

**IBM WebSphere eXtreme Scale**  
バージョン 8.5

**管理ガイド**  
2012 年 4 月

**IBM**

本書は、WebSphere eXtreme Scale バージョン 8 リリース 5、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM WebSphere eXtreme Scale  
Version 8.5  
Administration Guide  
April 2012

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.5

© Copyright IBM Corporation 2009, 2012.

# 目次

図 . . . . .	vii
表 . . . . .	ix
管理ガイド 情報 . . . . .	xi
<b>第 1 章 始めに . . . . .</b>	<b>1</b>
チュートリアル: WebSphere eXtreme Scale 入門 . . . . .	1
入門チュートリアル・レッスン 1: 構成ファイルを使用したデータ・グリッドの定義 . . . . .	1
入門チュートリアル・レッスン 2: クライアント・アプリケーションの作成 . . . . .	3
入門チュートリアル・レッスン 3: 入門用サンプル・クライアント・アプリケーションの実行 . . . . .	5
入門チュートリアル・レッスン 4: 環境のモニター . . . . .	7
<b>第 2 章 計画 . . . . .</b>	<b>11</b>
計画の概要 . . . . .	11
トポロジーの計画 . . . . .	12
ローカルのメモリー内のキャッシュ . . . . .	12
ピア複製されるローカル・キャッシュ . . . . .	14
組み込みキャッシュ . . . . .	16
分散キャッシュ . . . . .	17
データベース統合: 後書き、インライン、およびサイド・キャッシング . . . . .	19
複数データ・センター・トポロジーの計画 . . . . .	37
他の製品とのインターオペラビリティ . . . . .	52
構成の計画 . . . . .	53
ネットワーク・ポートの計画 . . . . .	53
セキュリティの概要 . . . . .	57
インストールの計画 . . . . .	59
ハードウェアおよびソフトウェア要件 . . . . .	59
Java SE の考慮事項 . . . . .	61
Java EE の考慮事項 . . . . .	62
ディレクトリー規則 . . . . .	63
環境キャパシティーの計画 . . . . .	65
メモリー・サイズ設定および区画数の計算 . . . . .	65
トランザクションの区画ごとの CPU 見積もり . . . . .	68
並列トランザクションの場合の CPU のサイズ設定 . . . . .	68
動的キャッシュのキャパシティー・プランニング . . . . .	69
<b>第 3 章 チュートリアル . . . . .</b>	<b>73</b>
チュートリアル: Java SE セキュリティーの構成 . . . . .	73
Java SE セキュリティー・チュートリアル - ステップ 1 . . . . .	73
Java SE セキュリティー・チュートリアル - ステップ 2 . . . . .	75
Java SE セキュリティー・チュートリアル - ステップ 3 . . . . .	76

Java SE セキュリティー・チュートリアル - ステップ 4 . . . . .	78
Java SE セキュリティー・チュートリアル - ステップ 5 . . . . .	82
Java SE セキュリティー・チュートリアル - ステップ 6 . . . . .	86
チュートリアル: WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合 . . . . .	90
概要: WebSphere Application Server 認証プラグインを使用した、WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合 . . . . .	90
モジュール 1: WebSphere Application Server の準備 . . . . .	92
モジュール 2: WebSphere Application Server 認証プラグインを使用するための WebSphere eXtreme Scale の構成 . . . . .	98
モジュール 3: トランSPORT・セキュリティの構成 . . . . .	105
モジュール 4: WebSphere Application Server での Java 認証・承認サービス (JAAS) 許可の使用 . . . . .	108
モジュール 5: データ・グリッドとマップのモニターのための <code>xscmd</code> ツールの使用 . . . . .	114
チュートリアル: 混合環境で WebSphere eXtreme Scale セキュリティーを外部オーセンティケーターと統合する . . . . .	115
概要: 混合環境のセキュリティ . . . . .	116
モジュール 1: WebSphere Application Server とスタンドアロンとの混合環境の準備 . . . . .	117
モジュール 2: 混合環境での WebSphere eXtreme Scale 認証の構成 . . . . .	123
モジュール 3: トランSPORT・セキュリティの構成 . . . . .	134
モジュール 4: WebSphere Application Server の Java 認証・承認サービス (JAAS) 許可の使用 . . . . .	137
モジュール 5: <code>xscmd</code> ユーティリティーを使用してデータ・グリッドとマップをモニターする . . . . .	141
チュートリアル: OSGi フレームワークでの eXtreme Scale バンドルの実行 . . . . .	143
概要: OSGi フレームワークで eXtreme Scale サーバーとコンテナを開始および構成してプラグインを実行する . . . . .	144
モジュール 1: eXtreme Scale サーバー・バンドルをインストールおよび構成する準備 . . . . .	145
モジュール 2: OSGi フレームワークでの eXtreme Scale バンドルのインストールおよび開始 . . . . .	150
モジュール 3: eXtreme Scale サンプル・クライアントの実行 . . . . .	155
モジュール 4: サンプル・バンドルの照会とアップグレード . . . . .	158

<b>第 4 章 インストール</b> . . . . .	<b>163</b>
インストールの概要 . . . . .	163
インストールの計画 . . . . .	167
インストール・トポロジ . . . . .	167
ハードウェアおよびソフトウェア要件 . . . . .	171
WebSphere eXtreme Scale product offering IDs . . . . .	173
Java SE の考慮事項 . . . . .	173
Java EE の考慮事項 . . . . .	174
ディレクトリー規則 . . . . .	175
WebSphere Application Server と統合された WebSphere eXtreme Scale 用のランタイム・ファ イル . . . . .	177
WebSphere eXtreme Scale スタンドアロン・イン ストール用のランタイム・ファイル . . . . .	180
IBM Installation Manager および WebSphere eXtreme Scale 製品オファリングのインストール . . . . .	182
GUI の使用による IBM Installation Manager の インストール . . . . .	183
コマンド行の使用による IBM Installation Manager のインストール . . . . .	188
応答ファイルの使用による IBM Installation Manager のインストール . . . . .	192
クライアントおよびサーバーの Eclipse Gemini を 持つ Eclipse Equinox OSGi フレームワークのイン ストール . . . . .	198
REST データ・サービスのインストール . . . . .	200
eXtreme Scale バンドルのインストール . . . . .	203
WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale ク ライアントのインストール . . . . .	205
IBM Installation Manager の使用によるフィックス パックのインストール . . . . .	208
GUI の使用によるフィックスパックのインスト ール . . . . .	208
コマンド行の使用によるフィックスパックのイン ストール . . . . .	210
応答ファイルの使用によるフィックスパックのイン ストール . . . . .	213
IBM Installation Manager の使用によるフィックス パックのアンインストール . . . . .	215
GUI の使用によるフィックスパックのアンイン ストール . . . . .	215
コマンド行の使用によるフィックスパックのアン インストール . . . . .	216
応答ファイルを使用したフィックスパックのアン インストール . . . . .	218
IBM Installation Manager の使用による 製品 のア ンインストール . . . . .	220
GUI の使用による the product のアンインスト ール . . . . .	220
コマンド行の使用によるこの製品 のアンインス トール . . . . .	221
応答ファイルを使用した製品 のアンインストー ール . . . . .	222
WebSphere eXtreme Scale のプロファイルの作成お よび拡張 . . . . .	225

プロファイルを作成するグラフィカル・ユーザ ー・インターフェースの使用 . . . . .	226
プロファイルを拡張するグラフィカル・ユーザ ー・インターフェースの使用 . . . . .	227
<b>manageprofiles</b> コマンド . . . . .	227
非 root プロファイル . . . . .	234
インストール後の最初のステップの実行 . . . . .	235
製品インストールのトラブルシューティング . . . . .	236

## 第 5 章 WebSphere eXtreme Scale のアップグレードおよびマイグレーション . . . . . 239

eXtreme Scale サーバーの更新 . . . . .	239
WebSphere eXtreme Scale バージョン 8.5 へのマイ グレーション . . . . .	242
WebSphere Application Server 上の WebSphere eXtreme Scale の更新 . . . . .	243
<b>xsadmin</b> ツールから <b>xscmd</b> ツールへのマイグレー ション . . . . .	245
推奨されないプロパティおよび API . . . . .	248
削除されたプロパティおよび API . . . . .	251

## 第 6 章 構成 . . . . . 253

構成方式 . . . . .	253
運用チェックリスト . . . . .	254
データ・グリッドの構成 . . . . .	256
ローカル・デプロイメントの構成 . . . . .	256
XML 構成の Evictor を使用可能にする . . . . .	257
ロック・ストラテジーの構成 . . . . .	258
JMS を使用したピアツーピア・レプリカ生成の 構成 . . . . .	260
デプロイメント・ポリシーの構成 . . . . .	268
分散デプロイメントの構成 . . . . .	268
ゾーンによる断片配置の制御 . . . . .	271
カタログ・サーバーおよびコンテナ・サーバーの 構成 . . . . .	285
カタログ・サーバーおよびカタログ・サービス・ ドメインの構成 . . . . .	285
コンテナ・サーバーの構成 . . . . .	312
IBM eXtremeMemory および IBM eXtremeIO の構 成 . . . . .	317
複数データ・センター・トポロジの構成 . . . . .	320
ポートの構成 . . . . .	324
スタンドアロン・モードでのポートの構成 . . . . .	324
WebSphere Application Server 環境でのポートの 構成 . . . . .	329
複数のネットワーク・カードを含むサーバー . . . . .	330
トランスポートの構成 . . . . .	330
オブジェクト・リクエスト・ブローカーの構成 . . . . .	330
クライアントの構成 . . . . .	335
XML 構成を使用したクライアントの構成 . . . . .	336
クライアント無効化メカニズムの使用可能化 . . . . .	338
要求再試行タイムアウト値の構成 . . . . .	340
eXtreme Scale 接続ファクトリーの構成 . . . . .	342
eXtreme Scale 接続ファクトリーを使用するため の Eclipse 環境の構成 . . . . .	344

eXtreme Scale に接続するためのアプリケーションの構成	345
キャッシュ統合の構成	345
HTTP セッション・マネージャーの構成	346
WebSphere eXtreme Scale の動的キャッシュ・プロバイダーの構成	373
JPA レベル 2 (L2) キャッシュ・プラグイン	378
Spring キャッシュ・プロバイダーの構成	402
データベース統合の構成	405
JPA ローダーの構成	405
REST データ・サービスの構成	409
REST データ・サービスの使用可能化	411
REST データ・サービス用のアプリケーション・サーバーの構成	420
REST データ・サービス ATOM フィードにアクセスする Web ブラウザーの構成	437
REST データ・サービスでの Java クライアントの使用	440
REST データ・サービスでの Visual Studio 2008 WCF クライアント	442
OSGi 用サーバーの構成	444
OSGi Blueprint での eXtreme Scale プラグインの構成	445
OSGi Blueprint でのサーバーの構成	447
OSGi 構成管理でのサーバーの構成	449
<b>第 7 章 管理</b>	<b>451</b>
スタンドアロン・サーバーの始動と停止	451
スタンドアロン・サーバーの始動	451
スタンドアロン・サーバーの停止	463
WebSphere Application Server 環境でのサーバーの開始と停止	467
組み込みサーバー API を使用したサーバーの開始と停止	468
組み込みサーバー API	471
xscmd ユーティリティによる管理	473
配置の制御	475
ObjectGrid の可用性の管理	477
データ・センター障害の管理	480
データの照会と無効化	483
xscmd ユーティリティを使用した eXtreme Scale 環境情報の取得	485
Eclipse Equinox OSGi フレームワークを使用した eXtreme Scale サーバーの始動	486
OSGi 対応プラグインのインストールと開始	489
xscmd ユーティリティによる OSGi 対応サービスの管理	491
xscmd による eXtreme Scale プラグインの OSGi サービスの更新	494
Managed Beans (MBeans) を使用した管理	497
wsadmin ツールを使用した Managed Beans (MBeans) へのアクセス	498
Managed Bean (MBean) へのプログラマチックなアクセス	498
J2C クライアント接続の管理	503

<b>第 8 章 モニター</b>	<b>505</b>
統計の概説	505
Web コンソールによるモニター	507
Web コンソールの開始とログオン	507
Web コンソールのカタログ・サーバーへの接続	509
Web コンソールでの統計の表示	511
カスタム・レポートによるモニター	518
CSV ファイルによるモニター	518
CSV ファイルの統計定義	519
統計の使用可能化	522
統計モジュール	523
統計 API によるモニター	525
xscmd ユーティリティによるモニター	527
WebSphere Application Server PMI によるモニター	529
PMI の使用可能化	530
PMI 統計の取得	532
PMI モジュール	534
wsadmin ツールを使用した Managed Beans (MBeans) へのアクセス	541
管理 Bean (MBean) を使用したモニター	542
ベンダー・ツールによるモニター	543
IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale のモニター	543
CA Wily Introscope による eXtreme Scale アプリケーションのモニター	550
Hyperic HQ による eXtreme Scale のモニター	553
DB2 内での eXtreme Scale 情報のモニター	556
<b>第 9 章 パフォーマンスのチューニング</b>	<b>559</b>
オペレーティング・システムおよびネットワーク設定のチューニング	559
ORB プロパティ	560
Java 仮想マシンのチューニング	564
フェイルオーバー検出のためのハートビート間隔設定のチューニング	567
WebSphere Real Time を使用したガーベッジ・コレクションのチューニング	569
スタンドアロン環境の WebSphere Real Time	570
WebSphere Application Server における WebSphere Real Time	572
動的キャッシュ・プロバイダーのチューニング	574
<b>第 10 章 セキュリティー</b>	<b>577</b>
アプリケーション・クライアントの認証	577
アプリケーション・クライアントの許可	579
データ・グリッドの認証	583
データ・グリッド・セキュリティー	583
トランスポート層セキュリティーおよび Secure Sockets Layer	585
セキュア・トランスポート・タイプの構成	586
クライアントまたはサーバーの Secure Sockets Layer (SSL) パラメーターの構成	587
Java Management Extensions (JMX) セキュリティー	588
外部プロバイダーとのセキュリティー統合	591
REST データ・サービスの保護	592

WebSphere Application Server とのセキュリティー 統合 . . . . .	596	トレース・オプション . . . . .	620
カタログ・サービス・ドメインのクライアント・ セキュリティーの構成 . . . . .	599	ログおよびトレース・データの分析 . . . . .	622
ローカル・セキュリティーの使用可能化 . . . . .	600	ログ分析の概要 . . . . .	623
セキュア・サーバーの始動と停止 . . . . .	601	ログ分析の実行 . . . . .	624
スタンドアロン環境でのセキュア・サーバーの始 動 . . . . .	601	ログ分析用カスタム・スキャナーの作成 . . . . .	625
WebSphere Application Server でのセキュア・サ ーバーの始動 . . . . .	602	ログ分析のトラブルシューティング . . . . .	627
セキュア・サーバーの停止 . . . . .	603	インストールのトラブルシューティング . . . . .	628
<b>xscmd</b> ユーティリティーのためのセキュリティー・ プロファイルの構成 . . . . .	604	キャッシュ統合のトラブルシューティング . . . . .	629
J2C クライアント接続の保護 . . . . .	605	JPA キャッシュ・プラグインのトラブルシューティ ング . . . . .	630
<b>第 11 章 トラブルシューティング . . . . .</b>	<b>609</b>	IBM eXtremeMemory および IBM eXtremeIO のト ラブルシューティング . . . . .	631
WebSphere eXtreme Scale のトラブルシューティン グおよびサポート . . . . .	609	管理のトラブルシューティング . . . . .	632
問題のトラブルシューティングのための手法 . . . . .	609	複数データ・センター構成のトラブルシューティン グ . . . . .	633
知識ベースの検索 . . . . .	611	ローダーのトラブルシューティング . . . . .	633
フィックスの入手 . . . . .	612	XML 構成のトラブルシューティング . . . . .	635
IBM サポートへの連絡 . . . . .	613	セキュリティーのトラブルシューティング . . . . .	639
IBM との情報の交換 . . . . .	614	IBM Support Assistant for WebSphere eXtreme Scale	640
サポート更新情報のサブスクライブ . . . . .	616	<b>特記事項 . . . . .</b>	<b>643</b>
ロギング可能化 . . . . .	617	<b>商標 . . . . .</b>	<b>645</b>
トレースの収集 . . . . .	618	<b>索引 . . . . .</b>	<b>647</b>





1. ローカルのメモリー内のキャッシュ・シナリオ	13	37. objectGridDeployment.xml ファイル	364
2. JMS によって変更が伝搬されるピア複製キャッシュ	14	38. objectGridStandAlone.xml ファイル	366
3. HA マネージャーによって変更が伝搬されるピア複製キャッシュ	15	39. objectGridDeploymentStandAlone.xml ファイル	367
4. 組み込みキャッシュ	16	40. JPA イントラドメイン・トポロジー	381
5. 分散キャッシュ	18	41. JPA 組み込みトポロジー	382
6. ニア・キャッシュ	18	42. JPA 組み込み区画化トポロジー	383
7. データベース・バッファとしての ObjectGrid	20	43. JPA リモート・トポロジー	385
8. サイド・キャッシュとしての ObjectGrid	21	44. 開始用 (getting started) サンプル・トポロジー	410
9. サイド・キャッシュ	22	45. Microsoft SQL Server Northwind サンプルのスキーマ図	411
10. インライン・キャッシュ	23	46. Customer および Order エンティティのスキーマ図	412
11. リードスルー・キャッシング	24	47. Category および Product エンティティのスキーマ図	413
12. ライトスルー・キャッシング	25	48. Customer および Order エンティティのスキーマ図	415
13. 後書きキャッシング	26	49. eXtreme Scale プラグインを含んでいる OSGi バンドルをインストールおよび開始する Eclipse Equinox プロセス	445
14. 後書きキャッシング	27	50. ObjectGrid インスタンスの可用性状態	478
15. ローダー	29	51. OSGi バンドルにすべての構成およびメタデータを含めるための Eclipse Equinox プロセス	487
16. Loader プラグイン	31	52. OSGi バンドルの外部で構成およびメタデータを指定するための Eclipse Equinox プロセス	488
17. クライアント・ローダー	32	53. CollectPlacementPlan.java	499
18. 定期的リフレッシュ	33	54. CollectContainerStatus.java	501
19. チュートリアル・トポロジー	93	55. CollectPlacementPlan.java	502
20. チュートリアル・トポロジー	119	56. 統計の概説	505
21. 認証フロー	123	57. MBean の概説	507
22. 開発ノード	168	58. ObjectGridModule モジュールの構造	534
23. 2 つのデータ・センターがあるスタンドアロン・トポロジー	169	59. ObjectGridModule モジュール構造の例	535
24. WebSphere Application Server トポロジー例	170	60. mapModule 構造	536
25. 混合トポロジー例	171	61. mapModule モジュール構造の例	536
26. WebSphere eXtreme Scale REST データ・サービスのファイル	202	62. hashIndexModule モジュール構造	538
27. XML により TimeToLive Evictor を使用可能にする	257	63. hashIndexModule モジュール構造の例	538
28. XML による Evictor のプラグ	258	64. agentManagerModule 構造	539
29. ゾーン内のプライマリーとレプリカ	278	65. agentManagerModule 構造の例	540
30. objectGridServer.properties ファイル	308	66. queryModule の構造	541
31. eXtremeMemory とヒープ・ストレージの応答時間の比較	318	67. QueryStats.jpg queryModule 構造の例	541
32. カタログ・サービス・ドメイン間のリンク	322	68. 同じセキュリティー・ドメイン内のサーバーの認証フロー	597
33. ハブおよびスポーク・トポロジー	323		
34. コマンド行の使用例	326		
35. ORB の選択	333		
36. objectGrid.xml ファイル	363		





## 表

1. アービトレーション・アプローチ . . . . .	47	18. configureClientSecurity ステップ引数 . . . . .	292
2. Java SE 5、Java SE 6、および Java SE 7 を必要とするフィーチャー . . . . .	61	19. modifyXSDomain コマンド引数 . . . . .	295
3. WebSphere eXtreme Scale 製品のオフライン ID . . . . .	173	20. modifyEndpoints ステップ引数 . . . . .	296
4. Java SE 5、Java SE 6、および Java SE 7 を必要とするフィーチャー . . . . .	174	21. addEndpoints ステップ引数 . . . . .	298
5. WebSphere eXtreme Scale用のランタイム・ファイル . . . . .	178	22. removeEndpoints ステップ引数 . . . . .	299
6. WebSphere eXtreme Scale クライアント用のランタイム・ファイル . . . . .	179	23. configureClientSecurity ステップ引数 . . . . .	300
7. WebSphere eXtreme Scale フルインストールのランタイム・ファイル . . . . .	180	24. カタログ・サーバー・エンドポイント状況	305
8. WebSphere eXtreme Scale クライアント用のランタイム・ファイル . . . . .	181	25. ハートビート間隔 . . . . .	310
9. xsadmin ユーティリティの引数と、xscmd 同等コマンド . . . . .	245	26. 接続ファクトリーを構成するためのカスタム・プロパティ . . . . .	343
10. 推奨されないプロパティおよび API	248	27. ObjectGrid を使用した SIP セッション管理のためのカスタム・プロパティ . . . . .	357
11. 推奨されないプロパティおよび API	249	28. リポジトリへのアーカイブの追加 . . . . .	427
12. 推奨されないプロパティおよび API	249	29. 新規アプリケーションのインストール	428
13. 推奨されないプロパティおよび API	250	30. リポジトリへのアーカイブの追加 . . . . .	428
14. 削除されたプロパティおよび API . . . . .	252	31. 新規アプリケーションのインストール	429
15. 運用チェックリスト . . . . .	254	32. リポジトリへのアーカイブ . . . . .	431
16. createXSDomain コマンド引数 . . . . .	291	33. インストール値 . . . . .	431
17. defineDomainServers ステップ引数 . . . . .	291	34. ハートビート間隔 . . . . .	567
		35. クライアントおよびサーバーの設定における資格情報認証 . . . . .	578
		36. クライアント・トランスポートおよびサーバー・トランスポートの設定で使用するトランスポート・プロトコル . . . . .	587
		37. エンティティ・アクセス権限 . . . . .	595



---

## 管理ガイド 情報

WebSphere® eXtreme Scale の資料セットには、WebSphere eXtreme Scale 製品の使用、プログラミング、および管理に必要な情報を提供する 3 つのボリュームがあります。

### WebSphere eXtreme Scale ライブラリー

WebSphere eXtreme Scale ライブラリーには、以下の資料が含まれます。

- **製品概要** には、ユース・ケース・シナリオ、およびチュートリアルなど、WebSphere eXtreme Scale 概念の高水準の観点が含まれます。
- 「**インストール・ガイド**」では、WebSphere eXtreme Scale の一般的なトポロジーをインストールする方法について説明しています。
- **管理ガイド** には、アプリケーション・デプロイメント計画の作成方法、容量計画の作成方法、製品のインストールと構成方法、サーバーの始動と停止方法、環境のモニター方法、環境の保護方法など、システム管理者に必要な情報が含まれます。
- **プログラミング・ガイド** には、掲載されている API 情報を使用して WebSphere eXtreme Scale 用のアプリケーションを開発する方法に関する、アプリケーション開発者のための情報が含まれます。

これらの資料をダウンロードするには、WebSphere eXtreme Scale ライブラリー・ページにアクセスしてください。

このライブラリーと同じ情報は、WebSphere eXtreme Scale バージョン 8.5 インフォメーション・センターから入手することもできます。

### オフラインでのブックの使用

WebSphere eXtreme Scale ライブラリー内のすべてのブックには、インフォメーション・センターへのリンクが含まれており、ルート URL は <http://publib.boulder.ibm.com/infocenter/wxsinfo/v8r5> です。これらのリンクを使用して、関連情報に直接アクセスできます。ただし、オフラインで作業していてこれらのリンクのいずれかを見つけた場合は、ライブラリー内の他のブックでそのリンクのタイトルを検索できます。API 資料、用語集、およびメッセージ解説書は、PDF ブックでは用意されていません。

### 本書の対象者

本書は、主にシステム管理者、セキュリティー管理者、およびシステム・オペレーターの方々を対象としています。

### 本書の更新の取得

本書の更新は、WebSphere eXtreme Scale ライブラリー・ページから最新のバージョンをダウンロードすることで取得できます。

## ご意見の送付方法

文書チームにご連絡ください。必要な情報が見つかりましたか？ それは正確で完全な情報でしたか？ 本書に関するご意見は、電子メールで [wasdoc@us.ibm.com](mailto:wasdoc@us.ibm.com) までお寄せください。

---

## 第 1 章 始めに



製品のインストール後に、開始用 (getting started) サンプルを使用して、インストールをテストし、初めて製品を使用できます。

---

### チュートリアル: WebSphere eXtreme Scale 入門

WebSphere eXtreme Scale をスタンドアロン環境にインストールしたら、メモリー内データ・グリッドとしての本製品の機能を簡単に紹介している入門用サンプル・アプリケーションを使用できます。入門用サンプル・アプリケーションは、WebSphere eXtreme Scale のフルインストール (クライアントおよびサーバーのインストール) にのみ含まれます。

#### 学習目標

- 環境の構成に使用する ObjectGrid 記述子 XML ファイルとデプロイメント・ポリシー記述子 XML ファイルについて学習する。
- 構成ファイルを使用してカタログ・サーバーとコンテナ・サーバーを開始する。
- クライアント・アプリケーションの開発方法を学習する。
- クライアント・アプリケーションを実行して、データをデータ・グリッドに挿入する。
- Web コンソールでデータ・グリッドをモニターする。

#### 所要時間

60 分

### 入門チュートリアル・レッスン 1: 構成ファイルを使用したデータ・グリッドの定義

コンテナ・サーバーを始動するために、objectgrid.xml ファイルと deployment.xml ファイルが必要です。

サンプルは、`wxs_install_root/ObjectGrid/gettingstarted/xml` ディレクトリーにある `objectgrid.xml` ファイルと `deployment.xml` ファイルを使用します。これらのファイルが開始コマンドに渡され、コンテナ・サーバーとカタログ・サーバーが開始されます。objectgrid.xml ファイルは ObjectGrid 記述子 XML ファイルです。deployment.xml ファイルは ObjectGrid デプロイメント・ポリシー記述子 XML ファイルです。これらのファイルが一緒になって、分散トポロジーを定義します。

#### ObjectGrid 記述子 XML ファイル

ObjectGrid 記述子 XML ファイルは、アプリケーションによって使用される ObjectGrid の構造を定義するのに使用されます。このファイルには、バックアップ・マップ構成のリストが含まれます。これらのバックアップ・マップはキャッシュ

ユ・データを保管します。以下の例は、objectgrid.xml ファイルのサンプルです。ファイルの最初の数行には、各 ObjectGrid XML ファイルの必須ヘッダーが含まれています。このサンプル・ファイルは、Map1 と Map2 というバックアップ・マップがある Grid ObjectGrid を定義しています。

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="Grid">
      <backingMap name="Map1" />
      <backingMap name="Map2" />
    </objectGrid>
  </objectGrids>

</objectGridConfig>
```

## デプロイメント・ポリシー記述子 XML ファイル

デプロイメント・ポリシー記述子 XML ファイルは、対応する ObjectGrid XML である objectgrid.xml ファイルと対で使用されることを想定しています。以下の例では、deployment.xml ファイルの最初の数行には、各デプロイメント・ポリシー XML ファイルの必須ヘッダーが含まれています。このファイルは、objectgrid.xml ファイル内に定義された Grid ObjectGrid の objectgridDeployment エレメントを定義しています。Grid ObjectGrid 内に定義された Map1 と Map2 の両 BackingMap は、mapSet mapSet に含まれていて、そこでは numberOfPartitions、minSyncReplicas、および maxSyncReplicas 属性が構成されています。

```
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="Grid">
    <mapSet name="mapSet" numberOfPartitions="13" minSyncReplicas="0"
      maxSyncReplicas="1" >
      <map ref="Map1"/>
      <map ref="Map2"/>
    </mapSet>
  </objectgridDeployment>

</deploymentPolicy>
```

mapSet エレメントの numberOfPartitions 属性は、mapSet の区画の数を指定します。これはオプションの属性であり、デフォルトは 1 です。この数は、データ・グリッドに予想される容量に適した値である必要があります。

mapSet の minSyncReplicas 属性は、mapSet 内の各区画の同期レプリカの最小数を指定します。これはオプションの属性であり、デフォルトは 0 です。この同期レプリカの最小数をドメインがサポートできるまでは、プライマリーおよびレプリカは配置されません。minSyncReplicas 値をサポートするには、minSyncReplicas の値よりも 1 つだけ多いコンテナが必要です。同期レプリカの数 minSyncReplicas の値よりも小さくなると、その区画に対しては書き込みトランザクションを行えなくなります。

mapSet の maxSyncReplicas 属性は、mapSet 内の各区画の同期レプリカの最大数を指定します。これはオプションの属性であり、デフォルトは 0 です。ある特定の区

画でこの同期レプリカ数にドメインが達すると、それ以降は、他の同期レプリカがその区画に対して配置されることはありません。まだ `maxSyncReplicas` 値を満たしていない場合には、この `ObjectGrid` をサポートできるコンテナを追加すると、同期レプリカの数を増やすことができます。上のサンプルでは `maxSyncReplicas` は 1 に設定されていますが、これは、ドメインが最大 1 つの同期レプリカを置くことを意味しています。複数のコンテナ・サーバー・インスタンスを開始する場合、それらのコンテナ・サーバー・インスタンスの 1 つに、同期レプリカが 1 つだけ置かれます。

## レッスンのチェックポイント

このレッスンでは、以下を学習しました。

- データを保管するマップを `ObjectGrid` 記述子 XML ファイル内で定義する方法
- デプロイメント記述子 XML ファイルを使用して、データ・グリッドの区画の数とレプリカ数を定義する方法

## 入門チュートリアル・レッスン 2: クライアント・アプリケーションの作成

データ・グリッドのデータを挿入、削除、更新、および取得するには、クライアント・アプリケーションを作成する必要があります。入門用サンプルには、独自のクライアント・アプリケーションの作成方法を学習できるクライアント・アプリケーションが組み込まれています。

`wxs_install_root/ObjectGrid/gettingstarted/client/src/` ディレクトリーにある `Client.java` ファイルは、カタログ・サーバーへの接続方法、`ObjectGrid` インスタンスの取得方法、および `ObjectMap` API の使用法を示したクライアント・プログラムです。`ObjectMap` API は、データをキー値ペアとして保管し、リレーションシップを持たないオブジェクトのキャッシングに適しています。以下のステップでは、`Client.java` ファイルの内容を検討します。

リレーションシップを持つオブジェクトをキャッシュに入れる必要がある場合は、`EntityManager` API を使用してください。

1. `ClientClusterContext` インスタンスを取得することで、カタログ・サービスに接続します。

カタログ・サーバーに接続するには、`ObjectGridManager` API の `connect` メソッドを使用します。使用する `connect` メソッドが必要とするのは、`hostname:port` という形式のカタログ・サーバー・エンドポイントのみです。`hostname:port` 値のリストをコンマで区切って、複数のカタログ・サーバー・エンドポイントを示すことができます。以下のコード・スニペットは、カタログ・サーバーへの接続方法と `ClientClusterContext` インスタンスの取得方法を示します。

```
ClientClusterContext ccc = ObjectGridManagerFactory.getObjectGridManager().connect("localhost:2809", null, null);
```

カタログ・サーバーへの接続が成功すれば、`connect` メソッドは `ClientClusterContext` インスタンスを戻します。この `ClientClusterContext` インスタンスは、`ObjectGridManager` API から `ObjectGrid` を取得するのに必要です。

2. `ObjectGrid` インスタンスを取得します。



ObjectGrid インスタンスを取得するには、ObjectGridManager API の getObjectGrid メソッドを使用します。 getObjectGrid メソッドは、ClientClusterContext インスタンスと、データ・グリッド・インスタンスの名前との両方を必要とします。ClientClusterContext インスタンスは、カタログ・サーバーへの接続中に取得されます。 ObjectGrid インスタンスの名前は、objectgrid.xml ファイルに指定されている Grid です。 以下のコード・スニペットは、ObjectGridManager API の getObjectGrid メソッドを呼び出すことによってデータ・グリッドを取得する方法を示します。

```
ObjectGrid grid = ObjectGridManagerFactory.getObjectGridManager().getObjectGrid(ccc, "Grid");
```

### 3. Session インスタンスを取得します。

取得した ObjectGrid インスタンスから、Session を取得することができます。Session インスタンスは、ObjectMap インスタンスの取得とトランザクション区分の実行のために必要です。以下のコード・スニペットは、ObjectGrid API の getSession メソッドを呼び出すことによって Session インスタンスを取得する方法を示します。

```
Session sess = grid.getSession();
```

### 4. ObjectMap インスタンスを取得します。

Session を取得した後、Session API の getMap メソッドを呼び出すことによって、Session インスタンスから ObjectMap インスタンスを取得することができます。ObjectMap インスタンスを取得するには、マップ名を getMap メソッドにパラメーターとして渡す必要があります。以下のコード・スニペットは、Session API の getMap メソッドを呼び出すことによって ObjectMap を取得する方法を示します。

```
ObjectMap map1 = sess.getMap("Map1");
```

### 5. ObjectMap メソッドを使用します。

ObjectMap インスタンスを取得した後、ObjectMap API を使用できます。ObjectMap インターフェースはトランザクション・マップであり、Session API の begin メソッドおよび commit メソッドを使用したトランザクション区分を必要とすることに注意してください。アプリケーションに明示的なトランザクション区分がない場合、ObjectMap 操作は自動コミット・トランザクションで実行します。

以下のコード・スニペットは、自動コミット・トランザクションでの ObjectMap API の使用方法を示しています。

```
map1.insert(key1, value1);
```

以下のコード・スニペットは、明示的なトランザクション区分がある場合の ObjectMap API の使用方法を示しています。

```
sess.begin();  
map1.insert(key1, value1);  
sess.commit();
```

### 6. オプション: セッションを閉じます。 Session 操作および ObjectMap 操作がすべて完了したら、Session.close() メソッドを使用してセッションを閉じます。このメソッドを実行すると、セッションで使用されていたリソースが返されます。

```
sess.close();
```

結果として、後続の `getSession()` メソッド呼び出しはより早く戻り、ヒープでの `Session` オブジェクトが少なくなります。

## レッスンのチェックポイント

このレッスンでは、データ・グリッドの操作を実行するシンプルなクライアント・アプリケーションを作成する方法について学習しました。

## 入門チュートリアル・レッスン 3: 入門用サンプル・クライアント・アプリケーションの実行

次の手順で最初のデータ・グリッドを開始し、データ・グリッドと対話するクライアントを実行します。この例では、カタログ・サーバー、コンテナ・サーバー、およびクライアントがすべて単一のサーバー上で実行されます。

`env.sh|bat`: このスクリプトは、他のスクリプトから呼び出されて、必要な環境変数を設定します。通常は、このスクリプトを変更する必要はありません。

- `UNIX` `Linux` `./env.sh`
- `Windows` `env.bat`

アプリケーションを実行するには、まずカタログ・サービス・プロセスを開始する必要があります。カタログ・サービスはデータ・グリッドのコントロール・センターです。カタログ・サービスは、コンテナ・サーバーの場所を追跡し、ホスト・コンテナ・サーバーへのデータの配置を制御します。カタログ・サービスが開始したら、データ・グリッドのアプリケーション・データを保管するコンテナ・サーバーを開始できます。データのコピーを複数保管する場合は、複数のコンテナ・サーバーを開始できます。すべてのサーバーが開始したら、クライアント・アプリケーションを実行して、データ・グリッドのデータを挿入、更新、削除、および取得できます。

1. 端末セッションまたはコマンド行ウィンドウを開きます。
2. 次のコマンドを使用して、`gettingstarted` ディレクトリーに移動します。  
`cd wxs_install_root/ObjectGrid/gettingstarted`
3. 次のスクリプトを実行して、ローカル・ホストでカタログ・サービス・プロセスを開始します。

- `UNIX` `Linux` `./runcat.sh`
- `Windows` `runcat.bat`

カタログ・サービス・プロセスは、現行の端末ウィンドウで実行されます。

カタログ・サービスは `startOgServer` コマンドを使用して開始することもできます。 `startOgServer` を `wxs_install_root/ObjectGrid/bin` ディレクトリーから実行します。

- `UNIX` `Linux` `./startOgServer.sh cs0 -catalogServiceEndPoints cs0:localhost:6600:6601 -listenerPort 2809`
- `Windows` `startOgServer.bat cs0 -catalogServiceEndPoints cs0:localhost:6600:6601 -listenerPort 2809`

- 別の端末セッションまたはコマンド行ウィンドウを開き、次のコマンドを実行して、コンテナ・サーバー・インスタンスを開始します。

- UNIX** **Linux** `./runcontainer.sh server0`
- Windows** `runcontainer.bat server0`

コンテナ・サーバーは、現行の端末ウィンドウで実行されます。レプリカ生成をサポートするためにさらに多くのコンテナ・サーバー・インスタンスを開始する場合、別のサーバー名を使用してこのステップを繰り返すことができます。

コンテナ・サーバーは **start0gServer** コマンドを使用して開始することもできます。 **start0gServer** を `wxs_install_root/ObjectGrid/bin` ディレクトリーから実行します。

- UNIX** **Linux** `./start0gServer.sh c0 -catalogServiceEndpoints localhost:2809 -objectgridFile gettingstarted$xml%objectgrid.xml -deploymentPolicyFile gettingstarted/xml/deployment.xml`
- Windows** `start0gServer.bat c0 -catalogServiceEndpoints localhost:2809 -objectgridFile gettingstarted$xml%objectgrid.xml -deploymentPolicyFile gettingstarted$xml%deployment.xml`

- クライアント・コマンドを実行するため、別の端末セッションまたはコマンド行ウィンドウを開きます。

`runclient.sh|bat` スクリプトは、単純な作成、検索、更新、および削除 (CRUD) クライアントを実行し、指定された操作を開始します。

`runclient.sh|bat` スクリプトには次のパラメーターを指定して実行します。

- UNIX** **Linux** `./runclient.sh command value1 value2`
- Windows** `runclient.bat command value1 value2`

`command` には、以下のいずれかのオプションを使用します。

- `value2` を、キー `value1` とともにデータ・グリッドに挿入するには、`i` と指定します。
  - `value1` のキーのオブジェクトを `value2` に更新するには、`u` と指定します。
  - `value1` のキーのオブジェクトを削除するには、`d` と指定します。
  - `value1` のキーのオブジェクトを検索して表示するには、`g` を指定します。
- データ・グリッドにデータを追加します。

**重要:** システムで 2 バイト文字セット (DBCS) を使用している場合、**runClient** スクリプトを使用してデータをデータ・グリッドに挿入するときに、テキストが文字化けまたは破損して表示されることがあります。このテキストは、出力またはキャッシュの中で表示されることがあります。この問題を回避するには、**runClient** スクリプトの中の Java 呼び出しを更新して **-Xargencoding** 引数が含まれるようにし、Unicode 文字セットとして DBCS を指定します。例えば、コマンド `%u runClient i key%u2e81 Hello%2e84World` を使用します。

- UNIX** **Linux** `./runclient.sh i key1 helloWorld`

- `Windows` `runclient.bat i key1 helloWorld`
- b. 値を検索して表示:
- `UNIX` `Linux` `./runclient.sh g key1`
  - `Windows` `runclient.bat g key1`
- c. 値の更新:
- `UNIX` `Linux` `./runclient.sh u key1 goodbyeWorld`
  - `Windows` `runclient.bat u key1 goodbyeWorld`
- d. 値の削除:
- `UNIX` `Linux` `./runclient.sh d key1`
  - `Windows` `runclient.bat d key1`

## レッスンのチェックポイント

このレッスンでは、以下を学習しました。

- カタログ・サーバーおよびコンテナー・サーバーを開始する方法
- サンプル・クライアント・アプリケーションを実行する方法

## 入門チュートリアル・レッスン 4: 環境のモニター

`xscmd` ユーティリティおよび Web コンソールのツールを使用して、データ・グリッド環境をモニターできます。

### Web コンソールによるモニター

Web コンソールでは、現在と過去の統計をグラフにできます。このコンソールには、概要を表示するように事前構成されたグラフがいくつか用意されているほか、使用可能な統計からグラフを作成できるカスタム・レポート・ページもあります。WebSphere eXtreme Scale のモニター・コンソールのグラフ機能を使用して、環境内のデータ・グリッドの全体的なパフォーマンスを表示できます。

インストール・ウィザードを実行するとき、オプション・フィーチャーとして Web コンソールをインストールします。

1. コンソール・サーバーを始動します。 コンソール・サーバーを始動する `startConsoleServer.bat|sh` スクリプトは、インストール済み環境の `wxs_install_root/ObjectGrid/bin` ディレクトリーにあります。
2. コンソールにログオンします。
  - a. Web ブラウザーから、`https://your.console.host:7443` に進み、`your.console.host` を、コンソールをインストールしたサーバーのホスト名に置き換えます。
  - b. コンソールにログオンします。
    - **ユーザー ID:** admin
    - **パスワード:** admin
 コンソールのウェルカム・ページが表示されます。

3. コンソール構成を編集します。「設定」 > 「構成」をクリックして、コンソール構成を確認します。コンソール構成には、以下のような情報があります。
  - WebSphere eXtreme Scale クライアントのトレース・ストリング (\*=`all=disabled` など)
  - 管理者の名前とパスワード
  - 管理者の E メール・アドレス
4. モニター対象のカatalog・サーバーへの接続を確立して維持します。次のステップを繰り返して、それぞれのカatalog・サーバーを構成に追加します。
  - a. 「設定」 > 「eXtreme Scale Catalog・サーバー」をクリックします。
  - b. 新規Catalog・サーバーを追加します。



- 1) 「追加」アイコン ( ) をクリックして、既存のカatalog・サーバーを登録します。
  - 2) ホスト名、リスナー・ポートなどの情報を指定します。ポートの構成およびデフォルトについて詳しくは、53 ページの『ネットワーク・ポートの計画』を参照してください。
  - 3) 「OK」をクリックします。
  - 4) Catalog・サーバーがナビゲーション・ツリーに追加されていることを確認します。
5. Catalog・サービス・ドメインの中に作成するCatalog・サーバーをグループにします。Catalog・サービス・ドメインでセキュリティー設定が構成されているため、Catalog・サーバーでセキュリティーが使用可能にされているときはCatalog・サービス・ドメインを作成する必要があります。
    - a. 「設定」 > 「eXtreme Scale ドメイン」ページをクリックします。
    - b. 新規Catalog・サービス・ドメインを追加します。



- 1) 「追加」アイコン ( ) をクリックして、Catalog・サービス・ドメインを登録します。Catalog・サービス・ドメインの名前を入力します。
- 2) Catalog・サービス・ドメインを作成した後、プロパティを編集できます。Catalog・サービス・ドメインのプロパティは次のとおりです。

**Name** 管理者によって割り当てられた、ドメインのホスト名を示します。

#### カATALOG・サーバー

選択したドメインに属する 1 つ以上のカATALOG・サーバーをリストします。前のステップで作成したカATALOG・サーバーを追加できます。

#### 生成プログラム・クラス

CredentialGenerator インターフェースを実装するクラスの名前を指定します。このクラスを使用して、クライアントの資格情報が取得されます。このフィールドに値を指定すると、

client.properties ファイルにある **credentialGeneratorClass** プロパティが、指定した値でオーバーライドされます。

### 生成プログラム・プロパティ

CredentialGenerator 実装クラスのプロパティを指定します。このプロパティが、setProperties(String) メソッドを使用してオブジェクトに設定されます。credentialGeneratorprops 値は、credentialGeneratorClass プロパティの値が非ヌルの場合にのみ使用されます。このフィールドに値を指定すると、client.properties ファイルにある **credentialGeneratorProps** プロパティが、指定した値でオーバーライドされます。

### eXtreme Scale クライアント・プロパティ・パス

前のステップでセキュリティ・プロパティを含める編集をしたクライアント・プロパティ・ファイルへのパスを指定します。例えば、c:¥ObjectGridProperties¥sampleclient.properties ファイルを示します。コンソールがセキュア接続を使用しないようにする場合は、このフィールドの値を削除できます。パスを設定した後、コンソールは非セキュアな接続を使用します。

- 3) 「OK」をクリックします。
- 4) ドメインがナビゲーション・ツリーに追加されていることを確認します。

既存のカタログ・サービス・ドメインに関する情報を表示するには、「設定」 > 「eXtreme Scale ドメイン」ページのナビゲーション・ツリーの中で、カタログ・サービス・ドメインの名前をクリックします。

6. 接続状況を表示します。「**現行ドメイン**」フィールドは、Web コンソールの中で情報を表示するために現在使用されているカタログ・サービス・ドメインの名前を示します。接続状況が、カタログ・サービス・ドメインの名前の隣に表示されます。
7. データ・グリッドおよびサーバーの統計を表示するか、カスタム・レポートを作成します。

## xscmd ユーティリティによるモニター

1. オプション: クライアント認証が使用可能な場合: コマンド行ウィンドウを開きます。コマンド行で、適切な環境変数を設定します。
  - a. CLIENT\_AUTH\_LIB 環境変数を設定します。
    - **Windows** set CLIENT\_AUTH\_LIB=<path\_to\_security\_JAR\_or\_classes>
    - **UNIX** set CLIENT\_AUTH\_LIB=<path\_to\_security\_JAR\_or\_classes>  
export CLIENT\_AUTH\_LIB
2. wxs\_home/bin ディレクトリーに移動します。

```
cd wxs_home/bin
```
3. 各種コマンドを実行して、環境に関する情報を表示します。
  - Grid データ・グリッドと mapSet マップ・セットのすべてのオンライン・コンテナ・サーバーを表示します。

```
xscmd -c showPlacement -g Grid -ms mapSet
```
  - データ・グリッドのルーティング情報を表示します。

```
xscmd -c routetable -g Grid
```

- データ・グリッド内のマップ・エントリーの数を表示します。

```
xscmd -c showMapSizes -g Grid -ms mapSet
```

## サーバーの停止

クライアント・アプリケーションの使用と入門用サンプル環境のモニターが終了したら、サーバーを停止できます。

- スクリプト・ファイルを使用してサーバーを開始した場合は、<ctrl+c> を使用して、カタログ・サービス・プロセスおよびコンテナ・サーバーをそれぞれのウィンドウで停止します。
- **startOgServer** コマンドを使用してサーバーを開始した場合は、**stopOgServer** コマンドを使用してサーバーを停止します。

コンテナ・サーバーを停止します。

```
- UNIX Linux stopOgServer.sh c0 -catalogServiceEndpoints  
localhost:2809  
- Windows stopOgServer.bat c0 -catalogServiceEndpoints  
localhost:2809
```

カタログ・サーバーを停止します。

