース」>「JMS リソース」>「送信先リソース」)。
  - ii. それぞれの送信先リソースを作成するために、「新規」をクリックします。  
「新しい JMS 送信先リソース」ページが開きます。
  - iii. 各送信先リソースについて、「JNDI 名」(jms/MyQueue など)を入力し、「リソースタイプ」に `javax.jms.Queue` を指定します。
  - iv. ページ最下部にリストされた利用可能なターゲットからクラスタを選択して、「追加」をクリックします。
  - v. 「了解」をクリックして、送信先リソースを作成します。
- 6 – retrieve オプションを指定して、アプリケーションをアプリケーションクライアント用に配備します。次に例を示します。
- ```
asadmin deploy --target cluster1
--retrieve /opt/work/MQapp/mdb-simple3.ear
```
- 7 アプリケーションにアクセスして、期待どおりの動作をするかテストします。

- 8 **Enterprise Server** をデフォルトの **JMS** 設定に戻す場合は、作成した **JMS** ホストを削除して、デフォルトを作成し直します。次に例を示します。

```
asadmin delete-jms-host --target cluster1 broker1
asadmin delete-jms-host --target cluster1 broker2
asadmin delete-jms-host --target cluster1 broker3
asadmin create-jms-host --target cluster1
  --mqhost myhost1 --mqport 7676
  --mquser admin --mqpassword admin
  default_JMS_host
```

管理コンソールを使用して、これに相当する操作を実行することもできます。

注意事項 問題が起きた場合は、次の点を考慮してください。

- サーバーログファイルを表示します。このファイルの場所は、`as-install-dir/nodeagents/node-agent-name/instance-name/logs/server.log` です。MQ ブローカがメッセージに 응답しないとログファイルに記録されている場合は、ブローカを停止してから再起動します。
- ブローカログを表示します。このログの場所は、`as-install-dir/nodeagents/node-agent-name/instance-name/imq/imq-instance-name/log/log.txt` です。
- リモートの JMS タイプについては、必ず、MQ ブローカを最初に起動してからインスタンスを起動するようにしてください。
- すべての MQ ブローカが停止した場合、Java Message Service のデフォルト値では、Enterprise Server の停止または起動までに 30 分かかります。Java Message Service の値を調整して、このタイムアウトを許容できる値にしてください。次に例を示します。

```
asadmin set --user admin --password administrator
cluster1.jms-service.reconnect-interval-in-seconds=5
```

RMI-IIOP 負荷分散とフェイルオーバー

この章では、RMI-IIOP 上のリモート EJB 参照と JNDI オブジェクトに高可用性 (HA) 機能を使用する方法について説明します。

- 171 ページの「概要」
- 173 ページの「RMI-IIOP 負荷分散とフェイルオーバーの設定」

概要

RMI-IIOP 負荷分散では、IIOP クライアント要求が別のサーバーインスタンスまたはネームサーバーに分散されます。目標は、負荷をクラスタ間に均等に拡散して、スケラビリティを実現することです。また、IIOP 負荷分散を EJB のクラスタリングおよび可用性と結合すれば、EJB フェイルオーバーも実現されます。

クライアントがオブジェクトに対して JNDI 検索を実行すると、ネームサービスは、特定のサーバーインスタンスに関連付けられた `InitialContext (IC)` オブジェクトを作成します。それ以降、その IC オブジェクトを使用して作成された検索要求はすべて、同じサーバーインスタンスに送信されます。その `InitialContext` を使用して検索された `EJBHome` オブジェクトはすべて、同じターゲットサーバーにホストされます。また、それ以降に取得された `Bean` 参照もすべて、同じターゲットホスト上に作成されます。`InitialContext` オブジェクトの作成時に、ライブターゲットサーバーのリストがすべてのクライアントによってランダムに選択されるため、これにより負荷分散が効果的に実現されます。ターゲットサーバーインスタンスが停止すると、検索または EJB メソッド呼び出しは、別のサーバーインスタンスに処理が引き継がれます。

RMI-IIOP 負荷分散とフェイルオーバーは、透過的に発生します。アプリケーションの配備中に、特別な手順は必要ありません。Enterprise Server の IIOP 負荷分散およびフェイルオーバーは、クラスタの動的再設定をサポートしています。アプリケーションクライアントが配備される Enterprise Server インスタンスがクラスタに参加する場合、Enterprise Server は、クラスタ内で現在アクティブなすべての IIOP 端点を自動的に検出します。したがって、新しいインスタンスがクラスタに追加され

た、またはインスタンスがクラスタから削除された場合に、端点のリストを手動で更新する必要はありません。ただし、端点の1つで障害が発生した場合に備えて、クライアントにはブートストラップ目的で少なくとも2つの端点を指定しておくことをお勧めします。

要件

Sun GlassFish Enterprise Server は、RMI-IIOP 上で、リモート EJB 参照と NameService オブジェクトの高可用性を提供します。それには、次のすべての要件を満たしている必要があります。

- 配備に、2つ以上のインスタンスのクラスタが含まれていること。
- Java EE アプリケーションが、負荷分散にかかわるすべてのインスタンスとクラスタに対して配備されていること。
- RMI-IIOP クライアントアプリケーションで、負荷分散が有効であること。

Enterprise Server は、Application Client Container (ACC) で動作している Java アプリケーションに対する負荷分散をサポートしています。173 ページの「RMI-IIOP 負荷分散とフェイルオーバーの設定」を参照してください。

注 - Enterprise Server は、SSL (Secure Socket Layer) 上の RMI-IIOP 負荷分散とフェイルオーバーをサポートしていません。

アルゴリズム

Enterprise Server は、ランダム化とラウンドロビンのアルゴリズムを使用して、RMI-IIOP 負荷分散とフェイルオーバーを実現しています。

RMI-IIOP クライアントは最初に新しい InitialContext オブジェクトを作成すると、そのクライアントで利用可能な Enterprise Server IIOP 端点のリストが、ランダムに選ばれます。その InitialContext オブジェクトに対して、ロードバランサは、ランダムに選択されたリストの最初の端点に検索要求とほかの InitialContext 操作を命令します。最初の端点を利用できない場合、リストの2番目の端点を使用され、以下同様です。

クライアントが続けて新しい InitialContext オブジェクトを作成するたびに、端点リストがローテーションし、異なる IIOP 端点が InitialContext 操作で使われます。

InitialContext オブジェクトによって確保される参照から Beans を入手または作成する場合、それらの Beans は、InitialContext オブジェクトに割り当てられた IIOP 端点を処理する Enterprise Server インスタンスで作成されます。それらの Beans に対する参照には、クラスタ内のすべての Enterprise Server インスタンスの IIOP 端点アドレスが含まれます。

プライマリ端点は、Bean の検索または作成に使用される `InitialContext` 端点に対応する Bean 端点です。クラスタ内のほかの IIOP 端点は、代替端点として指定されています。Bean のプライマリ端点を利用できなくなると、その Bean での追加の要求は、代替端点の1つにフェイルオーバーされます。

RMI-IIOP 負荷分散とフェイルオーバーは、ACC で動作しているアプリケーションとともに動作するように設定できます。

RMI-IIOP 負荷分散とフェイルオーバーの設定

RMI-IIOP 負荷分散とフェイルオーバーは、アプリケーションクライアントコンテナ (ACC) で動作しているアプリケーション用に設定できます。重み付きラウンドロビンによる負荷分散もサポートされています。

▼ アプリケーションクライアントコンテナ用に RMI-IIOP 負荷分散を設定する

この手順は、アプリケーションクライアントコンテナ (ACC) とともに RMI-IIOP 負荷分散とフェイルオーバーを使用するために必要な手順の概要を示しています。ACC の詳細については、『[Sun GlassFish Enterprise Server 2.1 Developer's Guide](#)』の「[Developing Clients Using the ACC](#)」を参照してください。

- 1 `install_dir/bin` ディレクトリに移動します。
- 2 `package-appclient` を実行します。
このユーティリティーによって、`appclient.jar` ファイルが生成されます。`package-appclient` の詳細については、[package-appclient\(1M\)](#) を参照してください。
- 3 `appclient.jar` ファイルを、クライアントを実行するマシンにコピーして展開します。
- 4 `asenv.conf` または `asenv.bat` パス変数を編集して、そのマシン上の正しいディレクトリ値を参照するようにします。
このファイルは、`appclient-install-dir/config/` に格納されています。
更新するパス変数の一覧については、[package-appclient\(1M\)](#) を参照してください。
- 5 必要に応じて、`appclient` スクリプト実行ファイルを作成します。
たとえば、UNIX では `chmod 700` を使用します。

- 6 クラスタ内の少なくとも2つのインスタンスのIIOPリスナーポート番号を調べます。
手順7でIIOPリスナーを端点として指定します。
各インスタンスに対して、次のようにしてIIOPリスナーポートを取得します。
 - a. 管理コンソールのツリーコンポーネントで、「クラスタ」ノードを展開します。
 - b. クラスタを展開します。
 - c. クラスタ内のインスタンスを選択します。
 - d. 右の区画で、「プロパティ」タブをクリックします。
 - e. インスタンスに対するIIOPリスナーポートを記録します。
- 7 sun-acc.xml ファイルに、少なくとも2つの target-server 要素を追加します。

注-クラスタ化機能は開発者プロファイルでは利用できません。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

手順6で取得した端点を使用します。

アプリケーションクライアントが配備される Enterprise Server インスタンスがクラスタに参加する場合、ACCは、クラスタ内で現在アクティブなすべてのIIOP端点を自動的に検出します。ただし、端点の1つで障害が発生した場合に備えて、クライアントにはブートストラップ目的で少なくとも2つの端点を指定しておくことをお勧めします。

target-server 要素は、負荷分散に使用される1つまたは複数のIIOP端点を指定します。address 属性はIPv4アドレスまたはホスト名であり、port 属性はポート番号を指定します。『[Sun GlassFish Enterprise Server 2.1 Application Deployment Guide](#)』の「[client-container](#)」を参照してください。

target-server 要素を使用する代わりに、endpoints プロパティを次のように使用できます。

```
jvmarg value = "-Dcom.sun.appserv.iiop.endpoints=host1:port1,host2:port2,..."
```

- 8 重み付きラウンドロビンによる負荷分散が必要な場合、次の手順を実行します。
 - a. 各サーバーインスタンスの負荷分散の重みを設定します。

```
asadmin set instance-name.lb-weight=weight
```

- b. sun-acc.xml で、ACC の com.sun.appserv.iiop.loadbalancingpolicy プロパティを ic-based-weighted に設定します。

```
...
<client-container send-password="true">
  <property name="com.sun.appserv.iiop.loadbalancingpolicy" value="ic-based-weighted"/>
...

```

- 9 --retrieve オプションを使用してクライアントアプリケーションを配備し、クライアントの JAR ファイルを取得します。

クライアントの JAR ファイルはクライアントマシンに置いたままにします。

次に例を示します。

```
asadmin deploy --user admin --passwordfile pw.txt --retrieve /my_dir myapp
```

- 10 アプリケーションクライアントを、次のように実行します。

```
appclient -client clientjar -name appname
```

例 9-1 RMI-IIOP 重み付きラウンドロビン負荷分散に使用する負荷分散の重みの設定

この例では、3つのインスタンスを含むクラスタでの負荷分散の重みを、次の表に示すように設定します。

インスタンス名	負荷分散の重み
i1	100
i2	200
i3	300

これらの負荷分散の重みを設定するための一連のコマンドは、次のようになります。

```
asadmin set i1.lb-weight=100
asadmin set i2.lb-weight=200
asadmin set i3.lb-weight=300
```

次の手順 フェイルオーバーをテストするには、クラスタ内の1つのインスタンスを停止し、アプリケーションが正常に動作するかどうかを調べます。また、クライアントアプリケーション内にブレークポイント(またはスリープ)を設定することもできます。

負荷分散をテストするには、複数のクライアントを使用し、すべての端点にわたって負荷がどのように分散されるかを調べます。

高可用性 (HA) データベースのインストールと設定

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

この章の内容は次のとおりです。

- 177 ページの「HADB の設定の準備」
- 185 ページの「インストール」
- 187 ページの「高可用性の設定」
- 190 ページの「HADB のアップグレード」

HADB の設定の準備

ここで説明する内容は次のとおりです。

- 178 ページの「前提条件と制限」
- 178 ページの「ネットワーク冗長性の設定」
- 181 ページの「共有メモリーとセマフォの設定」
- 184 ページの「システムクロックの同期」

これらのタスクを実行したあとは、第 11 章「高可用性データベースの管理」を参照してください。

HADB の最新情報については、『[Sun GlassFish Enterprise Server 2.1 リリースノート](#)』を参照してください。

前提条件と制限

HADB を設定する前に、ネットワークおよびハードウェア環境が『[Sun GlassFish Enterprise Server 2.1 リリースノート](#)』で説明されている要件を満たしていることを確認してください。また、特定のファイルシステム (Veritas など) では制限があります。詳細については、リリースノートを参照してください。

HADB では、共有メモリーセグメントの作成およびアタッチ時に、Intimate Shared Memory (SHM_SHARE_MMU フラグ) を使用します。このフラグを使用すると、必然的に共有メモリーセグメントが物理メモリーにロックされ、ページアウトできなくなります。その結果、HADB の共有メモリーは、ローエンドのマシンでのインストールに影響しやすい物理メモリーにロックされます。Application Server と HADB を同じマシンに配置するときは、推奨メモリー量を確保する必要があります。

ネットワーク冗長性の設定

冗長性のあるネットワークを設定すると、単一のネットワーク障害が発生した場合でも、HADB を使用可能なままにできます。冗長性のあるネットワークを設定するには、次の 2 つの方法があります。

- Solaris 9 では、ネットワークマルチパスを設定します。
- Windows Server 2003 を除くすべてのプラットフォームでは、二重ネットワークを設定します。

ネットワークマルチパスの設定

ネットワークマルチパスを設定する前に、docs.sun.com で『IP ネットワークマルチパスの管理』の「ネットワークマルチパスの管理」の章を参照してください。

▼ **すでに IP マルチパスを使用している HADB ホストマシンを設定する**

- 1 ネットワークインタフェース障害検出時間を設定します。

HADB でマルチパスのフェイルオーバーを適切にサポートするには、`/etc/default/mpathd` 内の `FAILURE_DETECTION_TIME` パラメータで指定されているネットワークインタフェース障害検出時間が 1 秒 (1000 ミリ秒) を超えないようにする必要があります。元の値がこの値を超えている場合は、このファイルを編集して、このパラメータの値を 1000 に変更します。

```
FAILURE_DETECTION_TIME=1000
```

変更を有効にするには、次のコマンドを使用します。

```
pkill -HUP in.mpathd
```

2 HADB で使用する IP アドレスを設定します。

『IP Network Multipathing Administration Guide』で説明されているように、マルチパスを使用するには、物理ネットワークインタフェースをマルチパスインタフェースグループにグループ化する必要があります。このようなグループ内の各物理インタフェースには、

- データの送信に使用される物理インタフェースアドレス。
- Solaris の内部でのみ使用されるテストアドレス。

hadbm create --hosts を使用するとき、マルチパスグループの物理インタフェースアドレスを1つだけ指定します。

例 10-1 マルチパスの設定

host1 および host2 という名前の 2 つのホストマシンがあるとします。それぞれに 2 つの物理ネットワークインタフェースがある場合は、この 2 つのインタフェースをマルチパスグループとして設定します。各ホストで、ifconfig -a を実行します。

host1 の出力は次のようになります。

```
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 5 inet 129.159.115.10 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 5 inet 129.159.115.11 netmask ffffffff00 broadcast 129.159.115.255

bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 6 inet 129.159.115.12 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 6 inet 129.159.115.13 netmask ff000000 broadcast 129.159.115.255
```

host2 の出力は次のようになります。

```
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 3 inet 129.159.115.20 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3 inet 129.159.115.21 netmask ff000000 broadcast 129.159.115.255

bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 4 inet 129.159.115.22 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0
```

```
bge1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>  
mtu 1500 index 4 inet 129.159.115.23 netmask ff000000 broadcast 129.159.115.255
```

この例では、両方のホスト上の物理ネットワークインタフェースが `bge0` および `bge1` のあとに表示されています。『IP Network Multipathing Administration Guide』で説明されているように、`bge0:1` および `bge1:1` のあとに表示されているのはマルチパステストインタフェースです。これは、`ifconfig` の出力では「非推奨」として指定されています。

この環境で HADB を設定するには、各ホストから 1 つの物理インタフェースアドレスを選択します。この例において、HADB は `host1` から 129.159.115.10、`host2` から 129.159.115.20 の IP アドレスを使用しています。ホストごとに 1 つのデータベースノードを含むデータベースを作成するには、コマンド `hadbm create --hosts` を使用します。次に例を示します。

```
hadbm create --hosts 129.159.115.10,129.159.115.20
```

ホストあたり 2 つのデータベースノードを含むデータベースを作成するには、次のコマンドを使用します。

```
hadbm create --hosts 129.159.115.10,129.159.115.20,  
129.159.115.10,129.159.115.20
```

どちらの場合も、マシン上のどちらのインタフェースをエージェントが使用するべきかを指定するために、`host1` と `host2` で別のパラメータを使用してエージェントを構成する必要があります。そのため、`host1` では次のパラメータを使用します。

```
ma.server.mainternal.interfaces=129.159.115.10
```

また、`host2` では次のパラメータを使用します。

```
ma.server.mainternal.interfaces=129.159.115.20
```

`ma.server.mainternal.interfaces` 変数については、203 ページの「設定ファイル」を参照してください。

二重ネットワークの設定

HADB が単一のネットワーク障害に耐えられるようにするには、オペレーティングシステム (たとえば、Solaris) でサポートされているならば IP マルチパスを使用します。Windows Server 2003 では、二重ネットワークを使用して HADB を構成しないでください。このオペレーティングシステムは、二重ネットワークでは正常に動作しません。

オペレーティングシステムに IP マルチパスが設定されておらず、HADB ホストに 2 枚の NIC が実装されている場合は、二重ネットワークを使用するように HADB を設定できます。すべてのホストについて、各ネットワークインタフェースカード (NIC) の IP アドレスを別の IP サブネットに配置する必要があります。

データベース内では、すべてのノードを単一のネットワークに接続するか、またはすべてのノードを2つのネットワークに接続する必要があります。

注-サブネット間のルーターは、サブネット間でUDP マルチキャストメッセージを転送するように設定する必要があります。

HADB データベースを作成するときは、`--hosts` オプションを使用して、各ノードに対して2つ(各NIC IP アドレスに対して1つ)の IP アドレス または ホスト名を指定します。各ノードで、1つ目の IP アドレスは `net-0` に、2つ目の IP アドレスは `net-1` にあります。構文は次のように、同じノードのホスト名をプラス記号 (+) で区切って指定します。

```
--hosts=node0net0name+node0net1name  
,node1net0name+node1net1name  
,node2net0name+node2net1name  
, ...
```