```
- UNIX Linux stopOgServer.sh cs1 -catalogServiceEndpoints  
localhost:2809  
- Windows stopOgServer.bat cs1 -catalogServiceEndpoints  
localhost:2809
```

## レッスンのチェックポイント

このレッスンでは、以下を学習しました。

- Web コンソールを開始して、カタログ・サーバーに接続する方法
- データ・グリッドおよびサーバーの統計をモニターする方法
- サーバーを停止する方法



---

## 第 2 章 計画



WebSphere eXtreme Scale をインストールして、データ・グリッド・アプリケーションをデプロイする前に、キャッシング・トポロジーを決定し、キャパシティー・プランニングを実行し、ハードウェア要件およびソフトウェア要件、ネットワークとチューニングの設定などを検討する必要があります。運用チェックリストを使用して、アプリケーションをデプロイできる環境になっているかどうかを確認することもできます。

使用する WebSphere eXtreme Scale アプリケーションを設計する際に利用できるベスト・プラクティスについては、developerWorks®: ハイパフォーマンスで高い回復力を持つ WebSphere eXtreme Scale アプリケーションを作成するための原則とベスト・プラクティス (Principles and best practices for building high performing and highly resilient WebSphere eXtreme Scale application) の記事を参照してください。

---

### 計画の概要

WebSphere eXtreme Scale を実稼働環境で使用する前に、デプロイメントを最適化するための以下の問題を検討してください。

#### キャッシング・トポロジーに関する考慮事項

キャッシュ・トポロジーのタイプごとに利点と欠点があります。実装するキャッシング・トポロジーは、環境とアプリケーションの要件によって異なります。各種キャッシング・トポロジーについて詳しくは、12 ページの『トポロジーの計画』を参照してください。

#### データ・キャパシティーに関する考慮事項

次のリストに、検討項目を示します。

- **システムおよびプロセッサの数:** 環境内には物理マシンとプロセッサがいくつ必要ですか?
- **サーバーの数:** いくつの eXtreme Scale サーバーが eXtreme Scale マップをホストしますか?
- **区画の数:** マップ内に保管されるデータの量は、必要な区画の数を決定する 1 つの要因です。
- **レプリカの数:** ドメイン内の各プライマリーに対してレプリカがいくつ必要ですか?
- **同期または非同期レプリカ生成:** データがきわめて重要であるため、同期レプリカ生成が必要ですか? それとも、パフォーマンスに高い優先度を置くため、非同期レプリカ生成が適切な選択ですか?
- **ヒープ・サイズ:** 各サーバーには、どれほどのデータが保管されますか?

上記の個々の考慮事項について詳しくは、65 ページの『環境キャパシティーの計画』を参照してください。

## インストールの注意点

WebSphere eXtreme Scale は、スタンドアロン環境にインストールできます。あるいは、そのインストールを WebSphere Application Server と統合できます。将来、サーバーをシームレスにアップグレードできるようにするには、そのように環境を計画する必要があります。最良のパフォーマンスのために、カタログ・サーバーは、コンテナ・サーバーと異なるマシンで実行してください。カタログ・サーバーとコンテナ・サーバーを同じマシン上で実行しなければならない場合は、カタログ・サーバーとコンテナ・サーバーで別個の WebSphere eXtreme Scale のインストールを使用してください。2 つのインストールを使用することにより、最初にカタログ・サーバーを実行しているインストールをアップグレードできます。239 ページの『eXtreme Scale サーバーの更新』を参照してください。

---

## トポロジーの計画

WebSphere eXtreme Scale を使用して、アーキテクチャーはローカルのメモリー内でのデータ・キャッシング、または分散クライアント/サーバーでのデータ・キャッシングを使用できます。アーキテクチャーは、データベースとさまざまな関係を持つことができます。複数のデータ・センターに及ぶトポロジーを構成することもできます。

WebSphere eXtreme Scale を作動させるには、最低限の追加インフラストラクチャーが必要です。インフラストラクチャーは、サーバー上で Java Platform, Enterprise Edition アプリケーションをインストール、開始、および停止するためのスクリプトで構成されます。キャッシュ・データはコンテナ・サーバー内に保管され、クライアントはリモート側でサーバーに接続します。

### メモリー内の環境

メモリー内のローカル環境にデプロイすると、WebSphere eXtreme Scale は、単一 Java 仮想マシン内で稼働するため、複製されません。ローカル環境を構成するには、ObjectGrid XML ファイルまたは ObjectGrid API を使用できます。

### 分散環境

分散環境にデプロイすると、WebSphere eXtreme Scale は Java 仮想マシンのセット内で稼働し、パフォーマンス、可用性、およびスケーラビリティが向上します。この構成では、データのレプリカ生成および区画化の使用が可能です。また、既存の eXtreme Scale サーバーを再始動せずに、別のサーバーを追加することもできます。ローカル環境の場合と同じように、分散環境でも ObjectGrid XML ファイル、または同等のプログラマチック構成が必要です。構成詳細を持つデプロイメント・ポリシー XML ファイルも提供する必要があります。

単純なデプロイメントを作成することも、数千ものサーバーが必要になる大規模なテラバイト・サイズのデプロイメントを作成することもできます。

## ローカルのメモリー内のキャッシュ

最も単純なケースでは、WebSphere eXtreme Scale は、ローカルの (非分散型の) メモリー内のデータ・グリッド・キャッシュとして使用できます。ローカルのケースは、特に複数のスレッドにより一時データにアクセスして変更する必要がある、高

い並行性を持つアプリケーションで有効になります。 ローカル・データ・グリッドに保持されるデータは、索引を付け、照会を使用して検索することができます。照会は、大規模なメモリー内データ・セットを処理するのに役立ちます。Java 仮想マシン (JVM)で提供されるサポートは、すぐに使用する準備ができているものの、データ構造に制限があります。

WebSphere eXtreme Scale でのローカルのメモリー内キャッシュ・トポロジーは、単一 Java 仮想マシン内で、一時データへの整合したトランザクション・アクセスを可能にするために使用されます。

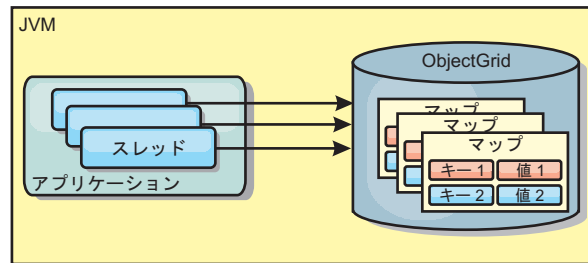


図1. ローカルのメモリー内のキャッシュ・シナリオ

## 利点

- セットアップが簡単: ObjectGrid は、プログラマチックに作成することも、ObjectGrid デプロイメント記述子 XML ファイルまたは Spring などのその他のフレームワークを使用して宣言的に作成することもできます。
- 高速: 各 BackingMap は、最適のメモリー使用効率および並行性が得られるように独立して調整できます。
- 扱うデータ・セットが小さい単一 Java 仮想マシン・トポロジー、また頻繁にアクセスされるデータのキャッシングに最適。
- トランザクション型。BackingMap 更新は、単一の作業単位にまとめることができ、Java Transaction Architecture (JTA) トランザクションなどの 2 フェーズ・トランザクションの最終参加者として統合することができます。

## 欠点

- フォールト・トレラントでない。
- データは複製されない。メモリー内キャッシュは読み取り専用参照データに最適。
- スケーラブルでない。データベースが必要とするメモリーの量が Java 仮想マシンを圧倒するおそれがある。
- Java 仮想マシンを追加するときに、次のような問題が発生する。
  - データを簡単には区画化できない
  - Java 仮想マシン間で状態を手動で複製しなければならない。そうしないと、各キャッシュ・インスタンスが同一データの別バージョンを保持するようになります
  - 無効化にかかるコストが高い。
  - 各キャッシュは個別にウォームアップが必要になる。ウォームアップは、有効なデータがキャッシュに設定されるようにデータをロードする期間です。

## 使用する場合

ローカルのメモリー内キャッシュのデプロイメント・トポロジーは、キャッシュに入れるデータ量が小さく (1 つの Java 仮想マシンに収まる場合)、比較的安定している場合に限って使用するようにしてください。このアプローチの場合、不整合データの存在を許容する必要があります。Evictor を使用して、最も使用頻度が高いデータまたは最近使用されたデータをキャッシュに保持するようにすると、キャッシュ・サイズを小さく維持し、データの関連性を高くすることができます。

## ピア複製されるローカル・キャッシュ

独立したキャッシュ・インスタンスを持つプロセスが複数ある場合は、確実にキャッシュが同期されるようにする必要があります。キャッシュ・インスタンスが確実に同期されるようにするには、Java Message Service (JMS) を使用して、ピア複製されるキャッシュを有効にします。

WebSphere eXtreme Scale には、ピア ObjectGrid インスタンス間にトランザクション変更を自動的に伝搬する 2 つのプラグインがあります。

JMSObjectGridEventListener プラグインは、JMS を使用して、eXtreme Scale 変更を自動的に伝搬します。

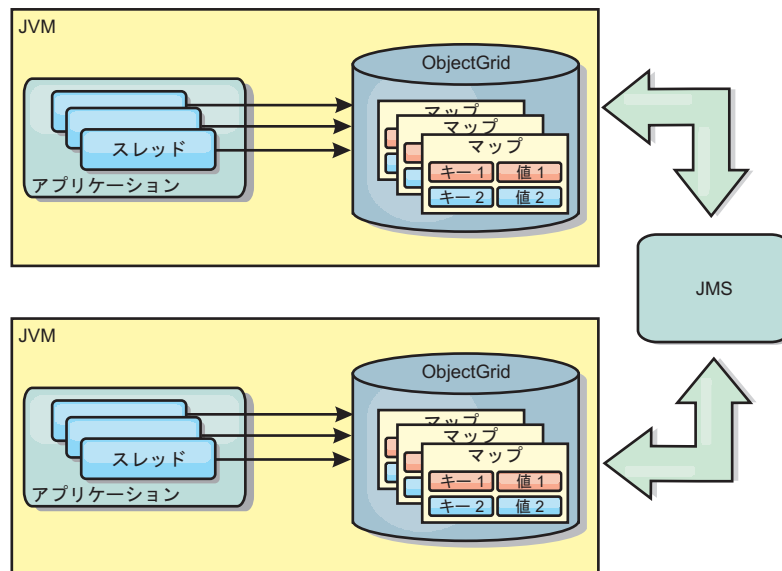


図2. JMS によって変更が伝搬されるピア複製キャッシュ

WebSphere Application Server 環境を実行している場合は、TranPropListener プラグインも使用可能です。TranPropListener プラグインは、高可用性 (HA) マネージャーを使用して、各ピア・キャッシュ・インスタンスに変更を伝搬します。

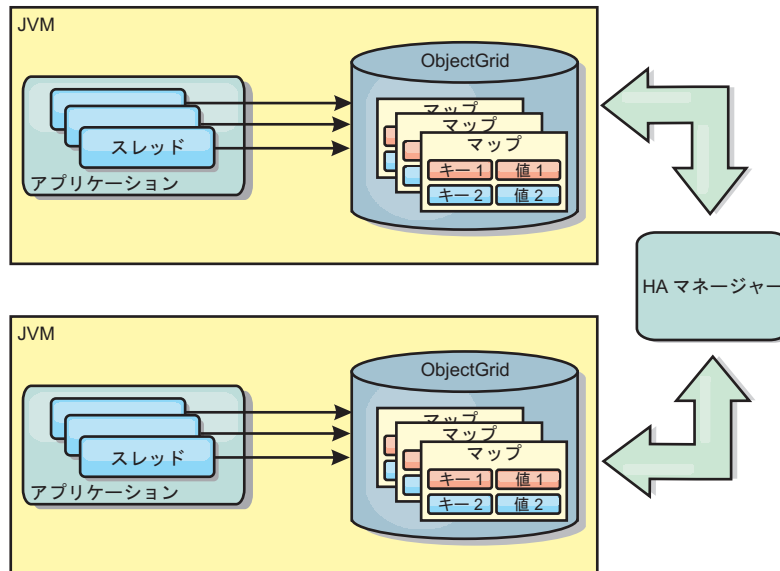


図 3. HA マネージャーによって変更が伝搬されるピア複製キャッシュ

## 利点

- より頻繁にデータが更新されるため、データが有効な場合が増えます。
- TranPropListener プラグインを使用すると、ローカル環境と同様、 eXtreme Scale デプロイメント記述子 XML ファイルや他のフレームワーク (Spring など) を使用して、eXtreme Scale をプログラマチックまたは宣言的に作成できます。HA マネージャーとの統合は自動的に行われます。
- 最適のメモリー使用効率および並行性が得られるように、各 BackingMap を独立して調整できます。
- BackingMap 更新は、単一の作業単位にまとめることができ、Java Transaction Architecture (JTA) トランザクションなどの 2 フェーズ・トランザクションの最終参加者として統合することができます。
- 十分小さなデータ・セットの少数 JVM トポロジー、または頻繁にアクセスされるデータのキャッシングに最適です。
- eXtreme Scale に対する変更は、すべてのピア eXtreme Scale インスタンスに複製されます。変更は、永続サブスクリプションが使用されている限り、整合性が保たれます。

## 欠点

- JMSObjectGridEventListener の構成および保守は、複雑になる場合があります。eXtreme Scale は、eXtreme Scale デプロイメント記述子 XML ファイルまたは Spring などのその他のフレームワークを使用して、プログラマチックまたは宣言的に作成できます。
- スケーラブルではありません。データベースが必要とするメモリー量が、JVM の負担になる場合があります。
- Java 仮想マシンを追加する場合に不適切な機能:
  - データを簡単には区画化できない
  - 無効化にコストがかかります。
  - 各キャッシュは個別にウォームアップが必要になります。

## 使用する場合

デプロイメント・トポロジーは、キャッシュに入れるデータ量が小さく、1つのJVMに収まり、かつ比較的安定している場合にのみ使用します。

## 組み込みキャッシュ

WebSphere eXtreme Scale グリッドは、組み込み eXtreme Scale サーバーとして既存のプロセス内で実行することも、外部プロセスとして管理することもできます。

組み込みグリッドは、WebSphere Application Server などのアプリケーション・サーバー内で実行する場合に便利です。組み込まれていない eXtreme Scale サーバーは、コマンド行スクリプトを使用し、Java プロセスで実行することによって開始できます。

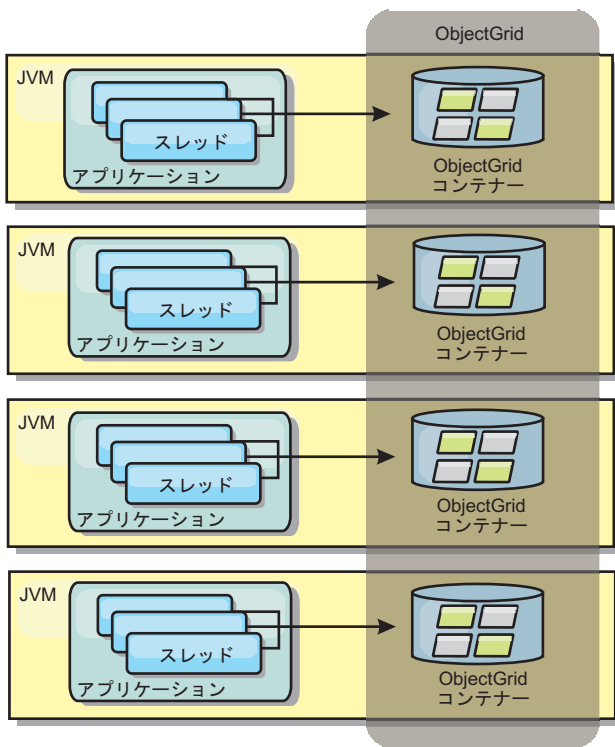


図 4. 組み込みキャッシュ

### 利点

- 管理するプロセスが減るため、管理が簡単になります。
- グリッドがクライアント・アプリケーションのクラス・ローダーを使用しているため、アプリケーションのデプロイメントが簡単です。
- 区画化と高可用性をサポートします。

### 欠点

- すべてのデータがプロセス内に連結されるため、クライアント・プロセスのメモリー占有スペースが増えます。
- クライアント要求にサービスを提供するための CPU 使用率が高くなります。

- クライアントがサーバーと同じアプリケーション Java アーカイブ・ファイルを使用しているため、アプリケーション・アップグレードの処理がさらに難しくなります。
- 柔軟性が低くなります。クライアントとグリッド・サーバーは、同じレートで拡張することができません。サーバーを外部で定義すると、プロセス数の管理の柔軟性が増します。

### 使用する場合

組み込みグリッドは、クライアント・プロセスにグリッド・データおよび潜在的なフェイルオーバー・データ用の空きメモリーが豊富にある場合に使用します。

詳しくは、管理ガイドのクライアント無効化メカニズムの使用可能化に関するトピックを参照してください。

## 分散キャッシュ

WebSphere eXtreme Scale は、共有キャッシュとして使用されることが最も多く、これまで使用されていたような従来のデータベースに代わり、データへのトランザクション・アクセスを複数のコンポーネントに提供します。共有キャッシュにより、データベースを構成する必要がなくなります。

### キャッシュのコヒーレンス

すべてのクライアントがキャッシュ内の同じデータを見るので、キャッシュはコヒーレントです。各データはキャッシュ内の 1 つのサーバーのみに保管されるため、さまざまなバージョンのデータを保管することになりかねない、レコードの無駄なコピーが防止されます。コヒーレントなキャッシュは、より多くのサーバーがデータ・グリッドに追加されるにつれて、より多くのデータを保持することができ、グリッドのサイズが増えるにつれて直線的に増加します。クライアントはこのデータ・グリッドからのデータに、リモート・プロシージャ・コールを使用してアクセスするので、このキャッシュはリモート・キャッシュまたは、ファール・キャッシュとも呼ばれます。データの区画化により、各プロセスは、全データ・セットの中から固有のサブセットを保持します。データ・グリッドが大きいほどより多くのデータを保持でき、そのデータに対するより多くの要求にサービスを提供できます。コヒーレントであることによって、失効データが存在しないため、データ・グリッドの周囲で無効化データをプッシュする必要がなくなります。コヒーレント・キャッシュは、各データの最新コピーのみを保持します。

WebSphere Application Server 環境を実行している場合は、TranPropListener プラグインも使用可能です。TranPropListener プラグインは、WebSphere Application Server 高可用性コンポーネント (HA マネージャー) を使用して、変更を各ピア ObjectGrid キャッシュ・インスタンスに伝搬します。



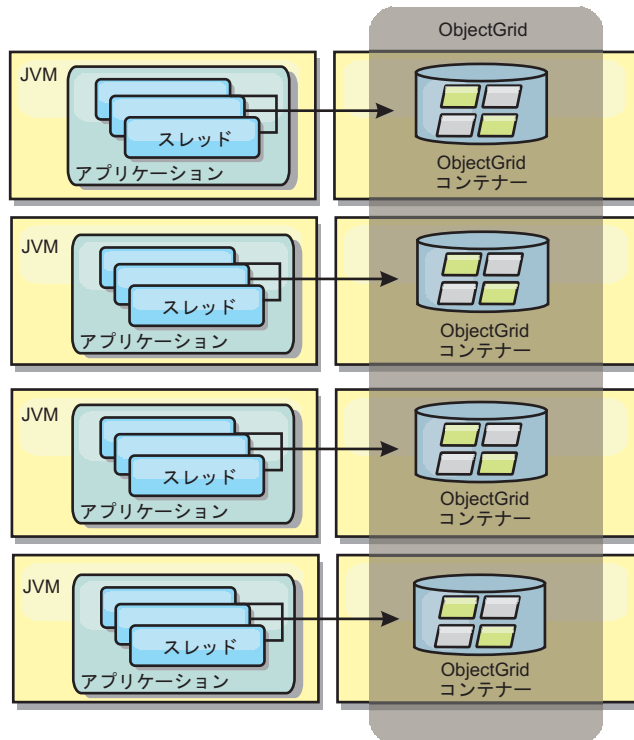


図5. 分散キャッシュ

## ニア・キャッシュ

クライアントは、eXtreme Scale が分散トポロジーで使用されている場合、オプションでローカルのインライン・キャッシュを持つことができます。オプションのこのキャッシュはニア・キャッシュと呼ばれます。これは、各クライアントにある独立した ObjectGrid であり、リモート用のキャッシュ (サーバー・サイド・キャッシュ) として機能します。ニア・キャッシュは、ロックがオプティミスティックまたはロックなしに構成されている場合、デフォルトで使用可能にされており、ロックがペシミスティックに構成されている場合は使用することができません。

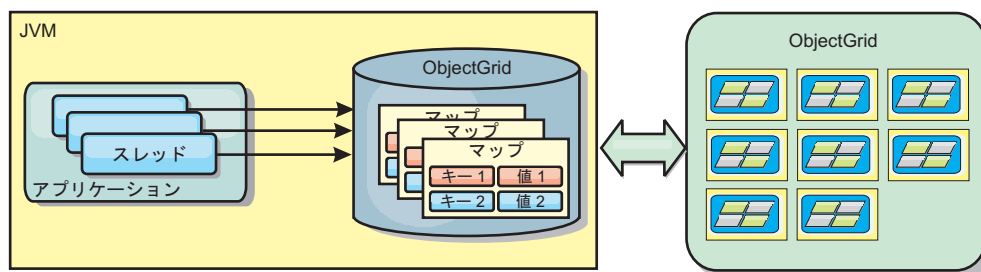


図6. ニア・キャッシュ

ニア・キャッシュは、リモート側で eXtreme Scale サーバーに保管されているキャッシュ・データ・セット全体のサブセットへのメモリー内アクセスを可能にするため、非常に高速です。ニア・キャッシュは区画化されず、任意のリモート eXtreme Scale 区画からのデータを含みます。WebSphere eXtreme Scale は、以下のように、3 つまでのキャッシュ層を持つことができます。

1. トランザクション層キャッシュには、単一トランザクションのすべての変更が含まれます。トランザクション・キャッシュは、トランザクションがコミットされるまで、データの作業用コピーを保持します。クライアント・トランザクションが ObjectMap のデータを要求すると、最初にトランザクションがチェックされます。
2. クライアント層のニア・キャッシュは、サーバー層のデータのサブセットを保持します。トランザクション層にデータがない場合、データはクライアント層にあればクライアント層から取り出され、トランザクション・キャッシュに挿入されます。
3. サーバー層のデータ・グリッドには大半のデータが含まれ、すべてのクライアント間で共有されます。サーバー層は区画に分割できるので、大量のデータをキャッシュに入れることができます。クライアントのニア・キャッシュにデータが存在しないと、サーバー層からデータがフェッチされ、クライアント・キャッシュに挿入されます。サーバー層は、Loader プラグインを保持することもできます。グリッドに要求されたデータがない場合、Loader が呼び出され、結果のデータがバックエンドのデータ・ストアからグリッドに挿入されます。

ニア・キャッシュを使用不可にするには、クライアント・オーバーライド eXtreme Scale 記述子構成で numberOfBuckets 属性を0 に設定します。eXtreme Scale のロック・ストラテジーについて詳しくは、マップ・エントリーのロックに関するトピックを参照してください。ニア・キャッシュは、クライアント・オーバーライド eXtreme Scale 記述子構成を使用して、別の除去ポリシーや異なるプラグインを使用するように構成することもできます。

#### 利点

- データへのアクセスがすべてローカルで行われるため、応答時間が速くなります。ニア・キャッシュ内でデータを探すことで、まず、サーバーのグリッドにいく手間が省け、リモート・データでさえもローカルでアクセス可能になります。

#### 欠点

- 各層のニア・キャッシュはデータ・グリッド内の現行データと同期していない場合があるため、失効データの期間が長くなります。
- メモリー不足を回避するため、エビクターに頼り、データを無効化する必要があります。

#### 使用する場合

応答時間が重要で、失効したデータは許容できる場合に使用します。

## データベース統合: 後書き、インライン、およびサイド・キャッシング

WebSphere eXtreme Scale が使用される目的は、従来のデータベースをその背後に置くことで、通常はデータベースにプッシュされる読み取りアクティビティをなくすることです。コヒーレント・キャッシュは、オブジェクト関連マッパーを直接または間接に使用することにより、アプリケーションで使用できます。コヒーレント・キャッシュは、データベースまたは読み取りからの下流工程の負荷を軽減します。シナリオがもう少し複雑で、一部のデータのみが従来のパーシスタンス保証を必要

とするデータ・セットへのトランザクション・アクセスなどの場合は、フィルター操作を使用して書き込みトランザクションの負荷を軽減します。

WebSphere eXtreme Scale は、高度にフレキシブルなメモリー内のデータベース処理スペースとして機能するように構成できます。ただし、WebSphere eXtreme Scale は、オブジェクト・リレーショナル・マッパー (ORM) ではありません。データ・グリッドに含まれているデータがどこから取得されたのかを認識しません。アプリケーションまたは ORM は、データを eXtreme Scale サーバーに配置できます。データの発生元であるデータベースとの一貫性を保つのは、データのソースの責任です。これは、データベースから取り出されたデータを eXtreme Scale は自動的に無効化できないことを意味します。アプリケーションまたはマッパーは、この機能を提供して、eXtreme Scale に保管されているデータを管理する必要があります。

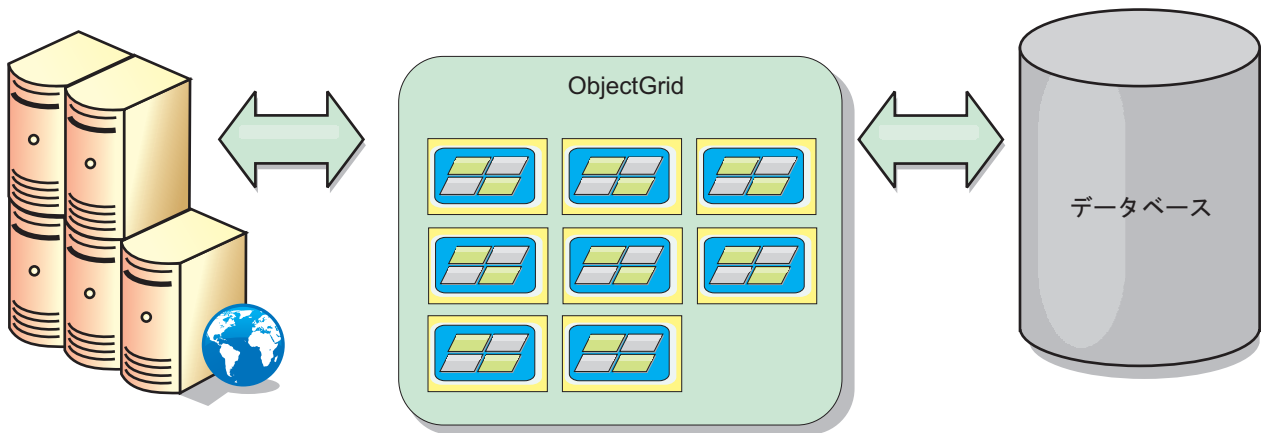


図7. データベース・バッファとしての ObjectGrid

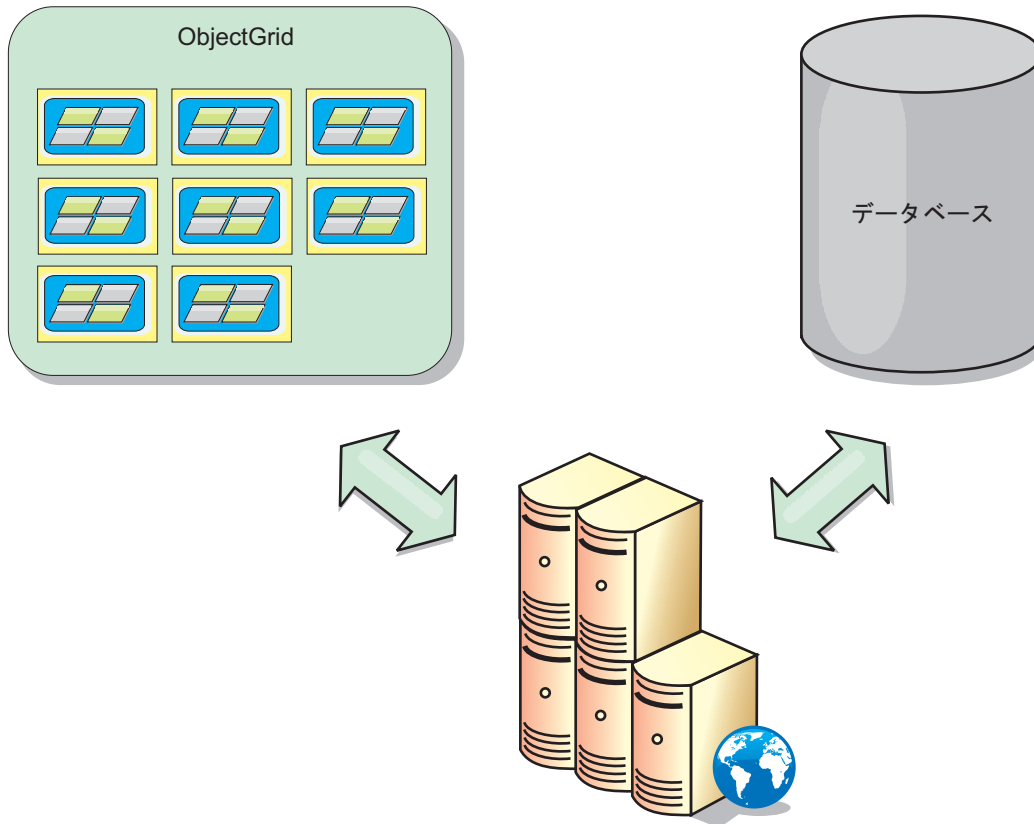


図8. サイド・キャッシュとしての ObjectGrid

### スパース・キャッシュおよび完全キャッシュ

WebSphere eXtreme Scale は、スパース・キャッシュまたは完全キャッシュとして使用できます。完全キャッシュがデータすべてを保持する一方で、スパース・キャッシュはデータ全体のサブセットしか保持しません。必要時には、データをゆっくりと取り込むことができます。通常、スパース・キャッシュは、データが部分的にしか使用可能でないため、キーを使用して (索引や照会を使用せず) アクセスされます。

#### スパース・キャッシュ

キーがスパース・キャッシュに存在しない場合、またはデータが使用できず、キャッシュ・ミスが発生している場合は、次の層が呼び出されます。データは、例えば、データベースからフェッチされ、データ・グリッド・キャッシュ層に挿入されます。照会または索引を使用する場合、現在ロードされている値のみがアクセスされ、要求は他の層に転送されません。

#### 完全キャッシュ

完全キャッシュには必要なすべてのデータが含まれ、索引または照会により非キー属性を使用してアクセスできます。データベースから完全キャッシュにデータがプリロードされた後、アプリケーションはデータへのアクセスを試みます。データがロードされた後は、完全キャッシュをデータベースの代わりとして使用できます。

すべてのデータがあるので、照会および索引を使用して、データの検出と集約を行うことができます。

## サイド・キャッシュ

WebSphere eXtreme Scale をサイド・キャッシュとして使用する場合は、データ・グリッドと一緒にバックエンドが使用されます。

## サイド・キャッシュ

アプリケーションのデータ・アクセス層のサイド・キャッシュとしてこの製品を構成できます。このシナリオの場合、WebSphere eXtreme Scale は、通常であればバックエンド・データベースから取得されるオブジェクトを一時的に保管するために使用されます。アプリケーションは、データがデータ・グリッドに含まれているかどうかチェックします。データがデータ・グリッドにあった場合、そのデータが呼び出し元に返されます。データがない場合、データがバックエンド・データベースから取得されます。そして、次の要求がキャッシュ・コピーを使用できるように、データがデータ・グリッドに挿入されます。次の図は、OpenJPA や Hibernate などの任意のデータ・アクセス層で WebSphere eXtreme Scale をサイド・キャッシュとして使用方法を示しています。

## Hibernate および OpenJPA 向けサイド・キャッシュ・プラグイン

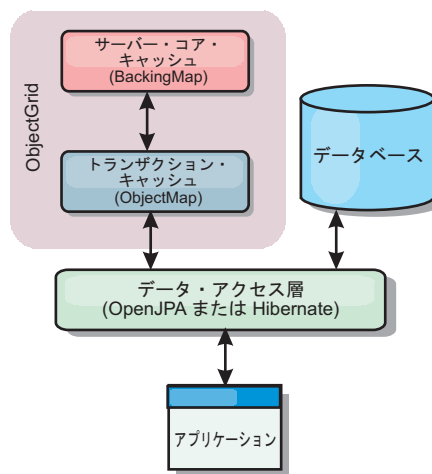


図9. サイド・キャッシュ

WebSphere eXtreme Scale には、この製品を自動サイド・キャッシュとして使用できるようにする、OpenJPA 用と Hibernate 用のどちらのキャッシュ・プラグインも組み込まれています。WebSphere eXtreme Scale をキャッシュ・プロバイダーとして使用すると、データの読み取りおよび照会時のパフォーマンスが高まり、データベースへの負荷が軽減されます。WebSphere eXtreme Scale ではキャッシュが自動的にすべてのプロセス間で複製されるので、組み込みキャッシュ実装をしのぐ利点があります。あるクライアントが値をキャッシュに入れると、他のすべてのクライアントがキャッシュに入れられた値を使用できるようになります。

## インライン・キャッシュ

インライン・キャッシングは、データベース・バックエンドに構成することも、データベースのサイド・キャッシュとして構成することもできます。インライン・キ

キャッシングは、データと対話するための基本手段として eXtreme Scale を使用します。eXtreme Scale がインライン・キャッシュとして使用される場合、アプリケーションは、Loader プラグインを使用してバックエンドと対話します。

## インライン・キャッシュ

インライン・キャッシュとして使用される場合、WebSphere eXtreme Scale は Loader プラグインを使用してバックエンドと対話します。このシナリオでは、アプリケーションが直接 eXtreme Scale API にアクセスできるため、データ・アクセスが単純化されます。キャッシュ内のデータとバックエンドのデータが確実に同期されるようにするための数種類のキャッシング・シナリオが、eXtreme Scale においてポートされています。次の図は、インライン・キャッシュがアプリケーションおよびバックエンドと対話する方法を示しています。

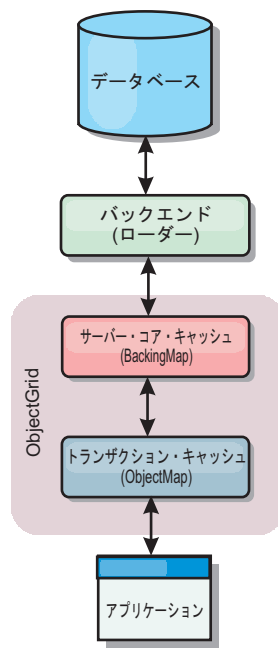


図 10. インライン・キャッシュ

インライン・キャッシング・オプションにより、アプリケーションが eXtreme Scale API に直接アクセスできるようになるため、データ・アクセスが単純化されます。WebSphere eXtreme Scale は、以下のような複数のインライン・キャッシング・シナリオをサポートします。

- リードスルー
- ライトスルー
- 後書き

### リードスルー・キャッシングのシナリオ

リードスルー・キャッシュは、データ・エントリーの要求時にキーによるそのロードが暫時的に行われるスパス・キャッシュです。これが行われる場合、呼び出し元は、エントリーがどのように取り込まれるかを知る必要はありません。データが eXtreme Scale キャッシュに見つからない場合、eXtreme Scale は、その欠落データを Loader プラグインから取得します。このプラグインは、バックエンド・データ

ベースからデータをロードして、そのデータをキャッシュに挿入します。同じデータ・キーに対する後続の要求は、削除、無効化、または除去されるまでキャッシュに存在します。

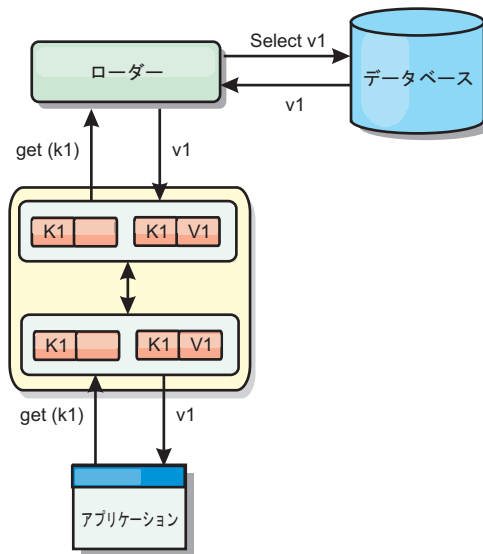


図 11. リードスルー・キャッシング

### ライトスルー・キャッシングのシナリオ

ライトスルー・キャッシュでは、キャッシュへの書き込みが行われるたびに、ローダーを使用してデータベースへの書き込みが同期的に行われます。このメソッドでは、バックエンドとの整合性はありますが、データベース操作が同期されるため、書き込みパフォーマンスは低下します。キャッシュとデータベースがともに更新されるため、同じデータに対する後続の読み取りはキャッシュに残り、データベース呼び出しが回避されます。ライトスルー・キャッシュは、多くの場合、リードスルー・キャッシュと一緒に使用されます。



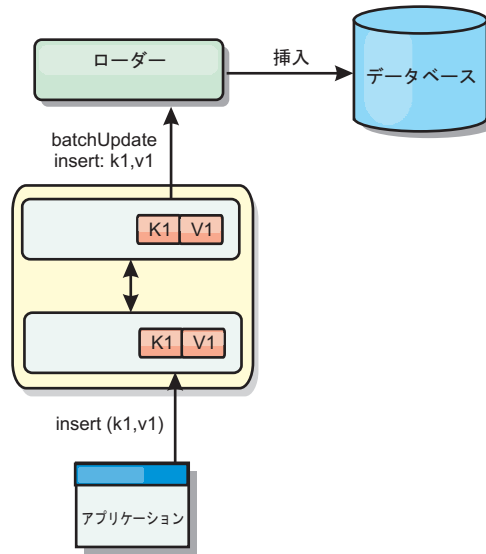


図 12. ライトスルー・キャッシング

### 後書きキャッシングのシナリオ

変更を非同期的に書き込むことにより、データベースの同期性が改善されます。後書きキャッシュまたはライト・バック・キャッシュとも呼ばれます。通常はローダーに対して同期的に書き込まれる変更は、eXtreme Scale 内でバッファ化されてから、バックグラウンド・スレッドを使用してデータベースに書き込まれます。データベース操作をクライアント・トランザクションから除去し、データベース書き込みを圧縮できるため、書き込みパフォーマンスが著しく向上します。

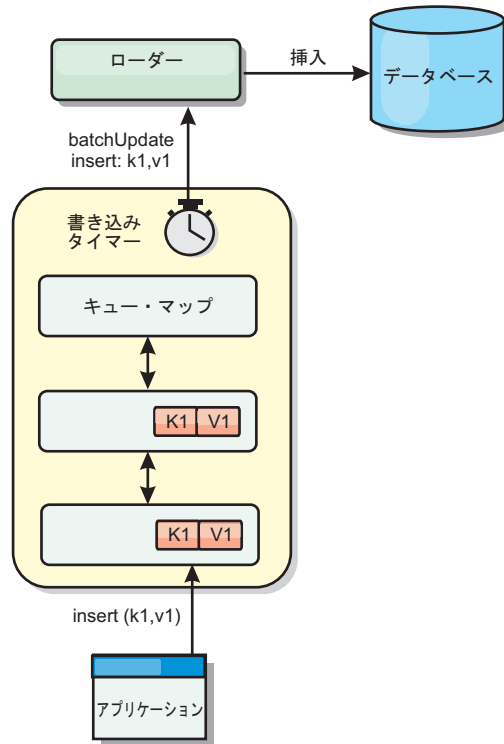


図 13. 後書きキャッシング

## 後書きキャッシング

後書きキャッシングを使用して、バックエンドとして使用しているデータベースを更新する際に発生するオーバーヘッドを減らすことができます。

### 後書きキャッシングの概要

後書きキャッシングでは、Loader プラグインの更新が非同期にキューに入れられます。eXtreme Scale トランザクションをデータベース・トランザクションから分離することにより、マップの更新、挿入、および除去の、パフォーマンスを改善できます。非同期更新は、時間ベースの遅延 (例えば 5 分) またはエントリー・ベースの遅延 (例えば 1000 エントリー) 後に実行されます。

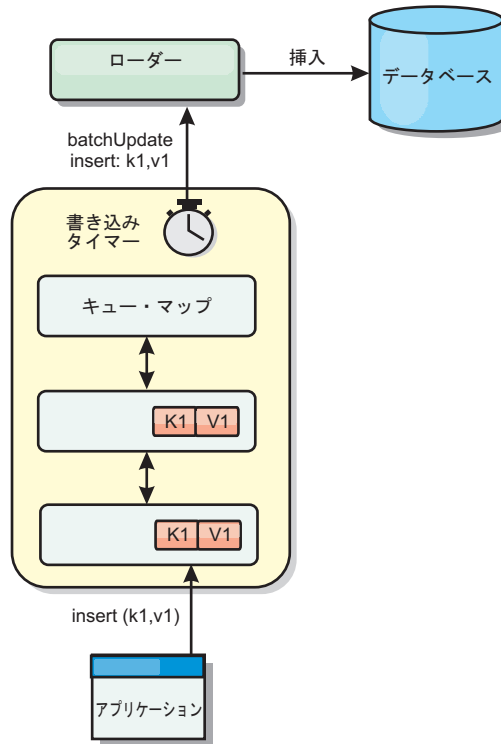


図 14. 後書きキャッシング

BackingMap の後書き構成により、ローダーとマップとの間にスレッドが作成されます。次に、ローダーは、BackingMap.setWriteBehind メソッド内の構成設定に従って、そのスレッドを通してデータ要求を委任します。eXtreme Scale トランザクションが、マップのエントリーを挿入、更新、または削除すると、これらの各レコードごとに 1 つずつ LogElement オブジェクトが作成されます。これらのエレメントは後書きローダーに送信され、キュー・マップと呼ばれる特別な ObjectMap 内でキューに入れます。後書き設定が有効になっているバックアップ・マップは、それぞれ独自のキュー・マップを持っています。後書きスレッドは、キューに入れられたデータをキュー・マップから定期的に除去して、実際のバックエンド・ローダーにプッシュします。

後書きローダーは、挿入、更新、および削除タイプの LogElement オブジェクトのみを実際のローダーに送信します。それ以外のタイプの LogElement オブジェクト (例えば、EVICT タイプ) はすべて無視されます。

後書きサポートは、eXtreme Scale をデータベースに組み込む際に使用する Loader プラグインの拡張機能です。例えば、JPA ローダーの構成については 405 ページの『JPA ローダーの構成』の情報を参照してください。

## 利点

後書きサポートを使用可能にすると、以下のような利点があります。

- **バックエンド障害の分離:** 後書きキャッシングは、バックエンド障害からの分離層を提供します。バックエンドのデータベースで障害が発生すると、更新はキュー・マップ内でキューに入れます。アプリケーションは、トランザクション

を eXtreme Scale に送り続けることができます。バックエンドが復旧すると、キュー・マップ内のデータはバックエンドにプッシュされます。

- **バックエンドの負荷の削減:** 後書きローダーは更新をキー単位でマージします。その結果、キュー・マップ内には、キーごとにマージされた更新が 1 つのみ存在します。このマージにより、バックエンド・データベースに対する更新の数が減ります。
- **トランザクション・パフォーマンスの改善:** データがバックエンドと同期されるのをトランザクションが待機する必要がないので、個別の eXtreme Scale トランザクション時間が削減されます。

## ローダー

Loader プラグインを使用すると、通常は、同一システムあるいは別システムの永続ストアに保持されるデータのメモリー・キャッシュとしてデータ・グリッド・マップを動作させることができます。通常、データベースまたはファイル・システムは永続ストアとして使用されます。リモート Java 仮想マシン (JVM) もデータのソースとして使用でき、eXtreme Scale を使用してハブ・ベースのキャッシュを構築できます。ローダーには、永続ストアとの間でデータの読み取りおよび書き込みを行うロジックが備わっています。

## 概要

ローダーは、変更がバックアップ・マップに対して行われた場合、または、バックアップ・マップがデータ要求を満足できない (キャッシュ・ミス) 場合に呼び出されるバックアップ・マップ・プラグインです。ローダーは、キーに関する要求をキャッシュが満足できなくなったときに起動され、リードスルー機能や、キャッシュにデータをゆっくり設定する機能を提供します。また、ローダーによって、キャッシュ値が変わったときのデータベース更新が可能になります。1 つのトランザクション内のすべての変更は、データベースとの対話の数を最小化できるよう、まとめてグループ化されます。ローダーと共に TransactionCallback プラグインが、バックエンド・トランザクションの境界をトリガーするために使用されます。このプラグインの使用は、複数のマップが 1 つのトランザクションに含まれている場合、または、トランザクション・データがコミットなしでキャッシュに書き込まれる場合に重要です。

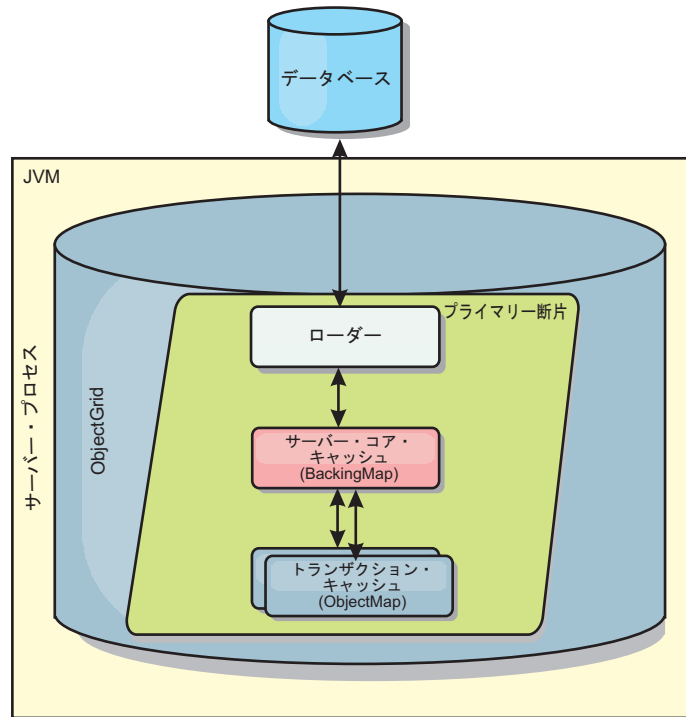


図 15. ローダー

ローダーは、データベース・ロックの保持を回避するために、資格過剰の更新を使用することもできます。バージョン属性をキャッシュ値の中に入れることによって、値がキャッシュ内で更新されるときにローダーは値の前と後のイメージを見ることが可能です。その後、データベースまたはバックエンドを更新する際にこの値を使用して、データが更新されていないことを検証できます。ローダーは、開始時にデータ・グリッドをプリロードするよう構成することもできます。区画に分割されている場合、各区画ごとに 1 つのローダー・インスタンスが関連付けられます。例えば、「Company」マップに 10 個の区画がある場合、プライマリー区画ごとに 1 つずつ、10 個のローダー・インスタンスがあります。このマップのプライマリー断片がアクティブにされると、ローダーに対して `preloadMap` メソッドが同期または非同期で呼び出され、マップ区画にバックエンドからのデータが自動的にロードされます。非同期で呼び出される場合、すべてのクライアント・トランザクションはブロックされ、データ・グリッドへの矛盾するアクセスを防止します。代わりに、クライアント・プリローダーを使用してデータ・グリッド全体にデータをロードできます。

2 つの組み込みローダーにより、リレーショナル・データベース・バックエンドとの統合が非常に単純化されます。JPA ローダーは、Java Persistence API (JPA) 仕様の OpenJPA および Hibernate 実装の両方のオブジェクト関係マッピング (ORM) 機能を使用します。詳しくは、JPA ローダーを参照してください。

複数データ・センター構成でローダーを使用する場合は、どのようにして改訂データとキャッシュの整合性をデータ・グリッド間で維持するかを検討する必要があります。詳しくは、43 ページの『マルチマスター・トポロジーでのローダーについての考慮事項』を参照してください。

## ローダーの構成

ローダーを BackingMap 構成に追加するには、プログラマチック構成または XML 構成を使用します。ローダーには、バックアップ・マップとの間で以下のような関係があります。

- 1 つのバックアップ・マップは 1 つのローダーしか持てない。
- クライアント・バックアップ・マップ (ニア・キャッシュ) はローダーを持ってない。
- 1 つのローダー定義を複数のバックアップ・マップに適用できるが、各バックアップ・マップは独自のローダー・インスタンスを持つ。

## データのプリロードおよびウォームアップ

ローダーのユーザーを組み込む多くのシナリオで、データ・グリッドをデータと一緒にプリロードして準備しておくことができます。

データ・グリッドは、完全キャッシュとして使用される場合、データのすべてを保持しなければならない、いずれかのクライアントが接続する前にデータがロードされている必要があります。スパース・キャッシュを使用する場合は、クライアントが接続時にデータにすぐにアクセスできるように、キャッシュをデータでウォームアップしておくことができます。

以下のセクションで説明するように、データをデータ・グリッドにプリロードする方法は 2 つあります。1 つは Loader プラグインを使用する方法で、もう 1 つはクライアント・ローダーを使用する方法です。

### Loader プラグイン

Loader プラグインは、各マップに関連付けられ、1 つのプライマリー区画断片をデータベースと同期化させる役割を担います。断片がアクティブになると、Loader プラグインの preloadMap メソッドが自動的に呼び出されます。例えば、100 の区画がある場合、ローダーのインスタンスは 100 存在し、それぞれが、各自の区画のためにデータをロードします。同期的に実行された場合、プリロードが完了するまですべてのクライアントがブロックされます。

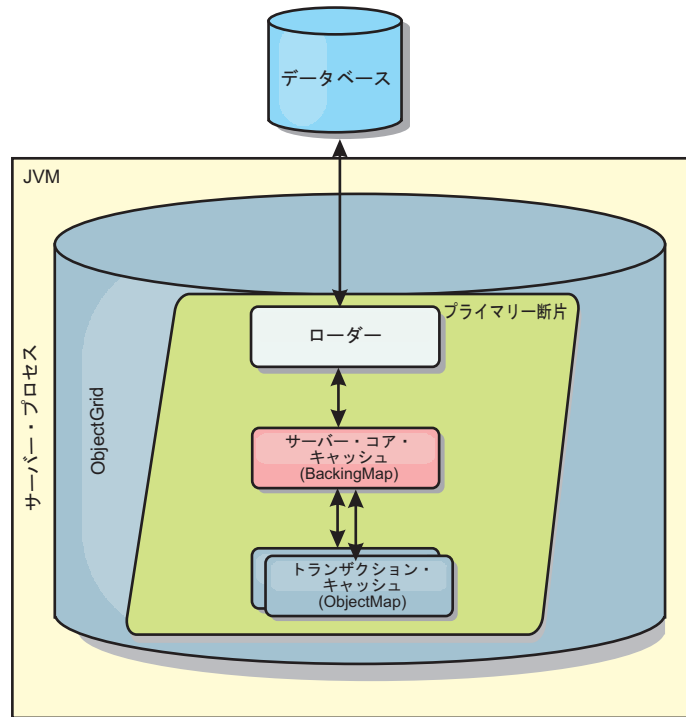


図 16. Loader プラグイン

### クライアント・ローダー

クライアント・ローダーは、1 つ以上のクライアントを使用してグリッドにデータをロードするパターンです。複数のクライアントを使用してグリッドにデータをロードすることは、区画スキーマがデータベースに保管されない場合は効率的です。クライアント・ローダーは手動で呼び出すか、データ・グリッドの開始時に自動的に呼び出すことができます。データ・グリッドにデータをプリロードしている間はクライアントがデータ・グリッドにアクセスできないように、クライアント・ローダーは、オプションで、StateManager を使用してデータ・グリッドの状態をプリロード・モードに設定できます。WebSphere eXtreme Scale には Java Persistence API (JPA) ベースのローダーが組み込まれていて、OpenJPA または Hibernate JPA プロバイダーのどちらかでデータ・グリッドに自動的にロードするために使用できます。キャッシュ・プロバイダーについて詳しくは、378 ページの『JPA レベル 2 (L2) キャッシュ・プラグイン』を参照してください。

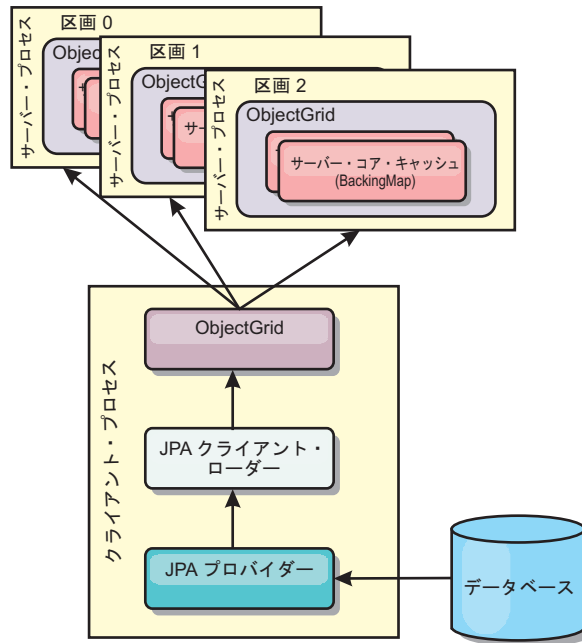


図 17. クライアント・ローダー

## データベースの同期手法

WebSphere eXtreme Scale をキャッシュとして使用する際、データベースを eXtreme Scale トランザクションとは独立して更新できる場合、失効データを許容するようにアプリケーションを作成する必要があります。同期されたメモリー内データベース処理スペースとして機能するため、eXtreme Scale はキャッシュを常に最新の状態に保つ方法をいくつか備えています。

## データベースの同期手法

### 定期的リフレッシュ

時間ベースの Java Persistence API (JPA) データベース・アップデーターを使用して、定期的なキャッシュの無効化または更新を自動的に実行できます。このアップデーターは、JPA プロバイダーを使用してデータベースを定期的に照会することによって、前回の更新以降に発生した更新または挿入があるかどうかを調べます。示された変更は、スパース・キャッシュで使用された場合、自動的に無効にされるか、更新されます。完全キャッシュで使用された場合、エントリーをディスクカバーして、キャッシュに挿入することができます。エントリーがキャッシュから除去されることはありません。



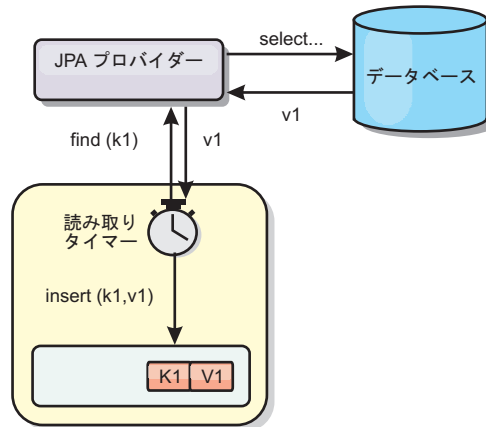


図 18. 定期的リフレッシュ

## 除去

スパス・キャッシュでは、除去ポリシーを使用して、データベースに影響を及ぼすことなく、キャッシュからデータを自動的に除去できます。eXtreme Scale には、Time-To-Live (存続時間)、Least-Recently-Used (最長未使用時間)、および Least-Frequently-Used (最も使用頻度の少ない) という 3 つの組み込みポリシーがあります。メモリー・ベースの除去オプションを使用可能にすると、メモリーが制約状態になるので、3 つのポリシーではすべて、必要であればデータをより積極的に除去することができます。

## イベント・ベースの無効化

スパス・キャッシュおよび完全キャッシュは、Java Message Service (JMS) などのイベント生成プログラムを使用して無効化または更新することができます。JMS を使用した無効化は、データベース・トリガーを使用してバックエンドを更新するなどのプロセスにも手動で関連付けることができます。サーバー・キャッシュで変更があった場合にクライアントに通知できる JMS ObjectGridEventListener プラグインが eXtreme Scale で提供されています。これにより、クライアントが失効データを表示する時間を短縮できます。

## プログラマチックな無効化

eXtreme Scale API により、`Session.beginNoWriteThrough()`、`ObjectMap.invalidate()`、および `EntityManager.invalidate()` API メソッドを使用したニア・キャッシュおよびサーバー・キャッシュの手動対話が可能になります。クライアントまたはサーバーのプロセスでデータの一部がもう必要ない場合、無効化メソッドを使用して、ニア・キャッシュまたはサーバー・キャッシュからデータを除去できます。`beginNoWriteThrough` メソッドは、ローダーを呼び出すことなく、`ObjectMap` または `EntityManager` 操作をローカル・キャッシュに適用します。クライアントから呼び出された場合のこの操作は、ニア・キャッシュのみに適用されます (リモート・ローダーは呼び出されません)。サーバーで呼び出された場合のこの操作は、ローダーを呼び出すことなく、サーバー・コア・キャッシュのみに適用されます。

## データの無効化

失効したキャッシュ・データを削除するには、管理上の、イベント・ベースまたはプログラマチックな無効化メカニズムを使用することができます。

### 8.5+

#### 管理上の無効化

Web コンソールまたは `xscmd` ユーティリティを使用して、キーに基づいてデータを無効化することができます。正規表現を使用してキャッシュ・データをフィルタリングしてから、その正規表現に基づいてデータを無効化することができます。

#### イベント・ベースの無効化

スパス・キャッシュおよび完全キャッシュは、Java Message Service (JMS) などのイベント生成プログラムを使用して無効化または更新することができます。JMS を使用した無効化は、データベース・トリガーを使用してバックエンドを更新するなどのプロセスにも手動で関連付けることができます。サーバー・キャッシュが変更した場合にクライアントに通知できる JMS `ObjectGridEventListener` プラグインが eXtreme Scale で提供されています。この通知タイプによって、クライアントが失効データを表示する時間を短縮します。

イベント・ベースの無効化は、一般的には以下の 3 つのコンポーネントで構成されます。

- **イベント・キュー:** イベント・キューには、データ変更イベントが保管されます。データ変更イベントを管理できるのであれば、イベント・キューは JMS キュー、データベース、メモリー内の FIFO キュー、またはすべての種類のマニフェストの可能性があります。
- **イベント・パブリッシャー:** イベント・パブリッシャーは、データ変更イベントをイベント・キューにパブリッシュします。イベント・パブリッシャーは、通常、作成されたアプリケーションまたは eXtreme Scale プラグインの実装です。イベント・パブリッシャーは、いつデータが変更されたかを知っています。あるいはイベント・パブリッシャーがデータ自体を変更します。トランザクションがコミットすると、変更されたデータに対してイベントが生成され、イベント・パブリッシャーはこれらのイベントをイベント・キューにパブリッシュします。
- **イベント・コンシューマー:** イベント・コンシューマーは、データ変更イベントをコンシュームします。イベント・コンシューマーは、通常アプリケーションで、ターゲット・グリッド・データが他のグリッドからの最新の変更を使用して更新されることを確認します。このイベント・コンシューマーは、イベント・キューと対話をして最新のデータ変更を取得し、ターゲット・グリッドのデータ変更を適用します。イベント・コンシューマーは eXtreme Scale API を使用して、失効データを無効にしたり、グリッドを最新データで更新することができます。

例えば、`JMSObjectGridEventListener` にはクライアント/サーバー・モデルのオプションがあり、そのイベント・キューは指定された JMS 宛先です。すべてのサーバー・プロセスがイベント・パブリッシャーです。トランザクションがコミットすると、サーバーはデータ変更を取得し、それを指定された JMS 宛先にパブリッシュ

します。すべてのクライアント・プロセスがイベント・コンシューマーです。指定された JMS 宛先からデータ変更を受信し、その変更をクライアントのニア・キャッシュに適用します。

詳しくは、「管理ガイド」でクライアント無効化メカニズムの使用可能化に関するトピックを参照してください。

## プログラマチックな無効化

WebSphere eXtreme Scale API により、`Session.beginNoWriteThrough()`、`ObjectMap.invalidate()`、および `EntityManager.invalidate()` API メソッドを使用したニア・キャッシュおよびサーバー・キャッシュの手動対話が可能になります。クライアントまたはサーバーのプロセスでデータの一部がもう必要ない場合、無効化メソッドを使用して、ニア・キャッシュまたはサーバー・キャッシュからデータを除去できます。`beginNoWriteThrough` メソッドは、ローダーを呼び出すことなく、`ObjectMap` または `EntityManager` 操作をローカル・キャッシュに適用します。クライアントから呼び出された場合のこの操作は、ニア・キャッシュのみに適用されます (リモート・ローダーは呼び出されません)。サーバーで呼び出された場合のこの操作は、ローダーを呼び出すことなく、サーバー・コア・キャッシュのみに適用されます。

他の手法と一緒にプログラマチックな無効化を使用して、データをいつ無効にするかを決定します。例えば、この無効化メソッドは、イベント・ベースの無効化メカニズムを使用してデータ変更イベントを受信し、API を使用して失効データを無効にします。

## 索引付け

`MapIndexPlugin` プラグインは、`BackingMap` 上にいくつかの索引を作成して、非キー・データ・アクセスをサポートするために使用します。

## 索引のタイプおよび構成

索引付けフィーチャーは、`MapIndexPlugin` プラグインと表されるか、または略して `Index` で表されます。`Index` は `BackingMap` プラグインです。`BackingMap` では、各索引プラグインが索引構成規則に従っている限り、複数の索引プラグインを構成できます。

索引付けフィーチャーは、1 つ以上の索引を `BackingMap` に作成する場合に使用できます。1 つの索引は、`BackingMap` 内の 1 つのオブジェクトの 1 つの属性または属性のリストから作成されます。このフィーチャーにより、アプリケーションはより迅速に特定のオブジェクトを見つけることができます。索引付けフィーチャーを使用すると、アプリケーションは特定の値を持つオブジェクトや、ある範囲の索引属性値内にあるオブジェクトを見つけることができます。

可能な索引付けには、静的および動的という 2 つのタイプがあります。静的索引付けの場合、`ObjectGrid` インスタンスを初期化する前に、`BackingMap` に索引プラグインを構成する必要があります。この構成を行うには、`BackingMap` を XML で構成するか、またはプログラマチックに構成します。静的索引付けでは、まず最初に、`ObjectGrid` の初期化中に索引を作成します。索引は常に `BackingMap` に同期しており、いつでも使用できる準備ができています。静的索引付けプロセスが既に開始し

ている場合、索引は、eXtreme Scale トランザクション管理プロセスの一環として保守されます。トランザクションが変更をコミットすると、それらの変更は静的索引も更新し、トランザクションがロールバックされれば索引の変更もロールバックされます。

動的索引付けの場合は、索引を含む `ObjectGrid` インスタンスの初期化の前または後に、`BackingMap` に索引を作成することができます。動的索引付けプロセスのライフサイクルはアプリケーションによって制御されるので、不要になったら動的索引を削除することができます。アプリケーションが動的索引を作成する場合は、索引作成プロセスを完了するまでに時間がかかるために、その索引をすぐに使用できないことがあります。この時間は索引付けされるデータの量に依存するので、特定の索引付けイベントが発生したときにそのことを通知してもらいたいアプリケーションのために、`DynamicIndexCallback` インターフェースが提供されています。これらのイベントには、準備完了、エラー、および破棄があります。アプリケーションは、このコールバック・インターフェースを実装し、動的索引付けプロセスに登録できます。

`BackingMap` に索引プラグインが構成されている場合、対応する `ObjectMap` からアプリケーション索引プロキシ・オブジェクトを取得することができます。`ObjectMap` の `getIndex` メソッドを呼び出し、索引プラグインの名前を渡すと、索引プロキシ・オブジェクトが戻されます。索引プロキシ・オブジェクトを適切なアプリケーション索引インターフェース (`MapIndex`、`MapRangeIndex`、またはカスタマイズされた索引インターフェースなど) にキャストする必要があります。索引プロキシ・オブジェクトを取得したら、アプリケーション索引インターフェースで定義されたメソッドを使用して、キャッシュされたオブジェクトを検出することができます。

次のリストに、索引付けの使用手順をまとめます。

- 静的または動的索引プラグインを `BackingMap` に追加します。
- `ObjectMap` の `getIndex` メソッドを発行して、アプリケーション索引プロキシ・オブジェクトを取得します。
- `MapIndex`、`MapRangeIndex` またはカスタマイズされた索引インターフェースなどの適切なアプリケーション索引インターフェースに、索引プロキシ・オブジェクトをキャストします。
- アプリケーション索引インターフェースで定義されたメソッドを使用して、キャッシュされたオブジェクトを検出します。

`HashIndex` クラスは、組み込みアプリケーション索引インターフェースである `MapIndex` と `MapRangeIndex` の両方をサポートすることのできる組み込み索引プラグイン実装です。ユーザー独自の索引を作成することもできます。`HashIndex` を静的索引または動的索引として `BackingMap` に追加して、`MapIndex` または `MapRangeIndex` の索引プロキシ・オブジェクトを取得し、その索引プロキシ・オブジェクトを使用してキャッシュ・オブジェクトを検索することができます。

## デフォルトの索引

ローカル・マップ内のキーを反復処理する場合は、デフォルトの索引を使用できます。この索引はまったく構成を必要としませんが、エージェントを使用するか `ShardEvents.shardActivated(ObjectGrid shard)` メソッドから取得した `ObjectGrid` イン

スタンスを使用して、断片に対して使用しなければなりません。

## データ品質に関する考慮事項

索引照会メソッドの結果が表わすのは、特定の時刻におけるデータのスナップショットのみです。結果がアプリケーションに戻された後には、データ・エンタリーに対するロックは取得されません。アプリケーションは、戻されたデータ・セットに対してデータ更新が発生する可能性があることに注意する必要があります。例えば、アプリケーションは `MapIndex` の `findAll` メソッドを実行して、キャッシュされたオブジェクトのキーを取得します。戻されたこのキー・オブジェクトは、キャッシュ内のデータ項目に関連付けられています。アプリケーションは、キー・オブジェクトを提供することにより、`ObjectMap` に対して `get` メソッドを実行して、オブジェクトを検出できるようになっている必要があります。`get` メソッドが呼び出される直前に、別のトランザクションがキャッシュからそのデータ・オブジェクトを削除した場合、戻される結果はヌルです。

## 索引付けのパフォーマンスに関する考慮事項

索引付けフィーチャーの主な目的の 1 つは、`BackingMap` の全体的なパフォーマンスを改善することです。索引付けの使い方が不適切な場合は、アプリケーションのパフォーマンスが低下する可能性があります。このフィーチャーを使用する前に、次の要因について検討します。

- **並行書き込みトランザクションの数:** 索引処理は、トランザクションが `BackingMap` にデータを書き込むたびに起こります。アプリケーションが索引照会操作を試行しているときに、多くのトランザクションがデータをマップに書き込んでいると、パフォーマンスが低下します。
- **照会操作で戻される結果セットのサイズ:** 結果セットのサイズが大きくなるにつれて、照会のパフォーマンスは低下します。結果セットのサイズが `BackingMap` の 15% 以上になるとパフォーマンスは低下する傾向にあります。
- **同じ `BackingMap` に作成される索引の数:** 各索引がシステム・リソースを消費します。`BackingMap` に作成される索引の数が増えると、パフォーマンスは低下します。

索引付け機能は、`BackingMap` パフォーマンスを大幅に改善できることがあります。理想的なケースは、`BackingMap` の大部分の操作が読み取りであり、照会の結果セットが `BackingMap` エントリーのわずかな割合に過ぎず、ごく少数の索引が `BackingMap` に対して作成される場合です。

## 複数データ・センター・トポロジーの計画

マルチマスター非同期レプリカ生成機能を使用すると、2 つ以上のデータ・グリッドを、互いの正確なミラーにすることができます。各データ・グリッドは独立したカタログ・サービス・ドメイン内でホストされ、独自のカタログ・サービス、コンテナ・サーバー、および固有の名前を所有しています。マルチマスター非同期レプリカ生成機能により、リンクを使用してカタログ・サービス・ドメインのコレクションを接続できます。すると、カタログ・サービス・ドメインは、リンクを介したレプリカ生成を使用して同期されます。カタログ・サービス・ドメイン間のリンクの定義を使用して、ほとんどどのトポロジーでも構成できます。

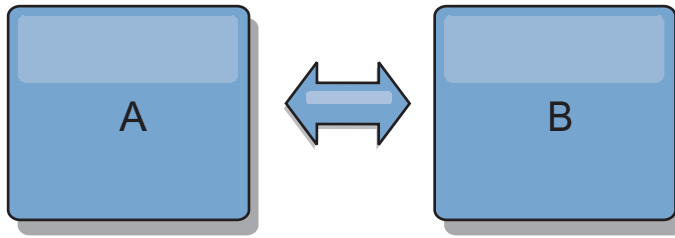


## マルチマスター・レプリカ生成のトポロジー

マルチマスター・レプリカ生成を組み込んだデプロイメントのトポロジーを選択する際、いくつかの異なるオプションがあります。

### カタログ・サービス・ドメイン を接続するリンク

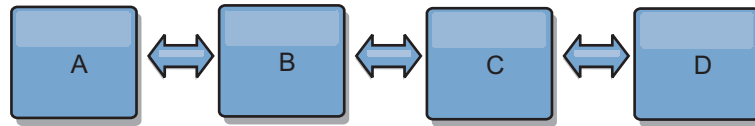
レプリカ生成データ・グリッドのインフラストラクチャーは、カタログ・サービス・ドメイン 間を双方向のリンクで接続したカタログ・サービス・ドメインのグラフです。リンクを使用して、2 つのカタログ・サービス・ドメイン はデータ変更内容をやりとりできます。例えば、最も単純なトポロジーは、カタログ・サービス・ドメイン間に単一のリンクを持つ 1 対のカタログ・サービス・ドメインです。カタログ・サービス・ドメイン は、左から A、B、C というようにアルファベット順で指定されています。リンクは、遠距離にわたる広域ネットワーク (WAN) を経由する場合もあります。リンクが遮断されたとしても、いずれかのカタログ・サービス・ドメイン でまだデータを変更できます。トポロジーは、リンクがカタログ・サービス・ドメイン と再接続したときに変更を調整します。ネットワーク接続が中断されると、リンクは自動的に再接続しようとします。



リンクをセットアップすると、この製品はまず、すべてのカタログ・サービス・ドメイン を同一にしようと試みます。次に、いずれかのカタログ・サービス・ドメイン で変更が発生すると、eXtreme Scale は同一の状態を維持するよう試みます。目標は、各カタログ・サービス・ドメイン がリンクで接続されたすべての他のカタログ・サービス・ドメイン の正確なミラーになることです。カタログ・サービス・ドメイン 間のレプリカ生成リンクは、1 つのカタログ・サービス・ドメイン で行われたすべての変更を確実に他のカタログ・サービス・ドメイン にコピーするのに役立ちます。

### ライン・トポロジー

ライン・トポロジーはこのような単純なデプロイメントですが、かなりのリンク品質を実証します。まず、変更を受け取るために、カタログ・サービス・ドメイン は直接すべての他のカタログ・サービス・ドメイン に接続する必要がありません。カタログ・サービス・ドメイン B はカタログ・サービス・ドメイン A から変更をプルします。カタログ・サービス・ドメイン C は、カタログ・サービス・ドメイン A と C を接続するカタログ・サービス・ドメイン B を介してカタログ・サービス・ドメイン A から変更を受信します。同様に、カタログ・サービス・ドメイン D はカタログ・サービス・ドメイン C を介して別のカタログ・サービス・ドメイン から変更を受信します。この機能により、変更配布の負荷が変更のソースから離れた場所に広がります。



カタログ・サービス・ドメイン C に障害が起こった場合、以下のアクションの発生が考えられることに注意してください。

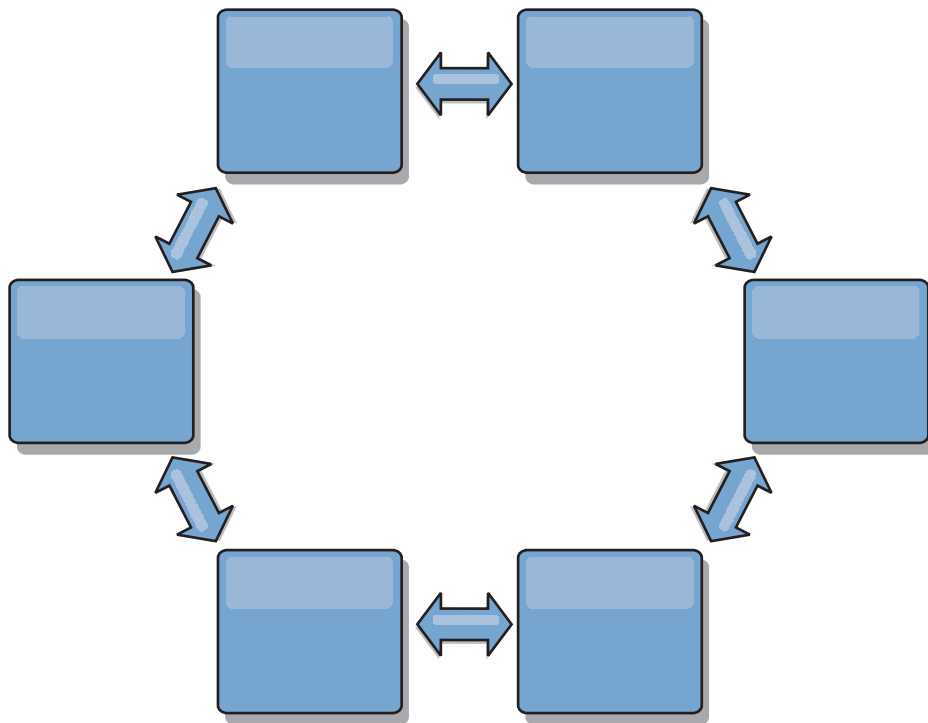
1. カタログ・サービス・ドメイン D は、カタログ・サービス・ドメイン C が再開されるまで孤立します。
2. カタログ・サービス・ドメイン C は、カタログ・サービス・ドメイン A のコピーであるカタログ・サービス・ドメイン B と自分自身を同期させます。
3. カタログ・サービス・ドメイン D は、カタログ・サービス・ドメイン C を使用して、カタログ・サービス・ドメイン A と B で発生した変更と自分自身を同期させます。これらの変更は最初は、カタログ・サービス・ドメイン D が孤立していた間 (カタログ・サービス・ドメイン C がダウンしていた間) に発生しました。

最終的に、カタログ・サービス・ドメイン A、B、C、および D はすべて、互いのドメインと再び同一になります。

### リング・トポロジー

リング・トポロジーは、より回復力のあるトポロジーの例です。カタログ・サービス・ドメイン または単一リンクに障害が起こった場合でも、残ったカタログ・サービス・ドメイン がまだ変更を取得できます。そのカタログ・サービス・ドメイン は、障害から離れて、リングの周りを回ります。リング・トポロジーの大きさには関係なく、各カタログ・サービス・ドメイン は他のカタログ・サービス・ドメイン とのリンクを最大 2 つ持ちます。変更を伝搬するための待ち時間は長くなる場合があります。特定のカタログ・サービス・ドメイン での変更は、すべてのカタログ・サービス・ドメイン にその変更が反映されるまで、複数のリンクを経由して伝搬する必要がある場合があります。ライン・トポロジーにも同じ特性があります。

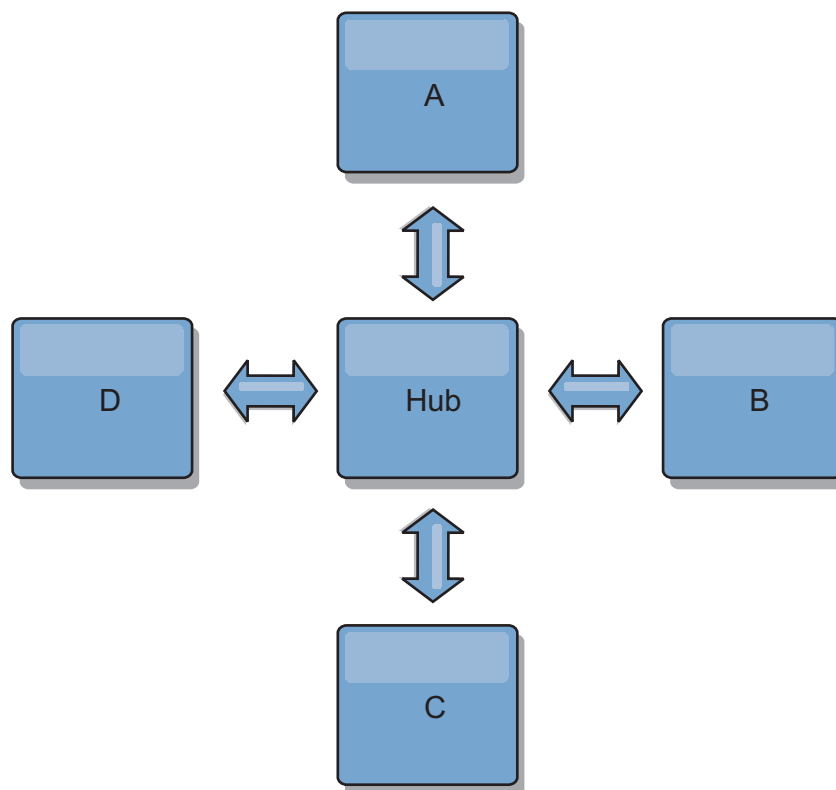




リングの中心に置いたルート・カタログ・サービス・ドメインを使用した、より洗練されたリング・トポロジをデプロイすることも可能です。ルート・カタログ・サービス・ドメインは、調整の中心点として機能します。他のカタログ・サービス・ドメインは、ルート・カタログ・サービス・ドメインで生じた変更に対する調整のリモート・ポイントとして働きます。ルート・カタログ・サービス・ドメインは、カタログ・サービス・ドメイン間の変更をアービトレーションすることができます。ルート・カタログ・サービス・ドメインを囲む複数のリングがリング・トポロジに含まれている場合、カタログ・サービス・ドメインは最も内側にあるリング間の変更のみをアービトレーションすることができます。ただし、アービトレーションの結果は他のリングのカタログ・サービス・ドメインにも広がります。

### ハブ・アンド・スポーク・トポロジ

ハブ・アンド・スポーク・トポロジでは、ハブ・カタログ・サービス・ドメインを経由して変更が伝搬します。ハブは指定される唯一の中間カタログ・サービス・ドメインであるため、ハブ・アンド・スポーク・トポロジでは待ち時間が短縮されます。ハブ・カタログ・サービス・ドメインは、リンク経由ですべてのスポーク・カタログ・サービス・ドメインに接続されています。ハブは、カタログ・サービス・ドメイン間で変更を配布します。ハブは、衝突に対して調整のポイントとして機能します。更新頻度の高い環境では、同期を保つために、スポークよりも多くのハードウェア上でハブを稼働する必要がある場合があります。WebSphere eXtreme Scaleは、直線的に拡大するように設計されています。つまり、問題なく、必要に応じてハブをさらに大きくすることができます。ただし、ハブに障害が起こった場合は、変更はハブが再始動するまで配布されません。スポーク・カタログ・サービス・ドメイン上の変更は、ハブが再接続された後に配布されます。



また、完全に複製したクライアントを使用したストラテジー、すなわち、ハブとして稼働しているサーバーのペアを使用するトポロジーのバリエーションを使用することもできます。各クライアントは、クライアント JVM 内に、必要なものを完備した単一コンテナ・データ・グリッドとカタログを作成します。クライアントは、そのデータ・グリッドを使用してハブ・カタログに接続します。この接続により、クライアントはハブへの接続を取得すると、すぐにハブと同期するようになります。

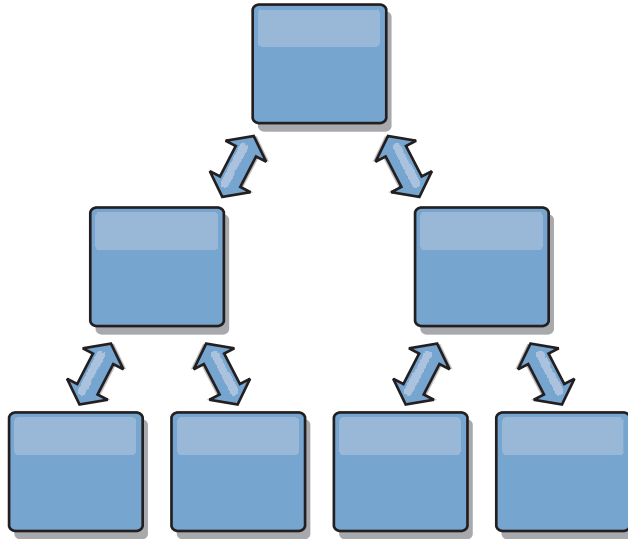
クライアントによって行われた変更は、クライアントに対してローカルで、非同期でハブに複製されます。ハブはアービトレーション・カタログ・サービス・ドメインとして機能し、すべての接続されたクライアントに変更を配布します。完全複製クライアントのトポロジーは、OpenJPA などのオブジェクト・リレーショナル・マップパーに信頼性の高い L2 キャッシュを提供します。変更はハブを介してクライアント JVM 間に迅速に配布されます。キャッシュ・サイズを使用可能なヒープ・スペース内に含むことができる場合、このトポロジーは L2 のこのスタイルにとって信頼できるアーキテクチャーです。

必要であれば、複数の区画を使用して、複数の JVM 上にハブ・カタログ・サービス・ドメインを拡張します。すべてのデータはまだ単一のクライアント JVM に収まらなければならないため、複数の区画を使用してハブの容量を増加させ、変更の配布とアービトレーションを行います。ただし、複数の区画を使用しても、単一カタログ・サービス・ドメインの容量は変更されません。

### ツリー・トポロジー

非循環有向ツリーを使用することもできます。非循環ツリーには循環やループはなく、有向セットアップにより、リンクの存在は親と子の間のみに制限されます。こ

の構成は、多くのカタログ・サービス・ドメインを含むトポロジーで役立ちます。これらのトポロジーでは、すべての可能なスポークに接続される中央ハブを使用することは実用的ではありません。また、このタイプのトポロジーは、ルート・カタログ・サービス・ドメイン を更新することなく子 カタログ・サービス・ドメイン を追加する必要がある場合にも便利です。



ツリー・トポロジーでもまだ、ルート・カタログ・サービス・ドメイン に調整の中心点を置くことができます。第 2 レベルはまだ、それらの下の カタログ・サービス・ドメイン で生じた変更に対する調整のリモート・ポイントとして機能します。ルート・カタログ・サービス・ドメイン は、第 2 レベルにある カタログ・サービス・ドメイン 間の変更のみをアービトレーションすることができます。それぞれが各レベルで  $N$  個の子を持つ、 $n$  進ツリーを使用することもできます。それぞれのカタログ・サービス・ドメイン は、 $n$  個のリンクに接続します。

### 完全複製クライアント

このトポロジー変化には、ハブとして稼働する 1 対のサーバーが含まれます。各クライアントは、クライアント JVM 内に、必要なものを完備した単一コンテナ・データ・グリッドとカタログを作成します。クライアントは、そのデータ・グリッドを使用してハブ・カタログに接続します。これにより、クライアントはハブへの接続を取得すると、すぐにハブと同期するようになります。

クライアントによって行われた変更は、クライアントに対してローカルで、非同期でハブに複製されます。ハブはアービトレーション・カタログ・サービス・ドメイン として機能し、すべての接続されたクライアントに変更を配布します。完全複製クライアントのトポロジーは、OpenJPA などのオブジェクト・リレーショナル・マッパーに適した L2 キャッシュを提供します。変更はハブを介してクライアント JVM 間に迅速に配布されます。キャッシュ・サイズをクライアントの使用可能なヒープ・スペース内に含むことができる限り、このトポロジーは L2 のこのスタイルに適したアーキテクチャーです。

必要であれば、複数の区画を使用して、複数の JVM 上にハブ・カタログ・サービス・ドメイン を拡張します。すべてのデータはまだ単一のクライアント JVM に収まらなければならないため、複数の区画を使用してハブの容量を増加させ、変更

の配布とアービトレーションを行います。単一 カタログ・サービス・ドメインの容量は変更しません。

## マルチマスター・トポロジーに関する構成の考慮事項

マルチマスター・レプリカ生成トポロジーを使用するかどうかを決定し、その使用方法について決定する際は、以下の問題を考慮してください。

### • マップ・セット要件

カタログ・サービス・ドメインのリンクを介して変更を複製するには、マップ・セットは以下の特性を持っている必要があります。

- カatalog・サービス・ドメイン内の ObjectGrid 名およびマップ・セット名は、他のカタログ・サービス・ドメインの ObjectGrid 名およびマップ・セット名と一致していなければならない。例えば、ObjectGrid 「og1」およびマップ・セット 「ms1」がカタログ・サービス・ドメイン A とカタログ・サービス・ドメイン B で構成されていないと、それらのカタログ・サービス・ドメイン間でマップ・セット内のデータを複製できません。
- FIXED\_PARTITION データ・グリッドである。PER\_CONTAINER データ・グリッドを複製できません。
- 
- 各カタログ・サービス・ドメイン内の同じデータ・タイプが複製される
- 各カタログ・サービス・ドメイン内に同じマップおよび動的マップ・テンプレートが含まれている。
- エンティティ・マネージャーを使用しない。エンティティ・マップを含むマップ・セットは、カタログ・サービス・ドメインを介して複製されません。
- 後書きキャッシング・サポートを使用しない。後書きサポートで構成されたマップを含むマップ・セットは、カタログ・サービス・ドメインを介して複製されません。

トポロジー内のカタログ・サービス・ドメインが開始されると、前述の特性を持つすべてのマップ・セットが複製を開始します。

### • 複数のカタログ・サービス・ドメインを使用するクラス・ローダー

カタログ・サービス・ドメインは、キーおよび値として使用されるクラスすべてへのアクセス権限を持たなければなりません。すべての依存関係は、すべてのドメインのデータ・グリッド・コンテナー Java 仮想マシン (JVM) に対するすべてのクラスパスに反映されなければなりません。CollisionArbiter プラグインがキャッシュ・エントリーの値を取得する場合、その値に対するクラスはアービターを開始するドメインに存在しなければなりません。

## マルチマスター・トポロジーでのローダーについての考慮事項

マルチマスター・トポロジーでローダーを使用する場合は、起こり得る衝突および改訂情報の維持についての問題を考慮する必要があります。データ・グリッドはその中の各項目について改訂情報を維持しており、構成内の他のプライマリ断片がデータ・グリッドにエントリーを書き込むときに衝突を検出できるようになっています。エントリーがローダーから追加されると、この改訂情報は含まれず、エントリーは新しい改訂を持つようになります。エントリーの改訂は新規挿入に見える

ため、別のプライマリー断片もこの状態を変更したり、ローダーから同じ情報を引き込んだりした場合に、偽の衝突が発生する場合があります。

レプリカ生成の変更は、データ・グリッド内に今はないが、レプリカ生成トランザクション中に変更されるキーのリストを使用して、ローダーに対して `get` メソッドを呼び出します。レプリカ生成が行われると、これらのエントリーは衝突エントリーとなります。衝突をアービトレーションし、改訂を適用すると、バッチ更新がローダーで呼び出されて変更内容がデータベースに適用されます。改訂ウィンドウで変更されたマップはすべて、同じトランザクションで更新されます。

## プリロードの問題

データ・センター A とデータ・センター B を使用した 2 つのデータ・センター・トポロジーがあるとします。2 つのデータ・センターはそれぞれ独立したデータベースを持っていますが、データ・センター A にのみ、実行中のデータ・グリッドがあります。マルチマスター構成でデータ・センター間のリンクを確立すると、データ・センター A 内のデータ・グリッドがデータ・センター B 内の新規データ・グリッドにデータをプッシュし始め、すべてのエントリーとの衝突を引き起こします。別の大きな問題は、データ・センター A 内のデータベースには存在せず、データ・センター B 内のデータベースにあるすべてのデータで発生します。これらの行にはデータが取り込まれず、アービトレーションされません。結果として、解決されない不整合が発生します。

## プリロードの問題に対する解決策

データベース内にのみ存在するデータは改訂を持つことができないため、常にローカル・データベースからデータ・グリッドを完全にプリロードした後、マルチマスター・リンクを設定する必要があります。次に、両方のデータ・グリッドはデータを改訂し、アービトレーションすることができ、最終的に整合した状態に達します。

## スパス・キャッシュの問題

スパス・キャッシュを使用すると、アプリケーションはまずデータ・グリッド内のデータの検索を試みます。データがデータ・グリッド内にないと、ローダーを使用してデータベースでデータが検索されます。キャッシュ・サイズを小規模に維持するために、エントリーは定期的にデータ・グリッドから除去されます。

このキャッシュ・タイプは、マルチマスター構成シナリオでは問題となる場合があります。なぜなら、データ・グリッド内のエントリーは、衝突が発生するときやどちら側が変更を行ったかを検出するのを助ける、改訂用メタデータを持っているためです。データ・センター間のリンクが機能していない場合、一方のデータ・センターがエントリーを更新し、最終的にデータ・グリッド内のデータベースを更新し、エントリーを無効化することができます。リンクが復旧すると、データ・センターは互いに改訂を同期しようとします。しかし、データベースが更新され、データ・グリッド・エントリーが無効化されているため、ダウンしていたデータ・センターの観点から見ると、変更が失われています。結果として、両側のデータ・グリッドで同期がとれず、整合性がなくなります。

## スパース・キャッシュの問題に対する解決策

### ハブおよびスポーク・トポロジー:

ハブおよびスポーク・トポロジーのハブでのみローダーを実行し、結果として、データの整合性を維持しながら、データ・グリッドをスケールアウトすることができます。ただし、このデプロイメントを検討している場合は、ローダーがデータ・グリッドを部分的にロードできることに注意してください。これは、Evictor が構成済みであることを意味します。構成のスポークがスパース・キャッシュだが、ローダーがない場合は、どのキャッシュ・ミスもデータベースからデータを取り出すことができません。この制約事項のため、ハブおよびスポーク構成では、完全に取り込まれたキャッシュ・トポロジーを使用する必要があります。

### 無効化および除去

無効化により、データ・グリッドとデータベース間の不整合が発生します。プログラマチックに、または除去機能を使用して、データ・グリッドからデータを削除できます。アプリケーションの開発時に、改訂処理では無効化された変更内容は複製されず、プライマリ断片間で不整合が発生しないよう注意する必要があります。

無効化イベントは、キャッシュ状態変更ではなく、レプリカ生成は生じません。いかなる構成済み Evictor も構成内の他の Evictor と独立して実行されます。例えば、カタログ・サービス・ドメインでのメモリーしきい値について構成済みの Evictor が 1 つあるが、リンクされている他のカタログ・サービス・ドメインに異なるタイプのあまり活動的でない Evictor がある場合があります。データ・グリッド・エントリーがメモリーしきい値ポリシーのために削除されても、他のカタログ・サービス・ドメイン内のエントリーは影響を受けません。

### データベースの更新およびデータ・グリッドの無効化

問題が発生するのは、バックグラウンドで直接データベースを更新しながら、マルチマスター構成で更新済みエントリーについてデータ・グリッドに対して無効化を呼び出しているときです。この問題は、いくつかのタイプのキャッシュ・アクセスがエントリーをデータ・グリッドに移動するまで、データ・グリッドが別のプライマリ断片への変更を複製できないために発生します。

### 単一論理データベースへの複数の書き込みプログラム

ローダーを介して接続された複数のプライマリ断片と一緒に単一データベースを使用していると、トランザクションの競合が発生します。ローダーの実装は、特にこれらのタイプのシナリオを処理する必要があります。

### マルチマスター・レプリカ生成を使用したデータのミラーリング

独立したカタログ・サービス・ドメインに接続された独立したデータベースを構成できます。この構成では、ローダーはあるデータ・センターの変更内容を別のデータ・センターにプッシュできます。

### マルチマスター・レプリカ生成での設計上の考慮事項

マルチマスター・レプリカ生成を実装する場合、アービトレーション、リンク作成、およびパフォーマンスなど、設計における側面を考慮する必要があります。



## トポロジー設計におけるアービトレーションの考慮事項

同じレコードが 2 個所で同時に変更される可能性がある場合には、変更の競合が生じることがあります。各カタログ・サービス・ドメインが、同程度のプロセッサ、メモリー、ネットワーク・リソースを持つようにセットアップしてください。変更の衝突処理 (アービトレーション) を実行しているカタログ・サービス・ドメインは、他のカタログ・サービス・ドメインよりも多くのリソースを使用することに気付くことがあります。衝突は、自動的に検出されます。衝突は、以下の 2 つのメカニズムの 1 つを使用して処理されます。

- **デフォルト衝突アービター:** デフォルトのプロトコルは、字句的に最も小さい名前の付いたカタログ・サービス・ドメインからの変更を使用します。例えば、カタログ・サービス・ドメイン A と B によってレコードの競合が生じる場合には、カタログ・サービス・ドメイン B の変更は無視されます。カタログ・サービス・ドメイン A はそのバージョンを保持し、カタログ・サービス・ドメイン B のレコードはカタログ・サービス・ドメイン A からのレコードに一致するように変更されます。この動作は、ユーザーやセッションが正常にバインドされているアプリケーション、またはユーザーやセッションがデータ・グリッドの 1 つにアフィニティーを持つ対象となるアプリケーションにも同様に適用されます。
- **カスタム衝突アービター:** アプリケーションはカスタム・アービターを提供することができます。カタログ・サービス・ドメインは衝突を検出すると、アービターを開始します。便利なカスタム・アービターの開発について詳しくは、マルチマスター・レプリカ生成のためのカスタム・アービターの作成を参照してください。

衝突が起こる可能性のあるトポロジーに対しては、ハブ・アンド・スポーク・トポロジーまたはツリー・トポロジーの実装を検討してください。これらの 2 つのトポロジーは、以下のシナリオで発生する可能性のある、恒常的な衝突の回避につながります。

1. 複数のカタログ・サービス・ドメインで衝突が発生します。
2. 各カタログ・サービス・ドメインが衝突をローカルで処理し、改訂を生成します。
3. 改訂が衝突し、その結果、改訂の改訂をもたらします。

衝突を回避するには、カタログ・サービス・ドメインのサブセットの衝突アービターとして、アービトレーション・カタログ・サービス・ドメイン と呼ばれる特定のカタログ・サービス・ドメインを選択します。例えば、ハブ・アンド・スポーク・トポロジーはハブを衝突ハンドラーとして使用する場合があります。スポーク衝突ハンドラーは、スポーク・カタログ・サービス・ドメインで検出されたすべての衝突を無視します。ハブ・カタログ・サービス・ドメインは、改訂を作成し、予期しない衝突の改訂を回避します。衝突を処理するように割り当てられたカタログ・サービス・ドメインは、衝突の処理に責任を持つすべてのドメインにリンクしていなければなりません。ツリー・トポロジーでは、内部の親ドメインが自分の直接の子の衝突を処理します。対照的に、リング・トポロジーを使用する場合、リング内の 1 つのカタログ・サービス・ドメインをアービターとして指定することはできません。

次の表に、さまざまなトポロジーと互換性のあるアービトレーション・アプローチをまとめました。



表 1. アービトレーション・アプローチ：この表は、アプリケーション・アービトレーションがさまざまなトポロジーと互換性があるかどうかについて記述します。

トポロジー	アプリケーション・アービトレーション?	注
2 つのカatalog・サービス・ドメインのライン	はい	1 つのカatalog・サービス・ドメインをアービターとして選択します。
3 つのカatalog・サービス・ドメインのライン	はい	真ん中のカatalog・サービス・ドメインがアービターでなければなりません。真ん中のカatalog・サービス・ドメインが、単純なハブ・アンド・スポーク・トポロジーのハブだと考えてください。
3 つより多いカatalog・サービス・ドメインのライン	いいえ	アプリケーション・アービトレーションはサポートされません。
N 個のスポークを持つハブ	はい	すべてのスポークへのリンクを持つハブがアービトレーション・カatalog・サービス・ドメインでなければなりません。
N 個のカatalog・サービス・ドメインのリング	いいえ	アプリケーション・アービトレーションはサポートされません。
非循環有向ツリー (n 進ツリー)	はい	すべてのルート・ノードは、自分の直接の子孫のみを評価する必要があります。

## トポロジー設計におけるリンクの考慮事項

変更待ち時間、フォールト・トレランス、およびパフォーマンス特性におけるトレードオフを最適化している間、トポロジーにはリンクの最小数が含まれているのが理想的です。

### • 変更待ち時間

変更待ち時間は、変更が特定のカatalog・サービス・ドメインに到着する前に経由しなければならない中間カatalog・サービス・ドメインの数によって決まります。

トポロジーが、すべてのカatalog・サービス・ドメインを他のすべてのカatalog・サービス・ドメインにリンクすることによって中間カatalog・サービス・ドメインを除去すれば、トポロジーの変更待ち時間は最善になります。ただし、カatalog・サービス・ドメインはそのリンク数に比例してレプリカ生成作業を実行しなければなりません。大規模トポロジーの場合、非常に多くのリンクが定義され、管理が負担になる場合があります。

変更が他のカatalog・サービス・ドメインにコピーされる速度は、以下の追加要因によって異なります。

- ソース・カatalog・サービス・ドメイン上のプロセッサとネットワーク帯域幅
- ソース・カatalog・サービス・ドメインとターゲット・カatalog・サービス・ドメインの間の中間カatalog・サービス・ドメイン数とリンク数

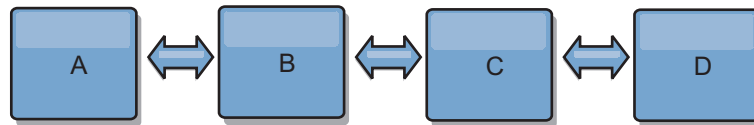
- ソース・カタログ・サービス・ドメイン、ターゲット・カタログ・サービス・ドメイン、および中間カタログ・サービス・ドメインで使用可能なプロセッサとネットワーク・リソース

#### • フォールト・トレランス

フォールト・トレランスは、変更のレプリカ生成のために、2 つのカタログ・サービス・ドメイン間に存在するパス数によって決定します。

特定のカタログ・サービス・ドメインのペア間に 1 つしかリンクがないと、リンク障害が発生した場合に変更を伝搬できません。同様に、中間ドメインのいずれかでリンク障害が発生すると、カタログ・サービス・ドメイン間で変更が伝搬されません。あるカタログ・サービス・ドメインから別のカタログ・サービス・ドメインへの単一リンクが中間ドメインを経由するトポロジーを考えることができます。その場合、中間カタログ・サービス・ドメインのいずれかがダウンすると、変更が伝搬されません。

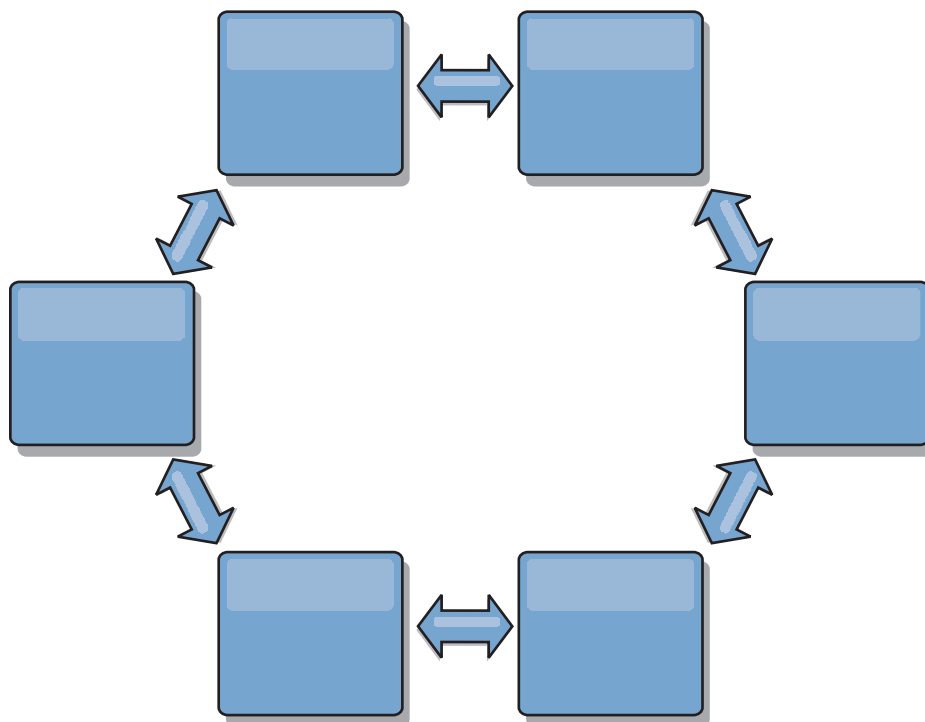
4 つのカタログ・サービス・ドメイン A、B、C、および D を持つライン・トポロジーを考えてみます。



以下のいくつかの状態のままであれば、ドメイン D は A からの変更はまったく見えません。

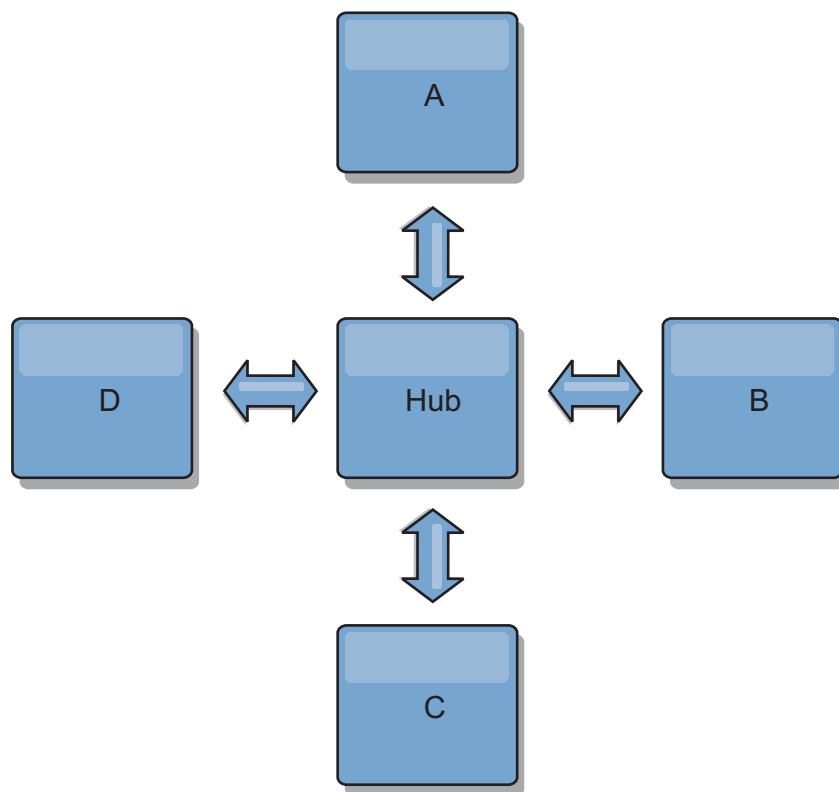
- ドメイン A が稼働中で B がダウン
- ドメイン A および B が稼働中で C がダウン
- A と B の間のリンクがダウン
- B と C の間のリンクがダウン
- C と D の間のリンクがダウン

対照的に、リング・トポロジーの場合、各カタログ・サービス・ドメインはどちらかの方向から変更を受け取ることができます。



例えば、リング・トポロジー内の特定のカタログ・サービスがダウンしている場合、2つの隣接ドメインはまだ互いに変更を直接プルすることができます。

すべての変更はハブを経由して伝搬されます。したがって、ライン・トポロジーやリング・トポロジーとは対照的に、ハブ・アンド・スポーク設計は、ハブに障害が起きた場合に機能停止となる可能性が高いといえます。



単一カタログ・サービス・ドメインは、ある量のサービス損失に対しては回復力があります。ただし、広域ネットワーク障害や物理データ・センター間のリンク障害などのより大規模な障害が発生した場合は、いずれかのカタログ・サービス・ドメインが中断される可能性があります。

#### • リンク作成およびパフォーマンス

カタログ・サービス・ドメイン上に定義されるリンク数は、パフォーマンスに影響します。リンクが多いと使われるリソースも多くなり、結果的にレプリカ生成パフォーマンスが落ちる場合もあります。他のドメインを介してドメイン A の変更を取得する機能は、そのトランザクションを各場所に複製するドメイン A の負荷を効果的に軽減します。ドメイン上の変更配布の負荷は、トポロジー内のドメインの数ではなく、ドメインが使用するリンクの数によって制限されます。このロード・プロパティは、スケーラビリティを提供するため、トポロジー内のドメインは変更の配布に伴う負荷を分配できます。

カタログ・サービス・ドメインは、他のカタログ・サービス・ドメインを間接的に経由して変更を取得できます。5 つのカタログ・サービス・ドメインを持つライン・トポロジーを考えてみます。

A <=> B <=> C <=> D <=> E

- A は、B、C、D、および E から B を介して変更をプルします。
- B は、A と C からは直接、D と E からは C を介して変更をプルします。
- C は、B と D からは直接、A からは B を介して、E からは D を介して変更をプルします。
- D は、C と E からは直接、A と B からは C を介して変更をプルします。

- E は、D からは直接、A、B、および C からは D を介して変更をプルします。

カタログ・サービス・ドメイン A および E は、それぞれ単一カタログ・サービス・ドメインへのリンクのみを持っているので、配布の負荷は最も低くなります。ドメイン B、C、および D は、それぞれ 2 つのドメインへのリンクを持っています。つまり、ドメイン B、C、および D 上の配布の負荷は、ドメイン A および E 上の負荷の 2 倍になります。ワークロードは、トポロジー内のドメイン総数ではなく、各ドメインのリンク数によって決まります。つまり、記述される負荷の分散は、ラインに 1000 ドメインを含んだとしても一定のままです。

## マルチマスター・レプリカ生成のパフォーマンスに関する考慮事項

マルチマスター・レプリカ生成トポロジーを使用する際は、以下の制限を考慮してください。

- **変更の配布のチューニング**は、前のセクションで説明したとおりです。
- **レプリカ生成リンクのパフォーマンス** WebSphere eXtreme Scale は、任意の一对の JVM 間で、単一の TCP/IP ソケットを作成します。JVM 間のすべてのトラフィックは、マルチマスター・レプリカ生成のトラフィックも含め、単一ソケットを経由して発生します。カタログ・サービス・ドメインは少なくとも  $n$  個のコンテナ JVM でホストされ、少なくとも  $n$  個の TCP リンクをピア・カタログ・サービス・ドメインに提供しています。つまり、コンテナ数をより多く持つカタログ・サービス・ドメインには、より高いレプリカ生成のパフォーマンス・レベルがあります。より多くのコンテナがあると、より多くのプロセッサとネットワーク・リソースが必要になります。
- **TCP スライディング・ウィンドウのチューニングおよび RFC 1323** リンクの両端の RFC 1323 サポートを使用して、より多くのデータが往復します。このサポートにより高いスループットが実現され、約 16,000 の要因でウィンドウの容量が拡張されます。

TCP ソケットが、スライディング・ウィンドウのメカニズムを使用して大量データのフローを制御することを思い出してください。このメカニズムは、通常、往復のインターバルのソケットを 64 KB に制限します。往復のインターバルが 100 ミリ秒の場合、追加チューニングをすることなく帯域幅は 640 KB/秒に制限されます。リンクで使用可能な帯域幅を完全に使用する場合は、オペレーティング・システムに固有のチューニングが必要になることがあります。ほとんどのオペレーティング・システムにはチューニング・パラメーターがあり、高度な待ち時間リンクのスループットを向上させる RFC 1323 オプションも含まれます。

以下の複数の要因がレプリカ生成のパフォーマンスに影響する可能性があります。

- eXtreme Scale が変更を取得する速度。
- eXtreme Scale が取得レプリカ生成要求をサービスできる速度。
- スライディング・ウィンドウの容量。
- リンクの両端のネットワーク・バッファをチューニングすると、eXtreme Scale は、効率的にソケット上の変更を取得します。
- **オブジェクト・シリアライゼーション** すべてのデータはシリアライズ可能でなければなりません。カタログ・サービス・ドメインが COPY\_TO\_BYTES を使用して

いない場合、そのカタログ・サービス・ドメインは Java シリアライゼーションまたは ObjectTransformers を使用して、シリアライゼーション・パフォーマンスを最適化する必要があります。

- **圧縮** WebSphere eXtreme Scale は、デフォルトでカタログ・サービス・ドメイン間で送信されるすべてのデータを圧縮します。現在、圧縮を使用不可にすることはできません。
- **メモリー・チューニング** マルチマスター・レプリカ生成トポロジーのメモリー使用量は、トポロジー内のカタログ・サービス・ドメイン数とはほとんど関係ありません。

マルチマスター・レプリカ生成を使用すると、バージョン管理を扱うマップ・エントリーごとに一定の処理量が追加されます。各コンテナはトポロジー内の各カタログ・サービス・ドメインの一定量のデータも追跡します。2つのカタログ・サービス・ドメインを持つトポロジーは、50 カタログ・サービス・ドメインを持つトポロジーとほぼ同じメモリーを使用します。WebSphere eXtreme Scale は、その実装環境のリプレイ・ログや類似のキューを使用しません。すなわち、レプリカ生成リンクがかなりの期間使用できず、後で再開する場合、リカバリー構造は準備されていません。

---

## 他の製品とのインターオペラビリティ

WebSphere eXtreme Scale を他のサーバー製品 (WebSphere Application Server や WebSphere Application Server Community Edition など) と統合することができます。

### WebSphere Application Server

WebSphere Application Server を WebSphere eXtreme Scale 構成のさまざまな側面に統合できます。データ・グリッド・アプリケーションをデプロイし、WebSphere Application Server を使用して、コンテナ・サーバーおよびカタログ・サーバーをホストできます。あるいは、スタンドアロンのカタログ・サーバーとコンテナ・サーバーが存在する WebSphere Application Server 環境に WebSphere eXtreme Scale クライアントがインストールされた混合環境を使用することもできます。

WebSphere Application Server セキュリティーを WebSphere eXtreme Scale 環境で使用することもできます。

### WebSphere Business Process Management および WebSphere Connectivity 製品

WebSphere Integration Developer、WebSphere Enterprise Service Bus、WebSphere Process Server などの WebSphere Business Process Management および WebSphere Connectivity 製品は、CICS、Web サービス、データベース、または JMS トピック およびキューといったバックエンド・システムを統合します。構成に WebSphere eXtreme Scale を追加して、これらのバックエンド・システムの出力をキャッシュすることで、構成の全体的パフォーマンスを向上させることができます。

### WebSphere Commerce

WebSphere Commerce では動的キャッシュの代わりに WebSphere eXtreme Scale キャッシングを利用できます。重複する動的キャッシュ・エントリーを除去

し、高ストレス状況下でキャッシュの同期を維持するために頻繁に行わなければならない無効化処理を回避することで、パフォーマンス、スケーリング、および高可用性を向上させることができます。

## WebSphere Portal

WebSphere Portal の HTTP セッションを WebSphere eXtreme Scale のデータ・グリッドに保持できます。

## WebSphere Application Server Community Edition

WebSphere Application Server Community Edition はセッション状態を共有できますが、効率的でスケーラブルな方法ではありません。 WebSphere eXtreme Scale は、状態の複製に使用できるハイパフォーマンスな分散パーシスタンス層を提供しますが、WebSphere Application Server の外部にあるアプリケーション・サーバーと容易には統合しません。この 2 つの製品を統合することで、スケーラブルなセッション管理ソリューションを提供することができます。

## WebSphere Real Time

WebSphere Real Time (業界最先端のリアルタイム Java 製品) のサポートにより、WebSphere eXtreme Scale は、Extreme Transaction Processing (XTP) アプリケーションが、より安定した予測可能な応答時間を得られるようにします。

## モニター

一般的によく使われるいくつかのエンタープライズ・モニタリング・ソリューションを使用して、WebSphere eXtreme Scale をモニターすることができます。パブリックにアクセス可能な管理 Bean を使用して WebSphere eXtreme Scale をモニターする IBM Tivoli Monitoring および Hyperic HQ 用に、プラグイン・エージェントが組み込まれています。CA Wily Introscope は Java メソッドのインスツルメンテーションを使用して、統計情報を収集します。

---

## 構成の計画

ハードウェアまたはソフトウェアを構成する前に、次の考慮事項について理解する必要があります。

### ネットワーク・ポートの計画

WebSphere eXtreme Scale は、分散キャッシュであり、オブジェクト・リクエスト・ブローカー (ORB) および伝送制御プロトコル (TCP) スタックを使用して Java 仮想マシン間で通信するためにオープン・ポートを必要とします。特に、ファイアウォールのある環境や、カタログ・サービスやコンテナを複数ポートで使用している場合は、ポートを計画し、制御します。

**重要:** ポート番号を指定する際は、オペレーティング・システムで一時ポート範囲にあるポートを設定することは避けてください。一時ポート範囲にあるポートを使用すると、ポートの競合が発生する場合があります。



## カタログ・サービス・ドメイン

カタログ・サービス・ドメインでは、以下のポートを定義する必要があります。

### peerPort

高可用性 (HA) マネージャーがピア・カタログ・サーバー間で TCP スタックを介して通信するためのポートを指定します。WebSphere Application Server では、この設定は HA マネージャー・ポート構成によって継承されます。

### clientPort

カタログ・サーバーがカタログ・サービス・データにアクセスするためのポートを指定します。WebSphere Application Server では、このポートは、カタログ・サービス・ドメイン構成を介して設定されます。

### listenerPort

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポート番号を指定します。この設定は、コンテナおよびクライアントが ORB を介してカタログ・サービスと通信するように構成します。WebSphere Application Server では、listenerPort は BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。

デフォルト: 2809

### JMXConnectorPort

Java Management Extensions (JMX) サービスのバインド先の Secure Sockets Layer (SSL) ポートを定義します。

### JMXServicePort

MBean サーバーが Java Management Extensions (JMX) との通信を listen するポート番号を指定します。JMXServicePort プロパティは、JMX 用の非 SSL ポートを指定します。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**JMXServicePort** とポート番号を明示的に指定してください。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。(スタンドアロン環境の場合のみ必須。)

デフォルト: カatalog・サーバーの場合は 1099

### jvmArgs (オプション)

Java 仮想マシン (JVM) 引数リストを指定します。セキュリティーが有効になっているときは、**startOgServer** スクリプトの **-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>** 引数を使用して、Secure Sockets Layer (SSL) ポートを構成する必要があります。

## コンテナ・サーバー

WebSphere eXtreme Scale コンテナ・サーバーも、いくつかのポートが作動することを必要とします。デフォルトでは、eXtreme Scale コンテナ・サーバーは、HA マネージャー・ポートおよび ORB リスナー・ポートを、動的ポートと共に自動的に生成します。ファイアウォールのある環境では、ポートの計画を立て、それらを

制御することが有益です。コンテナ・サーバーが特定のポートを使用して始動するようにするために、**startOgServer** コマンドで以下のオプションを使用できます。

#### **haManagerPort**

HA マネージャーが使用するポート番号を指定します。このプロパティが設定されていない場合、空きポートが選択されます。このプロパティは、WebSphere Application Server 環境では無視されます。

#### **listenerPort**

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポート番号を指定します。この設定は、コンテナおよびクライアントが ORB を介してカタログ・サービスと通信するように構成します。WebSphere Application Server では、listenerPort は BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。

デフォルト: 2809

#### **JMXConnectorPort**

Java Management Extensions (JMX) サービスのバインド先の Secure Sockets Layer (SSL) ポートを定義します。

#### **JMXServicePort**

MBean サーバーが Java Management Extensions (JMX) との通信を listen するポート番号を指定します。JMXServicePort プロパティは、JMX 用の非 SSL ポートを指定します。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**JMXServicePort** とポート番号を明示的に指定してください。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。(スタンドアロン環境の場合のみ必須。)

デフォルト: カatalog・サーバーの場合は 1099

#### **xioChannel.xioContainerTCPSecure.Port**

サーバー上の eXtremeIO の SSL ポート番号を指定します。**transportType** プロパティが SSL-Supported または SSL-Required に設定されている場合のみこのプロパティが使用されます。

#### **xioChannel.xioContainerTCPNonSecure.Port**

サーバー上の eXtremeIO の非セキュア・リスナー・ポート番号を指定します。値を設定しなければ、一時ポートが使用されます。**transportType** プロパティが TCP/IP に設定されている場合のみこのプロパティが使用されます。

**制約事項:** xioChannel.xioContainerTCPNonSecure.Port プロパティは Liberty プロファイルではサポートされません。

#### **jvmArgs (オプション)**

Java 仮想マシン (JVM) 引数リストを指定します。セキュリティが有効になっているときは、**startOgServer** スクリプトの **-jvmArgs** **-Dcom.ibm.CSI.SSLPort=<sslPort>** 引数を使用して、Secure Sockets Layer (SSL) ポートを構成する必要があります。

ポート制御の適切な計画は、数百の Java 仮想マシンを 1 台のマシンで開始する場合、不可欠です。ポートの競合があると、コンテナ・サーバーが始動しません。

## クライアント

WebSphere eXtreme Scale クライアントは、DataGrid API またはいくつかの他のコマンドを使用しているときに、サーバーからコールバックを受信できます。クライアントがサーバーからのコールバックを listen するポートを指定するには、クライアント・プロパティ・ファイル内の **listenerPort** プロパティを使用します。

### haManagerPort

HA マネージャーが使用するポート番号を指定します。このプロパティが設定されていない場合、空きポートが選択されます。このプロパティは、WebSphere Application Server 環境では無視されます。

### JVM arguments (オプション)

Java 仮想マシン (JVM) 引数リストを指定します。セキュリティーが有効になっているときは、クライアント・プロセスの開始時に `-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>` システム・プロパティを使用する必要があります。

### listenerPort

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポート番号を指定します。この設定は、コンテナおよびクライアントが ORB を介してカタログ・サービスと通信するように構成します。WebSphere Application Server では、listenerPort は BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。

デフォルト: 2809

## WebSphere Application Server のポート

- **listenerPort** 値は、各 WebSphere Application Server アプリケーション・サーバーの **BOOTSTRAP\_ADDRESS** 値から継承されます。
- **haManagerPort** および **peerPort** 値は、各 WebSphere Application Server アプリケーション・サーバーの **DCS\_UNICAST\_ADDRESS** 値から継承されます。

288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』で示されているように、カタログ・サービス・ドメインは管理コンソールで定義できます。

管理コンソールで以下のパスの 1 つをクリックして、特定のサーバーのポートを表示できます。

- WebSphere Application Server Network Deployment バージョン 6.1: 「サーバー」 > 「アプリケーション・サーバー」 > 「*server\_name*」 > 「ポート」 > 「*end\_point\_name*」。
- WebSphere Application Server Network Deployment バージョン 7.0: 「サーバー」 > 「サーバー・タイプ」 > 「WebSphere Application Server」 > 「*server\_name*」 > 「ポート」 > 「*port\_name*」。

## セキュリティの概要

WebSphere eXtreme Scale はデータ・アクセスを保護し、外部セキュリティー・プロバイダーと統合することができます。

**注:** データベースなど、既存の非キャッシュ・データ・ストアでは、積極的に構成したり、有効にしたりする必要のない組み込みセキュリティー・フィーチャーがある可能性があります。ただし、eXtreme Scale でデータをキャッシュした後では、その結果として生じる、バックエンドのセキュリティー・フィーチャーが効力を持たなくなるような重要な状況を考慮する必要があります。eXtreme Scale セキュリティーを必要なレベルで構成すると、データの新しいキャッシュ・アーキテクチャーも保護できます。

以下に、eXtreme Scale セキュリティー機能について簡単に説明します。セキュリティーの構成について詳しくは、「管理ガイド」および「プログラミング・ガイド」を参照してください。

### 分散セキュリティーの基礎

分散 eXtreme Scale セキュリティーは、次の 3 つの主要概念に基づいています。

#### 信頼できる認証

要求側の ID を判別する能力。WebSphere eXtreme Scale は、クライアントとサーバー間の認証も、サーバー相互間の認証もともにサポートします。

**許可** 要求側にアクセス権を付与する許可を与える能力。WebSphere eXtreme Scale は、さまざまな操作に対しさまざまな許可をサポートします。

#### セキュア・トランスポート

ネットワーク上での安全なデータ伝送。WebSphere eXtreme Scale は、Transport Layer Security/Secure Sockets Layer (TLS/SSL) プロトコルをサポートします。

### 認証

WebSphere eXtreme Scale は、分散クライアント・サーバー・フレームワークをサポートします。クライアント・サーバー・セキュリティー・インフラストラクチャーは、eXtreme Scale サーバーへのアクセスを安全にするために配置されています。例えば、認証が eXtreme Scale サーバーによって必要とされる場合、認証のための資格情報を eXtreme Scale クライアントがサーバーに提供する必要があります。これらの資格情報は、ユーザー名とパスワードのペア、クライアント証明書、Kerberos チケット、またはクライアントとサーバーが合意した形式で示されたデータなどです。

### 許可

WebSphere eXtreme Scale の許可は、サブジェクトおよびアクセス権に基づいています。Java 認証・承認サービス (JAAS) を使用してアクセスを許可したり、Tivoli® Access Manager (TAM) などのカスタム・アプローチを接続して許可を処理したりできます。クライアントまたはグループに対しては、以下の許可を与えることができます。

### マップ許可

マップに対して挿入、読み取り、更新、除去、または削除の操作を実行することを許可します。

### ObjectGrid 許可

ObjectGrid オブジェクトに対してオブジェクト照会またはエンティティー照会を実行することを許可します。

### DataGrid エージェント許可

DataGrid エージェントを ObjectGrid ヘデプロイすることを許可します。

### サーバー・サイド・マップ許可

サーバー・マップをクライアント・サイドに複製すること、またはサーバー・マップに動的索引を作成することを許可します。

### 管理許可

管理タスクを実行することを許可します。

## トランスポート・セキュリティ

クライアント・サーバー通信を保護するため、WebSphere eXtreme Scale は TLS/SSL をサポートします。これらのプロトコルは、eXtreme Scale クライアントとサーバー間のセキュア接続のための、認証性、保全性、および機密性を備えたトランスポート層セキュリティを提供します。

## グリッド・セキュリティ

セキュア環境では、サーバーは他のサーバーの認証性を確認できる必要があります。WebSphere eXtreme Scale は、この目的のために共有秘密ストリングのメカニズムを使用します。この秘密鍵のメカニズムは、共有パスワードと同様です。すべての eXtreme Scale サーバーは、共有秘密ストリングについて同意します。データ・グリッドに加わるサーバーは、秘密ストリングを提示するよう求められます。参加しようとするサーバーの秘密ストリングがマスター・サーバーのものと一致すると、そのサーバーはグリッドに参加できます。一致しない場合、結合要求は拒否されます。

平文の機密事項の送信は保護されません。eXtreme Scale セキュリティー・インフラストラクチャーには、サーバーがこの機密事項を送信前に保護できるようにするため、SecureTokenManager プラグインが用意されています。セキュア操作の実装方法を選択できます。WebSphere eXtreme Scale は、セキュア操作が実装され、機密事項が暗号化され署名されるような実装を提供します。

## 動的デプロイメント・トポロジーでの Java Management Extensions (JMX) セキュリティー

JMX MBean セキュリティーは、すべてのバージョンの eXtreme Scale でサポートされています。カタログ・サーバー MBean およびコンテナ・サーバー MBean のクライアントを認証可能にして、MBean 操作へのアクセスを実施できるようになります。

## ローカル eXtreme Scale セキュリティー

ローカル eXtreme Scale セキュリティーは、アプリケーションが ObjectGrid インスタンスを直接にインスタンス化して、使用するの、分散 eXtreme Scale モデルとは異なります。アプリケーションおよび eXtreme Scale インスタンスは、同じ Java 仮想マシン (JVM) 内にあります。このモデルにはクライアント/サーバーの概念が含まれていないので、認証はサポートされません。アプリケーションがそれ自身の認証を管理し、認証済みサブジェクト・オブジェクトを eXtreme Scale に渡す必要があります。ただし、ローカル eXtreme Scale プログラミング・モデルに使用される許可メカニズムは、クライアント/サーバー・モデルに使用されるものと同じです。

### 構成およびプログラミング

セキュリティーに関する構成とプログラミングについては、591 ページの『外部プロバイダーとのセキュリティー統合』およびセキュリティー API を参照してください。

---

## インストールの計画

製品をインストールする前に、ソフトウェア要件とハードウェア要件、および Java 環境の設定を検討する必要があります。

### ハードウェアおよびソフトウェア要件

ハードウェア要件およびオペレーティング・システム要件の概要をご覧ください。WebSphere eXtreme Scale に対して使用するハードウェアまたはオペレーティング・システムのレベルについて、特定のレベルの要件はありませんが、公式にサポートされるハードウェアおよびソフトウェアのオプションは、製品サポート・サイトの「システム要件」ページから入手できます。インフォメーション・センターの情報と「システム要件」ページの情報に違いがある場合は、Web サイトの情報を優先してください。インフォメーション・センターの前提条件の情報は、便宜上提供されているだけです。

ハードウェアおよびソフトウェア要件の正式なセットについては、システム要件ページを参照してください。

eXtreme Scale のインストールおよびデプロイ先として、オペレーティング・システムの特設レベルの要件はありません。Java Platform, Standard Edition (Java SE) および Java Platform, Enterprise Edition (Java EE) のそれぞれのインストールでは、異なるオペレーティング・システムのレベルまたはフィックスが必要です。

この製品は、Java EE および Java SE 環境にインストールしてデプロイできます。また、クライアント・コンポーネントを WebSphere Application Server に統合せず、直接 Java EE アプリケーションにバンドルすることができます。WebSphere eXtreme Scale は、Java SE 5 以降、および WebSphere Application Server バージョン 6.1 以降をサポートしています。



## ハードウェア要件

WebSphere eXtreme Scale では、ハードウェアの具体的なレベルの要件はありません。ハードウェア要件は、WebSphere eXtreme Scale を実行するのに使用される Java Platform, Standard Edition のインストール済み環境でサポートされるハードウェアによって異なります。eXtreme Scale を WebSphere Application Server または別の Java Platform, Enterprise Edition 実装環境で使用する場合は、これらのプラットフォームのハードウェア要件は WebSphere eXtreme Scale にとって十分です。

## オペレーティング・システム要件

**8.5+** WebSphere eXtreme Scale では、オペレーティング・システムの具体的なレベルの要件はありません。各 Java SE および Java EE 実装は、それぞれ異なるオペレーティング・システム・レベル、または、Java 実装のテスト中に発見された問題に対するフィックスを必要とします。これらの実装に必要なレベルは、eXtreme Scale にとって十分です。

### 8.5+

## Installation Manager の要件

WebSphere eXtreme Scale をインストールする前に、Installation Manager をインストールする必要があります。Installation Manager をインストールするには、製品メディアを使用するか、Passport Advantage<sup>®</sup> サイトから入手したファイルを使用するか、あるいは、IBM<sup>®</sup> Installation Manager ダウンロード Web サイトにある Installation Manager の最新バージョンが入っているファイルを使用します。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オフラインのインストール』を参照してください。

## Web ブラウザー要件

Web コンソールは、以下の Web ブラウザーをサポートしています。 **8.5+**

- Mozilla Firefox、バージョン 3.5.x 以降
- Microsoft Internet Explorer バージョン 7 以降

## WebSphere Application Server 要件

### 8.5+

- WebSphere Application Server バージョン 6.1.0.41 以降
- WebSphere Application Server バージョン 7.0.0.21 以降
- WebSphere Application Server バージョン 8.0.0.2 以降

詳しくは、WebSphere Application Server の推奨フィックスを参照してください。

## その他のアプリケーション・サーバー要件

その他の Java EE 実装は、ローカル・インスタンスとして、または、eXtreme Scale サーバーへのクライアントとして、eXtreme Scale ランタイムを使用できます。Java SE を実装する場合は、バージョン 5 以降を使用する必要があります。

## Java SE の考慮事項

WebSphere eXtreme Scale は Java SE 5、Java SE 6、または Java SE 7 を必要とします。一般に、Java SE は、バージョンが新しい方が機能性もパフォーマンスも優れています。

### サポートされるバージョン

WebSphere eXtreme Scale は Java SE 5、Java SE 6、および Java SE 7 と一緒に使用することができます。使用するバージョンは、Java ランタイム環境 (JRE) ベンダーによって現在サポートされているものでなければなりません。

IBM Runtime Environment, Java Technology Edition バージョン 5、バージョン 6、およびバージョン 7 は、製品と一緒に広く使用するためにサポートされています。バージョン 6 サービス・リリース 9 フィックスバック 2 は完全にサポートされる JRE です。この JRE は、スタンドアロン WebSphere eXtreme Scale および WebSphere eXtreme Scale クライアント インストールの一部として `wxs_install_root/java` ディレクトリにインストールされるもので、クライアントとサーバーの両方で使用することができます。WebSphere Application Server 内に WebSphere eXtreme Scale をインストールする場合は、WebSphere Application Server インストールに含まれている JRE を使用できます。Web コンソールの場合は、IBM Runtime Environment, Java Technology Edition バージョン 6 サービス・リリース 7 およびそれ以降のサービス・リリースのみを使用する必要があります。

WebSphere eXtreme Scale は、Java Development Kit (JDK) バージョン 5、バージョン 6、およびバージョン 7 の機能を、これが使用可能になったときに使用します。一般に、Java Development Kit (JDK) および Java SE は、バージョンが新しい方がパフォーマンスおよび機能が優れています。

詳しくは、サポートされるソフトウェアを参照してください。

### Java に依存している WebSphere eXtreme Scale フィーチャー

表 2. Java SE 5、Java SE 6、および Java SE 7 を必要とするフィーチャー:

WebSphere eXtreme Scale は、Java SE 5 または Java SE 6 で導入された機能を使用して、以下の製品フィーチャーを提供します。

フィーチャー	Java SE 5 以降のサービス・リリースでサポートされる	Java SE バージョン 6、バージョン 7 以降のサービス・リリースでサポートされる
EntityManager API アノテーション (オプション: XML ファイルも使用できます)	X	X
Java Persistence API (JPA): JPA ローダー、JPA クライアント・ローダー、および JPA 時間ベース・アップデーター	X	X
メモリー・ベース除去 (MemoryPoolMXBean を使用)	X	X



表 2. Java SE 5、Java SE 6、および Java SE 7 を必要とするフィーチャー (続き):

WebSphere eXtreme Scale は、Java SE 5 または Java SE 6 で導入された機能を使用して、以下の製品フィーチャーを提供します。

フィーチャー	Java SE 5 以降のサービス・リリースでサポートされる	Java SE バージョン 6、バージョン 7 以降のサービス・リリースでサポートされる
インストールメンテーション・エージェント: <ul style="list-style-type: none"> <li>• wxssizeagent.jar: 使用されるバイト・マップ・メトリックの精度を上げます。</li> <li>• ogagent.jar: フィールド・アクセス・エンティティのパフォーマンスを向上させます。</li> </ul>	X	X
モニター用 Web コンソール		X

## Java EE の考慮事項

WebSphere eXtreme Scale を Java Platform, Enterprise Edition 環境に統合する準備をするときは、バージョン、構成オプション、要件と制約、およびアプリケーションのデプロイメントと管理などを考慮します。

### Java EE 環境での eXtreme Scale アプリケーションの実行

Java EE アプリケーションは、eXtreme Scale のリモート・アプリケーションに接続できます。さらに、WebSphere Application Server 環境は、アプリケーションがアプリケーション・サーバーで開始するときに eXtreme Scale サーバーの始動をサポートします。

ObjectGrid インスタンスの作成に XML ファイルを使用する場合、かつ XML ファイルがエンタープライズ・アーカイブ (EAR) ファイルのモジュール内にある場合、`getClass().getClassLoader().getResource("META-INF/objGrid.xml")` メソッドを使用してファイルにアクセスし、ObjectGrid インスタンスの作成に使用する URL オブジェクトを取得してください。メソッド呼び出しで使用している XML ファイルの名前に置き換えます。

アプリケーションの開始 Bean を使用して、アプリケーションが起動する際に ObjectGrid インスタンスをブートストラップし、アプリケーションが停止する際にそのインスタンスを破棄することができます。開始 Bean は、`com.ibm.websphere.startupservice.AppStartupHome` リモート・ロケーションと `com.ibm.websphere.startupservice.AppStartup` リモート・インターフェースを持つ Stateless Session Bean です。リモート・インターフェースには `start` メソッドと `stop` メソッドという 2 つのメソッドがあります。`start` メソッドを使用してインスタンスをブートストラップし、`stop` メソッドを使用してインスタンスを破棄します。アプリケーションは `ObjectGridManager.getObjectGrid` メソッドを使用して、インスタンスへの参照を保持します。詳しくは、「プログラミング・ガイド」の

ObjectGridManager を使用した ObjectGrid へのアクセスに関する情報を参照してください。

## クラス・ローダーの使用

別のクラス・ローダーを使用するアプリケーション・モジュールが Java EE アプリケーションの単一 ObjectGrid インスタンスを共有する場合、eXtreme Scale に保管されるオブジェクトと製品のプラグインがアプリケーションの共通ローダーにあることを確認してください。

## サーブレット内の ObjectGrid インスタンスのライフサイクルの管理

サーブレットで ObjectGrid インスタンスのライフサイクルを管理するためには、init メソッドを使用してインスタンスを作成したり、destroy メソッドを使用してインスタンスを除去することができます。インスタンスがキャッシュされた場合、サーブレット・コードで検索および操作を行います。詳しくは、「プログラミング・ガイド」の ObjectGridManager インターフェースを使用した ObjectGrid へのアクセスに関する情報を参照してください。

## ディレクトリー規則

`wxs_install_root` や `wxs_home` など、参照が必要な特別のディレクトリーに対して、資料全体で、次のディレクトリー規則が使用されます。インストール中、およびコマンド行ツールの使用時も含めて、さまざまなシナリオで、これらのディレクトリーにアクセスします。

### `wxs_install_root`

`wxs_install_root` ディレクトリーは、WebSphere eXtreme Scale 製品ファイルがインストールされているルート・ディレクトリーです。`wxs_install_root` ディレクトリーは、試用版のアーカイブが解凍されたディレクトリー、または WebSphere eXtreme Scale 製品がインストールされているディレクトリーの可能性があります。

- 試用版を解凍した場合の例:

**例:** `/opt/IBM/WebSphere/eXtremeScale`

- WebSphere eXtreme Scale がスタンドアロン・ディレクトリーにインストールされている場合の例:

**UNIX** **例:** `/opt/IBM/eXtremeScale`

**Windows** **例:** `C:¥Program Files¥IBM¥WebSphere¥eXtremeScale`

- WebSphere eXtreme Scale が WebSphere Application Server に統合されている場合の例:

**例:** `/opt/IBM/WebSphere/AppServer`

### `wxs_home`

`wxs_home` ディレクトリーは、WebSphere eXtreme Scale 製品ライブラリー、サンプル、およびコンポーネントのルート・ディレクトリーです。このディレクトリーは、試用版を解凍した場合は、`wxs_install_root` ディレクトリーと同じです。スタンドアロンのインストール済み環境の場合、`wxs_home` ディレクトリー

は、`wxs_install_root` ディレクトリー内の `ObjectGrid` サブディレクトリーです。WebSphere Application Server に統合されているインストール済み環境の場合、このディレクトリーは、`wxs_install_root` ディレクトリー内の `optionalLibraries/ObjectGrid` ディレクトリーです。

- 試用版を解凍した場合の例:

例: `/opt/IBM/WebSphere/eXtremeScale`

- WebSphere eXtreme Scale がスタンドアロン・ディレクトリーにインストールされている場合の例:

**UNIX** 例: `/opt/IBM/eXtremeScale/ObjectGrid`

**Windows** 例: `wxs_install_root¥ObjectGrid`

- WebSphere eXtreme Scale が WebSphere Application Server に統合されている場合の例:

例: `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid`

#### **was\_root**

`was_root` ディレクトリーは、WebSphere Application Server インストール済み環境のルート・ディレクトリーです。

例: `/opt/IBM/WebSphere/AppServer`

#### **restservice\_home**

`restservice_home` ディレクトリーは、WebSphere eXtreme Scale REST データ・サービスのライブラリーおよびサンプルが配置されるディレクトリーです。このディレクトリーは `restservice` という名前で、`wxs_home` ディレクトリー内のサブディレクトリーです。

- スタンドアロン・デプロイメントの場合の例:

例: `/opt/IBM/WebSphere/eXtremeScale/ObjectGrid/restservice`

例: `wxs_home\restservice`

- WebSphere Application Server 統合デプロイメントの場合の例:

例: `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid/restservice`

#### **tomcat\_root**

`tomcat_root` は、Apache Tomcat インストール済み環境のルート・ディレクトリーです。

例: `/opt/tomcat5.5`

#### **wasce\_root**

`wasce_root` は、WebSphere Application Server Community Edition インストール済み環境のルート・ディレクトリーです。

例: `/opt/IBM/WebSphere/AppServerCE`

### java\_home

*java\_home* は、Java Runtime Environment (JRE) インストール済み環境のルート・ディレクトリーです。

**UNIX** 例: /opt/IBM/WebSphere/eXtremeScale/java

**Windows** 例: *wxs\_install\_root*\%java

### samples\_home

*samples\_home* は、チュートリアルに使用するサンプル・ファイルを解凍したディレクトリーです。

**UNIX** 例: *wxs\_home*/samples

**Windows** 例: *wxs\_home*\%samples

### dvd\_root

*dvd\_root* ディレクトリーは、製品が含まれた DVD のルート・ディレクトリーです。

例: *dvd\_root*/docs/

### equinox\_root

*equinox\_root* ディレクトリーは、Eclipse Equinox OSGi フレームワークのインストール済み環境のルート・ディレクトリーです。

例: /opt/equinox

### user\_home

*user\_home* ディレクトリーは、ユーザー・ファイル (セキュリティー・プロファイルなど) が保管されている場所です。

**Windows** c:\%Documents and Settings%\%*user\_name*

**UNIX** /home/*user\_name*

---

## 環境キャパシティーの計画

初期データ・セット・サイズおよび予測されるデータ・セット・サイズがわかっている場合、WebSphere eXtreme Scale を実行するために必要なキャパシティーを計画できます。これらの計画の策定を使用して、将来の変更に向けて WebSphere eXtreme Scale を効率よくデプロイし、データ・グリッドの柔軟性を最大限にすることができます。このような利点は、メモリー内のデータベースや他のタイプのデータベースなど、異なるシナリオでは実現されません。

## メモリー・サイズ設定および区画数の計算

特定の構成に必要なメモリーの容量および区画数を計算できます。

**重要:** このトピックは、COPY\_TO\_BYTES コピー・モードを使用していないときに当てはまります。COPY\_TO\_BYTES モードを使用している場合は、メモリー・サイズがはるかに小さくなり、計算手順が異なります。このモードについて詳しくは、コピー・モードのチューニングを参照してください。

WebSphere eXtreme Scale は、データを Java 仮想マシン (JVM) のアドレス・スペースに保管します。各 JVM は、JVM に保管されたデータの作成、取得、更新、および削除の呼び出しをサービスするためのプロセッサ・スペースを提供します。さらに、各 JVM は、データ・エンタリーおよびレプリカ用のメモリー・スペースを提供します。Java オブジェクトのサイズはさまざまなので、必要なメモリー量を見積もるための測定が必要です。

必要なメモリーのサイズを設定するには、アプリケーション・データを 1 つの JVM にロードします。ヒープ使用量が 60% に達している場合、使用されるオブジェクトの数に注意してください。この数値は、Java 仮想マシンのそれぞれの推奨されるオブジェクトの最大数です。最も確かなサイズ設定を行うには、現実に近いデータを使用します。索引もまたメモリーを消費するので、定義されているすべての索引をサイズ設定に含めてください。メモリー使用量のサイズ見積もりを行う最良の方法は、ガーベッジ・コレクション **verbosegc** 出力を実行することです。この出力によって、ガーベッジ・コレクション後の数値が分かるからです。ヒープ使用量については MBean またはプログラムでいつでも照会できますが、そういった照会ではヒープの現行スナップショットしか取得できません。このスナップショットには未収集のガーベッジが含まれている可能性があるため、この方法では消費メモリーは正確には示されません。

## 構成の拡張

### 区画当たりの断片数 (numShardsPerPartition 値)

区画当たりの断片数である numShardsPerPartition 値を計算するには、プライマリー断片の 1 に、必要なレプリカ断片の総数を加算します。区画化について詳しくは、区画化を参照してください。

```
numShardsPerPartition = 1 + total_number_of_replicas
```

### Java 仮想マシン 数 (minNumJVMs 値)

構成を拡張するには、まず保管する必要のある合計オブジェクトの最大数を決定します。必要な Java 仮想マシンの数を決めるには、次の式を使用します。

```
minNumJVMs=(numShardsPerPartition * numObjs) / numObjsPerJVM
```

この値を切り上げて、最も近い整数値にしてください。

### 断片数 (numShards 値)

最終的な増加サイズでは、それぞれの JVM に 10 個の断片を使用します。前述したように、各 JVM には、1 つのプライマリー断片と (N-1) 個のレプリカ断片があります。この場合、9 個のレプリカがあります。データ保管用に既に多数の Java 仮想マシンがあるため、Java 仮想マシンの数に 10 を掛けて、断片の数を決定することができます。

```
numShards = minNumJVMs * 10 shards/JVM
```

### 区画数

区画に 1 つのプライマリー断片と 1 つのレプリカ断片がある場合、区画には 2 つの断片 (プライマリーとレプリカ) があります。区画数は、断片数を 2 で割って、

一番近い素数に丸めたものです。区画に 1 つのプライマリーと 2 つのレプリカがある場合、区画数は、断片数を 3 で割って、一番近い素数に丸めたものになります。

```
numPartitions = numShards / numShardsPerPartition
```

## 拡張の例

この例では、エントリー数は当初 250,000,000 であるとしてます。毎年、エントリー数は 14% 増加します。7 年後には、エントリーの合計数が 500,000,000 となるため、それに応じた容量計画が必要になります。高可用性を実現するため、1 つのレプリカが必要になります。レプリカを使用すると、エントリー数は 2 倍、すなわち 1,000,000,000 となります。テストとして、2,000,000 のエントリーを各 JVM に保管できます。このシナリオの計算を使用すると、以下の構成が必要になります。

- 最終的な数のエントリーを保管するために 500 の Java 仮想マシン。
- 5000 の断片 (Java 仮想マシン数の 500 に 10 を掛けたもの)。
- 2500 の区画、すなわち次位の素数である 2503 (5000 の断片を 2 (プライマリー断片と複製断片) で割ったもの)。

## 構成の開始

前述の計算に基づいて、250 の Java 仮想マシンから始めて、5 年間で 500 の Java 仮想マシンに増やします。この構成を使用して、最終的なエントリー数に達するまで段階的な増加を管理できます。

この構成では、区画ごとに約 200,000 (500,000,000 のエントリーを区画数の 2503 で割ったもの) のエントリーが保管されます。numberOfBuckets パラメーターを次位の素数 (この例では 70887) までのエントリーを収めるマップで設定します。これにより約 3 の比率が保たれます。

## Java 仮想マシンの最大数に達した場合

500 という Java 仮想マシンの最大数に達しても、データ・グリッドを拡張できません。Java 仮想マシンの数が 500 を超えるまでになると、各 JVM について断片数が 10 (推奨数) を下回り始めます。つまり断片が大きくなり始めます。これは問題となる場合があります。将来の成長を考慮しながら、サイズ設定処理を繰り返し、区画数を設定し直します。この作業では、データ・グリッド全体の再始動が必要になります。さもないとデータ・グリッド不足となります。

## サーバー数

**重要:** どのような場合にも、サーバーについてはページングは使用しないでください。

1 つの JVM は、ヒープ・サイズを超えるメモリーを使用します。例えば、JVM の 1 GB のヒープでは、実際には 1.4 GB の実メモリーが使用されます。サーバー上の使用可能な空き RAM を判別してください。RAM の容量を JVM あたりのメモリーで割って、サーバー上の Java 仮想マシンの最大数を計算してください。



## トランザクションの区画ごとの CPU 見積もり

eXtreme Scale の主要な機能は、その弾力的なスケーリングですが、拡大のための CPU のサイズ変更の考慮および理想的な CPU 数の調整も重要です。

プロセッサのコストには、以下が含まれます。

- クライアントからの作成、取得、更新、および削除操作にサービスを提供するコスト
- 他の Java 仮想マシンのレプリカ生成のコスト
- 無効化のコスト
- 除去ポリシーのコスト
- ガーベッジ・コレクションのコスト
- アプリケーション・ロジックのコスト
- シリアライゼーションのコスト

### サーバーごとの Java 仮想マシン

サーバーは 2 台使用し、サーバーごとに最大の JVM 数を開始します。前のセクションで計算した区画数を使用します。その後、それら 2 台のコンピューターに収まるだけの十分なデータと Java 仮想マシンをプリロードします。クライアントには別のサーバーを使用してください。この 2 台のサーバーからなるデータ・グリッドに対し、現実的なトランザクション・シミュレーションを実行します。

ベースラインを計算するには、プロセッサ使用が飽和状態になるようにしてください。プロセッサを飽和状態にできない場合は、ネットワークが飽和している可能性があります。ネットワークが飽和している場合は、ネットワーク・カードをさらに追加し、複数のネットワーク・カードで Java 仮想マシンをラウンドロビンさせてください。

プロセッサ使用量 60% でコンピューターを実行し、作成、取得、更新、および削除トランザクションの速度を測定します。この測定により、2 台のサーバーでのスループットがわかります。この数値は、サーバー 4 台で倍になり、サーバー 8 台でさらにその倍になり、以降も同様の割合で大きくなります。この拡張の前提は、ネットワーク容量とクライアントのキャパシティーも同様に拡大可能であることです。

結果的に、eXtreme Scale 応答時間は、サーバーの数が増しても安定しています。トランザクション・スループットは、データ・グリッドにコンピューターが追加されるにつれて直線的に増加します。

## 並列トランザクションの場合の CPU のサイズ設定

データ・グリッドが拡張するにつれ、単一区画トランザクションのスループットが直線的に増加します。並列トランザクションは、サーバーの集合 (これはすべてのサーバーの可能性がありますが) にタッチするので、単一区画トランザクションとは異なります。

トランザクションがすべてのサーバーにタッチする場合、スループットは、トランザクションを開始したクライアントまたはタッチされた最低速のサーバーのスループットに制限されます。データ・グリッドが大きくなれば、データはより広く分散



され、提供されるプロセッサ・スペース、メモリー、ネットワークなどが拡張されます。ただし、クライアントは最低速のサーバーの応答を待たなければならず、クライアントはトランザクションの結果を消費しなければならなくなります。

トランザクションがサーバーのサブセットにタッチする場合、 $N$  個のサーバーのうち  $M$  個が要求を受け取ります。この結果、スループットは、最低速のサーバーのスループットより、 $N/M$  倍した分だけ速くなります。例えば、20 のサーバーがある場合に、トランザクションが 5 つのサーバーにタッチすると、スループットは、データ・グリッド内の最低速のサーバーのスループットを 4 倍したものになります。

並列トランザクションが完了すると、結果がそのトランザクションを開始したクライアント・スレッドに送信されます。ここでこのクライアントは、単一スレッド化された結果を集約する必要があります。トランザクションでタッチされるサーバーの数が増えると、この集約時間が増加します。ただし、データ・グリッドが拡大すると、各サーバーが戻す結果が小さくなる可能性もあるので、この時間はアプリケーションに依存します。

一般的には、グリッド全体で区画が均等に配分されるので、並列トランザクションはデータ・グリッド内のすべてのサーバーにタッチします。この場合、スループットは、最初の事例に制限されます。

## 要約

このサイズ設定により、以下の 3 つのメトリックが得られます。

- 区画の数。
- 必須メモリーに必要なサーバーの数。
- 必須スループットに必要なサーバーの数。

メモリー所要量に対して 10 個のサーバーが必要であるが、プロセッサが飽和状態であるため必要とするスループットの 50% しか得られない場合、2 倍の数のサーバーが必要になります。

最高の安定度を実現するために、60% のプロセッサ負荷でサーバー、さらに 60% のヒープ負荷で JVM ヒープを稼働してください。スパイクでプロセッサ使用量を 80% から 90% の間に引き上げることができますが、通常はこのレベルより高いレベルでサーバーを稼働しないようにしてください。

## 動的キャッシュのキャパシティー・プランニング

WebSphere Application Server にデプロイされた Java EE アプリケーションでは、動的キャッシュ API を使用できます。動的キャッシュを使用して、ビジネス・データや生成された HTML をキャッシュに入れたり、データ複製サービス (DRS) を使用してセル内のキャッシュ・データを同期化したりすることができます。

### 概要

WebSphere eXtreme Scale 動的キャッシュ・プロバイダーで作成されたすべての動的キャッシュ・インスタンスは、デフォルトで、高い可用性を持ちます。高可用性のレベルおよびメモリー・コストは、使用されるトポロジーによって変わります。

組み込みトポロジーを使用している場合、キャッシュ・サイズは、1つのサーバー・プロセス内の空きメモリーの量に制限され、各サーバー・プロセスはキャッシュの完全コピーを保管します。1つのサーバー・プロセスが実行し続けている限り、キャッシュは存続します。キャッシュ・データが失われるのは、キャッシュにアクセスするすべてのサーバーがシャットダウンされた場合のみです。

組み込み区画化トポロジーを使用するキャッシングの場合、キャッシュ・サイズの上限は、すべてのサーバー・プロセス内にある空きスペースの総計です。デフォルトでは、eXtreme Scale 動的キャッシュ・プロバイダーは各プライマリー断片ごとに1つのレプリカを使用します。したがって、各キャッシュ・データは2回ずつ保管されます。

組み込み区画化キャッシュの容量を判定するには、以下の式 A を使用してください。

#### 式 A

$$F * C / (1 + R) = M$$

各部の意味は、次のとおりです。

- F = コンテナ・プロセス当たりの空きメモリー
- C = コンテナの数
- R = レプリカの数
- M = キャッシュの合計サイズ

各プロセスに 256 MB の使用可能なスペースがあり、サーバー・プロセスが合計 4 つある WebSphere Application Server Network Deployment データ・グリッドの場合、それらの全サーバーにまたがるキャッシュ・インスタンスは 512 メガバイトまでのデータを保管できます。このモードでは、キャッシュは、1つのサーバーが破損しても、データを失うことなく存続できます。また、最大 2 つのサーバーが順次シャットダウンしても、データを失うことはありません。この例では、上の式は次のようになります。

$$256\text{mb} * 4 \text{ コンテナ} / (1 \text{ プライマリー} + 1 \text{ レプリカ}) = 512\text{mb}$$

リモート・トポロジーを使用するキャッシュのサイジング特性は、組み込み区画化トポロジーを使用するキャッシュと似ていますが、すべての eXtreme Scale コンテナ・プロセス内の使用可能なスペースの総量に制限されます。

リモート・トポロジーでは、レプリカを増やすことによって、メモリーのオーバーヘッドが余分にかかる代わりにアベイラビリティのレベルを上げることが可能です。これは大部分の動的キャッシュ・アプリケーションでは不必要ですが、`dynacache-remote-deployment.xml` ファイルを編集してレプリカを増やすことができます。

以下の式 B と C を使用して、キャッシュの高可用性のためにレプリカを追加することの影響を判定できます。

#### 式 B

$$N = \text{Minimum}(T - 1, R)$$

各部の意味は、次のとおりです。

- N = 同時に破損してもかまわないプロセスの数
- T = コンテナの総数
- R = レプリカの総数

## 式 C

$$\text{Ceiling}(T / (1+N)) = m$$

各部の意味は、次のとおりです。

- T = コンテナの総数
- N = レプリカの総数
- m = キャッシュ・データをサポートするのに必要な最小コンテナ数

動的キャッシュ・プロバイダーでのパフォーマンス・チューニングについては、574 ページの『動的キャッシュ・プロバイダーのチューニング』を参照してください。

## キャッシュのサイズ見積もり

WebSphere eXtreme Scale 動的キャッシュ・プロバイダーを使用するアプリケーションをデプロイする前に、前のセクションに記述されている一般的な規則に、実動システムの環境データを組み合わせて検討する必要があります。まず最初に確定する必要がある数値は、コンテナ・プロセスの総数と、キャッシュ・データを保持するための各プロセス内の使用可能メモリーの量です。組み込みトポロジーを使用している場合、キャッシュ・コンテナは WebSphere Application Server プロセスの内部の同一場所に配置され、キャッシュを共有する各サーバーごとにコンテナが 1 つずつある状態になります。キャッシングを使用可能にしていないアプリケーションと WebSphere Application Server のメモリー・オーバーヘッドを判定することが、プロセス内で使用可能なスペース量を計算する最良の方法です。これは、詳細ガーベッジ・コレクション・データを分析することで行えます。リモート・トポロジーを使用している場合、この情報は、新しく開始され、キャッシュ・データがまだ設定されたことのないスタンドアロン・コンテナの、詳細ガーベッジ・コレクション出力を見れば分かります。キャッシュ・データ用に使用可能な、プロセス当たりのスペース量を計算する際には、ガーベッジ・コレクション用にいくらかのヒープ・スペースを確保しておくことにも注意が必要です。コンテナ (WebSphere Application Server またはスタンドアロン) のオーバーヘッドに、キャッシュ用に予約されたサイズを加えた結果は、合計ヒープの 70 % 以下になっているべきです。

この情報を収集できたら、前述の式 A に値を挿入して、区画化されたキャッシュの最大サイズを判定してください。最大サイズが判明したら、次のステップは、サポートできるキャッシュ・エントリー総数を判定することです。これには、キャッシュ・エントリー当たりの平均サイズを決定することが必要です。これを行う簡単な方法は、カスタマー・オブジェクトのサイズに 10% 追加することです。動的キャッシュを使用している場合のキャッシュ・エントリーのサイズ見積もりについて詳しくは、動的キャッシュおよびデータ・レプリカ生成サービスのチューニング・ガイドを参照してください。

圧縮が有効にされている場合、圧縮はカスタマー・オブジェクトのサイズには影響しますが、キャッシング・システムのオーバーヘッドには影響しません。圧縮を使用している場合のキャッシュ・オブジェクトのサイズを判定するには、以下の式を使用します。

$$S = O * C + O * 0.10$$

各部の意味は、次のとおりです。

- S = キャッシュ・オブジェクトの平均サイズ
- O = 圧縮されていないカスタマー・オブジェクトの平均サイズ
- C = 分数で表した圧縮率

つまり、2 を 1 にする場合の圧縮率は  $1/2 = 0.50$  です。これは小さいほど良い値です。保管されるオブジェクトが通常の POJO で、大部分がプリミティブ型の場合、圧縮率を 0.60 から 0.70 に想定してください。キャッシュされるオブジェクトが、サーブレット、JSP、または WebServices オブジェクトの場合、圧縮率を判定する最適の方法は、代表的なサンプルを ZIP 圧縮ユーティリティで圧縮することです。これが不可能な場合、このタイプのデータの一般的な圧縮率は 0.2 から 0.35 です。

次に、この情報を使用して、サポートできるキャッシュ・エントリーの総数を判定します。以下の式 D を使用してください。

#### 式 D

$$T = S / A$$

各部の意味は、次のとおりです。

- T = キャッシュ・エントリーの総数
- S = 式 A を使用して算出され、キャッシュ・データ用に使用可能な合計サイズ
- A = 各キャッシュ・エントリーの平均サイズ

最後に、動的キャッシュ・インスタンスにキャッシュ・サイズを設定して、この制限を強制する必要があります。WebSphere eXtreme Scale 動的キャッシュ・プロバイダーは、この点で、デフォルトの動的キャッシュ・プロバイダーと異なります。以下の式を使用して、動的キャッシュ・インスタンスのキャッシュ・サイズに設定する値を判定してください。以下の式 E を使用してください。

#### 式 E

$$Cs = Ts / Np$$

各部の意味は、次のとおりです。

- Ts = キャッシュの合計目標サイズ
- Cs = 動的キャッシュ・インスタンスに設定するキャッシュ・サイズ設定値
- Np = 区画の数。デフォルトは 47 です。

キャッシュ・インスタンスを共有する各サーバーで、動的キャッシュ・インスタンスのサイズを、式 E で計算した値に設定してください。

---

## 第 3 章 チュートリアル



チュートリアルを使用することで、エンティティ・マネージャー、照会、およびセキュリティを含めた製品使用のシナリオを理解しやすくなります。

---

### チュートリアル: Java SE セキュリティーの構成

以下のチュートリアルにより、Java Platform, Standard Edition 環境で分散 eXtreme Scale 環境を作成できます。

#### 始める前に

分散 eXtreme Scale 構成の基本をよく理解している必要があります。

#### このタスクについて

このチュートリアルでは、カタログ・サーバー、コンテナ・サーバー、およびクライアントのすべてが Java SE 環境で実行されています。このチュートリアルの各ステップは直前のステップを踏まえて進行します。このステップを一つ一つ実行して、分散 eXtreme Scale を保護し、その保護された eXtreme Scale にアクセスするシンプルな Java SE アプリケーションを作成してください。

チュートリアルの開始

### Java SE セキュリティー・チュートリアル - ステップ 1

チュートリアルの残りの部分を実施には、単純な Java プログラムと 2 つの XML ファイルを作成してパッケージ化する必要があります。これらのファイルのセットは、accounting という名前の 1 つの ObjectGrid インスタンスと customer マップを含む、単純な ObjectGrid 構成を定義します。SimpleDP.xml ファイルは、1 つの区画と最小必要数がゼロ個のレプリカで構成される 1 つのマップ・セットのデプロイメント・ポリシーを特徴とします。

#### 手順

1. コマンド行ウィンドウで、`wxs_home` ディレクトリーに移動します。
2. `applib` というディレクトリーを作成します。
3. 次の `SimpleApp.java` クラスを作成してコンパイルします。

```
SimpleApp.java
// This sample program is provided AS IS and may be used, executed, copied and modified
// without royalty payment by customer
// (a) for its own instruction and study,
// (b) in order to develop applications designed to run with an IBM WebSphere product,
// either for customer's own internal use or for redistribution by customer, as part of such an
// application, in customer's own products.
// Licensed Materials - Property of IBM
// 5724-J34 (C) COPYRIGHT International Business Machines Corp. 2007-2009
package com.ibm.websphere.objectgrid.security.sample.guide;

import com.ibm.websphere.objectgrid.ClientClusterContext;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
```

```

import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;

public class SimpleApp {

    public static void main(String[] args) throws Exception {

        SimpleApp app = new SimpleApp();
        app.run(args);
    }

    /**
     * read and write the map
     * @throws Exception
     */
    protected void run(String[] args) throws Exception {
        ObjectGrid og = getObjectGrid(args);

        Session session = og.getSession();

        ObjectMap customerMap = session.getMap("customer");

        String customer = (String) customerMap.get("0001");

        if (customer == null) {
            customerMap.insert("0001", "fName lName");
        } else {
            customerMap.update("0001", "fName lName");
        }
        customer = (String) customerMap.get("0001");
        // Close the session (optional in Version 7.1.1 and later) for improved performance
        session.close();
        System.out.println("The customer name for ID 0001 is " + customer);
    }

    /**
     * Get the ObjectGrid
     * @return an ObjectGrid instance
     * @throws Exception
     */
    protected ObjectGrid getObjectGrid(String[] args) throws Exception {
        ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();

        // Create an ObjectGrid
        ClientClusterContext ccContext = ogManager.connect("localhost:2809", null, null);
        ObjectGrid og = ogManager.getObjectGrid(ccContext, "accounting");

        return og;
    }
}

```

4. このファイルを使用してパッケージをコンパイルし、JAR に `sec_sample.jar` という名前を付けます。
5. `wxs_home` ディレクトリに移動し、`xml` というディレクトリを作成します。
6. `wxs_home/xml` ディレクトリで、以下の構成ファイルを作成します。

#### SimpleApp.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="accounting">
            <backingMap name="customer" readOnly="false" copyKey="true"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>

```

下記の XML ファイルはデプロイメント環境を構成します。

#### SimpleDP.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

```



```

<objectgridDeployment objectgridName="accounting">
  <mapSet name="mapSet1" numberOfPartitions="1" minSyncReplicas="0" maxSyncReplicas="2"
    maxAsyncReplicas="1">
    <map ref="customer"/>
  </mapSet>
</objectgridDeployment>
</deploymentPolicy>

```

## タスクの結果

これらのファイルは、accounting という名前の 1 つの ObjectGrid インスタンスと customer マップを含む、単純な ObjectGrid 構成を作成します。

## Java SE セキュリティ・チュートリアル - ステップ 2

SimpleApp.java サンプルが実行することを確認する前に、カタログ・サーバーとコンテナ・サーバーを開始する必要があります。これらのサービスを正常に開始した後、クライアントを起動し、サンプルを実行することができます。また、利用可能な統合セキュリティを強化するため、このチュートリアルのステップごとにセキュリティ機能を順次追加していきます。

### 始める前に

チュートリアルのこのステップを正常に完了するには、以下のファイルにアクセスできなければなりません。

- コンパイルされた sec\_sample.jar パッケージにアクセスできるようにします。このパッケージには SimpleApp.java プログラムが含まれています。
- 必要な構成ファイル SimpleApp.xml および SimpleDP.xml にアクセスできるようにします。

これらのファイルは、このチュートリアルの 73 ページの『Java SE セキュリティ・チュートリアル - ステップ 1』で作成したはずです。

また、以下のことを行う方法も知っているはずです。

- カタログ・サービスの開始と停止。カタログ・サービスの開始について詳しくは、451 ページの『スタンドアロン・カタログ・サービスの開始』を参照してください。
- コンテナ・サービスの開始と停止。コンテナ・サービスの開始について詳しくは、454 ページの『コンテナ・サーバーの始動』を参照してください。
- グリッドに挿入されたマップ・サイズを確認するための **xcmd** ユーティリティの実行。

### 手順

1. コマンド行ウィンドウで、`wxs_home/bin` ディレクトリーに移動し、カタログ・サービスを開始します。

• **UNIX** **Linux** `startOgServer.sh catalogServer`

• **Windows** `startOgServer.bat catalogServer`

2. `c0` という名前のコンテナ・サービスを開始します。

• **UNIX** **Linux** `startOgServer.sh c0 -objectGridFile  
 ../xml/SimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml  
 -catalogServiceEndpoints localhost:2809`



- **Windows** startOgServer.bat c0 -objectGridFile ../xml/SimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809
3. カタログ・サーバーとコンテナ・サーバーが始動されたならば、次のようにして sec\_sample.jar サンプルを実行します。 java -classpath ../lib/objectgrid.jar;../applib/sec\_sample.jar com.ibm.websphere.objectgrid.security.sample.guide.SimpleApp サンプルの出力: The customer name for ID 0001 is fName lName このクラスの getObjectGrid メソッドは ObjectGrid を取得し、run メソッドは customer マップからレコードを読み取り、accounting グリッドの値を更新します。
4. 次のように **xscmd** コマンド・ユーティリティを実行して、「accounting」グリッドに挿入された「customer」マップのサイズを確認します。
- **UNIX** **Linux** xscmd.sh -c showMapSizes -g accounting -ms mapSet1
  - **Windows** xscmd.bat -c showMapSizes -g accounting -ms mapSet1
5. 次のいずれかのスクリプトを使用して、c0 という名前のコンテナ・サーバーを停止します。
- **UNIX** **Linux** stopOgServer.sh c0 -catalogServiceEndPoints localhost:2809
  - **Windows** stopOgServer.bat c0 -catalogServiceEndPoints localhost:2809
- サーバーが正常に停止した場合は、次のメッセージが表示されます。
- CW0BJ2512I: ObjectGrid server c0 stopped.
6. 次のいずれかのスクリプトを使用してカタログ・サーバーを停止します。
- **UNIX** **Linux** stopOgServer.sh catalogServer -catalogServiceEndPoints localhost:2809
  - **Windows** stopOgServer.bat catalogServer -catalogServiceEndPoints localhost:2809
- サーバーが正常に停止した場合は、次のメッセージが表示されます。
- CW0BJ2512I: ObjectGrid server catalogServer stopped.

## Java SE セキュリティー・チュートリアル - ステップ 3

チュートリアルの残りの部分は、eXtreme Scale サーバーに接続する前にクライアント認証を有効にする方法を示しています。このチュートリアルの次のステップに備えるために、SecureSimpleApp.java プログラムを JAR にパッケージ化し、構成ファイルのセットを作成します。これらの構成ファイルは、security.xml ファイルと 2 つの JAAS 構成ファイルを含みます。security.xml ファイルは、認証を環境に書き込めるようにします。JAAS 構成ファイルは、サーバーへの接続時に認証メカニズムを提供します。

## 手順

1. コマンド行ウィンドウで、73 ページの『Java SE セキュリティー・チュートリアル - ステップ 1』で作成した `wxs_home/applib` ディレクトリーに移動します。
2. 次の `SecureSimpleApp.java` クラスを作成してコンパイルします。

```
SecureSimpleApp.java
package com.ibm.websphere.objectgrid.security.sample.guide;

import com.ibm.websphere.objectgrid.ClientClusterContext;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory;
import com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator;
import com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator;

public class SecureSimpleApp extends SimpleApp {

    public static void main(String[] args) throws Exception {

        SecureSimpleApp app = new SecureSimpleApp();
        app.run(args);
    }

    /**
     * Get the ObjectGrid
     * @return an ObjectGrid instance
     * @throws Exception
     */
    protected ObjectGrid getObjectGrid(String[] args) throws Exception {
        ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
        ogManager.setTraceFileName("logs/client.log");
        ogManager.setTraceSpecification("ObjectGrid*all=enabled:ORBRas=all=enabled");

        // Creates a ClientSecurityConfiguration object using the specified file
        ClientSecurityConfiguration clientSC = ClientSecurityConfigurationFactory
            .getClientSecurityConfiguration(args[0]);

        // Creates a CredentialGenerator using the passed-in user and password.
        CredentialGenerator credGen = new UserPasswordCredentialGenerator(args[1], args[2]);
        clientSC.setCredentialGenerator(credGen);

        // Create an ObjectGrid by connecting to the catalog server
        ClientClusterContext ccContext = ogManager.connect("localhost:2809", clientSC, null);
        ObjectGrid og = ogManager.getObjectGrid(ccContext, "accounting");

        return og;
    }
}
```

3. これらのファイルを使用してパッケージをコンパイルし、JAR に `sec_sample.jar` という名前を付けます。
4. `wxs_home` ディレクトリーに切り替えます。
5. `security` というディレクトリーを作成します。
6. `security.xml` という構成ファイルを作成します。このファイルにはサーバー・セキュリティ・プロパティーが指定されます。これらのプロパティーは、カタログ・サーバーとコンテナ・サーバーの両方に共通します。

```
security.xml
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config/security">

    <security securityEnabled="true" loginSessionExpirationTime="300" >

        <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.
            KeyStoreLoginAuthenticator">
```

```
        </authenticator>
    </security>

</securityConfig>
```

## Java SE セキュリティー・チュートリアル - ステップ 4

前のステップに基づいて、以下のトピックでは、分散 eXtreme Scale 環境でクライアント認証を実装する方法を示します。

### 始める前に

76 ページの『Java SE セキュリティー・チュートリアル - ステップ 3』を完了していなければなりません。SecureSimpleApp.java サンプルの作成および sec\_sample.jar ファイルへのコンパイル、ならびに security.xml という構成ファイルの作成が完了していなければなりません。

### このタスクについて

クライアント認証が有効になっていると、クライアントは eXtreme Scale サーバーに接続する前に認証されます。このセクションでは、eXtreme Scale サーバー環境でどのようにすればクライアント認証が可能であるかを、サンプル SecureSimpleApp.java を使用して実証します。**クライアント資格情報**

SecureSimpleApp.java サンプルでは、次の 2 つのプラグイン実装を使用してクライアント資格情報を取得します。

```
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredential
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
```

これらのプラグインについて詳しくは、クライアント認証プログラミングを参照してください。

### サーバー・オーセンティケーター

この例では、テストとサンプルが目的である eXtreme Scale 組み込み実装 KeyStoreLoginAuthenticator を使用します (鍵ストアは単純なユーザー・レジストリーであり、実動には使用しないようにしてください)。詳しくは、クライアント認証プログラミングのオーセンティケーター・プラグインについてのトピックを参照してください。

### 手順

1. コマンド行ウィンドウで、wxs\_home ディレクトリーに移動します。
2. 76 ページの『Java SE セキュリティー・チュートリアル - ステップ 3』で作成した wxs\_home/security ディレクトリーに切り替えます。
3. サーバーに対する認証の方法を実施する JAAS 構成ファイル (og\_jaas.config) を作成します。 security.xml ファイルで参照されている KeyStoreLoginAuthenticator は、JAAS ログイン・モジュール「KeyStoreLogin」を使用することによって鍵ストアを使用します。鍵ストアは、KeyStoreLoginModule クラスに対するオプションとして構成できます。

```
og_jaas.config
KeyStoreLogin{
com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule required
    keyStoreFile="../security/sampleKS.jks" debug = true;
};
```

4. パスワードが `sampleKS1` である鍵ストア `sampleKS.jks` を作成します。
5. `java_home/bin` ディレクトリーに切り替え、`keytool` を実行します。
6. `wxs_home /security` ディレクトリーに切り替え、それぞれ独自のパスワードを持つ 2 人のユーザー「`manager`」および「`cashier`」を作成します。
  - a. `keytool` を使用して、パスワード「`manager1`」を持つユーザー「`manager`」を鍵ストア `sampleKS.jks` に作成します。
 

```
keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
        -alias manager -keypass manager1
        -dname CN=manager,O=acme,OU=OGSample -validity 10000
```
  - b. `keytool` を使用して、パスワード「`cashier1`」を持つユーザー「`cashier`」を鍵ストア `sampleKS.jks` に作成します。
 

```
keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
        -alias cashier -keypass cashier1 -dname CN=cashier,O=acme,OU=OGSample
        -validity 10000
```
7. `wxs_home/properties` ディレクトリーに切り替えます。
8. `sampleClient.properties` ファイルをコピーします。
 

```
cp ../properties/sampleClient.properties client.properties
```
9. `wxs_home/security` ディレクトリーで、これを `client.properties` として保存します。
 

`client.properties` ファイルに対して以下の変更を行います。

  - a. **securityEnabled: securityEnabled** を `true` (デフォルト値) に設定します。認証を含むクライアント・セキュリティーが有効になります。
  - b. **credentialAuthentication: credentialAuthentication** を `Supported` (デフォルト値) に設定すると、クライアントで資格情報認証がサポートされます。
  - c. **transportType: transportType** を `TCP/IP` に設定すると、`SSL` は使用されません。
10. `sampleServer.properties` ファイルをコピーします。
 

```
cp ../properties/sampleServer.properties server.properties
```
11. `wxs_home/security` ディレクトリーで、これを `server.properties` として保存します。
 

`server.properties` ファイルで以下の変更を行います。

  - a. **securityEnabled: securityEnabled** 属性を `true` に設定します。
  - b. **transportType: transportType** 属性を `TCP/IP` に設定します。すなわち、`SSL` は使用されません。
  - c. **secureTokenManagerType: secureTokenManagerType** 属性を `none` に設定します。これで、セキュア・トークン・マネージャーが構成されなくなります。
12. `wxs_home/bin` ディレクトリーに移動し、ご使用のプラットフォームに応じて、次のいずれかのコマンドを実行してカタログ・サーバーを始動します。セキュリティー・プロパティーを渡すために、**`-clusterFile`** および **`-serverProps`** のコマンド行オプションを実行する必要があります。

- UNIX Linux

```
startOgServer.sh catalogServer -clusterSecurityFile ../security/security.xml
-serverProps ../security/server.properties -jvmArgs
-Djava.security.auth.login.config=../security/og_jaas.config"
```

- **Windows**

```
startOgServer.bat catalogServer -clusterSecurityFile ../security/security.xml
-serverProps ../security/server.properties -jvmArgs
-Djava.security.auth.login.config=../security/og_jaas.config"
```

13. 次のいずれかのスクリプトを使用して、`c0` という名前のコンテナ・サーバーを始動します。**-serverProps** を発行するとサーバー・プロパティ・ファイルが渡されます。

a.

- **Linux**    **UNIX**

```
startOgServer.sh c0 -objectgridFile ../xml/SimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml
-catalogServiceEndPoints localhost:2809
-serverProps ../security/server.properties
-jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config"
```

- **Windows**

```
startOgServer.bat c0 -objectgridFile ../xml/SimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml
-catalogServiceEndPoints localhost:2809
-serverProps ../security/server.properties
-jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config"
```

14. カタログ・サーバーとコンテナ・サーバーが始動されたならば、次のようにして `sec_sample.jar` サンプルを実行します。

```
java -classpath ../lib/objectgrid.jar;../applib/sec_sample.jar
com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
../security/client.properties manager manager1
```

**Linux**    前の例にあるように、セミコロン (;) ではなくコロン (:) をクラスパスの分離文字として使用します。

クラスを発行すると、以下の出力が得られます。

```
The customer name for ID 0001 is fName lName.
```

15. 次のように `xscmd` コマンド・ユーティリティーを実行して、「accounting」グリッドに挿入された「customer」マップのサイズを確認します。

- **UNIX**    **Linux**    `xscmd.sh -c showMapSizes -g accounting -m customer -username manager -password manager1`

- **Windows**    `xscmd.bat -c showMapSizes -g accounting -m customer -username manager -password manager1`

16. オプション: コンテナ・サーバーまたはカタログ・サーバーを停止する場合は、**stopOgServer** を使用することができます。ただし、セキュリティ構成ファイルを指定する必要があります。サンプル・クライアント・プロパティ・ファイルは、以下の 2 つのプロパティを定義して、ユーザー ID とパスワードの資格情報 (`manager/manager1`) を生成します。

```
credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
credentialGeneratorProps=manager manager1
```

次のコマンドを使用してコンテナ `c0` を停止します。

- `UNIX` `Linux` `stopOgServer.sh c0 -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..¥security¥client.properties`
- `Windows` `stopOgServer.bat c0 -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..¥security¥client.properties`

**-clientSecurityFile** オプションを指定しないと、次のメッセージを伴う例外が表示されます。

```
>> SERVER (id=39132c79, host=9.10.86.47) TRACE START:
```

```
>> org.omg.CORBA.NO_PERMISSION: Server requires credential authentication but there is no security context from the client. This usually happens when the client does not pass a credential the server.
```

```
vmcid: 0x0
```

```
minor code: 0
```

```
completed: No
```

また、以下のコマンドを使用してカタログ・サーバーをシャットダウンすることもできます。ただし、チュートリアル次のステップに続行する場合は、このカタログ・サーバーを実行させたままにしておいてかまいません。

- `UNIX` `Linux` `stopOgServer.sh catalogServer -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..¥security¥client.properties`
- `Windows` `stopOgServer.bat catalogServer -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..¥security¥client.properties`

カタログ・サーバーをシャットダウンすると、次の出力が表示されます。

```
CW0BJ2512I: ObjectGrid server catalogServer stopped
```

これで、認証を有効にすることにより、正常にシステムが部分的にセキュアになりました。サーバーを構成してユーザー・レジストリーをプラグインし、クライアントを構成してクライアント資格情報を提供するようにし、クライアント・プロパティ・ファイルおよびクラスター XML ファイルを変更して認証を有効にしています。

無効なパスワードを入力すると、ユーザー名およびパスワードが誤っていることを示す例外が表示されます。

クライアント認証について詳しくは、577 ページの『アプリケーション・クライアントの認証』を参照してください。

次のチュートリアル・ステップ

## Java SE セキュリティー・チュートリアル - ステップ 5

前のステップのようにクライアントを認証した後、eXtreme Scale 許可メカニズムによりセキュリティ特権を付与することができます。

### 始める前に

このタスクを続行する前に 78 ページの『Java SE セキュリティー・チュートリアル - ステップ 4』を完了している必要があります。

### このタスクについて

このチュートリアルの前のステップでは、eXtreme Scale グリッドで認証を使用可能にする方法について説明しました。この結果として、非認証クライアントは、サーバーに接続することができず、システムに要求の実行依頼をすることができません。ただし、認証されている各クライアントは、ObjectGrid マップに格納されているデータの読み取り、書き込み、削除など、サーバーに対して同じアクセス権または特権を持っています。クライアントは、どのような照会でも実行できます。このセクションでは、eXtreme Scale 許可を使用してさまざまな認証済みユーザーにさまざまな特権を付与する方法について説明します。

他の多くのシステムと同様、eXtreme Scale でもアクセス権ベースの許可メカニズムを採用しています。WebSphere eXtreme Scale には、各種の許可クラスによって表されるさまざまな許可カテゴリがあります。このトピックでは、MapPermission について説明します。許可のすべてのカテゴリは、クライアント許可プログラミングを参照してください。

WebSphere eXtreme Scale では、com.ibm.websphere.objectgrid.security.MapPermission クラスは eXtreme Scale リソース、特に ObjectMap インターフェースまたは JavaMap インターフェースのメソッドに対する許可を表しています。WebSphere eXtreme Scale は、ObjectMap および JavaMap のメソッドにアクセスするための以下の許可ストリングを定義します。

- read: マップからデータを読み取る許可を与えます。
- write: マップのデータを更新する許可を与えます。
- insert: マップにデータを挿入する許可を与えます。
- remove: マップからデータを削除する許可を与えます。
- invalidate: マップからのデータを無効にする許可を与えます。
- all: read、write、insert、remove、および invalidate に対するすべての許可を与えます。

クライアントが ObjectMap または JavaMap のメソッドを呼び出すと許可が行われます。eXtreme Scale ランタイムが、さまざまなメソッドの異なるマップ許可を検査します。必要な許可がクライアントに与えられていない場合は、AccessControlException が発生します。

このチュートリアルでは、Java 認証・承認サービス (JAAS) 許可を使用して、さまざまなユーザーに対する許可マップ・アクセスを付与する方法について説明します。



## 手順

1. **eXtreme Scale 許可を使用可能にします。** ObjectGrid で許可を使用可能にするには、XML ファイルで、その特定の ObjectGrid の `securityEnabled` 属性を `true` に設定する必要があります。ObjectGrid でセキュリティーを使用可能にするということは、許可を使用可能にするということです。以下のコマンドを使用して、セキュリティーが使用可能な新しい ObjectGrid XML ファイルを作成します。

- a. xml ディレクトリに移動します。

```
cd objectgridRoot/xml
```

- b. SimpleApp.xml ファイルを SecureSimpleApp.xml ファイルにコピーします。

```
cp SimpleApp.xml SecureSimpleApp.xml
```

- c. SecureSimpleApp.xml ファイルを開いて、以下の XML に示すように、ObjectGrid レベルで `securityEnabled="true"` を追加します。

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="accounting" securityEnabled="true">
      <backingMap name="customer" readOnly="false" copyKey="true"/>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

2. **許可ポリシーを定義します。** クライアントごとの認証のセクションで、鍵ストアに 3 人のユーザー (cashier、manager、および administrator) を作成しました。この例では、ユーザー「cashier」はすべてのマップに対する読み取り許可のみを持ち、ユーザー「manager」はすべての許可を持ちます。この例では、JAAS 許可が使用されます。JAAS 許可では許可ポリシー・ファイルを使用して、プリンシパルに許可を付与します。次の `og_auth.policy` ファイルはセキュリティー・ディレクトリに定義されています。

```
og_auth.policy
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
  principal javax.security.auth.x500.X500Principal "CN=cashier,0=acme,OU=OGSample" {
  permission com.ibm.websphere.objectgrid.security.MapPermission "accounting.*", "read ";
};

grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
  principal javax.security.auth.x500.X500Principal "CN=manager,0=acme,OU=OGSample" {
  permission com.ibm.websphere.objectgrid.security.MapPermission "accounting.*", "all";
};
```

注:

- `codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"` は、ObjectGrid 用の特別予約 URL です。プリンシパルに付与されているすべての ObjectGrid 許可では、この特別なコードベースを使用します。
- 1 番目の `grant` ステートメントでは、「read」マップ許可がプリンシパル `"CN=cashier,0=acme,OU=OGSample"` に付与されるので、cashier には、ObjectGrid アカウンティングのすべてのマップに対するマップ読み取り許可のみが付与されます。
- 2 番目の `grant` ステートメントでは「all」マップ許可がプリンシパル `"CN=manager,0=acme,OU=OGSample"` に付与されるので、manager には、ObjectGrid アカウンティングのマップに対するすべての許可が付与されます。

これで、許可ポリシーを使用してサーバーを起動することができます。次のように標準の `-D` プロパティを使用して JAAS 許可ポリシー・ファイルを設定することができます。`-Djava.security.auth.policy=../security/og_auth.policy`

### 3. アプリケーションを実行します。

上記のファイルを作成すると、アプリケーションを実行することができます。

以下のコマンドを使用して、カタログ・サーバーを始動します。カタログ・サービスの開始について詳しくは、451 ページの『スタンドアロン・カタログ・サービスの開始』を参照してください。

- a. `bin` ディレクトリーに移動します。 `cd objectgridRoot/bin`
- b. カタログ・サーバーを始動します。

- **UNIX Linux** `startOgServer.sh catalogServer -clusterSecurityFile ../security/security.xml -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config"`
- **Windows** `startOgServer.bat catalogServer -clusterSecurityFile ../security/security.xml -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config"`

`security.xml` ファイルおよび `server.properties` ファイルは、このチュートリアル前のステップで作成されています。

T

- c. 次に、以下のスクリプトを使用して、セキュア・コンテナ・サーバーを始動できます。 `bin` ディレクトリーから以下のスクリプトを実行します。

- **UNIX Linux** `# startOgServer.sh c0 -objectGridFile ../xml/SecureSimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809 -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config" -Djava.security.auth.policy=../security/og_auth.policy"`
- **Windows** `startOgServer.bat c0 -objectGridFile ../xml/SecureSimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809 -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config" -Djava.security.auth.policy=../security/og_auth.policy"`

前のコンテナ・サーバー始動コマンドとの以下の違いに注意してください。

- `SimpleApp.xml` ファイルの代わりに、`SecureSimpleApp.xml` ファイルを使用します。
- 別の `-Djava.security.auth.policy` 引数を追加して、JAAS 許可ポリシー・ファイルをコンテナ・サーバー・プロセスに設定します。

このチュートリアル直前のステップで使用したのと同じコマンドを使用します。

- a. bin ディレクトリーに移動します。
- b. 

```
java -classpath ../lib/objectgrid.jar;../applib/sec_sample.jar
com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
../security/client.properties manager manager1
```

ユーザー「manager」にはアカウントिंग ObjectGrid のマップに対するすべての許可が付与されているため、アプリケーションは正しく実行されます。

次に、ユーザー「manager」を使用する代わりに、ユーザー「cashier」を使用して、クライアント・アプリケーションを開始します。

- c. bin ディレクトリーに移動します。
- d. 

```
java -classpath ../lib/objectgrid.jar;../applib/sec_sample.jar
com.ibm.ws.objectgrid.security.sample.guide.SecureSimpleApp
../security/client.properties cashier cashier1
```

以下の例外が発生します。

```
Exception in thread "P=387313:0=0:CT" com.ibm.websphere.objectgrid.TransactionException:
rolling back transaction, see caused by exception
at com.ibm.ws.objectgrid.SessionImpl.rollbackPMapChanges(SessionImpl.java:1422)
at com.ibm.ws.objectgrid.SessionImpl.commit(SessionImpl.java:1149)
at com.ibm.ws.objectgrid.SessionImpl.mapPostInvoke(SessionImpl.java:2260)
at com.ibm.ws.objectgrid.ObjectMapImpl.update(ObjectMapImpl.java:1062)
at com.ibm.ws.objectgrid.security.sample.guide.SimpleApp.run(SimpleApp.java:42)
at com.ibm.ws.objectgrid.security.sample.guide.SecureSimpleApp.main(SecureSimpleApp.java:27)
Caused by: com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException:
Client Services - received exception from remote server:
com.ibm.websphere.objectgrid.TransactionException: transaction rolled back,
see caused by Throwable
at com.ibm.ws.objectgrid.client.RemoteTransactionCallbackImpl.processReadWriteResponse(
RemoteTransactionCallbackImpl.java:1399)
at com.ibm.ws.objectgrid.client.RemoteTransactionCallbackImpl.processReadWriteRequestAndResponse(
RemoteTransactionCallbackImpl.java:2333)
at com.ibm.ws.objectgrid.client.RemoteTransactionCallbackImpl.commit(RemoteTransactionCallbackImpl.java:557)
at com.ibm.ws.objectgrid.SessionImpl.commit(SessionImpl.java:1079)
... 4 more
Caused by: com.ibm.websphere.objectgrid.TransactionException: transaction rolled back, see caused by Throwable
at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processLogSequence(ServerCoreEventProcessor.java:1133)
at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processReadWriteTransactionRequest
(ServerCoreEventProcessor.java:910)
at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processClientServerRequest(ServerCoreEventProcessor.java:1285)

at com.ibm.ws.objectgrid.ShardImpl.processMessage(ShardImpl.java:515)
at com.ibm.ws.objectgrid.partition.IDLShardPOA._invoke(IDLShardPOA.java:154)
at com.ibm.CORBA.poa.POAServerDelegate.dispatchToServant(POAServerDelegate.java:396)
at com.ibm.CORBA.poa.POAServerDelegate.internalDispatch(POAServerDelegate.java:331)
at com.ibm.CORBA.poa.POAServerDelegate.dispatch(POAServerDelegate.java:253)
at com.ibm.rmi.iiop.ORB.process(ORB.java:503)
at com.ibm.CORBA.iiop.ORB.process(ORB.java:1553)
at com.ibm.rmi.iiop.Connection.respondTo(Connection.java:2680)
at com.ibm.rmi.iiop.Connection.doWork(Connection.java:2554)
at com.ibm.rmi.iiop.WorkUnitImpl.doWork(WorkUnitImpl.java:62)
at com.ibm.rmi.iiop.WorkerThread.run(ThreadPoolImpl.java:202)
at java.lang.Thread.run(Thread.java:803)
Caused by: java.security.AccessControlException: Access denied (
com.ibm.websphere.objectgrid.security.MapPermission accounting.customer write)
at java.security.AccessControlContext.checkPermission(AccessControlContext.java:155)
at com.ibm.ws.objectgrid.security.MapPermissionCheckAction.run(MapPermissionCheckAction.java:141)
at java.security.AccessController.doPrivileged(AccessController.java:275)
at javax.security.auth.Subject.doAsPrivileged(Subject.java:727)
at com.ibm.ws.objectgrid.security.MapAuthorizer$1.run(MapAuthorizer.java:76)
at java.security.AccessController.doPrivileged(AccessController.java:242)
at com.ibm.ws.objectgrid.security.MapAuthorizer.check(MapAuthorizer.java:66)
at com.ibm.ws.objectgrid.security.SecuredObjectMapImpl.checkMapAuthorization(SecuredObjectMapImpl.java:429)
```

```
at com.ibm.ws.objectgrid.security.SecuredObjectMapImpl.update(SecuredObjectMapImpl.java:490)
at com.ibm.ws.objectgrid.SessionImpl.processLogSequence(SessionImpl.java:1913)
at com.ibm.ws.objectgrid.SessionImpl.processLogSequence(SessionImpl.java:1805)
at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processLogSequence(ServerCoreEventProcessor.java:1011)
... 14 more
```

この例外は、ユーザー「cashier」に書き込み許可が付与されていないため、map customer を更新できないことが原因です。

これで、システムは許可をサポートするようになりました。許可ポリシーを定義して、ユーザーごとに各種の許可を付与することができます。許可について詳しくは、579 ページの『アプリケーション・クライアントの許可』を参照してください。

## 次のタスク

チュートリアル次のステップを完了します。『Java SE セキュリティー・チュートリアル - ステップ 6』を参照してください。

## Java SE セキュリティー・チュートリアル - ステップ 6

以下のステップでは、ご使用環境のエンドポイント間の通信にセキュリティ層を使用可能にする方法について説明します。

### 始める前に

このタスクを続行する前に 82 ページの『Java SE セキュリティー・チュートリアル - ステップ 5』を完了している必要があります。

### このタスクについて

eXtreme Scale トポロジーは、ObjectGrid エンドポイント (クライアント、コンテナ・サーバー、およびカタログ・サーバー) 間のセキュア通信のために Transport Layer Security/Secure Sockets Layer (TLS/SSL) をサポートします。このチュートリアル・ステップでは、それ以前のステップに基づいてトランスポート・セキュリティを使用可能にします。

### 手順

#### 1. TLS/SSL 鍵および鍵ストアの作成

トランスポート・セキュリティを使用可能にするためには、鍵ストアとトラストストアを作成する必要があります。この練習課題では、鍵ストアとトラストストアのペアのみを作成します。これらのストアは ObjectGrid クライアント、コンテナ・サーバー、およびカタログ・サーバーのために使用されるもので、JDK 鍵ツールを使用して作成されます。

- 鍵ストアに秘密鍵を作成します

```
keytool -genkey -alias ogsample -keystore key.jks -storetype JKS
-keyalg rsa -dname "CN=ogsample, OU=OGSample, O=acme, L=Your City,
S=Your State, C=Your Country" -storepass ogpass -keypass ogpass
-validity 3650
```

このコマンドを使用すると、「ogsample」という鍵を含む鍵ストア key.jks が作成されます。この鍵ストア key.jks は SSL 鍵ストアとして使用されます。

- パブリック証明書をエクスポートします

```
keytool -export -alias ogsample -keystore key.jks -file temp.key
-storepass ogpass
```

このコマンドを使用すると、「ogsample」という鍵の公開証明書が抽出されて、ファイル temp.key に格納されます。

- クライアントのパブリック証明書をトラストストアにインポートします

```
keytool -import -noprompt -alias ogsamplepublic -keystore trust.jks
-file temp.key -storepass ogpass
```

このコマンドを使用すると、公開証明書が鍵ストア trust.jks に追加されます。この trust.jks は SSL トラストストアとして使用されます。

## 2. ObjectGrid プロパティ・ファイルを構成します

このステップでは、トランスポート・セキュリティを使用可能にするように ObjectGrid プロパティ・ファイルを構成する必要があります。

まず、key.jks ファイルと trust.jks ファイルを objectgridRoot/security ディレクトリにコピーします。

client.properties および server.properties ファイルで以下のプロパティを設定します。

```
transportType=SSL-Required

alias=ogsample
contextProvider=IBMSSE2
protocol=SSL
keyStoreType=JKS
keyStore=./security/key.jks
keyStorePassword=ogpass
trustStoreType=JKS
trustStore=./security/trust.jks
trustStorePassword=ogpass
```

**transportType:** transportType の値は「SSL-Required」に設定されます。つまり、トランスポートに SSL が必要となります。したがって、すべての ObjectGrid エンドポイント (クライアント、カタログ・サーバー、およびコンテナ・サーバー) で SSL 構成が設定され、すべてのトランスポート通信が暗号化されます。

その他のプロパティは SSL 構成を設定するために使用されます。詳しくは、585 ページの『トランスポート層セキュリティおよび Secure Sockets Layer』を参照してください。必ずこのトピックの説明に従って、orb.properties ファイルを更新してください。

必ずこのページに従って、orb.properties ファイルを更新してください。

server.properties ファイルでは、別のプロパティ clientAuthentication を追加し、それを false に設定する必要があります。サーバー・サイドでは、クライアントを信頼する必要はありません。

clientAuthentication=false

### 3. アプリケーションの実行

使用するコマンドは 76 ページの『Java SE セキュリティー・チュートリアル - ステップ 3』トピックのコマンドと同じです。

以下のコマンドを使用してカタログ・サーバーを始動します。

- a. bin ディレクトリーに移動します。cd objectgridRoot/bin
- b. カタログ・サーバーを始動します。

- **Linux**      **UNIX**

```
startOgServer.sh catalogServer -clusterSecurityFile ../security/security.xml
-serverProps ../security/server.properties -JMXServicePort 11001
-jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config"
```

- **Windows**

```
startOgServer.bat catalogServer -clusterSecurityFile ../security/security.xml
-serverProps ../security/server.properties -JMXServicePort 11001 -jvmArgs
-Djava.security.auth.login.config=../security/og_jaas.config"
```

security.xml ファイルおよび server.properties ファイルは、75 ページの『Java SE セキュリティー・チュートリアル - ステップ 2』で作成されています。

**-JMXServicePort** オプションを使用して、サーバーの JMX ポートを明示的に指定してください。このオプションは、**xscmd** ユーティリティーを使用するために必要です。

セキュア ObjectGrid コンテナ・サーバーを実行します。

- c. 再度、bin ディレクトリーに移動します。cd objectgridRoot/bin
- d.

- **Linux**      **UNIX**

```
startOgServer.sh c0 -objectGridFile ../xml/SecureSimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints
localhost:2809 -serverProps ../security/server.properties
-JMXServicePort 11002 -jvmArgs
-Djava.security.auth.login.config=../security/og_jaas.config"
-Djava.security.auth.policy=../security/og_auth.policy"
```

- **Windows**

```
startOgServer.bat c0 -objectGridFile ../xml/SecureSimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809
-serverProps ../security/server.properties -JMXServicePort 11002
-jvmArgs -Djava.security.auth.login.config=../security/og_jaas.config"
-Djava.security.auth.policy=../security/og_auth.policy"
```

前のコンテナ・サーバー始動コマンドとの以下の違いに注意してください。

- SimpleApp.xml ファイルではなく、SecureSimpleApp.xml ファイルを使用します。
- 別の -Djava.security.auth.policy を追加して、JAAS 許可ポリシー・ファイルをコンテナ・サーバー・プロセスに設定します。

クライアント認証のために次のコマンドを実行します。

- a. cd objectgridRoot/bin

b.

```
javaHome/java -classpath ../lib/objectgrid.jar;../applib/sec_sample.jar
com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
../security/client.properties manager manager1
```

ユーザー「manager」にはアカウントिंग ObjectGrid のすべてのマップに対する許可が付与されているため、アプリケーションは正常に実行されます。

**xscmd** ユーティリティーを使用して「accounting」グリッドのマップ・サイズを表示できます。

- ディレクトリー objectgridRoot/bin に移動します。
- **xscmd** コマンドを使用して、マップ・サイズを表示します。

```
— UNIX Linux
xscmd.sh -c showMapSizes -g accounting -m customer -prot SSL
-ts .%security%\trust.jks -tsp ogpass -tst jks
-user manager -pwd manager1 -ks ..\security\key.jks -ksp ogpass -kst JKS
-cxpv IBMJSSE2 -tt SSL-Required

— Windows
xscmd.bat -c showMapSizes -g accounting -m customer -prot SSL
-ts .%security%\trust.jks -tsp ogpass -tst jks
-user manager -pwd manager1 -ks ..\security\key.jks -ksp ogpass -kst JKS
-cxpv IBMJSSE2 -tt SSL-Required
```

ここで、`-p 11001` を使用してカタログ・サービスの JMX ポートを指定することに注意してください。

以下の出力が表示されます。

```
This administrative utility is provided as a sample only and is not to
be considered a fully supported component of the WebSphere eXtreme Scale product.
Connecting to Catalog service at localhost:1099
***** Displaying Results for Grid - accounting, MapSet - customer *****
*** Listing Maps for c0 ***
Map Name: customer Partition #: 0 Map Size: 1 Shard Type: Primary
Server Total: 1
Total Domain Count: 1
```

## 間違った鍵ストアを使用したアプリケーションの実行

鍵ストア内の秘密鍵の公開証明書がトラストストアに含まれていないと、鍵がトラステッド鍵でありえないことを示す例外が発生します。

このことを示すために、もう 1 つの鍵ストア `key2.jks` を作成します。

```
keytool -genkey -alias ogsample -keystore key2.jks -storetype JKS
-keyalg rsa -dname "CN=ogsample, OU=Your Organizational Unit, O=Your
Organization, L=Your City, S=Your State, C=Your Country" -storepass
ogpass -keypass ogpass -validity 3650
```

次に、`server.properties` を変更して、`keyStore` が、この新規の鍵ストア `key2.jks` をポイントするようにします。

```
keyStore=../security/key2.jks
```

次のコマンドを実行してカタログ・サーバーを始動します。

- a. bin ディレクトリーに移動します。 `cd objectgridRoot/bin`



- b. カタログ・サーバーを始動します。

Linux

UNIX

```
startOgServer.sh c0 -objectGridFile ../xml/SecureSimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809
-serverProps ../security/server.properties -jvmArgs
-Djava.security.auth.login.config="../security/og_jaas.config"
-Djava.security.auth.policy="../security/og_auth.policy"
```

Windows

```
startOgServer.bat c0 -objectGridFile ../xml/SecureSimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809
-serverProps ../security/server.properties -jvmArgs
-Djava.security.auth.login.config="../security/og_jaas.config"
-Djava.security.auth.policy="../security/og_auth.policy"
```

次の例外が表示されます。

```
Caused by: com.ibm.websphere.objectgrid.ObjectGridRPCException:
com.ibm.websphere.objectgrid.ObjectGridRuntimeException:
SSL connection fails and plain socket cannot be used.
```

最後に、key.jks ファイルを使用するように server.properties ファイルを元に戻します。

---

## チュートリアル: WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合

このチュートリアルでは、WebSphere Application Server 環境で WebSphere eXtreme Scale サーバー・デプロイメントを保護する方法について説明します。

### 学習目標

このチュートリアルの学習目標は次のとおりです。

- WebSphere Application Server 認証プラグインを使用するための WebSphere eXtreme Scale の構成
- WebSphere Application Server CSIv2 構成を使用するための WebSphere eXtreme Scale トランスポート・セキュリティの構成
- WebSphere Application Server での Java 認証・承認サービス (JAAS) 許可の使用
- グループ・ベースの JAAS 許可のカスタム・ログイン・モジュールの使用
- WebSphere Application Server 環境での WebSphere eXtreme Scale `xscmd` ユーティリティーの使用

### 所要時間

このチュートリアルは、開始してから終了するまで約 4 時間かかります。

## 概要: WebSphere Application Server 認証プラグインを使用した、WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合

このチュートリアルでは、WebSphere eXtreme Scale セキュリティーを WebSphere Application Server と統合します。まず、現行スレッドからの認証ユーザー資格情報

を使用して ObjectGrid に接続する単純な Web アプリケーションの認証を構成します。次に、Transport Layer Security でクライアントとサーバー間を転送されるデータの暗号化を詳細に調べます。ユーザーにさまざまなレベルの許可を与えるために、Java 認証・承認サービス (JAAS) を構成できます。構成が終了すると、`xscmd` ユーティリティを使用して、データ・グリッドとマップをモニターできます。

このチュートリアルでは、すべての WebSphere eXtreme Scale クライアント、コンテナ・サーバー、およびカタログ・サーバーは、WebSphere Application Server 環境にデプロイされていると想定しています。

## 学習目標

このチュートリアルの学習目標は次のとおりです。

- WebSphere Application Server 認証プラグインを使用するための WebSphere eXtreme Scale の構成
- WebSphere Application Server CSIv2 構成を使用するための WebSphere eXtreme Scale トランスポート・セキュリティの構成
- WebSphere Application Server での Java 認証・承認サービス (JAAS) 許可の使用
- グループ・ベースの JAAS 許可のカスタム・ログイン・モジュールの使用
- WebSphere Application Server 環境での WebSphere eXtreme Scale `xscmd` ユーティリティの使用

## 所要時間

このチュートリアルは、開始してから終了するまで約 4 時間かかります。

## スキル・レベル

中級。

## 対象者

WebSphere eXtreme Scale と WebSphere Application Server の間のセキュリティの統合に関心のある開発者および管理者。

## システム要件およびトポロジー

- WebSphere Application Server バージョン 6.1 またはバージョン 7.0.0.11 以降
- 次のフィックスを適用して、Java ランタイムを更新してください。IZ79819: IBMJDK FAILS TO READ PRINCIPAL STATEMENT WITH WHITESPACE FROM SECURITY FILE

このチュートリアルでは、4 つのアプリケーション・サーバー (WebSphere Application Server) と 1 つのデプロイメント・マネージャーを使用してサンプル・デモを行います。

## 前提条件

このチュートリアルを開始するにあたって、次の項目についての基本的な知識があると便利です。

- WebSphere eXtreme Scale プログラミング・モデル

- 基本的な WebSphere eXtreme Scale セキュリティーの概念
- 基本的な WebSphere eXtreme Scale セキュリティーの概念

WebSphere eXtreme Scale と WebSphere Application Server のセキュリティー統合のバックグラウンド情報については、596 ページの『WebSphere Application Server とのセキュリティー統合』を参照してください。

## モジュール 1: WebSphere Application Server の準備

WebSphere eXtreme Scale との統合を行うチュートリアルを開始する前に、WebSphere Application Server に基本セキュリティー構成を作成する必要があります。

### 学習目標

このモジュールのレッスンでは、以下の方法について学習します。

- ユーザー・アカウント・レジストリーとして内部ファイル・ベースの統合リポジトリを使用するための、WebSphere Application Server セキュリティーの構成。
- ユーザー・グループおよびユーザーの作成。
- アプリケーションおよび WebSphere eXtreme Scale サーバー用のクラスターの作成。

### 所要時間

このモジュールの所要時間は約 60 分です。

### レッスン 1.1: トポロジーの理解とチュートリアル・ファイルの入手

チュートリアル用の環境を準備するには、WebSphere Application Server セキュリティーを構成する必要があります。ユーザー・アカウント・レジストリーとして内部ファイル・ベースの統合リポジトリを使用して、管理およびアプリケーション・セキュリティーを構成します。

このレッスンでは、チュートリアルで使用するサンプル・トポロジーとアプリケーションを紹介します。チュートリアルの実行を開始するには、アプリケーションをダウンロードし、環境内の正しい場所に構成ファイルを配置する必要があります。サンプル・アプリケーションは WebSphere eXtreme Scale wiki からダウンロードできます。

**WebSphere Application Server サンプル・トポロジー:** このチュートリアルでは、セキュリティーを使用可能に設定したサンプル・アプリケーションを使用してデモンストレーションする 4 つの WebSphere Application Server アプリケーション・サーバーを作成します。これらのアプリケーション・サーバーは、それぞれ 2 つのサーバーが入った、2 つのクラスターにグループ化されます。

- **appCluster クラスター:** EmployeeManagement サンプル・エンタープライズ・アプリケーションをホストします。このクラスターには、s1 と s2 の 2 つのアプリケーション・サーバーがあります。
- **xsCluster クラスター:** eXtreme Scale コンテナ・サーバーをホストします。このクラスターには、xs1 と xs2 の 2 つのアプリケーション・サーバーがあります。

このデプロイメント・トポロジーでは、s1 および s2 のアプリケーション・サーバーは、データ・グリッドに保管されたデータにアクセスするクライアント・サーバーです。xs1 サーバーと xs2 サーバーは、データ・グリッドをホストするコンテナ・サーバーです。

デフォルトでは、カタログ・サーバーがデプロイメント・マネージャー・プロセスでデプロイされます。このチュートリアルは、デフォルトの振る舞いを使用します。デプロイメント・マネージャー内でカタログ・サーバーをホストすることは、実稼働環境ではお勧めしません。実稼働環境では、カタログ・サーバーの始動場所を定義するカタログ・サービス・ドメインを作成する必要があります。詳しくは、288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』を参照してください。

**代替の構成:** すべてのアプリケーション・サーバーを、単一のクラスター内で (例えば appCluster クラスター内で) ホストすることができます。この構成では、クラスター内のすべてのサーバーがクライアントとコンテナ・サーバーの両方を兼ねます。このチュートリアルでは、2 つのクラスターを使用して、クライアントをホストしているアプリケーション・サーバーとコンテナ・サーバーをホストしているアプリケーション・サーバーを区別しています。

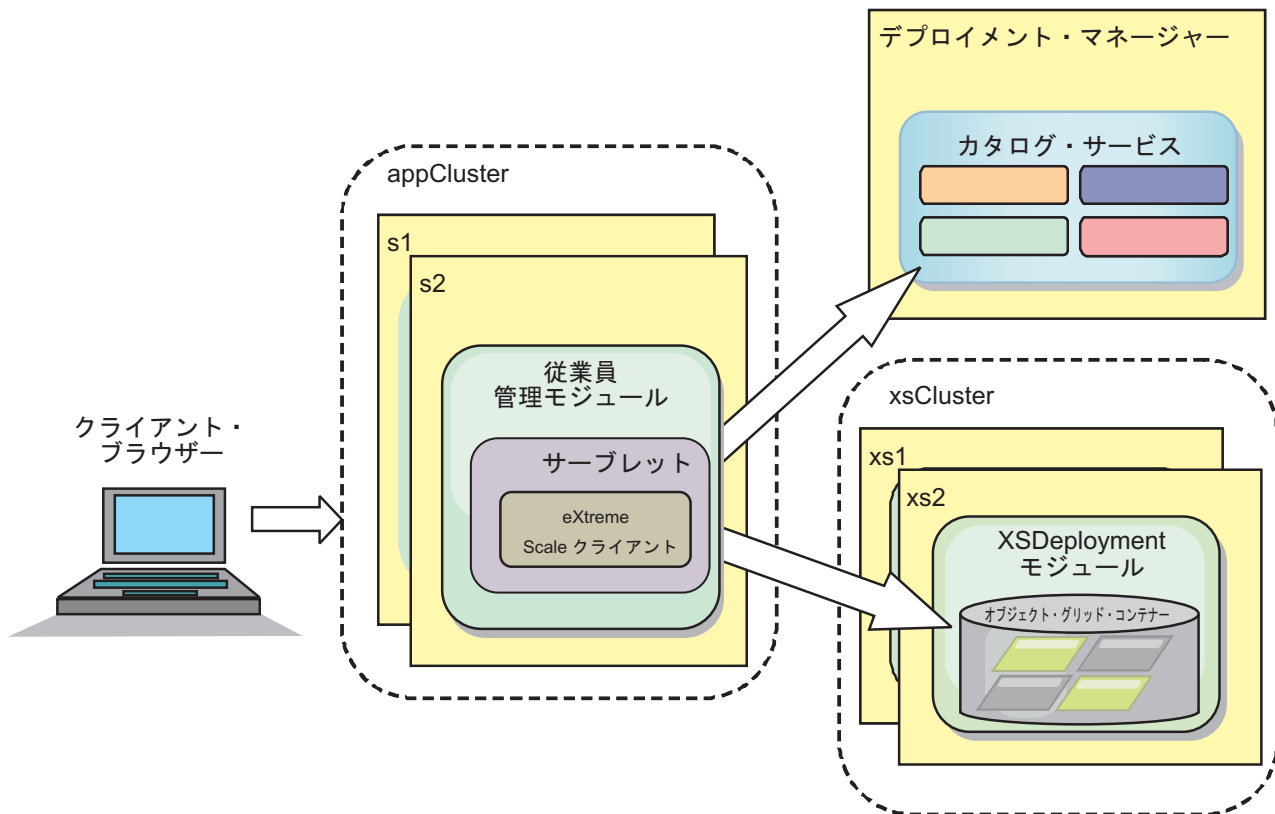


図 19. チュートリアルのトポロジー

**アプリケーション:** このチュートリアルでは、2 つのアプリケーションと、1 つの共有ライブラリー・ファイルを使用します。

- **EmployeeManagement.ear:** EmployeeManagement.ear アプリケーションは、単純化された Java 2 Platform, Enterprise Edition (J2EE) エンタープライズ・アプリケー

ションです。これには、従業員プロフィールを管理するための Web モジュールが含まれます。Web モジュールには、コンテナ・サーバーに保管された従業員プロフィールを表示、挿入、更新、および削除する `management.jsp` ファイルが含まれます。

- **XSDeployment.ear:** このアプリケーションにはエンタープライズ・アプリケーション・モジュールが含まれ、アプリケーション成果物は含まれません。キャッシュ・オブジェクトは `EmployeeData.jar` ファイルにパッケージ化されます。`EmployeeData.jar` ファイルは、`XSDeployment.ear` ファイルがクラスにアクセスできるように、`XSDeployment.ear` ファイルの共有ライブラリーとしてデプロイされます。このアプリケーションの目的は、eXtreme Scale 構成ファイルをパッケージ化することにあります。このエンタープライズ・アプリケーションが開始されると、eXtreme Scale ランタイムによって eXtreme Scale 構成ファイルが自動的に検出され、その結果コンテナ・サーバーが作成されます。これらの構成ファイルには、`objectGrid.xml` と `objectGridDeployment.xml` ファイルが含まれます。
- **EmployeeData.jar:** この JAR ファイルは `com.ibm.websphere.sample.xs.data.EmployeeData` クラスという 1 つのクラスを含んでいます。このクラスは、グリッドに保管される従業員データを表します。この Java アーカイブ (JAR) ファイルは、共有ライブラリーとして `EmployeeManagement.ear` および `XSDeployment.ear` ファイルと一緒にデプロイされます。

#### チュートリアル・ファイルの入手:

1. `WASSecurity.zip` ファイルと `security.zip` ファイルをダウンロードします。サンプル・アプリケーションは [WebSphere eXtreme Scale wiki](#) からダウンロードできます。
2. `WASSecurity.zip` ファイルを、バイナリーおよびソース成果物を表示するためのディレクトリー、例えば `/wxs_samples/` ディレクトリーに解凍します。今後、チュートリアルの中ではこのディレクトリーを `samples_home` と呼びます。`WASSecurity.zip` ファイルの内容の説明、およびソースを Eclipse ワークスペースにロードする方法については、パッケージの中の `README.txt` ファイルを参照してください。
3. `security.zip` ファイルを `samples_home` ディレクトリーに解凍します。`security.zip` ファイルには、このチュートリアルで使用する次のセキュリティ構成が含まれます。
  - `catServer2.props`
  - `server2.props`
  - `client2.props`
  - `securityWAS2.xml`
  - `xsAuth2.props`

#### 構成ファイルについて:

`objectGrid.xml` ファイルと `objectGridDeployment.xml` ファイルは、アプリケーション・データを保管するデータ・グリッドとマップを作成します。

これらの構成ファイルには、`objectGrid.xml` と `objectGridDeployment.xml` という名前を付ける必要があります。アプリケーション・サーバーが始動すると、eXtreme

Scale は、EJB および Web モジュールの META-INF ディレクトリーで、これらのファイルを検出します。これらのファイルが検出された場合、Java 仮想マシン (JVM) は構成ファイルの中に定義されたデータ・グリッドのコンテナ・サーバーとして機能するとみなされます。

### objectGrid.xml ファイル

objectGrid.xml ファイルは、Grid という名前の ObjectGrid を 1 つ定義します。Grid データ・グリッドには、アプリケーションの従業員プロフィールを保管する 1 つの Map1 というマップがあります。

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="Grid" txTimeout="15">
      <backingMap name="Map1" />
    </objectGrid>
  </objectGrids>

</objectGridConfig>
```