たとえば、次の引数では2つのノードが作成され、それぞれに2つのネットワークインタフェースを持ちます。これらのノードの作成に、次のホストオプションが使用されます。

```
--hosts 10.10.116.61+10.10.124.61,10.10.116.62+10.10.124.62
```

これにより、ネットワークアドレスが次のように設定されます。

- `node0` には、`10.10.116.61` と `10.10.124.61`。
- `node1` には、`10.10.116.62` と `10.10.124.62`。

`10.10.116.61` と `10.10.116.62`、および `10.10.124.61` と `10.10.124.62` がそれぞれ同じサブネット上にあることに注目してください。

この例の場合、管理エージェントは同じサブネットを使用する必要があります。そのため、設定変数 `ma.server.mainternal.interfaces` を、たとえば、`10.10.116.0/24` に設定する必要があります。この設定は、この例の両方のエージェントに使用できます。

共有メモリーとセマフォの設定

HADB をインストールする前に、共有メモリーとセマフォを設定する必要があります。この手順は、使用しているオペレーティングシステムによって異なります。

ホストで HADB 以外のアプリケーションを実行する場合は、これらのアプリケーションが使用する共有メモリーおよびセマフォを計算し、HADB で必要な値に追加します。各ホストで動作する HADB ノードが最大6つの場合は、この節で推

奨めている値で十分です。値を増やす必要があるのは、6つを超える HADB ノードを動作させる場合、または追加の共有メモリーおよびセマフォアを必要とするアプリケーションがホストで動作している場合だけです。

セマフォアの数が必要な場合、HADB がエラーになって次のエラーメッセージが表示されることがあります。「デバイスに空き容量がありません」。これは、データベースの起動時や実行時に発生する可能性があります。

▼ Solaris で共有メモリーとセマフォアを設定する

セマフォアは、グローバルなオペレーティングシステムリソースであるため、設定は、HADB だけでなく、ホストで稼働しているすべてのプロセスに依存します。Solaris では、`/etc/system` ファイルを編集してセマフォアを設定します。

- 1 ルートとしてログインします。
- 2 共有メモリーを設定します。
 - `shminfo_shmmax` を設定します。この値は、ホスト上の単一共有メモリーセグメントの最大サイズを指定します。16 進数表現で、この値を HADB ホストマシンの RAM の合計サイズ (ただし 2G バイト以下) に設定します。
たとえば、RAM が 2G バイトの場合は、`/etc/system` ファイルで次のように値を設定します。

```
set shmsys:shminfo_shmmax=0x80000000
```

注- ホストマシンのメモリーを確認するには、次のコマンドを使用します。

```
prtconf | grep Memory
```

- Solaris 8 以前では、`shminfo_shmseg` (プロセスあたり接続可能な共有メモリーセグメントの最大数) を設定します。ホストあたりのノード数の 6 倍に値を設定します。ホストあたりに 6 ノード以下の場合、次の行を `/etc/system` ファイルに追加します。

```
set shmsys:shminfo_shmseg=36
```

Solaris 9 以降では、`shmsys:shminfo_shmseg` は廃止されています。

- `shminfo_shmmni` (システム全体の共有メモリーセグメントの最大数) を設定します。各 HADB ノードは 6 つの共有メモリーセグメントを割り当てるため、HADB に必要な値は、ホストあたりのノード数の 6 倍以上にしてください。Solaris 9 では、ホストあたりに 6 ノード以下の場合、デフォルト値を変更する必要はありません。

3 セマフォールを設定します。

`/etc/system` ファイルを確認して、たとえば次のようなセマフォール設定のエントリがないかどうか調べます。

```
set semsys:seminfo_semmni=10
set semsys:seminfo_semmns=60
set semsys:seminfo_semmnu=30
```

エントリが存在する場合は、次に示すように値を増やします。

これらのエントリが `/etc/system` ファイルに含まれていない場合は、ファイルの最後に次のエントリを追加します。

- `seminfo_semmni` (セマフォール識別子の最大数) を設定します。各 HADB ノードにはセマフォール識別子が 1 つ必要です。Solaris 9 では、ホストあたりに 6 ノード以下の場合は、デフォルト値を変更する必要はありません。次に例を示します。

```
set semsys:seminfo_semmni=10
```

- `seminfo_semmns` (システム全体のセマフォールの最大数) を設定します。各 HADB ノードにはセマフォールが 8 つ必要です。Solaris 9 の場合、またはホストあたりに 6 ノード以下の場合は、デフォルト値を変更する必要はありません。次に例を示します。

```
set semsys:seminfo_semmns=60
```

- `seminfo_semmnu` (システム中の `undo` 構造体の最大数) を設定します。各接続 (設定変数 `NumberOfSessions`、デフォルト値は 100) には `undo` 構造体が 1 つ必要です。ホストあたりに 6 ノード以下の場合は、600 に設定します。

```
set semsys:seminfo_semmnu=600
```

4 マシンをリブートします。

▼ Linux で共有メモリーを設定する

Linux では、共有メモリー設定を設定する必要があります。デフォルトのセマフォール設定を調整する必要はありません。

1 ルートとしてログインします。

2 ファイル `/etc/sysctl.conf` を編集します。

Redhat Linux では、`sysctl.conf` を変更してカーネルパラメータを設定することもできます。

3 `kernel.shmax` および `kernel.shmall` の値を次のように設定します。

```
echo MemSize > /proc/sys/shmmax
echo MemSize > /proc/sys/shmall
```

ここで、*MemSize* はバイト数です。

`kernel.shmax` パラメータは、共有メモリーセグメントの最大サイズをバイト単位で定義します。`kernel.shmall` パラメータは、システムで一度に使用できるページ内の共有メモリーの合計容量を設定します。これらの両方のパラメータの値を、マシンの物理メモリーの総量に設定します。この値は、10 進数のバイト数で指定します。

たとえば、両方の値を 2G バイトに設定するには、次のように指定します。

```
echo 2147483648 > /proc/sys/kernel/shmmax
echo 2147483648 > /proc/sys/kernel/shmall
```

- 4 次のコマンドを使用してマシンを再起動します。

sync; sync; reboot

Windows の場合の手順

Windows では、特別なシステム設定は必要ありません。ただし、既存の J2SE インストールを使用する場合は、`JAVA_HOME` 環境変数を J2SE がインストールされている場所に設定します。

システムクロックの同期

HADB はシステムクロックに基づくタイムスタンプを使用するため、HADB ホストでクロックを同期化する必要があります。HADB はシステムクロックを使用して、タイムアウトの管理や、履歴ファイルに記録されるイベントへのタイムスタンプを行います。HADB は分散システムであるため、トラブルシューティングを行うには、すべての履歴ファイルをまとめて分析する必要があります。そのため、すべてのホストのクロックが同期化されていることが重要です。

稼働中の HADB システムでは、システムクロックを調整しないでください。それを行うと、オペレーティングシステムやその他のソフトウェアコンポーネントに問題が発生し、それにより HADB ノードのハングアップや再起動などの問題が次々に引き起こされる場合があります。クロックを前に戻すと、クロックが調整されたときに一部の HADB サーバードキュメントがハングアップする場合があります。

クロックを同期化するには、次のようにします。

- Solaris では、`xntpd` (ネットワークタイムプロトコルデーモン) を使用します。
- Linux では、`ntpd` を使用します。
- Windows では、Windows の `NTPTIME` を使用します。

HADB で 1 秒を超えるクロック調整が検出されると、ノードの履歴ファイルにログ記録されます。次に例を示します。

```
NSUP INF 2003-08-26 17:46:47.975 Clock adjusted.
Leap is +195.075046 seconds.
```

インストール

一般に、HADBは、Enterprise Serverと同じシステム(共存トポロジ)、または別のホスト(分離層トポロジ)のどちらにもインストールできます。

`asadmin configure-ha-cluster` コマンドを使用して高可用性を設定できるようにするには、HADB管理クライアントをインストールする必要があります。Java Enterprise System インストーラを使用している場合は、ノードが別の層にインストールされている場合でも、管理クライアントをインストールするためにHADBインスタンス全体をインストールする必要があります。

HADBのインストール

シングルまたはデュアルCPUシステムでは、システムに少なくとも2Gバイトのメモリーがあれば、HADBとEnterprise Serverの両方をインストールできます。メモリーが不足している場合は、HADBを別のシステムにインストールするか、または追加のハードウェアを使用します。`asadmin configure-ha-cluster` コマンドを使用するには、HADBとEnterprise Serverの両方をインストールする必要があります。

各HADBノードには512Mバイトのメモリーが必要なため、2つのHADBノードを実行するにはマシンに1Gバイトのメモリーが必要です。マシンのメモリーが不足している場合は、各ノードを別のマシンに設定します。たとえば、次のシステムに2つのノードをインストールすることができます。

- メモリーがそれぞれ512Mバイト～1Gバイトの、2つのシングルCPUシステム
- メモリーが1G～2Gバイトの、シングルまたはデュアルCPUシステム

HADBは、Java Enterprise System インストーラまたはEnterprise Serverのスタンドアロンインストーラのどちらでもインストールできます。どちらのインストーラでも、「コンポーネントの選択」ページでHADB(Java ESではHigh Availability Session Storeと呼ばれる)をインストールするオプションを選択します。ホストでのインストールを完了します。Enterprise Serverのスタンドアロンインストーラを使用していて、2つの別々のマシンでHADBを実行することを選択した場合は、両方のマシンで同じインストールディレクトリを選択する必要があります。

デフォルトのインストールディレクトリ

このドキュメントの全体にわたって、`HADB_install_dir`は、HADBをインストールするディレクトリを表します。デフォルトのインストールディレクトリは、HADBをJava Enterprise Systemの一部としてインストールするかどうかによって異なります。Java Enterprise Systemの場合、デフォルトのインストールディレクトリは`/opt/SUNWhadb/4`です。スタンドアロンのApplication Serverインストーラの場合は、`/opt/SUNWappserver/hadb/4`になります。

ノードスーパーバイザープロセスの権限

ノードスーパーバイザープロセス (NSUP) は、「I'm alive」メッセージを互いに交換することにより、HADB の可用性を保証します。NSUP 実行可能ファイルは、できるだけ迅速に応答できるように、root 権限を持っている必要があります。clu_nsup_srv プロセスは CPU リソースを大量に消費せず、フットプリントも小さいため、リアルタイムプライオリティーで実行してもパフォーマンスには影響しません。

注 - Java Enterprise System インストーラを使用した場合は、NSUP の権限が自動的に正しく設定されるため、それ以上の操作は必要ありません。ただし、スタンドアロン Enterprise Server の (ルートでない) インストーラを使用する場合は、データベースを作成する前に、この権限を手動で設定する必要があります。

権限が不足している場合の症状

NSUP の権限が正しく設定されていない場合は、次のようなリソース枯渇の症状がみられることもあります。

- 誤ったネットワークパーティションや、ノードの再起動。その前に、HADB 履歴ファイルに「Process blocked for *n* seconds」という警告が記録されます。
- トランザクションの中止や、その他の例外。

制限事項

NSUP がリアルタイムプライオリティーを設定できない場合、Solaris および Linux では EPERM に errno が設定されます。Windows の場合は、「Could not set real-time priority」という警告が発行されます。ma.log ファイルにエラーが書き込まれ、プロセスはリアルタイムプライオリティーがない状態で継続されます。

次の場合は、リアルタイムプライオリティーを設定できません。

- HADB が Solaris 10 の非大域ゾーンにインストールされている場合
- Solaris 10 で、PRIV_PROC_LOCK_MEMORY (プロセスが物理メモリー内のページをロックできる) 特権または PRIV_PROC_PRIORCTL 特権、あるいはその両方が無効になっている場合
- ユーザーが setuid アクセス権を無効にした場合
- ユーザーがソフトウェアを tar ファイルとしてインストールした場合 (Application Server での、ルートでないインストールオプション)

▼ ノードスーパーバイザープロセスに **root** 権限を許可する

- 1 ルートとしてログインします。
- 2 作業用ディレクトリを `HADB_install_dir/lib/server` に変更します。
NSUP 実行可能ファイルは `clu_nsup_srv` です。
- 3 次のコマンドを使用して、ファイルの `suid` ビットを設定します。

```
chmod u+s clu_nsup_srv
```
- 4 次のコマンドを使用して、ファイルの所有者をルートに設定します。

```
chown root clu_nsup_srv
```

これにより、`clu_nsup_srv` プロセスがルートとして起動され、プロセス自身にリアルタイムプライオリティーを許可できるようになります。

セキュリティへの影響を回避するために、リアルタイムプライオリティーはプロセスが起動されるとすぐに設定され、プライオリティーが変更されたらプロセスは実効 UID に戻ります。ほかの HADB プロセスは、標準のプライオリティーで実行されます。

高可用性の設定

この節では、高可用性クラスタを作成し、HTTP セッション持続性をテストするための手順について説明します。

この節では、次の項目について説明します。

- 187 ページの「高可用性に対応するシステムを用意する」
- 188 ページの「HADB 管理エージェントの起動」
- 188 ページの「高可用性のためのクラスタの設定」
- 189 ページの「高可用性のためのアプリケーションの設定」
- 189 ページの「クラスタの再起動」
- 189 ページの「Web Server の再起動」
- 190 ページの「ロードバランサとして機能している Web Server インスタンスをクリーンアップする」

▼ 高可用性に対応するシステムを用意する

- 1 **Application Server** インスタンスとロードバランサプラグインをインストールします。
詳細については、『Java Enterprise System インストールガイド』(Java ES を使用している場合) または 『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』 (スタンドアロンの Enterprise Server インストーラを使用している場合) を参照してください。

- 2 **Enterprise Server** ドメインおよびクラスタを作成します。
ドメインの作成方法については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「ドメインの作成」を参照してください。クラスタの作成方法については、[39 ページ](#)の「クラスタを作成する」を参照してください。
- 3 **Web** サーバーソフトウェアをインストールおよび設定します。
- 4 負荷分散をセットアップおよび設定します。
詳細については、[74 ページ](#)の「**HTTP 負荷分散の設定**」を参照してください。

HADB 管理エージェントの起動

管理エージェント (ma) は、HADB ホストで管理コマンドを実行するとともに、HADB ノードスーパーバイザープロセスが失敗した場合は再起動することによってその可用性を保証します。

管理エージェントは2とおりの方法で起動できます。

- サービスとして本稼働環境で使用する場合。[196 ページ](#)の「サービスとしての管理エージェントの起動」を参照してください。管理エージェントの可用性を保証するには、システムの再起動時に管理エージェントが自動的に再起動される必要があります。[198 ページ](#)の「管理エージェントの自動再起動の実現」を参照してください。
- コンソールモードで、評価、テスト、または開発における通常のプロセスとして使用する場合。[200 ページ](#)の「コンソールモードでの管理エージェントの起動」を参照してください。

いずれの場合も、使用しているのが **Java Enterprise System** であるかスタンドアロン **Application Server** であるかによって、手順が異なります。

高可用性のためのクラスタの設定

この節の操作を開始する前に、1つ以上の **Enterprise Server** クラスタが作成されている必要があります。クラスタの作成方法については、[39 ページ](#)の「クラスタを作成する」を参照してください。

ドメイン管理サーバーが稼働しているマシンで、次のコマンドを使用して、HADB を使用するようクラスタを設定します。

```
asadmin configure-ha-cluster --user admin --hosts hadb_hostname1  
,hadb_hostname2 [,...] --devicesize 256 clusterName
```

hadb_hostname1、*hadb_hostname2*などをHADBが稼働している各マシンのホスト名に、*clusterName*をクラスタの名前に置き換えます。次に例を示します。

```
asadmin configure-ha-cluster --user admin --hosts host1,host2,host1,host2
--devicesize 256 cluster1
```

この例では、各マシンに2つのノードが作成され、HADBのフェールオーバー時でも高可用性が確保されます。`-hosts` オプションに続くホスト名の順序は重要であるため、この例は `--hosts host1,host1,host2,host2` とは異なります。

1つのマシンだけを使用している場合は、そのホスト名を2回指定する必要があります。本稼働の設定では、複数のマシンを使用することをお勧めします。

高可用性のためのアプリケーションの設定

管理コンソールで、「アプリケーション」>「エンタープライズアプリケーション」の下のアプリケーションを選択します。可用性を有効にするチェックボックスをチェックし、「保存」をクリックします。

クラスタの再起動

管理コンソールでクラスタを再起動するには、「クラスタ」>「*cluster-name*」を選択します。「インスタンスの停止」をクリックします。インスタンスが停止したら、「インスタンスの起動」をクリックします。

あるいは、次の `asadmin` コマンドを使用します。

```
asadmin stop-cluster --user admin cluster-name
asadmin start-cluster --user admin cluster-name
```

これらのコマンドの詳細については、[stop-cluster\(1\)](#)および[start-cluster\(1\)](#)を参照してください。

Web Server の再起動

Web Server を再起動するには、次の Web Server コマンドを入力します。

```
web_server_root/https-hostname/reconfig
```

web_server_root を Web Server のルートディレクトリに、*hostname* をホストマシンの名前に置き換えます。

▼ ロードバランサとして機能している Web Server インスタンスをクリーンアップする

- 1 次に示すように、ロードバランサ設定を削除します。

```
asadmin delete-http-lb-ref --user admin --config MyLbConfig FirstCluster  
asadmin delete-http-lb-config --user admin MyLbConfig
```
- 2 新しい Web Server インスタンスを作成した場合は、次の方法で削除できます。
 - a. Web Server の管理コンソールにログオンします。
 - b. インスタンスを停止します。
インスタンスを削除します。

HADB のアップグレード

HADB は、ソフトウェアのアップグレードによっても中断されることのない「常時有効な」サービスを提供するように設計されています。この節では、データベースをオフラインにしたり、可用性の低下を招いたりすることなく、新しいバージョンの HADB にアップグレードする方法について説明します。これは、オンラインアップグレードと呼ばれます。

以下の節では、HADB インストールをアップグレードする方法について説明します。

- 190 ページの「HADB をより新しいバージョンにアップグレードする」
- 191 ページの「HADB パッケージの登録」
- 192 ページの「HADB パッケージの登録の解除」
- 193 ページの「管理エージェントの起動スクリプトの置き換え」
- 193 ページの「HADB アップグレードの確認」

▼ HADB をより新しいバージョンにアップグレードする

- 1 新しいバージョンの HADB をインストールします。
- 2 191 ページの「HADB パッケージの登録」の説明に従って、新しい HADB バージョンを登録します。

HADB パッケージを HADB 管理ドメインに登録すると、HADB パッケージのアップグレードや変更が容易になります。管理エージェントは、ソフトウェアパッケージ

が配置されている場所や、ドメイン内のホストに関するバージョン情報を常時監視します。デフォルトのパッケージ名は、Vの文字で始まり、hadbmプログラムのバージョン番号が含まれた文字列です。