### objectGridDeployment.xml ファイル

objectGridDeployment.xml ファイルは、Grid データ・グリッドのデプロイ方法を指定します。グリッドがデプロイされると、グリッドは 5 つの区画と 1 つの同期レプリカを持ちます。

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="Grid">
    <mapSet name="mapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="1" >
      <map ref="Map1"/>
    </mapSet>
  </objectgridDeployment>

</deploymentPolicy>
```

### レッスンのチェックポイント:

このレッスンでは、チュートリアル用のトポロジーについて学習し、構成ファイルとサンプル・アプリケーションを環境に追加しました。

コンテナ・サーバーの自動始動について詳しくは、314 ページの『コンテナ・サーバーの自動始動のための WebSphere Application Server アプリケーションの構成』を参照してください。

## レッスン 1.2: WebSphere Application Server 環境の構成

チュートリアル用の環境を準備するには、WebSphere Application Server セキュリティーを構成する必要があります。内部ファイル・ベースの統合リポジトリーをユーザー・アカウント・レジストリーとして使用して、管理セキュリティーおよびアプリケーション・セキュリティーを使用可能にします。その後、クライアント・アプリケーションとコンテナ・サーバーをホスティングするサーバー・クラスターを作成できます。



次のステップは、WebSphere Application Server バージョン 7.0 を使用した記述になっています。しかし、考え方は、それより前の WebSphere Application Server バージョンにも適用できます。

### WebSphere Application Server セキュリティーの構成:

1. WebSphere Application Server セキュリティーを構成します。
  - a. WebSphere Application Server 管理コンソールで、「セキュリティ」 > 「グローバル・セキュリティ」をクリックします。
  - b. 「ユーザー・アカウント・リポジトリ」に「統合リポジトリ」を選択します。「現在の値で設定」をクリックします。
  - c. 「構成..」をクリックして、「統合リポジトリ」パネルに進みます。
  - d. 「1 次管理ユーザー名」を入力します。例えば、admin です。「適用」をクリックします。
  - e. プロンプトが表示されたら、管理ユーザー・パスワードを指定して、「OK」をクリックします。変更を保存します。
  - f. 「グローバル・セキュリティ」ページで、「統合リポジトリ」設定が、現行ユーザー・アカウント・レジストリーに設定されていることを確認します。
  - g. 「管理セキュリティを使用可能にする」、「アプリケーション・セキュリティを使用可能にする」、および「Java 2 セキュリティーを使用して、アプリケーション・アクセスをローカル・リソースに制限する」の項目を選択します。「適用」をクリックして、変更を保存します。
  - h. デプロイメント・マネージャーを再始動し、実行中のアプリケーションがあれば再開します。

ユーザー・アカウント・レジストリーとして内部ファイル・ベースの統合リポジトリを使用して、WebSphere Application Server 管理セキュリティが使用可能になりました。

2. adminGroup と operatorGroup の 2 つのユーザー・グループを作成します。
  - a. 「ユーザーおよびグループ」 > 「グループの管理」 > 「作成...」をクリックします。
  - b. グループ名に「adminGroup」を入力します。説明に「管理グループ」を入力します。「作成」をクリックします。
  - c. 「同じものを作成」をクリックします。グループ名に「operatorGroup」を入力します。説明に「オペレーター・グループ」を入力します。「作成」をクリックします。
  - d. 「閉じる」をクリックします。
3. ユーザー admin1 と operator1 を作成します。
  - a. 「ユーザーおよびグループ」 > 「ユーザーの管理」 > 「作成...」をクリックします。
  - b. admin1 というユーザーを作成します。名を Joe、姓を Doe、パスワードを admin1 にします。「作成」をクリックします。



- c. 2 番目のユーザーを作成します。「同じものを作成」をクリックして、operator1 というユーザーを作成します。名を Jane、姓を Doe、パスワードを operator1 にします。「作成」をクリックします。「閉じる」をクリックします。
4. ユーザーをユーザー・グループに追加します。admin1 ユーザーを adminGroup に、operator1 ユーザーを operatorGroup に追加します。
  - a. 「ユーザーおよびグループ」 > 「ユーザーの管理」をクリックします。
  - b. グループに追加するユーザーを検索します。「検索..」をクリックし、すべてのユーザーを表示するために検索対象値にアスタリスク (\*) を設定します。
  - c. 検索結果から、「admin1」ユーザーを選択し、「グループ」タブをクリックします。「追加」をクリックして、グループを追加します。
  - d. 使用可能なグループを見つけるために、グループを検索します。「adminGroup」をクリックし、「追加」をクリックします。
  - e. 上記のステップを繰り返して、operator1 ユーザーを operatorGroup ユーザー・グループに追加します。
5. 変更を保存し、管理コンソールからログアウトします。そして、デプロイメント・マネージャーおよびノード・エージェントを再始動して、セキュリティ設定を使用可能にします。

セキュリティを使用可能にし、WebSphere Application Server 構成に対して管理アクセス権限とオペレーター・アクセス権限を持つ、ユーザーとユーザー・グループを作成しました。

#### サーバー・クラスタの作成:

WebSphere Application Server 構成の中に、次の 2 つのサーバー・クラスタを作成します。appCluster クラスタはチュートリアルサンプル・アプリケーションをホストし、xsCluster クラスタはデータ・グリッドをホストします。

1. WebSphere Application Server 管理コンソールで、クラスタのパネルを開きます。「サーバー」 > 「クラスタ」 > 「WebSphere Application Server クラスタ」 > 「新規」をクリックします。
2. クラスタ名に「appCluster」を入力し、「ローカルを優先」オプションを選択したままにして、「次へ」をクリックします。
3. クラスタの中にサーバーを作成します。デフォルト・オプションのままにして、s1 という名前のサーバーを作成します。追加の s2 という名前のユーザー・メンバーを追加します。
4. ウィザードの残りのステップを完了して、クラスタを作成します。変更を保存します。
5. 上記のステップを繰り返して、xsCluster クラスタを作成します。このクラスタには、xs1 および xs2 という名前の 2 つのサーバーが含まれています。

#### レッスンのチェックポイント:

WebSphere Application Server セルのグローバル・セキュリティを使用可能にし、ユーザーおよびユーザー・グループを作成しました。また、アプリケーションおよびデータ・グリッドをホストするクラスタを作成しました。

## モジュール 2: WebSphere Application Server 認証プラグインを使用するための WebSphere eXtreme Scale の構成

WebSphere Application Server 構成を作成した後、WebSphere Application Server に WebSphere eXtreme Scale 認証を統合できます。

WebSphere eXtreme Scale クライアントは、認証を必要とするコンテナ・サーバーに接続するときに、`com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator` インターフェースによって表される資格情報生成プログラムを提供する必要があります。資格情報生成プログラムは、クライアントの資格情報を作成するファクトリーです。クライアント資格情報には、ユーザー名とパスワードのペア、Kerberos チケット、クライアント証明書、またはクライアントとサーバーが同意する任意の形式でのクライアント識別データがあります。詳しくは、資格情報 API 資料を参照してください。このサンプルでは、WebSphere eXtreme Scale クライアントは、`appCluster` クラスターにデプロイされる `EmployeeManagment Web` アプリケーションです。クライアント資格情報は、Web ユーザー ID を表す WebSphere セキュリティー・トークンです。

### 学習目標

このモジュールのレッスンでは、以下の方法について学習します。

- クライアント・サーバー・セキュリティーの構成。
- カタログ・サーバー・セキュリティーの構成。
- コンテナ・サーバー・セキュリティーの構成。
- サンプル・アプリケーションをインストールして実行する。

### 所要時間

このモジュールの所要時間は約 60 分です。

### レッスン 2.1: クライアント・サーバー・セキュリティーの構成

クライアント・プロパティー・ファイルで、使用する `CredentialGenerator` 実装クラスを指示します。

**-Dobjectgrid.client.props** JVM プロパティーを使用して、クライアント・プロパティー・ファイルを構成します。このプロパティーに指定されるファイル名は、例えば `samples_home/security/client2.props` などの絶対ファイル・パスです。クライアント・プロパティー・ファイルについて詳しくは、クライアント・プロパティー・ファイルを参照してください。

#### クライアント・プロパティー・ファイルの内容:

この例では、クライアント資格情報として WebSphere Application Server セキュリティー・トークンを使用します。`client2.props` ファイルは、`samples_home/security` ディレクトリーにあります。`client2.props` ファイルには次の設定が含まれます。

#### **securityEnabled**

`true` に設定すると、クライアントが使用可能なセキュリティー情報をサーバーに送信しなければならないことを示します。

## credentialAuthentication

Supported に設定すると、クライアントは、資格情報認証をサポートすることを示します。

## credentialGeneratorClass

クライアントがスレッドからセキュリティー・トークンを取得するよう、`com.ibm.websphere.objectgrid.security.plugins.builtins.`

`WSTokenCredentialGenerator` クラスを指定します。セキュリティー・トークンの取得方法について詳しくは、596 ページの『WebSphere Application Server とのセキュリティー統合』を参照してください。

## Java 仮想マシン (JVM) プロパティーを使用したクライアント・プロパティー・ファイルの設定:

管理コンソールで、`appCluster` クラスター内の `s1` サーバーと `s2` サーバーのそれぞれに対し次のステップを実行します。別のトポロジーを使用している場合は、`EmployeeManagement` アプリケーションがデプロイされるすべてのアプリケーション・サーバーに対し次のステップを実行してください。

1. 「サーバー」 > 「WebSphere Application Server」 > 「*server\_name*」 > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」を選択します。
2. 次の汎用 JVM プロパティーを作成して、クライアント・プロパティー・ファイルの場所を設定します。  
`-Dobjectgrid.client.props=samples_home/security/client2.props`
3. 「OK」をクリックして、変更を保存します。

## レッスンのチェックポイント:

クライアント・プロパティー・ファイルを編集し、クライアント・プロパティー・ファイルを使用するよう `appCluster` クラスター内のサーバーを構成しました。このプロパティー・ファイルで、使用する `CredentialGenerator` 実装クラスを指示します。

## レッスン 2.2: カタログ・サーバー・セキュリティーの構成

カタログ・サーバーには、2 つの異なるレベルのセキュリティー情報が含まれます。カタログ・サービスとコンテナ・サーバーも含めた、すべての `WebSphere eXtreme Scale` サーバーに共通するセキュリティー・プロパティーと、カタログ・サーバーに固有のセキュリティー・プロパティーです。

カタログ・サーバーとコンテナ・サーバーに共通するセキュリティー・プロパティーは、セキュリティー XML 記述子ファイル内に構成します。共通プロパティーの例の 1 つは、ユーザー・レジストリーと認証メカニズムを表すオーセンティケーター構成です。セキュリティー・プロパティーの詳細については、セキュリティー記述子 XML ファイルを参照してください。

セキュリティー XML 記述子ファイルを構成するには、Java 仮想マシン (JVM) 引数の中に `-Dobjectgrid.cluster.security.xml.url` プロパティーを作成します。このプロパティーに指定するファイル名は、`file:///samples_home/security/securityWAS2.xml` のような URL 形式です。

## securityWAS2.xml ファイル:

このチュートリアルでは、securityWAS2.xml ファイルは *samples\_home/security* ディレクトリーにあります。コメントを削除した securityWAS2.xml ファイルの内容は次のとおりです。

```
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
xmlns="http://ibm.com/ws/objectgrid/config/security">
  <security securityEnabled="true">
    <authenticator
      className="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator">
    </authenticator>
  </security>
</securityConfig>
```

次のプロパティーが securityWAS2.xml ファイルの中で定義されます。

### securityEnabled

securityEnabled プロパティーは true に設定され、WebSphere eXtreme Scale グローバル・セキュリティーが使用可能なことをカタログ・サーバーに指示します。

### authenticator

オーセンティケーターは、com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator クラスとして構成されます。この組み込みの Authenticator プラグインの実装があれば、WebSphere eXtreme Scale サーバーは、セキュリティー・トークンを Subject オブジェクトに変換できます。セキュリティー・トークンの変換方法について詳しくは、596 ページの『WebSphere Application Server とのセキュリティー統合』を参照してください。

## catServer2.props ファイル:

サーバー・プロパティー・ファイルは、サーバー固有のプロパティーを保管し、これにはサーバー固有のセキュリティー・プロパティーも含まれます。詳しくは、サーバー・プロパティー・ファイルを参照してください。JVM 引数の中で -Dobjectgrid.server.props プロパティーを使用して、サーバー・プロパティー・ファイルを構成できます。このプロパティーのファイル名の値を、例えば *samples\_home/security/catServer2.props* などの絶対ファイル・パスで指定します。このチュートリアルでは、catServer2.props ファイルは *samples\_home/security* ディレクトリーの中にあります。コメントを削除した catServer2.props ファイルの内容は次のとおりです。

### securityEnabled

securityEnabled プロパティーは true に設定され、このカタログ・サーバーがセキュア・サーバーであることを示します。

### credentialAuthentication

credentialAuthentication プロパティーは Required に設定され、サーバーに接続するすべてのクライアントが資格情報の提供を要求されます。

### secureTokenManagerType

secureTokenManagerType は none に設定され、既存のサーバーに結合するとき認証の機密事項が暗号化されないことを示します。

## authenticationSecret

authenticationSecret プロパティは、ObjectGridDefaultSecret に設定されます。eXtreme Scale サーバー・クラスターに結合するとき、この秘密ストリングが使用されます。サーバーがデータ・グリッドに結合する場合、秘密ストリングの表示を求められます。結合サーバーの秘密ストリングがカタログ・サーバーのいずれかの秘密ストリングと一致する場合は、結合サーバーは受け入れられます。ストリングが一致しない場合、結合要求は拒否されます。

## transportType

transportType プロパティは、当初 TCP/IP に設定します。後ほどチュートリアルの中で、トランスポート・セキュリティを使用可能にします。

## JVM プロパティによるサーバー・プロパティ・ファイルの設定:

デプロイメント・マネージャー・サーバーにサーバー・プロパティ・ファイルを設定します。このチュートリアルのトポロジーとは異なるトポロジーを使用している場合は、コンテナ・サーバーをホストするために使用しているすべてのアプリケーション・サーバー上にサーバー・プロパティ・ファイルを設定します。

1. サーバーの Java 仮想マシン構成を開きます。管理コンソールで、「システム管理」 > 「デプロイメント・マネージャー」 > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」をクリックします。
2. 次の汎用 JVM 引数を追加します。

```
-Dobjectgrid.cluster.security.xml.url=file:///samples_home/security/securityWAS2.xml  
-Dobjectgrid.server.props=samples_home/security/catServer2.props
```

3. 「OK」をクリックして、変更を保存します。

## レッスンのチェックポイント:

securityWAS2.xml ファイルと catServer2.props ファイルをデプロイメント・マネージャーに関連付けることにより、カタログ・サーバー・セキュリティを構成しました。デプロイメント・マネージャーは、WebSphere Application Server 構成の中のカテゴリ・サーバー・プロセスをホストします。

## レッスン 2.3: コンテナ・サーバー・セキュリティの構成

コンテナ・サーバーは、カタログ・サービスに接続するときに、オブジェクト・グリッド・セキュリティ XML ファイルに構成されているすべてのセキュリティ構成 (オーセンティケータ構成、ログイン・セッションのタイムアウト値、その他の構成情報など) を取得します。コンテナ・サーバーは、サーバー・プロパティ・ファイル内にそのサーバー固有のセキュリティ・プロパティも保持します。

-Dobjectgrid.server.props Java 仮想マシン (JVM) プロパティを使用して、サーバー・プロパティ・ファイルを構成します。このプロパティのファイル名は、例えば `samples_home/security/server2.props` などの絶対ファイル・パスです。

このチュートリアルでは、コンテナ・サーバーは xsCluster クラスター内の xs1 および xs2 サーバーでホスティングされます。

## server2.props ファイル:

server2.props ファイルは、WASSecurity ディレクトリーの下の *samples\_home/security* ディレクトリーにあります。server2.props ファイルで定義されているプロパティーは次のとおりです。

#### **securityEnabled**

securityEnabled プロパティーは true に設定され、このコンテナ・サーバーがセキュア・サーバーであることを示します。

#### **credentialAuthentication**

credentialAuthentication プロパティーは Required に設定され、サーバーに接続するすべてのクライアントが資格情報の提供を要求されます。

#### **secureTokenManagerType**

secureTokenManagerType は none に設定され、既存のサーバーに結合するとき認証の機密事項が暗号化されないことを示します。

#### **authenticationSecret**

authenticationSecret プロパティーは、ObjectGridDefaultSecret に設定されます。eXtreme Scale サーバー・クラスターに結合するとき、この秘密ストリングが使用されます。サーバーがデータ・グリッドに結合する場合、秘密ストリングの表示を求められます。結合サーバーの秘密ストリングがカタログ・サーバーのいずれかの秘密ストリングと一致する場合は、結合サーバーは受け入れられます。ストリングが一致しない場合、結合要求は拒否されません。

### **JVM プロパティーによるサーバー・プロパティー・ファイルの設定:**

xs1 サーバーと xs2 サーバーにサーバー・プロパティー・ファイルを設定します。使用するトポロジーがこのチュートリアルと異なる場合は、コンテナ・サーバーのホスティングに使用するすべてのアプリケーション・サーバーにサーバー・プロパティー・ファイルを設定してください。

1. サーバーの Java 仮想マシン・ページを開きます。「サーバー」 > 「アプリケーション・サーバー」 > *server\_name* > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」
2. 汎用 JVM 引数を追加します。  
-Dobjectgrid.server.props=*samples\_home/security/server2.props*
3. 「OK」をクリックして、変更を保存します。

#### **レッスンのチェックポイント:**

これで、WebSphere eXtreme Scale サーバー認証は保護されます。このセキュリティを構成することで、WebSphere eXtreme Scale サーバーに接続しようとするすべてのアプリケーションが資格情報の提供を要求されます。このチュートリアルでは、WSTokenAuthenticator がオーセンティケーターです。この結果として、クライアントは、WebSphere Application Server セキュリティー・トークンの提供が必要です。

### **レッスン 2.4: サンプルのインストールと実行**

認証の構成が終了したら、サンプル・アプリケーションをインストールして実行できます。

#### **EmployeeData.jar ファイルの共有ライブラリーの作成:**



1. WebSphere Application Server 管理コンソールで、「共有ライブラリー」ページを開きます。「環境」 > 「共有ライブラリー」をクリックします。
2. 「セル」スコープを選択します。
3. 共有ライブラリーを作成します。「新規」をクリックします。「名前」に「EmployeeManagementLIB」を入力します。クラスパスに、EmployeeData.jar へのパスを入力します。例えば、*samples\_home/WASSecurity/EmployeeData.jar* です。
4. 「適用」をクリックします。

#### サンプルのインストール:

1. EmployeeManagement.ear ファイルをインストールします。
  - a. 「アプリケーション」 > 「新規アプリケーション」 > 「新規エンタープライズ・アプリケーション」をクリックして、インストールを開始します。アプリケーションをインストールする詳細なパスを選択します。
  - b. 「モジュールをサーバーにマップ」ステップで、appCluster クラスタを指定して、EmployeeManagementWeb モジュールをインストールします。
  - c. 「共有ライブラリーのマップ」ステップで、「EmployeeManagementWeb」モジュールを選択します。
  - d. 「Reference shared libraries」をクリックします。「EmployeeManagementLIB」ライブラリーを選択します。
  - e. webUser ロールを、「アプリケーションのレルム内のすべての認証済み」にマップします。
  - f. 「OK」をクリックします。

クライアントは、このクラスタ内の s1 サーバーと s2 サーバーで実行されません。

2. サンプルの XSDeployment.ear ファイルをインストールします。
  - a. 「アプリケーション」 > 「新規アプリケーション」 > 「新規エンタープライズ・アプリケーション」をクリックして、インストールを開始します。アプリケーションをインストールする詳細なパスを選択します。
  - b. 「モジュールをサーバーにマップ」ステップで、xsCluster クラスタを指定して、XSDeploymentWeb Web モジュールをインストールします。
  - c. 「共有ライブラリーのマップ」ステップで、「XSDeploymentWeb」モジュールを選択します。
  - d. 「Reference shared libraries」をクリックします。「EmployeeManagementLIB」ライブラリーを選択します。
  - e. 「OK」をクリックします。

このクラスタ内の xs1 サーバーと xs2 サーバーがコンテナ・サーバーをホスティングします。

3. デプロイメント・マネージャーを再始動します。デプロイメント・マネージャーが始動すると、カタログ・サーバーも始動します。デプロイメント・マネージャーの SystemOut.log ファイルを表示すると、eXtreme Scale サーバー・プロパティ・ファイルがロードされたことを示す次のメッセージを見ることができます。



CWOBJ0913I: 次のサーバー・プロパティ・ファイルがロードされました。  
/wxs\_samples/security/catServer2.props

4. xsCluster クラスタを再始動します。xsCluster が始動すると、XSDeployment アプリケーションが開始し、xs1 と xs2 サーバーのそれぞれでコンテナ・サーバーが開始します。xs1 サーバーと xs2 サーバーの SystemOut.log ファイルを調べると、サーバー・プロパティ・ファイルがロードされたことを示す次のメッセージが表示されます。

CWOBJ0913I: 次のサーバー・プロパティ・ファイルがロードされました。  
/wxs\_samples/security/server2.props

5. appClusters クラスタを再始動します。クラスタ appCluster が始動すると、EmployeeManagement アプリケーションも開始します。s1 サーバーと s2 サーバーの SystemOut.log ファイルを調べると、クライアント・プロパティ・ファイルがロードされたことを示す次のメッセージが表示されます。

CWOBJ0924I: クライアント・プロパティ・ファイル {0} がロードされました。

authenticationRetryCount、transportType、および clientCertificateAuthentication プロパティに関する警告メッセージは無視してかまいません。プロパティ・ファイルの中に値が指定されていないため、デフォルト値が使用されます。

WebSphere eXtreme Scale バージョン 7.0 を使用している場合は、クライアント・プロパティ・ファイルがロードされたことを示す CWOBJ9000I メッセージ (英語のみ) が表示されます。予期されるメッセージが表示されない場合は、JVM 引数に -Dobjectgrid.server.props または -Dobjectgrid.client.props プロパティを適切に構成したか確認してください。プロパティを確実に構成済みの場合、ダッシュ (-) が UTF 文字であるか確認してください。

#### サンプル・アプリケーションの実行:

1. management.jsp ファイルを実行します。Web ブラウザーで、  
`http://<your_servername>:<port>/EmployeeManagementWeb/management.jsp` にアクセスします。例えば、次のような URL を使用できます。  
`http://localhost:9080/EmployeeManagementWeb/management.jsp`
2. アプリケーションに認証を提供します。webUser ロールにマップしたユーザーの資格情報を入力します。デフォルトでは、このユーザー・ロールは、すべての認証済みユーザーにマップされます。ユーザー ID に admin1、パスワードに admin1 を入力します。従業員を表示、追加、更新、および削除するページが表示されます。
3. 従業員を表示します。「**Display an Employee**」をクリックします。E メール・アドレスに「emp1@acme.com」を入力し、「**Submit**」をクリックします。従業員が見つからないというメッセージが表示されます。
4. 従業員を追加します。「**Add an Employee**」をクリックします。E メール・アドレスに「emp1@acme.com」、名に「Joe」、姓に「Doe」と入力します。「**Submit**」をクリックします。emp1@acme.com アドレスを持つ従業員が追加されたというメッセージが表示されます。
5. 新しい従業員を表示します。「**Display an Employee**」をクリックします。E メール・アドレスに「emp1@acme.com」を入力し、姓と名のフィールドは空のままにして「**Submit**」をクリックします。従業員が見つかったというメッセージが表示され、名フィールドと姓フィールドに正しい名前が表示されます。

6. 従業員を削除します。「Delete an employee」をクリックします。  
「emp1@acme.com」を入力し、「Submit」をクリックします。従業員が削除されたというメッセージが表示されます。

#### レッスンのチェックポイント:

サンプル・アプリケーションをインストールして実行しました。このチュートリアルは WebSphere Application Server 統合を使用するため、クライアントが eXtreme Scale サーバーに対する認証に失敗する場合のシナリオは見ることができません。WebSphere Application Server に対するユーザー認証が成功すると、eXtreme Scale の認証も成功します。

## モジュール 3: トランスポート・セキュリティの構成

構成の中のクライアントとサーバー間のデータ転送をセキュアにするために、トランスポート・セキュリティを構成します。

前のチュートリアル・モジュールにおいて、WebSphere eXtreme Scale 認証を使用可能に設定しました。認証を使用可能に設定すると、WebSphere eXtreme Scale サーバーに接続を試みるすべてのアプリケーションは、資格情報の提供を要求されます。したがって、非認証クライアントは WebSphere eXtreme Scale サーバーに接続できません。クライアントは、WebSphere Application Server セルの中で実行される認証アプリケーションでなければなりません。

このモジュールまでの構成では、appCluster クラスター内のクライアントと xsCluster クラスター内のサーバー間のデータ転送は、暗号化されません。ご使用の WebSphere Application Server クラスターがファイアウォールで囲まれたサーバーにインストールされている場合は、この構成を受け入れることができます。しかし、シナリオによっては、たとえポートがファイアウォールで保護されるとしても、何らかの理由で、暗号化されていないトラフィックは、受け入れられない場合もあります。例えば、政府の方針で、暗号化トラフィックが強制される場合もあります。WebSphere eXtreme Scale は、ObjectGrid エンドポイント (クライアント・サーバー、コンテナ・サーバー、およびカタログ・サーバーが含まれる) 間のセキュア通信のために Transport Layer Security/Secure Sockets Layer (TLS/SSL) をサポートします。

このサンプル・デプロイメントでは、eXtreme Scale クライアント・サーバーとコンテナ・サーバーは、すべて WebSphere Application Server 環境で実行されます。eXtreme Scale トランスポート・セキュリティは Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) トランスポート設定によって管理されるため、クライアント・プロパティとサーバー・プロパティは、SSL 設定の構成には必要ありません。WebSphere eXtreme Scale サーバーは、このサーバーが実行されるアプリケーション・サーバーと同じオブジェクト・リクエスト・ブローカー (ORB) インスタンスを使用します。この CSIV2 トランスポート設定を使用して、WebSphere Application Server 構成内のクライアント・サーバーとコンテナ・サーバーに対してすべての SSL 設定を指定してください。カタログ・サーバーには、Internet Inter-ORB Protocol (IIOP) もリモート・メソッド呼び出し (RMI) も使用しない専有トランスポート・パスがあります。この専用のトランスポート・パスのために、カタログ・サーバーは、WebSphere Application Server CSIV2 トランスポート

ート設定では管理できません。したがって、カタログ・サーバーのサーバー・プロパティ・ファイルの中に、SSL プロパティを構成する必要があります。

## 学習目標

このモジュールのレッスンを完了すると、以下の作業の実行方法を理解できます。

- CSIv2 のインバウンド・トランスポートとアウトバウンド・トランスポートの構成。
- SSL プロパティのカタログ・サーバー・プロパティ・ファイルへの追加。
- ORB プロパティ・ファイルの確認。
- サンプルを実行します。

## 所要時間

このモジュールの所要時間は約 60 分です。

## 前提条件

このチュートリアル・ステップは、これまでのモジュールの上に組み立てられています。トランスポート・セキュリティーを構成する前に、このチュートリアルのこれまでのモジュールを完了してください。

## レッスン 3.1: CSIv2 のインバウンド・トランスポートとアウトバウンド・トランスポートの構成

サーバー・トランスポートの Transport Layer Security/Secure Sockets Layer (TLS/SSL) を構成するには、クライアント、カタログ・サーバー、およびコンテナ・サーバーをホストするすべての WebSphere Application Server サーバーに対して、Common Secure Interoperability Protocol Version 2 (CSIv2) インバウンド・トランスポートと CSIv2 アウトバウンド・トランスポートを SSL-Required に設定します。

チュートリアルで使用するサンプルのトポロジーでは、s1、s2、xs1、および xs2 アプリケーション・サーバーに対して、これらのプロパティを設定する必要があります。次のステップでは、構成内のすべてのサーバーのインバウンド・トランスポートとアウトバウンド・トランスポートを構成します。

管理コンソールで、インバウンド・トランスポートとアウトバウンド・トランスポートを設定します。管理セキュリティーが使用可能であることを確認します。

- **WebSphere Application Server バージョン 6.1** の場合: 「セキュリティー」 > 「セキュア管理」 > 「アプリケーション」 > 「RMI/IIOP セキュリティー」をクリックし、トランスポート・タイプを「**SSL 必須**」に変更します。
- **WebSphere Application Server バージョン 7.0** の場合: 「セキュリティー」 > 「グローバル・セキュリティー」 > 「RMI/IIOP セキュリティー」 > 「CSIv2 インバウンド通信」をクリックします。CSIv2 Transport Layer の下のトランスポート・タイプを「**SSL 必須**」に変更します。このステップを繰り返して、CSIv2 アウトバウンド通信を構成します。

中央で管理されるエンドポイント・セキュリティー設定を使用したり、SSL リポジトリを構成したりできます。詳しくは、Common Secure Interoperability バージョン 2 トランスポート・インバウンド設定を参照してください。

## レッスン 3.2: SSL プロパティのカタログ・サーバー・プロパティ・ファイルへの追加

カタログ・サーバーには、WebSphere Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) トランスポート設定が管理できない専用のトランスポート・パスがあります。したがって、カタログ・サーバーのサーバー・プロパティ・ファイルで Secure Sockets Layer (SSL) プロパティを構成する必要があります。

カタログ・サーバーには専用のトランスポート・パスがあるため、カタログ・サーバー・セキュリティーを構成するには、追加のステップが必要です。このトランスポート・パスは、Application Server CSIV2 トランスポート設定では管理できません。

1. `catServer2.props` ファイル内の SSL プロパティを編集します。カタログ・サーバー・セキュリティーを構成するには、カタログ・サーバー・プロパティ・ファイルの中の次の SSL プロパティのコメントを外します。このチュートリアルでは、カタログ・サーバー・プロパティは `catServer2.props` ファイルにあります。`keyStore` プロパティと `trustStore` プロパティを更新して、使用している環境の適切な場所を参照するようにします。

```
#alias=default
#contextProvider=IBMSSE2
#protocol=SSL
#keyStoreType=PKCS12
#keyStore=/<WAS_HOME>/IBM/WebSphere/AppServer/profiles/<DMGR_NAME>/config/cells/<CELL_NAME>/nodes/<NODE_NAME>/key.p12
#keyStorePassword=WebAS
#trustStoreType=PKCS12
#trustStore=/<WAS_HOME>/IBM/WebSphere/AppServer/profiles/<DMGR_NAME>/config/cells/<CELL_NAME>/nodes/<NODE_NAME>/trust.p12
#trustStorePassword=WebAS
#clientAuthentication=false
```

`catServer2.props` ファイルは、デフォルトの WebSphere Application Server ノード・レベルの鍵ストアとトラストストアを使用しています。より複雑なデプロイメント環境をデプロイする場合は、それにふさわしい鍵ストアとトラストストアを選択する必要があります。場合によっては、鍵ストアとトラストストアを作成し、他のサーバーの鍵ストアから鍵をインポートする必要があります。

WebSphere Application Server 鍵ストアおよびトラストストアのデフォルト・パスワードは `WebAS` スtring ですので、忘れないでください。詳しくは、デフォルト自己署名証明書の構成を参照してください。

2. `catServer2.props` ファイルの中の `transportType` プロパティの値を更新します。チュートリアルのこれまでのステップで、この値は `TCP/IP` に設定されています。この値を `SSL-Required` に変更します。
3. カatalog・サーバー・セキュリティー設定に対する変更をアクティブにするために、デプロイメント・マネージャーを再始動します。

### レッスンのチェックポイント:

カタログ・サーバーの SSL プロパティを構成しました。

### レッスン 3.3: サンプルの実行

すべてのサーバーを再始動し、再度、サンプル・アプリケーションを実行します。問題なくステップを実行できるはずです。

サンプル・アプリケーションの実行およびインストールについては、102ページの『レッスン 2.4: サンプルのインストールと実行』を参照してください。

#### レッスンのチェックポイント:

トランスポート・セキュリティーを使用可能に設定したサンプル・アプリケーションを実行しました。

## モジュール 4: WebSphere Application Server での Java 認証・承認サービス (JAAS) 許可の使用

クライアントの認証を構成したので、さまざまな許可を異なるユーザーに与えるために、さらに認証を構成することができます。例えば、オペレーター・ユーザーはデータ表示のみが可能である一方で、アドミニストレーター・ユーザーはすべての操作が実行可能であるなどです。

このチュートリアルの前のもジュールと同様に、クライアントを認証した後、eXtreme Scale 許可メカニズムによりセキュリティー特権を付与することができます。このチュートリアルの前のもジュールでは、WebSphere Application Server との統合を使用して、データ・グリッドの認証を使用可能にする方法について説明しました。結果として、非認証クライアントは、eXtreme Scale サーバーに接続したり、システムに要求を実行依頼したりすることができません。ただし、認証されている各クライアントは、ObjectGrid マップに格納されているデータの読み取り、書き込み、削除など、サーバーに対して同じアクセス権または特権を持っています。クライアントは、どのような照会でも実行できます。

チュートリアルこの部分では、eXtreme Scale 許可を使用して認証ユーザーにさまざまな特権を付与する方法について説明します。WebSphere eXtreme Scale はアクセス権ベースの許可メカニズムを使用します。さまざまな許可クラスによって表されるさまざまな許可カテゴリーを割り当てることができます。このモジュールでは、MapPermission クラスを取り上げます。使用できるすべての許可のリストについては、クライアント許可プログラミングを参照してください。

WebSphere eXtreme Scale では、

`com.ibm.websphere.objectgrid.security.MapPermission` クラスは eXtreme Scale リソース、特に `ObjectMap` インターフェースまたは `JavaMap` インターフェースのメソッドに対する許可を表しています。WebSphere eXtreme Scale は、`ObjectMap` および `JavaMap` のメソッドにアクセスするための以下の許可ストリングを定義します。

- **read:** マップからデータを読み取る許可を与えます。
- **write:** マップのデータを更新する許可を与えます。
- **insert:** マップにデータを挿入する許可を与えます。
- **remove:** マップからデータを削除する許可を与えます。
- **invalidate:** マップからのデータを無効にする許可を与えます。



- **all**: read、write、insert、remove、および invalidate に対するすべての許可を与えます。

許可は、eXtreme Scale クライアントがデータ・アクセス API (ObjectMap API、JavaMap API、EntityManager API など) を使用したときに発生します。メソッドが呼び出されるたびに、ランタイムは、対応するマップ許可を検査します。必要な許可がクライアントに与えられていない場合は、AccessControlException 例外になります。このチュートリアルでは、Java 認証・承認サービス (JAAS) 許可を使用して、さまざまなユーザーに対する許可マップ・アクセスを付与する方法について説明します。

## 学習目標

このモジュールのレッスンを完了すると、以下の作業の実行方法を理解できます。

- WebSphere eXtreme Scale の許可を使用可能にする。
- ユーザー・ベースの許可を使用可能にする。
- グループ・ベースの許可の構成。

## 所要時間

このモジュールの所要時間は約 60 分です。

## 前提条件

認証を構成する前に、このチュートリアルのこれまでのモジュールを完了する必要があります。

## レッスン 4.1: WebSphere eXtreme Scale 許可を使用可能にする

WebSphere eXtreme Scale で許可を使用可能にするには、特定の ObjectGrid のセキュリティを使用可能にする必要があります。

ObjectGrid で許可を使用可能にするには、XML ファイルで、その特定の ObjectGrid の **securityEnabled** 属性を true に設定する必要があります。このチュートリアルでは、既に objectGrid.xml ファイルの中でセキュリティが設定されている、*samples\_home*/WASSecurity ディレクトリー内の XSDeployment\_sec.ear ファイルを使用するか、既存の objectGrid.xml ファイルを編集してセキュリティを使用可能に設定するかのどちらかを行うことができます。このレッスンでは、ファイルを編集してセキュリティを使用可能にする方法を例示します。

1. XSDeployment.ear ファイル内のファイルを抽出してから、XSDeploymentWeb.war ファイルを unzip します。
2. objectGrid.xml ファイルを開いて、ObjectGrid レベルで **securityEnabled** 属性を true に設定します。次のサンプルでこの属性の例を参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
```

```
  <objectGrids>
    <objectGrid name="Grid" securityEnabled="true">
      <backingMap name="Map1" />
```

```
</objectGrid>
</objectGrids>
```

```
</objectGridConfig>
```

複数の ObjectGrid が定義されている場合は、各データ・グリッドでこの属性を設定する必要があります。

3. XSDeploymentWeb.war ファイルと XSDeployment.ear ファイルをパッケージ化し直して、変更を組み込みます。オリジナル・パッケージを上書きしないように、当該ファイルに XSDeployment\_sec.ear という名前を付けます。
4. 既存の XSDeployment アプリケーションをアンインストールし、XSDeployment\_sec.ear ファイルをインストールします。アプリケーションのデプロイについて詳しくは、102 ページの『レッスン 2.4: サンプルのインストールと実行』を参照してください。

#### レッスンのチェックポイント:

ObjectGrid のセキュリティーを使用可能にすることで、データ・グリッドの許可も使用可能にしました。

### レッスン 4.2: ユーザー・ベースの許可を使用可能にする

このチュートリアル認証モジュールで、operator1 および admin1 という 2 人のユーザーを作成しました。Java 認証・承認サービス (JAAS) 許可を使用して、これらのユーザーにさまざまな許可を割り当てることができます。

**ユーザー・プリンシパルを使用した、Java 認証・承認サービス (JAAS) 許可ポリシーの定義:**

前に作成したユーザーに、許可を割り当てることができます。operator1 ユーザーに、すべてのマップに対する読み取り許可のみを割り当てます。admin1 ユーザーに、すべての許可を割り当てます。JAAS 許可ポリシー・ファイルを使用して、プリンシパルに許可を付与します。

JAAS 許可ファイルを編集します。xsAuth2.policy ファイルは、*samples\_home/security* ディレクトリーにあります。

```
grant codebase http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction
Principal com.ibm.ws.security.common.auth.WSPrincipalImpl "defaultWIMFileBasedRealm/operator1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission "Grid.Map1", "read";
};
```

```
grant codebase http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction
Principal com.ibm.ws.security.common.auth.WSPrincipalImpl "defaultWIMFileBasedRealm/admin1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission "Grid.Map1", "all";
};
```

このファイルにある <http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction> コードベースは、ObjectGrid 用に特別に予約された URL です。プリンシパルに付与されているすべての ObjectGrid 許可では、この特別なコードベースを使用します。このファイルの中で、次の許可が割り当てられます。

- 最初の grant ステートメントは、read マップ許可を operator1 プリンシパルに付与します。operator1 ユーザーは、ObjectGrid インスタンスの中の Map1 マップに対するマップ読み取り許可のみを持ちます。



- 2 番目の grant ステートメントは、all マップ許可を admin1 プリンシパルに付与します。admin1 は、ObjectGrid インスタンスの中の Map1 マップに対するすべての許可を持ちます。
- プリンシパル名は defaultWIMFileBasedRealm/operator1 で、Operator1 ではありません。統合リポジトリがユーザー・アカウント・レジストリーとして使用されるときに、WebSphere Application Server は自動的にレルム名をプリンシパル名に追加します。必要であれば、この値を調整します。

### JVM プロパティを使用した JAAS 許可ポリシー・ファイルの設定:

次のステップを使用して、xsCluster クラスター内の xs1 サーバーと xs2 サーバーの JVM プロパティを設定します。このチュートリアルで使用するサンプル・トポロジとは異なるトポロジを使用する場合は、すべてのコンテナ・サーバーにファイルを設定してください。

1. 管理コンソールで、「サーバー」 > 「アプリケーション・サーバー」 > *server\_name* > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」をクリックします。
2. 次の汎用 JVM 引数を追加します。  
-Djava.security.auth.policy=samples\_home/security/xsAuth2.policy
3. 「OK」をクリックして、変更を保存します。

### サンプル・アプリケーションを実行して許可をテストする:

サンプル・アプリケーションを使用して、許可設定をテストすることができます。アドミニストレーター・ユーザーは、従業員の表示および追加も含めて、引き続き、Map1 マップ内のすべての許可を持ちます。オペレーター・ユーザーは、読み取り許可しか割り当てられていないため、従業員の表示のみが可能です。

1. コンテナ・サーバーを実行しているすべてのアプリケーション・サーバーを再始動します。
2. EmployeeManagementWeb アプリケーションを開きます。Web ブラウザーで、<http://<host>:<port>/EmployeeManagementWeb/management.jsp> を開きます。
3. アドミニストレーター・ユーザーとしてアプリケーションにログインします。ユーザー名に admin1、パスワードに admin1 を使用します。
4. 従業員を表示してみます。「Display an Employee」をクリックし、E メール・アドレス「authemp1@acme.com」を検索します。ユーザーが見つからないというメッセージが表示されます。
5. 従業員を追加します。「Add an Employee」をクリックします。E メール・アドレス「authemp1@acme.com」、名「Joe」、および姓「Doe」を追加し、「Submit」をクリックします。従業員が追加されたというメッセージが表示されます。
6. オペレーター・ユーザーとしてログインします。2 つ目の Web ブラウザー・ウィンドウを開いて、<http://<host>:<port>/EmployeeManagementWeb/management.jsp> を開きます。ユーザー名に operator1、パスワードに operator1 を使用します。
7. 従業員を表示してみます。「Display an Employee」をクリックし、E メール・アドレス「authemp1@acme.com」を検索します。従業員が表示されます。

- 従業員を追加します。「Add an Employee」をクリックします。E メール・アドレス「authemp2@acme.com」、名「Joe」、および姓「Doe」を追加し、「Submit」をクリックします。次のメッセージが表示されます。

An exception occurs when Add the employee. See below for detailed exception messages.

次の例外が例外チェーンに入っています。

```
java.security.AccessControlException: Access denied
(com.ibm.websphere.objectgrid.security.MapPermission Grid.Map1 insert)
```

このメッセージは、operator1 ユーザーが Map1 マップへのデータ挿入を許可されていないために表示されます。

バージョン 7.0.0.11 より前のバージョンの WebSphere Application Server で実行している場合、コンテナ・サーバーで java.lang.StackOverflowError エラーが表示されることがあります。このエラーの原因は IBM Developer Kit の問題です。この問題は、WebSphere Application Server バージョン 7.0.0.11 以上に同梱されている IBM Developer Kit では修正済みです。

#### レッスンのチェックポイント:

このレッスンでは、特定のユーザーに許可を割り当てて、許可を構成しました。

### レッスン 4.3: グループ・ベースの許可の構成

前回のレッスンでは、Java 認証・承認サービス (JAAS) 許可ポリシーを使用して、個々のユーザー・ベースの許可をユーザー・プリンシパルに割り当てました。しかし、数百または数千のユーザーがある場合、個々のユーザーではなくグループに基づいてアクセス権限を付与する、グループ・ベースの許可を使用します。

残念ながら、WebSphere Application Server から認証される Subject オブジェクトは、ユーザー・プリンシパルしか含みません。このオブジェクトは、グループ・プリンシパルを含みません。カスタム・ログイン・モジュールを追加することによって、Subject オブジェクトにグループ・プリンシパルを取り込むことができます。

このチュートリアルでは、カスタム・ログイン・モジュールの名前は com.ibm.websphere.samples.objectgrid.security.lm.WASAddGroupLoginModule です。このモジュールは groupLM.jar ファイルの中にあります。この JAR ファイルを、WAS-INSTALL/lib/ext ディレクトリ内に置きます。

WASAddGroupLoginModule は、WebSphere Application Server サブジェクトから公開グループ資格情報を取得し、com.ibm.websphere.samples.objectgrid.security.WSGroupPrincipal というグループ・プリンシパルを作成してそのグループを表します。その結果、このグループ・プリンシパルをグループ許可のために使用できます。グループは、xsAuthGroup2.policy ファイルの中で定義されます。

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
principal com.ibm.websphere.sample.xs.security.WSGroupPrincipal
    "defaultWIMFileBasedRealm/cn=operatorGroup,o=defaultWIMFileBasedRealm" {
    permission com.ibm.websphere.objectgrid.security.MapPermission "Grid.Map1", "read";
};

grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
```

```
principal com.ibm.websphere.sample.xs.security.WSGroupPrincipal
"defaultWIMFileBasedRealm/cn=adminGroup,o=defaultWIMFileBasedRealm" {
    permission com.ibm.websphere.objectgrid.security.MapPermission "Grid.Map1", "all";
};
```

プリンシパル名は `WSGroupPrincipal` で、これはグループを表します。

### カスタム・ログイン・モジュールの追加:

次のシステム・ログイン・モジュール・エントリーのそれぞれにカスタム・ログイン・モジュールを追加する必要があります。Lightweight Third Party Authentication (LTPA) を使用する場合は、そのエントリーを `RMI_INBOUND` ログイン・モジュールに追加してください。LTPA は、WebSphere Application Server バージョン 7.0 のデフォルトの認証メカニズムです。WebSphere Application Server Network Deployment 構成の場合は、LTPA 認証メカニズム構成エントリーを構成すれば十分です。

次のステップを使用して、提供された

`com.ibm.websphere.samples.objectgrid.security.lm.WASAddGroupLoginModule` ログイン・モジュールを構成します。

1. 管理コンソールで、「セキュリティ」 > 「グローバル・セキュリティ」 > 「Java 認証・承認サービス」 > 「システム・ログイン」 > *login\_module\_name* > 「JAAS ログイン・モジュール」 > 「新規」をクリックします。
2. クラス名として `com.ibm.websphere.sample.xs.security.lm.WASAddGroupLoginModule` を入力します。
3. オプション: プロパティ `debug` を追加し、値を `true` に設定します。
4. 「適用」をクリックして、新規モジュールをログイン・モジュール・リストに追加します。

### JVM プロパティを使用した JAAS 許可ポリシー・ファイルの設定:

管理コンソールで、`xsCluster` 内の `xs1` サーバーと `xs2` サーバーに対して次のステップを実行します。別のデプロイメント・トポロジーを使用している場合は、コンテナ・サーバーをホストするアプリケーション・サーバーに対して次のステップを実行します。

1. 管理コンソールで、「サーバー」 > 「アプリケーション・サーバー」 > *server\_name* > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」をクリックします。
2. 次の汎用 JVM 引数を入力するか、`-Djava.security.auth.policy` エントリーを次のテキストで置き換えます。  
`-Djava.security.auth.policy=samples_home/security/xsAuthGroup2.policy`
3. 「OK」をクリックして、変更を保存します。

### サンプル・アプリケーションによるグループ許可のテスト:

サンプル・アプリケーションを使用して、ログイン・モジュールで構成されたグループ許可をテストできます。

1. コンテナ・サーバーを再始動します。このチュートリアルでは、コンテナ・サーバーは、`xs1` サーバーおよび `xs2` サーバーです。

2. サンプル・アプリケーションにログインします。 Web ブラウザーで、  
`http://<host>:<port>/EmployeeManagementWeb/management.jsp` を開き、ユーザー名を `admin1`、パスワードを `admin1` でログインします。
3. 従業員を表示します。「**Display an Employee**」をクリックして、E メール・アドレス `authemp2@acme.com` を検索します。ユーザーが見つからないというメッセージが表示されます。
4. 従業員を追加します。「**Add an Employee**」をクリックします。 E メール・アドレス「`authemp2@acme.com`」、名「`Joe`」、および姓「`Doe`」を追加し、「**Submit**」をクリックします。従業員が追加されたというメッセージが表示されます。
5. オペレーター・ユーザーとしてログインします。 2 つ目の Web ブラウザー・ウィンドウを開いて、URL `http://<host>:<port>/EmployeeManagementWeb/management.jsp` を開きます。ユーザー名に `operator1`、パスワードに `operator1` を使用します。
6. 従業員を表示してみます。「**Display an Employee**」をクリックして、E メール・アドレス `authemp2@acme.com` を検索します。従業員が表示されます。
7. 従業員を追加します。「**Add an Employee**」をクリックします。 E メール・アドレス「`authemp3@acme.com`」、名「`Joe`」、および姓「`Doe`」を追加し、「**Submit**」をクリックします。次のメッセージが表示されます。

An exception occurs when Add the employee. See below for detailed exception messages.

次の例外が例外チェーンに入っています。

```
java.security.AccessControlException: Access denied
(com.ibm.websphere.objectgrid.security.MapPermission Grid.Map1 insert)
```

オペレーター・ユーザーには、データを `Map1` マップに挿入する許可がないため、このメッセージが表示されます。

#### レッスンのチェックポイント:

アプリケーションのユーザーに対する許可の割り当てを簡単にするために、グループを構成しました。

## モジュール 5: データ・グリッドとマップのモニターのための `xscmd` ツールの使用

`xscmd` ツールを使用して、Grid データのプライマリー・データ・グリッドとマップ・サイズを表示できます。`xscmd` ツールは MBean を使用して、プライマリー断片、レプリカ断片、コンテナ・サーバー、マップ・サイズなどのすべてのデータ・グリッド成果物を照会します。

このチュートリアルでは、コンテナおよびカタログ・サーバーは、WebSphere Application Server アプリケーション・サーバーの中で実行中です。WebSphere eXtreme Scale ランタイムは、Managed Bean (MBean) を、WebSphere Application Server ランタイムによって作成される MBean サーバーに登録します。`xscmd` ツールが使用するセキュリティは、WebSphere Application Server MBean セキュリティによって提供されます。したがって、WebSphere eXtreme Scale 固有のセキュリティ構成は必要ありません。

1. コマンド行ツールを使用して、`DMGR_PROFILE/bin` ディレクトリーを開きます。

2. **xscmd** ツールを実行します。

**-c listObjectGridPlacement -sf P** コマンドを使用して、プライマリー断片の配置をリストします。 Linux UNIX

```
xscmd.sh -g Grid -ms mapSet -c showPlacement -sf P
```

Windows

```
xscmd.bat -g Grid -ms mapSet -c showPlacement -sf P
```

出力を表示する前に、WebSphere Application Server の ID およびパスワードを使用してログインするように促すプロンプトが出されます。

## レッスンのチェックポイント

WebSphere Application Server の中で、**xscmd** ツールを使用しました。

---

## チュートリアル: 混合環境で WebSphere eXtreme Scale セキュリティーを外部オーセンティケーターと統合する

このチュートリアルでは、WebSphere Application Server 環境に部分的にデプロイされる WebSphere eXtreme Scale サーバーを保護する方法を例示します。

このチュートリアルのデプロイメントでは、コンテナ・サーバーは WebSphere Application Server 内にデプロイされます。カタログ・サーバーはスタンドアロン・サーバーとしてデプロイされ、Java Standard Edition (Java SE) 環境内で開始されます。

カタログ・サーバーは WebSphere Application Server 内にデプロイされないため、WebSphere Application Server 認証プラグインを使用することはできません。WebSphere Application Server 認証プラグインを構成するプロセスの詳細については、90 ページの『チュートリアル: WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合』を参照してください。このチュートリアルでは、カタログ・サーバー認証用に別のオーセンティケーターが必要です。鍵ストア・オーセンティケーターを構成して、クライアントを認証します。

### 学習目標

このチュートリアルの学習目標は次のとおりです。

- WebSphere eXtreme Scale を構成して、KeyStoreLoginAuthenticator プラグインを使用する。
- WebSphere eXtreme Scale トランスポート・セキュリティを構成して、WebSphere Application Server CSIv2 構成と WebSphere eXtreme Scale プロパティ・ファイルを使用する。
- WebSphere Application Server の Java 認証・承認サービス (JAAS) 許可を使用する。
- **xscmd** ユーティリティーを使用して、チュートリアルで作成したデータ・グリッドやマップをモニターする。

## 所要時間

このチュートリアルは、開始してから終了するまで約 4 時間かかります。

## 概要: 混合環境のセキュリティー

このチュートリアルでは、混合環境内の WebSphere eXtreme Scale セキュリティーを統合します。コンテナ・サーバーは WebSphere Application Server 内で稼働し、カタログ・サービスはスタンドアロン・モードで稼働します。カタログ・サーバーはスタンドアロン・モードであるため、外部オーセンティケーターを構成しなければなりません。

**重要:** コンテナ・サーバーとカタログ・サーバーの両方を WebSphere Application Server 内で実行する場合は、WebSphere Application Server 認証プラグインを使用しても、外部オーセンティケーターを使用してもかまいません。WebSphere Application Server 認証プラグインの使用については、90 ページの『チュートリアル: WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合』を参照してください。

## 学習目標

このチュートリアルの学習目標は次のとおりです。

- WebSphere eXtreme Scale を構成して、KeyStoreLoginAuthenticator プラグインを使用する。
- WebSphere eXtreme Scale トランスポート・セキュリティーを構成して、WebSphere Application Server CSIv2 構成と WebSphere eXtreme Scale プロパティ・ファイルを使用する。
- WebSphere Application Server の Java 認証・承認サービス (JAAS) 許可を使用する。
- `xscmd` ユーティリティーを使用して、チュートリアルで作成したデータ・グリッドやマップをモニターする。

## 所要時間

このチュートリアルは、開始してから終了するまで約 4 時間かかります。

## スキル・レベル

中級

## 対象者

WebSphere eXtreme Scale と WebSphere Application Server 間のセキュリティー統合および外部オーセンティケーターの構成に関心がある開発者および管理者

## システム要件

- WebSphere Application Server バージョン 6.1 または次のフィックスが適用されたバージョン 7.0.0.11 以上: インテリム・フィックス PM20613 およびインテリム・フィックス PM15818。
- カatalog・サーバーは、WebSphere Application Server と統合されるインストール環境でなく、スタンドアロン・インストール環境で実行しなければなりません。



- 次のフィックスを適用して、Java ランタイムを更新してください。IZ79819: IBMJDK FAILS TO READ PRINCIPAL STATEMENT WITH WHITESPACE FROM SECURITY FILE
- カタログ・サービスを実行するスタンドアロン・ノードは IBM Software Development Kit バージョン 1.6 J9 を使用する必要があります。この Software Development Kit は WebSphere Application Server インストールに組み込まれています。WebSphere Application Server 上の WebSphere eXtreme Scale インストール済み環境内では **startOgServer** コマンドを実行できないため、カタログ・サーバー・ノードはスタンドアロン・インストールにしなければなりません。

このチュートリアルでは、4 つのアプリケーション・サーバー (WebSphere Application Server) と 1 つのデプロイメント・マネージャーを使用してサンプル・デモを行います。

## 前提条件

このチュートリアルを開始するにあたって、次の項目についての基本的な知識があると便利です。

- WebSphere eXtreme Scale プログラミング・モデル
- 基本的な WebSphere eXtreme Scale セキュリティーの概念
- 基本的な WebSphere eXtreme Scale セキュリティーの概念

WebSphere eXtreme Scale と WebSphere Application Server のセキュリティー統合のバックグラウンド情報については、596 ページの『WebSphere Application Server とのセキュリティー統合』を参照してください。

## モジュール 1: WebSphere Application Server とスタンドアロンとの混合環境の準備

チュートリアルを開始する前に、WebSphere Application Server 内で稼働するコンテナ・サーバーを組み込む基本的なトポロジーを作成する必要があります。このチュートリアルでは、カタログ・サーバーはスタンドアロン・モードで稼働します。

### 学習目標

このモジュールのレッスンでは、以下の方法について学習します。

- チュートリアルに必要な混合トポロジーとファイルについて理解する。
- コンテナ・サーバーを実行する WebSphere Application Server を構成する。

### 所要時間

このモジュールの所要時間は約 60 分です。

### レッスン 1.1: トポロジーの理解とチュートリアル・ファイルの入手

チュートリアル用の環境を準備するには、トポロジーのカタログ・サーバーとコンテナ・サーバーを構成する必要があります。

このレッスンでは、チュートリアルで使用するサンプル・トポロジーとアプリケーションを紹介합니다。チュートリアルの実行を開始するには、アプリケーションを



ダウンロードし、環境内の正しい場所に構成ファイルを配置する必要があります。  
サンプル・アプリケーションは [WebSphere eXtreme Scale wiki](#) からダウンロード  
できます。

**トポロジー:** このチュートリアルでは、WebSphere Application Server セル内に次  
のクラスターを作成します。

- **appCluster クラスター:** EmployeeManagement サンプル・エンタープライズ・ア  
プリケーションをホストします。このクラスターには、s1 と s2 の 2 つのアプ  
リケーション・サーバーがあります。
- **xsCluster クラスター:** eXtreme Scale コンテナ・サーバーをホストします。こ  
のクラスターには、xs1 と xs2 の 2 つのアプリケーション・サーバーがありま  
す。

このデプロイメント・トポロジーでは、s1 および s2 のアプリケーション・サーバ  
ーは、データ・グリッドに保管されたデータにアクセスするクライアント・サーバ  
ーです。xs1 サーバーと xs2 サーバーは、データ・グリッドをホストするコンテナ  
ー・サーバーです。

**代替の構成:** すべてのアプリケーション・サーバーを、単一のクラスター内で (例え  
ば appCluster クラスター内で) ホストすることができます。この構成では、クラス  
ター内のすべてのサーバーがクライアントとコンテナ・サーバーの両方を兼ねま  
す。このチュートリアルでは、2 つのクラスターを使用して、クライアントをホス  
トしているアプリケーション・サーバーとコンテナ・サーバーをホストしている  
アプリケーション・サーバーを区別しています。

このチュートリアルでは、WebSphere Application Server セルに含まれないリモ  
ート・サーバーで構成されるカタログ・サービス・ドメインを構成します。この構成  
はデフォルト構成ではなく、カタログ・サーバーはデプロイメント・マネージャー  
上で稼働し、その他のプロセスは WebSphere Application Server セル内で稼働する  
こととなります。リモート・サーバーで構成されるカタログ・サービス・ドメイン  
の作成について詳しくは、288 ページの『WebSphere Application Server でのカタロ  
グ・サービス・ドメインの作成』を参照してください。

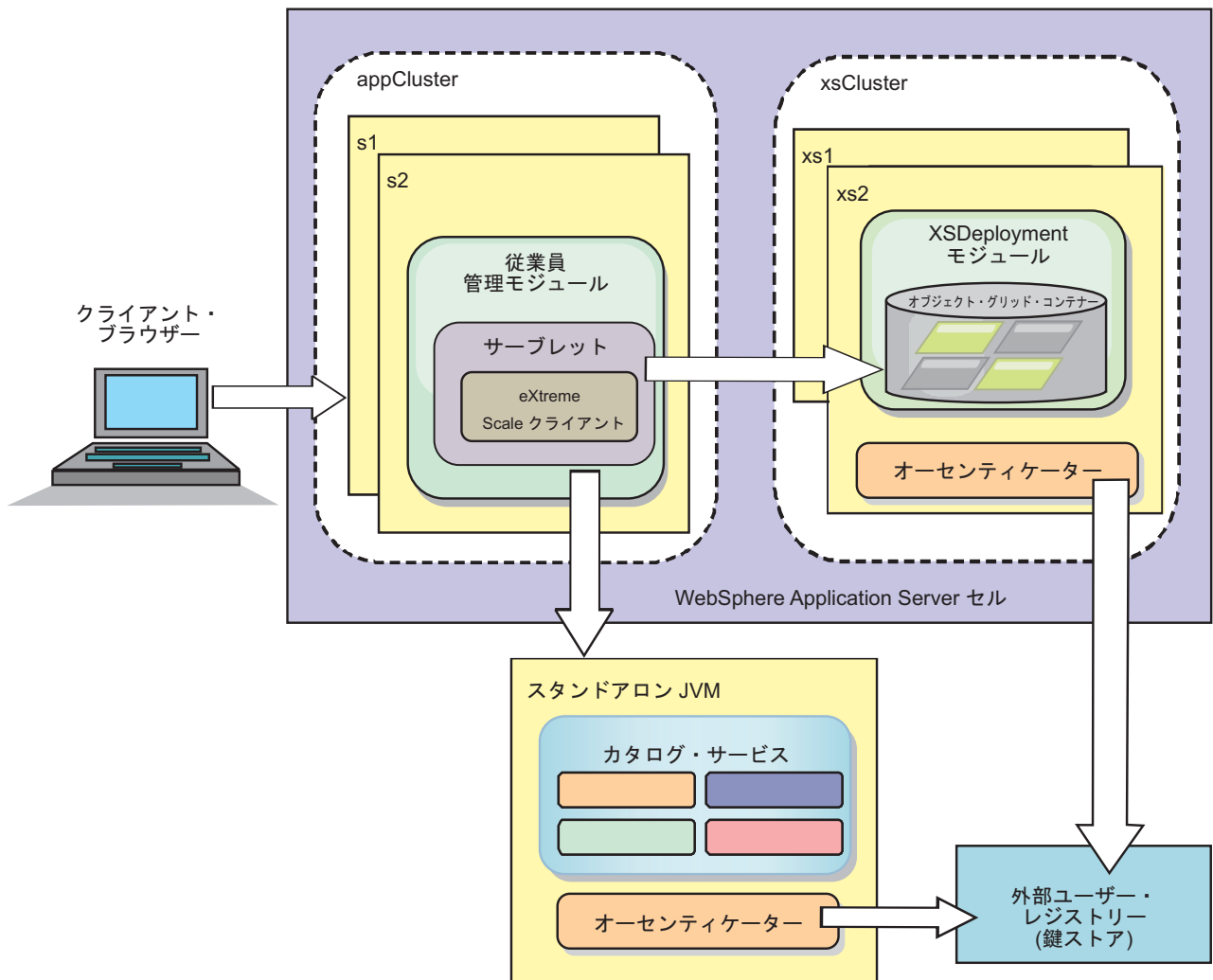


図 20. チュートリアル・トポロジー

**アプリケーション:** このチュートリアルでは、2 つのアプリケーションと 1 つの共有ライブラリー・ファイルを使用します。

- **EmployeeManagement.ear:** EmployeeManagement.ear アプリケーションは、単純化された Java 2 Platform, Enterprise Edition (J2EE) エンタープライズ・アプリケーションです。これには、従業員プロフィールを管理するための Web モジュールが含まれます。Web モジュールには、コンテナ・サーバーに保管された従業員プロフィールを表示、挿入、更新、および削除する management.jsp ファイルが含まれます。
- **XSDeployment.ear:** このアプリケーションにはエンタープライズ・アプリケーション・モジュールが含まれ、アプリケーション成果物は含まれません。キャッシュ・オブジェクトは EmployeeData.jar ファイルにパッケージ化されます。EmployeeData.jar ファイルは、XSDeployment.ear ファイルがクラスにアクセスできるように、XSDeployment.ear ファイルの共有ライブラリーとしてデプロイされます。このアプリケーションの目的は、eXtreme Scale 構成ファイルとプロパティ・ファイルをパッケージ化することです。このエンタープライズ・アプリケーションが開始されると、eXtreme Scale ランタイムによって eXtreme Scale 構成

ファイルが自動的に検出され、その結果コンテナ・サーバーが作成されます。これらの構成ファイルには、`objectGrid.xml` と `objectGridDeployment.xml` ファイルが含まれます。

- **EmployeeData.jar:** この JAR ファイルは `com.ibm.websphere.sample.xs.data.EmployeeData` クラスという 1 つのクラスを含んでいます。このクラスは、グリッドに保管される従業員データを表します。この Java アーカイブ (JAR) ファイルは、共有ライブラリーとして `EmployeeManagement.ear` および `XSDeployment.ear` ファイルと一緒にデプロイされます。

#### チュートリアル・ファイルの入手:

1. `WASSecurity.zip` および `security_extauth.zip` ファイルを WebSphere eXtreme Scale wiki からダウンロードします。
2. バイナリー成果物やソース成果物を表示するために `WASSecurity.zip` ファイルを、例えば、`wxs_samples/` ディレクトリーなどのディレクトリーに解凍します。今後、チュートリアルの中ではこのディレクトリーを `samples_home` と呼びます。内容の説明やソースを Eclipse ワークスペースにロードする方法については、パッケージ内の `README.txt` ファイルを参照してください。次の `ObjectGrid` 構成ファイルは `META-INF` ディレクトリーにあります。
  - `objectGrid.xml`
  - `objectGridDeployment.xml`
3. この環境を保護するために使用するプロパティ・ファイルを保管するディレクトリーを作成します。例えば、`/opt/wxs/security` ディレクトリーを作成します。
4. `security_extauth.zip` ファイルを `samples_home` に解凍します。`security_extauth.zip` ファイルには、このチュートリアルで使用される次のセキュリティ構成ファイルが含まれています。構成ファイルは次のとおりです。
  - `catServer3.props`
  - `server3.props`
  - `client3.props`
  - `security3.xml`
  - `xsAuth3.props`
  - `xsjaas3.config`
  - `sampleKS3.jks`

#### 構成ファイルについて:

`objectGrid.xml` ファイルと `objectGridDeployment.xml` ファイルは、アプリケーション・データを保管するデータ・グリッドとマップを作成します。

これらの構成ファイルには、`objectGrid.xml` と `objectGridDeployment.xml` という名前を付ける必要があります。アプリケーション・サーバーが始動すると、eXtreme Scale は、EJB および Web モジュールの `META-INF` ディレクトリーで、これらのファイルを検出します。これらのファイルが検出された場合、Java 仮想マシン (JVM) は構成ファイルの中に定義されたデータ・グリッドのコンテナ・サーバーとして機能するとみなされます。

## objectGrid.xml ファイル

objectGrid.xml ファイルは、Grid という名前の ObjectGrid を 1 つ定義します。Grid データ・グリッドには、アプリケーションの従業員プロファイルを保管する 1 つの Map1 というマップがあります。

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploy/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="Grid" txTimeout="15">
      <backingMap name="Map1" />
    </objectGrid>
  </objectGrids>

</objectGridConfig>
```

## objectGridDeployment.xml ファイル

objectGridDeployment.xml ファイルは、Grid データ・グリッドのデプロイ方法を指定します。グリッドがデプロイされると、グリッドは 5 つの区画と 1 つの同期レプリカを持ちます。

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="Grid">
    <mapSet name="mapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="1" >
      <map ref="Map1"/>
    </mapSet>
  </objectgridDeployment>

</deploymentPolicy>
```

### レッスンのチェックポイント:

このレッスンでは、チュートリアル用のトポロジーについて学習し、構成ファイルとサンプル・アプリケーションを環境に追加しました。

## レッスン 1.2: WebSphere Application Server 環境の構成

チュートリアル用の環境を準備するには、WebSphere Application Server セキュリティーを構成する必要があります。内部ファイル・ベースの統合リポジトリをユーザー・アカウント・レジストリーとして使用して、管理セキュリティーおよびアプリケーション・セキュリティーを使用可能にします。その後、クライアント・アプリケーションとコンテナ・サーバーをホスティングするサーバー・クラスターを作成できます。カタログ・サーバーも作成して開始する必要があります。

次のステップは、WebSphere Application Server バージョン 7.0 を使用した記述になっています。しかし、考え方は、それより前の WebSphere Application Server バージョンにも適用できます。

### WebSphere Application Server セキュリティーの構成:

デプロイメント・マネージャー用のプロファイルと WebSphere eXtreme Scale の各ノード用のプロファイルを作成し、拡張します。詳しくは、205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。

WebSphere Application Server セキュリティーを構成します。

1. WebSphere Application Server 管理コンソールで、「**セキュリティ**」 > 「**グローバル・セキュリティ**」をクリックします。
2. 「**ユーザー・アカウント・リポジトリ**」に「**統合リポジトリ**」を選択します。「**現在の値で設定**」をクリックします。
3. 「**構成..**」をクリックして、「**統合リポジトリ**」パネルに進みます。
4. 「**1 次管理ユーザー名**」を入力します。例えば、admin です。「**適用**」をクリックします。
5. プロンプトが表示されたら、管理ユーザー・パスワードを指定して、「**OK**」をクリックします。変更を保存します。
6. 「**グローバル・セキュリティ**」ページで、「**統合リポジトリ**」設定が、現行ユーザー・アカウント・レジストリーに設定されていることを確認します。
7. 「**管理セキュリティを使用可能にする**」、「**アプリケーション・セキュリティを使用可能にする**」、および「**Java 2 セキュリティーを使用して、アプリケーション・アクセスをローカル・リソースに制限する**」の項目を選択します。「**適用**」をクリックして、変更を保存します。
8. デプロイメント・マネージャーを再始動し、実行中のアプリケーションがあれば再開します。

ユーザー・アカウント・レジストリーとして内部ファイル・ベースの統合リポジトリを使用して、WebSphere Application Server 管理セキュリティが使用可能になりました。

#### サーバー・クラスターの作成:

WebSphere Application Server 構成の中に、次の 2 つのサーバー・クラスターを作成します。appCluster クラスターはチュートリアルサンプル・アプリケーションをホストし、xsCluster クラスターはデータ・グリッドをホストします。

1. WebSphere Application Server 管理コンソールで、クラスターのパネルを開きます。「**サーバー**」 > 「**クラスター**」 > 「**WebSphere Application Server クラスター**」 > 「**新規**」をクリックします。
2. クラスター名に「appCluster」を入力し、「**ローカルを優先**」オプションを選択したままにして、「**次へ**」をクリックします。
3. クラスターの中にサーバーを作成します。デフォルト・オプションのままにして、s1 という名前のサーバーを作成します。追加の s2 という名前のクラスター・メンバーを追加します。
4. ウィザードの残りのステップを完了して、クラスターを作成します。変更を保存します。
5. 上記のステップを繰り返して、xsCluster クラスターを作成します。このクラスターには、xs1 および xs2 という名前の 2 つのサーバーが含まれています。

#### カタログ・サービス・ドメインの作成:

サーバー・クラスターとセキュリティの構成が終了したら、カタログ・サーバーを開始する場所を定義する必要があります。

#### WebSphere eXtreme Scale のカタログ・サービス・ドメインの定義

1. WebSphere Application Server 管理コンソールで、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」をクリックします。
2. カatalog・サービス・ドメインを作成します。「新規」をクリックします。catalogService1 という名前でカatalog・サービス・ドメインを作成し、このカatalog・サービス・ドメインをデフォルトとして使用可能にします。
3. リモート・サーバーをカatalog・サービス・ドメインに追加します。「リモート・サーバー」を選択します。カatalog・サーバーを実行するホスト名を指定します。このサンプルの場合、リスナー・ポート値 16809 を使用します。
4. 「OK」をクリックして、変更を保存します。

#### レッスンのチェックポイント:

WebSphere Application Server のセキュリティーを使用可能にし、WebSphere eXtreme Scale のサーバー・トポロジーを作成しました。

## モジュール 2: 混合環境での WebSphere eXtreme Scale 認証の構成

認証を構成することで、要求側の ID を確実に判断できます。WebSphere eXtreme Scale は、クライアントとサーバー間の認証も、サーバー相互間の認証もともにサポートします。

#### 認証フロー

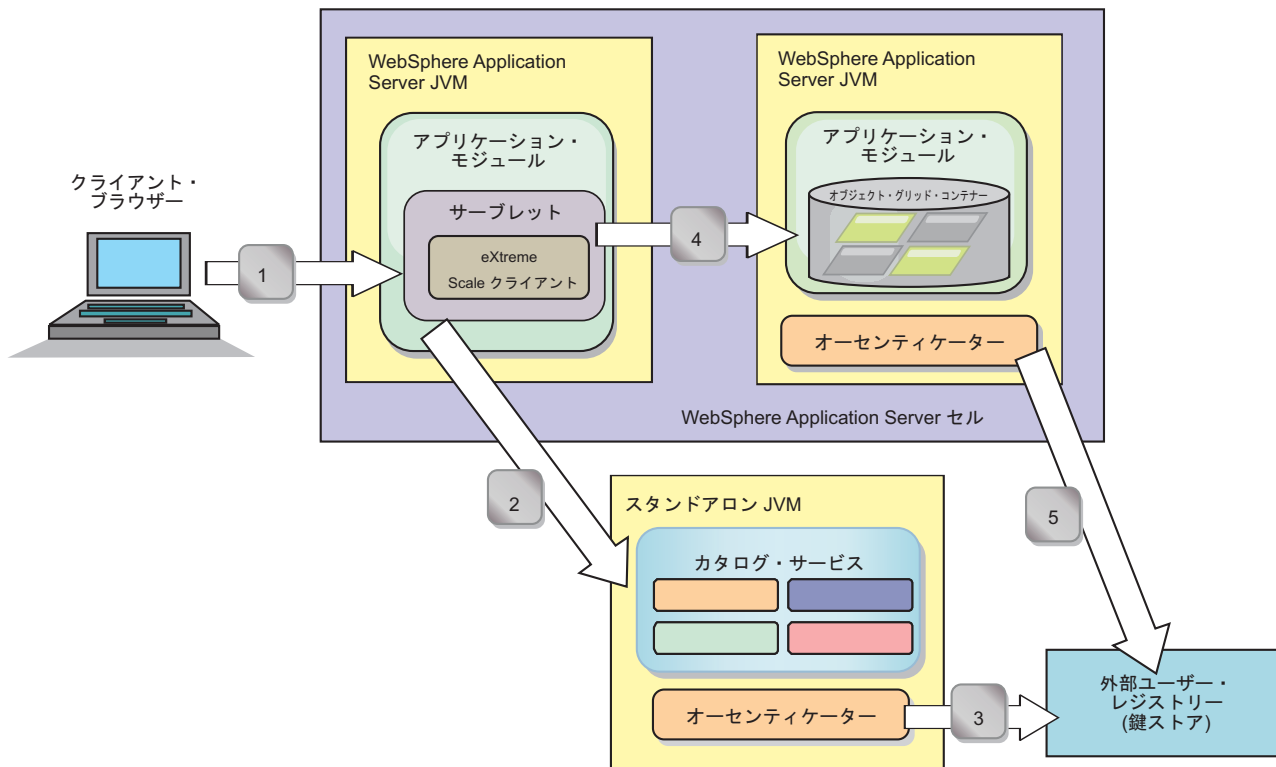


図 21. 認証フロー



直前のダイアグラムには 2 つのアプリケーション・サーバーがあります。最初のアプリケーション・サーバーは、WebSphere eXtreme Scale クライアントでもある Web アプリケーションをホスティングします。2 番目のアプリケーション・サーバーは、コンテナ・サーバーをホスティングします。カタログ・サーバーは、WebSphere Application Server でなくスタンドアロンの Java 仮想マシン (JVM) 内で実行されます。

ダイアグラム内の番号付き矢印は認証の流れを示します。

1. エンタープライズ・アプリケーション・ユーザーが Web ブラウザーにアクセスし、ユーザー名とパスワードを指定して最初のアプリケーション・サーバーにログインします。最初のアプリケーション・サーバーは、ユーザー・レジストリーでの認証のために、クライアントのユーザー名とパスワードをセキュリティー・インフラストラクチャーに送信します。このユーザー・レジストリーは鍵ストアです。結果として、セキュリティー情報が WebSphere Application Server スレッドに保管されます。
2. JavaServer Pages (JSP) ファイルが WebSphere eXtreme Scale クライアントとして機能して、クライアント・プロパティー・ファイルからセキュリティー情報を取得します。WebSphere eXtreme Scale クライアントとして機能する JSP アプリケーションは、要求と一緒に WebSphere eXtreme Scale クライアントのセキュリティー資格情報をカタログ・サーバーに送信します。要求とセキュリティー資格情報を一緒に送信すると、*runAs* モデルと見なされます。runAs モデルでは、Web ブラウザー・クライアントが WebSphere eXtreme Scale クライアントとして稼働して、コンテナ・サーバーに保管されているデータにアクセスします。クライアントは、Java 仮想マシン (JVM) レベルのクライアント資格情報を使用して WebSphere eXtreme Scale サーバーに接続します。runAs モデルの使用は、データ・ソース・レベルのユーザー ID とパスワードでデータベースに接続するのと似ています。
3. カタログ・サーバーが WebSphere eXtreme Scale クライアント資格情報を受け取ります。これには、WebSphere Application Server セキュリティー・トークンが含まれています。次に、カタログ・サーバーはクライアント資格情報の認証のためにオーセンティケーター・プラグインを呼び出します。オーセンティケーターは外部ユーザー・レジストリーに接続し、認証のためにクライアント資格情報をユーザー・レジストリーに送信します。
4. クライアントがユーザー ID とパスワードをアプリケーション・サーバー内でホスティングされるコンテナ・サーバーに送信します。
5. アプリケーション・サーバー内でホスティングされるコンテナ・サービスが、ユーザー ID とパスワードがペアになった WebSphere eXtreme Scale クライアント資格情報を受け取ります。次に、コンテナ・サーバーはクライアント資格情報の認証のためにオーセンティケーター・プラグインを呼び出します。オーセンティケーターは鍵ストアのユーザー・レジストリーに接続し、認証のためにクライアント資格情報をユーザー・レジストリーに送信します。

## 学習目標

このモジュールのレッスンでは、以下の方法について学習します。

- WebSphere eXtreme Scale クライアント・セキュリティーを構成する。
- WebSphere eXtreme Scale カタログ・サーバー・セキュリティーを構成する。

- WebSphere eXtreme Scale コンテナ・サーバー・セキュリティーを構成する。
- サンプル・アプリケーションをインストールして実行する。

## 所要時間

このモジュールの所要時間は約 60 分です。

## レッスン 2.1: WebSphere eXtreme Scale クライアント・セキュリティーの構成

クライアント・プロパティはプロパティ・ファイルを使用して構成します。クライアント・プロパティ・ファイルで、使用する `CredentialGenerator` 実装クラスを指示します。

### クライアント・プロパティ・ファイルの内容:

このチュートリアルでは、クライアント資格情報に WebSphere Application Server セキュリティー・トークンを使用します。 `samples_home/security_extauth` ディレクトリーには `client3.props` ファイルがあります。

`client3.props` ファイルは、次の設定を含んでいます。

#### **securityEnabled**

WebSphere eXtreme Scale クライアント・セキュリティーを使用可能にします。値を `true` に設定して、クライアントが有効なセキュリティー情報をサーバーに送信する必要があることを示します。

#### **credentialAuthentication**

クライアントの資格情報認証のサポートを指定します。値を `Supported` に設定して、クライアントが資格情報認証をサポートすることを示します。

#### **credentialGeneratorClass**

`com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator` インターフェースを実装するクラスの名前を指定します。値を `com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator` クラスに設定して、クライアントが `UserPasswordCredentialGenerator` クラスからセキュリティー情報を取得できるようにします。

#### **credentialGeneratorProps**

ユーザー名とパスワード (`manager manager1`) を指定します。ユーザー名は `manager` で、パスワードは `manager1` です。 `FilePasswordEncoder.bat|sh` コマンドを使用して、排他 OR (`xor`) アルゴリズムを使用してこのプロパティをエンコードすることもできます。

### Java 仮想マシン (JVM) プロパティを使用したクライアント・プロパティ・ファイルの設定:

管理コンソールで、`appCluster` クラスター内の `s1` サーバーと `s2` サーバーのそれぞれに対し次のステップを実行します。別のトポロジーを使用している場合は、`EmployeeManagement` アプリケーションがデプロイされるすべてのアプリケーション・サーバーに対し次のステップを実行してください。

1. 「サーバー」 > 「WebSphere Application Server」 > 「*server\_name*」 > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」を選択します。
2. 次の汎用 JVM プロパティを作成して、クライアント・プロパティ・ファイルの場所を設定します。  
-Dobjectgrid.client.props=*samples\_home*/security\_extauth/client3.props
3. 「OK」をクリックして、変更を保存します。

#### レッスンのチェックポイント:

クライアント・プロパティ・ファイルを編集し、クライアント・プロパティ・ファイルを使用するよう `appCluster` クラスター内のサーバーを構成しました。このプロパティ・ファイルで、使用する `CredentialGenerator` 実装クラスを指示します。

### レッスン 2.2: カタログ・サーバー・セキュリティの構成

カタログ・サーバーには、2 つの異なるレベルのセキュリティ情報があります。第 1 レベルの情報には、カタログ・サービスとコンテナ・サーバーを含むすべての WebSphere eXtreme Scale サーバーに共通するセキュリティ・プロパティが含まれます。第 2 レベルの情報には、カタログ・サーバーに固有のセキュリティ・プロパティが含まれます。

カタログ・サーバーとコンテナ・サーバーに共通するセキュリティ・プロパティは、セキュリティ XML 記述子ファイル内に構成します。共通プロパティの例の 1 つは、ユーザー・レジストリーと認証メカニズムを表すオーセンティケータ構成です。セキュリティ・プロパティの詳細については、セキュリティ記述子 XML ファイルを参照してください。

Java SE 環境でセキュリティ XML 記述子ファイルを構成するには、`startOgServer` コマンドの実行時に `-clusterSecurityFile` オプションを使用します。値はファイル・フォーマット (*samples\_home*/security\_extauth/security3.xml など) で指定します。

#### security3.xml ファイル:

このチュートリアルで、`security3.xml` ファイルは、*samples\_home*/security\_extauth ディレクトリーにあります。コメントを削除した `security3.xml` ファイルの内容を次に示します。

```
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config/security">
  <security securityEnabled="true">
    <authenticator
      className="com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginAuthenticator">
    </authenticator>
  </security>
</securityConfig>
```

`security3.xml` ファイル内に定義されているプロパティは次のとおりです。

### **securityEnabled**

`securityEnabled` プロパティは `true` に設定され、WebSphere eXtreme Scale グローバル・セキュリティーが使用可能なことをカタログ・サーバーに指示します。

### **authenticator**

オーセンティケーターは、

`com.ibm.websphere.objectgrid.security.plugins.builtins.`

`KeyStoreLoginAuthenticator` クラスとして構成されます。この Authenticator プラグインの組み込み実装により、ユーザー ID とパスワードが渡され、それらが鍵ストア・ファイル内に構成されているか検査されます。

`KeyStoreLoginAuthenticator` クラスは `KeyStoreLogin` ログイン・モジュール別名を使用するため、Java 認証・承認サービス (JAAS) ログイン構成が必要です。

### **catServer3.props** ファイル:

サーバー・プロパティ・ファイルは、サーバー固有のプロパティを保管し、これにはサーバー固有のセキュリティー・プロパティも含まれます。詳しくは、サーバー・プロパティ・ファイルを参照してください。**startOgServer** コマンドの実行時に **-serverProps** オプションを使用して、カタログ・サーバー・プロパティを指定できます。このチュートリアルの場合、`catServer3.props` ファイルは C ディレクトリにあります。コメントを削除した `catServer3.props` ファイルの内容を次に示します。

```
securityEnabled=true
credentialAuthentication=Required
transportType=TCP/IP
secureTokenManagerType=none
authenticationSecret=ObjectGridDefaultSecret
```

### **securityEnabled**

`securityEnabled` プロパティは `true` に設定され、このカタログ・サーバーがセキュア・サーバーであることを示します。

### **credentialAuthentication**

`credentialAuthentication` プロパティは `Required` に設定され、サーバーに接続するすべてのクライアントが資格情報の提供を要求されます。クライアント・プロパティ・ファイル内では `credentialAuthentication` の値が `Supported` に設定されるため、サーバーはクライアントによって送信された資格情報を受け取ります。

### **secureTokenManagerType**

`secureTokenManagerType` は `none` に設定され、既存のサーバーに結合するとき認証の機密事項が暗号化されないことを示します。

### **authenticationSecret**

`authenticationSecret` プロパティは `ObjectGridDefaultSecret` に設定されます。eXtreme Scale サーバー・クラスターに結合するとき、この秘密ストリングが使用されます。サーバーがデータ・グリッドに結合する場合、秘密ストリングの表示を求められます。結合サーバーの秘密ストリングがカタログ・サーバーのいずれかの秘密ストリングと一致する場合は、結合サーバーは受け入れられます。ストリングが一致しない場合、結合要求は拒否されません。

## transportType

transportType プロパティは、当初 TCP/IP に設定します。後ほどチュートリアルの中で、トランスポート・セキュリティーを使用可能にします。

## xsjaas3.config ファイル:

KeyStoreLoginAuthenticator 実装はログイン・モジュールを使用するため、JAAS 認証ログイン構成ファイルを使用してログイン・モデルを構成する必要があります。

xsjaas3.config ファイルの内容を次に示します。

```
KeyStoreLogin{
com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule required
  keyStoreFile="samples_home/security_extauth/sampleKS3.jks" debug = true;
};
```

*samples\_home* に */wxs\_samples/* 以外の場所を使用した場合は、keyStoreFile の場所を更新する必要があります。このログイン構成は、ログイン・モジュールとして com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule モジュールを使用することを指示します。鍵ストア・ファイルは sampleKS3.jks ファイルに設定されます。

sampleKS3.jks サンプル鍵ストア・ファイルは、2 組のユーザー ID とパスワード (manager/manager1 と cashier/cashier1) を保管します。

次の **keytool** コマンドを使用して、この鍵ストアを作成できます。

- keytool -genkey -v -keystore ./sampleKS3.jks -storepass sampleKS1 -alias manager -keypass manager1 -dname CN=manager,O=acme,OU=OGSample -validity 10000
- keytool -genkey -v -keystore ./sampleKS3.jks -storepass sampleKS1 -alias operator -keypass operator1 -dname CN=operator,O=acme,OU=OGSample -validity 10000

## セキュリティーが使用可能なカタログ・サーバーを開始する:

カタログ・サーバーを開始するには、セキュリティー・プロパティを渡す **-clusterFile** および **-serverProps** パラメーターを指定して **startOgServer** コマンドを実行します。

カタログ・サーバーを開始するときはスタンドアロン・インストールの WebSphere eXtreme Scale を使用してください。スタンドアロン・インストール・イメージを使用するときは、IBM SDK を使用しなければなりません。IBM SDK を指すように *JAVA\_HOME* 変数を設定すると、WebSphere Application Server 内に組み込まれている SDK を使用できます。例: set *JAVA\_HOME*=*was\_root*/IBM/WebSphere/AppServer/java/

1. bin ディレクトリーに移動します。  
cd *wxs\_home*/bin
2. **startOgServer** コマンドを実行します。

Linux UNIX

```
./startOgServer.sh cs1 -listenerPort 16809 -JMXServicePort 16099 -catalogServiceEndPoints
cs1:[HOST_NAME]:16601:16602 -clusterSecurityFile samples_home/security_extauth/security3.xml
-serverProps samples_home/security_extauth/catServer3.props -jvmArgs
-Djava.security.auth.login.config="samples_home/security_extauth/xsjaas3.config"
```

#### Windows

```
startOgServer.bat cs1 -listenerPort 16809 -JMXServicePort 16099 -catalogServiceEndPoints
cs1:[HOST_NAME]:16601:16602 -clusterSecurityFile samples_home/security_extauth/security3.xml
-serverProps samples_home/security_extauth/catServer3.props -jvmArgs
-Djava.security.auth.login.config="samples_home/security_extauth/xsjaas3.config"
```

**startOgServer** コマンドを実行すると、リスナー・ポート 16809、クライアント・ポート 16601、ピア・ポート 16602、および JMX ポート 16099 を使用するセキュア・サーバーが開始します。ポートが競合する場合は、未使用のポート番号にポート番号を変更してください。

**セキュリティーが使用可能なカタログ・サーバーを停止する:**

**stopOgServer** コマンドを使用して、カタログ・サーバーを停止できます。

1. bin ディレクトリーに移動します。

```
cd wxs_home/bin
```

2. **stopOgServer** コマンドを実行します。

Linux

UNIX

```
stopOgServer.sh cs1 -catalogServiceEndPoints localhost:16809 -clientSecurityFile
samples_home/security_extauth/client3.props
```

#### Windows

```
stopOgServer.bat cs1 -catalogServiceEndPoints localhost:16809 -clientSecurityFile
samples_home/security_extauth/client3.props
```

**レッスンのチェックポイント:**

security3.xml、catServer3.props、xsjaas3.config ファイルをカタログ・サービスに関連付けることで、カタログ・サーバー・セキュリティーを構成しました。

## レッスン 2.3: コンテナ・サーバー・セキュリティーの構成

コンテナ・サーバーがカタログ・サービスに接続すると、コンテナ・サーバーは、ObjectGrid セキュリティー XML ファイル内に構成されているセキュリティー構成をすべて取得します。ObjectGrid セキュリティー XML ファイルは、オーセンティケーター構成、ログイン・セッション・タイムアウト値、およびその他の構成情報を定義します。コンテナ・サーバーは、サーバー・プロパティー・ファイル内にそのサーバー固有のセキュリティー・プロパティーも保持します。

-Dobjectgrid.server.props Java 仮想マシン (JVM) プロパティーを使用して、サーバー・プロパティー・ファイルを構成します。このプロパティーに指定するファイル名は、samples\_home/security\_extauth/server3.props などの絶対ファイル・パスです。

このチュートリアルでは、コンテナ・サーバーは xsCluster クラスター内の xs1 および xs2 サーバーでホスティングされます。

**server3.props ファイル:**

server3.props ファイルは、samples\_home/security\_extauth/ ディレクトリーにあります。server3.props ファイルの内容を次に示します。



```
securityEnabled=true
credentialAuthentication=Required
secureTokenManagerType=none
authenticationSecret=ObjectGridDefaultSecret
```

### securityEnabled

securityEnabled プロパティは true に設定され、このコンテナ・サーバーがセキュア・サーバーであることを示します。

### credentialAuthentication

credentialAuthentication プロパティは Required に設定され、サーバーに接続するすべてのクライアントが資格情報の提供を要求されます。クライアント・プロパティ・ファイル内では credentialAuthentication プロパティが Supported に設定されるため、サーバーはクライアントによって送信された資格情報を受け取ります。

### secureTokenManagerType

secureTokenManagerType は none に設定され、既存のサーバーに結合するとき認証の機密事項が暗号化されないことを示します。

### authenticationSecret

authenticationSecret プロパティは ObjectGridDefaultSecret に設定されます。eXtreme Scale サーバー・クラスターに結合するとき、この秘密ストリングが使用されます。サーバーがデータ・グリッドに結合する場合、秘密ストリングの表示を求められます。結合サーバーの秘密ストリングがカタログ・サーバーのいずれかの秘密ストリングと一致する場合は、結合サーバーは受け入れられます。ストリングが一致しない場合、結合要求は拒否されず。

## JVM プロパティによるサーバー・プロパティ・ファイルの設定:

xs1 サーバーと xs2 サーバーにサーバー・プロパティ・ファイルを設定します。使用するトポロジーがこのチュートリアルと異なる場合は、コンテナ・サーバーのホスティングに使用するすべてのアプリケーション・サーバーにサーバー・プロパティ・ファイルを設定してください。

1. サーバーの Java 仮想マシン・ページを開きます。「サーバー」 > 「WebSphere Application Server」 > 「server\_name」 > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」を選択します。
2. 汎用 JVM 引数を追加します。  
-Dobjectgrid.server.props=samples\_home/security\_extauth/server3.props
3. 「OK」をクリックして、変更を保存します。

## カスタム・ログイン・モジュールの追加:

コンテナ・サーバーは、カタログ・サーバーと同じ KeyStoreAuthenticator 実装を使用します。KeyStoreAuthenticator 実装は、KeyStoreLogin ログイン・モジュール別名を使用します。このため、カスタム・ログイン・モジュールをアプリケーション・ログイン・モデルのエントリに追加する必要があります。

1. WebSphere Application Server 管理コンソールで、「セキュリティ」 > 「グローバル・セキュリティ」 > 「Java 認証・承認サービス」をクリックします。
2. 「アプリケーション・ログイン」をクリックします。



3. 「新規」をクリックし、別名 `KeyStoreLogin` を追加します。「適用」をクリックします。
4. 「JAAS ログイン・モジュール」の下で「新規」をクリックします。
5. モジュール・クラス名に  
`com.ibm.websphere.objectgrid.security.plugins.builtins.  
KeyStoreLoginModule` を入力し、認証ストラテジーとして **SUFFICIENT** を選択  
します。「適用」をクリックします。
6. `keyStoreFile` カスタム・プロパティーを追加し、値 `samples_home/  
security_extauth/sampleKS.jks` を指定します。
7. オプション: `debug` カスタム・プロパティーを追加し、値 `true` を指定します。
8. 構成を保存します。

#### レッスンのチェックポイント:

これで、WebSphere eXtreme Scale サーバー認証は保護されます。このセキュリティ  
を構成することで、WebSphere eXtreme Scale サーバーに接続しようとするすべて  
のアプリケーションが資格情報の提供を要求されます。このチュートリアルでは、  
`KeyStoreLoginAuthenticator` がオーセンティケーターです。結果として、クライアント  
はユーザー名とパスワードの提供を要求されます。

## レッスン 2.4: サンプルのインストールと実行

認証の構成が終了したら、サンプル・アプリケーションをインストールして実行で  
きます。

#### EmployeeData.jar ファイルの共有ライブラリーの作成:

1. WebSphere Application Server 管理コンソールで、「共有ライブラリー」ページ  
を開きます。「環境」 > 「共有ライブラリー」をクリックします。
2. 「セル」スコープを選択します。
3. 共有ライブラリーを作成します。「新規」をクリックします。「名前」に  
「EmployeeManagementLIB」を入力します。クラスパスに、`EmployeeData.jar` へ  
のパスを入力します。例えば、`samples_home/WASSecurity/EmployeeData.jar` で  
す。
4. 「適用」をクリックします。

#### サンプルのインストール:

1. `samples_home/security_extauth` ディレクトリーの下にある  
`EmployeeManagement_extauth.ear` ファイルをインストールします。

**重要:** `EmployeeManagement_extauth.ear` ファイルは、`samples_home/  
WASSecurity/EmployeeManagement.ear` ファイルとは異なります。ObjectGrid セ  
ッションを取得する方法が更新され、`EmployeeManagement_extauth.ear` アプリ  
ケーションでクライアント・プロパティー・ファイルのキャッシュに入れられた  
資格情報を使用するようになりました。この変更に応じて更新されたコードを確  
認するには、`samples_home/WASSecurity/EmployeeManagementWeb` プロジェクト  
の `com.ibm.websphere.sample.xs.DataAccessor` クラスのコメントを参照してくださ  
い。

- a. 「アプリケーション」 > 「新規アプリケーション」 > 「新規エンタープライズ・アプリケーション」をクリックして、インストールを開始します。アプリケーションをインストールする詳細なパスを選択します。
- b. 「モジュールをサーバーにマップ」ステップで、appCluster クラスタを指定して、EmployeeManagementWeb モジュールをインストールします。
- c. 「共有ライブラリーをマップ」ステップで、「EmployeeManagementWeb」モジュールを選択します。
- d. 「**Reference shared libraries**」をクリックします。  
「EmployeeManagementLIB」ライブラリーを選択します。
- e. webUser ロールを、「アプリケーションのレルム内のすべての認証済み」にマップします。
- f. 「**OK**」をクリックします。

クライアントは、このクラスタ内の s1 サーバーと s2 サーバーで実行されません。

2. *samples\_home*/WASSecurity ディレクトリーにあるサンプル XSDeployment.ear ファイルをインストールします。

- a. 「アプリケーション」 > 「新規アプリケーション」 > 「新規エンタープライズ・アプリケーション」をクリックして、インストールを開始します。アプリケーションをインストールする詳細なパスを選択します。
- b. 「モジュールをサーバーにマップ」ステップで、xsCluster クラスタを指定して、XSDeploymentWeb Web モジュールをインストールします。
- c. 「共有ライブラリーをマップ」ステップで、「XSDeploymentWeb」モジュールを選択します。
- d. 「**Reference shared libraries**」をクリックします。  
「EmployeeManagementLIB」ライブラリーを選択します。
- e. 「**OK**」をクリックします。

このクラスタ内の xs1 サーバーと xs2 サーバーがコンテナ・サーバーをホスティングします。

3. カタログ・サーバーが開始していることを確認します。このチュートリアル用のカタログ・サーバーを開始する方法については、128 ページの『セキュリティが使用可能なカタログ・サーバーを開始する』を参照してください。
4. xsCluster クラスタを再始動します。xsCluster が始動すると、XSDeployment アプリケーションが開始し、xs1 と xs2 サーバーのそれぞれでコンテナ・サーバーが開始します。xs1 サーバーと xs2 サーバーの SystemOut.log ファイルを調べると、サーバー・プロパティー・ファイルがロードされたことを示す次のメッセージが表示されます。

CW0BJ0913I: 次のサーバー・プロパティー・ファイルがロードされました。  
*samples\_home*/security\_extauth/server3.props.

5. appClusters クラスタを再始動します。クラスタ appCluster が始動すると、EmployeeManagement アプリケーションも開始します。s1 サーバーと s2 サーバーの SystemOut.log ファイルを調べると、クライアント・プロパティー・ファイルがロードされたことを示す次のメッセージが表示されます。

CW0BJ0924I: クライアント・プロパティー・ファイル {0} がロードされました。

WebSphere eXtreme Scale バージョン 7.0 を使用している場合は、クライアント・プロパティ・ファイルがロードされたことを示す CWOBJ9000I メッセージ (英語のみ) が表示されます。 予期されるメッセージが表示されない場合は、JVM 引数に `-Dobjectgrid.server.props` または `-Dobjectgrid.client.props` プロパティを適切に構成したか確認してください。 プロパティを確実に構成済みの場合、ダッシュ (-) が UTF 文字であるか確認してください。

#### サンプル・アプリケーションの実行:

1. `management.jsp` ファイルを実行します。 Web ブラウザーで、`http://<your_servername>:<port>/EmployeeManagementWeb/management.jsp` にアクセスします。 例えば、次のような URL を使用できます。  
`http://localhost:9080/EmployeeManagementWeb/management.jsp`
2. アプリケーションに認証を提供します。 `webUser` ロールにマップしたユーザーの資格情報を入力します。 デフォルトでは、このユーザー・ロールはすべての認証済みユーザーにマップされます。 有効なユーザー名とパスワード (管理ユーザー名とパスワードなど) を入力します。 従業員を表示、追加、更新、および削除するページが表示されます。
3. 従業員を表示します。 「**Display an Employee**」をクリックします。 E メール・アドレスに「`emp1@acme.com`」を入力し、「**Submit**」をクリックします。 従業員が見つからないというメッセージが表示されます。
4. 従業員を追加します。 「**Add an Employee**」をクリックします。 E メール・アドレスとして `emp1@acme.com` を入力し、名として `Joe` を入力し、姓として `Doe` を入力します。「**Submit**」をクリックします。 `emp1@acme.com` アドレスを持つ従業員が追加されたというメッセージが表示されます。
5. 新しい従業員を表示します。 「**Display an Employee**」をクリックします。 E メール・アドレスとして `emp1@acme.com` を入力し、姓と名のフィールドは空のままにして「**Submit**」をクリックします。 従業員が見つかったというメッセージが表示され、名フィールドと姓フィールドに正しい名前が表示されます。
6. 従業員を削除します。 「**Delete an employee**」をクリックします。「`emp1@acme.com`」を入力し、「**Submit**」をクリックします。 従業員が削除されたというメッセージが表示されます。

カタログ・サーバーのトランスポート・タイプは TCP/IP に設定されているため、サーバー `s1` および `s2` のアウトバウンド・トランスポート設定が `SSL-Required` に設定されていないことを確認してください。 そうでないと、例外が発生します。 カタログ・サーバーのシステム出力ファイルである `logs/cs1/SystemOut.log` ファイルを調べると、鍵ストア認証を示す次のデバッグ出力があります。

```
SystemOut    0 [KeyStoreLoginModule] initialize: Successfully loaded key store
SystemOut    0 [KeyStoreLoginModule] login: entry
SystemOut    0 [KeyStoreLoginModule] login: user entered user name: manager
SystemOut    0   Print out the certificates:
...
```

#### レッスンのチェックポイント:

サンプル・アプリケーションをインストールして実行しました。

## モジュール 3: トランスポート・セキュリティの構成

構成の中のクライアントとサーバー間のデータ転送をセキュアにするために、トランスポート・セキュリティを構成します。

前のチュートリアル・モジュールにおいて、WebSphere eXtreme Scale 認証を使用可能に設定しました。認証を使用可能に設定すると、WebSphere eXtreme Scale サーバーに接続を試みるすべてのアプリケーションは、資格情報の提供を要求されます。したがって、非認証クライアントは WebSphere eXtreme Scale サーバーに接続できません。クライアントは、WebSphere Application Server セルの中で実行される認証アプリケーションでなければなりません。

このモジュールまでの構成では、appCluster クラスター内のクライアントと xsCluster クラスター内のサーバー間のデータ転送は、暗号化されません。ご使用の WebSphere Application Server クラスターがファイアウォールで囲まれたサーバーにインストールされている場合は、この構成を受け入れることができます。しかし、シナリオによっては、たとえトポロジーがファイアウォールで保護されるとしても、何らかの理由で、暗号化されていないトラフィックは、受け入れられない場合もあります。例えば、政府の方針で、暗号化トラフィックが強制される場合もあります。WebSphere eXtreme Scale は、ObjectGrid エンドポイント (クライアント・サーバー、コンテナ・サーバー、およびカタログ・サーバーが含まれる) 間のセキュア通信のために Transport Layer Security/Secure Sockets Layer (TLS/SSL) をサポートします。

このサンプル・デプロイメントでは、eXtreme Scale クライアント・サーバーとコンテナ・サーバーは、すべて WebSphere Application Server 環境で実行されます。eXtreme Scale トランスポート・セキュリティは Application Server Common Secure Interoperability Protocol Version 2 (CSIv2) トランスポート設定によって管理されるため、クライアント・プロパティとサーバー・プロパティは、SSL 設定の構成には必要ありません。WebSphere eXtreme Scale サーバーは、このサーバーが実行されるアプリケーション・サーバーと同じオブジェクト・リクエスト・ブローカー (ORB) インスタンスを使用します。この CSIv2 トランスポート設定を使用して、WebSphere Application Server 構成内のクライアント・サーバーとコンテナ・サーバーに対してすべての SSL 設定を指定してください。カタログ・サーバーの場合は、そのサーバー・プロパティ・ファイルの中で SSL プロパティを構成する必要があります。

### 学習目標

このモジュールのレッスンを完了すると、以下の作業の実行方法を理解できます。

- CSIv2 のインバウンド・トランスポートとアウトバウンド・トランスポートの構成。
- SSL プロパティのカタログ・サーバー・プロパティ・ファイルへの追加。
- ORB プロパティ・ファイルの確認。
- サンプルを実行します。

### 所要時間

このモジュールの所要時間は約 60 分です。

## 前提条件

このチュートリアル・ステップは、これまでのモジュールの上に組み立てられています。トランスポート・セキュリティーを構成する前に、このチュートリアルのこれまでのモジュールを完了してください。

### レッスン 3.1: CSiv2 のインバウンド・トランスポートとアウトバウンド・トランスポートの構成

サーバー・トランスポートの Transport Layer Security/Secure Sockets Layer (TLS/SSL) を構成するには、クライアント、カタログ・サーバー、およびコンテナ・サーバーをホストするすべての WebSphere Application Server サーバーに対して、Common Secure Interoperability Protocol Version 2 (CSiv2) インバウンド・トランスポートと CSiv2 アウトバウンド・トランスポートを SSL-Required に設定します。

チュートリアルで使用するサンプルのトポロジーでは、s1、s2、xs1、および xs2 アプリケーション・サーバーに対して、これらのプロパティーを設定する必要があります。次のステップでは、構成内のすべてのサーバーのインバウンド・トランスポートとアウトバウンド・トランスポートを構成します。

管理コンソールで、インバウンド・トランスポートとアウトバウンド・トランスポートを設定します。管理セキュリティーが使用可能であることを確認します。

- **WebSphere Application Server バージョン 6.1** の場合: 「セキュリティー」 > 「セキュア管理」 > 「アプリケーション」 > 「RMI/IIOP セキュリティー」をクリックし、トランスポート・タイプを「**SSL 必須**」に変更します。
- **WebSphere Application Server バージョン 7.0** の場合: 「セキュリティー」 > 「グローバル・セキュリティー」 > 「RMI/IIOP セキュリティー」 > 「CSiv2 インバウンド通信」をクリックします。CSiv2 Transport Layer の下のトランスポート・タイプを「**SSL 必須**」に変更します。このステップを繰り返して、CSiv2 アウトバウンド通信を構成します。

中央で管理されるエンドポイント・セキュリティー設定を使用したり、SSL リポジトリを構成したりできます。詳しくは、Common Secure Interoperability バージョン 2 トランスポート・インバウンド設定を参照してください。

### レッスン 3.2: SSL プロパティーのカタログ・サーバー・プロパティー・ファイルへの追加

カタログ・サーバーは WebSphere Application Server の外側で実行されるため、サーバー・プロパティー・ファイルを使用して SSL プロパティーを構成する必要があります。

サーバー・プロパティー・ファイルで SSL プロパティーを構成するには他にも理由があります。つまり、カタログ・サーバーには、WebSphere Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) トランスポート設定によって管理できない専用のトランスポート・パスがあるということです。したがって、カタログ・サーバーのサーバー・プロパティー・ファイルで Secure Sockets Layer (SSL) プロパティーを構成する必要があります。

**catServer3.props** ファイル内の SSL プロパティー:



```

alias=default
contextProvider=IBMSSE2
protocol=SSL
keyStoreType=PKCS12
keyStore=/was_root/IBM/WebSphere/AppServer/profiles/
<deployment_manager_name>/config/cells/<cell_name>/nodes/
<node_name>/key.p12
keyStorePassword=WebAS
trustStoreType=PKCS12
trustStore=/was_root/IBM/WebSphere/AppServer/profiles/
<deployment_manager_name>/config/cells/<cell_name>/nodes/
<node_name>/trust.p12
trustStorePassword=WebAS
clientAuthentication=false

```

catServer3.props ファイルは、デフォルトの WebSphere Application Server ノード・レベルの鍵ストアとトラストストアを使用しています。より複雑なデプロイメント環境をデプロイする場合は、それにふさわしい鍵ストアとトラストストアを選択する必要があります。場合によっては、鍵ストアとトラストストアを作成し、他のサーバーの鍵ストアから鍵をインポートする必要があります。 WebSphere Application Server 鍵ストアおよびトラストストアのデフォルト・パスワードは WebAS スtring ですので、忘れないでください。詳しくは、デフォルト自己署名証明書の構成を参照してください。

これらのエントリは、既にコメントとして `samples_home/security_extauth/catServer3.props` ファイルに含まれています。これらのエントリーのコメントを外し、ご使用のインストール済み環境に合わせて `was_root`、`<deployment_manager_name>`、`<cell_name>`、および `<node_name>` の各変数を更新することができます。

SSL プロパティの構成が終わったならば、`transportType` プロパティ値を TCP/IP から SSL-Required に変更してください。

### client3.props ファイル内の SSL プロパティ:

client3.props ファイルの中でも SSL プロパティを構成する必要があります。このファイルは、WebSphere Application Server の外側で実行中のカタログ・サーバーを停止するときに使用されます。

これらのプロパティは、WebSphere Application Server で実行中のクライアント・サーバーには影響しません。というのも、これらのクライアント・サーバーは WebSphere Application Server Common Security Interoperability Protocol Version 2 (CSIV2) トランスポート設定を使用するからです。ただし、カタログ・サーバーを停止する場合は、`stopOgServer` コマンドでクライアント・プロパティ・ファイルを指定する必要があります。<SAMPLES\_HOME>/security\_extauth/client3.props ファイル中の以下のプロパティを、上記の catServer3.props ファイルで指定した値と一致するように設定してください。

```

#contextProvider=IBMSSE2
#protocol=SSL
#keyStoreType=PKCS12
#keyStore=/was_root/IBM/WebSphere/AppServer/profiles/
<deployment_manager_name>/config/cells/<cell_name>/nodes/
<node_name>/key.p12
#keyStorePassword=WebAS
#trustStoreType=PKCS12
#trustStore=/was_root/IBM/WebSphere/AppServer/profiles/
<deployment_manager_name>/config/cells/<cell_name>/nodes/
<node_name>/trust.p12
#trustStorePassword=WebAS

```



catServer3.props ファイルと同じように、*samples\_home/security\_extauth/client3.props* ファイルの中に既に提供されているコメントを利用して、*was\_root*、*<deployment\_manager\_name>*、*<cell\_name>*、および *<node\_name>* の各変数を、ご使用の環境に合うように更新することができます。

#### レッスンのチェックポイント:

カタログ・サーバーの SSL プロパティを構成しました。

### レッスン 3.3: サンプルの実行

すべてのサーバーを再始動し、再度、サンプル・アプリケーションを実行します。問題なくステップを実行できるはずです。

サンプル・アプリケーションの実行およびインストールについて詳しくは、131 ページの『レッスン 2.4: サンプルのインストールと実行』を参照してください。

## モジュール 4: WebSphere Application Server の Java 認証・承認サービス (JAAS) 許可の使用

クライアントの認証の構成が完了したので、さらに細かい許可を構成して、ユーザーごとに異なるアクセス権を付与できます。例えば、「operator」ユーザーはデータの表示のみが可能で、一方「manager」ユーザーはすべての操作を実行できます。

このチュートリアルの前のもジュールと同様に、クライアントを認証した後、eXtreme Scale 許可メカニズムによりセキュリティー特権を付与することができます。このチュートリアルの前のもジュールでは、WebSphere Application Server との統合を使用して、データ・グリッドの認証を使用可能にする方法について説明しました。結果として、非認証クライアントは、eXtreme Scale サーバーに接続したり、システムに要求を実行依頼したりすることができません。ただし、認証されている各クライアントは、ObjectGrid マップに格納されているデータの読み取り、書き込み、削除など、サーバーに対して同じアクセス権または特権を持っています。クライアントは、どのような照会でも実行できます。

チュートリアルはこの部分では、eXtreme Scale 許可を使用して認証ユーザーにさまざまな特権を付与する方法について説明します。WebSphere eXtreme Scale はアクセス権ベースの許可メカニズムを使用します。さまざまな許可クラスによって表されるさまざまな許可カテゴリーを割り当てることができます。このモジュールでは、MapPermission クラスを取り上げます。使用できるすべての許可のリストについては、クライアント許可プログラミングを参照してください。

WebSphere eXtreme Scale では、

`com.ibm.websphere.objectgrid.security.MapPermission` クラスは eXtreme Scale リソース、特に ObjectMap インターフェースまたは JavaMap インターフェースのメソッドに対する許可を表しています。WebSphere eXtreme Scale は、ObjectMap および JavaMap のメソッドにアクセスするための以下の許可ストリングを定義します。

- **read:** マップからデータを読み取る許可を与えます。
- **write:** マップのデータを更新する許可を与えます。
- **insert:** マップにデータを挿入する許可を与えます。

- **remove:** マップからデータを削除する許可を与えます。
- **invalidate:** マップからのデータを無効にする許可を与えます。
- **all:** read、write、insert、remove、および invalidate に対するすべての許可を与えます。

許可は、eXtreme Scale クライアントがデータ・アクセス API (ObjectMap API、JavaMap API、EntityManager API など) を使用したときに発生します。メソッドが呼び出されるたびに、ランタイムは、対応するマップ許可を検査します。必要な許可がクライアントに与えられていない場合は、AccessControlException 例外になります。このチュートリアルでは、Java 認証・承認サービス (JAAS) 許可を使用して、さまざまなユーザーに対する許可マップ・アクセスを付与する方法について説明します。

## 学習目標

このモジュールのレッスンを完了すると、以下の作業の実行方法を理解できます。

- WebSphere eXtreme Scale の許可を使用可能にする。
- ユーザー・ベースの許可を使用可能にする。

## 所要時間

このモジュールの所要時間は約 60 分です。

## レッスン 4.1: WebSphere eXtreme Scale 許可を使用可能にする

WebSphere eXtreme Scale の許可を使用可能にするには、特定の ObjectGrid のセキュリティを使用可能にする必要があります。

ObjectGrid で許可を使用可能にするには、XML ファイルで、その特定の ObjectGrid の **securityEnabled** 属性を true に設定する必要があります。このチュートリアルの場合、*samples\_home*/WASSecurity ディレクトリーにある XSDeployment\_sec.ear ファイルを使用するか (このファイルは、objectGrid.xml ファイル内で既にセキュリティが設定されています)、既存の objectGrid.xml ファイルを編集して、セキュリティを使用可能にできます。このレッスンでは、ファイルを編集してセキュリティを使用可能にする方法を例示します。

1. オプション: XSDeployment.ear ファイル内のファイルを抽出してから、XSDeploymentWeb.war ファイルを unzip します。
2. オプション: objectGrid.xml ファイルを開いて、ObjectGrid レベルで **securityEnabled** 属性を true に設定します。次のサンプルでこの属性の例を参照してください。

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Grid" txTimeout="15" securityEnabled="true">
      <backingMap name="Map1" />
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

複数の ObjectGrids を定義している場合は、この属性を各グリッドに設定する必要があります。

3. オプション: XSDeploymentWeb.war ファイルと XSDeployment.ear ファイルをパッケージ化し直して、変更を組み込みます。
4. 必須: XSDeployment.ear ファイルをアンインストールしてから、更新済み XSDeployment.ear をインストールします。前のステップで変更したファイルを使用しても、*samples\_home/WASSecurity* ディレクトリーに用意されている XSDeployment\_sec.ear ファイルをインストールしてもかまいません。アプリケーションのインストールの詳細については、131 ページの『レッスン 2.4: サンプルのインストールと実行』を参照してください。
5. すべてのアプリケーション・サーバーを再始動して、WebSphere eXtreme Scale 許可を使用可能にします。

#### レッスンのチェックポイント:

ObjectGrid のセキュリティーを使用可能にすることで、データ・グリッドの許可も使用可能にしました。

### レッスン 4.2: ユーザー・ベースの許可を使用可能にする

このチュートリアル of 認証モジュールの中で、operator と manager の 2 つのユーザーを作成しました。Java 認証・承認サービス (JAAS) 許可を使用して、これらのユーザーに異なる許可を割り当てることができます。

#### ユーザー・プリンシパルを使用した Java 認証・承認サービス (JAAS) 許可ポリシーの定義:

前に作成したユーザーに許可を割り当てることができます。operator ユーザーには、すべてのマップに対する読み取り許可のみを割り当てます。manager ユーザーには、すべての許可を割り当てます。JAAS 許可ポリシー・ファイルを使用して、プリンシパルに許可を付与します。

JAAS 許可ファイルを編集します。xsAuth3.policy ファイルは、*samples\_home/security\_extauth* ディレクトリーにあります。

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
principal javax.security.auth.x500.X500Principal
  "CN=operator,O=acme,OU=OGSample" {
  permission com.ibm.websphere.objectgrid.security.MapPermission "Grid.Map1", "read";
};

grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
principal javax.security.auth.x500.X500Principal
  "CN=manager,O=acme,OU=OGSample" {
  permission com.ibm.websphere.objectgrid.security.MapPermission "Grid.Map1", "all";
};
```

このファイルにある <http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction> コードベースは、ObjectGrid 用に特別に予約された URL です。プリンシパルに付与されているすべての ObjectGrid 許可では、この特別なコードベースを使用します。このファイルでは次の許可が割り当てられます。

- 最初の grant ステートメントは、read マップ許可を "CN=operator,O=acme,OU=OGSample" プリンシパルに付与します。 "CN=operator,O=acme,OU=OGSample" ユーザーは、Grid ObjectGrid インスタンス内の Map1 マップに対するマップ読み取り許可のみを保持します。

- 2 番目の grant ステートメントは、all マップ許可を  
 "CN=manager,O=acme,OU=OGSample" プリンシパルに付与します。  
 "CN=manager,O=acme,OU=OGSample" ユーザーは、Grid ObjectGrid インスタンス内  
 の Map1 マップに対するすべての許可を保持します。

### JVM プロパティを使用した JAAS 許可ポリシー・ファイルの設定:

次のステップを使用して、xsCluster クラスター内の xs1 サーバーと xs2 サーバーの JVM プロパティを設定します。このチュートリアルで使用するサンプル・トポロジとは異なるトポロジを使用する場合は、すべてのコンテナ・サーバーにファイルを設定してください。

1. 管理コンソールで、「サーバー」 > 「アプリケーション・サーバー」 > 「*server\_name*」 > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」をクリックします。
2. 次の汎用 JVM 引数を追加します。  
`-Djava.security.auth.policy=samples_home/security_extauth/xsAuth3.policy`
3. 「OK」をクリックして、変更を保存します。

### サンプル・アプリケーションを実行して許可をテストする:

サンプル・アプリケーションを使用して、許可設定をテストできます。マネージャー・ユーザーは、従業員の表示や追加を含め、Map1 マップでのすべての許可をそのまま保持しています。オペレーター・ユーザーは、読み取り許可しか割り当てられていないため、従業員の表示のみが可能です。

1. コンテナ・サーバーを実行しているすべてのアプリケーション・サーバーを再始動します。このチュートリアルの場合は、xs1 サーバーと xs2 サーバーを再始動します。
2. EmployeeManagementWeb アプリケーションを開きます。Web ブラウザーで、`http://<host>:<port>/EmployeeManagementWeb/management.jsp` を開きます。
3. 有効なユーザー名とパスワードを使用してアプリケーションにログインします。
4. 従業員を表示してみます。「Display an Employee」をクリックし、E メール・アドレス「authemp1@acme.com」を検索します。ユーザーが見つからないというメッセージが表示されます。
5. 従業員を追加します。「Add an Employee」をクリックします。E メール・アドレス authemp1@acme.com、名 Joe、および姓 Doe を追加し、「Submit」をクリックします。従業員が追加されたというメッセージが表示されます。
6. `samples_home/security_extauth/client3.props` ファイルを編集します。credentialGeneratorProps プロパティの値を manager manager1 から operator operator1 に変更します。ファイルを編集すると、サブレットが WebSphere eXtreme Scale サーバーへの認証にユーザー名「operator」とパスワード「operator1」を使用するようになります。
7. appCluster クラスターを再始動して、`samples_home/security_extauth/client3.props` ファイル内の変更を反映します。
8. 従業員を表示してみます。「Display an Employee」をクリックし、E メール・アドレス「authemp1@acme.com」を検索します。従業員が表示されます。

9. 従業員を追加します。「Add an Employee」をクリックします。E メール・アドレス authemp2@acme.com、名 Joe、および姓 Doe を追加し、「Submit」をクリックします。次のメッセージが表示されます。

An exception occurs when Add the employee. See below for detailed exception messages.

詳細な例外テキストが続きます。

```
java.security.AccessControlException: Access denied
(com.ibm.websphere.objectgrid.security.MapPermission Grid.Map1 insert)
```

operator ユーザーには、データを Map1 マップに挿入する許可がないため、このメッセージが表示されます。

バージョン 7.0.0.11 より前のバージョンの WebSphere Application Server で実行している場合、コンテナ・サーバーで java.lang.StackOverflowError エラーが表示されることがあります。このエラーの原因は IBM Developer Kit の問題です。この問題は、WebSphere Application Server バージョン 7.0.0.11 以上に同梱されている IBM Developer Kit では修正済みです。

#### レッスンのチェックポイント:

このレッスンでは、特定のユーザーに許可を割り当てて、許可を構成しました。

## モジュール 5: xscmd ユーティリティーを使用してデータ・グリッドとマップをモニターする

**xscmd** ユーティリティーを使用して、プライマリー・データ・グリッドと Grid データ・グリッドのマップ・サイズを表示できます。**xscmd** ツールは MBean を使用して、プライマリー断片、レプリカ断片、コンテナ・サーバー、マップ・サイズおよびそれ以外のデータなど、すべてのデータ・グリッド成果物を照会します。

このチュートリアルでは、カタログ・サーバーはスタンドアロン Java SE サーバーとして稼働します。コンテナ・サーバーは WebSphere Application Server アプリケーション・サーバー内で稼働します。

カタログ・サーバーの場合、スタンドアロン Java 仮想マシン (JVM) 内に MBean サーバーが作成されます。カタログ・サーバーで **xscmd** ツールを使用するときは、WebSphere eXtreme Scale セキュリティーが使用されます。

コンテナ・サーバーの場合、WebSphere eXtreme Scale ランタイムが、WebSphere Application Server ランタイムによって作成される Managed Bean (MBean) サーバーに MBean を登録します。**xscmd** ツールが使用するセキュリティは WebSphere Application Server MBean セキュリティーによって提供されます。

1. コマンド行ツールを使用して、**DMGR\_PROFILE/bin** ディレクトリーを開きます。
2. **xscmd** ツールを実行します。次の例のように **-c showPlacement -st P** パラメーターを使用します。

Linux UNIX

```
xscmd.sh -c listObjectGridPlacement -cep localhost:16099 -g Grid -ms mapSet -sf P
-user manager -pwd manager1
```

Windows

```
xscmd.bat -c listObjectGridPlacement -cep localhost:16099 -g Grid -m mapSet -sf P
-user manager -pwd manager1
```

ユーザー名とパスワードが認証のためにカタログ・サーバーに渡されます。

3. コマンドの結果を表示します。

```
*** Showing all primaries for grid - Grid & mapset - mapSet
Partition Container Host Server
0 myCell102¥myNode04¥xs2_C-1 myhost.mycompany.com myCell102¥myNode04¥xs2
1 myCell102¥myNode04¥xs2_C-1 myhost.mycompany.com myCell102¥myNode04¥xs2
2 myCell102¥myNode04¥xs2_C-1 myhost.mycompany.com myCell102¥myNode04¥xs2
3 myCell102¥myNode04¥xs2_C-1 myhost.mycompany.com myCell102¥myNode04¥xs2
4 myCell102¥myNode04¥xs2_C-1 myhost.mycompany.com myCell102¥myNode04¥xs2
```

4. **xscmd** ツールを実行します。 次の例のように **-c showMapSizes** パラメーターを使用します。

Linux UNIX

```
xscmd.sh -c showMapSizes -cep localhost:16099 -g Grid -ms mapSet -user manager -pwd manager1
```

Windows

```
xscmd.bat -c showMapSizes -cep localhost:16099 -g Grid -ms mapSet -user manager -pwd manager1
```

ユーザー名とパスワードが認証のためにカタログ・サーバーに渡されます。コマンドを実行すると、WebSphere Application Server での認証のために WebSphere Application Server ユーザー ID とパスワードを要求するプロンプトが出されます。 **-c showMapSizes** オプションは各コンテナ・サーバーからマップ・サイズを取得し、コンテナ・サーバーでは WebSphere Application Server セキュリティーが要求されるため、このログイン情報を提供する必要があります。

5. オプション: PROFILE/properties/sas.client.props ファイルを変更して、ユーザー ID とパスワードを要求されないコマンドを実行できます。

com.ibm.CORBA.loginSource プロパティを prompt から properties に変更してから、ユーザー ID とパスワードを指定します。PROFILE/properties/sas.client.props ファイル内のプロパティの例を次に示します。

```
com.ibm.CORBA.loginSource=properties
# RMI/IIOP user identity
com.ibm.CORBA.loginUserId=Admin
com.ibm.CORBA.loginPassword=xxxxxx
```

6. オプション: WebSphere eXtreme Scale スタンドアロン・インストール済み環境で **xscmd** コマンドを使用する場合は、次のオプションを追加する必要があります。

- WebSphere eXtreme Scale セキュリティーを使用している場合:

```
-user
-pwd
```

- カスタム資格情報生成を指定した WebSphere eXtreme Scale セキュリティーを使用している場合:

```
-user
-pwd
-cgc
-cgp
```

- SSL が使用可能な場合:



```
-tt
-cxpv
-prot
-ks
-ksp
-kst
-ts
-tsp
-tst
```

WebSphere eXtreme Scale セキュリティーと SSL の両方が使用可能な場合は、両方のパラメーター・セットが必要です。

## レッスンのチェックポイント

`xscmd` ツールを使用して、構成内のデータ・グリッドとマップをモニターしました。

---

## チュートリアル: OSGi フレームワークでの eXtreme Scale バンドルの実行

OSGi サンプルは、Google Protocol Buffers シリアライザー・サンプル上でビルドします。この一連のレッスンを完了すると、OSGi フレームワークでのシリアライザー・サンプル・プラグインの実行も完了します。

### 学習目標

このサンプルは OSGi バンドルのデモです。シリアライザー・プラグインは付随的なプラグインであり、必須ではありません。OSGi サンプルは、WebSphere eXtreme Scale Samples Gallery から入手できます。サンプルをダウンロードし、それを `wxs_home/samples` ディレクトリーに抽出する必要があります。OSGi サンプルのルート・ディレクトリーは `wxs_home/samples/OSGiProto` です。

このチュートリアルのサンプル・コマンドは、ユーザーが UNIX オペレーティング・システムで実行していることを前提としています。Windows オペレーティング・システムで実行する場合は、サンプル・コマンドを調整してください。

このチュートリアルのレッスンを完了すると、OSGi サンプルの概念を理解し、次の目的を達成する方法がわかります。

- eXtreme Scale サーバーを開始する OSGi コンテナに WebSphere eXtreme Scale サーバー・バンドルをインストールする。
- サンプル・クライアントを実行する eXtreme Scale 開発環境をセットアップする。
- `xscmd` コマンドを使用して、サンプル・バンドルのサービス・ランキングを照会したり、それを新しいサービス・ランキングにアップグレードしたり、新しいサービス・ランキングを検査する。

### 所要時間

このモジュールの所要時間は約 60 分です。

## 前提条件

シリアルライザー・サンプルのダウンロードと抽出に加えて、このチュートリアルには次の前提条件もあります。

- eXtreme Scale 製品のインストールと抽出
- Eclipse Equinox 環境のセットアップ

## 概要: OSGi フレームワークで eXtreme Scale サーバーとコンテナを開始および構成してプラグインを実行する

このチュートリアルでは、OSGi フレームワーク内で eXtreme Scale サーバーを開始し、eXtreme Scale コンテナを開始し、サンプル・プラグインと eXtreme Scale ランタイム環境を接続します。

## 学習目標

このチュートリアルのレッスンを完了すると、OSGi サンプルの概念を理解し、次の目的を達成する方法がわかります。

- eXtreme Scale サーバーを開始する OSGi コンテナに WebSphere eXtreme Scale サーバー・バンドルをインストールする。
- サンプル・クライアントを実行する eXtreme Scale 開発環境をセットアップする。
- xscmd コマンドを使用して、サンプル・バンドルのサービス・ランキングを照会したり、それを新しいサービス・ランキングにアップグレードしたり、新しいサービス・ランキングを検査する。

## 所要時間

このチュートリアルの所要時間は約 60 分です。このチュートリアルに関連した他の概念も調べる場合、完了までの所要時間はこれより長くなります。

## スキル・レベル

中級

## 対象者

OSGi フレームワークで eXtreme Scale バンドルをビルド、インストール、および実行する必要がある開発者と管理者

## システム要件

- Luminis OSGi Configuration Admin command line client バージョン 0.2.5
- Apache Felix File Install バージョン 3.0.2
- Blueprint コンテナ・プロバイダーとして Eclipse Gemini を使用する場合、以下が必要です。
  - Eclipse Gemini Blueprint バージョン 1.0.0
  - Spring Framework バージョン 3.0.5
  - SpringSource AOP Alliance API バージョン 1.0.0
  - SpringSource Apache Commons Logging バージョン 1.1.1

- Blueprint コンテナ・プロバイダーとして Apache Aries を使用する場合、以下の要件を満たしている必要があります。
  - Apache Aries (最新のスナップショット)
  - ASM ライブラリー
  - PAX logging

## 前提条件

このチュートリアルを実行するには、サンプルをダウンロードし、それを `wxs_home/samples` ディレクトリーに抽出する必要があります。OSGi サンプルのルート・ディレクトリーは `wxs_home/samples/OSGiProto` です。

## 予想される結果

このチュートリアルを完了すると、サンプル・バンドルのインストールが完了し、eXtreme Scale クライアントを実行してデータをグリッドに挿入できる状態になります。また、OSGi コンテナが提供する動的な機能を使用して、それらのサンプル・バンドルの照会や更新も可能になります。

# モジュール 1: eXtreme Scale サーバー・バンドルをインストールおよび構成する準備

このモジュールを実行して、OSGi サンプル・バンドルを探索し、eXtreme Scale サーバーの構成に使用する構成ファイルを調べます。

## 学習目標

このモジュールのレッスンを完了すれば、概念を理解し、以下の目標を達成する方法が分かります。

- OSGi サンプルに組み込まれているバンドルを探して、調べる。
- eXtreme Scale グリッドおよびサーバーの構成に使用する構成ファイルを調査する。

## レッスン 1.1: OSGi サンプル・バンドルの理解

このレッスンを実行して、OSGi サンプル内に用意されているバンドルを探して、調べます。

### OSGi サンプル・バンドル:

Eclipse Equinox 環境のセットアップについてのトピックで記載している `config.ini` ファイル内に構成されているバンドル以外にも、OSGi サンプルでは次のバンドルが追加で使用されます。

#### **objectgrid.jar**

WebSphere eXtreme Scale サーバー・ランタイム・バンドル。このバンドルは `wxs_home/lib` ディレクトリーにあります。

#### **com.google.protobuf\_2.4.0a.jar**

Google Protocol Buffers バージョン 2.4.0a バンドル。このバンドルは `wxs_sample_osgi_root/lib` ディレクトリーにあります。

### **ProtoBufSamplePlugins-1.0.0.jar**

サンプル ObjectGridEventListener および MapSerializerPlugin プラグイン実装を備えたバージョン 1.0.0 のユーザー・プラグイン・バンドル。このバンドルは `wxs_sample_osgi_root/lib` ディレクトリーにあります。サービスはサービス・ランキング 1 で構成されます。

このバージョンは、標準 Blueprint XML を使用して、eXtreme Scale プラグイン・サービスを構成します。サービス・クラスは WebSphere eXtreme Scale インターフェースである

`com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory` のユーザー実装クラスです。ユーザー実装クラスは、要求ごとに Bean を作成し、プロトタイプ・スコープの Bean と似た動きをします。

### **ProtoBufSamplePlugins-2.0.0.jar**

サンプル ObjectGridEventListener および MapSerializerPlugin プラグイン実装を備えたバージョン 2.0.0 のユーザー・プラグイン・バンドル。このバンドルは `wxs_sample_osgi_root/lib` ディレクトリーにあります。サービスはサービス・ランキング 2 で構成されます。

このバージョンは、標準 Blueprint XML を使用して、eXtreme Scale プラグイン・サービスを構成します。サービス・クラスは、WebSphere eXtreme Scale 組み込みクラスである

`com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactoryImpl` を使用し、この組み込みクラスは BlueprintContainer サービスを使用します。標準 Blueprint XML 構成を使用して、プロトタイプ・スコープまたは singleton スコープの Bean を構成できます。Bean は断片スコープとしては構成されません。

### **ProtoBufSamplePlugins-Gemini-3.0.0.jar**

サンプル ObjectGridEventListener および MapSerializerPlugin プラグイン実装を備えたバージョン 3.0.0 のユーザー・プラグイン・バンドル。このバンドルは `wxs_sample_osgi_root/lib` ディレクトリーにあります。サービスはサービス・ランキング 3 で構成されます。

このバージョンは、Eclipse Gemini 固有の Blueprint XML を使用して、eXtreme Scale プラグイン・サービスを構成します。サービス・クラスは、WebSphere eXtreme Scale 組み込みクラスである

`com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactoryImpl` を使用し、この組み込みクラスは BlueprintContainer サービスを使用します。断片スコープ Bean の構成には Gemini 固有のアプローチが使用されます。このバージョンは、スコープ値に `{http://www.ibm.com/schema/objectgrid}shard` を指定し、カスタム・スコープが Gemini に認識されるようダミー属性を構成することで、`myShardListener` Bean を断片スコープの Bean として構成します。こうする理由は、Eclipse の問題 ([https://bugs.eclipse.org/bugs/show\\_bug.cgi?id=348776](https://bugs.eclipse.org/bugs/show_bug.cgi?id=348776)) にあります。

### **ProtoBufSamplePlugins-Aries-4.0.0.jar**

サンプル ObjectGridEventListener および MapSerializerPlugin プラグイン実装を備えたバージョン 4.0.0 のユーザー・プラグイン・バンドル。このバンドルは `wxs_sample_osgi_root/lib` ディレクトリーにあります。サービスはサービス・ランキング 4 で構成されます。

このバージョンは、標準 Blueprint XML を使用して、eXtreme Scale プラグイン・サービスを構成します。サービス・クラスは、WebSphere eXtreme Scale 組み込みクラスである

com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactoryImpl を使用し、この組み込みクラスは BlueprintContainer サービスを使用します。標準 Blueprint XML 構成を使用して、カスタム・スコープの Bean を構成できます。このバージョンは、スコープ値に {http://www.ibm.com/schema/objectgrid}shard を指定することで、myShardListenerbean を断片スコープの Bean として構成します。

#### **ProtoBufSamplePlugins-Activator-5.0.0.jar**

サンプル ObjectGridEventListener および MapSerializerPlugin プラグイン実装を備えたバージョン 5.0.0 のユーザー・プラグイン・バンドル。このバンドルは `wxs_sample_osgi_root/lib` ディレクトリにあります。サービスはサービス・ランキング 5 で構成されます。

このバージョンは、Blueprint コンテナを一切使用しません。このバージョンでは、サービスは OSGi サービス登録を使用して登録されます。サービス・クラスは WebSphere eXtreme Scale インターフェースである com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory のユーザー実装クラスです。ユーザー実装クラスは、要求ごとに Bean を作成します。それはプロトタイプ・スコープの Bean と似た動きをします。

#### **レッスンのチェックポイント:**

OSGi サンプルで提供されるバンドルを調べることで、OSGi コンテナ内で実行する独自の実装を開発する方法がさらによくわかります。

以下について学習しました。

- OSGi サンプルに組み込まれているバンドル
- それらのバンドルの場所
- 各バンドルに構成されているサービス・ランキング

### **レッスン 1.2: OSGi 構成ファイルの理解**

OSGi サンプルには、WebSphere eXtreme Scale グリッドおよびサーバーを開始したり構成したりする際に使用する構成ファイルが含まれています。

#### **OSGi 構成ファイル:**

このレッスンでは、OSGi サンプルに含まれている以下の構成ファイルを検討することとします。

- `collocated.server.properties`
- `protoBufObjectGrid.xml`
- `protoBufDeployment.xml`
- `blueprint.xml`

#### **collocated.server.properties**

サーバーを開始するにはサーバー構成が必要です。eXtreme Scale サーバー・バンドルを開始しても、サーバーは開始されません。バンドルは、サーバー・プロパティ

ー・ファイルが指定された構成 PID `com.ibm.websphere.xs.server` が作成されるのを待ちます。このサーバー・プロパティ・ファイルが、サーバー名、ポート番号、その他のサーバー・プロパティを指定します。

ほとんどの場合は、サーバー・プロパティ・ファイルを設定するための構成を作成します。まれには、すべてのプロパティがデフォルト値に設定されたままでサーバーを開始すれば済むことがあります。そのような場合は、値が `default` に設定された `com.ibm.websphere.xs.server` という構成を作成できます。

サーバー・プロパティ・ファイルの詳細については、サーバー・プロパティ・ファイルのトピックを参照してください。

OSGi サンプル・サーバー・プロパティ・ファイルは単一のカタログを開始します。このサンプル・プロパティ・ファイルは、OSGi フレームワーク・プロセス内で単一のカタログ・サービスとコンテナ・サーバーを開始します。eXtreme Scale クライアントはポート 2809 に接続し、JMX クライアントはポート 1099 に接続します。サンプルのサーバー・プロパティ・ファイルの内容は以下のとおりです。

```
serverName=collocatedServer
isCatalog=true
catalogClusterEndpoints=collocatedServer:localhost:6601:6602
traceSpec=ObjectGridOSGi=all=enabled
traceFile=logs/trace.log
listenerPort=2809
JMXServicePort=1099
```

### protoBufObjectGrid.xml

サンプル `protoBufObjectGrid.xml` ObjectGrid 記述子 XML ファイルは次の内容を含んでいます (コメントは削除してあります)。

```
<objectGridConfig
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Grid" txTimeout="15">
      <bean id="ObjectGridEventListener"
        osgiService="myShardListener"/>
      <backingMap name="Map" readOnly="false"
        lockStrategy="PESSIMISTIC" lockTimeout="5"
        copyMode="COPY_TO_BYTES"
        pluginCollectionRef="serializer"/>
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="serializer">
      <bean id="MapSerializerPlugin"
        osgiService="myProtoBufSerializer"/>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

この ObjectGrid 記述子 XML ファイルには次の 2 つのプラグインが構成されています。

#### ObjectGridEventListener

断片レベル・プラグイン。ObjectGrid インスタンスごとに、ObjectGridEventListener のインスタンスが存在します。それは OSGi サービス



ス myShardListener を使用するよう構成されています。これは、グリッドの作成時、ObjectGridEventListener プラグインが、使用可能な最も高いサービス・ランキングが設定された OSGi サービス myShardListener を使用することを意味します。

### MapSerializerPlugin

マップ・レベル・プラグイン。Map という名前のバックアップ・マップに対し、MapSerializerPlugin プラグインが構成されています。それは OSGi サービス myProtoBufSerializer を使用するよう構成されています。これは、マップの作成時、MapSerializerPlugin プラグインが、使用可能な最も高いランクのサービス・ランキングが設定されたサービス myProtoBufSerializer を使用することを意味します。

### protoBufDeployment.xml

デプロイメント記述子 XML ファイルは、5 つの区画を使用する Grid という名前のグリッドのデプロイメント・ポリシーを記述したものです。XML ファイルの次のサンプル・コードを参照してください。

```
<deploymentPolicy
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="Grid">
    <mapSet name="MapSet" numberOfPartitions="5">
      <map ref="Map"/>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

### blueprint.xml

collocated.server.properties ファイルと構成 PID com.ibm.websphere.xs.server を組み合わせて使用する代わりに、次の例で示すように、ObjectGrid XML ファイルとデプロイメント XML ファイルを Blueprint XML ファイルと一緒に OSGi バンドルに組み込むことができます。

```
<blueprint
  xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
  default-activation="lazy">

  <objectgrid:server id="server" isCatalog="true"
    name="server"
    tracespec="ObjectGridOSGi=all=enabled"
    tracefile="C:/Temp/logs/trace.log"
    workingDirectory="C:/Temp/working"
    jmxport="1099">
    <objectgrid:catalog host="localhost" port="2809"/>
  </objectgrid:server>

  <objectgrid:container id="container"
    objectgridxml="/META-INF/objectgrid.xml"
    deploymentxml="/META-INF/deployment.xml"
    server="server"/>
</blueprint>
```

### レッスンのチェックポイント:

このレッスンでは、OSGi サンプル内で使用している構成ファイルについて学習しました。これで、eXtreme Scale グリッドおよびサーバーを開始して構成するとき、OSGi フレームワーク内で、それらのプロセスにどのファイルが使用され、それらのファイルがプラグインとどのように相互作用するかがわかります。

## モジュール 2: OSGi フレームワークでの eXtreme Scale バンドルのインストールおよび開始

このモジュールのレッスンを使用して、eXtreme Scale サーバー・バンドルを OSGi コンテナにインストールし、WebSphere eXtreme Scale サーバーを始動します。

OSGi フレームワークでサーバーを始動しても、OSGi バンドルが実行可能状態になるわけではありません。インストールした OSGi バンドルが認識されて正しく実行できるように、サーバー・プロパティおよびコンテナを構成する必要があります。

### 学習目標

このモジュールのレッスンを完了すると、概念を理解し、以下の作業を行う方法が分かります。

- Equinox OSGi コンソールを使用した eXtreme Scale バンドルのインストール。
- eXtreme Scale サーバーを構成します。
- eXtreme Scale コンテナを構成します。
- eXtreme Scale サンプル・バンドルをインストールして開始します。

### 前提条件

このモジュールを完了するには、開始の前に次のタスクを行う必要があります。

- eXtreme Scale 製品のインストールと抽出
- Eclipse Equinox 環境のセットアップ

このモジュールのレッスンを完了するには、次のファイルに対するアクセスについても準備する必要があります。

- objectgrid.jar バンドル。この eXtreme Scale バンドルをインストールします。
- collocated.server.properties ファイル。サーバー・プロパティをこの構成ファイルに追加します。

次のバンドルをインストールして開始する予定です。

- protobuf-java-2.4.0a-bundle.jar バンドル
- ProtoBufSamplePlugins-1.0.0.jar バンドル

### レッスン 2.1: コンソールの開始と eXtreme Scale サーバー・バンドルのインストール

このレッスンでは、Equinox OSGi コンソールを使用して、WebSphere eXtreme Scale サーバー・バンドルをインストールします。

1. 次のコマンドを使用して、Equinox OSGi コンソールを開始します。

```
cd equinox_root
java -jar plugins\org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

2. OSGi コンソールが開始した後、コンソールの中で `ss` コマンドを発行すると、次のバンドルが開始します。

**重要:** タスク「eXtreme Scale バンドルのインストール」を完了している場合は、バンドルが既にアクティブになっています。バンドルが開始された場合は、このステップを完了する前にバンドルを停止してください。

**Eclipse Gemini output:**

```

osgi> ss
Framework is launched.
id State Bundle
0 ACTIVE org.eclipse.osgi_3.6.1.R36x_v20100806
1 ACTIVE org.eclipse.osgi.services_3.2.100.v20100503
2 ACTIVE org.eclipse.osgi.util_3.2.100.v20100503
3 ACTIVE org.eclipse.equinox.cm_1.0.200.v20100520
4 ACTIVE com.springsource.org.apache.commons.logging_1.1.1
5 ACTIVE com.springsource.org.aopalliance_1.0.0
6 ACTIVE org.springframework.aop_3.0.5.RELEASE
7 ACTIVE org.springframework.asm_3.0.5.RELEASE
8 ACTIVE org.springframework.beans_3.0.5.RELEASE
9 ACTIVE org.springframework.context_3.0.5.RELEASE
10 ACTIVE org.springframework.core_3.0.5.RELEASE
11 ACTIVE org.springframework.expression_3.0.5.RELEASE
12 ACTIVE org.apache.felix.fileinstall_3.0.2
13 ACTIVE net.luminis.cmc_0.2.5
14 ACTIVE org.eclipse.gemini.blueprint.core_1.0.0.RELEASE
15 ACTIVE org.eclipse.gemini.blueprint.extender_1.0.0.RELEASE
16 ACTIVE org.eclipse.gemini.blueprint.io_1.0.0.RELEASE

```

**Apache Aries output:**

```

osgi> ss
Framework is launched.
id State Bundle
0 ACTIVE org.eclipse.osgi_3.6.1.R36x_v20100806
1 ACTIVE org.eclipse.osgi.services_3.2.100.v20100503
2 ACTIVE org.eclipse.osgi.util_3.2.100.v20100503
3 ACTIVE org.eclipse.equinox.cm_1.0.200.v20100520
4 ACTIVE org.ops4j.pax.logging.pax-logging-api_1.6.3
5 ACTIVE org.ops4j.pax.logging.pax-logging-service_1.6.3
6 ACTIVE org.objectweb.asm.all_3.3.0
7 ACTIVE org.apache.aries.blueprint_0.3.2.SNAPSHOT
8 ACTIVE org.apache.aries.util_0.4.0.SNAPSHOT
9 ACTIVE org.apache.aries.proxy_0.4.0.SNAPSHOT
10 ACTIVE org.apache.felix.fileinstall_3.0.2
11 ACTIVE net.luminis.cmc_0.2.5

```

- objectgrid.jar バンドルをインストールします。Java 仮想マシン (JVM) でサーバーを始動するには、eXtreme Scale サーバー・バンドルをインストールする必要があります。この eXtreme Scale サーバー・バンドルは、サーバーの始動およびコンテナの作成を行うことができます。次のコマンドを使用して、objectgrid.jar ファイルをインストールします。

```
osgi> install file:///wxs_home/lib/objectgrid.jar
```

次の例を参照してください。

```
osgi> install file:///opt/wxs/ObjectGrid/lib/objectgrid.jar
```

Equinox は、そのバンドル ID を表示します。例えば次のとおりです。

```
Bundle id is 19
```

**要確認:** 表示されるバンドル ID はこれとは異なる可能性があります。ファイル・パスは、バンドル・パスに対する絶対 URL でなければなりません。相対パスはサポートされません。

**レッスンのチェックポイント:**

このレッスンでは、Equinox OSGi コンソールを使用して objectgrid.jar バンドルをインストールしました。このチュートリアルの後半で、このバンドルを使用して、サーバーを始動し、コンテナを作成します。

## レッスン 2.2: eXtreme Scale サーバーのカスタマイズと構成

このレッスンでは、サーバー・プロパティをカスタマイズし、WebSphere eXtreme Scale サーバーに追加します。

1. `wxs_sample_osgi_root/projects/server/properties/collocated.server.properties` ファイルを編集します。
  - a. `traceFile` プロパティを `equinox_root/logs/trace.log` に変更します。
2. ファイルを保存します。
3. OSGI コンソールで次のコード行を入力して、ファイルからサーバー構成を作成します。以下の例は、印刷の都合上、複数行で表示されています。

```
osgi> cm create com.ibm.websphere.xs.server
```

```
osgi> cm put com.ibm.websphere.xs.server objectgrid.server.props wxs_sample_osgi_root/projects/server/properties/collocated.server.properties
```

4. 構成を表示するため、次のコマンドを実行します。

```
osgi> cm get com.ibm.websphere.xs.server
Configuration for service (pid) "com.ibm.websphere.xs.server"
(bundle location = null)
key                               value
----                               ----
objectgrid.server.props           wxs_sample_osgi_root/projects/server/properties/collocated.server.properties
service.pid                       com.ibm.websphere.xs.server
```

### レッスンのチェックポイント:

このレッスンでは、`wxs_sample_osgi_root/projects/server/properties/collocated.server.properties` ファイルを編集して、作業ディレクトリーやトレース・ログ・ファイルの場所などのサーバー設定を指定しました。

## レッスン 2.3: eXtreme Scale コンテナの構成

このレッスンを実行して、コンテナを構成します。この構成には、WebSphere eXtreme Scale ObjectGrid 記述子 XML ファイルと ObjectGrid デプロイメント XML ファイルが含まれます。これらのファイルには、グリッドの構成とそのトポロジーが含まれます。

コンテナを作成するには、最初に、管理サービス・ファクトリーのプロセス識別番号 (PID) である `com.ibm.websphere.xs.container` を使用して構成サービスを作成します。サービス構成は管理サービス・ファクトリーであるため、ファクトリー PID から複数のサービス PID を作成できます。次に、コンテナ・サービスを開始するため、各サービス PID に `objectgridFile` および `deploymentPolicyFile` PID を設定します。

次のステップを実行して、サーバー・プロパティをカスタマイズし、OSGi フレームワークに追加します。

1. OSGI コンソールで、次のコマンドを入力して、ファイルからコンテナを作成します。

```
osgi> cm createf com.ibm.websphere.xs.container
PID: com.ibm.websphere.xs.container-1291179621421-0
```

2. 次のコマンドを入力して、新しく作成した PID を ObjectGrid XML ファイルにバインドします。

**要確認:** 実際の PID 番号は、このサンプルに記載されるものとは異なります。

```
osgi> cm put com.ibm.websphere.xs.container-1291179621421-0 objectgridFile wxs_sample_osgi_root/projects/server/META-INF/protoBufObjectgrid.xml
osgi> cm put com.ibm.websphere.xs.container-1291179621421-0 deploymentPolicyFile wxs_sample_osgi_root/projects/server/META-INF/protoBufDeployment.xml
```

3. 次のコマンドを使用して、構成を表示します。

```
osgi> cm get com.ibm.websphere.xs.container-1291760127968-0
Configuration for service (pid) "com.ibm.websphere.xs.container-1291760127968-0"
(bundle location = null)
```

key	value
deploymentPolicyFile	/opt/wxs/ObjectGrid/samples/OSGiProto/server/META-INF/protoBufDeployment.xml
objectgridFile	/opt/wxs/ObjectGrid/samples/OSGiProto/server/META-INF/protoBufObjectgrid.xml
service.factoryPid	com.ibm.websphere.xs.container
service.pid	com.ibm.websphere.xs.container-1291760127968-0

### レッスンのチェックポイント:

このレッスンでは、eXtreme Scale コンテナを作成するために使用する構成サービスを作成しました。ObjectGrid XML ファイルには、グリッドの構成とそのトポロジーが含まれるため、作成したコンテナをそれらの ObjectGrid XML ファイルにバインドする必要があります。この構成により、eXtreme Scale コンテナが、後ほどこのチュートリアルで実行する OSGi バンドルを認識できます。

## レッスン 2.4: Google Protocol Buffers バンドルとサンプル・プラグイン・バンドルのインストール

このチュートリアルでは、Equinox OSGi コンソールを使用して、`protobuf-java-2.4.0a-bundle.jar` バンドルと `ProtoBufSamplePlugins-1.0.0.jar` プラグイン・バンドルをインストールします。

### Google Protocol Buffers プラグインのインストール:

次のステップを実行して、Google Protocol Buffers プラグインをインストールします。

OSGi コンソールで、次のコマンドを入力して、プラグインをインストールします。

```
osgi> install file:///wxs_sample_osgi_root/lib/com.google.protobuf_2.4.0a.jar
```

以下の出力が表示されます。

```
Bundle ID is 21
```

### サンプル・プラグイン・バンドルの概要:

OSGi サンプルには、カスタム `ObjectGridEventListener` や `MapSerializerPlugin` プラグインなどの eXtreme Scale プラグインを含む 5 つのサンプル・バンドルが含まれています。`MapSerializerPlugin` プラグインは Google Protocol Buffers サンプルと、`MapSerializerPlugin` サンプルが提供するメッセージを使用します。

次のバンドル、`ProtoBufSamplePlugins-1.0.0.jar` と `ProtoBufSamplePlugins-2.0.0.jar` は、`wxs_sample_osgi_root/lib` ディレクトリにあります。

`blueprint.xml` ファイルの内容は次のとおりです (コメントは削除してあります)。

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <bean id="myShardListener" class="com.ibm.websphere.samples.xs.proto.osgi.MyShardListenerFactory"/>
  <service ref="myShardListener" interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory" ranking="1">
  </service>
```

```

<bean id="myProtoBufSerializer" class="com.ibm.websphere.samples.xs.proto.osgi.ProtoMapSerializerFactory">
  <property name="keyType" value="com.ibm.websphere.samples.xs.serializer.app.proto.DataObjects1$OrderKey" />
  <property name="valueType" value="com.ibm.websphere.samples.xs.serializer.app.proto.DataObjects1$Order" />
</bean>

<service ref="myProtoBufSerializer" interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory"
  ranking="1">
</service>
</blueprint>

```

Blueprint XML ファイルは 2 つのサービス、myShardListener と myProtoBufSerializer をエクスポートします。これら 2 つのサービスは、protoBufObjectgrid.xml ファイル内で参照されます。

### サンプル・プラグイン・バンドルのインストール:

次のステップを実行して、ProtoBufSamplePlugins-1.0.0.jar バンドルをインストールします。

Equinox OSGi コンソールで次のコマンドを実行して、ProtoBufSamplePlugins-1.0.0.jar プラグイン・バンドルをインストールします。

```
osgi> install file:///wxs_sample_osgi_root/lib/ProtoBufSamplePlugins-1.0.0.jar
```

以下の出力が表示されます。

```
Bundle ID is 22
```

### レッスンのチェックポイント:

このレッスンでは、protobuf-java-2.4.0a-bundle.jar バンドルと ProtoBufSamplePlugins-1.0.0.jar プラグイン・バンドルをインストールしました。

## レッスン 2.5: OSGi バンドルの開始

WebSphere eXtreme Scale サーバーは、OSGi サーバー・バンドルとしてパッケージされます。このレッスンを完了して、eXtreme Scale サーバー・バンドル、およびインストールした他の OSGi バンドルをインストールします。

1. **ss** コマンドを実行して、各バンドルの ID を表示します。

```
osgi> ss
```

```
Framework is launched.
```

```

id State Bundle
0 ACTIVE org.eclipse.osgi_3.6.1.R36x_v20100806
1 ACTIVE org.eclipse.osgi.services_3.2.100.v20100503
2 ACTIVE org.eclipse.osgi.util_3.2.100.v20100503
3 ACTIVE org.eclipse.equinox.cm_1.0.200.v20100520
4 ACTIVE com.springsource.org.apache.commons.logging_1.1.1
5 ACTIVE com.springsource.org.aopalliance_1.0.0
6 ACTIVE org.springframework.aop_3.0.5.RELEASE
7 ACTIVE org.springframework.asm_3.0.5.RELEASE
8 ACTIVE org.springframework.beans_3.0.5.RELEASE
9 ACTIVE org.springframework.context_3.0.5.RELEASE
10 ACTIVE org.springframework.core_3.0.5.RELEASE
11 ACTIVE org.springframework.expression_3.0.5.RELEASE
12 ACTIVE org.apache.felix.fileinstall_3.0.2
13 ACTIVE net.luminis.cmc_0.2.5
15 ACTIVE org.eclipse.gemini.blueprint.core_1.0.0.RELEASE
16 ACTIVE org.eclipse.gemini.blueprint.extender_1.0.0.RELEASE

```



```
17 ACTIVE org.eclipse.gemini.blueprint.io_1.0.0.RELEASE
19 RESOLVED com.ibm.websphere.xs.server_7.1.1
21 RESOLVED Google_Protobuf_2.4.0
22 RESOLVED ProtoBufPlugins_1.0.0
```

2. インストールした各バンドルを開始します。特定の順序でバンドルを開始する必要があります。前の例でバンドル ID の順序を確認してください。
  - a. サンプル・プラグイン・バンドル ProtoBufPlugins\_1.0.0 を開始します。Equinox OSGi コンソールで次のコマンドを実行して、バンドルを開始します。この例では、サンプル・プラグインのバンドル ID は 22 です。

```
osgi> start 22
```
  - b. Google Protocol Buffers バンドル Google\_Protobuf\_2.4.0 を開始します。Equinox OSGi コンソールで次のコマンドを実行して、バンドルを開始します。この例では、Google Protocol Buffers プラグインのバンドル ID は 21 です。

```
osgi> start 21
```
  - c. サーバー・バンドル com.ibm.websphere.xs.server\_7.1.1 を開始します。OSGi コンソールで次のコマンドを実行して、サーバーを始動します。この例では、eXtreme Scale サーバー・バンドルのバンドル ID は 19 です。

```
osgi> start 19
```

サーバーを始動した後、MyShardListener イベント・リスナーが開始され、レコードの挿入または更新が可能になります。OSGi コンソールに次の出力が表示されると、プラグイン・バンドルが正常に開始されたことが確認できます。

```
SystemOut 0 MyShardListener@1253853884(version=1.0.0) order
com.ibm.websphere.samples.xs.serializer.proto.DataObjects1$Order$Builder
@1ab1aba(22) inserted
```

#### レッスンのチェックポイント:

このレッスンでは、OSGi フレームワーク用に構成した eXtreme Scale コンテナの中で、2 つのプラグイン・バンドルとサーバー・バンドルを開始しました。

## モジュール 3: eXtreme Scale サンプル・クライアントの実行

WebSphere eXtreme Scale サーバーが現在 OSGi 環境で実行中です。このモジュールのステップを完了して、データをグリッドに挿入する WebSphere eXtreme Scale クライアントを実行します。

### 学習目標

このモジュールのレッスンを完了すると、以下の作業を行う方法が分かります。

- グリッドに接続し、グリッドに対していくつかのデータを挿入または取得を行うクライアント・アプリケーションを実行します。
- 非 OSGi クライアント・アプリケーションを使用して、オーダーを開始します。

### 前提条件

モジュール 2: OSGi フレームワークでの eXtreme Scale バンドルのインストールおよび開始を完了していること。

## レッスン 3.1: クライアントを実行しサンプルをビルドする Eclipse のセットアップ

このレッスンを実行して、クライアントの実行とサンプル・プラグインのビルドに使用する Eclipse プロジェクトをインポートします。

サンプルには、グリッドに接続し、そのデータを挿入したり取得したりする Java SE クライアント・プログラムが含まれています。また、OSGi バンドルのビルドと再デプロイに使用できるプロジェクトも含まれています。

提供されるプロジェクトは、Eclipse 3.x 以上でテスト済みであり、標準の Java 開発プロジェクト・パースペクティブのみを必要とします。次のステップを実行して、WebSphere eXtreme Scale 開発環境をセットアップします。

1. Eclipse を新規ワークスペースまたは既存のワークスペースに開きます。
2. 「ファイル」メニューの「インポート」を選択します。
3. 「General」フォルダーを展開します。「既存プロジェクトをワークスペースへ」を選択し、「次へ」をクリックします。
4. 「ルート・ディレクトリーの選択」フィールドで、`wxs_sample_osgi_root` ディレクトリーと入力するか、参照して指定します。「終了」をクリックします。ワークスペースに新規プロジェクトがいくつか表示されます。2 つのユーザー・ライブラリーを定義することによってビルド・エラーは修正されます。次のステップを実行して、ユーザー・ライブラリーを定義します。
5. 「ウィンドウ」メニューから「設定」を選択します。
6. 「Java」 > 「ビルド・パス」ブランチを展開し、「ユーザー・ライブラリー」を選択します。
7. eXtreme Scale ユーザー・ライブラリーを定義します。
  - a. 「新規」をクリックします。
  - b. 「ユーザー・ライブラリー名」フィールドに「eXtremeScale」と入力し、「OK」をクリックします。
  - c. 新規ユーザー・ライブラリーを選択し、「JAR の追加」をクリックします。
    - 1) `wxs_install_root/lib` ディレクトリーを参照し、`objectgrid.jar` ファイルを選択します。「OK」をクリックします。
    - 2) ObjectGrid API の API 資料を組み込むには、前のステップで追加した `objectgrid.jar` ファイルの API 資料のロケーションを選択します。「編集」をクリックします。
    - 3) API 資料のロケーション・パス・ボックスで、ディレクトリー `wxs_install_root/docs/javadoc.zip` に含まれている `Javadoc.zip` ファイルを選択します。
8. Google Protocol Buffers ユーザー・ライブラリーを定義します。
  - a. 「新規」をクリックします。
  - b. 「ユーザー・ライブラリー名」フィールドに `com.google.protobuf` と入力し、「OK」をクリックします。
  - c. 新規ユーザー・ライブラリーを選択し、「JAR の追加」をクリックします。

- 1) `wxs_sample_osgi_root/lib` ディレクトリーから、  
`com.google.protobuf_2.4.0.a.jar` ファイルを参照して選択します。  
「OK」をクリックします。

### レッスンのチェックポイント:

このレッスンでは、サンプル Eclipse プロジェクトをインポートし、ビルド・エラーを修正するユーザー・ライブラリーを定義しました。

## レッスン 3.2: クライアントの始動とグリッドへのデータの挿入

このレッスンを完了して、非 OSGi クライアントを始動して、クライアント・アプリケーションを実行します。

Java クライアント・アプリケーションは、`com.ibm.websphere.samples.xs.proto.client.Client` です。この Java クライアント・アプリケーションは Eclipse プロジェクト `wxs.sample.osgi.protobuf.client` に含まれています。メイン・クラス・ファイルは `com.ibm.websphere.samples.xs.proto.client.Client` です。

このクライアントはクライアント・オーバーライド ObjectGrid 記述子 XML ファイルを使用して OSGi 構成をオーバーライドします。その結果、このクライアントは非 OSGi 環境で実行可能となります。コメントおよびヘッダーが削除された、次のファイルの内容を参照してください。フォーマット設定のために、コードの 1 行が複数行に分けられている場合があります。

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="Grid" txTimeout="15">
      <bean id="ObjectGridEventListener" className="" osgiService="" />
      <backingMap name="Map" readOnly="false"
        lockStrategy="PESSIMISTIC" lockTimeout="5"
        copyMode="COPY_TO_BYTES" pluginCollectionRef="serializer"/>
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="serializer">

      <bean id="MapSerializer"
        className="com.ibm.websphere.samples.xs.serializer.proto.ProtoMapSerializer"
        osgiService="">
        <property name="keyType" type="java.lang.String"
          value="com.ibm.websphere.samples.xs.serializer.proto.DataObjects2$OrderKey" />
        <property name="valueType" type="java.lang.String"
          value="com.ibm.websphere.samples.xs.serializer.proto.DataObjects2$Order" />
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

「次を実行」 > 「Java アプリケーション」をクリックして、クライアント・アプリケーションを実行します。

アプリケーションを実行すると、次のメッセージが表示されます。メッセージは、オーダーが挿入されたことを示します。

```
order
com.ibm.websphere.samples.xs.serializer.proto.DataObjects1$Order$Builder@5d165d16(5000000) inserted
```

## レッスンのチェックポイント:

このレッスンでは、オーダーを生成する、`com.ibm.websphere.samples.xs.proto.client.Client` アプリケーションを開始しました。

## モジュール 4: サンプル・バンドルの照会とアップグレード

このモジュールのレッスンでは、`xscmd` コマンドを使用して、サンプル・バンドルのサービス・ランキングを照会したり、それを新しいサービス・ランキングにアップグレードしたり、新しいサービス・ランキングを検査したりします。

### 学習目標

このモジュールのレッスンを完了すると、以下のタスクの実行方法がわかります。

- サービスの現在のサービス・ランキングを照会する。
- すべてのサービスの現在のランキングを照会する。
- サービスのすべての使用可能なランキングを照会する。
- すべての使用可能なサービス・ランキングを照会する。
- `xscmd` ツールを使用して、特定のサービス・ランキングが使用可能かどうか確認する。
- サンプル OSGi サービスのサービス・ランキングを更新する。

### 前提条件

モジュール 3: eXtreme Scale サンプル・クライアントの実行を完了してください。

### レッスン 4.1: サービス・ランキングの照会

このレッスンを実行して、現在のサービス・ランキングやアップグレードに使用可能なサービス・ランキングを照会します。

- サービスの現在のサービス・ランキングを照会します。 次のコマンドを入力して、サービス `myShardListener` に現在使用されているサービス・ランキングを照会します。このサービスは、`Grid` という `ObjectGrid` と `MapSet` というマップ・セットで使用されます。

1. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

2. 次のコマンドを入力して、サービス `myShardListener` の現在のサービス・ランキングを照会します。

```
./xscmd.sh -c osgiCurrent -g Grid -ms MapSet -sn myShardListener
```

以下の出力が表示されます。

```
OSGi Service Name: myShardListener
ObjectGrid Name MapSet Name Server Name      Current Ranking
-----
Grid           MapSet      collocatedServer  1
```

```
CWXS10040I: The command osgiCurrent has completed successfully.
```

- すべてのサービスの現在のランキングを照会します。 次のコマンドを入力して、Grid という ObjectGrid と MapSet というマップ・セットで使用されるすべてのサービスの現在のサービス・ランキングを照会します。

1. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

2. 次のコマンドを入力して、すべてのサービスの現在のサービス・ランキングを照会します。

```
./xscmd.sh -c osgiCurrent -g Grid -ms MapSet
```

以下の出力が表示されます。

OSGi Service Name	Current Ranking	ObjectGrid Name	MapSet Name	Server Name
myProtoBufSerializer	1	Grid	MapSet	collocatedServer
myShardListener	1	Grid	MapSet	collocatedServer

```
CWXS10040I: The command osgiCurrent has completed successfully.
```

- サービスのすべての使用可能なランキングを照会します。 次のコマンドを入力して、myShardListener というサービスのすべての使用可能なサービス・ランキングを照会します。

1. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

2. 次のコマンドを入力して、サービスのすべての使用可能なランキングを照会します。

```
./xscmd.sh -c osgiAll -sn myShardListener
```

以下の出力が表示されます。

```
Server: collocatedServer
OSGi Service Name Available Rankings
-----
myShardListener 1
```

```
Summary - All servers have the same service rankings.
```

```
CWXS10040I: The command osgiAll has completed successfully.
```

出力はサーバー別にグループ化されます。この例の場合は、サーバー collocatedServer しか存在しません。

- すべての使用可能なサービス・ランキングを照会します。 次のコマンドを入力して、すべてのサービスのすべての使用可能なサービス・ランキングを照会します。

1. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

2. 次のコマンドを入力して、すべての使用可能なサービス・ランキングを照会します。

```
./xscmd.sh -c osgiAll
```

以下の出力が表示されます。

```
Server: collocatedServer
OSGi Service Name Available Rankings
-----
myProtoBufSerializer 1
```

Summary - All servers have the same service rankings.

- バージョン 2 のプラグイン・バンドルをインストールして開始します。サーバー OSGi コンソールで、新規バージョンの Order クラスと MapSerializerPlugin プラグインを含んでいる新規バンドルをインストールします。

ProtoBufSamplePlugins-2.0.0.jar バンドルのインストール方法の詳細については、レッスン 2.4: Google Protocol Buffers バンドルとサンプル・プラグイン・バンドルのインストールを参照してください。

- インストール後、新規バンドルを開始します。新規バンドルのサービスは使用可能ですが、eXtreme Scale サーバーはまだそれを使用していません。特定バージョンのサービスを使用するには、サービス更新要求を実行しなければなりません。
- ここで、すべての使用可能なサービス・ランキングを再度照会すると、サービス・ランキング 2 が出力に追加されます。
  - 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

- 次のコマンドを入力して、すべての使用可能なサービス・ランキングを照会します。

```
./xscmd.sh -c osgiAll
```

以下の出力が表示されます。

```
Server: collocatedServer
  OSGi Service Name  Available Rankings
-----
myProtoBufSerializer 1, 2
myShardListener      1, 2
```

Summary - All servers have the same service rankings.