- 3 データベースが使用するパッケージを変更します。
以下のコマンドを入力します。
`hadbm set PackageName=package`
ここで、*package* は、新しいHADBパッケージのバージョン番号です。
- 4 192ページの「HADBパッケージの登録の解除」の説明に従って、既存のHADBインストールの登録を解除します。
- 5 必要に応じて、管理エージェントの起動スクリプトを置き換えます。
詳細については、193ページの「管理エージェントの起動スクリプトの置き換え」を参照してください。
- 6 193ページの「HADBアップグレードの確認」の説明に従って、結果を確認します。
- 7 (省略可能)古いHADBバージョンのバイナリファイルを削除します。
HADBが正しくアップグレードされたことを確認したら、古いHADBパッケージを削除できます。

HADBパッケージの登録

`hadbm registerpackage` コマンドを使用して、管理ドメイン内のホストにインストールされているHADBパッケージを登録します。HADBパッケージはまた、`hadbm create` を使用してデータベースを作成するときにも登録できます。

`hadm registerpackage` コマンドを使用する前に、ホストリスト内のすべてのホストですべての管理エージェントが設定および実行されていること、管理エージェントのリポジトリが更新用に使用できること、および同じパッケージ名ですでに登録されているソフトウェアパッケージがないことを確認してください。

コマンド構文は次のとおりです。

```
hadbm registerpackage --packagepath=path [--hosts=hostlist]  
[--adminpassword=password | --adminpasswordfile=file] [--agent=maurl]  
[[package-name]]
```

package-name オペランドがパッケージの名前です。

次の表は、特殊な `hadbm registerpackage` コマンドオプションを示しています。ほかのコマンドオプションについては、205ページの「セキュリティオプション」および207ページの「一般的なオプション」を参照してください。

表 10-1 hadbm registerpackage のオプション

オプション	説明
--hosts= <i>hostlist</i>	コマンドで区切られているか、または二重引用符で囲まれ空白で区切られている、ホストのリスト。
-H	
--packagepath= <i>path</i>	HADBソフトウェアパッケージへのパス。
-L	

たとえば、次のコマンドは、ソフトウェアパッケージ v4 をホスト host1、host2、および host3 に登録します。

```
hadbm registerpackage
--packagepath=hadb_install_dir/SUNWHadb/4.4
--hosts=host1,host2,host3 v4
```

応答は次のようになります。

```
Package successfully registered.
```

--hosts オプションを省略した場合は、ドメイン内で有効になっているすべてのホストにそのパッケージが登録されます。

HADB パッケージの登録の解除

hadbm unregisterpackage コマンドを使用して、管理ドメインに登録されている HADB パッケージを削除します。

hadbm unregisterpackage コマンドを使用する前に、次の点を確認してください。

- すべての管理エージェントが設定され、hostlist 内のすべてのホスト上で実行中である。
- 管理エージェントのリポジトリが更新用に使用できる。
- 新しい HADB パッケージが管理ドメインに登録されている。
- 登録を解除しようとしているパッケージで動作するように設定された既存のデータベースがない。

コマンド構文は次のとおりです。

```
hadbm unregisterpackage
--hosts=hostlist
[--adminpassword=password | --adminpasswordfile= file]
[--agent= maurl]
[package-name ]
```

package-name オペランドがパッケージの名前です。

--hosts オプションについては、191 ページの「HADB パッケージの登録」を参照してください。--hosts オプションを省略した場合は、パッケージが登録された、有効になっているホストがホストリストのデフォルトになります。ほかのコマンドオプションについては、205 ページの「セキュリティオプション」および207 ページの「一般的なオプション」を参照してください。

例 10-2 HADB の登録解除の例

ドメイン内の特定のホストからソフトウェアパッケージ v4 の登録を解除するには、次のコマンドを実行します。

```
hadbm unregisterpackage --hosts=host1,host2,host3 v4
```

応答は次のようになります。

```
Package successfully unregistered.
```

管理エージェントの起動スクリプトの置き換え

新しいバージョンの HADB をインストールすると、`/etc/init.d/ma-initd` にある管理エージェントの起動スクリプトの置き換えが必要になる場合があります。ファイル `HADB_install_dir/lib/ma-initd` の内容を確認してください。古い `ma-initd` ファイルと異なっている場合は、古いファイルを新しいファイルに置き換えます。

▼ HADB アップグレードの確認

次の手順に従って、HADB が正しくアップグレードされていることを確認します。

- 稼働している HADB プロセスのバージョンを確認します。

すべての HADB ノードに次のコマンドを入力して、HADB バージョンを表示します。

```
new-path/bin/ma -v
```

```
new-path/bin/hadbm -v
```

ここで、*new-path* は新しい HADB インストールへのパスです。

この結果、新しい HADB バージョン番号が表示されるはずです。

- データベースが稼働していることを確認します。

次のコマンドを入力します。

```
new-path/bin/hadbm status -n
```

アップグレードに成功すると、running 状態のすべての HADB ノードが表示されます。

- 3 **HADB** を使用している製品の設定が新しい **HADB** パスに変更されている必要があります。
- 4 **HADB** を使用している製品に対して、あらゆるアップグレードの評価を実行します。

高可用性データベースの管理

注 - HADB ソフトウェアは、Sun GlassFish Enterprise Server の Enterprise Server スタンドアロン配布で提供されます。Sun GlassFish Enterprise Server の利用可能な配布の詳細については、『[Sun GlassFish Enterprise Server 2.1 Installation Guide](#)』の「[Distribution Types and Their Components](#)」を参照してください。HADB 機能はエンタープライズプロファイルでのみ利用可能です。プロファイルの詳細については、『[Sun GlassFish Enterprise Server 2.1 管理ガイド](#)』の「[プロファイル](#)」を参照してください。

この章では、Sun GlassFish Enterprise Server 環境における高可用性データベース (HADB) について説明します。HADB を設定および管理する方法について解説します。HADB の作成および管理をする前に、まずシステムのトポロジを決定して、各種マシンに HADB ソフトウェアをインストールする必要があります。

この章では、次の内容について説明します。

- 196 ページの「HADB 管理エージェントの使用法」
- 204 ページの「hadbm 管理コマンドの使用法」
- 209 ページの「HADB の設定」
- 225 ページの「HADB の管理」
- 234 ページの「HADB の拡張」
- 240 ページの「HADB の監視」
- 248 ページの「HADB マシンの管理」

HADB 管理エージェントの使用法

管理エージェント `ma` は HADB ホスト上で管理コマンドを実行します。HADB ノードスーパーバイザプロセスが失敗すると、管理エージェントはそのプロセスを再起動して、その可用性を確保します。

- 196 ページの「管理エージェントの起動」
- 201 ページの「管理エージェントコマンドの構文」
- 202 ページの「管理エージェント設定のカスタマイズ」

管理エージェントの起動

管理エージェントは、次のように起動できます。

- サービスとして本稼働環境で使用する場合。196 ページの「サービスとしての管理エージェントの起動」を参照してください。管理エージェントの有効性を確保するには、システムの再起動時に管理エージェントが自動的に起動することを確認してください。198 ページの「管理エージェントの自動再起動の実現」を参照してください。
- コンソールモードで、評価、テスト、または開発における通常のプロセスとして使用する場合。200 ページの「コンソールモードでの管理エージェントの起動」を参照してください。
- Solaris 10 でサービス管理機能 (SMF) とともに使用する場合。201 ページの「Solaris 10 サービス管理機能による管理エージェントの実行」を参照してください。

いずれの場合も、使用しているのが Java Enterprise System であるかスタンドアロン Application Server であるかによって、手順が異なります。

サービスとしての管理エージェントの起動

サービスとして管理エージェントを起動すると、システムが停止するかまたは操作によりシステムを明示的に停止するまで、実行を継続します。コマンドは、インストール環境やプラットフォームによって異なります。

- 196 ページの「Solaris または Linux 上の Java Enterprise System」
- 197 ページの「Windows 上の Java Enterprise System」
- 197 ページの「Solaris または Linux 上のスタンドアロン Application Server」
- 197 ページの「Windows 上のスタンドアロン Application Server」

Solaris または Linux 上の Java Enterprise System

管理エージェントをサービスとして起動するには、次のコマンドを使用します。

```
/etc/init.d/ma-initd start
```

サービスを停止するには、次のコマンドを使用します。

```
/etc/init.d/ma-initd stop
```

Windows 上の Java Enterprise System

管理エージェントを Windows サービスとして起動するには、次のコマンドを使用します。 `HADB_install_dir\bin\ma -i [config-file]`

省略可能な引数 `config-file` は、管理エージェントの設定ファイルを指定します。設定ファイルは、デフォルトの管理エージェント設定を変更する場合にのみ使用してください。詳細については、[202 ページの「管理エージェント設定のカスタマイズ」](#)を参照してください。

管理エージェントを停止してサービスから削除(登録解除)するには、次のコマンドを使用します。 `HADB_install_dir\bin\ma -r [config-file]`

管理を実行するには、「管理ツール」->「サービス」を選択します。表示されるウィンドウで、サービスの起動と停止、自動起動の無効化などを行えます。

Solaris または Linux 上のスタンドアロン Application Server

管理エージェントをサービスとして起動するには、次のコマンドを使用します。

```
HADB_install_dir/bin/ma-initd start
```

サービスを停止するには、次のコマンドを使用します。

```
HADB_install_dir/bin/ma-initd stop
```

デフォルト値を変更するには、シェルスクリプト `HADB_install_dir/bin/ma-initd` を編集します。 `ma-initd` をディレクトリ `/etc/init.d` にコピーします。スクリプト内の `HADB_ROOT` と `HADB_MA_CFG` のデフォルト値を、実際のインストールを反映するように置き換えます。

- `HADB_ROOT` は HADB インストールディレクトリ `HADB_install_dir` です。
- `HADB_MA_CFG` は管理エージェント設定ファイルのある場所です。詳細については、[202 ページの「管理エージェント設定のカスタマイズ」](#)を参照してください。

Windows 上のスタンドアロン Application Server

管理エージェントを Windows サービスとして起動するには、次のコマンドを使用します。 `HADB_install_dir\bin\ma -i [config-file]`

省略可能な引数 `config-file` は、管理エージェントの設定ファイルを指定します。設定ファイルは、デフォルトの管理エージェント設定を変更する場合にのみ使用してください。

管理エージェントを停止してサービスから削除 (登録解除) するには、次のコマンドを使用します。 `HADB_install_dir\bin\ma -r [config-file]`

管理を実行するには、「管理ツール」->「サービス」を選択します。表示されるウィンドウで、サービスの起動と停止、自動起動の無効化などを行えます。

管理エージェントの自動再起動の実現

本稼働環境で、管理エージェントが自動的に再起動するように設定します。そのようにすることで、`ma` プロセスが失敗する場合またはオペレーティングシステムが再起動する場合における管理エージェントの有効性が確実にになります。

Windows プラットフォームでは、管理エージェントをサービスとして起動したあとに、Windows 管理ツールを使用して、サービスの「スタートアップの種類」を「自動」に設定し、必要に応じて「回復」オプションを指定します。

Solaris および Linux プラットフォームでは、この節の手順を用いて、管理エージェントの自動再起動を設定します。以下の手順を行うと、システムが次のレベルになったときにのみ、管理エージェントが起動します。

- Solaris での実行レベル 3 (デフォルト)。
- RedHat Linux での実行レベル 5 (グラフィックモードでのデフォルト)。

それ以外の実行レベルになると、管理エージェントは停止します。

▼ Solaris または Linux 上の Java Enterprise System で自動再起動を設定する

始める前に この節は、オペレーティングシステムの初期化と実行レベルについての基本を理解していることを前提としています。これらのトピックについては、使用しているオペレーティングシステムのドキュメントを参照してください。

- 1 システムのデフォルト実行レベルが **3** または **5** であることを確認します。
システムのデフォルト実行レベルを確認するには、`/etc/inittab` ファイルを調べ、ファイル上部にある次のような行を探します。
`id:5:initdefault:`
この例は、デフォルト実行レベル 5 を示しています。
- 2 [199 ページの「ソフトリンクの作成」](#) の説明に従って、ファイル `/etc/init.d/ma-initd` へのソフトリンクを作成します。
- 3 マシンをリブートします。

次の手順 エージェントの自動起動および停止を解除するには、これらのリンクを削除するか、リンク名中の文字 **K** と **S** を小文字に変更します。

▼ Solaris または Linux 上のスタンドアロン Application Server で自動再起動を設定する

- 1 シェルで、カレントディレクトリを *HADB_install_dir* /bin に変更します。
- 2 シェルスクリプト *ma-initd* を編集します。
スクリプト内の *HADB_ROOT* および *HADB_MA_CFG* のデフォルト値を確認して、インストールを反映させます。
 - *HADB_ROOT* は HADB インストールディレクトリ *HADB_install_dir* です。
 - *HADB_MA_CFG* は管理エージェント設定ファイルのある場所です。詳細については、[202 ページの「管理エージェント設定のカスタマイズ」](#)を参照してください。
- 3 *ma-initd* をディレクトリ */etc/init.d* にコピーします。
- 4 [199 ページの「ソフトリンクの作成」](#)の説明に従って、ファイル */etc/init.d/ma-initd* へのソフトリンクを作成します。

次の手順 エージェントの自動起動および停止を解除するには、これらのリンクを削除するか、リンク名中の文字 K と S を小文字に変更します。

ソフトリンクの作成

Solaris の場合、次のソフトリンクを作成します。

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc2.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/K20ma-initd (only for Sun 4m and 4u architecture)
/etc/rc6.d/K20ma-initd
/etc/rcS.d/K20ma-initd
```

Linux の場合、次のソフトリンクを作成します。

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/S99ma-initd
/etc/rc6.d/K20ma-initd
```

コンソールモードでの管理エージェントの起動

評価やテストのために、コンソールモードで管理エージェントを起動することができます。本稼働環境ではこの方法で管理エージェントを起動しないでください。システムやプロセスの障害のあとで `ma` プロセスが再起動しなかったり、コマンドウィンドウを閉じたときにプロセスが終了したりするからです。コマンドは、プラットフォームやインストール環境によって異なります。

- 200 ページの「Solaris または Linux 上の Java Enterprise System」
- 200 ページの「Windows 上の Java Enterprise System」
- 200 ページの「Windows 上のスタンドアロン Application Server」
- 201 ページの「Solaris または Linux 上のスタンドアロン Application Server」

Solaris または Linux 上の Java Enterprise System

コンソールモードで HADB 管理エージェントを起動するには、次のコマンドを使用します。

```
opt/SUNWhadb/bin/ma [config-file]
```

デフォルトの管理エージェント設定ファイルは `/etc/opt/SUNWhadb/mgt.cfg` です。

管理エージェントを停止するには、プロセスを終了するか、またはシェルウィンドウを閉じます。

Windows 上の Java Enterprise System

コンソールモードで管理エージェントを起動するには、次のコマンドを使用します。

```
HADB_install_dir\bin\ma [config-file]
```

省略可能な引数 *config-file* は、管理エージェント設定ファイルの名前です。設定ファイルの詳細については、202 ページの「管理エージェント設定のカスタマイズ」を参照してください。

エージェントを停止するには、プロセスを終了します。

Windows 上のスタンドアロン Application Server

コンソールモードで管理エージェントを起動するには、次のコマンドを使用します。

```
HADB_install_dir\bin\ma [config-file]
```

省略可能な引数 *config-file* は、管理エージェント設定ファイルの名前です。詳細については、202 ページの「管理エージェント設定のカスタマイズ」を参照してください。

管理エージェントを停止するには、プロセスを終了します。

Solaris または Linux 上のスタンドアロン Application Server

コンソールモードで HADB 管理エージェントを起動するには、次のコマンドを使用します。

```
HADB_install_dir/bin/ma [config-file]
```

デフォルトの管理エージェント設定ファイルは `HADB_install_dir/bin/ma.cfg` です。

管理エージェントを停止するには、プロセスを終了するか、またはシェルウィンドウを閉じます。

Solaris 10 サービス管理機能による管理エージェントの実行

サービス管理機能 (SMF) では、Solaris 10 でサービスを再起動、表示、および管理するメカニズムが提供されます。SMF を使用して、HADB 管理エージェントを起動、再起動、および管理できます。

管理エージェントの障害管理リソース識別子 (FMRI) は `svc:/application/hadb-ma` です。

管理エージェントコマンドの構文

管理エージェント `ma` コマンドの構文は、次のとおりです。

```
ma [common-options]  
[ service-options]  
config-file
```

引数の意味はそれぞれ以下のとおりです。

- `common-options` は、201 ページの「管理エージェントコマンドの構文」で説明されている 1 つ以上の共通オプションです。
- `service-options` は、201 ページの「管理エージェントコマンドの構文」で説明されている Windows サービスオプションのいずれかです。
- `config-file` は、管理エージェント設定ファイルへのフルパスです。詳細については、202 ページの「管理エージェント設定のカスタマイズ」を参照してください。

表 11-1 管理エージェント共通オプション

オプション	説明	デフォルト
<code>--define name=value-D</code>	プロパティ <i>name</i> に <i>value</i> を割り当てます。このプロパティは 203 ページの「設定ファイル」に定義されているプロパティのいずれかです。このオプションは、複数回繰り返すことができます。	なし
<code>--help-?</code>	ヘルプ情報を表示します。	False
<code>--javahome path-j</code>	<i>path</i> にある Java Runtime 環境 (1.4 以降) を使用します。	なし
<code>--systemroot path-y</code>	通常は %SystemRoot% で設定されているオペレーティングシステムルートへのパス。	なし
<code>--version-V</code>	バージョン情報を表示します。	False

表 11-2 では、管理サービスを Windows サービスとして起動するためのオプションを説明しています。-i、-r、および -s オプションは相互に排他的であるため、一度に 1 つだけを使用してください。

Windows では、設定ファイルまたはコマンド行にプロパティ値のパスを指定する際に、スペースを含むファイルパスを二重引用符 (") で囲んでエスケープします。コロン(:) ドライブセパレータと円記号 (\) ディレクトリセパレータは、二重引用符と円記号を用いて \"; および "\\ のようにエスケープします。

表 11-2 管理エージェントサービスオプション (Windows のみ)

オプション	説明	デフォルト
<code>--install-i</code>	エージェントを Windows サービスとしてインストールして、サービスを開始します。-i、-r、および -s オプションから、1 つだけを使用します。	False
<code>--name servicename-n</code>	ホスト上で複数のエージェントを実行している場合に、サービスに対して指定した名前を使用します。	HADBMgrAgent
<code>--remove-r</code>	サービスを停止し、Windows のサービスマネージャーからエージェントを削除します。-i、-r、および -s オプションから、1 つだけを使用します。	False
<code>--service-s</code>	エージェントを Windows サービスとして実行します。-i、-r、および -s オプションから、1 つだけを使用します。	False