#### レッスンのチェックポイント:

このチュートリアルでは、現在指定されているサービス・ランキングとすべての使用可能なサービス・ランキングを照会しました。また、インストールして開始した新規バンドルのサービス・ランキングも表示しました。

### レッスン 4.2: 特定のサービス・ランキングが使用可能かどうかの判別

このレッスンを完了して、指定したサービス名の特定のサービス・ランキングが使用可能かどうか判別します。

- 次のコマンドを入力して、サービス・ランキング 2 の myShardListener という名前のサービスと、サービス・ランキング 2 の myProtoBufSerializer という名前のサービスが使用可能かどうか判別します。サービス・ランキング・リストは、-sr オプションを使用して渡されます。

- 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

- 次のコマンドを入力して、サービスが使用可能かどうか判別します。

```
./xscmd.sh -c osgiCheck -sr "myShardListener;2,myProtoBufSerializer;2"
```



以下の出力が表示されます。

```
CWXS10040I: The command osgiCheck has completed successfully.
```

2. 次のコマンドを入力して、サービス・ランキング 2 の myShardListener という名前のサービスと、サービス・ランキング 3 の myProtoBufSerializer という名前のサービスが使用可能かどうか判別します。

- a. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

- b. 次のコマンドを入力して、サービスが使用可能かどうか判別します。

```
./xscmd.sh -c osgiCheck -sr "myShardListener;2,myProtoBufSerializer;3"
```

以下の出力が表示されます。

Server	OSGi Service	Unavailable Rankings
-----	-----	-----
collocatedServer	myProtoBufSerializer	3

### レッスンのチェックポイント:

このレッスンでは、myShardListener および myProtoBufSerializer というサービスを特定のサービス・ランキングと一緒に指定して、これらのランキングが使用可能かどうか判別しました。

## レッスン 4.3: サービス・ランキングの更新

このレッスンを完了して、照会した現行サービス・ランキングを更新します。

1. サービス myShardListener および myProtoBufSerializer のサービス・ランキングをサービス・ランキング 2 に更新します。サービス・ランキング・リストは -sr オプションを使用して渡されます。

- a. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

- b. 次のコマンドを入力して、サービス・ランキングを更新します。

```
./xscmd.sh -c osgiUpdate -g Grid -ms MapSet -sr "myShardListener;2,myProtoBufSerializer;2"
```

以下の出力が表示されます。

```
Update succeeded for the following service rankings:
```

Service	Ranking
-----	-----
myProtoBufSerializer	2
myShardListener	2

```
CWXS10040I: The command osgiUpdate has completed successfully.
```

OSGi コンソールに、次の出力が表示されます。

```
SystemOut 0 MyShardListener@326505334(version=2.0.0) order  
com.ibm.websphere.samples.xs.serializer.proto.DataObjects2$Order$Builder@  
22342234(34) updated
```

MyShardListener サービスがバージョン 2.0.0 で、それがサービス・ランキング 2 になっていることに注意してください。

2. **xscmd** コマンドを実行して、Grid という名前の ObjectGrid および MapSet という名前のマップ・セットが使用するすべてのサービスの現在のサービス・ランキングを照会します。

- a. 次のディレクトリーに切り替えます。

```
cd wxs_home/bin
```

- b. 次のコマンドを入力して、Grid および MapSet によって使用されるすべてのサービスのサービス・ランキングを照会します。

```
./xscmd.sh -c osgiCurrent -g Grid -ms MapSet
```

以下の出力が表示されます。

OSGi Service Name	Current Ranking	ObjectGrid Name	MapSet Name	Server Name
myProtoBufSerializer	2	Grid	MapSet	collocatedServer
myShardListener	2	Grid	MapSet	collocatedServer

```
CWXS10040I: The command osgiCurrent has completed successfully.
```

### レッスンのチェックポイント:

このレッスンでは、myShardListener サービスと myProtoBufSerializer サービスのサービス・ランキングを更新しました。

---

## 第 4 章 インストール



WebSphere eXtreme Scale は、複数のサーバーにまたがるアプリケーション・データおよびビジネス・ロジックの区画化、複製、および管理を動的に行うために使用できるメモリー内のデータ・グリッドです。デプロイメントの目的および要件を決定した後に、eXtreme Scale をシステムにインストールします。

### 開始する前に

- インストールを開始する前に、WebSphere eXtreme Scale キャッシング・アーキテクチャー、キャッシュおよびデータベース統合、シリアライゼーション、スケラビリティ、および可用性について理解しておく必要があります。詳しくは、製品概要を参照してください。
- WebSphere eXtreme Scale デプロイメントを計画します。各キャッシング・トポロジー、見積もり情報などの詳細については、11 ページの『第 2 章 計画』を参照してください。
- ご使用の環境が eXtreme Scale をインストールするための前提条件を満たしていることを確認してください。詳しくは、59 ページの『ハードウェアおよびソフトウェア要件』を参照してください。
- 環境やその他の要件の詳細については、167 ページの『インストールの計画』を参照してください。
- 前のバージョンの WebSphere eXtreme Scale のアップグレードをインストールする場合は、239 ページの『eXtreme Scale サーバーの更新』のステップに従ってください。

---

## インストールの概要

WebSphere eXtreme Scale は、スタンドアロン環境または WebSphere Application Server 環境にインストールすることができます。WebSphere eXtreme Scale バージョン 8.5 のインストールには、IBM Install Manager のインストールと製品ファイルの取得が前提となります。Installation Manager をインストールし、適切な製品リポジトリへのアクセスをセットアップした後は 2 つの選択肢があります。1 つは、WebSphere eXtreme Scale のフルインストールまたはクライアント・インストールをスタンドアロン環境で行う選択肢で、もう 1 つは、同製品を WebSphere Application Server 環境にインストールするというものです。

### 8.5+

#### IBM Installation Manager

Installation Manager は、リモートまたはローカルにあるソフトウェアのフラット・ファイル・リポジトリを使用して、新しい WebSphere eXtreme Scale 製品をインストール、変更、更新することができる単一のインストール・プログラムです。Installation Manager は、使用可能なパッケージ (製品、フィックスパック、インテリム・フィックスなど) の判別および表示、前提条件および相互依存性のチェッ



ク、選択されたパッケージのインストールを行います。また、Installation Manager を使用して、Installation Manager によりインストールされたパッケージを容易にアンインストールすることもできます。

**IBM Installation Manager の概要:** IBM Installation Manager は、広範囲のコンピューター・システムで実行できる、ソフトウェアのインストールおよび更新の汎用ツールです。Installation Manager は、グラフィカル・ユーザー・インターフェース (GUI) またはコマンド行インターフェースから呼び出すことができます。また、XML で応答ファイルを作成することもでき、それに従って Installation Manager タスクをサイレント・モードで実行させることもできます。

Installation Manager の使用について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

**パッケージとパッケージ・グループ:** Installation Manager を使用してインストールすることができるソフトウェア製品は、パッケージと呼ばれます。インストールされたパッケージには製品レベルとインストール・ロケーションがあります。パッケージ・グループは、単一のロケーションにインストールされるすべての製品で構成されます。

**Installation Manager のモード:** IBM Installation Manager は以下の 3 つのモードのいずれかでインストールすることができます。

- 管理者モードでは、Installation Manager は管理者 ID または root ID からインストールされ、すべての管理者または root ユーザーが呼び出すことができます。
- 非管理者モード (ユーザー・モードとも呼ばれます) では、Installation Manager は、それをインストールしたユーザーのみが呼び出すことができます。
-   グループ・モードでは、Installation Manager は、それをインストールしたユーザーのデフォルト・グループに属しているすべてのユーザー ID から呼び出すことができます。

これは、IBM Installation Manager の単一インスタンスを 2 人のユーザーが同時に使用できるという意味ではありません。

**必要な Installation Manager の数:** 製品コードをインストールまたは更新するシステム上で実行する必要がある Installation Manager は 1 つだけです。Installation Manager は複数の製品のインストールを把握することができるため、通常、1 つのシステムに必要な Installation Manager は 1 つだけです。

**Installation Manager のインストール:** インストール・キットがシステムで使用可能になると、Installation Manager をインストールすることができます。Installation Manager は、インストール・キットからコピーされるバイナリーのセットと、特にこの Installation Manager がインストールした製品について記述するランタイム・データのセットで構成されます。Installation Manager をインストールする前に、Installation Manager が作動するときのモードと、バイナリーおよびランタイム・データ (エージェント・データあるいはアプリケーション・データと呼ばれる) が常駐する場所を決める必要があります。その後、適切なユーザー ID から Installation Manager のインストール・コマンドを出して Installation Manager をインストールします。

**製品リポジトリへのアクセス:** IBM Installation Manager でインストールされるすべてのソフトウェア資料は、フラット・ファイル・リポジトリに保管されています。各リポジトリには、1 つ以上のパッケージ (すなわち、特定のレベルのソフトウェア製品) に関するプログラム・オブジェクトおよびメタデータが含まれています。リポジトリにはフィックスバックやインテリム・フィックスなどの製品保守が含まれることもあります。新しい製品をインストールする際は、アクセス可能なすべてのリポジトリで使用可能な製品レベルから選択することができます。

**製品のインストール:** Installation Manager のインストールが完了し、必要なすべての製品リポジトリへのアクセスが可能になったならば、Installation Manager の GUI、コマンド行コマンド、または応答ファイルを使用して実際に製品をインストールすることができます。製品をインストールするには、パッケージ名、インストールする製品レベル、製品ロケーション、およびその他のプロパティを指定します (パッケージ名と製品ロケーション以外はオプションです)。例えば、製品によっては、インストール時に選択できるオプション機能や、選択可能なオプションのサポート対象言語バックのリストを備えたものもあります。

**インストール済み製品に関する作業:** Installation Manager のコマンドを使用して、インストール済みの製品および製品レベルのリストを取得することができます。この情報、つまり WebSphere eXtreme Scale 製品のインストール済みコピーについての情報は、製品ファイル・システムから **versionInfo** コマンドを出すことによっても取得することができます。Installation Manager のコマンドまたは応答ファイルを使用して、新しい製品レベルをインストールしたり、以前のレベルにロールバックしたり、オプション機能や言語バックの追加や削除によって製品を変更したりすることができます。

**IBM Packaging Utility の使用:** Packaging Utility を使用すれば、インストール・リポジトリ用のパッケージを作成したり管理したりすることができます。複数のパッケージを 1 つのリポジトリにコピーしたり、1 つの製品の複数のディスクを同じリポジトリにコピーしたりすることができます。例えば、Passport Advantage からリポジトリにパッケージをコピーすることができます。Packaging Utility について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターに移動してください。

#### 制約事項:

- **Windows** 非管理者がユーザー アカウント制御 (UAC) が有効にされたままの状態では Windows Vista、Windows 7、または Windows Server 2008 オペレーティング・システムの Program Files または Program Files (x86) ディレクトリに WebSphere eXtreme Scale バージョン 8.5 をインストールすると、WebSphere eXtreme Scale は正しく機能しません。

UAC は、非管理者に対して Program Files または Program Files (x86) ディレクトリへのソフトウェア製品のインストールを許可するアクセス制御メカニズムですが、インストールが完了すると、そのディレクトリへの書き込みアクセスがすべて禁止されてしまいます。WebSphere eXtreme Scale は、正しく機能するためには、`app_server_root` ディレクトリへの書き込みアクセスを必要とします。

この問題を解決するには、以下の作業のいずれかを実行してください。

- WebSphere eXtreme Scale を Program Files または Program Files (x86) 以外のディレクトリーにインストールします。

例:

C:\IBM\WebSphere\AppServer

- UAC を使用不可に設定します。

- ローカル・リポジトリーがある Installation Manager を使用して製品をインストールする場合、圧縮されたりリポジトリー・ファイルを解凍せずにそのまま使用すると、インストールにかなり長い時間がかかります。

ローカル・リポジトリーを使用して製品をインストールする前に、圧縮リポジトリー・ファイルをローカル・システム上のロケーションに解凍し、その後で Installation Manager を使用してリポジトリー・ファイルにアクセスするようにしてください。

- Installation Manager コンソール・モード (これは Installation Manager バージョン 1.4.3 以降に含まれています) は、z/OS® 以外のシステム上の WebSphere eXtreme Scale バージョン 8.5 オファリングでは機能しません。

**重要:** リポジトリーの内容を非バイナリー・モードで転送したり、解凍時に内容を変換したりしないでください。

**ヒント:** インフォメーション・センターのこのセクションに記載されている指示や説明はそのほとんどが IBM Installation Manager の旧バージョンに対応しますが、ここに示す情報は Installation Manager バージョン 1.5 以降のインストール・ユーザーまたはアップグレード・ユーザー向けに最適化したものです。

## スタンドアロン環境または WebSphere Application Server 環境における WebSphere eXtreme Scale のインストールについて

スタンドアロン環境での WebSphere eXtreme Scale のインストールを選択した場合は、カタログ・サーバーとコンテナ・サーバーの両方を実行します。データ・グリッドにアクセスするクライアント・アプリケーションを実行しているサーバーがある場合は、クライアントのみのインストールを使用することができます。カタログ・サーバーまたはコンテナ・サーバーを実行しているノード上では、サーバーのみのインストールまたはサーバーとクライアントのインストールを選択してください。

### • フルインストール (クライアントとサーバーのインストール):

- WebSphere Application Server へのインストール中に、クライアントのみをインストールするか、サーバーとクライアントの両方をインストールするかを選択できます。
- スタンドアロン環境でインストールしている場合は、サーバーとクライアントの両方をインストールできます。

### • クライアントのインストール:

クライアント・アプリケーションを実行しているノード上では、クライアントのみのインストールを使用できます。

### • WebSphere Application Server 環境:



WebSphere eXtreme Scale を WebSphere Application Server 環境のノードにインストールすることによって、デプロイメント・マネージャーやその他のアプリケーション・サーバーと同じセル内でカタログ・サーバーとコンテナ・サーバーを自動的に始動できます。

- **スタンドアロン環境:**

スタンドアロン・インストールでは、WebSphere Application Server のない環境に WebSphere eXtreme Scale をインストールします。スタンドアロン環境の場合、カタログ・サーバーとコンテナ・サーバーのプロセスを手動で構成し、開始します。

## 8.5+

### Installation Manager で使用可能な製品オフリング

Installation Manager のインストールを完了し、かつ必要なすべての製品リポジトリにアクセスできるようになると、Installation Manager は使用可能な製品オフリングのリストを提供します。本製品の製品リポジトリをポイントすると、以下の製品オフリングが表示されるはずです。

- WebSphere eXtreme Scale (スタンドアロン環境)
- WebSphere eXtreme Scale クライアント (スタンドアロン環境)
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 6.0
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 7.0
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 8.0
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6.0
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7.0
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8.0

**重要:** 製品オフリング WebSphere eXtreme Scale for WebSphere Application Server をインストールする場合は、WebSphere Application Server 製品リポジトリをポイントする必要があります。これらは 2 つの別々の製品としてインストールされますが、Installation Manager では同時にインストールすることができます。

---

## インストールの計画

製品をインストールする前に、使用する環境について検討する必要があります。

### インストール・トポロジー

WebSphere eXtreme Scale を使用すると、スタンドアロン・サーバーまたは WebSphere Application Server、あるいはその両方を含む多くのインストール・トポロジーを作成できます。

## 開発ノード

最も単純なインストールのシナリオは、開発ノードを作成することです。このシナリオでは、アプリケーションを開発するノード上で WebSphere eXtreme Scale のクライアントおよびサーバー・インストール済み環境を一度にインストールします。

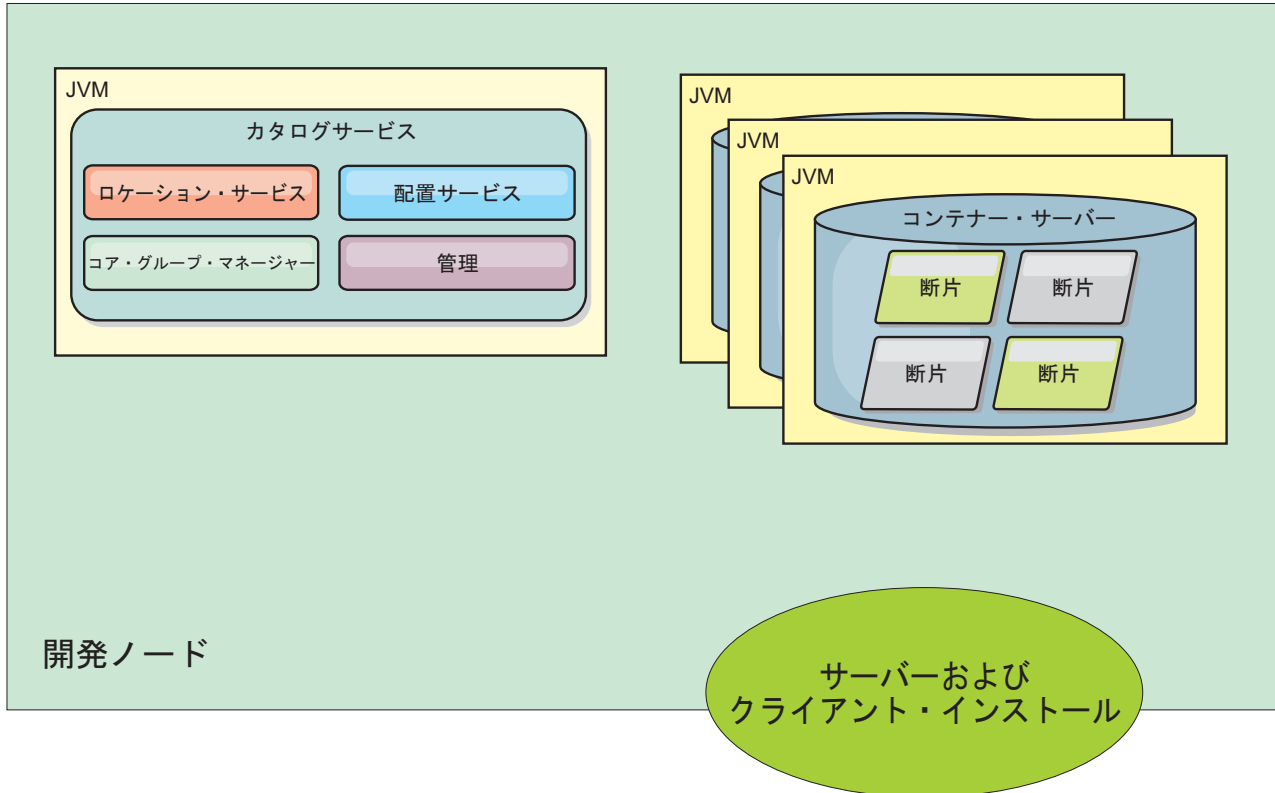


図 22. 開発ノード

開発ノード上でインストールが完了した後、開発環境を構成して、アプリケーションの作成を開始することができます。

### スタンドアロン・トポロジー

スタンドアロン・トポロジーは、WebSphere Application Server 上で実行中でないサーバーから構成されます。多くのさまざまなスタンドアロン・トポロジーを作成できますが、次のトポロジーが例として含まれます。このトポロジーには、2 つのデータ・センターがあります。各データ・センターでは、WebSphere eXtreme Scale のフルインストール済み環境 (クライアントおよびサーバー) とクライアントのみのインストール済み環境が、物理サーバーにインストールされています。クライアントのみのインストール済み環境は、データ・グリッドを使用している Web アプリケーションが実行されているノード上にあります。これらのノードは、カタログ・サーバーもコンテナ・サーバーも実行しないため、サーバー・インストールは必要ありません。構成の中で、マルチマスター・リンクが 2 つのカタログ・サービス・ドメインを接続しています。マルチマスター・リンクによって、異なるデータ・センターのコンテナ・サーバー内の断片間のレプリカ生成が可能になります。

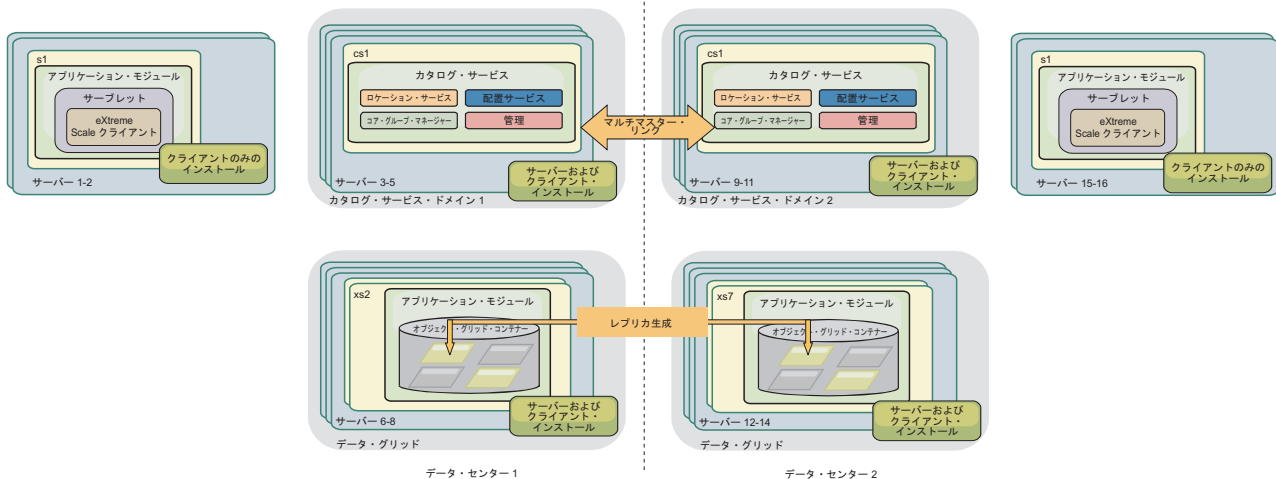


図 23. 2 つのデータ・センターがあるスタンドアロン・トポロジー

スタンドアロン・トポロジーを使用する利点は次のとおりです。

- ベンダー・フレームワークおよびライブラリーに組み込むことができる、柔軟な統合オプション。
- WebSphere Application Server トポロジーよりも少ない占有スペース。
- WebSphere Application Server トポロジーよりも少ないライセンス交付要件。
- 拡張された Java ランタイム環境 (JRE) オプション。

## WebSphere Application Server トポロジー

完全に WebSphere Application Server セルの中で稼働するインストール済み環境を作成することもできます。クライアント、カタログ・サーバー、およびコンテナ・サーバーには、それぞれ関連付けられたクラスターがあります。アプリケーションを実行するノードには、クライアントのみのインストール済み環境があります。その他のノードには、クライアントおよびサーバー・インストール済み環境があります。

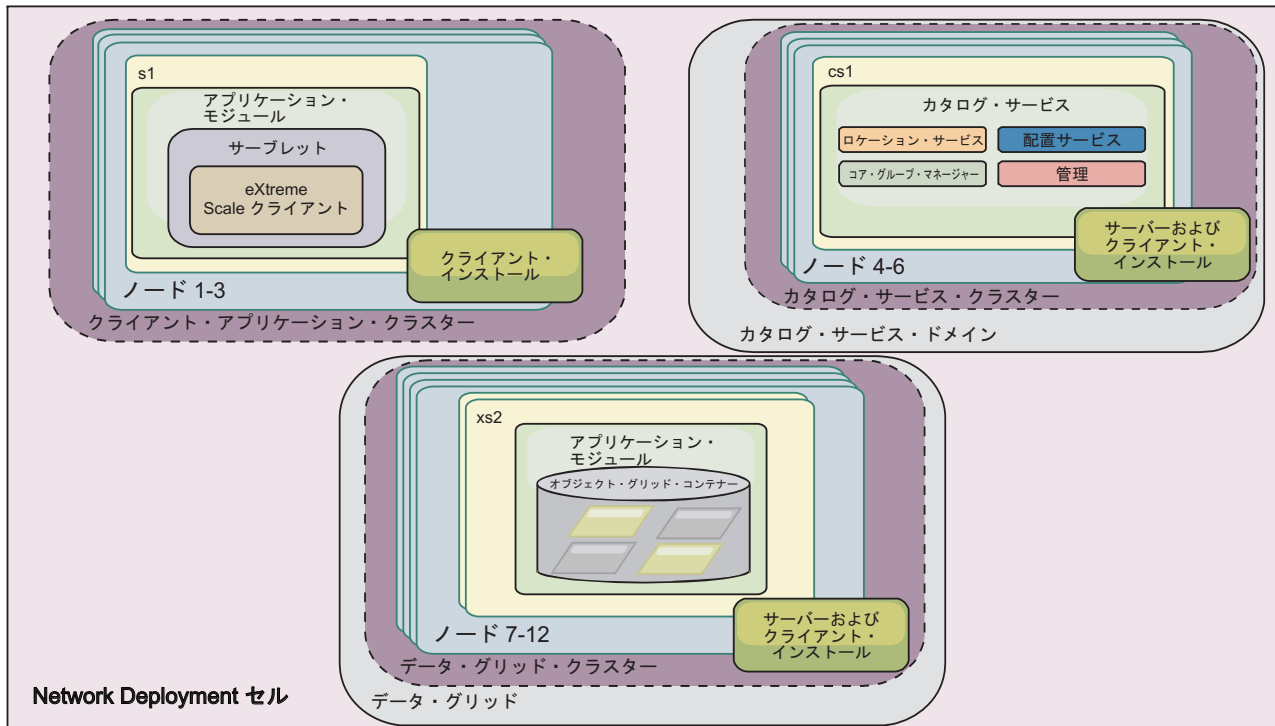


図 24. WebSphere Application Server トポロジー例

WebSphere Application Server トポロジーを使用する利点は次のとおりです。

- 集中化され、一貫した管理および構成。
- セキュリティー統合。
- Java EE アプリケーション統合。
- Performance Monitoring Infrastructure (PMI) 統合。
- WebSphere Application Server コンポーネント (OpenJPA L2 キャッシュ、動的キャッシュ、および HTTP セッション・パーシスタンス) との統合。

### 混合トポロジー

WebSphere Application Server とスタンドアロン・サーバーの両方を含んだ混合トポロジーを作成できます。次の例では、クライアント・アプリケーションは WebSphere Application Server セルの中で実行される一方、カタログ・サーバーおよびコンテナ・サーバーはスタンドアロン・モードで実行されます。

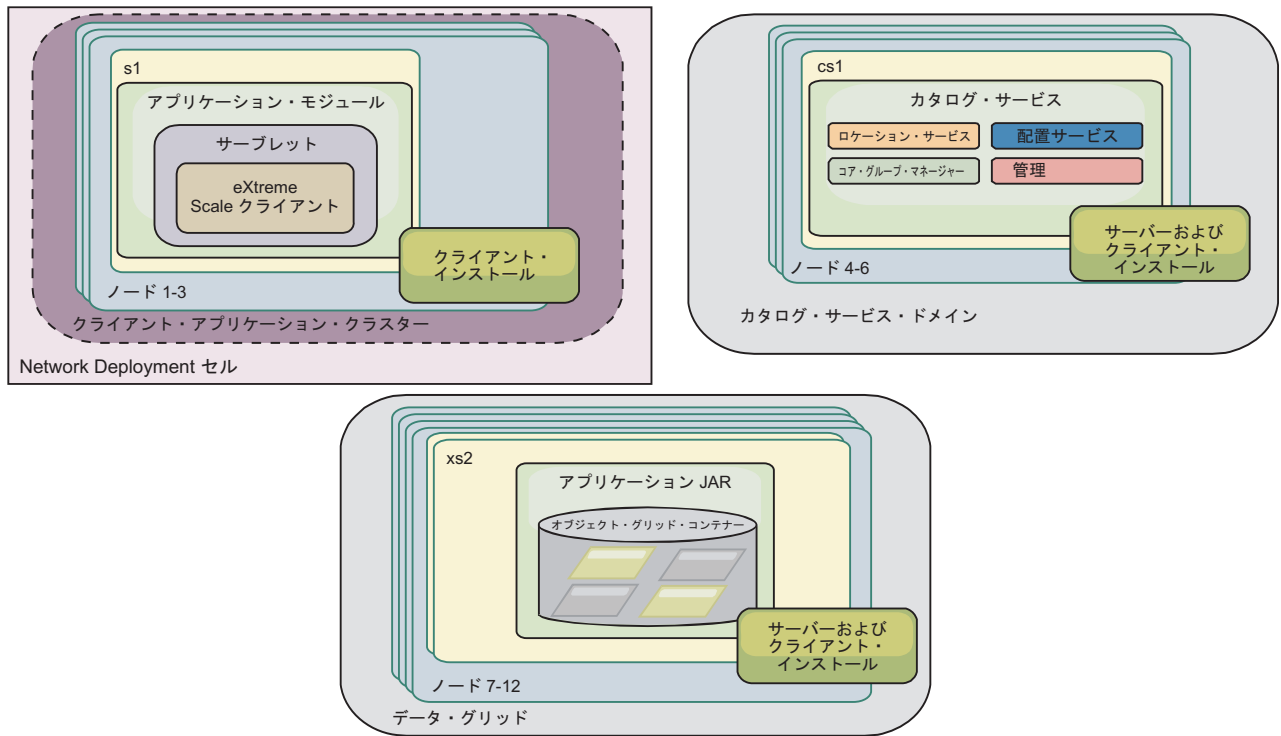


図 25. 混合トポロジー例

## ハードウェアおよびソフトウェア要件

ハードウェア要件およびオペレーティング・システム要件の概要をご覧ください。WebSphere eXtreme Scale に対して使用するハードウェアまたはオペレーティング・システムのレベルについて、特定のレベルの要件はありませんが、公式にサポートされるハードウェアおよびソフトウェアのオプションは、製品サポート・サイトの「システム要件」ページから入手できます。インフォメーション・センターの情報と「システム要件」ページの情報に違いがある場合は、Web サイトの情報を優先してください。インフォメーション・センターの前提条件の情報は、便宜上提供されているだけです。

ハードウェアおよびソフトウェア要件の正式なセットについては、システム要件ページを参照してください。

eXtreme Scale のインストールおよびデプロイ先として、オペレーティング・システムの特定期限の要件はありません。Java Platform, Standard Edition (Java SE) および Java Platform, Enterprise Edition (Java EE) のそれぞれのインストールでは、異なるオペレーティング・システムのレベルまたはフィックスが必要です。

この製品は、Java EE および Java SE 環境にインストールしてデプロイできます。また、クライアント・コンポーネントを WebSphere Application Server に統合せずに、直接 Java EE アプリケーションにバンドルすることができます。WebSphere eXtreme Scale は、Java SE 5 以降、および WebSphere Application Server バージョン 6.1 以降をサポートしています。

## ハードウェア要件

WebSphere eXtreme Scale では、ハードウェアの具体的なレベルの要件はありません。ハードウェア要件は、WebSphere eXtreme Scale を実行するのに使用される Java Platform, Standard Edition のインストール済み環境でサポートされるハードウェアによって異なります。eXtreme Scale を WebSphere Application Server または別の Java Platform, Enterprise Edition 実装環境で使用する場合、これらのプラットフォームのハードウェア要件は WebSphere eXtreme Scale にとって十分です。

## オペレーティング・システム要件

**8.5+** WebSphere eXtreme Scale では、オペレーティング・システムの具体的なレベルの要件はありません。各 Java SE および Java EE 実装は、それぞれ異なるオペレーティング・システム・レベル、または、Java 実装のテスト中に発見された問題に対するフィックスを必要とします。これらの実装に必要なレベルは、eXtreme Scale にとって十分です。

### 8.5+

## Installation Manager の要件

WebSphere eXtreme Scale をインストールする前に、Installation Manager をインストールする必要があります。Installation Manager をインストールするには、製品メディアを使用するか、Passport Advantage サイトから入手したファイルを使用するか、あるいは、IBM Installation Manager ダウンロード Web サイトにある Installation Manager の最新バージョンが入っているファイルを使用します。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品 オファリングのインストール』を参照してください。

## Web ブラウザー要件

Web コンソールは、以下の Web ブラウザーをサポートしています。 **8.5+**

- Mozilla Firefox、バージョン 3.5.x 以降
- Microsoft Internet Explorer バージョン 7 以降

## WebSphere Application Server 要件

### 8.5+

- WebSphere Application Server バージョン 6.1.0.41 以降
- WebSphere Application Server バージョン 7.0.0.21 以降
- WebSphere Application Server バージョン 8.0.0.2 以降

詳しくは、WebSphere Application Server の推奨フィックスを参照してください。

## その他のアプリケーション・サーバー要件

その他の Java EE 実装は、ローカル・インスタンスとして、または、eXtreme Scale サーバーへのクライアントとして、eXtreme Scale ランタイムを使用できます。Java SE を実装する場合は、バージョン 5 以降を使用する必要があります。



## WebSphere eXtreme Scale product offering IDs

製品の更新をインストールする場合、あるいはフィックスをロールバックする場合、コマンド行でオファリング ID を指定する必要があります。製品オファリングは、下の表を使用して特定してください。

表 3. WebSphere eXtreme Scale 製品のオファリング ID

製品名	オファリング ID
スタンドアロン環境での WebSphere eXtreme Scale	com.ibm.websphere.WXS.v85
スタンドアロン環境での WebSphere eXtreme Scale クライアント	com.ibm.websphere.WXSCLIENT.v85
WebSphere eXtreme Scale for WebSphere Application Server バージョン 6	com.ibm.websphere.WXS.was6.v85
WebSphere eXtreme Scale for WebSphere Application Server バージョン 7	com.ibm.websphere.WXS.was7.v85
WebSphere eXtreme Scale for WebSphere Application Server Version 8	com.ibm.websphere.WXS.was8.v85
WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6	com.ibm.websphere.WXSCLIENT.was6.v85
WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7	com.ibm.websphere.WXSCLIENT.was7.v85
WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8	com.ibm.websphere.WXSCLIENT.was8.v85

## Java SE の考慮事項

WebSphere eXtreme Scale は Java SE 5、Java SE 6、または Java SE 7 を必要とします。一般に、Java SE は、バージョンが新しい方が機能性もパフォーマンスも優れています。

### サポートされるバージョン

WebSphere eXtreme Scale は Java SE 5、Java SE 6、および Java SE 7 と一緒に使用することができます。使用するバージョンは、Java ランタイム環境 (JRE) ベンダーによって現在サポートされているものでなければなりません。

IBM Runtime Environment, Java Technology Edition バージョン 5、バージョン 6、およびバージョン 7 は、製品と一緒に広く使用するためにサポートされています。バージョン 6 サービス・リリース 9 フィックスパック 2 は完全にサポートされる JRE です。この JRE は、スタンドアロン WebSphere eXtreme Scale および WebSphere eXtreme Scale クライアント インストールの一部として `wxs_install_root/java` ディレクトリーにインストールされるもので、クライアントとサーバーの両方で使用することができます。WebSphere Application Server 内に WebSphere eXtreme Scale をインストールする場合は、WebSphere Application Server インストールに含まれている JRE を使用できます。Web コンソールの場合は、IBM Runtime Environment, Java Technology Edition バージョン 6 サービス・リリース 7 およびそれ以降のサービス・リリースのみを使用する必要があります。

WebSphere eXtreme Scale は、Java Development Kit (JDK) バージョン 5、バージョン 6、およびバージョン 7 の機能を、これが使用可能になったときに使用します。一般に、Java Development Kit (JDK) および Java SE は、バージョンが新しい方がパフォーマンスおよび機能が優れています。

詳しくは、サポートされるソフトウェアを参照してください。

## Java に依存している WebSphere eXtreme Scale フィーチャー

表 4. Java SE 5、Java SE 6、および Java SE 7 を必要とするフィーチャー：

WebSphere eXtreme Scale は、Java SE 5 または Java SE 6 で導入された機能を使用して、以下の製品フィーチャーを提供します。

フィーチャー	Java SE 5 以降のサービス・リリースでサポートされる	Java SE バージョン 6、バージョン 7 以降のサービス・リリースでサポートされる
EntityManager API アノテーション (オプション: XML ファイルも使用できます)	X	X
Java Persistence API (JPA): JPA ローダー、JPA クライアント・ローダー、および JPA 時間ベース・アップデーター	X	X
メモリー・ベース除去 (MemoryPoolMXBean を使用)	X	X
インスツルメンテーション・エージェント: <ul style="list-style-type: none"> <li>• wxssizeagent.jar: 使用されるバイト・マップ・メトリックの精度を上げます。</li> <li>• ogagent.jar: フィールド・アクセス・エンティティのパフォーマンスを向上させます。</li> </ul>	X	X
モニター用 Web コンソール		X

## Java EE の考慮事項

WebSphere eXtreme Scale を Java Platform, Enterprise Edition 環境に統合する準備をするときは、バージョン、構成オプション、要件と制約、およびアプリケーションのデプロイメントと管理などを考慮します。

### Java EE 環境での eXtreme Scale アプリケーションの実行

Java EE アプリケーションは、eXtreme Scale のリモート・アプリケーションに接続できます。さらに、WebSphere Application Server 環境は、アプリケーションがアプリケーション・サーバーで開始するときに eXtreme Scale サーバーの始動をサポートします。

ObjectGrid インスタンスの作成に XML ファイルを使用する場合、かつ XML ファイルがエンタープライズ・アーカイブ (EAR) ファイルのモジュール内にある場合、`getClass().getClassLoader().getResource("META-INF/objGrid.xml")` メソッドを使用してファイルにアクセスし、ObjectGrid インスタンスの作成に使用する URL オブジェクトを取得してください。メソッド呼び出しで使用している XML ファイルの名前に置き換えます。

アプリケーションの開始 Bean を使用して、アプリケーションが起動する際に ObjectGrid インスタンスをブートストラップし、アプリケーションが停止する際にそのインスタンスを破棄することができます。開始 Bean は、`com.ibm.websphere.startupservice.AppStartUpHome` リモート・ロケーションと `com.ibm.websphere.startupservice.AppStartUp` リモート・インターフェースを持つ Stateless Session Bean です。リモート・インターフェースには `start` メソッドと `stop` メソッドという 2 つのメソッドがあります。`start` メソッドを使用してインスタンスをブートストラップし、`stop` メソッドを使用してインスタンスを破棄します。アプリケーションは `ObjectGridManager.getObjectGrid` メソッドを使用して、インスタンスへの参照を保持します。詳しくは、「プログラミング・ガイド」の `ObjectGridManager` を使用した `ObjectGrid` へのアクセスに関する情報を参照してください。

## クラス・ローダーの使用

別のクラス・ローダーを使用するアプリケーション・モジュールが Java EE アプリケーションの単一 `ObjectGrid` インスタンスを共有する場合、eXtreme Scale に保管されるオブジェクトと製品のプラグインがアプリケーションの共通ローダーにあることを確認してください。

## サーブレット内の ObjectGrid インスタンスのライフサイクルの管理

サーブレットで `ObjectGrid` インスタンスのライフサイクルを管理するためには、`init` メソッドを使用してインスタンスを作成したり、`destroy` メソッドを使用してインスタンスを除去することができます。インスタンスがキャッシュされた場合、サーブレット・コードで検索および操作を行います。詳しくは、「プログラミング・ガイド」の `ObjectGridManager` インターフェースを使用した `ObjectGrid` へのアクセスに関する情報を参照してください。

## ディレクトリー規則

`wxs_install_root` や `wxs_home` など、参照が必要な特別のディレクトリーに対して、資料全体で、次のディレクトリー規則が使用されます。インストール中、およびコマンド行ツールの使用時も含めて、さまざまなシナリオで、これらのディレクトリーにアクセスします。

### `wxs_install_root`

`wxs_install_root` ディレクトリーは、WebSphere eXtreme Scale 製品ファイルがインストールされているルート・ディレクトリーです。`wxs_install_root` ディレクトリーは、試用版のアーカイブが解凍されたディレクトリー、または WebSphere eXtreme Scale 製品がインストールされているディレクトリーの可能性があります。

- 試用版を解凍した場合の例:

例: `/opt/IBM/WebSphere/eXtremeScale`

- WebSphere eXtreme Scale がスタンドアロン・ディレクトリーにインストールされている場合の例:

UNIX

例: `/opt/IBM/eXtremeScale`

**Windows** 例: C:\Program Files\IBM\WebSphere\eXtremeScale

- WebSphere eXtreme Scale が WebSphere Application Server に統合されている場合の例:

例: /opt/IBM/WebSphere/AppServer

#### wxs\_home

*wxs\_home* ディレクトリーは、WebSphere eXtreme Scale 製品ライブラリー、サンプル、およびコンポーネントのルート・ディレクトリーです。このディレクトリーは、試用版を解凍した場合は、*wxs\_install\_root* ディレクトリーと同じです。スタンドアロンのインストール済み環境の場合、*wxs\_home* ディレクトリーは、*wxs\_install\_root* ディレクトリー内の ObjectGrid サブディレクトリーです。WebSphere Application Server に統合されているインストール済み環境の場合、このディレクトリーは、*wxs\_install\_root* ディレクトリー内の optionalLibraries/ObjectGrid ディレクトリーです。

- 試用版を解凍した場合の例:

例: /opt/IBM/WebSphere/eXtremeScale

- WebSphere eXtreme Scale がスタンドアロン・ディレクトリーにインストールされている場合の例:

**UNIX** 例: /opt/IBM/eXtremeScale/ObjectGrid

**Windows** 例: *wxs\_install\_root*\ObjectGrid

- WebSphere eXtreme Scale が WebSphere Application Server に統合されている場合の例:

例: /opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid

#### was\_root

*was\_root* ディレクトリーは、WebSphere Application Server インストール済み環境のルート・ディレクトリーです。

例: /opt/IBM/WebSphere/AppServer

#### restservice\_home

*restservice\_home* ディレクトリーは、WebSphere eXtreme Scale REST データ・サービスのライブラリーおよびサンプルが配置されるディレクトリーです。このディレクトリーは *restservice* という名前で、*wxs\_home* ディレクトリー内のサブディレクトリーです。

- スタンドアロン・デプロイメントの場合の例:

例: /opt/IBM/WebSphere/eXtremeScale/ObjectGrid/restservice

例: *wxs\_home*\restservice

- WebSphere Application Server 統合デプロイメントの場合の例:

例: /opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid/  
restservice

### tomcat\_root

*tomcat\_root* は、Apache Tomcat インストール済み環境のルート・ディレクトリーです。

例: /opt/tomcat5.5

### wasce\_root

*wasce\_root* は、WebSphere Application Server Community Edition インストール済み環境のルート・ディレクトリーです。

例: /opt/IBM/WebSphere/AppServerCE

### java\_home

*java\_home* は、Java Runtime Environment (JRE) インストール済み環境のルート・ディレクトリーです。

**UNIX** 例: /opt/IBM/WebSphere/eXtremeScale/java

**Windows** 例: *wxs\_install\_root*¥java

### samples\_home

*samples\_home* は、チュートリアルに使用するサンプル・ファイルを解凍したディレクトリーです。

**UNIX** 例: *wxs\_home*/samples

**Windows** 例: *wxs\_home*¥samples

### dvd\_root

*dvd\_root* ディレクトリーは、製品が含まれた DVD のルート・ディレクトリーです。

例: dvd\_root/docs/

### equinox\_root

*equinox\_root* ディレクトリーは、Eclipse Equinox OSGi フレームワークのインストール済み環境のルート・ディレクトリーです。

例: /opt/equinox

### user\_home

*user\_home* ディレクトリーは、ユーザー・ファイル (セキュリティー・プロファイルなど) が保管されている場所です。

**Windows** c:¥Documents and Settings¥*user\_name*

**UNIX** /home/*user\_name*

## WebSphere Application Server と統合された WebSphere eXtreme Scale 用のランタイム・ファイル

Java アーカイブ (JAR) ファイルは、インストールに含まれます。ここには、含まれる JAR ファイルとそのインストール先が示されます。

表 5. WebSphere eXtreme Scale用のランタイム・ファイル： 次の表に、このインストールに含まれる Java アーカイブ (JAR) ファイルをリストします。インストールの場所は、インストール時に選択した `wxs_home` ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
wxdynacache.jar	クライアントおよびサーバー	lib	wxdynacache.jar ファイルには、動的キャッシュ・プロバイダーと一緒に使用するために必要なクラスが含まれています。
wsobjectgrid.jar	ローカルおよびクライアント	lib	wsobjectgrid.jar には、eXtreme Scale のローカル、クライアント、およびサーバー・ランタイムが含まれています。
ogagent.jar	ローカル、クライアント、およびサーバー	lib	ogagent.jar ファイルには、EntityManager API と一緒に使用される Java インストゥルメンテーション・エージェントの実行に必要なランタイム・クラスが含まれています。
ogsip.jar	サーバー	lib	ogsip.jar ファイルには、WebSphere Application Server バージョン 6.1.x との互換性がある、eXtreme Scale Session Initiation Protocol (SIP) セッション管理ランタイムが含まれています。
sessionobjectgrid.jar	クライアントおよびサーバー	lib	sessionobjectgrid.jar ファイルには、eXtreme Scale HTTP セッション管理ランタイムが含まれています。
sessionobjectgridsip.jar	サーバー	lib	sessionobjectgridsip.jar ファイルには、WebSphere Application Server バージョン 7.x との互換性がある、eXtreme Scale SIP セッション管理ランタイムが含まれています。
wsogclient.jar	ローカルおよびクライアント	lib	wsogclient.jar ファイルは、WebSphere Application Server バージョン 6.0.2 以降を含む環境を使用した場合にインストールされます。このファイルには、ローカル・ランタイム環境およびクライアント・ランタイム環境のみが含まれています。
wssizeagent.jar	ローカル、クライアント、およびサーバー	lib	wssizeagent.jar ファイルは、Java ランタイム環境 (JRE) バージョン 1.5 以上の使用時に、より正確なキャッシュ・エントリー・サイジング情報を提供するために使用されます。
oghibernate-cache.jar	クライアントおよびサーバー	optionalLibraries/ObjectGrid	oghibernate-cache.jar ファイルには、JBoss Hibernate 用の eXtreme Scale レベル 2 キャッシュ・プラグインが含まれています。
ogspring.jar	ローカル、クライアント、およびサーバー	optionalLibraries/ObjectGrid	ogspring.jar ファイルには、SpringSource Spring フレームワーク統合用のサポート・クラスが含まれています。
xsadmin.jar	ユーティリティ	optionalLibraries/ObjectGrid	xsadmin.jar ファイルには、eXtreme Scale 管理サンプル・ユーティリティが含まれています。
ibmcfw.jar ibmorb.jar ibmorbapi.jar	クライアントおよびサーバー	optionalLibraries/ObjectGrid/endorsed	このファイル・セットには、Java SE プロセスでアプリケーションを実行するために使用されるオブジェクト・リクエスト・ブローカー (ORB) ランタイムが含まれています。
wshyperic.jar	ユーティリティ	optionalLibraries/ObjectGrid/hyperic/lib	SpringSource Hyperic モニター・エージェントの WebSphere eXtreme Scale サーバー検出プラグイン。
restservice.ear	クライアント	optionalLibraries/ObjectGrid/restservice/lib	restservice.ear ファイルには、WebSphere Application Server 環境用の eXtreme Scale REST データ・サービス・アプリケーション・エンタープライズ・アーカイブが含まれています。
restservice.war	クライアント	optionalLibraries/ObjectGrid/restservice/lib	restservice.war ファイルには、別のベンダーから取得されたアプリケーション・サーバー用の eXtreme Scale REST データ・サービス Web アーカイブが含まれています。
splicerlistener.jar	ユーティリティ	optionalLibraries/ObjectGrid/session/lib	splicerlistener.jar ファイルには、eXtreme Scale HTTP セッション・マネージャー・フィルター用のスプライサー・ユーティリティが含まれています。



表 5. WebSphere eXtreme Scale用のランタイム・ファイル (続き): 次の表に、このインストールに含まれる Java アーカイブ (JAR) ファイルをリストします。インストールの場所は、インストール時に選択した `wxs_home` ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
<code>splicer.jar</code>	ユーティリティ	<code>optionalLibraries/ObjectGrid/legacy/session/lib</code>	<code>splicer.jar</code> には、eXtreme Scale HTTP セッション・マネージャー・フィルター用のバージョン 7.0 スプライサー・ユーティリティが含まれています。
<code>wxsra.rar</code>	ユーティリティ	<code>optionalLibraries/ObjectGrid/wxsra.rar</code>	<code>wxsra.rar</code> には、接続ファクトリーを使用してグリッドに接続するための eXtreme Scale リソース・アダプターが含まれています。

表 6. WebSphere eXtreme Scale クライアント用のランタイム・ファイル: 次の表に、このインストールに含まれる Java アーカイブ (JAR) ファイルをリストします。インストールの場所は、インストール時に選択した `wxs_home` ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
<code>wxsdynacache.jar</code>	クライアントおよびサーバー	<code>lib</code>	<code>wxsdynacache.jar</code> ファイルには、動的キャッシュ・プロバイダーと一緒に使用するために必要なクラスが含まれています。
<code>ogagent.jar</code>	ローカル、クライアント、およびサーバー	<code>lib</code>	<code>ogagent.jar</code> ファイルには、EntityManager API と一緒に使用される Java インストルメンテーション・エージェントの実行に必要なランタイム・クラスが含まれています。
<code>ogsip.jar</code>	サーバー	<code>lib</code>	<code>ogsip.jar</code> ファイルには、WebSphere Application Server バージョン 6.1.x との互換性がある、eXtreme Scale Session Initiation Protocol (SIP) セッション管理ランタイムが含まれています。
<code>sessionobjectgrid.jar</code>	クライアントおよびサーバー	<code>lib</code>	<code>sessionobjectgrid.jar</code> ファイルには、eXtreme Scale HTTP セッション管理ランタイムが含まれています。
<code>sessionobjectgridsip.jar</code>	サーバー	<code>lib</code>	<code>sessionobjectgridsip.jar</code> ファイルには、WebSphere Application Server バージョン 7.x との互換性がある、eXtreme Scale SIP セッション管理ランタイムが含まれています。
<code>wsogclient.jar</code>	ローカルおよびクライアント	<code>lib</code>	<code>wsogclient.jar</code> ファイルは、WebSphere Application Server バージョン 6.0.2 以降を含む環境を使用した場合にインストールされます。このファイルには、ローカル・ランタイム環境およびクライアント・ランタイム環境のみが含まれています。
<code>wxssizeagent.jar</code>	ローカル、クライアント、およびサーバー	<code>lib</code>	<code>wxssizeagent.jar</code> ファイルは、Java ランタイム環境 (JRE) バージョン 1.5 以上の使用時に、より正確なキャッシュ・エントリー・サイジング情報を提供するために使用されます。
<code>oghibernate-cache.jar</code>	クライアントおよびサーバー	<code>optionalLibraries/ObjectGrid</code>	<code>oghibernate-cache.jar</code> ファイルには、JBoss Hibernate 用の eXtreme Scale レベル 2 キャッシュ・プラグインが含まれています。
<code>ogspring.jar</code>	ローカル、クライアント、およびサーバー	<code>optionalLibraries/ObjectGrid</code>	<code>ogspring.jar</code> ファイルには、SpringSource Spring フレームワーク統合用のサポート・クラスが含まれています。
<code>xsadmin.jar</code>	ユーティリティ	<code>optionalLibraries/ObjectGrid</code>	<code>xsadmin.jar</code> ファイルには、eXtreme Scale 管理サンプル・ユーティリティが含まれています。
<code>ibmcfw.jar</code> <code>ibmorb.jar</code> <code>ibmorbapi.jar</code>	クライアントおよびサーバー	<code>optionalLibraries/ObjectGrid/endorsed</code>	このファイル・セットには、Java SE プロセスでアプリケーションを実行するために使用されるオブジェクト・リクエスト・ブローカー (ORB) ランタイムが含まれています。
<code>wxshyperic.jar</code>	ユーティリティ	<code>optionalLibraries/ObjectGrid/hyperic/lib</code>	SpringSource Hyperic モニター・エージェントの WebSphere eXtreme Scale サーバー検出プラグイン。

表 6. WebSphere eXtreme Scale クライアント用のランタイム・ファイル (続き): 次の表に、このインストールに含まれる Java アーカイブ (JAR) ファイルをリストします。インストールの場所は、インストール時に選択した `wxs_home` ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
restservice.ear	クライアント	optionalLibraries/ObjectGrid/ restservice/lib	restservice.ear ファイルには、WebSphere Application Server 環境用の eXtreme Scale REST データ・サービス・アプリケーション・エンタープライズ・アーカイブが含まれています。
restservice.war	クライアント	optionalLibraries/ObjectGrid/ restservice/lib	restservice.war ファイルには、別のベンダーから取得されたアプリケーション・サーバー用の eXtreme Scale REST データ・サービス Web アーカイブが含まれています。
splicerlistener.jar	ユーティリティ	optionalLibraries/ObjectGrid/ session/lib	splicerlistener.jar ファイルには、eXtreme Scale HTTP セッション・マネージャー・フィルター用のスプライサー・ユーティリティが含まれています。
splicer.jar	ユーティリティ	optionalLibraries/ObjectGrid/ legacy/session/lib	splicer.jar には、eXtreme Scale HTTP セッション・マネージャー・フィルター用のバージョン 7.0 スプライサー・ユーティリティが含まれています。

## WebSphere eXtreme Scale スタンドアロン・インストール用のランタイム・ファイル

Java アーカイブ (JAR) ファイルは、インストールに含まれます。ここには、含まれる JAR ファイルとそのインストール先が示されます。

表 7. WebSphere eXtreme Scale フルインストールのランタイム・ファイル: WebSphere eXtreme Scale は、ObjectGrid プロセスおよび関連 API に依存しています。次の表に、このインストールに含まれる JAR ファイルをリストします。インストールの場所は、インストール時に選択した `wxs_home` ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
wxsdynacache.jar	クライアントおよびサーバー	dynacache/lib	wxsdynacache.jar ファイルには、動的キャッシュ・プロバイダーと一緒に使用するために必要なクラスが含まれています。提供されているスクリプトを使用すると、このファイルは自動的にサーバー・ランタイム環境に組み込まれます。
wxshyperic.jar	ユーティリティ	hyperic/lib	SpringSource Hyperic モニター・エージェントの WebSphere eXtreme Scale サーバー検出プラグイン。
objectgrid.jar	ローカル、クライアント、およびサーバー	lib	objectgrid.jar ファイルは、Java SE 5.0 以降のサーバー・ランタイム環境によって使用される OSGi バンドルです。提供されているスクリプトを使用すると、このファイルは自動的にサーバー・ランタイム環境に組み込まれます。
ogagent.jar	ローカル、クライアント、およびサーバー	lib	ogagent.jar ファイルには、EntityManager API と一緒に使用される Java インストゥルメンテーション・エージェントの実行に必要なランタイム・クラスが含まれています。
ogclient.jar	ローカルおよびクライアント	lib	ogclient.jar ファイルは、ローカル・ランタイム環境とクライアント・ランタイム環境のみが含まれる OSGi バンドルです。このファイルは、Java SE 5.0 以降で使用できます。
ogspring.jar	ローカル、クライアント、およびサーバー	lib	ogspring.jar ファイルには、SpringSource Spring フレームワーク統合用のサポート・クラスが含まれています。
wsogclient.jar	ローカルおよびクライアント	lib	wsogclient.jar ファイルは、WebSphere Application Server バージョン 6.0.2 以降を含む環境を使用した場合にインストールされます。このファイルには、ローカル・ランタイム環境およびクライアント・ランタイム環境のみが含まれています。
wxssizeagent.jar	ローカル、クライアント、およびサーバー	lib	wxssizeagent.jar ファイルは、Java ランタイム環境 (JRE) バージョン 1.5 以上の使用時に、より正確なキャッシュ・エントリー・サイズング情報を提供するために使用されます。

表7. *WebSphere eXtreme Scale* フルインストールのランタイム・ファイル (続き): *WebSphere eXtreme Scale* は、ObjectGrid プロセスおよび関連 API に依存しています。次の表に、このインストールに含まれる JAR ファイルをリストします。インストールの場所は、インストール時に選択した *wxs\_home* ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
ibmcfw.jar ibmorb.jar ibmorbapi.jar	クライアントおよびサーバー	lib/endorsed	このファイル・セットには、Java SE プロセスでアプリケーションを実行するために使用されるオブジェクト・リクエスト・ブローカー (ORB) ランタイムが含まれています。
restservice.ear	クライアント	restservice/lib	restservice.ear ファイルには、WebSphere Application Server 環境用の eXtreme Scale REST データ・サービス・アプリケーション・エンタープライズ・アーカイブが含まれています。
restservice.war	クライアント	restservice/lib	restservice.war ファイルには、別のベンダーから取得されたアプリケーション・サーバー用の eXtreme Scale REST データ・サービス Web アーカイブが含まれています。
xsadmin.jar	ユーティリティ	samples	xsadmin.jar ファイルには、eXtreme Scale 管理サンプル・ユーティリティが含まれています。
sessionobjectgrid.jar	クライアントおよびサーバー	session/lib	sessionobjectgrid.jar ファイルには、eXtreme Scale HTTP セッション管理ランタイムが含まれています。
splicerlistener.jar	ユーティリティ	session/lib	splicerlistener.jar ファイルには、eXtreme Scale バージョン 7.1 以降の HTTP セッション・リスナー用のスプライサー・ユーティリティが含まれています。
xsgbean.jar	サーバー	wasce/lib	xsgbean.jar ファイルには、eXtreme Scale サーバーを WebSphere Application Server Community Edition アプリケーション・サーバーに組み込むための GBean が含まれています。
splicer.jar	ユーティリティ	legacy/session/lib	WebSphere eXtreme Scale バージョン 7.0 HTTP セッション・マネージャー・フィルター用のスプライサー・ユーティリティ
wxsra.rar	クライアントおよびサーバー	session/lib	wxsra.rar には、接続ファクトリーを使用してグリッドに接続するための eXtreme Scale リソース・アダプターが含まれています。

表8. *WebSphere eXtreme Scale* クライアント用のランタイム・ファイル: *WebSphere eXtreme Scale* クライアントは、ObjectGrid プロセスおよび関連 API に依存しています。次の表に、このインストールに含まれる JAR ファイルをリストします。インストールの場所は、インストール時に選択した *wxs\_home* ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
wxsdynacache.jar	クライアントおよびサーバー	dynacache/lib	wxsdynacache.jar ファイルには、動的キャッシュ・プロバイダーと一緒に使用するために必要なクラスが含まれています。提供されているスクリプトを使用すると、このファイルは自動的にサーバー・ランタイム環境に組み込まれます。
wxshyperic.jar	ユーティリティ	hyperic/lib	SpringSource Hyperic モニター・エージェントの WebSphere eXtreme Scale サーバー検出プラグイン。
ogagent.jar	ローカル、クライアント、およびサーバー	lib	ogagent.jar ファイルには、EntityManager API と一緒に使用される Java インストルメンテーション・エージェントの実行に必要なランタイム・クラスが含まれています。
ogclient.jar	ローカルおよびクライアント	lib	ogclient.jar ファイルは、ローカル・ランタイム環境とクライアント・ランタイム環境のみが含まれる OSGi パンドルです。このファイルは、Java SE 5 以降で使用できます。
ogspring.jar	ローカル、クライアント、およびサーバー	lib	ogspring.jar ファイルには、SpringSource Spring フレームワーク統合用のサポート・クラスが含まれています。
wsogclient.jar	ローカルおよびクライアント	lib	wsogclient.jar ファイルは、WebSphere Application Server バージョン 6.0.2 以降を含む環境を使用した場合にインストールされます。このファイルには、ローカル・ランタイム環境およびクライアント・ランタイム環境のみが含まれています。

表 8. WebSphere eXtreme Scale クライアント用のランタイム・ファイル (続き): WebSphere eXtreme Scale クライアントは、ObjectGrid プロセスおよび関連 API に依存しています。次の表に、このインストールに含まれる JAR ファイルをリストします。インストールの場所は、インストール時に選択した `wxs_home` ディレクトリーからの相対的な場所です。

ファイル名	環境	インストールの場所	説明
wxssizeagent.jar	ローカル、クライアント、およびサーバー	lib	wxssizeagent.jar ファイルは、Java ランタイム環境 (JRE) バージョン 1.5 以上の使用時に、より正確なキャッシュ・エントリー・サイジング情報を提供するために使用されます。
ibmcfw.jar ibmorb.jar ibmorbapi.jar	クライアントおよびサーバー	lib/endorsed	このファイル・セットには、Java SE プロセスでアプリケーションを実行するために使用されるオブジェクト・リクエスト・ブローカー (ORB) ランタイムが含まれています。
restservice.ear	クライアント	restservice/lib	restservice.ear ファイルには、WebSphere Application Server 環境用の eXtreme Scale REST データ・サービス・アプリケーション・エンタープライズ・アーカイブが含まれています。
restservice.war	クライアント	restservice/lib	restservice.war ファイルには、別のベンダーから取得されたアプリケーション・サーバー用の eXtreme Scale REST データ・サービス Web アーカイブが含まれています。
xsadmin.jar	ユーティリティー	samples	xsadmin.jar ファイルには、eXtreme Scale 管理サンプル・ユーティリティーが含まれています。
sessionobjectgrid.jar	クライアントおよびサーバー	session/lib	sessionobjectgrid.jar ファイルには、eXtreme Scale HTTP セッション管理ランタイムが含まれています。
splicerlistener.jar	ユーティリティー	session/lib	splicerlistener.jar ファイルには、eXtreme Scale バージョン 7.1 以降の HTTP セッション・リスナー用のスプライサー・ユーティリティーが含まれています。
splicer.jar	ユーティリティー	legacy/session/lib	WebSphere eXtreme Scale バージョン 7.0 HTTP セッション・マネージャー・フィルター用のスプライサー・ユーティリティー

## IBM Installation Manager および WebSphere eXtreme Scale 製品オフリングのインストール

WebSphere eXtreme Scale バージョン 8.5 製品オフリングは、製品リポジトリーで使用可能です。これらのリポジトリーにアクセスするには、まず IBM Installation Manager をインストールする必要があります。

Installation Manager は、製品メディア内のファイル、パスポート・アドバンテージのサイトから入手したファイル、もしくは IBM Installation Manager ダウンロード Web サイトから入手したファイルのいずれかを使用してインストールできます。ファイルはインストール・イメージが収められた圧縮ファイルです。

Installation Manager を使用すると、必要な製品リポジトリーにアクセスできます。WebSphere eXtreme Scale バージョン 8.5 製品オフリングをインストールするには、これらのリポジトリーにアクセスする必要があります。

製品リポジトリーにアクセスするには、2 つのオプションがあります。

### オプション 1: 物理メディア上の製品リポジトリーにアクセスして、ローカル・インストールを行う

1. システムに Installation Manager をインストールします。
2. Installation Manager を使用して、メディア上の製品リポジトリーから製品オフリングをインストールします。

## オプション 2: パスポート・アドバンテージから製品リポジトリをダウンロードして、ローカル・インストールを行う

1. パスポート・アドバンテージのサイトからリポジトリをダウンロードします。

注: IBM パスポート・アドバンテージ・オンライン Web サイトからダウンロード可能な IBM WebSphere eXtreme Scale 8.5 のインストール・イメージのリストとその他の情報については、<http://www-01.ibm.com/support/docview.wss?uid=swg24032428> を参照してください。

2. システムに Installation Manager をインストールします。

3. Installation Manager を使用して、ダウンロードした製品リポジトリから製品をインストールします。

## GUI の使用による IBM Installation Manager のインストール

WebSphere eXtreme Scale 製品オフリングをインストールできるように、必要な製品リポジトリにアクセスするには、IBM Installation Manager をインストールする必要があります。Installation Manager はウィザード・コンソールを使用してインストールできます。

### 始める前に

IBM Installation Manager をインストールし、必要なリポジトリにアクセス可能でなければなりません。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オフリングのインストール』を参照してください。

### 手順

1. Installation Manager のインストール・ファイルが含まれているロケーションに移動して、以下のコマンドのいずれかを実行します。

#### 管理インストール:

- **Windows** `install.exe`
- **UNIX** **Linux** `./install`

#### 非管理インストール:

- **Windows** `userinst.exe`
- **UNIX** **Linux** `./userinst`

#### グループ・モードでのインストール:

- **UNIX** **Linux** `./groupinst`

#### グループ・モードに関する注記:

- グループ・モードを使用すると、複数のユーザーが IBM Installation Manager の 1 つのインスタンスを使用してソフトウェア・パッケージを管理できるようになります。

これは、IBM Installation Manager の単一インスタンスを 2 人のユーザーが同時に使用できるという意味ではありません。

- **Windows** グループ・モードは、Windows オペレーティング・システムでは使用できません。
  - グループ・モードを使用せずに Installation Manager をインストールする場合、この Installation Manager を使用して今後インストールする製品はグループ・モードを使用して管理できません。
  - インストール・ロケーションは、必ず現在のユーザーのホーム・ディレクトリー内のデフォルトのロケーションから、グループ内のすべてのユーザーがアクセス可能なロケーションに変更してください。
  - グループ・モードでインストールする前に、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップの説明のとおり、グループ、権限、および環境変数を設定してください。
  - グループ・モードの使用方法については、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップを参照してください。
2. Installation Manager のパッケージが選択されていることを確認して、「次へ」をクリックします。
  3. 使用条件の条項に同意し、「次へ」をクリックします。
  4. 「次へ」をクリックします。
  5. 要約情報を検討して、「インストール」をクリックします。正常にインストールされた場合は、プログラムにより、インストールが正常に行われたことを示すメッセージが表示されます。インストールが正常に行われなかった場合は、「ログ・ファイルの表示」をクリックして、問題のトラブルシューティングを行います。
  6. 製品リポジトリーを Installation Manager 設定に追加します。
    - a. Installation Manager を開始します。
    - b. トップ・メニューで、「ファイル」>「設定」をクリックします。
    - c. 「リポジトリー」を選択します。
    - d. 「リポジトリーの追加」をクリックします。
    - e. リポジトリー・ファイルを含むロケーション内の repository.config ファイルのパスを入力します。以下に例を示します。
      - **Windows** C:%repositories%product\_name%local-repositories
      - **UNIX** **Linux** /var/repositories/product\_name/local-repositories
    - f. 「OK」をクリックします。
  7. 「リポジトリー」ウィンドウにリストされているロケーションのうち使用しないものをクリアします。
  8. 「適用」をクリックします。
  9. 「OK」をクリックします。
  10. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。



## 次のタスク

Installation Manager のインストールとリポジトリのセットアップが正常に完了したら、引き続き製品オフリング用の任意の WebSphere eXtreme Scale スタンドアロン、または WebSphere eXtreme Scale for WebSphere Application Server をインストールできます。詳しくは、『GUI の使用による製品 のインストール』を参照してください。

## GUI の使用による製品 のインストール

ウィザード・コンソールから Installation Manager を使用し、WebSphere eXtreme Scale 製品オフリングをインストールします。

### 始める前に

Installation Manager に必要な製品ファイルをインストールし、必要なりポジトリにアクセス可能でなければなりません。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オフリングのインストール』を参照してください。

### 手順

1. Installation Manager を開始します。

**ヒント:**   `./IBMIM` コマンドを使用して、Installation Manager をグループ・モードで開始することができます。

- グループ・モードを使用すると、複数のユーザーが IBM Installation Manager の 1 つのインスタンスを使用してソフトウェア・パッケージを管理できるようになります。
  - グループ・モードの使用方法について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップを参照してください。
2. 「インストール」をクリックします。

**注:** 認証のプロンプトが出されたら、プログラムの Web サイトで登録した IBM ID とパスワードを使用してください。

Installation Manager は、定義済みリポジトリ内にある使用可能なパッケージを検索します。

3. 以下のいずれかの製品オフリングと適切なバージョンを選択します。
  - スタンドアロン環境での WebSphere eXtreme Scale
  - スタンドアロン環境での WebSphere eXtreme Scale クライアント
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7

- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8

ご使用のシステムに既に当該製品がインストールされている場合は、その製品が既にインストールされていることを示すメッセージが表示されます。別のロケーションに製品のインストール済み環境を別途作成するには、「**続行**」をクリックします。

**ヒント:** 「インストール・マネージャー・リポジトリ設定」 ページで「**インストール中および更新中にサービス・リポジトリの検索**」 オプションが選択されていて、インターネットに接続している場合、「**他のバージョンと拡張機能の確認**」 をクリックできます。これにより、選択したパッケージのデフォルト更新リポジトリで更新を検索することができます。この場合、「インストール・マネージャー・リポジトリ設定」 ページに特定のサービス・リポジトリ URL を追加する必要はありません。

- a. インストールするフィックスを選択します。

デフォルトでは、任意の推奨フィックスが選択されます。

推奨フィックスがある場合、推奨フィックスのみを表示し、非推奨フィックスを非表示にするオプションを選択できます。

- b. 「**次へ**」 をクリックします。

**注:** Installation Manager は、リポジトリに接続する際に、最新レベルの Installation Manager への更新を要求するプロンプトを出します。プロンプトが出されたら、新しいバージョンに更新してから続行します。自動更新については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

4. 使用条件の条項に同意し、「**次へ**」 をクリックします。
5. 製品のインストール・ルート・ディレクトリを指定します。

パネルには共有リソース・ディレクトリおよびディスク・スペース情報も指定されます。

**注:** 初めて Installation Manager を使用してパッケージをインストールする場合、共有リソース・ディレクトリを指定します。共有リソース・ディレクトリは、1 つ以上のパッケージ・グループで使用できるインストール成果物が置かれるディレクトリです。このインストールには、最も大きいドライブを使用してください。すべてのパッケージをアンインストールし終わるまで、このディレクトリ・ロケーションは変更できません。

#### **制約事項:**

- デフォルト・ターゲット・ロケーションを削除し、インストール・ディレクトリ・フィールドを空のままにすると、続行できなくなります。
- 宛先ディレクトリとしてシンボリック・リンクを使用しないでください。

シンボリック・リンクはサポートされていません。

- ディレクトリ名にセミコロンを使用しないでください。

ターゲット・ディレクトリーにセミコロンが含まれている場合、WebSphere eXtreme Scale は想定どおりにインストールされません。

**Windows** セミコロンは、Windows システムでは、クラスパスを構成するために使用する文字です。

- **Windows** Windows Server 2008、Windows Vista、および Windows 7 オペレーティング・システムでの最大パス長は 60 文字です。

6. 「次へ」をクリックします。
7. インストールするコンテンツの翻訳言語を選択します。

常に英語が選択されています。

8. 「次へ」をクリックします。
9. インストールするフィーチャーを選択します。

選択した製品オファリングに応じて、次のフィーチャーから選択できます。

- クライアント

WebSphere eXtreme Scale をスタンドアロン環境にインストールする場合、または WebSphere eXtreme Scale for WebSphere Application Server 製品オファリングをインストールする場合、必須フィーチャーとして使用可能です。これらの製品オファリング用にクライアントがインストールされている必要があります。

- サーバー

WebSphere eXtreme Scale をスタンドアロン環境にインストールする場合、または WebSphere eXtreme Scale for WebSphere Application Server をインストールする場合に、使用可能です。これらの製品オファリングについては、サーバーをインストールしないという選択も可能です。

- コンソール

すべての WebSphere eXtreme Scale 製品オファリングで使用可能です。モニター・コンソールのインストールを選択することができます。Web コンソールでは、現在と過去の統計をグラフにできます。このコンソールには、概要を表示するようにグラフがいくつか用意されているほか、使用可能な統計からグラフを作成できるカスタム・レポート・ページもあります。WebSphere eXtreme Scale のモニター・コンソールのグラフ機能を使用して、環境内のデータ・グリッドの全体的なパフォーマンスを表示できます。

- サンプル

すべての WebSphere eXtreme Scale 製品オファリングで使用可能です。

10. 「次へ」をクリックします。
11. 要約情報を検討して、「インストール」をクリックします。

- 正常にインストールされた場合は、プログラムにより、インストールが正常に行われたことを示すメッセージが表示されます。

注: さらに、プログラムによって、重要なポストインストール指示も指定されることがあります。

- インストールが正常に行われなかった場合は、「ログ・ファイルの表示」をクリックして、問題のトラブルシューティングを行います。
12. このインストールの完了後に開始するツールを選択します。
    - 実稼働環境に適した設定のアプリケーション・サーバー・プロファイルを新規に作成する場合は、「プロファイルを作成するためのプロファイル管理ツール」を選択します。
    - 開発環境に適した設定のアプリケーション・サーバー・プロファイルを作成する場合は、「開発環境のアプリケーション・サーバー・プロファイルを作成するためのプロファイル管理ツール (Profile Management Tool to create an application server profile for a development environment)」を選択します。

注: development 設定は、アプリケーションの更新が頻繁に行われ、システム・リソースが最小限である開発環境に適しています。実動サーバーには development 設定を使用しないでください。

    - このインストールの完了後に、新しいプロファイルを作成しない場合は、「なし」を選択します。

**制約事項:** プロファイル管理ツールを起動するオプションは、プロファイル管理ツールを含むバージョンの WebSphere Application Server がインストールされている場合にのみ使用可能です。
  13. 「終了」をクリックします。
  14. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。

## コマンド行の使用による IBM Installation Manager のインストール

WebSphere eXtreme Scale 製品オファリングをインストールできるように、必要な製品リポジトリにアクセスするには、IBM Installation Manager をインストールする必要があります。Installation Manager はコマンド行からインストールできます。

### 始める前に

Installation Manager に必要な製品ファイルをインストールし、必要なリポジトリにアクセス可能でなければなりません。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オファリングのインストール』を参照してください。

### 手順

1. Installation Manager のインストール・ファイルが含まれているロケーションに移動して、以下のコマンドのいずれかを実行します。

管理インストール:

- **Windows** `installc.exe -acceptLicense -log  
log_file_path_and_name`
- **UNIX** **Linux** `./installc -acceptLicense -log  
log_file_path_and_name`

## 非管理インストール:

- **Windows** `userinstc.exe -acceptLicense -log  
log_file_path_and_name`
- **UNIX** **Linux** `./userinstc -acceptLicense -log  
log_file_path_and_name`

## グループ・モードでのインストール:

**UNIX** **Linux** `./groupinstc -acceptLicense -dataLocation  
application_data_location -log log_file_path_and_name  
-installationDirectory Installation_Manager_home`

## グループ・モードに関する注記:

- グループ・モードを使用すると、複数のユーザーが IBM Installation Manager の 1 つのインスタンスを使用してソフトウェア・パッケージを管理できるようになります。
  - **Windows** グループ・モードは、Windows オペレーティング・システムでは使用できません。
  - グループ・モードを使用せずに Installation Manager をインストールする場合、この Installation Manager を使用して今後インストールする製品はグループ・モードを使用して管理できません。
  - インストール・ロケーションは、必ず現在のユーザーのホーム・ディレクトリー内のデフォルトのロケーションから、グループ内のすべてのユーザーがアクセス可能なロケーションに変更してください。
  - グループ・モードでインストールする前に、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップの説明のとおり、グループ、権限、および環境変数を設定してください。
  - グループ・モードの使用方法について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップを参照してください。
2. オプション: リポジトリーでユーザー名とパスワードが必要な場合、このリポジトリーにアクセスするために鍵リング・ファイルを作成します。

Installation Manager の鍵リング・ファイルの作成について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

**ヒント:** 鍵リング・ファイルを作成する際、`imutilsc` コマンドで指定された URL が見つからない場合は、リポジトリー URL ロケーションの最後に `/repository.config` を追加してください。

## 次のタスク

Installation Manager のインストールとリポジトリーのセットアップが正常に完了したら、引き続き 任意の WebSphere eXtreme Scale スタンドアロン、または WebSphere eXtreme Scale for WebSphere Application Server 製品オフアリングをイ

インストールできます。詳しくは、185 ページの『GUI の使用による製品 のインストール』を参照してください。

## コマンド行の使用による この製品 のインストール

コマンド行から Installation Manager を使用し、WebSphere eXtreme Scale 製品オファリングをインストールします。

### 始める前に

Installation Manager に必要な製品ファイルをインストールし、必要なりポジトリリーにアクセス可能でなければなりません。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オファリングのインストール』を参照してください。

### 手順

1. システムにログオンします。
2. Installation Manager をインストールしたディレクトリーの eclipse/tools サブディレクトリーに移動します。
3. 製品リポジトリリーが使用可能であることを確認してください。

Windows

```
imcl.exe listAvailablePackages -repositories source_repository
```

UNIX

Linux

```
./imcl listAvailablePackages -repositories source_repository
```

オファリングの 1 つ以上のレベルが表示されるはずですが。

4. **imcl** コマンドを使用して、この製品をインストールします。

Windows

```
imcl.exe install com.ibm.websphere.v85_offering_version,optional_feature_ID  
-repositories source_repository  
-installationDirectory installation_directory  
-sharedResourcesDirectory shared_directory  
-accessRights access_mode  
-preferences preference_key=value  
-properties property_key=value  
-keyring keyring_file -password password  
-acceptLicense
```

UNIX

Linux

```
./imcl install com.ibm.websphere.offering_version,optional_feature_ID  
-repositories source_repository  
-installationDirectory installation_directory  
-sharedResourcesDirectory shared_directory  
-accessRights access_mode  
-preferences preference_key=value  
-properties property_key=value  
-keyring keyring_file -password password  
-acceptLicense
```

ヒント:



- *offering\_ID* は、173 ページの『WebSphere eXtreme Scale product offering IDs』 にリストされるオファリング ID です。
- *offering\_version* は、アンダースコアを使用してオファリング ID にオプションで付加することができますが、これはインストール対象のオファリングの特定バージョンです (例: 8.5.0.20110503\_0200)。
  - *offering\_version* が指定されていない場合、オファリングの最新バージョンと、そのバージョンのすべてのインテリム・フィックスがインストールされます。
  - *offering\_version* が指定されている場合、オファリングの指定バージョンがインストールされ、そのバージョンのインテリム・フィックスはインストールされません。

リポジトリに対して次のコマンドを実行する際、オファリングのバージョンは、アンダースコアを使用してオファリング ID の最後に付加されている場合があります:

```
imcl listAvailablePackages -repositories source_repository
```

- また、none、recommended、または all を -installFixes 引数と共に指定し、オファリングと一緒にインストールするインテリム・フィックスを示すこともできます。
  - オファリングのバージョンが指定されていない場合、-installFixes オプションはデフォルトで all になります。
  - オファリングのバージョンが指定されている場合、-installFixes オプションはデフォルトで none になります。
- コンマで区切られたフィーチャーのリストを追加できます。以下に例を示します。

```
imcl -acceptLicense install com.ibm.websphere.WXS.v85,xs.console.feature,xs.samples.feature
```

- client WebSphere eXtreme Scale をスタンドアロン環境にインストールする場合、または WebSphere eXtreme Scale for WebSphere Application Server 製品オファリングをインストールする場合、必須フィーチャーとして使用可能です。
- server インストールする製品オファリングに応じて、サーバーのインストールを選択できます。サーバーは以下の製品オファリングで選択可能なフィーチャーです。
  - スタンドアロン環境での WebSphere eXtreme Scale
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
- console すべての製品オファリングで使用可能です。モニター・コンソールのインストールを選択することができます。Web コンソールでは、現在と過去の統計をグラフにできます。このコンソールには、概要を表示するように事前構成されたグラフがいくつか用意されているほか、使用可能な統計からグラフを作成できるカスタム・レポート・ページもあります。

WebSphere eXtreme Scale のモニター・コンソールのグラフ機能を使用して、環境内のデータ・グリッドの全体的なパフォーマンスを表示できます。

- samples すべての製品オファリングで使用可能です。サンプルのインストールを選択できます。

**注:**

- 以前に Installation Manager をインストールするモードを指定した場合、-accessRights パラメーターは必要ありません
- 後で問題が生じた場合のために、Installation Manager はパッケージの以前のバージョンをロールバックの対象として保存することができます。Installation Manager が以前のバージョンにパッケージをロールバックする際は、現行バージョンのファイルはアンインストールされて、以前のバージョンが再インストールされます。ロールバック用にファイルを保存しないことを選択する場合は、次の設定を指定することによりファイルが保存されないようにできます。

```
-preference com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts=False
```

Installation Manager 設定の設定方法については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

**ヒント:** ロールバック用にファイルを保存しないよう選択した場合でも、リポジトリからロールバック用の製品ファイルにアクセスすることは可能です。

- プログラムが重要なポストインストール指示を標準出力に書き込むことがあります。

**imcl** コマンドを使用して製品をインストールする方法については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

## 応答ファイルの使用による IBM Installation Manager のインストール

WebSphere eXtreme Scale 製品オファリングをインストールできるように、必要な製品リポジトリにアクセスするには、IBM Installation Manager をインストールする必要があります。Installation Manager は応答ファイルを使用してインストールできます。

### 始める前に

Installation Manager に必要な製品ファイルをインストールし、必要なリポジトリにアクセス可能でなければなりません。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オファリングのインストール』を参照してください。

### 手順

Installation Manager のインストール・ファイルが含まれているロケーションに移動して、以下のコマンドのいずれかを実行して Installation Manager をインストールします。

### 管理インストール:

- **Windows** `installc.exe -acceptLicense -log log_file_path_and_name`
- **UNIX** **Linux** `./installc -acceptLicense -log log_file_path_and_name`

### 非管理インストール:

- **Windows** `userinstc.exe -acceptLicense -log log_file_path_and_name`
- **UNIX** **Linux** `./userinstc -acceptLicense -log log_file_path_and_name`

### グループ・モードでのインストール:

- **UNIX** **Linux** `./groupinstc -acceptLicense -dataLocation application_data_location -log log_file_path_and_name -installationDirectory Installation_Manager_home`

### グループ・モードに関する注記:

- グループ・モードを使用すると、複数のユーザーが IBM Installation Manager の 1 つのインスタンスを使用してソフトウェア・パッケージを管理できるようになります。

グループ・モードは、IBM Installation Manager の単一インスタンスを 2 人のユーザーが同時に使用できるという意味ではありません。

- **Windows** グループ・モードは、Windows オペレーティング・システムでは使用できません。
- グループ・モードを使用せずに Installation Manager をインストールする場合、この Installation Manager を使用して今後インストールする製品はグループ・モードを使用して管理できません。
- インストール・ロケーションは、必ず現在のユーザーのホーム・ディレクトリ内のデフォルトのロケーションから、グループ内のすべてのユーザーがアクセス可能なロケーションに変更してください。
- グループ・モードでインストールする前に、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップの説明のとおり、グループ、権限、および環境変数を設定してください。
- グループ・モードの使用方法について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターのグループ・モードのロードマップを参照してください。

## 次のタスク

Installation Manager のインストールとリポジトリのセットアップが正常に完了したら、引き続き製品オフリング用の任意の WebSphere eXtreme Scale スタンドアロン、または WebSphere eXtreme Scale for WebSphere Application Server をインストールできます。詳しくは、185 ページの『GUI の使用による製品 のインストール』を参照してください。

## 応答ファイルの使用による製品 のインストール

応答ファイルで Installation Manager を使用し、WebSphere eXtreme Scale 製品オフアリングをインストールします。

### 始める前に

Installation Manager に必要な製品ファイルをインストールし、必要なりポジトリーにアクセス可能でなければなりません。詳しくは、182 ページの『IBM Installation Manager および WebSphere eXtreme Scale 製品オフアリングのインストール』を参照してください。

### このタスクについて

Installation Manager を使用して、応答ファイルを処理してさまざまな方法で製品のインストールを行うことができます。GUI を使用して応答ファイルを記録できます。

### 手順

1. コマンド行で、Installation Manager をインストールしたディレクトリーの Eclipse サブディレクトリーに移動します。
2. `-record` オプションを使用して、コマンド行から Installation Manager を開始します。

以下に例を示します。

- **Windows** 管理者または非管理者:

```
IBMIM.exe -skipInstall "C:%temp%imRegistry"  
-record C:%temp%install_response_file.xml
```

- **UNIX** **Linux** 管理者:

```
./IBMIM -skipInstall /var/temp/imRegistry  
-record /var/temp/install_response_file.xml
```

- **UNIX** **Linux** 非管理者:

```
./IBMIM -skipInstall user_home/var/temp/imRegistry  
-record user_home/var/temp/install_response_file.xml
```

**ヒント:** 新しい応答ファイルを記録する際に、`-skipInstall` パラメーターを指定することができます。このパラメーターには、以下の利点があります。

- ファイルをインストールしないので、記録の速度が速くなります。
- `-skipInstall` パラメーターを指定して一時データ・ロケーションを使用すると、Installation Manager は、記録する際に、インストール・レジストリーを指定されたデータ・ロケーションに書き込みます。`-skipInstall` パラメーターを指定せずに Installation Manager を再度開始すると、応答ファイルを使用して実際のインストール・レジストリーにインストールすることができます。

`-skipInstall` の操作は、Installation Manager が使用する実際のエージェント・データ・ロケーションで使用することはできません。この操作はサポートされていません。クリーンで書き込み可能なロケーションを使用し、今後の記録セッションの際にはこのロケーションを再利用してください。

詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

3. 適切なリポジトリを Installation Manager 設定に追加します。
  - a. トップ・メニューで、「ファイル」>「設定」をクリックします。
  - b. 「リポジトリ」を選択します。
  - c. 各リポジトリに対して以下のアクションを実行します。
    - 1) 「リポジトリの追加」をクリックします。
    - 2)
    - 3) Web ベースのリモート・リポジトリ内、またはリポジトリ・ファイルを解凍したローカル・ディレクトリ内の、`repository.config` ファイルのパスを入力します。

以下に例を示します。

- リモート・リポジトリ:

`https://downloads.mycorp.com:8080/WXS_85_repository`

- ローカル・リポジトリ:

– **Windows** C:¥repositories¥wxs85¥local-repositories

– **UNIX** **Linux** /var/repositories/wxs85/local-repositories

- 4) 「OK」をクリックします。
  - 5) 「適用」をクリックします。
  - 6) 「OK」をクリックします。
- d. 「インストール」をクリックします。

**注:** 認証のプロンプトが出されたら、プログラムの Web サイトで登録した IBM ID とパスワードを使用してください。

Installation Manager は、定義済みリポジトリ内にある使用可能なパッケージを検索します。

4. 以下のいずれかの製品オフラインと適切なバージョンを選択します。
  - スタンドアロン環境での WebSphere eXtreme Scale
  - スタンドアロン環境での WebSphere eXtreme Scale クライアント
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8

ご使用のシステムに既に当該製品がインストールされている場合は、その製品が既にインストールされていることを示すメッセージが表示されます。別のロケーションに製品のインストール済み環境を別途作成するには、「**続行**」をクリックします。

**ヒント:** 「インストール・マネージャー・リポジトリ設定」ページで「**インストール中および更新中にサービス・リポジトリの検索**」オプションが選択されていて、インターネットに接続している場合、「**他のバージョンと拡張機能の確認**」をクリックできます。これにより、選択したパッケージのデフォルト更新リポジトリで更新を検索することができます。この場合、「インストール・マネージャー・リポジトリ設定」ページに特定のサービス・リポジトリ URL を追加する必要はありません。

5. インストールするフィックスを選択します。

デフォルトでは、任意の推奨フィックスが選択されます。

推奨フィックスがある場合、推奨フィックスのみを表示し、非推奨フィックスを非表示にするオプションを選択できます。

6. 「**次へ**」をクリックします。
7. 使用条件の条項に同意し、「**次へ**」をクリックします。
8. 製品のインストール・ルート・ディレクトリーを指定します。

パネルには共有リソース・ディレクトリーおよびディスク・スペース情報も指定されます。

**注:** 初めて Installation Manager を使用してパッケージをインストールする場合、共有リソース・ディレクトリーを指定します。共有リソース・ディレクトリーは、1 つ以上のパッケージ・グループで使用できるインストール成果物が置かれるディレクトリーです。このインストールには、最も大きいドライブを使用してください。すべてのパッケージをアンインストールし終わるまで、このディレクトリー・ロケーションは変更できません。

#### 制約事項:

- デフォルト・ターゲット・ロケーションを削除し、インストール・ディレクトリー・フィールドを空のままにすると、続行できなくなります。
- 宛先ディレクトリーとしてシンボリック・リンクを使用しないでください。

シンボリック・リンクはサポートされていません。

- **Windows** Windows Server 2008、Windows Vista、および Windows 7 オペレーティング・システムでの最大パス長は 60 文字です。
9. 「**次へ**」をクリックします。
  10. インストールするコンテンツの翻訳言語を選択します。

常に英語が選択されています。

11. 「**次へ**」をクリックします。
12. インストールするフィーチャーを選択します。
  - クライアント



WebSphere eXtreme Scale をスタンドアロン環境にインストールする場合、または WebSphere eXtreme Scale for WebSphere Application Server 製品オフアリングをインストールする場合、必須フィーチャーとして使用可能です。これらの製品オフアリング用にクライアントがインストールされている必要があります。

- サーバー

WebSphere eXtreme Scale をスタンドアロン環境にインストールする場合、または WebSphere eXtreme Scale for WebSphere Application Server をインストールする場合に、使用可能です。これらの製品オフアリングについては、サーバーをインストールしないという選択も可能です。

- コンソール

すべての WebSphere eXtreme Scale 製品オフアリングで使用可能です。モニター・コンソールのインストールを選択することができます。Web コンソールでは、現在と過去の統計をグラフにできます。このコンソールには、概要を表示するようにグラフがいくつか用意されているほか、使用可能な統計からグラフを作成できるカスタム・レポート・ページもあります。WebSphere eXtreme Scale のモニター・コンソールのグラフ機能を使用して、環境内のデータ・グリッドの全体的なパフォーマンスを表示できます。

- サンプル

すべての WebSphere eXtreme Scale 製品オフアリングで使用可能です。

13. 「次へ」をクリックします。

14. 要約情報を検討して、「インストール」をクリックします。

- 正常にインストールされた場合は、プログラムにより、インストールが正常に行われたことを示すメッセージが表示されます。

注: さらに、プログラムによって、重要なポストインストール指示も指定されることがあります。

- インストールが正常に行われなかった場合は、「ログ・ファイルの表示」をクリックして、問題のトラブルシューティングを行います。

15. 「終了」をクリックします。

16. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。

#### 鍵リングの作成:

WebSphere eXtreme Scale 製品オフアリングをインストールするための応答ファイルを、Installation Manager を使用して記録した後、鍵リング・ファイルの作成を選択することができます。認証を必要とするリモート・リポジトリを使用している場合、インストールのための鍵リングを作成できます。

#### 始める前に

応答ファイルを記録する必要があります。詳しくは、194 ページの『応答ファイルの使用による製品 のインストール』を参照してください。

## 手順

1. コマンド行で、Installation Manager をインストールしたディレクトリーの Eclipse サブディレクトリーに移動します。
2. -record オプションを使用して、コマンド行から Installation Manager を開始します。

以下に例を示します。

- **Windows** 管理者または非管理者:

```
IBMIM.exe -skipInstall "C:%temp%imRegistry"  
-keyring C:%IM%im.keyring  
-record C:%temp%keyring_response_file.xml
```

- **UNIX** **Linux** 管理者:

```
./IBMIM -skipInstall /var/temp/imRegistry  
-keyring /var/IM/im.keyring  
-record /var/temp/keyring_response_file.xml
```

- **UNIX** **Linux** 非管理者:

```
./IBMIM -skipInstall user_home/var/temp/imRegistry  
-keyring user_home/var/IM/im.keyring  
-record user_home/var/temp/keyring_response_file.xml
```