管理エージェント設定のカスタマイズ

HADB には設定ファイルが組み込まれており、管理エージェント設定のカスタマイズに使用できます。設定ファイルを指定せずに管理エージェントを起動した場合は、デフォルト値が使用されます。設定ファイルを指定した場合、管理エージェントはそのファイルの設定を使用します。同じ設定ファイルをドメイン内のすべてのホストで繰り返し使用することができます。

▼ HADB ホストの管理エージェント設定をカスタマイズする

- 1 管理エージェント設定ファイルを編集して、希望する値を設定します。
- 2 カスタマイズした設定ファイルを引数に指定して、管理エージェントを起動します。

設定ファイル

Java Enterprise System では、設定ファイル内のすべてのエントリがコメントにされています。デフォルトの設定を使用する場合、変更の必要はありません。管理エージェント設定をカスタマイズするには、ファイルからコメントを削除し、必要に応じて値を変更してから、設定ファイルを引数に指定して、管理エージェントを起動します。

管理エージェント設定ファイルは次の場所にインストールされます。

- Solaris および Linux: `/etc/opt/SUNWhadb/mgt.cfg`
- Windows: `install_dir\lib\mgt.cfg`。

スタンドアロンインストールプログラムでは、管理エージェント設定ファイルは次の場所にインストールされます。

- Solaris および Linux: `HADB_install_dir/bin/ma.cfg`
- Windows: `HADB_install_dir/bin\ma.cfg`。

次の表で、設定ファイルの設定値について説明します。

表 11-3 設定ファイルの設定値

設定値名	説明	デフォルト
console.loglevel	コンソールログレベル。有効な値は、SEVERE、ERROR、WARNING、INFO、FINE、FINER、FINEST。	WARNING
logfile.loglevel	ログファイルのログレベル。有効な値は、SEVERE、ERROR、WARNING、INFO、FINE、FINER、FINEST。	INFO
logfile.name	ログファイルの名前と場所。読み込み/書き込みアクセスに対して有効なパスである必要があります。	Solaris および Linux: <code>/var/opt/SUNWhadb/ma/ma.log</code> Windows: <code>HADB_install_dir\ma.log</code>
ma.server.type	クライアントプロトコル。JMXMP のみサポートされています。	jmxp
ma.server.jmxmp.port	内部 (UDP) および外部 (TCP) 通信用のポート番号。正の整数である必要があります。推奨される範囲は 1024 ~ 49151 です。	1862

表 11-3 設定ファイルの設定値 (続き)

設定値名	説明	デフォルト
ma.server.mainternal.interfaces	複数のインタフェースを持つマシンの内部通信用のインタフェース。有効な IPv4 アドレスマスクである必要があります。ドメイン内のすべての管理エージェントが必ず同じサブネットを使用する必要があります。 たとえば、ホストに 10.10.116.61 と 10.10.124.61 の 2 つのインタフェースがある場合、最初のインタフェースを使用するには 10.10.116.0/24 を指定します。スラッシュの後ろの数字は、サブネットマスクのビット数を示します。	なし
ma.server.dbdevicepath	HADB デバイス情報を格納するパス。	Solaris および Linux: /var/opt/SUNWhadb/4 Windows: HADB_install_dir\device
ma.server.dbhistorypath	HADB 履歴ファイルを格納するパス。	Solaris および Linux: /var/opt/SUNWhadb Windows: REPLACEDIR (実行時に実際の URL に置換される)
ma.server.dbconfigpath	ノード設定データを格納するパス。	Solaris および Linux: /var/opt/SUNWhadb/dbdef Windows: C:\Sun\SUNWhadb\dbdef
repository.dr.path	ドメインリポジトリファイルのパス。	Solaris および Linux: /var/opt/SUNWhadb/repository Windows: C:\Sun\SUNWhadb\repository

hadbm 管理コマンドの使用法

hadbm コマンド行ユーティリティを使用して、HADB ドメイン、そのデータベースインスタンス、およびノードを管理します。hadbm ユーティリティ (管理クライアントとも呼ばれる) は、管理サーバーとして動作している、指定された管理エージェントに管理要求を送信します。管理エージェントにはリポジトリからデータベース設定へのアクセスがあります。

この節では、次のトピックで、hadbm コマンド行ユーティリティについて説明します。

- [205 ページの「コマンド構文」](#)
- [205 ページの「セキュリティオプション」](#)
- [207 ページの「一般的なオプション」](#)
- [208 ページの「環境変数」](#)

コマンド構文

hadbm ユーティリティは、`HADB_install_dir/bin` ディレクトリにあります。hadbm コマンドの汎用構文は次のとおりです。

```
hadbm subcommand
[-short-option [option-value]]
[--long-option [option-value]]
[operands]
```

サブコマンドで実行する操作またはタスクを識別します。サブコマンドは大文字と小文字を区別します。ほとんどのコマンドはオペランドを1つ(通常は dbname)とります。

オプションを指定することにより、hadbm がサブコマンドを実行する方法を変更できます。オプションは大文字と小文字を区別します。各オプションには長い書式と短い書式があります。省略形の場合はダッシュ1つ(-)を前に付け、長い書式の場合はダッシュ2つ(--)を前に付けます。boolean 型のオプションを除くほとんどのオプションは引数値を必要とし、この引数値により機能がオンに切り替わります。オプションを指定しないとコマンドが正常に実行されないということではありません。

サブコマンドにデータベース名が必要な場合にデータベース名を指定しないと、hadbm はデフォルトデータベース hadb を使用します。

例 11-1 hadbm コマンドの例

次に示すのは、`status` サブコマンドの例です。

```
hadbm status --nodes
```

セキュリティオプション

セキュリティ上の理由で、すべての hadbm コマンドには管理者パスワードが必要です。データベースまたはドメインを作成する際に、`--adminpassword` オプションを使用してパスワードを設定します。それ以降、そのデータベースまたはドメイン上で操作を実行するときには、そのパスワードを指定する必要があります。

さらにセキュリティを強化するには、パスワードをコマンド行に入力する代わりに、`--adminpasswordfile` オプションを使用してパスワードを含むファイルを指定します。次の行を用いてパスワードファイルにパスワードを定義します。

```
HADBM_ADMINPASSWORD=password
```

`password` をパスワードに置き換えてください。ファイル内のそれ以外の内容は無視されます。

--adminpassword と --adminpasswordfile の両方のオプションを指定した場合は、--adminpassword が優先されます。パスワードが必要なのに、コマンド中に指定されていない場合には、hadbm からパスワードを要求されます。

注- 管理者パスワードはデータベースまたはドメインを作成するときのみ設定することができ、あとで変更することはできません。

管理者パスワードに加えて、データベーススキーマを変更する操作を実行するために、HADB ではデータベースパスワードも要求されます。次のコマンドを使用するときには、これらのパスワードが両方必要となります。hadbm create、hadbm addnodes、および hadbm refragment。

--dbpassword オプションを使用して、コマンド行にデータベースパスワードを指定します。管理者パスワードと同じように、ファイルにパスワードを書き込み、--dbpasswordfile オプションでファイルの場所を指定することも可能です。次の行を用いてパスワードファイルにパスワードを定義します。

```
HADBM_DBPASSWORD=password
```

テストまたは評価の場合は、データベースまたはドメインを作成する際に --no-adminauthentication オプションを指定して、パスワード認証をオフにすることもできます。詳細については、[210 ページの「データベースの作成」](#) および [209 ページの「管理ドメインの作成」](#) を参照してください。

次の表に、hadbm セキュリティーコマンド行オプションの要約を示します。

表 11-4 hadbm セキュリティーオプション

オプション (省略形)	説明
--adminpassword=password -w	データベースまたはドメインの管理者パスワードを指定します。データベースまたはドメイン作成時にこのオプションを使用すると、hadbm を使用してデータベースまたはドメインで操作をするたびに、パスワードの提供が必要になります。 このオプションまたは --adminpasswordfile を使用し、両方は使用しないでください。
--adminpasswordfile=filepath -W	データベースまたはドメインの管理者パスワードを含むファイル指定します。データベースまたはドメイン作成時にこのオプションを使用すると、hadbm を使用してデータベースまたはドメインで操作をするたびに、パスワードの提供が必要になります。 このオプションまたは --adminpassword を使用し、両方は使用しないでください。

表 11-4 hadbm セキュリティーオプション (続き)

オプション(省略形)	説明
--no-adminauthentication -U	データベースまたはドメイン作成時に管理者パスワードは不要であることを指定する場合に、このオプションを使用します。セキュリティ上の理由で、本稼働配備環境ではこのオプションを使用しないでください。
--dbpassword= <i>password</i> -P	データベースパスワードを指定します。データベース作成時にこのオプションを使用すると、hadbm コマンドを使用してデータベースで操作をするたびに、パスワードの提供が必要になります。HADB システムユーザー用のパスワードが作成されます。パスワードには 8 文字以上が必要です。このオプションまたは --dbpasswordfile を使用し、両方は使用しないでください。
--dbpasswordfile= <i>filepath</i> -P	HADB システムユーザー用のパスワードを含むファイルを指定します。このオプションまたは --dbpassword を使用し、両方は使用しないでください。

一般的なオプション

汎用コマンドオプションは、どの hadbm サブコマンドにも使用できます。すべてが boolean 型オプションで、デフォルトは false です。次の表で、hadbm 汎用コマンドオプションについて説明します。

表 11-5 hadbm 汎用オプション

オプション(省略形)	説明
--quiet -q	説明メッセージを何も表示せずにサブコマンドを実行します。
--help -?	このコマンドの簡単な説明とサポートされているすべてのサブコマンドを表示します。サブコマンドは不要です。
--version -V	hadbm コマンドのバージョン詳細を表示します。サブコマンドは不要です。
--yes -y	非対話型モードでサブコマンドを実行します。
--force -f	非対話式にコマンドを実行し、コマンドの後置条件をすでに満たしている場合には、エラーをスローしません。
--echo -e	サブコマンドを、すべてのオプションとそれらについてユーザーが定義した値またはデフォルト値とともに表示してから、サブコマンドを実行します。

表 11-5 hadbm 汎用オプション (続き)

オプション(省略形)	説明
--agent= <i>URL</i>	管理エージェントの URL。 <i>URL</i> の書式は <i>hostlist:port</i> です。ここで、 <i>hostlist</i> はホスト名または IP アドレスのコンマ区切りリストで、 <i>port</i> は管理エージェントが動作しているポート番号です。
-m	デフォルトは localhost: 1862 です。
	注: このオプションは hadbm addnodes には無効です。

環境変数

便宜上、コマンドオプションを指定する代わりに、環境変数を設定することもできます。次の表で、hadbm コマンドオプションに対応する環境変数について説明します。

表 11-6 HADB オプションと環境変数

長い書式	短縮書式	デフォルト	環境変数
--adminpassword	-w	なし	\$HADBM_ADMINPASSWORD
--agent	--m	localhost: 1862	\$HADBM_AGENT
--datadevices	-a	1	\$HADBM_DATADEVICES
dbname	なし	hadb	\$HADBM_DB
--dbpassword	-p	なし	\$HADBM_DBPASSWORD
--dbpasswordfile	-P	なし	\$HADBM_DBPASSWORDFILE
--devicepath	-d	Solaris および Linux: /var/opt/SUNWhadb Windows: C:\Sun\AppServer\SUNWhadb\vers。ここで、 <i>vers</i> は HADB バージョン番号です。	\$HADBM_DEVICEPATH
--devicesize	-z	なし	\$HADBM_DEVICESIZE
--echo	-e	False	\$HADBM_ECHO
--fast	-F	False	\$HADBM_FAST
--force	-f	False	\$HADBM_FORCE
--help	-?	False	\$HADBM_HELP

表 11-6 HADB オプションと環境変数 (続き)

長い書式	短縮書式	デフォルト	環境変数
--historypath	-t	Solaris および Linux: /var/opt/SUNWhadb Windows: REPLACEDIR (実行時に実際の URL に置換される)	\$HADBM_HISTORYPATH
--hosts	-H	なし	\$HADBM_HOSTS
--interactive	-i	True	\$HADBM_INTERACTIVE
--no-refragment	-r	False	\$HADBM_NOREFRAGMENT
--portbase	-b	15200	\$HADBM_PORTBASE
--quiet	-q	False	\$HADBM_QUIET
--repair	-R	True	\$HADBM_REPAIR
--rolling	-g	True	\$HADBM_ROLLING
--saveto	-o	なし	\$HADBM_SAVETO
--set	-S	なし	\$HADBM_SET
--spares	-s	0	\$HADBM_SPARES
--startlevel	-l	normal	\$HADBM_STARTLEVEL
--version	-V	False	\$HADBM_VERSION
--yes	-y	False	\$HADBM_YES

HADB の設定

この節では、次の基本的な HADB 設定作業について説明します。

- [209 ページの「管理ドメインの作成」](#)
- [210 ページの「データベースの作成」](#)
- [216 ページの「設定属性の表示と変更」](#)
- [222 ページの「JDBC 接続プールの設定」](#)

管理ドメインの作成

コマンド `hadbm createdomain` を実行すると、指定した HADB ホストを含む管理ドメインが作成されます。このコマンドは、ホストと持続性設定ストアとの間の内部通信チャンネルを初期化します。

コマンドの構文は次のとおりです。

```
hadbm createdomain
  [--adminpassword=password | --adminpasswordfile=file |
  --no-adminauthentication] [--agent=maurl]
  hostlist
```

hostlist オペランドは、それぞれが有効な IPv4 ネットワークアドレスである HADB ホストのコンマ区切りリストです。新規ドメインに組み込むすべてのホストを *hostlist* に含めてください。

コマンドオプションの説明は、[207 ページの「一般的なオプション」](#)を参照してください。

このコマンドを使用する前に、HADB 管理エージェントが *hostlist* に含まれているすべてのホスト上で実行中であることを確認してください。さらに、管理エージェントは次の条件を満たしている必要があります。

- 既存のドメインのメンバーではない。
- 同一のポートを使用するように設定されている。
- UDP や TCP を介して、および IP マルチキャストを使用して相互に通信できる。

hadbm が管理ドメインを作成すると、ドメイン内のすべてのホストが使用可能になります。これで、管理エージェントがデータベースを管理する用意は整いました。HADB ドメインを作成したら、次のステップは、HADB データベースの作成です。HADB データベースの作成に関する詳細については、[210 ページの「データベースの作成」](#)を参照してください。

例 11-2 HADB 管理ドメインの作成

次の例では、指定した 4 つのホスト上に管理ドメインが作成されます。

```
hadbm createdomain --adminpassword=password host1,host2,host3,host4
```

hadbm がコマンドを正常に実行すると、次のメッセージが表示されます。

```
Domain host1,host2,host3, host4 created.
```

HADB ドメインを作成したあと、HADB パッケージのパスとバージョンを管理エージェントに登録します。

データベースの作成

hadbm create コマンドを使用して、データベースを手動で作成します。

このコマンドを使用してデータベースを作成する前に、管理ドメインを作成し、HADB パッケージに登録します。hadbm create を実行する時点でこの 2 つのステップをまだ行っていない場合は、コマンドによってそれらのステップが暗黙に実行されます。このようにすれば行う作業は減るように思えますが、いずれかのコ

マンドでエラーが生じたときに、デバッグが困難になる場合があります。さらに、`hadbm create` は不可分ではありません。つまり、暗黙的なコマンドのいずれかが失敗した場合に、正常に実行されたコマンドはロールバックされません。したがって、ドメインを作成し HADB パッケージを登録したあとのみ、データベースを作成するのが最善です。

たとえば、`hadbm createdomain` と `hadbm registerpackage` は正常に実行されるものの `hadbm create database` は失敗する場合、`hadbm createdomain` と `hadbm registerpackage` によって加えられた変更は持続します。

▼ データベースを作成する

- 1 管理ドメインを作成します。
詳細については、209 ページの「管理ドメインの作成」を参照してください。
- 2 HADB パッケージを登録します。
詳細については、191 ページの「HADB パッケージの登録」を参照してください。
- 3 `hadbm create` コマンドを使用してデータベースを作成します。
コマンド構文については、次の節を参照してください。

hadbm create コマンド構文

```
hadbm create [--package= name] [--packagepath= path] [--historypath= path]
[--devicepath= path] [--datadevices= number] [--portbase= number]
[--spares=number] [--set=attr-val-list] [--agent=maurl] [--no-cleanup]
[ --no-clear ] [ --devicesize =size] [--dbpassword=password | --dbpasswordfile=file]
] --hosts=host list [--adminpassword=password | --adminpasswordfile=file |
--no-adminauthentication ] [dbname]
```

dbname オペランドにはデータベース名を指定します。この名前は一意でなければなりません。データベース名が一意であることを確認するために、`hadbm list` コマンドを使用して既存のデータベース名を一覧表示します。複数のデータベースを作成する必要がなければ、デフォルトのデータベース名を使用してください。たとえば、同じセットの HADB マシン上に独立データベースで複数のクラスタを作成するには、クラスタごとに別個のデータベース名を使用します。

`hadbm create` コマンドは、エラーメッセージをログファイルではなくコンソールに書き込みます。

表 11-7 には、`hadbm create` コマンドの特殊なオプションが説明されています。追加のコマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。

表 11-7 hadbm create オプション

オプション (省略形)	説明	デフォルト
--datadevices= <i>number</i> -a	各ノード上のデータデバイスの数。1~8 を含みます。データデバイスには 0 から始まる番号が付けられます。	1
--devicepath= <i>path</i> -d	デバイスへのパス。デバイスには次の 4 つがあります。 <ul style="list-style-type: none"> ■ DataDevice ■ NiLogDevice (ノード内部ログデバイス) ■ RelalgDevice (関係代数クエリーデバイス) ■ NoManDevice (ノードマネージャーデバイス)。 このパスは存在していて、書き込み可能であることが必要です。このパスをノードまたはデバイスごとに異なる設定にする場合は、 215 ページの「異機種システム混在デバイスパスの設定」 を参照してください。	Solaris および Linux: /var/opt/SUNWhadb Windows: C:\Sun\AppServer\SUNWhadb\vers。ここで、vers は HADB バージョン番号です。 デフォルトは、管理エージェント設定ファイル内の <code>ma.server.dbdevicepath</code> によって指定されます。詳細については、 203 ページの「設定ファイル」 を参照してください。
--devicesize= <i>size</i> -z	各ノードのデバイスサイズ。詳細については、 214 ページの「デバイスサイズの指定」 を参照してください。 235 ページの「既存ノードへの記憶スペースの追加」 の説明に従って、デバイスサイズを増やします。	1024M バイト 最大サイズは、オペレーティングシステムのファイルサイズまたは 256G バイトの小さい方となります。最小サイズは次のとおりです。 $(4 \times \text{LogbufferSize} + 16\text{M バイト}) / n$ ここで、 <i>n</i> は --datadevices オプションで指定されたデータデバイスの番号です。
--historypath= <i>path</i> -t	履歴ファイルへのパス。このパスはすでに存在していて、書き込み可能であることが必要です。 履歴ファイルの詳細については、 250 ページの「履歴ファイルの消去と保存」 を参照してください。	デフォルトは、管理エージェント設定ファイル内の <code>ma.server.dbhistorypath</code> によって指定されます。詳細については、 203 ページの「設定ファイル」 を参照してください。 Solaris および Linux: /var/opt/SUNWhadb Windows: REPLACEDIR (実行時に実際の URL に置換される)

表 11-7 hadbm create オプション (続き)

オプション(省略形)	説明	デフォルト
--hosts= <i>hostlist</i> -H	データベース内のノードのホスト名または IP アドレス (IPv4 のみ) のコンマ区切りリスト。DNS 検索への依存を避けるため、IP アドレスを使用してください。ホスト名は必ず絶対名にします。localhost や 127.0.0.1 をホスト名として使用することはできません。Host names 詳細については、214 ページの「ホストの指定」を参照してください。	なし
--package= <i>name</i> -k	HADB パッケージの名前 (バージョン)。パッケージが見つからない場合は、デフォルトパッケージが登録されます。 このオプションは推奨されていません。hadbm registerpackage コマンドを使用して、パッケージをドメインに登録してください。	なし
--packagepath= <i>path</i> -L	HADB ソフトウェアパッケージへのパス。パッケージがドメインに登録されていない場合のみ使用します。 このオプションは推奨されていません。hadbm registerpackage コマンドを使用して、パッケージをドメインに登録してください。	なし
--portbase= <i>number</i> -b	ノード 0 で使用されるポートベース番号。以降のノードは、この番号から 20 の間隔でポートベース番号が自動的に割り当てられます。各ノードはそれ自身のポートベース番号とそれに続く 5 つの連続する番号のポートを使用します。 同じマシン上で複数のデータベースを実行するには、明示的にポート番号を割り当てるように計画します。	15200
--spares= <i>number</i> -s	スペアノードの数。この数は、偶数かつ --hosts オプションに指定したノード数より少ない数でなければいけません。	0
--set= <i>attr-val-list</i> -S	<i>name =value</i> 書式のデータベース設定属性のコンマ区切りリスト。データベース設定属性の説明は、250 ページの「履歴ファイルの消去と保存」を参照してください。	なし

例11-3 データベースの作成例

次に示すのは、データベースを作成するコマンドの例です。

```
hadbm create --spares 2 --devicesize 1024 --hosts n0,n1,n2,n3,n4,n5
```

ホストの指定

--hosts オプションを使用して、データベース内のノードのホスト名または IP アドレスのコンマ区切りリストを指定します。hadbm create コマンドは、リスト内のホスト名(または IP アドレス)ごとに1つのノードを作成します。ノードの数は偶数でなければなりません。重複するホスト名を使用すると、同じマシン上に異なるポート番号が指定された複数のノードが作成されます。同じマシン上のノードがミラーノードではなく、異なる DRU からでもないことを必ず確認してください。

ノードには、このオプションでリストされている順番で、ゼロから始まる番号が付けられます。最初のミラー化されたペアはノード0と1、2番目のミラーペアはノード2と3となり、以下同様です。奇数番号のノードが一方のDRUに配置され、偶数番号のノードは他方のDRUに配置されます。--spares オプションを指定すると、もっとも大きい番号のノードがスペアノードとなります。

二重ネットワークインタフェースの設定については、[178 ページの「ネットワーク冗長性の設定」](#)を参照してください。

デバイスサイズの指定

--devicesize オプションを使用して、デバイスサイズを指定します。推奨されているデバイスサイズは次のとおりです。

$$(4x / nd + 4l/d) / 0.99$$

説明:

- x は、ユーザーデータの合計サイズです。
- n は、--hosts オプションで指定されたノード数です。
- d は、--datadevices オプションで指定された、ノードあたりのデバイス数です。
- l は、属性 LogBufferSize で指定されたログバッファサイズです。

hadbm addnodes を使用するなどして再度の断片化が行われる可能性がある場合は、推奨されるデバイスサイズは次のようになります。

$$(8x / nd + 4l/d) / 0.99$$

異機種システム混在デバイスパスの設定

ノードまたはサービスごとに異なるデバイスパスを設定するには、`hadbm create` の `--set` オプションを使用します。デバイスには、`DataDevice`、`NiLogDevice` (ノード内部ログデバイス)、`RelalgDevice` (関係代数クエリーデバイス)、および `NoManDevice` (ノードマネージャーデバイス) の 4 種類があります。各 `name=value` ペアの構文は次のとおりです。ただし、`-devno` は、`device` が `DataDevice` の場合にのみ必要です。

```
node-nodeno.device-devno.Devicepath
```

次に例を示します。

```
--set Node-0.DataDevice-0.DevicePath=/disk0,
Node-1.DataDevice-0.DevicePath=/disk 1
```

次のようにして、履歴ファイルへの異機種システム混在パスを設定することも可能です。

```
node-nodeno.historypath=path
```

履歴ファイルについては、250 ページの「履歴ファイルの消去と保存」を参照してください。

特定のノードまたはデバイス用に設定されていないデバイスパスは、すべて `--devicepath` の値にデフォルト設定されます。

注 - デバイスパスおよび履歴ファイルの場所の変更は、`hadbm set` および `hadbm addnodes` コマンドを使用して行います。

障害追跡

データベースの作成がうまくいかない場合は、次の点をチェックしてください。

- すべてのホスト上で管理エージェントを起動し、HADB ドメインを定義したことを確認します。詳細については、196 ページの「管理エージェントの起動」を参照してください。
- ファイルおよびディレクトリのアクセス権は、次のユーザーに対して、インストールパス、履歴パス、デバイスパス、設定パスへの読み取り、書き込み、および実行のアクセスを許可するように設定されている必要があります。
 - Sun Java System Application Server 管理ユーザー (インストール時に設定)
 - HADB システムユーザー
 - ユーザーアクセス権の設定に関する詳細については、177 ページの「HADB の設定の準備」を参照してください。

Enterprise Server および HADB ポート割り当てが、同じマシン上のほかのポート割り当てと競合しないようにする必要があります。推奨されているデフォルトのポート割り当ては次のとおりです。

- Sun GlassFishMessage Queue: 7676
- IIOP: 3700
- HTTP サーバー: 80
- 管理サーバー: 4848
- HADB ノード: 各ノードは連続する 6 つのポートを使用します。たとえばデフォルトポート 15200 の場合、ノード 0 は 15200 ~ 15205、ノード 1 は 15220 ~ 15225 を使用し、以下同様です。

ディスク容量が適切であることも重要です。『[Sun GlassFish Enterprise Server 2.1 リリースノート](#)』を参照してください。

設定属性の表示と変更

データベース設定属性の表示および変更は、それぞれ `hadbm get` および `hadbm set` コマンドを使用して行えます。

設定属性の値の取得

設定属性の値を取得するには、`hadbm get` コマンドを使用します。有効な属性のリストについては、[218 ページの「設定属性」](#)を参照してください。コマンド構文は次のとおりです。

```
hadbm get attribute-list | --all  
[dbname]  
[--adminpassword=password | --adminpasswordfile=file]  
[--agent=maurl]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`attribute-list` オペランドは、コマンドで区切られたまたは引用符で囲まれスペースで区切られた、属性のリストです。--all オプションはすべての属性の値を表示します。`hadbm get` のすべての属性のリストについては、[218 ページの「設定属性」](#)を参照してください。

コマンドオプションの説明は、[207 ページの「一般的なオプション」](#)を参照してください。

例 11-4 `hadbm get` の使用例

```
hadbm get jdbcUrl,NumberOfSessions
```

設定属性の値の設定

設定属性の値を設定するには、`hadbm set` コマンドを使用します。有効な属性のリストについては、[218 ページの「設定属性」](#)を参照してください。

```
hadbm set [dbname] attribute
=value[,attribute=
value...]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`attribute=value` リストは、コンマで区切られたまたは引用符で囲まれスペースで区切られた、属性のリストです。

コマンドオプションの説明は、[207 ページの「一般的なオプション」](#)を参照してください。

このコマンドが正常に実行されると、データベースは以前の状態またはよりよい状態で再起動されます。データベースの状態については、[241 ページの「HADB の状態の取得」](#)を参照してください。[230 ページの「データベースの再起動」](#)で説明されている手順に従って、HADB を再起動します。

次の属性は、`hadbm set` では設定できません。その代わりに、データベース作成時に設定します ([210 ページの「データベースの作成」](#)を参照)。

- DatabaseName
- DevicePath
- HistoryPath
- NumberOfDatadevices
- Portbase
- JdbcUrl (この値は、データベース作成時に `--hosts` および `--portbase` オプションに基づいて設定される)。

注 - `hadbm set` を使用して `ConnectionTrace` と `SQLTraceMode` 以外のいずれかの設定属性を設定すると、HADB の順次再起動が実行されます。順次再起動では、各ノードが一度に1つずつ、停止して新規の設定で起動します。このとき HADB サービスは中断されません。

`ConnectionTrace` または `SQLTraceMode` を設定した場合、順次再起動は実行されませんが、変更は Application Server インスタンスから作成された新規の HADB 接続に対してのみ反映されます。

設定属性

次の表に、`hadbm set` での変更と `hadbm get` での検出が可能な設定属性を一覧表示します。

表 11-8 設定属性

属性	説明	デフォルト	範囲
ConnectionTrace	True に設定すると、クライアント接続 (JDBC、ODBC) が開始または終了したときに、メッセージが HADB 履歴ファイルに記録されます。	False	True または False
CoreFile	デフォルト値を変更しないでください。	False	True または False
DatabaseName	データベースの名前。	hadb	
DataBufferPoolSize	共用メモリーに割り当てられるデータバッファープールのサイズ。	200M バイト	16 ~ 2047M バイト
DataDeviceSize	ノードのデバイスサイズを指定します。推奨される DataDeviceSize のサイズについては、214 ページの「デバイスサイズの指定」を参照してください。 最大値は、256G バイトとオペレーティングシステムの最大ファイルサイズの小さい方です。最小値は次のとおりです。 $(4 \times \text{LogbufferSize} + 16\text{M バイト}) / n$ ここで、 n はデータデバイスの番号です。	1024M バイト	32 ~ 262144M バイト
PackageName	データベースが使用する HADB ソフトウェアパッケージの名前。	V4.x.x.x	なし
DevicePath	デバイスの場所。デバイスは次のとおりです。 <ul style="list-style-type: none"> ■ データデバイス (DataDevice) ■ ノード内部ログデバイス (NiLogDevice) ■ 関係代数クエリーデバイス (RelalgDevice) 	Solaris および Linux: /var/opt/SUNWhadb Windows: C:\Sun\AppServer\SUNWhadb\vers。ここで、vers は HADB バージョン番号です。	

表 11-8 設定属性 (続き)

属性	説明	デフォルト	範囲
EagerSessionThreshold	<p>通常または高速処理 (eager) アイドルセッション有効期限を使用するかどうかを判別します。</p> <p>通常のアイドルセッション有効期限では、アイドル状態が SessionTimeout 秒を超過したセッションが期限切れとなります。</p> <p>並行セッションの数がセッション最大数の EagerSessionThreshold パーセントを超えている場合は、アイドル状態が EagerSessionTimeout 秒を超過したセッションが期限切れとなります。</p>	NumberOfSessions 属性の半分	0 ~ 100
EagerSessionTimeout	<p>高速処理 (eager) セッション有効期限を使用している場合に、データベース接続がアイドル状態になってから期限切れになるまでの秒数。</p>	120 秒	0 ~ 2147483647 秒
EventBufferSize	<p>データベースイベントが記録されるイベントバッファのサイズ。0 に設定すると、イベントバッファへのロギングは実行されません。</p> <p>障害が起きている間、イベントバッファはダンプされます。これは、障害の原因に関する有用な情報を提供し、試験的な配備の際に役立ちます。</p> <p>イベントをメモリーに書き込むと、パフォーマンスが犠牲になります。</p>	0M バイト	0 ~ 2097152M バ イト
HistoryPath	<p>HADB 履歴ファイルの場所。このファイルの内容は、情報、警告、およびエラーメッセージです。</p> <p>これは読み取り専用属性です。</p>	<p>Solaris および Linux: /var/opt/SUNWhadb</p> <p>Windows: REPLACEDIR (実行時に実際の URL に置換される)</p>	
InternalLogbufferSize	<p>ノード内部ログデバイスのサイズ。データの格納に関連する操作のトラックが保持されます。</p>	12M バイト	4 ~ 128M バ イト
JdbcUrl	<p>データベースの JDBC 接続 URL。</p> <p>これは読み取り専用属性です。</p>	なし	
LogbufferSize	<p>ログバッファのサイズ。データに関連する操作のトラックが保持されます。</p>	48M バイト	4 ~ 2048M バイト

表 11-8 設定属性 (続き)

属性	説明	デフォルト	範囲
MaxTables	HADB データベース内で許可される表の最大数。	1100	100 ~ 1100
NumberOfDatadevices	HADB ノードで使用されるデータデバイスの数。 これは読み取り専用属性です。	1	1 ~ 8
NumberOfLocks	HADB ノードによって割り当てられるロックの数。	50000	20000 ~ 1073741824
NumberOfSessions	HADB ノード用に開くことが可能なセッション (データベース接続) の最大数。	100	1 ~ 10000
PortBase	異なる HADB プロセス用に異なるポート番号を作成する際に使用するベースポート番号。 これは読み取り専用属性です。	15200	10000 ~ 63000
RelalgDeviceSize	関係代数クエリーに使用するデバイスのサイズ。	128M バイト	32 ~ 262144M バイト
SessionTimeout	通常のセッション有効期限を使用している場合に、データベース接続がアイドル状態になってから期限切れになるまでの時間。	1800 秒	0 ~ 2147483647 秒
SQLTraceMode	履歴ファイルに書き込まれる実行された SQL クエリーに関する情報の量。 SHORT に設定すると、SQL セッションのログインとログアウトが記録されます。FULL に設定すると、準備中および実行中のすべての SQL が、パラメータ値を含めて記録されます。	なし	NONE/SHORT/FULL
StartRepairDelay	スペアノードが、障害の発生したアクティブノードに対してノード復旧の実行を許可する最大時間。障害の発生したノードがこの時間内に回復できない場合、スペアノードが障害の発生したノードのミラーからデータのコピーを開始してアクティブになります。デフォルト値を変更しないことをお勧めします。	20 秒	0 ~ 100000 秒

表 11-8 設定属性 (続き)

属性	説明	デフォルト	範囲
StatInterval	<p>HADB ノードがスループットと応答時間の統計情報を履歴ファイルに書き込む間隔。無効にする場合は、0 に設定します。</p> <p>次に示すのは、統計情報の行の例です。</p> <pre>Req-reply time: # 123, min= 69 avg= 1160 max= 9311 %=100.0</pre> <p>ハッシュ記号 (#) の後ろの数字は、StatInterval の間に処理された要求の数です。次の 3 つの数字は、StatInterval の間に完了したトランザクションが処理に要した時間の最小値、平均値、最大値をマイクロ秒で表したものです。パーセント記号 (%) の後ろの数字は、StatInterval の間に 15 ミリ秒以内に正常に完了したトランザクションの数です。</p>	600 秒	0 ~ 600 秒
SyslogFacility	<p>syslog にレポートするとき使用する機能。syslog デモンを設定しておくことをお勧めします (詳細は man syslogd.conf を参照)。</p> <p>同じマシン上で実行中のほかのアプリケーションによって使用されていない機能を使用します。</p> <p>syslog ログイングを無効にするには、none に設定します。</p>	local0	local0、local1、local2、
SysLogging	True に設定すると、HADB ノードは情報をオペレーティングシステムの syslog ファイルに書き込みます。	True	True または False
SysLogLevel	オペレーティングシステムの syslog ファイルに保存される HADB メッセージの最小レベル。指定したレベル以上のすべてのメッセージが記録されます。たとえば、「info」に設定した場合は、すべてのメッセージが記録されます。	警告	nonealert errorwarninginfo
SyslogPrefix	HADB によって書き込まれるすべての syslog メッセージの前に挿入されるテキスト文字列。	hadb -dbname	
TakeoverTime	ノードに障害が発生してから、処理がミラーに引き継がれるまでの時間。デフォルト値を変更しないでください。	10000 (ミリ秒)	500 ~ 16000 ミリ秒

JDBC 接続プールの設定

Enterprise Server は Java Database Connectivity (JDBC) API を使用して HADB と通信します。asadmin configure-ha-cluster コマンドは、クラスタの *cluster-name* の JDBC 接続プールを HADB 用に自動的に作成します。接続プールの名前は「*cluster-name-hadb-pool*」です。JDBC リソースの JNDI URL は「jdbc/*cluster-name-hastore*」です。

通常、接続プールは初期設定のままで十分です。ノードを追加する場合は、通常プールサイズを変更して、アクティブな HADB ノードがそれぞれ 8 つの接続を持つようにします。236 ページの「ノードの追加」を参照してください。

この章の内容は次のとおりです。

- 222 ページの「JDBC URL の取得」
- 222 ページの「接続プールの作成」
- 224 ページの「JDBC リソースの作成」

接続プールと JDBC リソースに関する一般情報については、『[Sun GlassFish Enterprise Server 2.1 高可用性 \(HA\) 管理ガイド](#)』を参照してください。

JDBC URL の取得

JDBC 接続ツールをセットアップする前に、次のように hadbm get コマンドを使用して、HADB の JDBC URL を決定する必要があります。

```
hadbm get JdbcUrl [dbname]
```

次に例を示します。

```
hadbm get JdbcUrl
```

このコマンドを実行すると、JDBC URL が次の書式で表示されます。

```
jdbc:sun:hadb:host:port,  
host:port,...
```

jdbc:sun:hadb: 接頭辞を削除し、*host:port, host:port...* の部分を、表 11-10 で説明されている serverList 接続プールの値として使用します。

接続プールの作成

次の表に、HADB 用に必須の接続プール設定を要約します。ノードを追加する際には「通常プールサイズ」を変更し、それ以外の設定は変更しないでください。

表 11-9 HADB 接続プール設定

設定	HADB 用に必要な値
名前	HADB JDBC リソースの「プール名」設定がこの名前を参照している必要があります
データベースベンダー	HADB 4.4
グローバルトランザクションのサポート	チェックしない/False
データソースクラス名	com.sun.hadb.jdbc.ds.HadbDataSource
通常プールサイズ	アクティブな HADB ノードごとに 8 つの接続を使用します。詳細については、『System Deployment Guide』を参照してください。
接続検証が必要	チェックする/True
検証方法	meta-data
テーブル名	指定しない
すべての障害ですべての接続を閉じる	チェックしない/False
トランザクション遮断	repeatable-read
遮断レベルを保証	チェックする/True

次の表に、HADB 用に必須の接続プールのプロパティを要約します。ノードを追加する際には `serverList` を変更し、それ以外のプロパティは変更しないでください。