3. 認証済みリモート・リポジトリを使用するための資格情報を要求するウィンドウが開いたら、正しい資格情報を入力して、その情報を保存します。
4. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。

詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

---

## クライアントおよびサーバーの Eclipse Gemini を持つ Eclipse Equinox OSGi フレームワークのインストール

OSGi フレームワークに WebSphere eXtreme Scale をデプロイするには、Eclipse Equinox 環境をセットアップする必要があります。

### このタスクについて

このタスクを実行するには、Blueprint フレームワークをダウンロードしてインストールする必要があります。そうすれば、後で、JavaBeans を構成し、それをサービスとして公開することができます。サービスの使用が重要である理由は、プラグインを OSGi サービスとして公開すれば、そのサービスを eXtreme Scale ランタイム環境が使用できるからです。製品は、Eclipse Equinox コア OSGi フレームワークの中で、Eclipse Gemini と Apache Aries の 2 つの blueprint コンテナをサポートします。次の手順を使用して、Eclipse Gemini コンテナをセットアップします。

### 手順

1. Eclipse Web サイト から、Eclipse Equinox SDK Version 3.6.1 以降をダウンロードします。Equinox フレームワーク用のディレクトリーを作成します。例えば、/opt/equinox です。以下の説明では、このディレクトリーを equinox\_root と呼びます。圧縮ファイルを equinox\_root ディレクトリーに解凍します。

2. Eclipse Web サイトから、gemini-blueprint 1.0.0 圧縮ファイルをダウンロードします。ファイルの内容を一時ディレクトリーに解凍し、解凍された次のファイルを `equinox_root/plugins` ディレクトリーにコピーします。

```
dist/gemini-blueprint-core-1.0.0.jar
dist/gemini-blueprint-extender-1.0.0.jar
dist/gemini-blueprint-io-1.0.0.jar
```

**重要:** 圧縮された Blueprint ファイルをダウンロードするロケーションに応じて、解凍されたファイルは、次のステップにある Spring フレームワーク JAR ファイルとよく似た拡張 `RELEASE.jar` を持つことがあります。ファイル名が `config.ini` ファイル内のファイル参照と一致することを確認する必要があります。

3. 次の SpringSource Web ページから、Spring Framework Version 3.0.5 をダウンロードします。 <http://www.springsource.com/download/community> それを一時ディレクトリーに解凍し、解凍された次のファイルを `equinox_root/plugins` ディレクトリーにコピーします。

```
org.springframework.aop-3.0.5.RELEASE.jar
org.springframework.asm-3.0.5.RELEASE.jar
org.springframework.beans-3.0.5.RELEASE.jar
org.springframework.context-3.0.5.RELEASE.jar
org.springframework.core-3.0.5.RELEASE.jar
org.springframework.expression-3.0.5.RELEASE.jar
```

4. SpringSource Web ページから、AOP Alliance Java アーカイブ (JAR) ファイルをダウンロードします。 `com.springsource.org.aopalliance-1.0.0.jar` を `equinox_root/plugins` ディレクトリーにコピーします。

5. SpringSource Web ページから、Apache commons logging 1.1.1 JAR ファイルをダウンロードします。 `com.springsource.org.apache.commons.logging-1.1.1.jar` ファイルを `equinox_root/plugins` ディレクトリーにコピーします。

6. Luminis OSGi Configuration Admin コマンド行クライアントをダウンロードします。このバンドルを使用して、OSGi 管理構成を管理します。次の Web ページから、JAR ファイルをダウンロードできます。 <https://opensource.luminis.net/wiki/display/SITE/OSGi+Configuration+Admin+command+line+client> `net.luminis.cmc-0.2.5.jar` を `equinox_root/plugins` ディレクトリーにコピーします。

7. 次の Web ページから、Apache Felix file installation Version 3.0.2 バンドルをダウンロードします。 <http://felix.apache.org/site/index.html> `org.apache.felix.fileinstall-3.0.2.jar` ファイルを `equinox_root/plugins` ディレクトリーにコピーします。

8. `equinox_root/plugins` ディレクトリーの中に、構成ディレクトリーを作成します。例えば次のとおりです。

```
mkdir equinox_root/plugins/configuration
```

9. 次の `config.ini` ファイルを、`equinox_root/plugins/configuration` ディレクトリーの中に作成します。このとき、`equinox_root` を、使用する `equinox_root` ディレクトリーの絶対パスに置き換え、各行の円記号 (¥) の後のすべての後続スペースを削除します。ファイルの最後に、ブランク行を含める必要があります。例えば次のとおりです。

```

osgi.noShutdown=true
osgi.java.profile.bootdelegation=none
org.osgi.framework.bootdelegation=none
eclipse.ignoreApp=true
osgi.bundles=¥
org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, ¥
org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, ¥
org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, ¥
com.springsource.org.apache.commons.logging-1.1.1.jar@1:start, ¥
com.springsource.org.aopalliance-1.0.0.jar@1:start, ¥
org.springframework.aop-3.0.5.RELEASE.jar@1:start, ¥
org.springframework.asm-3.0.5.RELEASE.jar@1:start, ¥
org.springframework.beans-3.0.5.RELEASE.jar@1:start, ¥
org.springframework.context-3.0.5.RELEASE.jar@1:start, ¥
org.springframework.core-3.0.5.RELEASE.jar@1:start, ¥
org.springframework.expression-3.0.5.RELEASE.jar@1:start, ¥
org.apache.felix.fileinstall-3.0.2.jar@1:start, ¥
net.luminis.cmc-0.2.5.jar@1:start, ¥
gemini-blueprint-core-1.0.0.jar@1:start, ¥
gemini-blueprint-extender-1.0.0.jar@1:start, ¥
gemini-blueprint-io-1.0.0.jar@1:start

```

既に環境をセットアップしている場合は、次のディレクトリーを削除することで、Equinox プラグイン・リポジトリーをクリーンアップできます。

```
equinox_root¥plugins¥configuration¥org.eclipse.osgi
```

10. 次のコマンドを実行して、Equinox コンソールを開始します。

別のバージョンの Equinox を実行している場合、JAR ファイル名は次の例のものとは異なります。

```
java -jar plugins¥org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

---

## REST データ・サービスのインストール

このトピックでは、WebSphere eXtreme Scale REST データ・サービスを Web サーバーにインストールする方法について説明します。

### 始める前に

#### ソフトウェア要件

WebSphere eXtreme Scale REST データ・サービスは、Java Web アプリケーションであり、Java サブレット仕様バージョン 2.3 および Java ランタイム環境バージョン 5 以上をサポートする任意のアプリケーション・サーバーにデプロイできます。

以下のソフトウェアが必要です。

- Java Standard Edition 5 以上
- 以下のいずれかを含んだ、Web サブレット・コンテナのバージョン 2.3 以上
  - WebSphere Application Server バージョン 6.1.0.25 以上
  - WebSphere Application Server バージョン 7.0.0.5 以上
  - WebSphere Community Edition バージョン 2.1.1.3 以上
  - Apache Tomcat バージョン 5.5 以上
- WebSphere eXtreme Scale バージョン 7.1 以上 (試用版を含む)

#### このタスクについて

WebSphere eXtreme Scale REST データ・サービスには、単一 wxsrestservice.war ファイルが含まれます。wxsrestservice.war には、WCF Data Services クライアン

ト・アプリケーションまたはその他の HTTP REST クライアントとデータ・グリッド間のゲートウェイとして機能する単一のサーブレットが含まれています。

REST データ・サービスには、迅速にデータ・グリッドを作成し、eXtreme Scale クライアントまたは REST データ・サービスを使用してそのグリッドと対話できるようにするサンプルが含まれています。サンプルの使用法の詳細については、409 ページの『REST データ・サービスの構成』を参照してください。

WebSphere eXtreme Scale 7.1 をインストールするか、eXtreme Scale バージョン 7.1 試用版を解凍した場合には、以下のディレクトリーおよびファイルが含まれます。

- `restservice_home/lib`

`lib` ディレクトリーには、以下のファイルが含まれます。

- `wxsrestservice.ear` – WebSphere Application Server および WebSphere Application Server CE で使用するための REST データ・サービス・エンタープライズ・アプリケーション・アーカイブ。
- `wxsrestservice.war` – Apache Tomcat で使用するための REST データ・サービス Web モジュール。

`wxsrestservice.ear` ファイルには、`wxsrestservice.war` ファイルが含まれており、ともに WebSphere WebSphere eXtreme Scale ランタイムに密結合されています。WebSphere eXtreme Scale を新しいバージョンにアップグレードするかフィックスパックを適用した場合には、`wxsrestservice.war` ファイルまたは `wxsrestservice.ear` ファイルを、このディレクトリーにインストールされたバージョンに手動でアップグレードする必要があります。

- `restservice_home/gettingstarted`

`gettingstarted` ディレクトリーには、WebSphere eXtreme Scale REST データ・サービスをデータ・グリッドで使用方法を説明する単純なサンプルが含まれます。

## 手順

REST データ・サービスをパッケージ化し、デプロイします。

REST データ・サービスは、必要なものを完備した WAR モジュールとして設計されています。REST データ・サービスを構成するには、まず、REST データ・サービス構成およびオプションの WebSphere eXtreme Scale 構成ファイルを JAR ファイルまたはディレクトリーにパッケージ化する必要があります。このアプリケーション・パッケージは、Web コンテナ・サービス・ランタイムによって参照されます。次の図に、eXtreme Scale REST データ・サービスで使用されるファイルを示します。

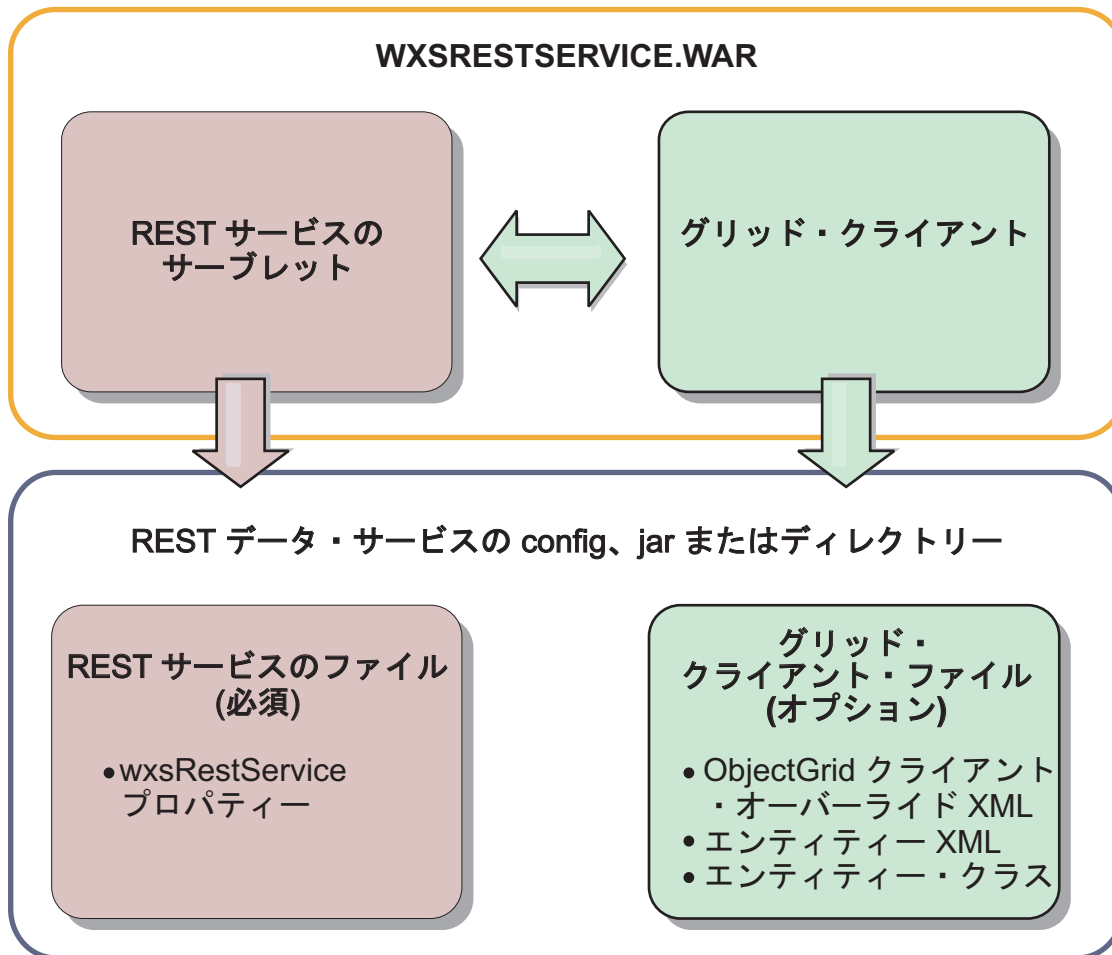


図 26. WebSphere eXtreme Scale REST データ・サービスのファイル

REST サービス構成 JAR またはディレクトリーには、以下のファイルが含まれている必要があります。

wxsRestService.properties: wxsRestService.properties ファイルには、REST データ・サービスの構成オプションが含まれます。これには、カタログ・サービス・エンドポイント、公開する ObjectGrid 名、トレース・オプションなどがあります。REST データ・サービスのプロパティー・ファイルを参照してください。

以下の ObjectGrid クライアント・ファイルはオプションです。

- META-INF/objectGridClient.xml: ObjectGrid クライアント・オーバーライド XML ファイルは、リモート・データ・グリッドに接続するために使用します。デフォルトでは、このファイルは必要ではありません。このファイルが存在しない場合には、REST サービスはサーバー構成を使用して、ニア・キャッシュを使用不可にします。

ファイルの名前は、objectGridClientXML REST データ・サービス構成プロパティーを使用してオーバーライドできます。この XML ファイルを提供する場合には、ファイルに以下を含める必要があります。

1. REST データ・サービスに公開するすべての ObjectGrid。
2. 各 ObjectGrid 構成に関連付けられたエンティティー記述子 XML ファイルへの参照。



- META-INF/エンティティ記述子 XML ファイル: クライアントでクライアントのエンティティ定義をオーバーライドする必要がある場合にのみ、1 つ以上のエンティティ記述子 XML ファイルが必要です。エンティティ記述子 XML ファイルは、ObjectGrid クライアント・オーバーライド XML 記述子ファイルと組み合わせて使用する必要があります。
- エンティティ・クラス。アノテーションが付けられたエンティティ・クラスまたはエンティティ記述子 XML ファイルを使用して、エンティティ・メタデータを記述できます。REST サービスでは、eXtreme Scale サーバーがエンティティ・メタデータ・クラスを使用して構成されていて、クライアント・オーバーライド・エンティティ XML 記述子を使用しない場合にのみ、クラスパス内にエンティティ・クラスが必要になります。

エンティティがサーバー上で XML で定義された、最小要件の構成ファイルを使用した例:

```
restserviceconfig.jar:
wxsRestService.properties
```

プロパティ・ファイルには、以下が含まれます。

```
catalogServiceEndpoints=localhost:2809
objectGridNames=NorthwindGrid
```

単一エンティティ、オーバーライド XML ファイル、およびエンティティ・クラスの例:

```
restserviceconfig.jar:
wxsRestService.properties
```

プロパティ・ファイルには、以下が含まれます。

```
catalogServiceEndpoints=localhost:2809
objectGridNames=NorthwindGrid
```

```
com/acme/entities/Customer.class
META-INF/objectGridClient.xml
```

クライアント ObjectGrid 記述子 XML ファイルには、以下が含まれます。

```
<objectGrid name="CustomerGrid" entityMetadataXMLFile="emd.xml"/>
META-INF/emd.xml
```

エンティティ・メタデータ記述子 XML ファイルには、以下が含まれます。

```
<entity class-name="com.acme.entities.Customer" name="Customer"/>
```

## eXtreme Scale バンドルのインストール

WebSphere eXtreme Scale には、Eclipse Equinox OSGi フレームワークにインストールできるバンドルが組み込まれています。OSGi 内で eXtreme Scale サーバーを開始したり、eXtreme Scale クライアントを使用したりするには、これらのバンドルが必要です。eXtreme Scale バンドルは、Equinox コンソールまたは config.ini 構成ファイルを使用してインストールすることができます。

### 始める前に

このタスクは、以下の製品をインストールしたことを前提としています。

- Eclipse Equinox OSGi フレームワーク

- eXtreme Scale スタンドアロン・クライアントまたはサーバー

## このタスクについて

eXtreme Scale には、2 つのバンドルが組み込まれています。各 OSGi フレームワークでは、次のバンドルのいずれか 1 つのみが必要になります。

### objectgrid.jar

サーバー・バンドルは objectgrid.jar ファイルであり、eXtreme Scale スタンドアロン・サーバーのインストールによってインストールされます。eXtreme Scale サーバーを実行するために必要なバンドルですが、eXtreme Scale クライアントまたはローカルのメモリー内キャッシュの実行にも使用できます。objectgrid.jar ファイルのバンドル ID は com.ibm.websphere.xs.server\_<version> で、バージョンのフォーマットは <Version>.<Release>.<Modification> です。例えば、このリリースのサーバー・バンドルは、com.ibm.websphere.xs.server\_8.5.0. です。

### ogclient.jar

ogclient.jar バンドルは、eXtreme Scale スタンドアロンおよびクライアントのインストール済み環境にインストールされ、eXtreme Scale クライアントまたはローカルのメモリー内キャッシュを実行するために使用されます。ogclient.jar ファイルのバンドル ID は com.ibm.websphere.xs.client\_<version> で、バージョンのフォーマットは <Version>\_<Release>\_<Modification> です。例えば、このリリースのクライアント・バンドルは、com.ibm.websphere.xs.server\_8.5.0 です。

eXtreme Scale プラグインの作成法の詳細については、システム API とプラグインのトピックを参照してください。

## Equinox コンソールを使用した Eclipse Equinox OSGi フレームワークへの eXtreme Scale クライアントまたはサーバー・バンドルのインストール

### 手順

1. コンソールを有効にするよう指定して Eclipse Equinox フレームワークを開始します。例えば、次のようにします。

```
java_home/bin/java -jar <equinox_root>/plugins/  
org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

2. Equinox コンソールで、eXtreme Scale クライアントまたはサーバー・バンドルをインストールします。

```
osgi> install file:///<path to bundle>
```

3. Equinox が、新しくインストールされたバンドルのバンドル ID を表示します。  
Bundle id is 25

4. Equinox コンソールで、次のようにバンドルを開始します。ここで、<id> は、バンドルのインストール時に割り当てられたバンドル ID です。

```
osgi> start <id>
```

5. Equinox コンソールで、サービス状況を取得して、バンドルが開始したことを確認します。例えば、次のようにします。

```
osgi> ss
```

バンドルが正常に開始されると、バンドルは ACTIVE 状態を表示します。例えば、次のとおりです。

```
25      ACTIVE      com.ibm.websphere.xs.server_8.5.0
```

**config.ini ファイルを使用して、eXtreme Scale クライアントまたはサーバー・バンドルを Eclipse Equinox OSGi フレームワークにインストールするには、以下のようになります。**

#### 手順

1. eXtreme Scale クライアントまたはサーバー (objectgrid.jar または ogclient.jar) バンドルを <wxs\_install\_root>/ObjectGrid/lib から、次の例のような Eclipse Equinox プラグイン・ディレクトリーにコピーします。 <equinox\_root>/plugins
2. Eclipse Equinox config.ini 構成ファイルを編集し、バンドルを osgi.bundles プロパティーに追加します。例えば、次のとおりです。

```
osgi.bundles=¥  
org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, ¥  
org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, ¥  
org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, ¥  
objectgrid.jar@1:start
```

**重要:** 最後のバンドル名の後に空白行があることを確認してください。各バンドルはコンマで区切ります。

3. コンソールを有効にするよう指定して Eclipse Equinox フレームワークを開始します。例えば、次のようになります。

```
java_home/bin/java -jar <equinox_root>/plugins/  
org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

4. Equinox コンソールで、サービス状況を取得して、バンドルが開始したことを確認します。

```
osgi> ss
```

バンドルが正常に開始されると、バンドルは ACTIVE 状態を表示します。例えば、次のとおりです。

```
25      ACTIVE      com.ibm.websphere.xs.server_8.5.0
```

#### タスクの結果

Eclipse Equinox OSGi フレームワークに eXtreme Scale サーバーまたはクライアント・バンドルがインストールされ、開始されました。

---

## WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール

WebSphere eXtreme Scale バージョン 8.5 または WebSphere eXtreme Scale クライアント バージョン 8.5 を、WebSphere Application Server または WebSphere Application Server Network Deployment がインストールされている環境にインストールできます。WebSphere Application Server または WebSphere Application Server Network Deployment の既存のフィーチャーを使用して、eXtreme Scale アプリケーションを拡張できます。

## 始める前に

- ターゲット・インストール・ディレクトリーに WebSphere eXtreme Scale および WebSphere eXtreme Scale クライアントの既存のインストール済み環境が含まれていないことを確認します。
- WebSphere Application Server または WebSphere Application Server Network Deployment 環境で実行中のすべてのプロセスを停止します。**stopManager**、**stopNode**、および **stopServer** コマンドについて詳しくは、コマンド行ユーティリティー (Command-line utilities) を参照してください。

### 注意:

すべての実行中のプロセスを必ず停止してください。実行中のプロセスを停止しなくてもインストールは続行しますが、一部のプラットフォームでは予測不能な結果が生じて、インストールが不確定状態になります。

**重要:** WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント をインストールする際には、WebSphere Application Server をインストールしたのと同じディレクトリーにインストールする必要があります。例えば、WebSphere Application Server を `C:\was_root` にインストールした場合、`C:\was_root` を、WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストールのターゲット・ディレクトリーとして選択する必要があります。

## このタスクについて

eXtreme Scale を WebSphere Application Server または WebSphere Application Server Network Deployment に統合して、eXtreme Scale の機能をご使用の Java Platform, Enterprise Edition アプリケーションに適用します。Java EE アプリケーションはデータ・グリッドをホスティングし、クライアント接続を使用してそのデータ・グリッドにアクセスします。

## 手順

1. WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント バージョン 8.5 を WebSphere Application Server バージョン 8 環境内にインストールする場合、以下のステップを実行します。
  - a. IBM Installation Manager をインストールします。詳しくは、183 ページの『GUI の使用による IBM Installation Manager のインストール』を参照してください。
  - b. Installation Manager を使用して、適切な eXtreme Scale 製品オフアリングをインストールします。
    - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
    - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8詳しくは、185 ページの『GUI の使用による製品 のインストール』を参照してください。
  - c. パスポート・アドバンテージのサイトから必要な WebSphere Application Server バージョン 8 のリポジトリーをダウンロードします。詳しくは、How to download WebSphere Application Server V8.0 from Passport Advantage Online を参照してください。

- d. WebSphere Application Server バージョン 8 をインストールします 詳しくは、 GUI を使用した、分散オペレーティング・システムでの製品のインストールを参照してください。
2. WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント バージョン 8.5 を WebSphere Application Server バージョン 7 環境内にインストールする必要がある場合、以下のステップを実行します。
  - a. IBM Installation Manager をインストールします。詳しくは、183 ページの『GUI の使用による IBM Installation Manager のインストール』を参照してください。
  - b. Installation Manager を使用して、適切な eXtreme Scale 製品オフオファリングをインストールします。
    - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
    - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7詳しくは、185 ページの『GUI の使用による製品 のインストール』を参照してください。
  - c. ISMP インストーラーを使用して WebSphere Application Server バージョン 7 をインストールします。詳しくは、 アプリケーション・サービス提供環境のインストールを参照してください。
  - d. WebSphere Application Server バージョン 7 を Installation Manager にインポートして、インストールを完了します。 WebSphere Application Server バージョン 7 を Installation Manager にインポートすると、製品のフィックスパックの管理とインストールが 1 つのロケーションから行えます。フィックスパックおよび更新にアクセスできるように、必要なリポジトリが Installation Manager 内にセットアップされていることを確認してください。WebSphere Application Server 7 の既存のインストール済み環境を Installation Manager にインポートする方法について詳しくは、IBM Installation Manager を参照してください。
3. WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント バージョン 8.5 を WebSphere Application Server バージョン 6 環境内にインストールする場合、以下のステップを実行します。
  - a. IBM Installation Manager をインストールします。詳しくは、183 ページの『GUI の使用による IBM Installation Manager のインストール』を参照してください。
  - b. Installation Manager を使用して、適切な eXtreme Scale 製品オフオファリングをインストールします。
    - WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
    - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6詳しくは、185 ページの『GUI の使用による製品 のインストール』を参照してください。
  - c. ISMP インストーラーを使用して WebSphere Application Server バージョン 6 をインストールします。詳しくは、 アプリケーション・サービス提供環境のインストールを参照してください

## 次のタスク

- WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント バージョン 8.5 インストール済み環境の構成を開始します。詳しくは、235 ページの『インストール後の最初のステップの実行』を参照してください。

---

## IBM Installation Manager の使用によるフィックスパックのインストール

IBM Installation Manager を使用して、WebSphere eXtreme Scale 製品オフオファリングに有効なフィックスパックでこの製品を更新することができます。フィックスパックは GUI、コマンド行、または応答ファイルを使用してインストールできます。

### GUI の使用によるフィックスパックのインストール

IBM Installation Manager ウィザードを使用して、この製品 を新しいバージョンに更新できます。

#### 始める前に

を参照してください。WebSphere eXtreme Scale スタンドアロンまたは WebSphere eXtreme Scale for WebSphere Application Server 製品オフオファリング用のアップグレードに関する情報については、IBM ソフトウェア・サポート・センターにお問い合わせください。最新の情報は、IBM ソフトウェア・サポート・センターおよび Fix Central から入手できます。

IBM Installation Manager を使用して、以下の製品オフオファリングに製品保守を適用します。

- スタンドアロン環境での WebSphere eXtreme Scale
- スタンドアロン環境での WebSphere eXtreme Scale クライアント
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 6.0
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 7.0
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 8.0
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6.0
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7.0
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8.0

Web ベースのサービス・リポジトリまたはローカル・サービス・リポジトリのロケーションがリストされ、チェックされているかどうか、あるいは Installation Manager 設定の「リポジトリ」パネルで「インストール中および更新中にサービス・リポジトリの検索」オプションが選択されているかどうかを確認してください。Installation Manager でのサービス・リポジトリの使用について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。



## このタスクについて

**制約事項:** Installation Manager を使用してインストール済み環境をアップグレードしたり、完全な WebSphere Application Server プロファイルのフィーチャーを追加または削除したりすることはできません。

### 手順

1. ご使用の環境で実行中のすべてのプロセスを停止します。
  - スタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する場合の詳細は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。
  - WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (Command-line utilities) を参照してください。
2. システムにログオンします。
3. 更新される WebSphere Application Server インストール済み環境で、すべてのサーバーおよびアプリケーションを停止します。
4. Installation Manager を開始します。
5. 「更新」をクリックします。

**注:** 認証のプロンプトが出されたら、保護されている IBM ソフトウェア Web サイトへのアクセスに使用する IBM ID とパスワードを使用してください。

6. 更新するパッケージ・グループを選択します。

**ヒント:** 「すべて更新」を選択すると、Installation Manager は過去にインストールしたすべてのパッケージ・グループに対する更新の追加リポジトリおよび定義済みリポジトリをすべて検索します。このフィーチャーは、ターゲット・リポジトリ内に含まれるフィックスについて完全に制御できる場合のみ使用してください。インストール対象の特定のフィックスのみを含む一連のカスタム・リポジトリを作成してそれを指定する場合は、自信を持ってこのフィーチャーを使用できるはずですが、サービス・リポジトリの検索を使用可能にする場合や、他の Web ベースのライブ・リポジトリから直接フィックスをインストールする場合は、このオプションを選択しない方がよいかもしれません。そうすると、後続のパネルで各オファリングに対してインストールするフィックスのみを選択できます。

7. 「次へ」をクリックします。
8. 更新後のバージョンを選択します。
  - スタンドアロン環境での WebSphere eXtreme Scale
  - スタンドアロン環境での WebSphere eXtreme Scale クライアント
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7

- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8
9. インストールするフィックスを選択します。  
  
デフォルトでは、任意の推奨フィックスが選択されます。  
  
推奨フィックスがある場合、推奨フィックスのみを表示し、非推奨フィックスを非表示にするオプションを選択できます。
  10. 「次へ」をクリックします。
  11. 使用条件の条項に同意し、「次へ」をクリックします。
  12. 更新後のインストール済み環境に必要なオプション・フィーチャーを選択します。
  13. 要約情報を検討して、「更新」をクリックします。
    - 正常にインストールされた場合は、プログラムにより、インストールが正常に行われたことを示すメッセージが表示されます。
    - インストールが正常に行われなかった場合は、「ログ・ファイルの表示」をクリックして、問題のトラブルシューティングを行います。
  14. 「終了」をクリックします。
  15. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。

## コマンド行の使用によるフィックスパックのインストール

IBM Installation Manager をコマンド行から使用して、WebSphere eXtreme Scale 製品オフリングに有効なフィックスパックで製品を更新することができます。

### 始める前に

を参照してください。 WebSphere eXtreme Scale スタンドアロンまたは WebSphere eXtreme Scale for WebSphere Application Server 製品オフリングのアップグレードに関する情報については、IBM ソフトウェア・サポート・センターにお問い合わせください。最新の情報は、IBM ソフトウェア・サポート・センターおよび Fix Central から入手できます。

IBM Installation Manager を使用して、以下の製品オフリングに製品保守を適用します。

- スタンドアロン環境での WebSphere eXtreme Scale
- スタンドアロン環境での WebSphere eXtreme Scale クライアント
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
- WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7
- WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8

## このタスクについて

**制約事項:** Installation Manager を使用してインストール済み環境をアップグレードしたり、完全な WebSphere Application Server プロファイルのフィーチャーを追加または削除したりすることはできません。

### 手順

1. WebSphere eXtreme Scale 8.5 に有効なインテリム・フィックスとフィックスパックのリスト、および各フィックスに関する具体的な情報を入手するには、次のアクションを実行します。
  - a. Fix Central に移動します。
  - b. **WebSphere** を製品グループとして選択します。
  - c. WebSphere eXtreme Scaleを製品として選択します。
  - d. インストール・バージョンとして **8.5** を選択します。
  - e. プラットフォームとしてご使用のオペレーティング・システムを選択し、「**続行**」をクリックします。
  - f. 「**フィックスの参照**」を選択し、「**続行**」をクリックします。
  - g. 各フィックスの下の「**詳細**」をクリックすると、そのフィックスの情報が表示されます。
  - h. **推奨:** インストールしたいフィックスパックの名前をメモしておいてください。
2. 次の手順を使用して、フィックスパックで WebSphere eXtreme Scale バージョン 8.5 を更新してください。
  - フィックスパックを含むファイルを Fix Central からダウンロードし、ローカル更新を行います。

フィックスパックを含む圧縮ファイルは Fix Central からダウンロードできます。フィックスパックの各圧縮ファイルには、そのフィックスパックの Installation Manager リポジトリが含まれていて、通常は .zip という拡張子が付いています。フィックスパック・ファイルをダウンロードして解凍した後、Installation Manager を使用してフィックスパックで WebSphere Application Server バージョン 8.x を更新します。

- a. フィックスパックをダウンロードするには、以下の手順を行います。
  - 1) Fix Central に移動します。
  - 2) **WebSphere** を製品グループとして選択します。
  - 3) **WebSphere eXtreme Scale**を製品として選択します。
  4. インストール・バージョンとして **8.5** を選択します。
  - 5) プラットフォームとしてご使用のオペレーティング・システムを選択し、「**続行**」をクリックします。
  - 6) 「**フィックスの参照**」を選択し、「**続行**」をクリックします。
  - 7) ダウンロードするフィックスパックを選択して、「**続行**」をクリックします。
  - 8) ダウンロード・オプションを選択して、「**続行**」をクリックします。
  - 9) ご使用条件に同意する場合は「**同意します**」をクリックします。

- 10) 「**今すぐダウンロード**」をクリックしてフィックスパックをダウンロードします。
  - 11) 圧縮ファイルをバイナリー形式でインストール先のシステムに転送します。
  - 12) 圧縮されたりポジトリリー・ファイルをシステム上のディレクトリーに解凍します。
- b. ダウンロードしたファイルからフィックスパックをインストールするには、次のアクションを実行します。
- 1) システムにログオンします。
  - 2) ご使用の環境で実行中のすべてのプロセスを停止します。WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (Command-line utilities) を参照してください。
  - 3) *Installation\_Manager\_binaries/eclipse/tools* ディレクトリーに移動します。ここでは、*Installation\_Manager\_binaries* が Installation Manager のインストール・ルート・ディレクトリーです。
  - 4) フィックスパックをインストールします。

UNIX

Linux

```
./imcl install offering_ID offering_version,optional_feature_ID
-installationDirectory product_installation_location
-repositories location_of_expanded_files
-acceptLicense
```

Windows

```
imcl.exe install offering_ID offering_version,optional_feature_ID
-installationDirectory product_installation_location
-repositories location_of_expanded_files
-acceptLicense
```

#### ヒント:

- *offering\_ID* は、173 ページの『WebSphere eXtreme Scale product offering IDs』 にリストされるオファリング ID です。
- *offering\_version* は、アンダースコアを使用してオファリング ID にオプションで付加することができますが、これはインストール対象のオファリングの特定バージョンです (例: 8.5.0.20110503\_0200)。
  - *offering\_version* が指定されていない場合、オファリングの最新バージョンと、そのバージョンの**すべての**インテリム・フィックスがインストールされます。
  - *offering\_version* が指定されている場合、オファリングの指定バージョンがインストールされ、そのバージョンのインテリム・フィックスはインストールされません。

リポジトリリーに対して次のコマンドを実行する際、オファリングのバージョンは、アンダースコアを使用してオファリング ID の最後に付加されている場合があります:

```
imcl listAvailablePackages -repositories source_repository
```

- また、none、recommended、または all を -installFixes 引数と共に指定し、オフリングと一緒にインストールするインテリム・フィックスを示すこともできます。
  - オフリングのバージョンが指定されていない場合、-installFixes オプションはデフォルトで all になります。
  - オフリングのバージョンが指定されている場合、-installFixes オプションはデフォルトで none になります。
  - コンマで区切られたフィーチャーのリストを追加できます。フィーチャーのリストが指定されていない場合は、デフォルトのフィーチャーがインストールされます。
- 5) **オプション:** すべてのインストール済みパッケージをリストして、インストールを検証します。

UNIX

Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

## 応答ファイルの使用によるフィックスパックのインストール

IBM Installation Manager で応答ファイルを使用して、この製品 を新しいバージョンに更新できます。

### 始める前に

**ヒント:** 本書で説明した手順に代わる手段として、Installation Manager では、応答ファイルまたはコマンド行で **updateAll** コマンドを使用してすべてのインストール済みパッケージの検索と更新を行うことができます。このコマンドは、ターゲット・リポジトリ内に含まれるフィックスについて完全に制御できる場合のみ使用してください。インストール対象の特定のフィックスのみを含む一連のカスタム・リポジトリを作成してそれを指定する場合は、自信を持ってこのコマンドを使用できるはずですが、サービス・リポジトリの検索を使用可能にする場合や、他の Web ベースのライブ・リポジトリから直接フィックスをインストールする場合は、このオプションを選択しない方がよいかもしれません。そうすると、コマンド行から **install** コマンドと共に -installFixes オプションを使用するか、応答ファイルで installFixes 属性を使用して、インストールするフィックスのみを選択できません。

### 手順

1. WebSphere eXtreme Scale バージョン 8.5 に有効なインテリム・フィックスとフィックスパックのリスト、および各フィックスに関する具体的な情報を入手するには、次のアクションを実行します。
  - a. Fix Central に移動します。
  - b. **WebSphere** を製品グループとして選択します。
  - c. WebSphere eXtreme Scale を製品として選択します。

- d. インストール・バージョンとして **8.x** を選択します。
  - e. プラットフォームとしてご使用のオペレーティング・システムを選択し、「**続行**」をクリックします。
  - f. 「**フィックスの参照**」を選択し、「**続行**」をクリックします。
  - g. 各フィックスの下の「**詳細**」をクリックすると、そのフィックスの情報が表示されます。
  - h. **推奨:** インストールしたいフィックスパックの名前をメモしておいてください。
2. 次の手順を使用して、フィックスパックで WebSphere eXtreme Scale 8.5 を更新してください。
- フィックスパックを含むファイルを Fix Central からダウンロードし、ローカル更新を行います。

フィックスパックを含む圧縮ファイルは Fix Central からダウンロードできます。フィックスパックの各圧縮ファイルには、そのフィックスパックの Installation Manager リポジトリが含まれていて、通常は .zip という拡張子が付いています。フィックスパック・ファイルをダウンロードして解凍した後、Installation Manager を使用してフィックスパックで WebSphere eXtreme Scale バージョン 8.5 を更新します。

- a. フィックスパックをダウンロードするには、以下の手順を行います。
  - 1) Fix Central に移動します。
  - 2) **WebSphere** を製品グループとして選択します。
  - 3) **WebSphere eXtreme Scale** を製品として選択します。
  - 4) インストール・バージョンとして **8.5** を選択します。
  - 5) プラットフォームとしてご使用のオペレーティング・システムを選択し、「**続行**」をクリックします。
  - 6) 「**フィックスの参照**」を選択し、「**続行**」をクリックします。
  - 7) ダウンロードするフィックスパックを選択して、「**続行**」をクリックします。
  - 8) ダウンロード・オプションを選択して、「**続行**」をクリックします。
  - 9) ご使用条件に同意する場合は「**同意します**」をクリックします。
  - 10) 「**今すぐダウンロード**」をクリックしてフィックスパックをダウンロードします。
  - 11) 圧縮ファイルをバイナリー形式でインストール先のシステムに転送します。
  - 12) 圧縮されたりポジトリ・ファイルをシステム上のディレクトリに解凍します。
- b. 次のアクションを実行します。
  - 1) システムにログオンします。
  - 2) リポジトリでユーザー名とパスワードが必要な場合、このリポジトリにアクセスするために鍵リング・ファイルを作成します。



Installation Manager の鍵リング・ファイルの作成について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

**ヒント:** 鍵リング・ファイルを作成する際、`imutilsc` コマンドで指定された URL が見つからない場合は、リポジトリ URL ロケーションの最後に `/repository.config` を追加してください。

- 3) ご使用のスタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する方法については、463 ページの『スタンドアロン・サーバーの停止』を参照してください。ご使用の WebSphere Application Server 環境で実行中のすべてのプロセスを停止する方法については、コマンド行ユーティリティーを参照してください。
- 4) `Installation_Manager_binaries/eclipse/tools` ディレクトリに移動します。ここでは、`Installation_Manager_binaries` が Installation Manager のインストール・ルート・ディレクトリです。
- 5) 応答ファイルを使用してフィックスパックをインストールします。

以下に例を示します。

– **Windows** 管理者または非管理者:

```
imcl.exe -acceptLicense
input C:%temp%update_response_file.xml
-log C:%temp%update_log.xml
-keyring C:%IM%im.keyring
```

– **UNIX** **Linux** 管理者:

```
./imcl -acceptLicense
input /var/temp/update_response_file.xml
-log /var/temp/update_log.xml
-keyring /var/IM/im.keyring
```

– **UNIX** **Linux** 非管理者:

```
./imcl -acceptLicense
input user_home/var/temp/update_response_file.xml
-log user_home/var/temp/update_log.xml
-keyring user_home/var/IM/im.keyring
```

---

## IBM Installation Manager の使用によるフィックスパックのアンインストール

IBM Installation Manager を使用して WebSphere eXtreme Scale 製品オフリングを前のバージョンにロールバックできます。フィックスパックは GUI、コマンド行、または応答ファイルを使用してアンインストールできます。

### GUI の使用によるフィックスパックのアンインストール

IBM Installation Manager GUI を使用して、この製品 を前のバージョンにロールバックできます。

## 始める前に

ロールバック・プロセス中は、Installation Manager から前バージョンのパッケージのファイルにアクセスする必要があります。デフォルトでは、これらのファイルはパッケージのインストール時に、ご使用のコンピューターに保管されます。デフォルト設定を変更した場合や保存されたファイルを削除した場合は、前バージョンのインストールに使用したりレジストリーに Installation Manager からアクセスする必要があります。

## このタスクについて

**制約事項:** Installation Manager を使用してインストール済み環境をロールバックしたり、フィーチャーを追加または削除したりすることはできません。

## 手順

1. ご使用の環境で実行中のすべてのプロセスを停止します。
  - スタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する場合の詳細は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。
  - WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (Command-line utilities) を参照してください。
2. Installation Manager を開始します。
3. 「**ロールバック**」をクリックします。
4. ロールバックするパッケージ・グループを選択します。
5. 「**次へ**」をクリックします。
6. ロールバック後のバージョンを選択します。
7. 「**次へ**」をクリックします。
8. 要約情報を検討して、「**ロールバック**」をクリックします。
  - 正常にロールバックされた場合は、プログラムにより、ロールバックが正常に行われたことを示すメッセージが表示されます。
  - ロールバックが正常に行われなかった場合は、「**ログ・ファイルの表示**」をクリックして、問題のトラブルシューティングを行います。
9. 「**終了**」をクリックします。
10. 「**ファイル**」>「**終了**」をクリックして、Installation Manager を閉じます。

## コマンド行の使用によるフィックスパックのアンインストール

コマンド行から IBM Installation Manager を使用して、この製品 を前のバージョンにロールバックできます。

## 始める前に

**制約事項:** この手順を実行するには、ご使用のシステムに Installation Manager バージョン 1.5 以降がインストールされている必要があります。

ロールバック・プロセス中は、Installation Manager から前バージョンのパッケージのファイルにアクセスする必要があります。デフォルトでは、これらのファイルは

パッケージのインストール時に、ご使用のコンピューターに保管されます。デフォルト設定を変更した場合や保存されたファイルを削除した場合は、前バージョンのインストールに使用したりリポジトリに **Installation Manager** からアクセスする必要があります。

## このタスクについて

**制約事項:** **Installation Manager** を使用してインストール済み環境をロールバックしたり、完全な **WebSphere Application Server** プロファイルのフィーチャーを追加または削除したりすることはできません。

## 手順

1. オプション: リポジトリでユーザー名とパスワードが必要な場合、このリポジトリにアクセスするために鍵リング・ファイルを作成します。

**Installation Manager** の鍵リング・ファイルの作成について詳しくは、**IBM Installation Manager** バージョン 1.5 インフォメーション・センターを参照してください。

**ヒント:** 鍵リング・ファイルを作成する際、**imutilsc** コマンドで指定された URL が見つからない場合は、リポジトリ URL ロケーションの最後に `/repository.config` を追加してください。

2. システムにログオンします。
3. ご使用の環境で実行中のすべてのプロセスを停止します。
  - スタンドアロン **eXtreme Scale** 環境で実行中のすべてのプロセスを停止する場合の詳細は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。
  - **WebSphere Application Server** 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (**Command-line utilities**) を参照してください。
4. **Installation Manager** をインストールしたディレクトリーの `eclipse/tools` サブディレクトリーに移動します。
5. **imcl** コマンドを使用して、この製品をロールバックします。

UNIX

Linux

```
./imcl rollback offering_ID_offering_version
-repositories source_repository
-installationDirectory installation_directory
-preferences preference_key=value
-properties property_key=value
-keyring keyring_file -password password
-acceptLicense
```

Windows

```
imcl.exe rollback offering_ID_offering_version
-repositories source_repository
-installationDirectory installation_directory
-preferences preference_key=value
-properties property_key=value
-keyring keyring_file -password password
-acceptLicense
```

## ヒント:

- *offering\_ID* は、173 ページの『WebSphere eXtreme Scale product offering IDs』 にリストされるオフリング ID です。
- *offering\_version* は、アンダースコアを使用してオフリング ID にオプションで付加することができますが、これはロールバック対象のオフリングの特定バージョンです (例: 8.5.0.20110503\_0200)。
  - *offering\_version* が指定されていない場合、インストール済み環境はそのオフリングの以前インストールされたバージョンにロールバックし、そのバージョンのすべてのインテリム・フィックスがインストールされます。
  - *offering\_version* が指定されている場合、インストール済み環境はそのオフリングの指定された前のバージョンにロールバックし、そのバージョンのインテリム・フィックスはインストールされません。

オフリングのバージョンは、**historyInfo** コマンドまたは **genHistoryReport** コマンドを *app\_server\_root/bin* ディレクトリーから実行する際に生成されるレポートの「パッケージ」セクションで、アンダースコアを使用してオフリング ID の最後に付加されている場合があります。

Installation Manager の使用について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

6. オプション: すべてのインストール済みパッケージをリストして、ロールバックを検証します。

UNIX

Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

## 応答ファイルを使用したフィックスパックのアンインストール

IBM Installation Manager で応答ファイルを使用して、この製品 を前のバージョンにロールバックできます。

### 始める前に

ロールバック・プロセス中は、Installation Manager から前バージョンのパッケージのファイルにアクセスする必要があります。デフォルトでは、これらのファイルはパッケージのインストール時に、ご使用のコンピューターに保管されます。デフォルト設定を変更した場合や保存されたファイルを削除した場合は、前バージョンのインストールに使用したりレジストリーに Installation Manager からアクセスする必要があります。

### このタスクについて

**制約事項:** Installation Manager を使用してインストール済み環境をロールバックしたり、完全な WebSphere Application Server プロファイルのフィーチャーを追加または削除したりすることはできません。

## 手順

1. オプション: リポジトリでユーザー名とパスワードが必要な場合、このリポジトリにアクセスするために鍵リング・ファイルを作成します。

Installation Manager の鍵リング・ファイルの作成については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

**ヒント:** 鍵リング・ファイルを作成する際、`imutilsc` コマンドで指定された URL が見つからない場合は、リポジトリ URL ロケーションの最後に `/repository.config` を追加してください。

2. システムにログオンします。
3. ご使用の環境で実行中のすべてのプロセスを停止します。
  - スタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する場合の詳細は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。
  - WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (Command-line utilities) を参照してください。
4. 応答ファイルを使用して、この製品をロールバックします。

Installation Manager をインストールしたディレクトリーの `eclipse/tools` サブディレクトリーに移動して、この製品をロールバックします。

以下に例を示します。

- **Windows** 管理者または非管理者:

```
imcl.exe
input C:%temp%rollback_response_file.xml
-log C:%temp%rollback_log.xml
-keyring C:%IM%im.keyring
```

- **UNIX** **Linux** 管理者:

```
./imcl
input /var/temp/rollback_response_file.xml
-log /var/temp/rollback_log.xml
-keyring /var/IM/im.keyring
```

- **UNIX** **Linux** 非管理者:

```
./imcl
input user_home/var/temp/rollback_response_file.xml
-log user_home/var/temp/rollback_log.xml
-keyring user_home/var/IM/im.keyring
```

**注:** プログラムが重要なポストインストール指示を標準出力に書き込むことがあります。

Installation Manager の使用については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

5. オプション: すべてのインストール済みパッケージをリストして、ロールバックを検証します。

UNIX

Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

---

## IBM Installation Manager の使用による 製品 のアンインストール

IBM Installation Manager を使用して WebSphere eXtreme Scale 製品オフラインをアンインストールします。

### GUI の使用による the product のアンインストール

IBM Installation Manager のウィザード・コンソールを使用して製品 をアンインストールできます。

#### 手順

1. 製品をアンインストールします。
  - a. ご使用の環境で実行中のすべてのプロセスを停止します。
    - スタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する場合の詳細は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。
    - WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (Command-line utilities) を参照してください。
  - b. Installation Manager を開始します。
  - c. 「アンインストール」をクリックします。
  - d. 「パッケージのアンインストール」ウィンドウで、以下のアクションを実行します。
    - 1) 以下のいずれかと適切なバージョンを選択します。
      - スタンドアロン環境での WebSphere eXtreme Scale
      - スタンドアロン環境での WebSphere eXtreme Scale クライアント
      - WebSphere eXtreme Scale for WebSphere Application Server バージョン 6c
      - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
      - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
      - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6
      - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7
      - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8
    - 2) 「次へ」をクリックします。



- e. サマリー情報を検討します。
  - f. 「アンインストール」をクリックします。
    - アンインストールが正常に行われた場合には、正常に終了したことを示すメッセージがプログラムによって表示されます。
    - アンインストールが正常に行われなかった場合には、「ログの表示」をクリックして、問題をトラブルシューティングします。
  - g. 「終了」をクリックします。
  - h. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。
2. オプション: IBM Installation Manager をアンインストールします。

**重要:** IBM Installation Manager をアンインストールする前に、Installation Manager によってインストールされたすべてのパッケージをアンインストールする必要があります。

この手順の実行については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

## コマンド行の使用によるこの製品 のアンインストール

コマンド行から IBM Installation Manager を使用してこの製品 をアンインストールできます。

### 手順

1. システムにログオンします。
2. ご使用の環境で実行中のすべてのプロセスを停止します。
  - スタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する場合は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。
  - WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティー (Command-line utilities) を参照してください。
3. Installation Manager をインストールしたディレクトリーの eclipse/tools サブディレクトリーに移動します。
4. **imcl** コマンドを使用して、この製品をアンインストールします。

#### Windows

```
imcl.exe uninstall com.ibm.websphere.v85,optional_feature_ID  
-installationDirectory installation_directory
```

#### UNIX

#### Linux

```
./imcl uninstall com.ibm.websphere.v85,optional_feature_ID  
-installationDirectory installation_directory
```

### ヒント:

- *offering\_ID* は、173 ページの『WebSphere eXtreme Scale product offering IDs』 にリストされるオフERING ID です。

- コンマとフィーチャー ID で区切られたフィーチャーのリストを削除できます。以下に例を示します。
 

```
imcl uninstall com.ibm.websphere.WXS.v85,xs.console.feature,xs.samples.feature
```

  - client スタンドアロン・クライアントのフィーチャーを示します
  - server スタンドアロン・サーバーのフィーチャーを示します
  - console Web ベースのモニター・コンソールを示します
  - samples はサンプルを示します。
- フィーチャーのリストが指定されていない場合は、製品全体がアンインストールされます。

詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

5. オプション: IBM Installation Manager をアンインストールします。

**重要:** IBM Installation Manager をアンインストールする前に、Installation Manager によってインストールされたすべてのパッケージをアンインストールする必要があります。

アンインストール・スクリプトを使用してこの手順を実行する方法について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

## 応答ファイルを使用した製品 のアンインストール

IBM Installation Manager で応答ファイルを使用してこの製品 をアンインストールできます。

### 始める前に

**オプション:** Installation Manager のインストールおよび製品のインストールを実行して (あるいは記録のみを行って)、いずれかのシステムの一時インストール・レジストリーに記録します。こうすることにより、Installation Manager がインストールされている標準レジストリーを使用せずに、この一時レジストリーを使用して、アンインストールを記録することができます。

### このタスクについて

Installation Manager を使用して、応答ファイルを処理してさまざまな方法で製品のアンインストールを行うことができます。次の手順で説明するように GUI を使用して応答ファイルを記録することができます。あるいは、手動で、またはサンプルを入手して変更することによって、新しい応答ファイルを生成することもできます。

### 手順

1. ご使用の環境で実行中のすべてのプロセスを停止します。
  - スタンドアロン eXtreme Scale 環境で実行中のすべてのプロセスを停止する場合の詳細は、463 ページの『スタンドアロン・サーバーの停止』を参照してください。

- WebSphere Application Server 環境で実行中のすべてのプロセスを停止する場合は、コマンド行ユーティリティ (Command-line utilities) を参照してください。
2. オプション: 製品をアンインストールする応答ファイルの記録: いずれかのシステムで、以下のアクションを実行して、製品をアンインストールする応答ファイルを記録します。
    - a. コマンド行で、Installation Manager をインストールしたディレクトリーの Eclipse サブディレクトリーに移動します。
    - b. `-record` オプションを使用して、コマンド行から Installation Manager を開始します。

以下に例を示します。

- **Windows** 管理者または非管理者:  

```
IBMIM.exe -skipInstall "C:%temp%imRegistry"
          -record C:%temp%uninstall_response_file.xml
```
- **UNIX** **Linux** 管理者:  

```
./IBMIM -skipInstall /var/temp/imRegistry
          -record /var/temp/uninstall_response_file.xml
```
- **UNIX** **Linux** 非管理者:  

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
          -record user_home/var/temp/uninstall_response_file.xml
```

**ヒント:** 「始める前に」の説明に従って作成した一時インストール・レジストリーで `-skipInstall` パラメーターの使用を選択すると、Installation Manager は応答ファイルを記録する際に一時インストール・レジストリーを使用します。`-skipInstall` パラメーターが指定されていると、いずれの製品パッケージもインストールまたはアンインストールされないことにご注意ください。Installation Manager で行うすべてのアクションは、指定の一時レジストリーに保管されているインストール・データを更新するのみです。応答ファイルの生成後に、この応答ファイルを使用して、製品ファイルを削除して標準インストール・レジストリーを更新し、製品をアンインストールできます。

`-skipInstall` の操作は、Installation Manager が使用する実際のエージェント・データ・ロケーションで使用することはできません。これはサポートされていません。空の書き込み可能なロケーションを使用し、今後の記録セッションにはそのロケーションを再使用してください。

詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

- c. 「アンインストール」をクリックします。
- d. 「パッケージのアンインストール」ウィンドウで、以下のアクションを実行します。
  - 1) 以下のいずれかと適切なバージョンを選択します。
    - スタンドアロン環境での WebSphere eXtreme Scale
    - スタンドアロン環境での WebSphere eXtreme Scale クライアント

- WebSphere eXtreme Scale for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale for WebSphere Application Server バージョン 8
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 6
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 7
  - WebSphere eXtreme Scale クライアント for WebSphere Application Server バージョン 8
- 2) 「次へ」をクリックします。
  - 3) 「次へ」をクリックします。
- e. サマリー情報を検討します。
  - f. 「アンインストール」をクリックします。
    - アンインストールが正常に行われた場合には、正常に終了したことを示すメッセージがプログラムによって表示されます。
    - アンインストールが正常に行われなかった場合には、「ログの表示」をクリックして、問題をトラブルシューティングします。
  - g. 「終了」をクリックします。
  - h. 「ファイル」>「終了」をクリックして、Installation Manager を閉じます。
3. 応答ファイルを使用した製品のアンインストール: 製品をアンインストールする各システムのコマンド行で、Installation Manager をインストールしたディレクトリーの eclipse/tools サブディレクトリーに移動し、作成した応答ファイルを使用して製品をアンインストールします。

以下に例を示します。

- **Windows** 管理者または非管理者:

```
imcl.exe
input C:%temp%\uninstall_response_file.xml
-log C:%temp%\uninstall_log.xml
```

- **UNIX** **Linux** 管理者:

```
./imcl
input /var/temp/uninstall_response_file.xml
-log /var/temp/uninstall_log.xml
```

- **UNIX** **Linux** 非管理者:

```
./imcl
input user_home/var/temp/uninstall_response_file.xml
-log user_home/var/temp/uninstall_log.xml
```

詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

4. オプション: すべてのインストール済みパッケージをリストして、アンインストールを検証します。

UNIX

Linux

```
./imcl listInstalledPackages
```

Windows

```
imcl listInstalledPackages
```

5. オプション: IBM Installation Manager をアンインストールします。

**重要:** IBM Installation Manager をアンインストールする前に、Installation Manager によってインストールされたすべてのパッケージをアンインストールする必要があります。

アンインストール・スクリプトを使用してこの手順を実行する方法については、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

---

## WebSphere eXtreme Scale のプロファイルの作成および拡張

製品のインストール後、WebSphere eXtreme Scale の固有のタイプのプロファイルを作成し、既存のプロファイルを拡張します。

### 始める前に

WebSphere eXtreme Scale をインストールします。詳しくは、163 ページの『インストールの概要』を参照してください。

WebSphere eXtreme Scale で使用するプロファイルの拡張はオプションですが、以下の使用シナリオでは必須です。

- WebSphere Application Server プロセスでカタログ・サービスまたはコンテナを自動始動する場合。サーバー・プロファイルを拡張しないと、サーバーを始動するには、必ずプログラマチックに行う (ServerFactory API を使用するか、別プロセスとして **startOgServer** スクリプトを使用する) 必要があります。
- Performance Monitoring Infrastructure (PMI) を使用して、WebSphere eXtreme Scale メトリックをモニターする場合。
- WebSphere Application Server 管理コンソールで WebSphere eXtreme Scale のバージョンを表示する場合。

### このタスクについて

**WebSphere Application Server バージョン 6.1 またはバージョン 7.0 での実行**

ご使用の環境に WebSphere Application Server バージョン 6.1 またはバージョン 7.0 が含まれている場合、プロファイル管理ツール・プラグインまたは **manageprofiles** コマンドを使用して、プロファイルを作成および拡張することができます。

### 次のタスク

実行するタスクに応じて、ファースト・ステップ・コンソールを起動して、製品環境の構成とテストを支援します。ファースト・ステップ・コンソールは、

`wxs_install_root¥firststeps¥wxs¥firststeps.bat` ディレクトリーにあります。また、前のタスクのいずれかを繰り返すことによって、追加プロファイルを作成または拡張することもできます。

## プロファイルを作成するグラフィカル・ユーザー・インターフェースの使用

プロファイル管理ツール・プラグインで提供されているグラフィカル・ユーザー・インターフェース (GUI) を使用して WebSphere eXtreme Scale のプロファイルを作成します。プロファイルとは、ランタイム環境を定義するファイル・セットです。

### 始める前に

次のシナリオでは、プロファイルを拡張するとき GUI は使用できません。

#### • WebSphere Application Server の 64 ビット・インストール:

WebSphere Application Server の 64 ビット・インストールの場合、プロファイル管理ツールは存在しません。これらのインストールでは、コマンド行から **manageprofiles** スクリプトを使用してください。

### このタスクについて

製品フィーチャーを使用するために、プロファイル管理ツール・プラグインでは、GUI を使用してプロファイル (WebSphere Application Server プロファイル、デプロイメント・マネージャーのプロファイル、セルのプロファイル、およびカスタム・プロファイルなど) のセットアップを行うことができます。プロファイルは WebSphere eXtreme Scale のインストール時またはインストール後に拡張できます。

### 手順

プロファイル管理ツール GUI を使用してプロファイルを作成します。ウィザードを開始するには、以下のオプションのいずれかを選択します。

- ファースト・ステップ・コンソールから「**プロファイル管理ツール**」を選択します。
- 「**スタート**」メニューからプロファイル管理ツールにアクセスします。
- `install_root/bin/ProfileManagement` ディレクトリーから `./pmt.sh|bat` スクリプトを実行します。

### 次のタスク

追加のプロファイルを作成したり、既存のプロファイルを拡張したりできます。プロファイル管理ツールを再始動するには、`was_root/bin/ProfileManagement` ディレクトリーから `./pmt.sh|bat` コマンドを実行するか、ファースト・ステップ・コンソールで「**プロファイル管理ツール**」を選択します。

ご使用の WebSphere Application Server 環境で、カタログ・サービスの開始、コンテナの開始、および TCP ポートの構成を実行します。詳細については、287 ページの『WebSphere eXtreme Scale と WebSphere Application Server の構成』を参照してください。



## プロファイルを拡張するグラフィカル・ユーザー・インターフェースの使用

製品をインストールした後で、既存のプロファイルを拡張し、WebSphere eXtreme Scale と互換性を持たせることができます。

### このタスクについて

既存のプロファイルを拡張する場合、製品固有の拡張テンプレートを適用してプロファイルの変更をします。例えば、WebSphere eXtreme Scale サーバーは、サーバー・プロファイルが `xs_augment` テンプレートで拡張されていない限り、自動始動されません。

- eXtreme Scale クライアントまたは、クライアントとサーバーをインストールしている場合、`xs_augment` テンプレートを使用してプロファイルを拡張します。
- 区画化機能をインストールしている場合のみ、`pf_augment` テンプレートを使用してプロファイルを拡張します。
- ご使用の環境に eXtreme Scale クライアントと区画化機能が含まれている場合は、両方のテンプレートを適用します。

### 手順

プロファイル管理ツール GUI を使用して、eXtreme Scale のプロファイルを拡張します。ウィザードを開始するには、以下のオプションのいずれかを選択します。

- ファースト・ステップ・コンソールから「**プロファイル管理ツール**」を選択します。
- 「**スタート**」メニューからプロファイル管理ツールにアクセスします。
- `was_root/bin/ProfileManagement` ディレクトリーから `.pmt.sh|bat` スクリプトを実行します。

### 次のタスク

追加のプロファイルを拡張することもできます。プロファイル管理ツールを再始動するには、`was_root/bin/ProfileManagement` ディレクトリーから `./pmt.sh|bat` コマンドを実行するか、ファースト・ステップ・コンソールで「**プロファイル管理ツール**」を選択します。

ご使用の WebSphere Application Server 環境で、カタログ・サービスの開始、コンテナの開始、および TCP ポートの構成を実行します。詳しくは、287 ページの『WebSphere eXtreme Scale と WebSphere Application Server の構成』を参照してください。

## manageprofiles コマンド

**manageprofiles** ユーティリティーで、WebSphere eXtreme Scale テンプレートを使用してプロファイルを作成したり、eXtreme Scale 拡張テンプレートを使用して既存のアプリケーション・サーバー・プロファイルの拡張および拡張解除を行えます。製品のこの機能を使用するには、ご使用の環境に含まれる少なくとも 1 つのプロファイルが製品用に拡張されている必要があります。

- プロファイルを作成および拡張する前に、eXtreme Scale をインストールする必要があります。詳しくは、205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。

## 目的

**manageprofiles** コマンドは、プロファイルと呼ばれる一連のファイルに、製品プロセスのランタイム環境を作成します。プロファイルは、ランタイム環境を定義します。**manageprofiles** コマンドを使用して、以下のアクションを行うことができます。

- デプロイメント・マネージャー・プロファイルの作成および拡張
- カスタム・プロファイルの作成および拡張
- スタンドアロン・アプリケーション・サーバー・プロファイルの作成および拡張
- セル・プロファイルの作成および拡張
- 任意のタイプのプロファイルの拡張解除

既存のプロファイルを拡張する場合、製品固有の拡張テンプレートを適用してプロファイルの変更をします。

- eXtreme Scale のクライアント、または、そのクライアントとサーバーの両方をインストールしている場合、**xs\_augment** テンプレートを使用してプロファイルの拡張をします。
- 区画化機能のみをインストールしている場合は、**pf\_augment** テンプレートを使用してプロファイルの拡張をします。
- ご使用の環境に eXtreme Scale クライアントと区画化機能が含まれている場合は、両方のテンプレートを適用します。

## ロケーション

このコマンド・ファイルは、*install\_root/bin* ディレクトリにあります。

## 使用法

詳しいヘルプが必要な場合、以下のように **-help** パラメーターを使用してください。

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/dmgr -help
```

以降のセクションで、**manageprofiles** コマンドを使用して実行できる各タスクを、必須パラメーターのリストと共に説明します。各タスクに指定するオプション・パラメーターに関する詳細は、WebSphere Application Server インフォメーション・センターの **manageprofiles** コマンドを参照してください。

## デプロイメント・マネージャー・プロファイルの作成

**manageprofiles** コマンドを使用して、デプロイメント・マネージャー・プロファイルを作成できます。デプロイメント・マネージャーはセルに統合されているアプリケーション・サーバーを管理します。

## パラメーター

### -create

プロファイルを作成します。(必須)

### -templatePath *template\_path*

テンプレートへのファイル・パスを指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/dmgr
```

ここで、*template\_type* は `xs_augment` または `pf_augment` です。

## 例

- `xs_augment` テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/dmgr
```

- `pf_augment` テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/dmgr
```

## カスタム・プロファイルの作成

`manageprofiles` コマンドを使用して、はカスタム・プロファイルを作成します。カスタム・プロファイルは、アプリケーション・サーバー、クラスター、またはその他の Java プロセスを組み込むようにデプロイメント・マネージャーを介してカスタマイズする空のノードです。

## パラメーター

### -create

プロファイルを作成します。(必須)

### -templatePath *template\_path*

テンプレートへのファイル・パスを指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/managed
```

ここで、*template\_type* は `xs_augment` または `pf_augment` です。

## 例

- `xs_augment` テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/managed
```

- `pf_augment` テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/managed
```

## スタンドアロン・アプリケーション・サーバー・プロファイルの作成

`manageprofiles` コマンドを使用して、はスタンドアロン・アプリケーション・サーバー・プロファイルを作成します。

## パラメーター

### -create

プロファイルを作成します。(必須)

**-templatePath** *template\_path*

テンプレートへのファイル・パスを指定します。(必須)

以下のフォーマット設定を使用します。

**-templatePath** *install\_root/profileTemplates/template\_type/default*

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

## 例

- *xs\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/default
```

- *pf\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/default
```

## セル・プロファイルの作成

**manageprofiles** コマンドを使用して、デプロイメント・マネージャーおよびアプリケーション・サーバーからなるセル・プロファイルを作成します。

## パラメーター

デプロイメント・マネージャー・テンプレートに以下のパラメーターを指定します。

**-create**

プロファイルを作成します。(必須)

**-templatePath** *template\_path*

テンプレートへのファイル・パスを指定します。(必須)

以下のフォーマット設定を使用します。

**-templatePath** *install\_root/profileTemplates/template\_type/cell/dmgr*

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

アプリケーション・サーバー・テンプレートを使用して以下のパラメーターを指定します。

**-create**

プロファイルを作成します。(必須)

**-templatePath** *template\_path*

テンプレートへのファイル・パスを指定します。(必須)

以下のフォーマット設定を使用します。

**-templatePath** *install\_root/profileTemplates/template\_type/cell/default*

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

## 例

- *xs\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/cell/dmgr  
-nodeProfilePath install_root/profiles/AppSrv01 -cellName cell01dmgr -nodeName node01dmgr  
-appServerNodeName node01
```

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/cell/default
```

```
-dmgrProfilePath install_root/profiles/Dmgr01 -portsFile
install_root/profiles/Dmgr01/properties/portdef.props -nodePortsFile
install_root/profiles/Dmgr01/properties/nodeportdef.props -cellName cell01dmgr
-nodeName node01dmgr -appServerNodeName node01
```

- **pf\_augment** テンプレートを使用する場合:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/cell/dmgr
-nodeProfilePath install_root/profiles/AppSrv01 -cellName cell01dmgr -nodeName node01dmgr
-appServerNodeName node01
```

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/cell/default
-dmgrProfilePath install_root/profiles/Dmgr01 -portsFile
install_root/profiles/Dmgr01/properties/portdef.props -nodePortsFile
install_root/profiles/Dmgr01/properties/nodeportdef.props -cellName cell01dmgr
-nodeName node01dmgr -appServerNodeName node01
```

## デプロイメント・マネージャー・プロファイルの拡張

**manageprofiles** コマンドを使用して、デプロイメント・マネージャー・プロファイル  
を拡張します。

### パラメーター

#### **-augment**

既存のプロファイルを拡張します。(必須)

#### **-profileName**

プロファイルの名前を指定します。(必須)

#### **-templatePath** *template\_path*

インストール・ルート・ディレクトリーにあるテンプレート・ファイルへのパス  
を指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/dmgr
```

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

### 例

- **xs\_augment** テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/xs_augment/dmgr
```

- **pf\_augment** テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/pf_augment/dmgr
```

## カスタム・プロファイルの拡張

**manageprofiles** コマンドを使用して、カスタム・プロファイルを拡張します。

### パラメーター

#### **-augment**

既存のプロファイルを拡張します。(必須)

#### **-profileName**

プロファイルの名前を指定します。(必須)

#### **-templatePath** *template\_path*

インストール・ルート・ディレクトリーにあるテンプレート・ファイルへのパス  
を指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/managed
```

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

## 例

- *xs\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01  
-templatePath install_root/profileTemplates/xs_augment/managed
```

- *pf\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01  
-templatePath install_root/profileTemplates/pf_augment/managed
```

## スタンドアロン・アプリケーション・サーバー・プロファイルの拡張

**manageprofiles** コマンドを使用して、スタンドアロン・アプリケーション・サーバー・プロファイルを拡張します。

### パラメーター

#### **-augment**

既存のプロファイルを拡張します。(必須)

#### **-profileName**

プロファイルの名前を指定します。(必須)

#### **-templatePath *template\_path***

インストール・ルート・ディレクトリーにあるテンプレート・ファイルへのパスを指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/default
```

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

## 例

- *xs\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01  
-templatePath install_root/profileTemplates/xs_augment/default
```

- *pf\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01  
-templatePath install_root/profileTemplates/pf_augment/default
```

## セル・プロファイルの拡張

**manageprofiles** コマンドを使用して、セル・プロファイルを拡張します。

### パラメーター

デプロイメント・マネージャー・プロファイルに以下のパラメーターを指定します。

#### **-augment**

既存のプロファイルを拡張します。(必須)



### **-profileName**

プロファイルの名前を指定します。(必須)

### **-templatePath** *template\_path*

インストール・ルート・ディレクトリーにあるテンプレート・ファイルへのパスを指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/cell/dmgr
```

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

アプリケーション・サーバー・プロファイルに以下のパラメーターを指定します。

### **-augment**

既存のプロファイルを拡張します。(必須)

### **-profileName**

プロファイルの名前を指定します。(必須)

### **-templatePath** *template\_path*

インストール・ルート・ディレクトリーにあるテンプレート・ファイルへのパスを指定します。(必須)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/cell/default
```

ここで、*template\_type* は *xs\_augment* または *pf\_augment* です。

## **例**

- *xs\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01 -templatePath install_root
/profileTemplates/xs_augment/cell/dmgr
./manageprofiles.sh|bat -augment -profileName profile01 -templatePath install_root
/profileTemplates/xs_augment/cell/default
```

- *pf\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -augment -profileName profile01 -templatePath install_root
/profileTemplates/pf_augment/cell/dmgr
./manageprofiles.sh|bat -augment -profileName profile01 -templatePath install_root
/profileTemplates/pf_augment/cell/default
```

## **プロファイルの拡張解除**

プロファイルを拡張解除する場合は、必須の **-unaugment** パラメーターと **-profileName** パラメーターを指定した上で、**-ignoreStack** パラメーターを **-templatePath** パラメーターと共に指定します。

## **パラメーター**

### **-unaugment**

前に拡張されたプロファイルを拡張解除します。(必須)

### **-profileName**

プロファイルの名前を指定します。このパラメーターは、値が指定されていない場合にデフォルトで発行されます。(必須)

### **-templatePath** *template\_path*

インストール・ルート・ディレクトリーにあるテンプレート・ファイルへのパスを指定します。(オプション)

以下のフォーマット設定を使用します。

```
-templatePath install_root/profileTemplates/template_type/profile_type
```

ここで、*template\_type* は *xs\_augment* または *pf\_augment* で、*profile\_type* は次の 4 つのプロファイル・タイプのいずれかです。

- *dmgr*: デプロイメント・マネージャー・プロファイル
- *managed*: カスタム・プロファイル
- *default*: スタンドアロン・アプリケーション・サーバー・プロファイル
- *cell*: セル・プロファイル

### **-ignoreStack**

拡張されている特定のプロファイルを拡張解除するために、**-templatePath** パラメーターとともに使用されます。(オプション)

## 例

- *xs\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -unaugment -profileName profile01 -ignoreStack  
-templatePath install_root/profileTemplates/xs_augment/profile_type
```

- *pf\_augment* テンプレートを使用する場合:

```
./manageprofiles.sh|bat -unaugment -profileName profile01 -ignoreStack  
-templatePath install_root/profileTemplates/pf_augment/profile_type
```

## 非 root プロファイル

非 root ユーザーが製品のプロファイルを作成できるように、非 root ユーザーにファイルおよびディレクトリーへのアクセス権を付与してください。非 root ユーザーは、root ユーザー、別の非 root ユーザー、または同じ非 root ユーザーが作成したプロファイルを拡張することもできます。

WebSphere Application Server 環境では、非 root (非管理者) ユーザーは、それぞれの環境において可能なプロファイルの作成、および使用が制限されています。プロファイル管理ツール・プラグインでは、非 root ユーザーに対して固有名とポート値は使用不可になっています。非 root ユーザーは、プロファイル名、ノード名、セル名、およびポートの割り当てについて、プロファイル管理ツールのデフォルト・フィールド値を変更する必要があります。各フィールドで、非 root ユーザーに一定範囲の値を割り当てることを検討してください。非 root ユーザーに対して、適切な値の範囲を守る責任と、独自の定義の整合性を維持する責任を割り当てることができます。

用語「インストーラー」は、root ユーザーまたは非 root ユーザーのいずれかを指します。インストーラーとして、非 root ユーザーにプロファイルを作成し、独自の製品環境を確立する許可を与えることができます。例えば、非 root ユーザーが所有するプロファイルを持ったアプリケーション・デプロイメントをテストする製品環境を作成する場合があります。非 root ユーザーにプロファイルの作成を許可するため完了する具体的なタスクには、次の項目があります。

- 非 root ユーザーが特定のプロファイルの場合に WebSphere Application Server を開始できるように、プロファイルを作成し、プロファイル・ディレクトリーの所有権を非 root ユーザーに割り当てます。
- 非 root ユーザーに適切なファイルおよびディレクトリーの書き込み許可を与えます。これにより、非 root ユーザーはプロファイルを作成できるようになります。このタスクで、プロファイルの作成を許可されたユーザーのグループを作成したり、個々のユーザーがプロファイルを作成できるようにすることができます。
- 製品の保守パッケージをインストールします。これには、非ユーザーにより所有されている既存のプロファイルに必要なサービスが含まれます。インストーラーであれば、保守パッケージが作成するすべての新規ファイルの所有者です。

非 root ユーザーのプロファイルの作成について詳しくは、非 root ユーザーのプロファイルの作成を参照してください。

インストーラーとして、非 root ユーザーがプロファイルを拡張する許可を与えることもできます。例えば、非 root ユーザーはインストーラーによって作成されたプロファイルを拡張したり、作成するプロファイルを拡張したりすることができます。WebSphere Application Server Network Deployment 非 root ユーザー拡張プロセスに従ってください。

ただし、インストーラーによって作成されたプロファイルを非 root ユーザーが拡張する際に、非 root ユーザーは拡張前に以下のファイルを作成する必要はありません。以下のファイルは、プロファイル作成プロセス中に作成済みです。

- `was_root/logs/manageprofiles.xml`
- `was_root/properties/fsdb.xml`
- `was_root/properties/profileRegistry.xml`

root 以外のユーザーが作成したプロファイルを自ら拡張する場合は、eXtreme Scale プロファイル・テンプレート内に位置する文書の権限を変更する必要があります。

**重要:** WebSphere Application Server の外側、スタンドアロン環境で WebSphere eXtreme Scale の非 root (非管理者) プロファイルを使用することもできます。ObjectGrid ディレクトリーの所有者を非 root プロファイルに変更してください。その後、その非 root プロファイルでログインして、通常の root (管理者) プロファイルの場合と同様に eXtreme Scale を操作できます。

---

## インストール後の最初のステップの実行

インストールが完了し、インストールの検査も終了したら、WebSphere eXtreme Scale の使用を開始して、データ・グリッドを作成できます。

### 手順

1. 保守を適用してインストール済み環境を更新します。

**詳細情報:** 239 ページの『eXtreme Scale サーバーの更新』

2. WebSphere eXtreme Scale をはじめて使用する場合は、「始めに」の情報を使用して製品の使用方法について学習できます。

**詳細情報:** 1 ページの『第 1 章 始めに』

3. 製品を構成します。プロパティや XML ファイルを作成して、データ・グリッド、サーバー、およびクライアントの構成を定義します。キャッシュまたはデータベース統合、REST データ・サービス、OSGi プラグインも構成できます。

詳細情報: 253 ページの『第 6 章 構成』

4. データ・グリッドにアクセスするアプリケーションを作成します。

詳細情報: アプリケーションの開発

5. 構成ファイルやデータ・グリッド・アプリケーションを使用してコンテナ・サーバーとカタログ・サーバーを開始し、管理します。

詳細情報: 451 ページの『第 7 章 管理』

6. 各種モニター・ツールを使用して構成のパフォーマンスをモニターします。

詳細情報: 505 ページの『第 8 章 モニター』

---

## 製品インストールのトラブルシューティング

IBM Installation Manager は多くの IBM ソフトウェア製品のための共通インストーラーです。このバージョンの WebSphere eXtreme Scale をインストールするには、このインストーラーを使用します。

### タスクの結果

ロギングおよびトレースに関する注記:

- Installation Manager を開いて「ファイル」>「ログの表示」をクリックすると、ログを簡単に表示できます。表内のログ・ファイルを個々に選択して、「ログ・ファイルを開く」アイコンをクリックすることにより、個々のログ・ファイルを開くことができます。
- ログは、Installation Manager のアプリケーション・データ・ロケーションの logs ディレクトリにあります。次に例を示します。

– **Windows** 管理インストール:

C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager

– **Windows** 非管理インストール:

C:\Documents and Settings\user\_name\Application Data\IBM\Installation Manager

– **UNIX** **Linux** 管理インストール:

/var/IBM/InstallationManager

– **UNIX** **Linux** 非管理インストール:

user\_home/var/ibm/InstallationManager

- メイン・ログ・ファイルは、タイム・スタンプが記されている XML ファイルで、logs ディレクトリにあります。これらのログ・ファイルは、任意の標準 Web ブラウザーを使用して表示できます。
- logs ディレクトリ内の log.properties ファイルは、Installation Manager が使用するロギングまたはトレースのレベルを指定します。WebSphere eXtreme Scale プラグインのトレースをオンにするには、例えば、以下のような内容の log.properties ファイルを作成します。

```
com.ibm.ws=DEBUG
com.ibm.cic.agent.core.Engine=DEBUG
global=DEBUG
```

必要に応じて Installation Manager を再始動します。Installation Manager によって、WebSphere eXtreme Scale プラグインのトレースが出力されます。

#### トラブルシューティングに関する注記:

- **UNIX** **Linux** デフォルトで、一部の HP-UX システムは、ホスト名の解決に DNS を使用しないように構成されます。その場合、Installation Manager は外部のリポジトリに接続できない場合があります。

リポジトリを ping することはできますが、nslookup は何も返しません。

システム管理者と相談して、DNS を使用するようにご使用のマシンを構成するか、またはリポジトリの IP アドレスを使用してください。

- 場合によっては、Installation Manager の既存のチェック・メカニズムを迂回する必要がある場合もあります。
  - 一部のネットワーク・ファイル・システムでは、ディスク・スペースが正しく報告されない場合があります。この場合、ディスク・スペース・チェックを迂回して、インストールを続行する必要があります。

ディスク・スペース・チェックを使用不可にするには、`IM_install_root/eclipse/configuration` の `config.ini` ファイルで以下のシステム・プロパティを指定して、Installation Manager を再始動します。

```
cic.override.disk.space=sizeunit
```

ここで `size` は正整数です。`unit` は、バイトの場合は空白、キロの場合は `k`、メガバイトの場合は `m`、ギガバイトの場合は `g` です。以下に例を示します。

```
cic.override.disk.space=120 (120 バイト)
cic.override.disk.space=130k (130 キロバイト)
cic.override.disk.space=140m (140 メガバイト)
cic.override.disk.space=150g (150 ギガバイト)
cic.override.disk.space=true
```

Installation Manager は `Long.MAX_VALUE` のディスク・スペース・サイズを報告します。使用可能なディスク・スペースが非常に大容量の場合は、表示されずに `N/A` が表示されます。

- オペレーティング・システムの前条件チェックを迂回するには、`disableOSPrereqChecking=true` を `IM_install_root/eclipse/configuration` の `config.ini` ファイルに追加して、Installation Manager を再始動します。

これらの迂回メソッドのいずれかを使用する必要がある場合は、Installation Manager チェック・メカニズムを迂回しないソリューションの開発における支援を受けるため、IBM サポートにお問い合わせください。

- Installation Manager の使用について詳しくは、IBM Installation Manager バージョン 1.5 インフォメーション・センターを参照してください。

Installation Manager の最新バージョンの詳細については、リリース情報を参照してください。リリース情報にアクセスするには、以下のタスクを実行します。

- **Windows** 「スタート」 > 「プログラム」 > 「IBM Installation Manager」 > 「リリース情報」 をクリックします。

- **UNIX** **Linux** **Installation Manager** がインストールされているディレクトリーにある文書サブディレクトリーに移動し、`readme.html` ファイルを開きます。
- 製品をインストールしようとしている時に致命的エラーが生じた場合は、以下の手順に従ってください。
  - IBM サポートが後で確認する必要がある場合があるので、現在の製品インストール・ディレクトリーのバックアップ・コピーを取ってください。
  - **Installation Manager** を使用して、製品のインストール・ロケーション (パッケージ・グループ) にインストールしたものをすべてアンインストールします。このときエラーが生じる可能性もありますが、無視しても支障ありません。
  - 製品のインストール・ディレクトリーに残っているものをすべて削除します。
  - **Installation Manager** を使用し、同じロケーションか新しいロケーションに製品を再インストールします。

バージョンおよびヒストリー情報に関する注記: **versionInfo** コマンドおよび **historyInfo** コマンドは、システム上で実行されるインストール、アンインストール、更新、および、ロールバックのすべてのアクティビティーに基づいて、バージョンとヒストリー情報を戻します。



---

## 第 5 章 WebSphere eXtreme Scale のアップグレードおよびマイグレーション



前のバージョンからバージョン 8.5 にマイグレーションできます。また、バージョン 8.5 に保守パッケージを適用できます。停止を回避するには、構成内の、更新を適用するサーバーの順序を検討する必要があります。

- バージョン 7.1.0.x インストール済み環境をアップグレードするには、『eXtreme Scale サーバーの更新』および 208 ページの『IBM Installation Manager の使用によるフィックスパックのインストール』を参照してください。
- バージョン 7.0.x インストール済み環境をアップグレードするには、『eXtreme Scale サーバーの更新』および 242 ページの『WebSphere eXtreme Scale バージョン 8.5 へのマイグレーション』を参照してください。

---

### eXtreme Scale サーバーの更新

保守を適用するか、新しいバージョンをインストールすることで、サービスを中断せずに WebSphere eXtreme Scale を新しいバージョンにアップグレードできます。

#### 始める前に

メジャー・バージョン・リリースまたは適用する保守のバイナリー・ファイルを手入している必要があります。使用可能なリリースおよび保守パッケージについての最新情報は、WebSphere eXtreme Scale の IBM サポート・ポータルから入手できます。

#### このタスクについて

サービスを中断しないでアップグレードするには、まずカタログ・サーバーをアップグレードし、次にコンテナ・サーバーをアップグレードし、最後にクライアント・サーバーをアップグレードする必要があります。

#### 手順

1. データ・グリッド内の各カタログ・サーバーに対して、次のステップを繰り返して、カタログ・サービス層をアップグレードします。どのコンテナ・サーバーやクライアントをアップグレードする前に、カタログ・サービス層をアップグレードします。個々のカタログ・サーバーは、バージョン互換性により相互運用が可能です。したがって、サービスを中断せずに、一度に 1 つのカタログ・サーバーに対してアップグレードを適用できます。
  - a. 次の正常クォーラム状況を確認します。以下のコマンドを実行します。

```
xsadmin -quorumStatus  
xscmd -c showQuorumStatus
```

この結果は、すべてのカタログ・サーバーが接続されたことを示します。

- b. 2 つのカタログ・サービス・ドメインの間でマルチマスター・レプリカ生成を使用している場合、カタログ・サーバーをアップグレードする間、2 つのカタログ・サービス・ドメイン間のリンクは除去してください。

```
xsadmin -ch host -p 1099 -dismissLink domain_name
xscmd -c dismissLink -cep host:2809 -fd domain_name
```

このコマンドの実行は一方のカタログ・サービス・ドメインからのみ必要で、これで、2つのカタログ・サービス・ドメイン間のリンクを除去できます。

- c. カタログ・サーバーの1つをシャットダウンします。**stop0gserver** コマンドまたは **xscmd -c teardown** コマンドを使用できます。あるいは、WebSphere Application Server 内でカタログ・サービスを実行しているアプリケーション・サーバーをシャットダウンします。カタログ・サーバーはどのような順序で停止してもかまいませんが、プライマリー・カタログ・サーバーを最後にシャットダウンすると、ターンオーバーが減少します。ログ・ファイルの中の CWOBJ8106 メッセージを見ると、どれがプライマリー・カタログ・サーバーかが分かります。通常の場合、カタログ・サーバーがシャットダウンされるときにクォーラムは維持されますが、ベスト・プラクティスは、各シャットダウンの後、**xscmd -c showQuorumStatus** コマンドを使用して、クォーラムを照会することです。

**xscmd -c teardown** コマンドを使用する場合、サーバー名をフィルタリングできます。**stop0gServer** コマンドには、並行して停止する正確なサーバー名またはサーバー名のリストを入力する必要があります。多数のサーバーの停止またはティアダウン・プロセスを並行して呼び出すのではなく、シャットダウン・プロセスをグループにすべきです。シャットダウンするサーバーをグループにすると、データ・グリッドは、その周辺の断片を移動することで、シャットダウンされるサーバーに反応できます。次のいずれかのコマンドを使用して、サーバーをシャットダウンできます。

**stop0gServer** コマンドまたは **xscmd -c teardown** コマンドに、停止するサーバーの特定のリストを指定できます。