表 11-10 HADB 接続プールプロパティ

プロパティ	説明
username	asadmin create-session-store コマンドに使用する storeuser の名前。
password	asadmin create-session-store コマンドに使用するパスワード (storepassword)。
serverList	HADB の JDBC URL。この値を特定するには、 222 ページの「JDBC URL の取得」 を参照してください。 データベースにノードを追加する場合は、この値を変更する必要があります。 236 ページの「ノードの追加」 を参照してください。
cacheDatabaseMetaData	必要に応じて false を指定すると、 <code>Connection.getMetaData()</code> を呼び出すことによってデータベースが呼び出され、接続が有効になります。

表 11-10 HADB 接続プールプロパティ (続き)

プロパティ	説明
<code>eliminateRedundantEndTransaction</code>	必要に応じて <code>true</code> を指定すると、重複するコミットおよびロールバックの要求の削除、およびトランザクションが開いていない場合にはそれらの要求の無視により、パフォーマンスが向上します。
<code>maxStatement</code>	開いている接続あたりのドライバプールにキャッシュされる文の最大数。このプロパティは <code>20</code> に設定します。

例 11-5 接続プールの作成

次に示すのは、HADB JDBC 接続プールを作成する `asadmin create-jdbc-connection-pool` コマンドの例です。

```
asadmin create-jdbc-connection-pool
--user adminname --password secret
--datasourceclassname com.sun.hadb.jdbc.ds.HadbDataSource
--steadypoolsize=32
--isolationlevel=repeatable-read
--isconnectvalidatereq=true
--validationmethod=meta-data
--property username=storename:password=secret456:serverList=
host\:port,host\:port,
host\:port,host\:port,
host\:port,host\:port
:cacheDatabaseMetaData=false:eliminateRedundantEndTransaction=true hadbpool
```

Solaris では、プロパティ値に含まれるコロン文字 (:) は 2 つの円記号 (\) でエスケープします。Windows では、コロン文字 (:) を 1 つの円記号 (\) でエスケープします。

JDBC リソースの作成

次の表に、HADB 用に必須の JDBC リソース設定を要約します。

表 11-11 HADB JDBC リソース設定

設定	説明
JNDI 名	セッション持続性設定のデフォルトの JNDI 名は <code>jdbc/hastore</code> です。このデフォルト名または別の名前を使用することができます。 可用性サービスを使用可能にするには、 <code>store-pool-jndi-name</code> 「持続性ストア」プロパティの値にもこの JNDI 名を指定する必要があります。

表 11-11 HADB JDBC リソース設定 (続き)

設定	説明
プール名	リストから、この JDBC リソースが使用する HADB 接続プールの名前 (または ID) を選択します。詳細については、178 ページの「ネットワーク冗長性の設定」を参照してください。
データソースが有効	チェックする/True

HADBの管理

通常、ネットワーク、ハードウェア、オペレーティングシステム、または HADB ソフトウェアを交換またはアップグレードするには、管理オペレーションを実行する必要があります。次の節では、さまざまな管理オペレーションについて説明します。

- 225 ページの「ドメインの管理」
- 226 ページの「ノードの管理」
- 229 ページの「データベースの管理」
- 233 ページの「データセッション破損からの回復」

ドメインの管理

HADB ドメインで、次の操作を実行できます。

- ドメインの作成。詳細は、209 ページの「管理ドメインの作成」を参照してください。
- 225 ページの「ドメインの拡張」
- 226 ページの「ドメインの削除」
- 226 ページの「ドメイン内のホストの一覧表示」
- 226 ページの「ドメインからのホストの削除」

コマンドオプションの説明は、205 ページの「セキュリティーオプション」および 207 ページの「一般的なオプション」を参照してください。

ドメインの拡張

`extenddomain` を使用して、既存の管理ドメインにホストを追加します。コマンド構文は次のとおりです。

```
hadbm extenddomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
hostlist
```

HADB ホストの IP アドレスは、IPv4 アドレスである必要があります。

詳細については、[hadbm-extenddomain\(1\)](#)を参照してください。

ドメインの削除

`deletedomain` を使用して、管理ドメインを削除します。コマンド構文は次のとおりです。

```
hadbm deletedomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

詳細については、[hadbm-deletedomain\(1\)](#)を参照してください。

ドメインからのホストの削除

`reducedomain` を使用して、管理ドメインからホストを削除します。コマンド構文は次のとおりです。

```
hadbm reducedomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
host_list
```

詳細については、[hadbm-reducedomain\(1\)](#)を参照してください。

ドメイン内のホストの一覧表示

`listdomain` を使用して、管理ドメイン内に定義されているすべてのホストを一覧表示します。コマンド構文は次のとおりです。

```
hadbm listdomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

詳細については、[hadbm-listdomain\(1\)](#)を参照してください。

ノードの管理

個々のノードに対して、次の操作を実行できます。

- [227 ページの「ノードの起動」](#)
- [228 ページの「ノードの停止」](#)
- [228 ページの「ノードの再起動」](#)

ノードの起動

ハードウェアやソフトウェアのアップグレードや交換のためにホストをオフラインにしたために、停止した HADB ノードを手動で起動する必要がある場合があります。また、二重障害以外の何らかの理由でノードが再起動に失敗すると、手動でのノードの起動が必要な場合があります。二重障害から回復する方法の詳細については、[231 ページの「データベースの解除」](#)を参照してください。

たいいていの場合には、まず `normal` 起動レベルを使用してノードの起動を試行することをお勧めします。`normal` 起動レベルで失敗するかまたはタイムアウトになる場合には、`repair` 起動レベルを使用する必要があります。

データベース内のノードを起動するには、`hadbm startnode` コマンドを使用します。この構文は次のとおりです。

```
hadbm startnode
  [--adminpassword=password | --adminpasswordfile=file]
  [--agent=maurl]
  [--startlevel=level]
  nodeno
  [dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`nodeno` オペランドには起動するノードの番号を指定します。`hadbm status` を使用すると、データベース内のすべてのノードの番号を表示できます。

詳細については、[hadbm-startnode\(1\)](#)を参照してください。

起動レベルオプション

`hadbm startnode` コマンドには、ノードの起動レベルを指定する、1 つの特別なオプション `--startlevel` (省略形 `-l`) があります。

ノード起動レベルは次のとおりです。

- **normal** (デフォルト): ノード上 (メモリー内およびディスク上のデータデバイスファイル内) でローカルに検出されたデータを使用してノードを起動し、欠落している最新の更新内容をミラーとの間で同期します。
- **repair**: ノードに対して、ローカルデータの破棄およびミラーからのそのデータのコピーを強制します。
- **clear**: ノードのデバイスを再初期化し、ミラーノードから強制的にデータを修復します。デバイスファイルが損傷を受けたかまたはデバイスファイルを含むディスクを交換したため、デバイスファイルを初期化する必要がある場合に使用します。

その他のコマンドオプションの説明は、[207 ページの「一般的なオプション」](#)を参照してください。

例11-6 ノードを起動する例

```
hadbm startnode 1
```

ノードの停止

ホストマシンのハードウェアやソフトウェアを修復またはアップグレードするために、ノードの停止が必要な場合があります。ノードを停止するには、`hadbm stopnode` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm stopnode  
[--adminpassword=password | --adminpasswordfile=file]  
[--agent=maurl]  
[--no-repair]  
nodeno  
[dbname]
```

nodeno オペランドには停止するノードの番号を指定します。このノード番号のミラーノードが実行中でなければなりません。`hadbm status` を使用すると、データベース内のすべてのノードの番号を表示できます。

dbname オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`hadbm stopnode` コマンドには、停止したノードを置き換えるスペアノードがないことを示す、1つの特別なオプション `--no-repair` (省略形 `-R`) があります。このオプションを使用しない場合は、スペアノードが起動して、停止したノードの機能を引き継ぎます。

その他のコマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。詳細については、[hadbm-stopnode\(1\)](#) を参照してください。

例11-7 ノードを停止する例

```
hadbm stopnode 1
```

ノードの再起動

CPU の過剰な消費など異常な動作が見られる場合には、ノードの再起動が必要なこともあります。

データベース内のノードを再起動するには、`hadbm restartnode` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm restartnode  
[--adminpassword=password | --adminpasswordfile=file]  
[--agent=maurl]  
[--startlevel=level]  
nodeno  
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`nodeno` オペランドには再起動するノードの番号を指定します。 `hadbm status` を使用すると、データベース内のすべてのノードの番号を表示できます。

`hadbm restartnode` コマンドには、ノードの起動レベルを指定する、1つの特別なオプション `--startlevel` (省略形 `-l`) があります。詳細については、227 ページの「[起動レベルオプション](#)」を参照してください。

その他のコマンドオプションの説明は、207 ページの「[一般的なオプション](#)」を参照してください。詳細については、[hadbm-restartnode\(1\)](#) を参照してください。

例 11-8 ノードを再起動する例

```
hadbm restartnode 1
```

データベースの管理

HADB データベースで、次の操作を実行できます。

- 229 ページの「[データベースの起動](#)」
- 230 ページの「[データベースの停止](#)」
- 230 ページの「[データベースの再起動](#)」
- 231 ページの「[データベースの一覧表示](#)」
- 231 ページの「[データベースの解除](#)」
- 233 ページの「[データベースの削除](#)」

データベースの起動

データベースを起動するには、`hadbm start` コマンドを使用します。このコマンドは、データベースが停止する前に実行していたすべてのノードを起動します。停止後にデータベースを起動しても、個別に停止されてオフラインになっているノードは起動されません。

コマンド構文は次のとおりです。

```
hadbm start
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

コマンドオプションの説明は、207 ページの「[一般的なオプション](#)」を参照してください。詳細については、[hadbm-start\(1\)](#) を参照してください。

例11-9 データベースを起動する例

```
hadbm start
```

データベースの停止

データベースを停止してから起動するまでの、データベースが停止している間は、データを使用することができません。データを使用可能にするために、[230 ページ](#)の「[データベースの再起動](#)」で説明されているようにデータベースを再起動できます。

次の目的で、データベースを停止します。

- データベースを削除する。
- すべてのHADB ノードに影響するシステム保守を実行する。

データベースを停止する前に、そのデータベースを使用する依存 Enterprise Server インスタンスを停止するか、またはそれらのインスタンスが `ha` 以外の「持続型」を使用するように設定します。

データベースを停止すると、データベース内で実行中のすべてのノードが停止し、データベースの状態が「停止中」になります。データベースの状態の詳細については、[241 ページ](#)の「[データベースの状態](#)」を参照してください。

データベースを停止するには、`hadbm stop` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm stop  
[--adminpassword=password | --adminpasswordfile= file]  
[--agent=maurl]  
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

コマンドオプションの説明は、[207 ページ](#)の「[一般的なオプション](#)」を参照してください。詳細については、[hadbm-stop\(1\)](#)を参照してください。

例11-10 データベースを停止する例

```
hadbm stop
```

データベースの再起動

タイムアウトの問題が解消されないなどの異常な動作が見られる場合には、データベースの再起動が必要なこともあります。再起動で問題が解決する場合もあります。

データベースを再起動すると、データベースとそのデータは引き続き使用可能です。HADBを停止してから起動するまでの、HADBが停止している間は、データおよびデータベースサービスを使用することができません。これは、デフォルトで `hadbm restart` がノードの順次再起動を実行するためです。このときノードは1つずつ順番に停止して起動します。一方、`hadbm stop` を実行した場合には、すべてのノードが同時に停止します。

データベースを再起動するには、`hadbm restart` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm restart
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--no-rolling]
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

特別なオプション `--no-rolling` (省略形 `-g`) は、すべてのノードの一括再起動を指定しますが、サービスの低下を招きます。このオプションを使用しない場合は、データベース内の各ノードが現在の状態またはよりよい状態で再起動されます。

その他のコマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。詳細については、`hadbm-restart(1)` を参照してください。

次に例を示します。

```
hadbm restart
```

データベースの一覧表示

HADB インスタンス内のすべてのデータベースを一覧表示するには、`hadbm list` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm list
[--agent=maurl]
[--adminpassword=password | --adminpasswordfile=file]
```

コマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。詳細については、`hadbm-list(1)` を参照してください。

データベースの解除

次の場合には、データベースを解除します。

- `hadbm status` コマンドから、データベースが稼働していないことがわかる。241 ページの「HADBの状態の取得」を参照してください。

- 複数のノードが応答せず、長時間待機状態である。
- セッションデータ破損から回復している。233 ページの「データセッション破損からの回復」を参照してください。

`hadbm clear` コマンドはデータベースノードを停止し、データベースデバイスを解除してから、ノードを起動します。このコマンドはHADB内のApplication Server スキーマデータストア(テーブル、ユーザー名、パスワードを含む)を消去します。`hadbm clear` を実行したあとで、`asadmin configure-ha-cluster` を使用してデータスキーマを再作成し、JDBC 接続プールを再設定し、セッション持続性ストアを再ロードしてください。

コマンド構文は次のとおりです。

```
hadbm clear [--fast] [--spares=number]
[--dbpassword=password | --dbpasswordfile= file]
[--adminpassword=password | --adminpasswordfile= file]
[--agent=maurl]
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

次の表で、`hadbm clear` の特別なコマンドオプションについて説明します。その他のオプションの説明は、207 ページの「一般的なオプション」を参照してください。

詳細については、`hadbm-clear(1)` を参照してください。

表 11-12 `hadbm clear` オプション

オプション	説明	デフォルト
<code>--fast</code>	データベースを初期化している間、デバイスの初期化をスキップします。ディスク記憶装置デバイスが破損している場合は、使用しないでください。	なし
<code>-F</code>		
<code>--spares= number</code>	再初期化されたデータベースに配置されるスペアノードの数。この数は、偶数かつデータベース内のノードの数より少ない数である必要があります。	前回のスペアの数
<code>-s</code>		

次に例を示します。

```
hadbm clear --fast --spares=2
```

データベースの削除

既存のデータベースを削除するには、`hadbm delete` コマンドを使用します。このコマンドは、データベースの設定ファイル、デバイスファイル、および履歴ファイルを削除し、共用メモリーリソースを解放します。削除対象のデータベースは、存在していてかつ停止している必要があります。230 ページの「データベースの停止」を参照してください。

コマンド構文は次のとおりです。

```
hadbm delete
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

コマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。詳細については、`hadbm-delete(1)`を参照してください。

例 11-11 データベースを削除する例

次のコマンドを見てください。

```
hadbm delete
```

は、デフォルトのデータベース `hadb` を削除します。

データセッション破損からの回復

次のような症状が見られる場合、セッションデータが破損している可能性があります。

- アプリケーションがセッション状態を保存しようとするたびに、Enterprise Server システムログ (`server.log`) にエラーメッセージが表示される。
- サーバーログのエラーメッセージに、セッションが見つからなかったり、セッション活性化中にセッションをロードできなかったことが示されている。
- 以前非アクティブにされていたセッションをアクティブにしたところ、そのセッションに空のセッションデータまたは不正なセッションデータが含まれている。
- インスタンスに障害が発生する際に、処理を継続したセッションに、空のまたは不正なセッションデータが含まれている。
- インスタンスに障害が発生し、処理を継続したセッションをロードしようとするインスタンスがエラーを起こし、サーバーログに、セッションが見つからなかったりロードできなかったことが示されている。

▼ セッションストアを一貫性のある状態に戻すには

セッションストアが破損していると判断する場合には、次の手順に従って一貫性のある状態に戻します。

1 セッションストアを消去します。

この処置で問題が正されたかどうかを判定します。問題が正された場合は、これで処置は終わりです。引き続きサーバーログにエラーが表示されるなど、問題が正されていない場合は、処置を継続します。

2 すべてのノード上のデータスペースを再初期化して、データベース内のデータを消去します。

[231 ページの「データベースの解除」](#)を参照してください。

この処置で問題が正されたかどうかを判定します。問題が正された場合は、これで処置は終わりです。引き続きサーバーログにエラーが表示されるなど、問題が正されていない場合は、処置を継続します。

3 データベースを削除して再作成します。

[233 ページの「データベースの削除」](#)および[210 ページの「データベースの作成」](#)を参照してください。

HADBの拡張

元のHADB設定を拡張する2つの理由があります。

- 保存されているセッションデータのボリュームが増えて、データデバイス内の既存の記憶スペースを超過している。データデバイスが満杯になったために、トランザクションが異常終了し始める可能性があります。
- ユーザー側の負荷が増えて、システムリソースが使い果たされる。さらにホストを追加することが必要です。

この節では、Enterprise Server クラスタまたはデータベースを停止せずにHADBを拡張する方法について説明します。特に、次の点を扱います。

- [235 ページの「既存ノードへの記憶スペースの追加」](#)
- [235 ページの「マシンの追加」](#)
- [236 ページの「ノードの追加」](#)
- [238 ページの「データベースの再断片化」](#)
- [239 ページの「データベースの再作成によるノードの追加」](#)

[248 ページの「HADB マシンの管理」](#)にある関連情報も参照してください。

既存ノードへの記憶スペースの追加

次のような場合に、HADB 記憶スペースを追加します。

- ユーザートランザクションが、次のいずれかのエラーメッセージを出して繰り返し異常終了する。
 - 4592: データデバイスに空きブロックがありません
 - 4593: データデバイスに未予約ブロックがありません
- `hadbm deviceinfo` コマンドが終始空きサイズの不足を報告する。243 ページの「[デバイス情報の取得](#)」を参照してください。

ノードに使用されていないディスクスペースがある場合やディスク容量を追加する場合は、既存のノードに記憶スペースを追加することもできます。推奨されているデータデバイスサイズについては、214 ページの「[デバイスサイズの指定](#)」を参照してください。

ノードに記憶スペースを追加するには、`hadbm set` コマンドを使用してデータデバイスサイズを増やします。

コマンド構文は次のとおりです。

```
hadbm set DataDeviceSize=size
```

ここで、*size* は M バイト単位でのデータデバイスサイズです。

コマンドオプションの説明は、207 ページの「[一般的なオプション](#)」を参照してください。

FaultTolerant またはそれ以上のシステム状態にあるデータベースでは、データデバイスサイズを変更することによって、データや可用性を犠牲にすることなくシステムはアップグレードされます。再設定の間も、データベースは稼働状態を維持します。FaultTolerant またはそれ以上の状態ではないシステムでデバイスサイズを変更すると、データの喪失が生じます。データベースの状態の詳細については、241 ページの「[データベースの状態](#)」を参照してください。

例 11-12 データデバイスサイズを設定する例

次に示すのは、データデバイスサイズを設定するコマンドの例です。

```
hadbm set DataDeviceSize=1024
```

マシンの追加

HADB が処理能力や記憶容量をさらに必要としている場合には、マシンを追加します。HADB を実行するマシンを新たに追加するには、第 10 章「[高可用性 \(HA\) データベースのインストールと設定](#)」で説明されている手順に従って、HADB パッケージ

をインストールします。このとき、Enterprise Server を一緒にインストールしてもしなくてもかまいません。代替のノードトポロジについては、docs.sun.com で『Sun GlassFish Enterprise Server 2.1 配備計画ガイド』の「トポロジの選択」の章を参照してください。

▼ 既存の HADB インスタンスに新たなマシンを追加する

- 1 新規ノード上で管理エージェントを起動します。
- 2 管理ドメインを新規ホストへ拡張します。
詳細については、`hadbm extenddomain` コマンドを参照してください。
- 3 この新規ホスト上で新規ノードを起動します。
詳細については、236 ページの「ノードの追加」を参照してください。

ノードの追加

HADB システムの処理能力と記憶容量を増やすには、新規ノードを作成してデータベースに追加します。

ノードを追加したあとで、HADB JDBC 接続プールの次のプロパティを更新します。

- `serverlist` プロパティ。
- 通常プールサイズ。一般には、新規ノードにつき8つの接続を追加します。

ノードを追加するには、`hadbm addnodes` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm addnodes [--no-refragment] [--spares=sparecount]
[--historypath=path]
[--devicepath=path]
[--set=attr-name-value-list]
[--dbpassword=password | --dbpasswordfile=file ]
[--adminpassword=password | --adminpasswordfile=file]
--hosts=hostlist [dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。データベースの状態は、`HAFaultTolerant` または `FaultTolerant` である必要があります。データベースの状態の詳細については、241 ページの「データベースの状態」を参照してください。

`--devicepath` と `--historypath` オプションを指定しない場合、新規ノードは既存データベースと同じデバイスパスを持ち、同じ履歴ファイルを使用します。

ノードを追加すると、既存データの再断片化と再配布が実行されて、システムに新規ノードが組み込まれます。オンラインで再断片化を実行するには、再断片化が終了するまで古いデータと新しいデータを同時に保持できるだけの十分なスペースがHADBノードのディスクに必要です。つまり、ユーザーデータサイズは、ユーザーデータに使用可能なスペースの50%を超えてはいけません。詳細については、[243 ページの「デバイス情報の取得」](#)を参照してください。

注-システムの負荷が軽いときにノードを追加するのが最善です。

例 11-13 ノードを追加する例

次に例を示します。

```
hadbm addnodes -adminpassword=password --hosts n6,n7,n8,n9
```

次の表で、`hadbm addnodes` の特別なコマンドオプションについて説明します。その他のオプションの説明は、[207 ページの「一般的なオプション」](#)を参照してください。

表 11-13 hadbm addnodes オプション

オプション	説明	デフォルト
<code>--no-refragment</code> <code>-r</code>	ノード作成中はデータベースを再断片化しないでください。その場合には、あとで <code>hadbm refragment</code> コマンドを使用してデータベースを再断片化し、新規ノードを使用します。再断片化の詳細については、 238 ページの「データベースの再断片化」 を参照してください。 再断片化するための十分なデバイススペースがない場合は、より多い数のノードを持つデータベースを作成し直します。 239 ページの「データベースの再作成によるノードの追加」 を参照してください。	なし
<code>--spares= number</code> <code>-s</code>	すでに存在するスペアノードに追加する新規スペアノードの数。この数は、偶数、かつ追加するノードの数以下でなければなりません。	0

表 11-13 hadbm addnodes オプション (続き)

オプション	説明	デフォルト
--devicepath= <i>path</i> -d	<p>デバイスへのパス。デバイスは次のとおりです。</p> <ul style="list-style-type: none"> ■ DataDevice ■ NiLogDevice (ノード内部ログデバイス) ■ RelalgDevice (関係代数クエリデバイス) <p>このパスはすでに存在していて、書き込み可能であることが必要です。このパスをノードまたはデバイスごとに異なる設定にする場合は、215 ページの「異機種システム混在デバイスパスの設定」を参照してください。</p>	<p>Solaris および Linux: <i>HADB_install_dir/device</i></p> <p>Windows: C:\Sun\AppServer\SUNWhadb\vers。ここで、<i>vers</i> は HADB バージョン番号です。</p>
--hosts= <i>hostlist</i> -H	<p>データベース内の新規ノード用の新しいホスト名を一覧にしたコンマ区切りリスト。リスト中のコンマで区切られた項目ごとに1つのノードが作成されます。ノードの数は偶数でなければなりません。HADB ホストの IP アドレスは、IPv4 アドレスである必要があります。</p> <p>重複するホスト名を使用すると、同じマシン上に異なるポート番号が指定された複数のノードが作成されます。同じマシン上のノードがミラーノードではないことを確認してください。</p> <p>奇数番号のノードが一方の DRU に配置され、偶数番号のノードは他方の DRU に配置されます。--spares を使用すると、もっとも大きい番号のノードが新規スベアノードとなります。</p> <p>二重のネットワークインタフェースを持つデータベースを作成した場合も、同じ方法で新規ノードを構成する必要があります。178 ページの「ネットワーク冗長性の設定」を参照してください。</p>	なし

データベースの再断片化

データベースを再断片化して、新たに作成したノードにデータを格納します。再断片化により、すべてのアクティブなノードにデータが均一に分散します。

データベースを再断片化するには、`hadbm refragment` コマンドを使用します。コマンド構文は次のとおりです。

```
hadbm refragment [--dbpassword=password | --dbpasswordfile=file]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

`dbname` オペランドにはデータベース名を指定します。デフォルトは `hadb` です。データベースの状態は、`HAFaultTolerant` または `FaultTolerant` である必要があります。データベースの状態の詳細については、241 ページの「HADBの状態の取得」を参照してください。

コマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。詳細については、`hadbm-refragment(1)` を参照してください。

オンラインで再断片化を実行するには、再断片化が終了するまで古いデータと新しいデータを同時に保持できるだけの十分なスペースが HADB ノードのディスクに必要です。つまり、ユーザーデータサイズは、ユーザーデータに使用可能なスペースの 50% を超えてはいけません。詳細については、243 ページの「デバイス情報の取得」を参照してください。

注- システムの負荷が軽いときにデータベースを再断片化するのが最善です。

何回試してもこのコマンドが失敗する場合は、239 ページの「データベースの再作成によるノードの追加」を参照してください。

例 11-14 データベースを再断片化する例

次に例を示します。

```
hadbm refragment
```

データベースの再作成によるノードの追加

新規ノードを追加して、データデバイススペースの不足やその他の理由からオンラインでの再断片化が何度も失敗する場合は、新規ノードを持つデータベースを再作成します。これは、既存のユーザーデータとスキーマデータの喪失を招きます。

▼ データベースの再作成によりノードを追加する

次の手順により、プロセス全体の HADB 可用性を維持することができます。

- 1 各 **Enterprise Server** インスタンスに対して、次のようにします。
 - a. ロードバランサの **Enterprise Server** インスタンスを無効にします。
 - b. セッション持続性を無効にします。
 - c. **Enterprise Server** インスタンスを再起動します。
 - d. ロードバランサの **Enterprise Server** インスタンスを再度有効にします。

可用性を維持する必要がない場合は、ロードバランサのすべてのサーバーインスタンスを無効にしてすぐに再度有効にできます。こうすることで、時間を節約するとともに、古いセッションデータのフェイルオーバーを防ぎます。

- 2 [230 ページ](#)の「データベースの停止」で説明されている手順に従って、データベースを停止します。
- 3 [233 ページ](#)の「データベースの削除」で説明されている手順に従って、データベースを削除します。
- 4 [210 ページ](#)の「データベースの作成」で説明されている手順に従って、追加のノードでデータベースを再作成します。
- 5 [222 ページ](#)の「JDBC 接続プールの設定」で説明されている手順に従って、JDBC 接続プールを再設定します。
- 6 セッション持続性ストアを再ロードします。
- 7 各 Enterprise Server インスタンスに対して、次のようにします。
 - a. ロードバランサの Enterprise Server インスタンスを無効にします。
 - b. セッション持続性を有効にします。
 - c. Enterprise Server インスタンスを再起動します。
 - d. ロードバランサの Enterprise Server インスタンスを再度有効にします。

可用性を維持する必要がない場合は、ロードバランサのすべてのサーバーインスタンスを無効にしてすぐに再度有効にできます。こうすることで、時間を節約するとともに、古いセッションデータのフェイルオーバーを防ぎます。

HADBの監視

次の方法で、HADBのアクティビティを監視できます。

- [241 ページ](#)の「HADBの状態の取得」
- [243 ページ](#)の「デバイス情報の取得」
- [245 ページ](#)の「ランタイムリソース情報の取得」

これらの節では、`hadbm status`、`hadbm deviceinfo`、および `hadbm resourceinfo` コマンドについて簡潔に説明します。HADB情報の解釈については、『Sun GlassFish Enterprise Server 2.1』の「Performance」の章を参照してください。

HADBの状態の取得

hadbm status コマンドを使用して、データベースまたはそのノードの状態を表示します。コマンド構文は次のとおりです。

```
hadbm status
[--nodes]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

dbname オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`--nodes` オプション (省略形 `-n`) は、データベース内の各ノードに関する情報を表示します。詳細については、[242 ページの「ノードの状態」](#)を参照してください。その他のコマンドオプションの説明は、[207 ページの「一般的なオプション」](#)を参照してください。

詳細については、[hadbm-status\(1\)](#)を参照してください。

例 11-15 HADB 状態を取得する例

次に例を示します。

```
hadbm status --nodes
```

データベースの状態

データベースの状態には、データベースの現在の状況が要約されます。次の表で、データベースが取りうる状態の種類について説明します。

表 11-14 HADB の状態

データベースの状態	説明
高可用性耐障害 (HAFaultTolerant)	データベースに耐障害性があり、DRU ごとに少なくとも1つのスペアノードを備えている。
耐障害	すべてのミラーノードペアが実行中である。
稼働	各ミラーノードペア内の少なくとも1つのノードが実行中である。
非稼働	1つ以上のミラーノードペアで、両方のノードがなくなっている。 データベースが非稼働状態である場合は、 231 ページの「データベースの解除」 で説明されている手順に従って、データベースを解除します。
停止	データベース内に実行中のノードがない。

表 11-14 HADB の状態 (続き)

データベースの状態	説明
不明	データベースの状態を判定できない。

ノードの状態

--nodes オプションを使用して、`hadbm status` コマンドでデータベース内の各ノードに関する次の情報を表示させます。

- ノード番号
- ノードが実行中であるマシンの名前
- ノードのポート番号
- ノードのロール。ロールとその意味のリストについては、[242 ページの「ノードのロール」](#)を参照してください。
- ノードの状態。状態とその意味のリストについては、[242 ページの「ノードの状態」](#)を参照してください。
- 対応するミラーノードの番号。

次の節に説明されているように、ノードのロールと状態は変更される場合があります。

- [242 ページの「ノードのロール」](#)
- [242 ページの「ノードの状態」](#)

ノードのロール

ノードには作成時にロールが割り当てられます。次のいずれかのロールを担います。

- アクティブ: データを格納し、クライアントアクセスを許可します。アクティブノードはミラー化されたペアになっています。
- スペア: クライアントアクセスを許可しますが、データを格納しません。データデバイスを初期化したあとに、ほかのデータノードを監視して、あるノードが使用不能になれば修復を開始します。
- オフライン: ロールが変更されるまでサービスを提供しません。ふたたびオンラインになったときに、元のロールに戻される場合があります。
- シャットダウン: アクティブとオフラインの中間の段階で、スペアノードによる機能の引き継ぎを待機している状態です。スペアノードによる引き継ぎが完了すると、ノードはオフラインになります。

ノードの状態

ノードは次のいずれかの状態になります。

- 起動中: ノードは起動中です。

- 待機中: ノードは起動レベルを決定できず、オフラインになっています。ノードがこの状態のまま2分を経過した場合は、そのノードを停止し、repairレベルで起動してください。228 ページの「ノードの停止」、227 ページの「ノードの起動」、および 231 ページの「データベースの解除」を参照してください。
- 実行中: ノードはロールに応じたすべてのサービスを提供しています。
- 停止中: ノードは停止処理を行っています。
- 停止: ノードは停止しています。停止したノードの修復は禁止されています。
- 回復中: ノードは回復処理を行っています。ノードに障害が発生した場合、ミラーノードがそのノードの機能を引き継ぎます。障害が発生したノードは、メインメモリーまたはディスク内のデータとログレコードを使用して回復を試行します。また、ミラーノードのログレコードを使用して、障害発生時に実行していたトランザクションの回復に努めます。回復に成功した場合には、そのノードがふたたびアクティブになります。回復が失敗した場合は、ノードの状態が「修復中」に変更されます。
- 修復中: ノードは修復処理を行っています。この操作で、ノードは再初期化され、ミラーノードからデータとログレコードがコピーされます。修復には回復より時間がかかります。

デバイス情報の取得

次の目的で、HADB データ (ディスク記憶装置) デバイスの空き領域を監視します。

- ディスク容量の使用傾向を定期的にチェックする。
- 予防保守の一環として。ユーザー側の負荷が増え、データベース設定の大きさの変更やスケールを考慮している場合。
- データベースを拡大する操作の一部として。hadbm addnodes を実行して新規ノードをシステムに追加する前に、十分なデバイス空間があるかどうかをチェックします。ノードを追加するには、既存のノード上に 40 ~ 50% ほどの空き領域が必要となることを念頭に置いてください。
- 履歴ファイルや server.log ファイルに次のようなメッセージが表示された場合。
 - No free blocks on data devices
 - No unreserved blocks on data devices .

hadbm deviceinfo コマンドを使用して、データデバイス内の空き領域に関する情報を取得します。このコマンドを実行すると、データベースの各ノードについて次の情報が表示されます。

- 割り当て済みの合計デバイスサイズ (Totalsize)。単位は M バイト。
- 空き領域 (Freesize)。単位は M バイト。
- 現在使用されているデバイスの比率 (Usage)

コマンド構文は次のとおりです。

```
hadbm deviceinfo [--details]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl] [dbname]
```

dbname オペランドにはデータベース名を指定します。デフォルトは *hadb* です。

--details オプションを指定すると、次の追加情報が表示されます。

- デバイスが実行した読み取り操作の数。
- デバイスが実行した書き込み操作の数。
- デバイスの名前。

その他のコマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。

詳細については、[hadbm-deviceinfo\(1\)](#)を参照してください。

ユーザーデータ用に使用可能な空き容量を算定するには、合計デバイスサイズから HADB 用に予約済みの容量 (LogBufferSize の 4 倍 + デバイスサイズの 1%) を減算します。ログバッファのサイズがわからない場合は、コマンド `hadbm get logbufferSize` を使用してください。たとえば、合計デバイスサイズが 128M バイトで LogBufferSize が 24M バイトの場合、ユーザーデータ用に使用可能な容量は $128 - (4 \times 24) = 32$ M バイトです。この 32M バイトのうち、半分はレプリケートデータ用に使用され、約 1% が索引用に使用されるため、実ユーザーデータに使用できるのは 25% だけです。

ユーザーデータに使用可能な容量は、合計サイズと予約済みサイズの差です。将来的にデータを再断片化するのであれば、空き容量がユーザーデータに使用可能な領域の 50% にほぼ等しくなるようにする必要があります。再断片化がふさわしくない場合は、データデバイスを最大限度まで活用することができます。システムのデバイス容量が不足すると、リソース消費警告が履歴ファイルに書き込まれます。

HADB の調整に関する詳細については、『*Sun Java System Application Server パフォーマンスチューニングガイド*』を参照してください。

例11-16 デバイス情報を取得する例

コマンド

```
hadbm deviceinfo --details
```

を実行すると、次の例のような結果が表示されます。

NodeNO	Totalsize	Freesize	Usage	NReads	NWrites	DeviceName
0	128	120	6%	10000	5000	C:\Sun\SUNWhadb\hadb.data.0
1	128	124	3%	10000	5000	C:\Sun\SUNWhadb\hadb.data.1
2	128	126	2%	9500	4500	C:\Sun\SUNWhadb\hadb.data.2
3	128	126	2%	9500	4500	C:\Sun\SUNWhadb\hadb.data.3

ランタイムリソース情報の取得

hadbm resourceinfo コマンドは、HADB ランタイムリソース情報を表示します。この情報を使用して、リソースの競合を識別し、パフォーマンス上のボトルネックを削減するのに役立てることができます。詳細は、『Sun GlassFish Enterprise Server 2.1 配備計画ガイド』の「Tuning HADB」の章を参照してください。

コマンド構文は次のとおりです。

```
hadbm resourceinfo [--databuf] [--locks] [--logbuf] [--nilogbuf]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

dbname オペランドにはデータベース名を指定します。デフォルトは hadb です。

次の表で、hadbm resourceinfo の特別なコマンドオプションについて説明します。その他のコマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。

詳細については、[hadbm-resourceinfo\(1\)](#)を参照してください。

表 11-15 hadbm resourceinfo コマンドオプション

オプション	説明
--databuf	データバッファープール情報を表示します。
-d	詳細については、下記 245 ページの「データバッファープール情報」を参照してください。
--locks	ロック情報を表示します。
-l	詳細については、下記 246 ページの「ロック情報」を参照してください。
--logbuf	ログバッファープール情報を表示します。
-b	詳細については、下記 247 ページの「ログバッファープール情報」を参照してください。
--nilogbuf	ノードの内部ログバッファープール情報を表示します。
-n	詳細については、下記 247 ページの「ノード内部ログバッファープール情報」を参照してください。

データバッファープール情報

データバッファープール情報には、次の内容が含まれます。

- NodeNo: ノード番号
- Avail: プール内の使用可能容量の合計。単位は M バイト。

- Free: 使用可能な空き容量。単位はMバイト。
- Access: 起動時から現在までにデータベースがデータバッファーにアクセスした累積回数。
- Misses: データベース起動時から現在までに発生したページフォルトの累積回数。
- Copy-on-Write: チェックポイントのためにデータバッファーに内部的にコピーされたページの累積数。

ユーザーランザクションがレコードに対して操作を実行するときには、そのレコードを含むページはデータバッファープール内になければなりません。そのようなになっていないと、miss、つまりページフォルトが発生します。すると、ディスク上のデータデバイスフィルからページが取り出されるまで、ランザクションは待機する必要があります。

ミスの比率が高い場合は、データバッファープールを増やしてください。ミスのカウンタは累積回数なので、定期的に `hadbm resourceinfo` を実行し、二回分のカウンタの差を調べて、ミスの比率の傾向を確認します。空き容量が非常に少ないとしても、チェックポイントメカニズムによって新たに使用可能なブロックが作成されるので、心配する必要はありません。

例11-17 データバッファープール情報の例

次に例を示します。

```
NodeNo Avail Free Access Misses Copy-on-Write
0 256 128 100000 50000 10001 256 128 110000 45000 950
```

ロック情報

ロック情報には、次の内容が含まれます。

- NodeNo: ノード番号
- Avail: ノード上で使用可能なロックの合計数。
- Free: 使用されていないロックの数。
- Waits: ロックの獲得を待機しているランザクションの数。これは累積数です。

1つのランザクションが、ノード上で利用可能なロックの25%を超えて使用することはできません。そのため、規模の大きい操作を実行するランザクションは、この制限を認識している必要があります。そのようなランザクションはバッチ処理で実行するのが最善です。その場合、それぞれのバッチは別個のランザクションとして扱われ、バッチごとにコミット操作を行うこととなります。このような操作が必要となるのは、`repeatable read` 遮断レベルで実行する `read` 操作と、`delete`、`insert`、および `update` 操作が、ランザクション終了後のみ解放されるロックを使用するからです。

`NumberOfLocks` を変更するには、250 ページの「履歴ファイルの消去と保存」を参照してください。

例11-18 ロック情報の例

次に例を示します。

```
NodeNO Avail Free Waits
0 50000 20000 101 50000 20000 0
```

ログバッファ情報

ログバッファ情報には、次の内容が含まれます。

- NodeNo: ノード番号。
- Available: ログバッファ用に割り当てられたメモリの容量。単位はMバイト。
- Free: 空きメモリの容量。単位はMバイト。

空き容量が非常に少ないとしても、HADBがログバッファの圧縮を開始するので、心配する必要はありません。HADBは、リングバッファの先頭から圧縮を開始し、連続するログレコードに対して圧縮を実行します。ノードが実行していないのにミラーノードが受信しているログレコードをHADBが検出すると、圧縮は続行できなくなります。

例11-19 ログバッファ情報の例

次に例を示します。

```
NodeNO Avail Free
0 16 21 16 3
```

ノード内部ログバッファ情報

ノード内部ログバッファ情報には、次の内容が含まれます。

- ノード番号。
- 利用可能: ログデバイス用に割り当てられたメモリの容量。単位はMバイト。
- Free: 空きメモリの容量。単位はMバイト。

例11-20 内部ログバッファ情報の例

次に例を示します。

```
NodeNO Avail Free
0 16 21 16 3
```

HADB マシンの管理

HADB は、ミラーノードにデータをレプリケートすることによって耐障害性を実現します。本稼働環境では、ミラーノードはミラー化するノードとは別の DRU 上にあります。

障害とは、ハードウェアの故障、停電、オペレーティングシステムの再起動など予期しない出来事のことです。HADB は、単一の障害に対する耐性を備えています。したがって、単一のノード、ミラーノードペアを持たない単一のマシン、同一の DRU に属する 1 つ以上のマシン、単一の DRU 全体などが対象となります。しかし HADB は、二重障害、すなわち 1 つ以上のミラーノードペアで同時に起きた障害からは自動的に回復しません。二重障害が起きた場合は、HADB を解除してセッションストアを再作成する必要があるため、このとき HADB のデータはすべて消去されます。

対象のマシンが 1 つか複数かに応じて、保守手順は異なります。

▼ 単一のマシンに対して保守を実行する

この手順は計画的な保守と予定外の保守の両方に適用でき、それによって HADB の利用が中断されることはありません。

- 1 保守手順を実行し、マシンを稼働状態にします。
- 2 `ma` が実行中であることを確認します。
`ma` が Windows サービスとして実行されているか、または `init.d` スクリプトの下で実行されている場合 (配備環境で推奨されている方法)、おそらくそれはオペレーティングシステムによって起動されています。そうでない場合は `ma` を手動で起動します。196 ページの「[管理エージェントの起動](#)」を参照してください。
- 3 マシン上のすべてのノードを起動します。
詳細については、227 ページの「[ノードの起動](#)」を参照してください。
- 4 ノードがアクティブで実行状態であるかどうかを確認します。
詳細については、241 ページの「[HADB の状態の取得](#)」を参照してください。

▼ すべての HADB マシンに対して計画的な保守を実行する

計画的な保守には、ハードウェアとソフトウェアのアップグレードなどの操作が含まれます。この手順によって HADB の利用が中断されることはありません。

- 1 1つ目の DRU 内の各スペアマシンに対して、[248 ページ](#)の「[単一のマシンに対して保守を実行する](#)」で説明されている手順に従って、単一マシン用の手順を順番に繰り返します。
- 2 1つ目の DRU 内のアクティブな各マシンに対して、[248 ページ](#)の「[単一のマシンに対して保守を実行する](#)」で説明されている手順に従って、単一マシン用の手順を順番に繰り返します。
- 3 2番目の DRU に対して、ステップ1と2を繰り返します。

▼ すべての HADB マシンに対して計画的な保守を実行する

この手順は、HADB が1つまたは複数のマシン上に配置されている場合に適用されます。保守手順の実行中は、HADB サービスが中断されます。

- 1 HADB を停止します。[230 ページ](#)の「[データベースの停止](#)」を参照してください。
- 2 保守手順を実行し、すべてのマシンを稼働状態にします。
- 3 ma が実行中であることを確認します。
- 4 HADB を起動します。
詳細については、[229 ページ](#)の「[データベースの起動](#)」を参照してください。
最後のステップを完了したあとに、HADB はふたたび利用可能になります。

▼ 障害発生時に予定外の保守を実行する

- データベースの状態を確認します。
[241 ページ](#)の「[HADB の状態の取得](#)」を参照してください。
 - データベースの状態が「稼働」またはそれよりよい場合は、次のようにします。
予定外の保守を必要とするマシンに、ミラーノードは含まれません。DRU 別に、障害の発生した各マシンに対して、単一マシン用の保守手順を行います。HADB サービスは中断されません。
 - データベースの状態が「非稼働」の場合は、次のようにします。
予定外の保守を必要とするマシンに、ミラーノードが含まれます。たとえば、HADB 全体が障害の発生した単一のマシンに置かれているようなケースで

す。まず、すべてのマシンを稼働状態にします。次に、HADB を解除して、セッションストアを再作成します。231 ページの「データベースの解除」を参照してください。この手順により、HADB サービスは中断されます。

履歴ファイルの消去と保存

HADB 履歴ファイルには、すべてのデータベース操作とエラーメッセージが記録されます。HADB は既存の履歴ファイルの末尾に記録を追加していくため、時間の経過とともにファイルのサイズは大きくなります。ディスク容量を節約し、ファイルが大きくなりすぎないようにするために、履歴ファイルを定期的に消去および保存します。

データベースの履歴ファイルを消去するには、`hadbm clearhistory` コマンドを使用します。

コマンド構文は次のとおりです。

```
hadbm clearhistory
[--saveto=path]
[dbname]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

dbname オペランドにはデータベース名を指定します。デフォルトは `hadb` です。

`--saveto` オプション (省略形 `-o`) は、古い履歴ファイルを格納するディレクトリを指定します。このディレクトリには適切な書き込み権が必要です。その他のコマンドオプションの説明は、207 ページの「一般的なオプション」を参照してください。

詳細については、`hadbm-clearhistory(1)` を参照してください。

`hadbm create` コマンドの `--historypath` オプションは、履歴ファイルの場所を特定します。履歴ファイルの名前は `dbname.out.nodeno` という形式です。`hadbm create` については、210 ページの「データベースの作成」を参照してください。

履歴ファイルの書式

履歴ファイルの各メッセージには、次の情報が含まれています。

- メッセージを生成した HADB プロセスの省略名。
- メッセージの種類:
 - INF - 一般情報
 - WRN - 警告
 - ERR - エラー
 - DBG - デバッグ情報

- 時刻表示。時刻は、ホストマシンのシステムクロックから取得されます。
- ノードが停止または起動したときにシステムで生じるサービスセットの変更。

リソースの不足に関するメッセージには、文字列「HIGH LOAD」が含まれています。

履歴ファイルに含まれるすべての項目に関する詳しい知識は必要ありません。何らかの理由で履歴ファイルを詳細に分析する必要がある場合には、Sun カスタマサポートにご連絡ください。

索引

A

active-healthcheck-enabled, 84
AddressList, デフォルト JMS ホスト, 160
Apache Web Server
 インストーラによる変更, 120
 インストール後の設定, 123
 セキュリティファイル, 123
 ロードバランサ, 116
 ロードバランサプラグインによって行われる変更, 117-120
asadmin create-jms-host コマンド, 161
asadmin get コマンド, 158
asadmin set コマンド, 158
authPassthroughEnabled, 90

C

cacheDatabaseMetaData プロパティ, 223
checkpoint-at-end-of-method 要素, 154
ConnectionTrace 属性, 218
cookie ベースのスティッキ程度, 73
CoreFile 属性, 218
create-http-lb-config コマンド, 79
create-http-lb-ref コマンド, 81
create-node-agent コマンド, 61

D

DatabaseName 属性, 218
databuf オプション, 245

DataBufferPoolSize 属性, 218
datadevices オプション, 212
DataDeviceSize 属性, 218, 235
dbpassword オプション, 207
dbpasswordfile オプション, 207
default-config 設定, 132
delete-http-lb-ref コマンド, 81
delete-node-agent コマンド, 65
devicepath オプション, 212
DevicePath 属性, 218, 238
devicesize オプション, 212
disable-http-lb-application コマンド, 87
disable-http-lb-server コマンド, 87

E

EagerSessionThreshold 属性, 219
EagerSessionTimeout 属性, 219
EJB コンテナ, 可用性, 151-152
eliminateRedundantEndTransaction プロパティ, 224
enable-http-lb-application コマンド, 82
enable-http-lb-server コマンド, 81
EventBufferSize 属性, 219
export-http-lb-config コマンド, 84

F

fast オプション, 232

G

GMS, 36-38

H

HADB の設定, 177

ネットワーク構成, 178-181

ノードスーパーバイザープロセス, 186-187

時間の同期, 184

HADB 管理エージェント、起動, 188, 196-204

hadbm addnodes コマンド, 236

hadbm clear コマンド, 232

hadbm clearhistory コマンド, 250

hadbm create コマンド, 210

hadbm delete コマンド, 233

hadbm deviceinfo コマンド, 243

hadbm get コマンド, 216

hadbm list コマンド, 231

hadbm refragment コマンド, 238

hadbm resourceinfo コマンド, 245-247

hadbm restart コマンド, 231

hadbm restartnode コマンド, 228

hadbm start コマンド, 229

hadbm startnode コマンド, 227

hadbm status コマンド, 241-243

hadbm stop コマンド, 230

hadbm stopnode コマンド, 228

hadbm コマンド, 204-209

HADB

JDBC URL の取得, 222

JMS データに対する使用, 163-164

データベースの一覧表示, 231

データベースの解除, 232

データベースの起動, 229

データベースの再起動, 231

データベースの削除, 233

データベースの停止, 230

データベース名, 211

データ破損, 233-234

デバイス情報の取得, 243

ノード, 242

ノードの拡張, 235

ノードの起動, 227

ノードの再起動, 228

HADB (続き)

ノードの追加, 236

ノードの停止, 228

ポート割り当て, 216

マシンの追加, 235-236

マシンの保守, 248

リソース情報の取得, 245-247

異機種システム混在デバイスパス, 215

環境変数, 208

監視, 240-247

再断片化, 238

状態の取得, 241-243

接続プールプロパティ, 223-224

接続プール設定, 223

設定, 209-225

属性の設定, 213, 216

二重ネットワーク, 180-181

履歴ファイル, 250

historypath オプション, 212

HistoryPath 属性, 219

hosts オプション, 213, 238

HTTP_LISTENER_PORT プロパティ, 135

HTTP_SSL_LISTENER_PORT プロパティ, 135

HTTP セッション, 分散, 139-140

HTTPS ルーティング, 88-89

HTTPS

セッションフェイルオーバー, 88-89

ルーティング, 88-89

HTTP

HTTPS ルーティング, 88

セッションフェイルオーバー, 88-89

I

IOP_LISTENER_PORT プロパティ, 135

IOP_SSL_MUTUALAUTH_PORT プロパティ, 135

InternalLogbufferSize 属性, 219

IOP_SSL_LISTENER_PORT プロパティ, 135

J

JdbcUrl 属性, 219

JMS ホストリスト, 接続, 160

JMS

- ホストの作成, 160
- 高可用性, 163-164
- 接続フェイルオーバー, 161
- 接続プール, 23, 161
- 設定, 158

JMX_SYSTEM_CONNECTOR_PORT プロパティ, 135

JMX リスナー, ノードエージェント, 68

L

- loadbalancer.xml ファイル, 85
- locks オプション, 245
- logbuf オプション, 245
- LogbufferSize 属性, 219

M

- maxStatement プロパティ, 224
- MaxTables 属性, 220
- Microsoft Internet Information Services (IIS)、負荷分散のための変更, 127

N

- nilogbuf オプション, 245
- no-refragment オプション, 237
- no-repair オプション, 228
- nodes オプション, 242
- number-healthcheck-retries, 84
- NumberOfDatadevices 属性, 220
- NumberOfLocks 属性, 220
- NumberOfSessions 属性, 220

P

- password プロパティ, 223
- portbase オプション, 213
- Portbase 属性, 220

R

- RelalgdeviceSize 属性, 220
- rewrite-location プロパティ, 92

S

- saveto オプション, 250
- serverList プロパティ, 223
- SessionTimeout 属性, 220
- set オプション, 213, 215
- spares オプション, 213, 232, 237
- SQLTraceMode 属性, 220
- start-node-agent コマンド, 64
- startlevel オプション, 227, 229
- StartRepairDelay 属性, 220
- StatInterval 属性, 221
- stop-node-agent コマンド, 64
- sun-ejb-jar.xml ファイル, 154
- Sun Java System Message Queue, コネクタ, 158
- sun-passthrough.properties ファイル、ログレベル, 105
- sun-passthrough プロパティ, 129
- Sun Web サーバー, ロードバランサによる変更, 108
- SyslogFacility 属性, 221
- SysLogging 属性, 221
- SysLogLevel 属性, 221
- SyslogPrefix 属性, 221

T

- TakeoverTime 属性, 221

U

- username プロパティ, 223

W

- Web アプリケーション, 分散可能, 143
- web コンテナ, 可用性, 145-148

Web サーバー, 複数のインスタンスと負荷分散, 94

「

「JNDI 名」設定, 224

「すべての障害ですべての接続を閉じる」設定, 223

「グローバルランザクションのサポート」設定, 223

「テーブル名」設定, 223

「データソースが有効」設定, 225

「データソースクラス名」設定, 223

「データベースベンダー」設定, 223

「ランザクション遮断」設定, 223

「プール名」設定, 225

「検証方法」設定, 223

「遮断レベルを保証」設定, 223

「接続検証が必要」設定, 223

「通常プールサイズ」設定, 223

「名前」設定, 223

べ

べき等 URL, 93

ア

アプリケーション

可用性を低下させずにアップグレード, 95

休止, 87

負荷分散のための有効化, 82

アルゴリズム

HTTP 負荷分散, 72

RMI-IIOP フェイルオーバー, 172

ク

クラスタ, 35-36

スタンドアロン, 27

休止, 87

クラスタ (続き)

共有, 27

クラスタ化サーバーインスタンス, 設定, 132

グ

グループ管理サービス, 「GMS」を参照

サ

サーバー, クラスタ, 35-36

サーバーインスタンス

休止, 87

負荷分散のための有効化, 81

シ

シングルサインオン, セッション持続性, 149-150

ス

ステートフルセッション Bean 状態のチェックポイント設定, 143

ステートフルセッション Bean, 150

セッション持続性, 150, 153

スティッキラウンドロビン負荷分散, 72

セ

セッションストア

HTTP セッション用, 146

ステートフルセッション Bean, 152, 153

セッションフェイルオーバー, HTTP および HTTPS, 88-89

セッション持続性

Web モジュール用, 139-140

シングルサインオン, 149-150

ステートフルセッション Bean, 150, 153

セマフォ, 182

タ

ターゲット, ロードバランサ設定, 81

チ

チェックポイント設定, 150
のメソッドの選択, 151, 154

ト

トランザクション
およびセッション持続性, 151, 154

ド

ドメイン管理サーバー
サーバーインスタンスの同期化, 55
ノードエージェントの同期化, 54

ネ

ネットワーク構成の要件, 178-181

ノ

ノードエージェント
JMX リスナー, 68
について, 49
インストール, 53
ドメイン管理サーバーとの同期化, 54
プレースホルダ, 60, 61
ログ, 59
起動, 64
作成, 61
削除, 65, 66
停止, 64
認証レルム, 67
配備, 51
ノードスーパーバイザープロセスと高可用性, 186-187

フ

フェイルオーバー
HTTP について, 72
JMS 接続, 161
RMI-IIOP の要件, 172
Web モジュールセッション用, 139-140
ステートフルセッション Bean 状態, 150

プ

プライマリ端点, RMI-IIOP フェイル
オーバー, 173

ポ

ポート番号, 設定, 133

ラ

ラウンドロビン負荷分散, ステイッキ, 72

ル

ルート Cookie, 81

レ

レルム, ノードエージェントの認証, 67

ロ

ロギング, ノードエージェントログの表示, 59
ログ, ロードバランサ, 102

応

応答タイムアウト, 80

可

可用性

- EJB コンテナレベル, 153-154
- Web モジュール用, 139-140
- ステートフルセッション Bean, 150
- レベル, 143
- 有効化と無効化, 143

管

管理コンソール

- JMS サービスの設定, 158
- JMS ホスト作成のための使用, 161

休

休止

- アプリケーション, 87
- サーバーインスタンスまたはクラスタ, 87

健

健全性検査, 82

再

再読み込み間隔, 80

持

持続性ストア, ステートフルセッション Bean 状態, 150

時

時間の同期, 184

順

順次アップグレード, 95

正

正常でないサーバーインスタンス, 82

接

接続プール

- HADB 用のプロパティ, 223-224
- HADB 用の設定, 223

設

設定, 「名前付き設定」を参照

代

代替端点, RMI-IIOP フェイルオーバー, 173

端

端点, RMI-IIOP フェイルオーバー, 173

中

中央リポジトリ, ノードエージェントの同期化, 54

動

動的再設定, ロードバランサ, 86

認

認証レルム, ノードエージェント, 67

配

配備, 中の可用性の設定, 143

名前付き設定 (続き)

説明, 131

負

負荷分散

- Apache Web Server, 116
- HTTP アルゴリズム, 72
- HTTP, 72
- Microsoft IIS, 127
- RMI-IIOP の要件, 172
- Sun Web サーバー, 108
- べき等 URL, 93
- アプリケーションの休止, 87
- アプリケーションの有効化, 82
- サーバーインスタンスの有効化, 81
- サーバーインスタンスまたはクラスタの休止, 87
- スティッキラウンドロビン, 72
- セッションフェイルオーバー, 88-89
- ロードバランサ設定の作成, 79
- ログメッセージ, 102
- 健全性検査, 82
- 参照の作成, 81
- 設定, 75
- 設定の変更, 85
- 設定ファイルのエクスポート, 84
- 動的再設定, 86
- 複数の Web サーバーインスタンス, 94

分

分散 HTTP セッション, 139-140
分散可能 Web アプリケーション, 143

名

名前付き設定

- default-config, 132
- デフォルト名, 133
- ポート番号, 133
- 共有, 132