```
stop0gServer <server_name>[,<server_name>]
xsadmin -teardown <server_name>[,<server_name>]
xscmd -c teardown -sl <server_name>[,<server_name>]
```

上記の例では、**stop0gServer** コマンドまたは **xscmd -c teardown** コマンドは同じシャットダウン・タスクを完了しています。しかし、**xscmd -c teardown** コマンドでは、停止するサーバーをフィルタリングできます。ゾーンまたはホスト名によるサーバーのフィルタリングについて詳しくは、466ページの『**xscmd** ユーティリティによるサーバーの正常停止』を参照してください。**teardown** コマンドは、一致したサーバーをフィルタリングして、選択されたサーバーが正しいかどうかを尋ねます。

- d. カタログ・サーバーに更新をインストールします。カタログ・サーバーを製品のメジャー・リリースにマイグレーションするか、保守パッケージを適用することができます。詳しくは、次のトピックを参照してください。
- バージョン 7.1.x インストール済み環境からマイグレーションする場合:  
242ページの『WebSphere eXtreme Scale バージョン 8.5 へのマイグレーション』
  - バージョン またはバージョン 8.5.x インストール済み環境へ:
- e. カタログ・サーバーを再始動します。

スタンドアロン環境を使用している場合、詳しくは、451 ページの『スタンドアロン・カタログ・サービスの開始』を参照してください。WebSphere Application Server 環境を使用している場合、詳しくは、467 ページの『WebSphere Application Server 環境でのサーバーの開始と停止』を参照してください。

カタログ・サーバーは、すべてのカタログ・サーバーが同じレベルに移行するまで、互換モードで実行されます。マイグレーションされていないサーバーでは新しい機能を使用できないため、互換モードは、たいいていは、メジャー・リリースのマイグレーションで適用されます。カタログ・サーバーを互換モードで実行できる時間の長さについて制限はありませんが、ベスト・プラクティスは、できるだけ速やかにすべてのカタログ・サーバーを同じレベルにマイグレーションすることです。

- f. 構成内の残りのカタログ・サーバーに対して更新を適用します。
2. データ・グリッド内の各コンテナ・サーバーに対して、次のステップを繰り返して、コンテナ・サーバーをアップグレードします。コンテナ・サーバーは任意の順序でアップグレードできます。しかし、更新の中の新しい機能を使用している場合は、サーバーを最初に更新し、次にクライアントを更新するように考慮してください。

- a. アップグレードするコンテナ・サーバーを停止します。 **stop0gserver** コマンドまたは **teardown** コマンドを使用して、コンテナ・サーバー層をグループで停止できます。ティアダウン操作のバッチ処理およびサーバーの始動操作の並列実行により、配置メカニズムは断片を大きなグループで移動できます。

```
xsadmin -teardown -fz DefaultZone
```

```
xscmd -c teardown -z DefaultZone
```

```
Connecting to Catalog service at localhost:1099
```

サーバーのティアダウンのためのフィルター・オプションを処理しています

次のサーバーがティアダウンされます。

```
container00  
container01  
container02  
container03  
container04
```

リストされたサーバーをティアダウンしますか? (Y/N)

- b. コンテナ・サーバーに更新をインストールします。コンテナ・サーバーを製品のメジャー・リリースにマイグレーションするか、保守パッケージを適用することができます。詳しくは、次のトピックを参照してください。
- バージョン 7.1.x インストール済み環境からマイグレーションする場合:  
242 ページの『WebSphere eXtreme Scale バージョン 8.5 へのマイグレーション』
  - バージョン またはバージョン 8.5.x インストール済み環境へ:
- c. コンテナ・サーバーを再始動します。
- d. 構成の残りのコンテナ・サーバーをアップグレードします。

- マルチマスター・レプリカ生成を使用している場合、カタログ・サービス・ドメインに再接続します。カタログ・サービス・ドメインに再接続するには、**xscmd -c establishLink** コマンドを使用します。

```
xsadmin -ch host -p 1099 -establishLink dname fdHostA:2809,fdHostB:2809
```

```
xscmd -c establishLink -cep host:2809 -fd dname -fe fdHostA:2809,fdHostB:2809
```

- 8.5+** すべてのサーバーが WebSphere eXtreme Scale の新バージョンを使用しているか確認するには、**xscmd -c showinfo** コマンドを実行します。

```
xscmd -c showinfo
```

## 次のタスク

また、このステップは、旧バージョンに戻す場合や保守パッケージをアンインストール場合にも使用できます。ただし、マルチマスター・レプリカ生成を使用しているときにバージョン 7.1.0 に戻すと、リンクを再確立するときに、両方向レプリカ生成は正しく機能しないことがあります。この場合、両方のカタログ・サービス・ドメインを再始動し、**establishLink** コマンドを使用してカタログ・サービス・ドメインを再リンクします。

---

## WebSphere eXtreme Scale バージョン 8.5 へのマイグレーション

WebSphere eXtreme Scale インストーラーでは、前のインストール済み環境をアップグレードすることも変更することもできません。新しいバージョンをインストールする前に前のバージョンをアンインストールする必要があります。構成ファイルは後方互換性があるため、マイグレーションする必要はありません。ただし、製品に付属のスクリプト・ファイルのいずれかを変更した場合には、更新したスクリプト・ファイルにその変更を再適用する必要があります。

### 始める前に

ご使用のシステムが、マイグレーションおよびインストールしようとしている製品バージョンの最小限の要件を満たしていることを確認してください。詳細については、59 ページの『ハードウェアおよびソフトウェア要件』を参照してください。

### このタスクについて

変更済みの製品スクリプト・ファイルを /bin ディレクトリーにある新規の製品スクリプト・ファイルとマージして、変更の保守を行います。

**ヒント:** 製品にインストールされているスクリプト・ファイルを変更しなかった場合は、以下のマイグレーション・ステップを実行する必要はありません。代わりに、以前のバージョンをアンインストールしてから同じディレクトリーに新規バージョンをインストールすることで、バージョン 8.5 にアップグレードできます。

### 手順

- eXtreme Scale を使用しているすべてのプロセスを停止します。
  - スタンドアロンの eXtreme Scale 環境で実行されているすべてのプロセスを停止します。詳しくは、463 ページの『スタンドアロン・サーバーの停止』を参照してください。

- WebSphere Application Server または WebSphere Application Server Network Deployment 環境で実行しているすべてのプロセスを停止するには、 コマンド行ユーティリティを参照してください。
- 2. 現行インストール・ディレクトリーから変更済みのすべてのスクリプトを一時ディレクトリーに保存します。
- 3. 製品をアンインストールします。 詳しくは、 220 ページの『IBM Installation Manager の使用による 製品 のアンインストール』を参照してください。
- 4. eXtreme Scale バージョン 8.5 をインストールします。 詳しくは、 163 ページの『第 4 章 インストール』を参照してください。
- 5. 一時ディレクトリーにあるファイルから、 /bin ディレクトリーにある新規の製品スクリプト・ファイルに必要な変更をマージします。
- 6. すべての eXtreme Scale プロセスを開始して、製品の使用を開始します。 詳しくは、 451 ページの『第 7 章 管理』を参照してください。

---

## WebSphere Application Server 上の WebSphere eXtreme Scale の更新

WebSphere Application Server を新しいバージョンにマイグレーションするとき、WebSphere eXtreme Scale の構成を新しい WebSphere Application Server のインストール済み環境にマイグレーションすることもできます。

### 始める前に

- WebSphere eXtreme Scale バージョン 7 と WebSphere eXtreme Scale バージョン 8 をいずれも同じサーバー上にインストールすることを前提とします。
- WebSphere Application Server バージョン 7 を WebSphere Application Server バージョン 8 にマイグレーションします。詳しくは、製品構成のマイグレーションを参照してください。
- WebSphere eXtreme Scale バージョン 8.5 をお使いの WebSphere Application Server バージョン 8 のインストール済み環境にインストールします。詳しくは、205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。 WebSphere eXtreme Scale のマイグレーション・スクリプトはすべて、WebSphere eXtreme Scale バージョン 8.5 以降から実行する必要があります。例えば、バージョン 7.x からバージョン 8.5 にマイグレーションする場合、マイグレーション用のスクリプトを `<WXS_v8_install_root>/bin` ディレクトリーから実行します。

### このタスクについて

WebSphere eXtreme Scale を統合している WebSphere Application Server の新しいバージョンをインストールするときは、まず、通常のプロセスで WebSphere Application Server をアップグレードします。次に、その新しいインストール済み環境に WebSphere eXtreme Scale の新しいバージョンをインストールします。その後、**xsmigration** スクリプトを使用して WebSphere eXtreme Scale 構成情報を新しい WebSphere Application Server インストール済み環境に移動できます。

## 手順

1. デプロイメント・マネージャー関連の構成をバージョン 7 からバージョン 8 にマイグレーションします。
  - a. WebSphere Application Server バックアップ・スクリプトを実行します。詳しくは、WASPreUpgrade コマンドを参照してください。
  - b. デプロイメント・マネージャーを停止します。
  - c. WebSphere eXtreme Scale 構成内のデプロイメント・マネージャー・サーバーにアクセスして、マイグレーション・スクリプトを実行します。
    - 1) ディレクトリーを <WXS\_v8\_install\_root>/bin に移動します。
    - 2) 以下のコマンドを実行します。

```
xsmigration.bat|sh -targetwashome <WAS8x_HOME>  
-sourcewashome <WAS7x_HOME> -targetprofilepath <WAS8x_DmgrProfile>  
-sourceprofilepath <WAS7x_DmgrProfile>
```

説明:

- <WAS8x\_HOME> は、WebSphere Application Server バージョン 8.x のインストール済み環境のルート・ロケーションです。例:  
/opt/IBM/WebSphere8
  - <WAS7x\_HOME> は、WebSphere Application Server バージョン 7.x のインストール済み環境のルート・ロケーションです。例:  
/opt/IBM/WebSphere7
  - <WAS8x\_DmgrProfile> は、WebSphere Application Server バージョン 8.x のデプロイメント・マネージャーのプロファイルが置かれているロケーションです。例: /opt/IBM/WebSphere8/profiles/DMgr01
  - <WAS7x\_DmgrProfile> は、WebSphere Application Server バージョン 7.x のデプロイメント・マネージャーのプロファイルが置かれているロケーションです。例: /opt/IBM/WebSphere7/profiles/DMgr01
2. アプリケーション・サーバー関連の構成をバージョン 7 からバージョン 8 にマイグレーションします。
    - a. ディレクトリーを <WXS\_v8\_install\_root>/bin に移動します。
    - b. 以下のコマンドを実行します。

```
xsmigration.bat|sh -targetwashome <WAS8x_HOME>  
-sourcewashome <WAS7x_HOME> -targetprofilepath <WAS8x_AppServerProfile>  
-sourceprofilepath <WAS7x_AppServerProfile>
```

各部の意味は、次のとおりです。

- <WAS8x\_HOME> は、WebSphere Application Server バージョン 8.x のインストール済み環境のルート・ロケーションです。例: /opt/IBM/WebSphere8
- <WAS7x\_HOME> は、WebSphere Application Server バージョン 7.x のインストール済み環境のルート・ロケーションです。例: /opt/IBM/WebSphere7
- <WAS8x\_AppServerProfile> は、WebSphere Application Server バージョン 8.x のアプリケーション・サーバー・プロファイルが置かれているロケーションです。例: /opt/IBM/WebSphere8/profiles/AppServer01
- <WAS7x\_AppServerProfile> は、WebSphere Application Server バージョン 7.x のアプリケーション・サーバー・プロファイルが置かれているロケーションです。例: /opt/IBM/WebSphere7/profiles/AppServer01



3. WebSphere Application Server バージョン 8 のデプロイメント・マネージャーを再始動して、すべての管理対象ノードを同期化します。

## xsadmin ツールから xscmd ツールへのマイグレーション

これまでのリリースでは、**xsadmin** ツールは環境の状態をモニターするサンプルのコマンド行ユーティリティーでした。**xscmd** ツールは、管理およびモニター用の、正式にサポートされるコマンド行ツールとして導入されました。これまで **xsadmin** ツールを使用していた場合は、新しい **xscmd** ツールにコマンドをマイグレーションすることを検討してください。

### xsadmin および xscmd コマンド同等

表 9. **xsadmin** ユーティリティーの引数と、**xscmd** 同等コマンド：一部の **xscmd** コマンドは、短い形式と長い形式を持ちます。短い形式のコマンドは、1 つのダッシュ (-) を持ち、長い形式のコマンドは、2 つのダッシュ (--) を持ちます。どちらの形式も区別なく使用できます。

xsadmin コマンド行引数	xscmd 同等コマンド	xscmd コマンド・パラメーター
<b>-bp</b>	<ul style="list-style-type: none"> <li>• <b>-cep</b> <i>hostname:listener_port</i></li> <li>• <b>--catalogEndpoint</b> <i>hostname:listener_port</i></li> </ul>	n/a
<b>-ch</b>	<ul style="list-style-type: none"> <li>• <b>-cep</b> <i>hostname:listener_port</i></li> <li>• <b>--catalogEndpoint</b> <i>hostname:listener_port</i></li> </ul>	n/a
<b>-clear</b>	<b>-c clearGrid</b>	<b>-g, -ms, -v, -m, (-cep)</b>
<b>-containers</b>	<ul style="list-style-type: none"> <li>• <b>-c listCoreGroups</b></li> <li>• <b>-c listCoreGroupMembers -cg</b> <i>core_group</i></li> </ul>	<b>-e, -i, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms</b>
<b>-continuous</b>	n/a	n/a
<b>-coregroups</b>	<ul style="list-style-type: none"> <li>• <b>-c listCoreGroups</b></li> <li>• <b>-c listCoreGroupMembers -cg</b> <i>core_group</i></li> </ul>	n/a
<b>-dismissLink</b> <i>&lt;catalog_service_domain&gt;</i>	<b>-c dismissLink</b>	<ul style="list-style-type: none"> <li>• <b>-fd</b> <i>&lt;foreignCatalogServiceDomain&gt;</i></li> <li>• <b>--foreignCatalogServiceDomain</b> <i>&lt;foreignCatalogServiceDomain&gt;</i></li> </ul>
<b>-dmgr</b>	n/a - この引数は、 <b>xscmd</b> で自動的に判別される	n/a
<b>-empties</b>	新規コマンドに固有の引数	n/a
<b>-establishLink</b> <i>&lt;foreign_domain_name&gt;</i> <i>&lt;host1:port1,host2:port2...&gt;</i>	<b>-c establishLink</b>	<ul style="list-style-type: none"> <li>• <b>-fd</b> <i>&lt;foreignCatalogServiceDomain&gt;</i></li> <li>• <b>-fe</b> <i>&lt;host1:port1,host2:port2...&gt;</i></li> <li>• <b>--foreignCatalogServiceDomain</b> <i>&lt;foreignCatalogServiceDomain&gt;</i></li> <li>• <b>-foreignEndPoints</b> <i>&lt;host1:port1,host2:port2...&gt;</i></li> </ul>
<b>-fc</b>	<ul style="list-style-type: none"> <li>• <b>-ct</b></li> <li>• <b>--container</b></li> </ul>	n/a

表9. **xsadmin** ユーティリティの引数と、**xscmd** 同等コマンド (続き): 一部の **xscmd** コマンドは、短い形式と長い形式を持ちます。短い形式のコマンドは、1 つのダッシュ (-) を持ち、長い形式のコマンドは、2 つのダッシュ (-- ) を持ちます。どちらの形式も区別なく使用できます。

xsadmin コマンド行引数	xscmd 同等コマンド	xscmd コマンド・パラメーター
<b>-fh</b>	<ul style="list-style-type: none"> <li>• <b>-hf</b></li> <li>• <b>--hostFilter</b></li> </ul>	n/a
<b>-fm</b>	<ul style="list-style-type: none"> <li>• <b>-m</b></li> <li>• <b>--map</b></li> </ul>	n/a
<b>-fnp</b>	<ul style="list-style-type: none"> <li>• <b>-snp</b></li> <li>• <b>--serversWithNoPrimaries</b></li> </ul>	n/a
<b>-fp</b>	<ul style="list-style-type: none"> <li>• <b>-p</b></li> <li>• <b>--partitionId</b></li> </ul>	n/a
<b>-fs</b>	<ul style="list-style-type: none"> <li>• <b>-s</b></li> <li>• <b>--server</b></li> </ul>	n/a
<b>-fst</b>	<ul style="list-style-type: none"> <li>• <b>-st &lt;shard_type&gt;</b></li> <li>• <b>--shardType &lt;shard_type&gt;</b></li> </ul> 断片値: P=primary A=asyncReplica S=syncReplica	n/a
<b>-fz</b>	<ul style="list-style-type: none"> <li>• <b>-z</b></li> <li>• <b>--zone</b></li> </ul>	n/a
<b>-force</b>	新規コマンドに固有の引数	
<b>-g</b>	<ul style="list-style-type: none"> <li>• <b>-g</b></li> <li>• <b>--objectGrid</b></li> </ul>	n/a
<b>-getstatsspec</b>	<b>-c getStatsSpec</b>	n/a
<b>-getTraceSpec</b>	<b>-c getTraceSpec</b>	n/a
<b>-h</b>	特定のコマンド名を付けても付けなくても <b>help</b> を実行できる <ul style="list-style-type: none"> <li>• <b>-h</b></li> <li>• <b>--help</b></li> <li>• <b>-h &lt;command_name&gt;</b></li> <li>• <b>--help &lt;command_name&gt;</b></li> </ul>	n/a
<b>-hosts</b>	<b>-c listHosts</b>	<b>-g, -ms, -st, -c, -s, -hf, -z</b>
<b>-jmxUrl</b>	<ul style="list-style-type: none"> <li>• <b>-cep hostname:listener_port</b></li> <li>• <b>--catalogEndpoint hostname:listener_port</b></li> </ul>	n/a
<b>-l</b>	<b>-c listObjectGridNames</b>	n/a
<b>-m</b>	<ul style="list-style-type: none"> <li>• <b>-ms</b></li> <li>• <b>--mapSet</b></li> </ul>	n/a
<b>-mapsizes</b>	<b>-c showMapSizes</b>	<b>-g, -ms, -i, [-ct, -z, -s, -hf, sht [P,A,S], -p]</b>
<b>-mbeanservers</b>	<b>-c listAllJMXAddresses</b>	n/a

表9. **xsadmin** ユーティリティーの引数と、**xscmd** 同等コマンド (続き): 一部の **xscmd** コマンドは、短い形式と長い形式を持ちます。短い形式のコマンドは、1 つのダッシュ (-) を持ち、長い形式のコマンドは、2 つのダッシュ (-- ) を持ちます。どちらの形式も区別なく使用できます。

xsadmin コマンド行引数	xscmd 同等コマンド	xscmd コマンド・パラメーター
<b>-overridequorum</b>	<b>-c overrideQuorum</b>	n/a
<b>-password</b>	<ul style="list-style-type: none"> <li>• <b>-pwd</b></li> <li>• <b>--password</b></li> </ul>	n/a
<b>-p</b>	<ul style="list-style-type: none"> <li>• <b>-cep hostname:listener_port</b></li> <li>• <b>--catalogEndpoint hostname:listener_port</b></li> </ul>	n/a
<b>-placementStatus</b>	<b>-c placementServiceStatus</b>	<b>-g, -ms</b>
<b>-primaries</b>	<b>-c showPlacement -sf P</b>	<b>-e, -i, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms</b>
<b>-profile</b>	<p>現行セキュリティ設定をセキュリティ・プロファイルとして保存する場合:</p> <ul style="list-style-type: none"> <li>• <b>-ssp profile_name</b></li> <li>• <b>--saveSecProfile profile_name</b></li> </ul> <p>指定されたセキュリティ・プロファイルを使用する場合:</p> <ul style="list-style-type: none"> <li>• <b>-sp profile_name</b></li> <li>• <b>--securityProfile profile_name</b></li> </ul>	
<b>-quorumstatus</b>	<b>-c showQuorumStatus</b>	n/a
<b>-releaseShard</b> <container_server_name> <objectgrid_name> <map_set_name> <partition_name>	<b>-c releaseShard</b>	<b>-c, -g, -ms, -p</b>
<b>-reserved</b>	<ul style="list-style-type: none"> <li>• <b>-sf R</b></li> <li>• <b>--shardFilter R</b></li> </ul>	n/a
<b>-reserveShard</b> <container_server_name> <objectgrid_name> <map_set_name> <partition_name>	<b>-c reserveShard</b>	<b>-c, -g, -ms, -p</b>
<b>-resumeBalancing</b> <objectgrid_name> <map_set_name>	<b>-c resumeBalancing</b>	<b>-g, -ms</b>
<b>-revisions</b>	<b>-c revisions</b>	<b>-s, -p, -g, -m</b>
<b>-routetable</b>	<b>-c routetable</b>	<b>-z, -hf, -p, -g, -ms</b>
<b>-settracespec</b> <trace_string>	<b>-c setTraceSpec</b>	<b>-spec</b> <trace_string>
<b>-swapShardWithPrimary</b> <container_server_name> <objectgrid_name> <map_set_name> <partition_name>	<b>-c swapShardWithPrimary</b>	<b>-c -g, -ms, -p</b>
<b>-setstatsspec</b> <stats_spec>	<b>-c setStatsSpec</b>	<b>-spec</b> <stats_spec>

表 9. xsadmin ユーティリティーの引数と、xscmd 同等コマンド (続き): 一部の xscmd コマンドは、短い形式と長い形式を持ちます。短い形式のコマンドは、1 つのダッシュ (-) を持ち、長い形式のコマンドは、2 つのダッシュ (-- ) を持ちます。どちらの形式も区別なく使用できます。

xsadmin コマンド行引数	xscmd 同等コマンド	xscmd コマンド・パラメーター
<b>-suspendBalancing</b> <objectgrid_name> <map_set_name>	<b>-c suspendBalancing</b>	<b>-g, -ms</b>
<b>-ssl</b>	<ul style="list-style-type: none"> <li>• <b>-ssl</b></li> <li>• <b>--enableSSL</b></li> </ul>	n/a
<b>-teardown</b>	<b>-c teardown</b>	<b>-f, , -st, -snp, -c, -s, -p, -hf, -z, -g, -ms, -m</b>
<b>-triggerPlacement</b>	<b>-c triggerPlacement</b>	<b>-g, -ms</b>
<b>-trustPass</b>	<ul style="list-style-type: none"> <li>• <b>-tsp</b></li> <li>• <b>--trustStorePassword</b></li> </ul>	n/a
<b>-trustPath</b>	<ul style="list-style-type: none"> <li>• <b>-ts</b></li> <li>• <b>--trustStore</b></li> </ul>	n/a
<b>-trustType</b>	<ul style="list-style-type: none"> <li>• <b>-tst</b></li> <li>• <b>--trustStoreType</b></li> </ul>	n/a
<b>-unassigned</b>	<b>-c showPlacement -sf U</b>	<b>-e, -i, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms</b>
<b>-username</b>	<ul style="list-style-type: none"> <li>• <b>-user</b></li> <li>• <b>--username</b></li> </ul>	n/a
<b>-v</b>	<ul style="list-style-type: none"> <li>• <b>-v</b></li> <li>• <b>--verbose</b></li> </ul>	n/a
<b>-xml</b>	<b>-c showPlacement</b>	n/a

## 推奨されないプロパティおよび API

次に示すプロパティと API のリストは、指定されたリリースでは非推奨になりました。推奨されるマイグレーション・アクションを使用して、構成の更新方法を決定してください。

### 8.5+

#### バージョン 8.5 での推奨されない項目

表 10. 推奨されないプロパティおよび API

非推奨機能	推奨されるマイグレーション・アクション
<b>WebSphereTransactionCallback</b> このプラグインは、WebSphere Application Server 環境で実行されるエンタープライズ・アプリケーションを使用してデータ・グリッド・トランザクションを管理するために使用されていました。	<b>8.5+</b> WebSphereTransactionCallback インターフェースは、Java Transaction API (JTA) トランザクション管理を可能にする WebSphere eXtreme Scale リソース・アダプターで置き換えられました。このリソース・アダプターは、WebSphere Application Server または他の Java Platform, Enterprise Edition (Java EE) アプリケーション・サーバーにインストールすることができます。WebSphereTransactionCallback プラグインは登録済みの JTA API ではありません。したがって、このプラグインは、コミットが失敗したときに JTA トランザクションをロールバックするようには設計されていません。

## バージョン 7.1.1 での推奨されない項目

表 11. 推奨されないプロパティおよび API

非推奨機能	推奨されるマイグレーション・アクション
<b>com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener クラス</b> このクラスは、正常に実行された ObjectGrid トランザクション・コミット・プロセスを、ObjectGrid 名に基づいて、同じ ObjectGrid インスタンスをホストする他の WebSphere Application Server に伝搬するために使用されました。	TranPropListener インターフェースは、ObjectGridEventListener インターフェースの JMS ベース実装である JMSObjectGridEventListener インターフェースに取り替えられました。これは、クライアント側のニア・キャッシュ無効化、およびピアツーピア・レプリカ生成をサポートします。
<b>com.ibm.websphere.objectgrid.plugins.OptimisticCallback クラス</b> このクラスは、マップの値としてオプティミスティック比較演算を提供するために使用されました。	OptimisticCallback プラグインは、ValueDataSerializer.Versionable インターフェースに取り替えられました。このインターフェースは、COPY_TO_BYTES コピー・モードで DataSerializer プラグインを使用するとき、または EntityManager API で @Version アノテーションを使用するときに実装できます。詳しくは、API 資料を参照してください。
<b>com.ibm.websphere.objectgrid.plugins.NoVersioningOptimisticCallback プラグイン</b> このプラグインは、バージョン・チェックなしでオプティミスティック・ロックを行うために使用されました。この組み込み OptimisticCallback ハンドラーがあるにもかかわらず、ローダーがバージョン・チェックを行いました。読み取り時にコミット済みデータが常に返されるようにするためにオプティミスティック・ロックが使用されました。	NoVersioningOptimisticCallback インターフェースは、OptimisticCallback インターフェースを拡張します。したがって、トランザクション分離をデフォルトの READ_COMMITTED 以下に設定してベシミスティック・ロック・ストラテジーを使用します。詳しくは、ロック・パフォーマンスのチューニングを参照してください。
<b>com.ibm.websphere.objectgrid.plugins.ObjectTransformer クラス</b> このプラグインは、オブジェクトのシリアライズ、デシリアライズ、およびキャッシュへのコピーを行うために使用されました。	ObjectTransformer インターフェースは、DataSerializer プラグインで置換されました。これを使用して、既存の製品 API がデータと効率的に対話できるように WebSphere eXtreme Scale 内の任意のデータを効率的に格納できます。
<b>com.ibm.websphere.objectgrid.BackingMap.setMapEventListeners メソッド</b> このメソッドは、MapEventListener オブジェクトのリストを設定するために使用されました。	addMapEventListener(EventListener) メソッドを使用してバックアップ・マップにイベント・リスナーを追加するか、removeMapEventListener(EventListener) メソッドを使用してバックアップ・マップからイベント・リスナーを除去します。
<b>com.ibm.websphere.objectgrid.ObjectGrid.setEventListeners メソッド</b> このメソッドは、ObjectGridEventListener オブジェクトの現行リストを上書きし、それを、ObjectGridEventListeners オブジェクトの提供されたリストで取り替えるために使用されました。	addEventListener(EventListener) メソッドを使用してデータ・グリッドにイベント・リスナーまたはライフサイクル・リスナーを追加するか、removeEventListener(EventListener) メソッドを使用してデータ・グリッドからイベント・リスナーまたはライフサイクル・リスナーを除去します。

## バージョン 7.1.1 で安定化されたフィーチャー

IBM は、安定化されたとしてリストされているフィーチャーを、製品の以降のリリースで非推奨にしたり、除去したりする予定はありません。ただし、将来の投資は、代替機能に重点が置かれます。安定化された機能を使用する既存のアプリケーションおよびスクリプトを変更する必要はありませんが、新規アプリケーションに対しては戦略的な代替機能を使用するように検討してください。

表 12. 推奨されないプロパティおよび API

安定化されたフィーチャー	推奨されるマイグレーション・アクション
<b>xsadmin xsadmin</b> ユーティリティは、デプロイメントのカスタム・ユーティリティをどのように作成できるかを示すサンプルとして提供されています。	<b>xscmd</b> ユーティリティを使用して、マルチマスター・レプリカ生成リンクの確立、クォラムのオーバーライド、ティアダウン・コマンドを使用したサーバー・グループの停止などの管理用タスクを環境内で実行します。

## バージョン 7.1 での推奨されない項目

表 13. 推奨されないプロパティおよび API

非推奨機能	推奨されるマイグレーション・アクション
<b>catalog.services.cluster</b> セルおよびサーバー・プロパティ: このカスタム・プロパティは、WebSphere Application Server 構成におけるカタログ・サーバーのグループを定義するのに使用されました。	このカスタム・プロパティは、バージョン 7.1 リリース以降、推奨されなくなりました。  WebSphere Application Server 管理コンソールでカタログ・サービス・ドメインを作成します。これにより、カスタム・プロパティを使用する場合と同じ構成が作成されます。
<b>CoreGroupServicesMBean</b> MBean およびインターフェース	この MBean は、バージョン 7.1 リリース以降、推奨されなくなりました。  代わりに、CatalogServiceManagementMBean を使用します。
<b>ServerMBean.updateTraceSpec()</b> MBean 操作	この操作は、バージョン 7.1 リリース以降、推奨されなくなりました。  代わりに、DynamicServerMBean の TraceSpec 属性を使用します。
<b>CoreGroupServicesMBean</b> MBean	この MBean は、バージョン 7.1 リリース以降、推奨されなくなりました。  代わりに、CatalogServiceManagementMbean MBean を使用します。
<b>ServiceUnavailableException</b> 例外	この例外は、バージョン 7.1 リリース以降、推奨されなくなりました。  代わりに TargetNotAvailableException 例外を使用します。
	WPF の機能は、また、WebSphere eXtreme Scale でも実現することができます。
<b>StreamQuery:</b> ObjectGrid マップに保管されている伝送中のデータに対して連続的に行う照会。	なし
<b>静的グリッド構成:</b> クラスタ・デプロイメント XML ファイルを使用する静的なクラスタ・ベースのトポロジー。	大容量データ・グリッドを管理するために改善された動的デプロイメント・トポロジーに置き換えられています。



表 13. 推奨されないプロパティおよび API (続き)

非推奨機能	推奨されるマイグレーション・アクション
<p>非推奨システム・プロパティ: サーバーおよびクライアントのプロパティ・ファイルを指定するシステム・プロパティは推奨されていません。</p>	<p>これらの引数は、まだ使用できますが、ご使用のシステム・プロパティを新しい値に変更してください。</p> <p><b>-Dcom.ibm.websphere.objectgrid.CatalogServerProperties</b> このプロパティは、WebSphere eXtreme Scale バージョン 7.0 で使用すべきではありません。 <b>-Dobjectgrid.server.props</b> プロパティを使用してください。</p> <p><b>-Dcom.ibm.websphere.objectgrid.ClientProperties</b> このプロパティは、WebSphere eXtreme Scale バージョン 7.0 で使用すべきではありません。 <b>-Dobjectgrid.client.props</b> プロパティを使用してください。</p> <p><b>-Dobjectgrid.security.server.prop</b> このプロパティは、WebSphere eXtreme Scale バージョン 6.1.0.3 で使用すべきではありません。 <b>-Dobjectgrid.server.prop</b> プロパティを使用してください。</p> <p><b>-serverSecurityFile</b> この引数は、WebSphere eXtreme Scale バージョン 6.1.0.3 で使用すべきではありません。このオプションは、startOgServer スクリプトに渡されます。 <b>-serverProps</b> 引数を使用してください。</p>

## 削除されたプロパティおよび API

構成を WebSphere eXtreme Scale の旧リリースからマイグレーションすると、このリリースおよび旧リリースからフィーチャーがいくつか削除されることがあります。推奨されるマイグレーション・アクションを使用して、構成の更新方法を決定してください。

フィーチャーが非推奨フィーチャーに非推奨としてリストされていると、IBM® は製品の後継リリースでこの機能を削除する可能性があります。将来の投資は、「非推奨フィーチャー」の『推奨マイグレーション・アクション』にリストされている戦略的機能に重点が置かれます。一般に、フィーチャーが非推奨となったリリース以降、少なくとも 2 つのメジャー・リリースが提供されるか、または丸 3 年が経過するまで (いずれか期間が長い方)、そのフィーチャーは除去されません。それよりも短時間でフィーチャーを除去しなければならない場合がまれにあります。そのようなケースについては、非推奨フィーチャーの非推奨フィーチャーの説明に明示されています。

以下の情報は、削除されたフィーチャー、API、スクリプティング・インターフェース、ツール、および公開済み構成データについて説明しています。推奨される置換があれば、示されます。

### 8.5+

## バージョン 8.5 で削除された項目

表 14. 削除されたプロパティおよび API

削除された項目	推奨されるマイグレーション・アクション
<p><b>キーワードのサポート:</b> キーワードは、キャッシュ・エントリに適用され、後で ObjectMap API メソッドを使用して照会できるストリング・タグです。</p>	<p>索引または照会関数を使用して、特定の属性を持つオブジェクトを取得してください。</p>
<p><b>MapAuthorization インターフェース:</b> このプラグインは、Subject オブジェクトによって表されたプリンシパルにアクセスする目的で ObjectMap および JavaMap を許可するために使用されていました。</p>	<p>ObjectGridAuthorization を使用して、許可実装に接続してください。 ObjectGridAuthorization を使用して、ObjectGrid、ObjectMap、および JavaMap アクセスに対する権限を許可することができます。</p>
<p><b>WebSphere 区画化機能 (WPF):</b> 区画化機能は、Java EE アプリケーションによる非対称クラスタリングのサポートを可能にする一連のプログラミング API です。</p>	<p>WebSphere eXtreme Scale を使用して区画化を構成することができます。</p>
<p><b>StreamQuery:</b> ObjectGrid マップに保管されている伝送中のデータに対して連続的に行う照会。</p>	<p>なし</p>

---

## 第 6 章 構成



スタンドアロン環境で実行する WebSphere eXtreme Scale を構成することも、WebSphere Application Server または WebSphere Application Server Network Deployment を使用する環境で実行する eXtreme Scale を構成することもできます。データ・グリッドのサーバー・サイドの構成変更を WebSphere eXtreme Scale デプロイメントに反映するには、動的に適用するのではなく、プロセスを再始動して変更を有効にする必要があります。一方、クライアント・サイドでは、既存のクライアント・インスタンスの構成設定は変更できませんが、XML ファイルを使用するか、またはプログラムで、新しいクライアントに必要な設定で作成できます。クライアントの作成時には、現行のサーバー構成からのデフォルト設定をオーバーライド可能です。

---

### 構成方式

この製品のほとんどの部分は、XML ファイルとプロパティ・ファイルで構成できます。アプリケーション・プログラミング・インターフェース、システム・プログラミング・インターフェース、プラグイン、Managed Bean などのプログラマチックな方式も使用できます。

#### このタスクについて

次のファイルを使用して、基本的な構成を作成します。

##### サーバー・プロパティ・ファイル

サーバー・プロパティ・ファイルを使用して、トレース、ロギング、セキュリティ、ポートなど、カタログ・サーバーとコンテナ・サーバーの設定を定義します。サーバー・プロパティ・ファイルは **startOgServer** スクリプトに渡すか、クラスパス内に置くか、システム・プロパティを使用して定義できます。

##### クライアント・プロパティ・ファイル

クライアント・プロパティ・ファイルを使用して、ポートやセキュリティ設定など、クライアントのプロパティを設定します。使用するクライアント・プロパティ・ファイルを指定するには、システム・プロパティを使用するか、ファイルをクラスパス内に置くか、または `ClientClusterContext.getClientProperties` メソッドを使用できます。

##### ObjectGrid 記述子 XML ファイル

ObjectGrid 記述子 XML ファイルには、データ・グリッドとマップの構成を記述します。スタンドアロン構成の場合は、**startOgServer** スクリプトで使用するファイルを指定し、WebSphere Application Server 構成の場合はファイルをアプリケーション・モジュールに追加します。

##### デプロイメント・ポリシー記述子 XML ファイル

デプロイメント・ポリシー XML ファイルは、構成内のさまざまなコンテナ・サーバーの断片およびデータの配置を制御します。スタンドアロン構

成の場合は、**startOgServer** スクリプトで使用するファイルを指定し、**WebSphere Application Server** 構成の場合はファイルをアプリケーション・モジュールに追加します。

## 運用チェックリスト

この運用チェックリストを使用して、WebSphere eXtreme Scale のデプロイ用に環境を準備してください。

表 15. 運用チェックリスト

チェックリスト項目	詳細
<p>AIX<sup>®</sup> を使用している場合、以下のオペレーティング・システムの設定を調整してください。</p> <p><b>TCP_KEEPINTVL</b></p> <p>TCP_KEEPINTVL 設定は、ネットワーク障害の検出を可能にする、ソケットのキープアライブ・プロトコルの一部です。このプロパティは、接続を検証するために送信されるパケット間の間隔を指定します。WebSphere eXtreme Scale を指定している場合、この値は 10 に設定します。現行設定を確認するには、次のコマンドを実行します。</p> <pre># no -o tcp_keepintvl</pre> <p>現行設定を変更するには、次のコマンドを実行します。</p> <pre># no -o tcp_keepintvl=10</pre> <p>TCP_KEEPINTVL 設定は、0.5 秒単位で設定します。</p> <p><b>TCP_KEEPINIT</b></p> <p>TCP_KEEPINIT 設定は、ネットワーク障害の検出を可能にする、ソケットのキープアライブ・プロトコルの一部です。このプロパティは、TCP 接続の初期タイムアウト値を指定します。WebSphere eXtreme Scale を使用している場合、この値は 40 に設定します。現行設定を確認するには、次のコマンドを実行します。</p> <pre># no -o tcp_keepinit</pre> <p>現行設定を変更するには、次のコマンドを実行します。</p> <pre># no -o tcp_keepinit=40</pre> <p>TCP_KEEPINIT 設定は、0.5 秒単位で設定します。</p>	<ul style="list-style-type: none"> <li>AIX のシステム・チューニングについて詳しくは、AIX システムのチューニングを参照してください。</li> </ul>
<p>orb.properties ファイルを更新すると、グリッドのトランスポート動作を変更できません。orb.properties ファイルは、java/jre/lib ディレクトリにあります。</p>	<p>560 ページの『ORB プロパティ』</p>

表 15. 運用チェックリスト (続き)

チェックリスト項目	詳細
<p><b>startOgServer</b> スクリプトのパラメーターを使用します。特に、以下のパラメーターを使用します。</p> <ul style="list-style-type: none"> <li>• <b>-jvmArgs</b> パラメーターを使用してヒープ設定を設定します。</li> <li>• <b>-jvmArgs</b> パラメーターを使用してアプリケーション・クラスパスとプロパティを設定します。</li> <li>• エージェント・モニターの構成用に <b>-jvmArgs</b> パラメーターを設定します。</li> </ul> <p><b>ポートの設定</b></p> <p>WebSphere eXtreme Scale は、一部のトランスポートの通信用にポートを開く必要があります。これらのポートはすべて動的に定義されます。ただし、コンテナ間のファイアウォールが使用中の場合はポートを指定する必要があります。ポートに関する以下の情報を使用してください。</p> <p><b>リスナー・ポート</b></p> <p>プロセス間の通信に使用するポートを指定する場合、<b>-listenerPort</b> 引数を使用できます。</p> <p><b>コア・グループ・ポート</b></p> <p>障害検出に使用するポートを指定する場合、<b>-haManagerPort</b> 引数を使用できます。この引数は、<b>peerPort</b> と同じです。コア・グループは、ゾーンをまたいで通信する必要がないことに注意してください。したがって、ファイアウォールが単一ゾーンのすべてのメンバーに対してオープンである場合は、このポートを設定する必要はありません。</p> <p><b>JMX サービス・ポート</b></p> <p>JMX サービスが使用するポートを指定する場合、<b>-JMXServicePort</b> 引数を使用できます。</p> <p><b>SSL ポート</b></p> <p><b>-Dcom.ibm.CSI.SSLPort=1234</b> を <b>-jvmArgs</b> 引数として引き渡すと、SSL ポートが 1234 に設定されます。この SSL ポートは、リスナー・ポートと対等のセキュア・ポートです。</p> <p><b>クライアント・ポート</b></p> <p>カタログ・サービスのみで使用されます。この値は、<b>-catalogServiceEndpoints</b> 引数を使用して指定できます。このパラメーターの値は次の形式になります。</p> <pre>serverName:hostName:clientPort:peerPort</pre>	<p>457 ページの『startOgServer スクリプト』</p>
<p>次のセキュリティ設定が正しく構成されていることを検証します。</p> <ul style="list-style-type: none"> <li>• トランスポート (SSL)</li> <li>• アプリケーション (認証と許可)</li> </ul> <p>セキュリティ設定を検証するには、悪意のあるクライアントを使用してご使用の構成に接続を試みてください。例えば、SSL が必要な設定が構成されている場合に、TCP_IP 設定があるクライアント、あるいは、正しくないトラストストアのクライアントが、サーバーに接続できてはいけません。認証が必要な場合に、資格情報 (例えば、ユーザー ID とパスワードなど) を持たないクライアントは、サーバーに接続できてはいけません。許可が実行されている場合に、アクセス許可を持たないクライアントは、サーバー・リソースへのアクセスを認可されるべきではありません。</p>	<p>591 ページの『外部プロバイダーとのセキュリティ統合』</p>
<p>ご使用の環境をモニターする方法を選択します。</p> <ul style="list-style-type: none"> <li>• <b>xscmd</b> ツール:             <ul style="list-style-type: none"> <li>- カatalog・サーバーの JMX ポートが、<b>xscmd</b> ツールから可視である必要があります。コンテナ・サーバー・ポートも、コンテナからの情報を収集する一部のコマンドによってアクセス可能である必要があります。</li> </ul> </li> <li>• モニター・コンソール:             <ul style="list-style-type: none"> <li>- モニター・コンソールを使用して、現行統計およびヒストリカル統計をグラフに表すことができます。</li> </ul> </li> <li>• ベンダーのモニター・ツール:             <ul style="list-style-type: none"> <li>- Tivoli Enterprise Monitoring Agent</li> <li>- CA Wily Introscope</li> <li>- Hyperic HQ</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• 527 ページの『xscmd ユーティリティによるモニター』</li> <li>• 588 ページの『Java Management Extensions (JMX) セキュリティ』</li> <li>• 507 ページの『Web コンソールによるモニター』</li> <li>• 543 ページの『IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale のモニター』</li> <li>• 553 ページの『Hyperic HQ による eXtreme Scale のモニター』</li> <li>• 550 ページの『CA Wily Introscope による eXtreme Scale アプリケーションのモニター』</li> </ul>

## データ・グリッドの構成

ObjectGrid 記述子 XML ファイルを使用して、データ・グリッド、バックアップ・マップ、プラグインなどを構成します。WebSphere eXtreme Scale を構成するには、ObjectGrid ディスクリプター XML ファイルおよび ObjectGrid API を使用します。分散トポロジーの場合は、ObjectGrid 記述子 XML ファイルとデプロイメント・ポリシー XML ファイルが必要になります。

## ローカル・デプロイメントの構成

ローカルのメモリー内 eXtreme Scale 構成は、ObjectGrid 記述子 XML ファイルまたは API を使用して作成できます。

### このタスクについて

ローカル・デプロイメントを作成するには、ObjectGrid 記述子 XML ファイルを作成してから、そのファイルを ObjectGridManager インターフェースの createObjectGrid メソッドに渡します。

別の方法として、ObjectGridManager インターフェースを使用して、プログラマチックにデプロイメント全体を作成することもできます。

### 手順

1. ObjectGrid 記述子 XML ファイルを作成します。

以下の companyGrid.xml ファイルは、ObjectGrid 記述子 XML の例です。ファイルの最初の数行には、各 ObjectGrid XML ファイルの必須ヘッダーが含まれています。このファイルは、ObjectGrid インスタンス (「CompanyGrid」) および複数の BackingMap (「Customer」、「Item」、「OrderLine」、および「Order」) を定義しています。

#### companyGrid.xml ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="CompanyGrid">
      <backingMap name="Customer" />
      <backingMap name="Item" />
      <backingMap name="OrderLine" />
      <backingMap name="Order" />
    </objectGrid>
  </objectGrids>

</objectGridConfig>
```

2. ObjectGridManager インターフェース内で createObjectGrid メソッドのうちの 1 つに XML ファイルを受け渡します。

以下のコード・サンプルは、XML スキーマに対して companyGrid.xml ファイルを妥当性検査し、「CompanyGrid」という ObjectGrid インスタンスを作成しています。新規に作成された ObjectGrid インスタンスはキャッシュされません。

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid",
  new URL("file:etc/test/companyGrid.xml"), true, false);
```



## 次のタスク

ObjectGridManager インターフェースの createObjectGrid メソッドを使用して、プログラマチックにすべてのマップを定義する方法の詳細については、ObjectGridManager インターフェースを使用した ObjectGrid インスタンスの作成を参照してください。

## XML 構成の Evictor を使用可能にする

BackingMap インターフェースを使用して、TTL Evictor が使用する BackingMap 属性をプログラマチックに設定する代わりに、XML ファイルを使用して各 BackingMap インスタンスを構成することができます。以下のコードは、3 つの異なる BackingMap マップに対してこれらの属性を設定する方法を示しています。

### 始める前に

開始する前に、使用する Evictor のタイプを決定します。

- **デフォルトの時間ベースの TTL Evictor:** デフォルトの Evictor は、各 BackingMap インスタンスに対して、存続時間 (TTL、time-to-live) 除去ポリシーを使用します。
- **プラグ可能 Evictor 機構:** プラグ可能 Evictor は、通常、時間ではなく、エントリーの数に基づいた除去ポリシーを使用します。

Evictor のほとんどの設定は、ObjectGrid を初期化する前に設定しておく必要があります。

### 手順

- デフォルトの TTL Evictor を設定するには、ttlEvictorType 属性を ObjectGrid 記述子 XML ファイルに追加します。

次の例は、map1 BackingMap インスタンスが NONE TTL Evictor タイプを使用することを示しています。map2 BackingMap インスタンスは、LAST\_ACCESS\_TIME または LAST\_UPDATE\_TIME の TTL Evictor タイプを使用します。これらの設定うちのいずれかを指定してください。map2 BackingMap インスタンスは、存続時間値が 1800 秒 (30 分) です。map3 BackingMap インスタンスは、CREATION\_TIME TTL Evictor タイプを使用するように定義されており、その存続時間の値は 1200 秒、すなわち 20 分です。

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
  <objectGrid name="grid1">
    <backingMap name="map1" ttlEvictorType="NONE" />
    <backingMap name="map2" ttlEvictorType="LAST_ACCESS_TIME|LAST_UPDATE_TIME"
      timeToLive="1800" />
    <backingMap name="map3" ttlEvictorType="CREATION_TIME"
      timeToLive="1200" />
  </objectGrid>
</objectGrids>
</objectGridConfig>
```

図 27. XML により TimeToLive Evictor を使用可能にする

- プラグ可能 Evictor を設定するには、次の例を使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
  <objectGrid name="grid">
    <backingMap name="map1" ttlEvictorType="NONE" pluginCollectionRef="LRU" />
    <backingMap name="map2" ttlEvictorType="NONE" pluginCollectionRef="LFU" />
  </objectGrid>
</objectGrids>
<backingMapPluginCollections>
  <backingMapPluginCollection id="LRU">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      <property name="maxSize" type="int" value="1000" description="set max size
for each LRU queue" />
      <property name="sleepTime" type="int" value="15" description="evictor
thread sleep time" />
      <property name="numberOfLRUQueues" type="int" value="53" description="set number
of LRU queues" />
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="LFU">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor">
      <property name="maxSize" type="int" value="2000" description="set max size for each LFU heap" />
      <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
      <property name="numberOfHeaps" type="int" value="211" description="set number of LFU heaps" />
    </bean>
  </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

図 28. XML による Evictor のプラグ

## ロック・ストラテジーの構成

WebSphere eXtreme Scale 構成の各 BackingMap に対するオプティミスティック、ペシミスティック、あるいはロックなしのストラテジーを定義できます。

### このタスクについて

各 BackingMap インスタンスは、次のいずれかのロック・ストラテジーを使用するよう構成できます。

1. オプティミスティック・ロック・モード
2. ペシミスティック・ロック・モード
3. なし

デフォルトのロック・ストラテジーは、OPTIMISTIC です。データの変更が頻繁でない場合は、このオプティミスティック・ロックを使用します。データがキャッシュから読み取られ、トランザクションにコピーされる間、ロックは短期間だけ保持されます。トランザクション・キャッシュがメイン・キャッシュと同期されると、更新されたあらゆるキャッシュ・オブジェクトが元のバージョンに対してチェックされます。チェックが失敗すると、トランザクションはロールバックされ、OptimisticCollisionException 例外となります。

ペシミスティック・ロック・ストラテジーは、キャッシュ・エントリーに対してロックを取得するため、データが頻繁に変更される場合に使用するようにしてください。キャッシュ・エントリーが読み取られる場合は、必ずロックが取得され、トラ

ンザクションが完了するまでロックが条件付きで保持されます。ロックによっては、セッションのトランザクション分離レベルを使用して、その期間を調整することができます。

データがまったく更新されないか、静止期間のみに更新されるため、ロックが必要ない場合は、NONE ロック・ストラテジーを使用すれば、ロックを使用不可能にすることができます。このストラテジーは、ロック・マネージャーを必要としないため、非常に高速です。NONE ロック・ストラテジーは、ルックアップ表または読み取り専用のマップの場合に理想的です。

ロック・ストラテジーについて詳しくは、ロック・ストラテジー製品概要のロック・ストラテジーに関する説明を参照してください。

## 手順

### • オプティミスティック・ロック・ストラテジーの構成

- setLockStrategy メソッドを使用するプログラマチックな方法

```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
    ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("optimisticMap");
bm.setLockStrategy( LockStrategy.OPTIMISTIC );
```

- ObjectGrid 記述子 XML ファイル内の lockStrategy 属性を使用する方法

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="test">
      <backingMap name="optimisticMap"
        lockStrategy="OPTIMISTIC"/>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

### • ペシミスティック・ロック・ストラテジーの構成

- setLockStrategy メソッドを使用するプログラマチックな方法

プログラムでのペシミスティック・ストラテジーの指定

```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
    ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("pessimisticMap");
bm.setLockStrategy( LockStrategy.PESSIMISTIC );
```

- ObjectGrid 記述子 XML ファイル内の lockStrategy 属性を使用する方法

XML を使用したペシミスティック・ストラテジーの指定

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">
```

```

<objectGrids>
  <objectGrid name="test">
    <backingMap name="pessimisticMap"
      lockStrategy="PESSIMISTIC"/>
  </objectGrid>
</objectGrids>
</objectGridConfig>

```

#### • ロックなしストラテジーの構成

- setLockStrategy メソッドを使用するプログラマチックな方法

```

import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
  ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("noLockingMap");
bm.setLockStrategy( LockStrategy.NONE);

```

- ObjectGrid 記述子 XML ファイル内の lockStrategy 属性を使用する方法

```

<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="test">
      <backingMap name="noLockingMap"
        lockStrategy="NONE"/>
    </objectGrid>
  </objectGrids>
</objectGridConfig>

```

### 次のタスク

java.lang.IllegalStateException 例外を避けるために、ObjectGrid インスタンスで initialize メソッドまたは getSession メソッドを呼び出す前に、setLockStrategy メソッドを呼び出す必要があります。

## JMS を使用したピアツーピア・レプリカ生成の構成

Java Message Service (JMS) を使用したピアツーピア・レプリカ生成メカニズムは、分散およびローカルの両方の WebSphere eXtreme Scale 環境で使用されます。JMS は、コアツーコア・レプリカ生成プロセスであり、これにより、ローカル ObjectGrid と分散 ObjectGrid の間でデータ更新の流れが可能になります。例えば、このメカニズムを使用すると分散 eXtreme Scale データ・グリッドからローカル eXtreme Scale グリッドに、あるいは、あるグリッドから別のシステム・ドメインにある別のグリッドに、データ更新を移動させることができます。

### 始める前に

JMS ベースのピアツーピア・レプリカ生成メカニズムは、組み込まれた JMS ベースの ObjectGridEventListener

(com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener) に基づいています。ピアツーピア・レプリカ生成メカニズムの使用可能化に関する詳細については、265 ページの『JMS イベント・リスナー』を参照してください。

詳しくは、338 ページの『クライアント無効化メカニズムの使用可能化』を参照してください。

以下は、eXtreme Scale 構成上でピアツーピア・レプリカ生成メカニズムを使用可能にする XML 構成の例です。

ピアツーピア・レプリカ生成構成 - XML の例

```
<bean id="ObjectGridEventListener"
className="com.ibm.websphere.objectgrid.plugins.JMSObjectGridEventListener">
  <property name="replicationRole" type="java.lang.String" value="DUAL_ROLES" description="" />
  <property name="replicationStrategy" type="java.lang.String" value="PUSH" description="" />
  <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String"
value="defaultTCF" description="" />
  <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
  <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
  <property name="jms_userid" type="java.lang.String" value="" description="" />
  <property name="jms_password" type="java.lang.String" value="" description="" />
  <property name="jndi_properties" type="java.lang.String"
value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;
java.naming.provider.url=tcp://localhost:61616;connectionFactoryNames=defaultTCF;
topic.defaultTopic=defaultTopic"
description="jndi properties" />
</bean>
```

## ピア JVM 間の変更の配布

LogSequence オブジェクトおよび LogElement オブジェクトはピア JVM 間で変更を配布し、ObjectGridEventListener プラグインを使用して、eXtreme Scale トランザクションで発生した変更を通信します。

Java Message Service (JMS) を使用してトランザクションの変更を配布する方法について詳しくは、トランザクションの配布を参照してください。

ObjectGrid インスタンスが ObjectGridManager によりキャッシュされていることが前提条件です。詳しくは、createObjectGrid メソッドを参照してください。  
cacheInstance ブール値は、true に設定する必要があります。

このメカニズムを実装する必要はありません。この機能を使用するためのピアツーピア・レプリカ生成メカニズムが組み込まれています。260 ページの『JMS を使用したピアツーピア・レプリカ生成の構成』を参照してください。

オブジェクトは、アプリケーションがメッセージ・トランスポートを使用して、ObjectGrid で発生した変更をリモート Java 仮想マシン 内のピア ObjectGrids に簡単にパブリッシュし、それらの変更をその JVM 上で適用するための手段を提供します。LogSequenceTransformer クラスは、このサポートを使用可能にするために重要です。ここでは、メッセージを伝搬するために Java Message Service (JMS) メッセージング・システムを使用するリスナーをどのように作成すればいいのかを詳しく見ていきます。eXtreme Scale トランザクションのコミットが WebSphere Application Server クラスターの複数メンバーにわたって実行された結果として生じる LogSequence の伝送を、eXtreme Scale は IBM 提供のプラグインによってサポートします。この機能はデフォルトでは使用不可ですが、作動可能に構成することができます。ただし、コンシューマーまたはプロデューサーのいずれかが WebSphere Application Server ではない場合、外部 JMS メッセージング・システムが必要となる可能性があります。

## メカニズムの実装

LogSequenceTransformer クラス、ObjectGridEventListener、LogSequence および LogElement API により、信頼性のあるパブリッシュおよびサブスクライブを使用して、変更内容を配布し、配布するマップをフィルターに掛けることができます。このトピックのスニペットは、これらの API を JMS とともに使用して、ピアツーピア ObjectGrid をビルドする方法を示します。これは、共通メッセージ・トランスポートを共有するさまざまなプラットフォームのセットでホストされるアプリケーションによって共有されます。

### プラグインの初期化

ObjectGrid が開始するときに、ObjectGrid は、ObjectGridEventListener インターフェース契約の一部である、プラグインの initialize メソッドを呼び出します。initialize メソッドは、接続、セッション、およびパブリッシャーを含む JMS リソースを取得し、JMS リスナーであるスレッドを開始する必要があります。

以下の例は初期化メソッドを示しています。

```
initialize method example
public void initialize(Session session) {
    mySession = session;
    myGrid = session.getObjectGrid();
    try {
        if (mode == null) {
            throw new ObjectGridRuntimeException("No mode specified");
        }
        if (userid != null) {
            connection = topicConnectionFactory.createTopicConnection(userid,
password);
        } else
            connection = topicConnectionFactory.createTopicConnection();

        // need to start the connection to receive messages.
        connection.start();

        // the jms session is not transactional (false).
        jmsSession = connection.createTopicSession(false,
javax.jms.Session.AUTO_ACKNOWLEDGE);
        if (topic == null)
            if (topicName == null) {
                throw new ObjectGridRuntimeException("Topic not specified");
            } else {
                topic = jmsSession.createTopic(topicName);
            }
        publisher = jmsSession.createPublisher(topic);
        // start the listener thread.
        listenerRunning = true;
        listenerThread = new Thread(this);
        listenerThread.start();
    } catch (Throwable e) {
        throw new ObjectGridRuntimeException("Cannot initialize", e);
    }
}
```

スレッドを開始するためのコードは、Java 2 Platform, Standard Edition (Java SE) スレッドを使用します。WebSphere Application Server バージョン 6.x または WebSphere Application Server バージョン 5.x エンタープライズ・サーバーを実行している場合、非同期 Bean アプリケーション・プログラミング・インターフェース



(API) を使用して、このデーモン・スレッドを開始します。共通 API を使用することもできます。以下に、作業マネージャーを使用する同じアクションを示す置換の断片の例を示します。

```
// start the listener thread.
listenerRunning = true;
workManager.startWork(this, true);
```

また、プラグインは、実行可能なインターフェースの代わりに、作業インターフェースを実装する必要があります。また、リリース・メソッドを追加して、`listenerRunning` 変数を `false` に設定する必要があります。プラグインは、コンストラクター、または Inversion of Control (IoC) コンテナを使用している場合は注入によって、`WorkManager` インスタンスに提供されている必要があります。

## 変更の送信

以下は、ObjectGrid 上で行われるローカル変更をパブリッシュするためのサンプル `transactionEnd` メソッドです。このサンプルでは JMS を使用していますが、信頼できるパブリッシュおよびサブスクライブ・メッセージングの機能を持つ任意のメッセージ・トランスポートを使用することもできます。

```
transactionEnd method example
// This method is synchronized to make sure the
// messages are published in the order the transaction
// were committed. If we started publishing the messages
// in parallel then the receivers could corrupt the Map
// as deletes may arrive before inserts etc.
public synchronized void transactionEnd(String txid, boolean isWriteThroughEnabled,
boolean committed,
Collection changes) {
    try {
        // must be write through and committed.
        if (isWriteThroughEnabled && committed) {
            // write the sequences to a byte []
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(bos);
            if (publishMaps.isEmpty()) {
                // serialize the whole collection
                LogSequenceTransformer.serialize(changes, oos, this, mode);
            } else {
                // filter LogSequences based on publishMaps contents
                Collection publishChanges = new ArrayList();
                Iterator iter = changes.iterator();
                while (iter.hasNext()) {
                    LogSequence ls = (LogSequence) iter.next();
                    if (publishMaps.contains(ls.getMapName())) {
                        publishChanges.add(ls);
                    }
                }
                LogSequenceTransformer.serialize(publishChanges, oos, this, mode);
            }
            // make an object message for the changes
            oos.flush();
            ObjectMessage om = jmsSession.createObjectMessage(bos.toByteArray());
            // set properties
            om.setStringProperty(PROP_TX, txid);
            om.setStringProperty(PROP_GRIDNAME, myGrid.getName());
            // transmit it.
            publisher.publish(om);
        }
    } catch (Throwable e) {
        throw new ObjectGridRuntimeException("Cannot push changes", e);
    }
}
```

このメソッドは、いくつかのインスタンス変数を使用します。

- `jmsSession` 変数: メッセージをパブリッシュするために使用する JMS セッションです。これは、プラグインが初期設定される際に作成されます。



- mode 変数: 配布モードです。
- publishMaps 変数: パブリッシュする変更内容を持つ各マップの名前が含まれている集合です。この変数が空の場合、すべてのマップがパブリッシュされます。
- publisher 変数: プラグインの initialize メソッド中に作成される TopicPublisher オブジェクトです。

### 更新メッセージの受信および適用

以下は実行メソッドです。このメソッドは、アプリケーションがループを停止するまで、ループ内で実行します。各ループの反復は、JMS メッセージの受信、およびメッセージの ObjectGrid への適用を試行します。

#### JMS message run method example

```
private synchronized boolean isListenerRunning() {
    return listenerRunning;
}

public void run() {
    try {
        System.out.println("Listener starting");
        // get a jms session for receiving the messages.
        // Non transactional.
        TopicSession myTopicSession;
        myTopicSession = connection.createTopicSession(false, javax.jms.
        Session.AUTO_ACKNOWLEDGE);

        // get a subscriber for the topic, true indicates don't receive
        // messages transmitted using publishers
        // on this connection. Otherwise, we'd receive our own updates.
        TopicSubscriber subscriber = myTopicSession.createSubscriber(topic,
        null, true);
        System.out.println("Listener started");
        while (isListenerRunning()) {
            ObjectMessage om = (ObjectMessage) subscriber.receive(2000);
            if (om != null) {
                // Use Session that was passed in on the initialize...
                // very important to use no write through here
                mySession.beginNoWriteThrough();
                byte[] raw = (byte[]) om.getObject();
                ByteArrayInputStream bis = new ByteArrayInputStream(raw);
                ObjectInputStream ois = new ObjectInputStream(bis);
                // inflate the LogSequences
                Collection collection = LogSequenceTransformer.inflate(ois,
                myGrid);
                Iterator iter = collection.iterator();
                while (iter.hasNext()) {
                    // process each Maps changes according to the mode when
                    // the LogSequence was serialized
                    LogSequence seq = (LogSequence) iter.next();
                    mySession.processLogSequence(seq);
                }
                mySession.commit();
            } // if there was a message
        } // while loop
        // stop the connection
        connection.close();
    } catch (IOException e) {
        System.out.println("IO Exception: " + e);
    } catch (JMSException e) {
        System.out.println("JMS Exception: " + e);
    } catch (ObjectGridException e) {
        System.out.println("ObjectGrid exception: " + e);
        System.out.println("Caused by: " + e.getCause());
    } catch (Throwable e) {
```

```
        System.out.println("Exception : " + e);
    }
    System.out.println("Listener stopped");
}
```

## JMS イベント・リスナー

JMSObjectGridEventListener は、クライアント・サイド・ニア・キャッシュの無効化およびピアツーピア・レプリカ生成メカニズムをサポートするように設計されています。これは、ObjectGridEventListener インターフェースの Java Message Service (JMS) 実装です。

クライアント無効化メカニズムは、クライアントのニア・キャッシュ・データがサーバーまたは他のクライアントと同期するように、分散 eXtreme Scale 環境で使用できます。この機能がないと、クライアントのニア・キャッシュに失効データが保持される可能性があります。ただし、この JMS ベースのクライアント無効化メカニズムを使用しても、クライアント・ニア・キャッシュを更新する場合の時間帯を考慮に入れる必要があります。実行時に更新の公開に遅延があるためです。

ピアツーピア・レプリカ生成メカニズムは、分散とローカルの両方の eXtreme Scale 環境で使用できます。これは、ObjectGrid コアツーコア・レプリカ生成プロセスであり、これにより、ローカル ObjectGrid と分散 ObjectGrid の間で流れるデータ更新が可能になります。例えば、このメカニズムを使用すると、分散グリッドからローカル ObjectGrid に、あるいはあるグリッドから別のシステム・ドメインにある別のグリッドに、データ更新を移動させることができます。

JMSObjectGridEventListener の場合、ユーザーは、必要な JMS リソースを取得するために、JMS および Java Naming and Directory Interface (JNDI) 情報を構成する必要があります。さらに、複製関連のプロパティを正しく設定する必要があります。JEE 環境では、Web と Enterprise JavaBean (EJB) の両方のコンテナで JNDI が使用可能になっている必要があります。この場合、外部 JMS リソースを取得したい場合を除いて JNDI プロパティはオプションです。

このイベント・リスナーには、XML を使用するか、プログラマチックな方法を使用して構成できるプロパティがあります。これは、クライアント無効化のみ、ピアツーピア・レプリカ生成のみ、またはその両方に使用できます。必要な機能を実現するための振る舞いをカスタマイズする場合は、ほとんどのプロパティはオプションです。

詳しくは、JMSObjectGridEventListener API を参照してください。

## JMSObjectGridEventListener プラグインの拡張

JMSObjectGridEventListener プラグインを使用すると、グリッド内のデータが変更または除去されたときに、ピア ObjectGrid インスタンスが更新を受信できます。また、eXtreme Scale グリッドからエントリーが更新または除去されたときに、クライアントが通知を受信することも可能です。このトピックでは、JMS メッセージを受信されたときに、アプリケーションが通知を受け取れるように、JMSObjectGridEventListener プラグインを拡張する方法について説明します。これは、クライアント無効化に CLIENT\_SERVER\_MODEL 設定を使用する場合に最も役立ちます。

receiver ロールで実行中の場合、JMSObjectGridEventListener インスタンスがグリッドから JMS メッセージ更新を受信すると、オーバーライドされた JMSObjectGridEventListener.onMessage メソッドが eXtreme Scale ランタイムによって自動的に呼び出されます。これらのメッセージは、LogSequence オブジェクトの集まりを折り返します。LogSequence オブジェクトは onMessage メソッドに送られ、アプリケーションはこの LogSequence を使用して挿入、削除、更新、または無効化されたキャッシュ・エントリーを判別します。

onMessage 拡張ポイントを使用するために、アプリケーションは以下のステップを実行します。

1. JMSObjectGridEventListener クラスが拡張し、onMessage メソッドがオーバーライドする新規クラスを作成します。
2. ObjectGrid の ObjectGridEventListener と同様に拡張 JMSObjectGridEventListener を構成します。

拡張 JMSObjectGridEventListener クラスは、JMSObjectGridEventListener クラスの子クラスであり、initialize (オプション) と onMessage という 2 つのメソッドのみをオーバーライドできます。JMSObjectGridEventListener クラスの子クラスが onMessage メソッド内の ObjectGrid や Session などの ObjectGrid 成果物を使用する必要がある場合、その子クラスは、initialize メソッドでその成果物を取得して、それをインスタンス変数としてキャッシュできます。また、onMessage メソッドでは、キャッシュされた ObjectGrid 成果物は、渡された LogSequences の集まりを処理する場合に使用できます。

注: オーバーライドされた initialize メソッドは、親 JMSObjectGridEventListener を適切に初期化するために、super.initialize メソッドを呼び出す必要があります。

以下は、拡張 JMSObjectGridEventListener クラスの例です。

```
package com.ibm.websphere.samples.objectgrid.jms.price;

import java.util.*;
import com.ibm.websphere.objectgrid.*;
import com.ibm.websphere.objectgrid.plugins.LogElement;
import com.ibm.websphere.objectgrid.plugins.LogSequence;
import com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener;

public class ExtendedJMSObjectGridEventListener extends JMSObjectGridEventListener{
    protected static boolean debug = true;

    /**
     * This is the grid associated with this listener.
     */
    ObjectGrid grid;

    /**
     * This is the session associated with this listener.
     */
    Session session;

    String objectGridType;

    public List receivedLogSequenceList = new ArrayList();

    /* (non-Javadoc)
     * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
     * #initialize(com.ibm.websphere.objectgrid.Session)
     */
    public void initialize(Session session) {
        // Note: if need to use any ObjectGrid artifact, this class need to get ObjectGrid
        // from the passed Session instance and get ObjectMap from session instance
        // for any transactional ObjectGrid map operation.
    }
}
```

```

super.initialize(session); // must invoke super's initialize method.
this.session = session; // cache the session instance, in case need to
// use it to perform map operation.
this.grid = session.getObjectGrid(); // get ObjectGrid, in case need
// to get ObjectGrid information.

if (grid.getObjectGridType() == ObjectGrid.CLIENT)
    objectGridType = "CLIENT";
else if (grid.getObjectGridType() == ObjectGrid.SERVER)
    objectGridType = "Server";

if (debug)
    System.out.println("ExtendedJMSObjectGridEventListener[" +
        objectGridType + "].initialize() : grid = " + this.grid);
}

/* (non-Javadoc)
 * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
 * #onMessage(java.util.Collection)
 */
protected void onMessage(Collection logSequences) {
    System.out.println("ExtendedJMSObjectGridEventListener[" +
        objectGridType + "].onMessage(): ");

    Iterator iter = logSequences.iterator();

    while (iter.hasNext()) {
        LogSequence seq = (LogSequence) iter.next();

        StringBuffer buffer = new StringBuffer();
        String mapName = seq.getMapName();
        int size = seq.size();
        buffer.append("%nLogSequence[mapName=" + mapName + ", size=" + size + ",
            objectGridType=" + objectGridType
                + "]: ");

        Iterator logElementIter = seq.getAllChanges();
        for (int i = seq.size() - 1; i >= 0; --i) {
            LogElement le = (LogElement) logElementIter.next();
            buffer.append(le.getType() + " -> key=" + le.getCacheEntry().getKey() + ", ");
        }
        buffer.append("%n");

        receivedLogSequenceList.add(buffer.toString());

        if (debug) {
            System.out.println("ExtendedJMSObjectGridEventListener["
                + objectGridType + "].onMessage(): " + buffer.toString());
        }
    }
}

public String dumpReceivedLogSequenceList() {
    String result = "";
    int size = receivedLogSequenceList.size();
    result = result + "%nExtendedJMSObjectGridEventListener[" + objectGridType
        + "]: receivedLogSequenceList size = " + size + "%n";
    for (int i = 0; i < size; i++) {
        result = result + receivedLogSequenceList.get(i) + "%n";
    }
    return result;
}

public String toString() {
    return "ExtendedJMSObjectGridEventListener["
        + objectGridType + " - " + this.grid + "];"
}
}

```

## 構成

拡張 `JMSObjectGridEventListener` クラスは、クライアント無効化の場合にも、ピアツーピア・レプリカ生成メカニズムの場合にも同様に構成する必要があります。以下は XML 構成の例です。

```

<objectGrid name="PRICEGRID">
  <bean id="ObjectGridEventListener"
    className="com.ibm.websphere.samples.objectgrid.jms.
      price.ExtendedJMSObjectGridEventListener">
    <property name="invalidationModel" type="java.lang.String"
      value="CLIENT_SERVER_MODEL" description="" />
    <property name="invalidationStrategy" type="java.lang.String"
      value="INVALIDATE" description="" />
    <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String"
      value="jms/TCF" description="" />
    <property name="jms_topicJndiName" type="java.lang.String"
      value="GRID.PRICEGRID" description="" />
    <property name="jms_topicName" type="java.lang.String"
      value="GRID.PRICEGRID" description="" />
    <property name="jms_userid" type="java.lang.String" value=""
      description="" />
    <property name="jms_password" type="java.lang.String" value=""
      description="" />
  </bean>
  <backingMap name="PRICE" pluginCollectionRef="PRICE"></backingMap>
</objectGrid>

```

注: ObjectGridEventListener Bean の className は、一般 JMSObjectGridEventListener と同じプロパティを持つ拡張 JMSObjectGridEventListener クラスによって構成されます。

---

## デプロイメント・ポリシーの構成

デプロイメント・ポリシー記述子 XML ファイルおよび objectgrid 記述子 XML ファイルを使用して、分散トポロジーを管理します。デプロイメント・ポリシーは、コンテナ・サーバーに提供される XML ファイルとしてエンコードされます。デプロイメント・ポリシーは、マップ、マップ・セット、区画、レプリカなどの情報を提供します。また、断片配置の動作も制御します。

### 分散デプロイメントの構成

デプロイメント・ポリシー記述子 XML ファイルおよび ObjectGrid 記述子 XML ファイルを使用して、トポロジーを管理します。

デプロイメント・ポリシーは、eXtreme Scale コンテナ・サーバーに提供される XML ファイルとしてエンコードされます。XML ファイルでは、以下の情報が指定されます。

- 各マップ・セットに属するマップ
- 区画の数
- 同期レプリカおよび非同期レプリカの数

デプロイメント・ポリシーでは、以下の配置の振る舞いも制御されます。

- 配置を実行する前のアクティブ・コンテナ・サーバーの最小数
- 破損した断片の自動置き換え
- 単一区画の各断片の別のマシンへの配置

エンドポイント情報は、動的環境では事前構成されません。デプロイメント・ポリシーには、サーバー名も物理トポロジー情報もありません。データ・グリッド内のすべての断片は、カタログ・サービスによって自動的にコンテナ・サーバーに配

置されます。カタログ・サービスは、デプロイメント・ポリシーで定義されている制約を使用して、断片配置を自動的に管理します。この自動断片配置により、大きなデータ・グリッドの構成が容易になります。また必要に応じて、使用している環境にサーバーを追加することもできます。

**制約事項:** WebSphere Application Server 環境では、50 を超えるメンバーが入っているコア・グループ・サイズはサポートされません。

デプロイメント・ポリシー XML ファイルは、始動時にコンテナ・サーバーに渡されます。デプロイメント・ポリシーは、ObjectGrid XML ファイルと一緒に使用する必要があります。デプロイメント・ポリシーはコンテナ・サーバーを始動するために必須ではありませんが、使用されることをお勧めします。デプロイメント・ポリシーは、それと一緒に使用される ObjectGrid XML ファイルと互換性がある必要があります。デプロイメント・ポリシー内の各 objectgridDeployment エレメントごとに、対応する 1 つの objectGrid エレメントが ObjectGrid XML ファイル内に必要です。objectgridDeployment 内のマップは、ObjectGrid XML 内の backingMap エレメントと整合している必要があります。すべての backingMap は、1 つの mapSet エレメント内のみで参照する必要があります。

以下の例では、companyGridDpReplication.xml ファイルは、対応する companyGrid.xml ファイルとペアになっていることが想定されています。

```
companyGridDpReplication.xml
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="CompanyGrid">
    <mapSet name="mapSet1" numberOfPartitions="11"
      minSyncReplicas="1" maxSyncReplicas="1"
      maxAsyncReplicas="0" numInitialContainers="4">
      <map ref="Customer" />
      <map ref="Item" />
      <map ref="OrderLine" />
      <map ref="Order" />
    </mapSet>
  </objectgridDeployment>

</deploymentPolicy>

companyGrid.xml
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="CompanyGrid">
      <backingMap name="Customer" />
      <backingMap name="Item" />
      <backingMap name="OrderLine" />
      <backingMap name="Order" />
    </objectGrid>
  </objectGrids>

</objectGridConfig>
```

companyGridDpReplication.xml ファイルには、11 個の区画に分割されている mapSet エレメントが 1 つあります。各区画に含めることができる同期複製は 1 つだけです。同期レプリカの数、minSyncReplicas 属性および maxSyncReplicas 属性によって指定します。minSyncReplicas 属性が 1 に設定されているため、書き込みトランザクションを処理するためには、mapSet エレメント内の各区画では、少なくとも 1 つの同期レプリカが使用可能になっている必要があります。



maxSyncReplicas 属性が 1 に設定されているため、各区画の同期レプリカの数 が 1 つを超えることはできません。この mapSet エlement 内の区画には、非同期レプリカは含まれていません。

カタログ・サービスは、この ObjectGrid インスタンスをサポートするために、numInitialContainers 属性に従って、4 つのコンテナ・サーバーが使用可能になるまで配置を据え置きます。コンテナ・サーバーが指定数に達すると、numInitialContainers 属性は無視されます。

**placementDeferralInterval** プロパティと **xscmd -c suspendBalancing** コマンドを使用して、コンテナ・サーバーの断片の配置を遅らせることもできます。

companyGridDpReplication.xml ファイルは基本的な例ですが、デプロイメント・ポリシーによって、環境を完全に制御できます。

## 分散トポロジー

分散コヒーレント・キャッシュを使用すると、構成できるパフォーマンス、可用性、およびスケーラビリティが改善されます。

WebSphere eXtreme Scale は自動的にサーバーのバランスを取ります。WebSphere eXtreme Scale を再始動しなくても、追加サーバーを組み込むことができます。eXtreme Scale を再始動せずにサーバーを追加できることで、単純なデプロイメントだけでなく、数千ものサーバーが必要になる大規模なテラバイト・サイズのデプロイメントが可能になります。

このデプロイメント・トポロジーは柔軟です。カタログ・サービスを使用すると、キャッシュ全体を除去することなく、サーバーを追加および除去して、リソースを効率的に使用できるようになります。 **startOgServer** コマンドおよび **stopOgServer** コマンドを使用して、コンテナ・サーバーを始動および停止できます。これらのいずれのコマンドを使用した場合でも、**-catalogServiceEndpoints** オプションを指定する必要があります。分散トポロジーのすべてのクライアントは、Internet Interoperability Object Protocol (IIOP) を介してカタログ・サービスと通信します。すべてのクライアントは、ObjectGrid インターフェースを使用してサーバーと通信できます。

WebSphere eXtreme Scale の動的構成機能を使用すると、簡単にシステムにリソースを追加することができます。コンテナはデータをホスティングします。また、カタログ・サービスによって、クライアントはコンテナ・サーバーのグリッドと通信できます。カタログ・サービスは、要求を転送し、ホスト・コンテナ・サーバー内のスペースを割り振り、システム全体のヘルスと可用性を管理します。クライアントは、カタログ・サービスに接続し、コンテナ・サーバー・トポロジーの記述を取得し、必要に応じて各サーバーと直接通信します。新規サーバーの追加または他のサーバーの障害によってサーバー・トポロジーが変更されると、カタログ・サービスは、データをホスティングする適切なサーバーにクライアント要求を自動的に経路指定します。

通常、カタログ・サービスは、自身の Java 仮想マシン グリッド内に存在します。単一のカタログ・サーバーで複数のサーバーを管理できます。コンテナ・サーバーは、JVM 内で単独で開始することも、別のサーバーの他のコンテナ・サーバーとともに任意の JVM にロードすることもできます。クライアントはどの JVM 内



にも存在でき、1 つ以上のサーバーと通信できます。また、クライアントはコンテナ・サーバーと同じ JVM 内に存在することも可能です。

既存の Java プロセスまたはアプリケーションにコンテナ・サーバーを組み込む際には、プログラムでデプロイメント・ポリシーを作成することもできます。詳細については、DeploymentPolicy API の資料を参照してください。

## ゾーンによる断片配置の制御

デプロイメント・ポリシーを使用して、ゾーンを定義します。ゾーンを使用すると、WebSphere eXtreme Scale での断片配置を制御できます。ゾーンとは、物理サーバーの論理グループを表すために使用される、ユーザー定義の論理的な概念です。

### レプリカ配置のためのゾーンの構成

ゾーン・サポートは、データ・センター間でのレプリカ配置のための高度な構成を可能にします。この機能により、少数のオプションの配置ルールを使用して、何千という区画のグリッドを簡単に管理することができます。データ・センターは、ゾーン・ルールによる構成に従って、ビルの別フロア、別ビル、または異なる都市にさえ配置することも、またはその他の区分に配置することができます。

### ゾーンの柔軟性

ゾーンに断片を配置することができます。この機能により、eXtreme Scale がグリッド上に断片をどのように配置するかをさらに制御できるようになります。eXtreme Scale サーバーをホストする Java 仮想マシンは、ゾーン ID によってタグ付けすることができます。現在、デプロイメント・ファイルには 1 つ以上のゾーン・ルールを含めることができます。これらのゾーン・ルールは、断片タイプに関連付けられます。次のセクションでは、ゾーンの使用方法について概要を示します。詳しくは、「管理ガイド」のゾーンを使用した断片配置の制御に関する情報を参照してください。

配置ゾーンは、高度なトポロジーを構成するために、eXtreme Scale がプライマリーとレプリカの割り当てをどのように達成するかを制御します。

Java 仮想マシンは、複数のコンテナを持つことができますが、サーバーは 1 つだけしか持てません。コンテナは、単一の ObjectGrid の複数の断片をホストできません。

この機能を利用すると、レプリカとプライマリーを異なるロケーションまたはゾーンに配置して、より優れた高可用性を実現することができます。通常、eXtreme Scale は、同じ IP アドレスでプライマリー断片とレプリカ断片を Java 仮想マシンに配置することはありません。この単純なルールにより、一般的には、2 つの eXtreme Scale サーバーが同じ物理コンピューターに配置されることがなくなります。ただし、より柔軟なメカニズムが必要になる場合もあります。例えば、2 つのブレード・シャーシを使用していて、プライマリーを両方のシャーシ間でストライプし、それぞれのプライマリーのレプリカがそのプライマリーの他方のシャーシに配置されるようにしたい場合があります。

ストライプ・プライマリーは、プライマリーが各ゾーンに配置され、その各プライマリーのレプリカが対向ゾーンに配置されることを意味します。例えば、プライマ

リー 0 は zoneA に入り、同期レプリカ 0 は zoneB に入ります。プライマリー 1 は zoneB に入り、同期レプリカ 1 は zoneA に入ります。

この場合、シャーシ名がゾーン名となります。代替方法として、ビルのフロアの後にゾーンを命名し、ゾーンを使用して、同じデータのプライマリーとレプリカが別フロアに配置されるようにすることができます。ビルおよびデータ・センターでも可能です。データ・センター間でデータが適切に複製されるようにするためのメカニズムとしてゾーンを使用し、データ・センター全体に渡るテストが実行されています。eXtreme Scale の HTTP セッション・マネージャーを使用している場合も、ゾーンを使用することができます。このフィーチャーを使用すると、1 つの Web アプリケーションを 3 つのデータ・センターに渡ってデプロイできます。これにより、ユーザーの HTTP セッションがデータ・センター間で複製され、1 つのデータ・センター全体が障害を起こした場合にも、セッションを回復できるようになります。

WebSphere eXtreme Scale は、複数のデータ・センターに渡る大きなグリッドを管理する必要性を配慮されています。これにより、同一区画のバックアップとプライマリーを必要に応じて異なるデータ・センターに配置することが可能です。すべてのプライマリーをデータ・センター 1 に、すべてのレプリカをデータ・センター 2 に配置したり、両方のデータ・センター間でプライマリーとレプリカをラウンドロビンしたりすることができます。ルールは柔軟であり、さまざまなシナリオが考えられます。また eXtreme Scale は数千ものサーバーを管理することができます。これにより、データ・センター認識による完全な自動配置を同時に使用することによって、管理の観点から手ごろな大規模グリッドの作成が可能です。管理者は、簡単かつ効果的に実行するものを指定できます。

管理者の場合、配置ゾーンを使用して、プライマリー断片とレプリカ断片の配置場所を制御できます。これにより、高性能でかつ高可用性のトポロジーのセットアップが可能になります。前述したように、eXtreme Scale プロセスのいずれの論理グループに対してもゾーンを定義できます。これらのゾーンは、データ・センター、データ・センターのフロア、ブレード・シャーシなど、物理ワークステーション・ロケーションに対応付けることができます。ゾーン間でデータをストライプすることができます。これにより、可用性が増します。またホット・スタンバイが必要な場合に、プライマリーとレプリカを別々のゾーンに分割することができます。

## eXtreme Scale サーバーの WebSphere Extended Deployment を使用しないゾーンとの関連付け

eXtreme Scale が Java Standard Edition で使用されるか、あるいは、WebSphere Extended Deployment バージョン 6.1 をベースにしていないアプリケーション・サーバーで使用される場合、断片コンテナである JVM は、以下の手法を使用して、ゾーンに関連付けられます。

### startOgServer スクリプトを使用するアプリケーション

startOgServer スクリプトは、既存のサーバーに組み込まれない eXtreme Scale アプリケーションを開始するために使用されます。**-zone** パラメーターを使用すると、サーバー内のすべてのコンテナに使用するゾーンを指定できます。

### API を使用するコンテナを開始する場合のゾーンの指定

## WebSphere Extended Deployment ノードのゾーンとの関連付け

eXtreme Scale を WebSphere Extended Deployment Java EE アプリケーションで使用している場合、WebSphere Extended Deployment ノード・グループを使用して、サーバーを特定のゾーンに配置できます。

eXtreme Scale では、JVM は、単一ゾーンのメンバーになることだけを許可されています。ただし、WebSphere は、ノードが複数ノード・グループの一部になることを許可しています。各ノードが 1 つのゾーンのノード・グループ内のみにあると確認できれば、eXtreme Scale のゾーンのこの機能を使用できます。

ノード・グループがゾーンであると宣言するために、次の構文を使用して、ノード・グループに名前を付けてください。ReplicationZone<UniqueSuffix> そのようなノード・グループの一部であるノード上で稼働しているサーバーは、ノード・グループ名で指定されるゾーンに組み込まれます。以下に、トポロジーの例を説明します。

まず、4 つのノードを構成します。node1、node2、node3、および node4 で、各ノードには 2 台のサーバーがあります。その後、ReplicationZoneA という名前のノード・グループと ReplicationZoneB という名前のノード・グループを作成します。次に、node1 と node2 を ReplicationZoneA に追加し、node3 と node4 を ReplicationZoneB に追加します。

node1 と node2 のサーバーが始動すると、それらのサーバーは ReplicationZoneA の一部になり、同様に、node3 と node4 のサーバーは ReplicationZoneB の一部になります。

グリッド・メンバー JVM は、始動時にのみゾーン・メンバーシップをチェックします。新規ノード・グループを追加するか、メンバーシップを変更した場合、それは新たに始動されるか、再始動される JVM のみに影響します。

### ゾーン・ルール

eXtreme Scale 区画には、1 つのプライマリー断片とゼロ個以上のレプリカ断片があります。この例の場合、これらの断片について以下の命名規則を考慮してください。P はプライマリー断片で、S は同期レプリカで、A は非同期レプリカです。ゾーン・ルールは以下の 3 つの部分からなります。

- ルール名
- ゾーンのリスト
- 包含または排他フラグ

コンテナのゾーン名は、471 ページの『組み込みサーバー API』の文書に記述されているとおりに指定できます。ゾーン・ルールは、断片を配置できるゾーンのセットを指定します。包含フラグは、1 つの断片がリストからゾーンに配置されると、他のすべての断片もそのゾーンに配置されることを示します。排他設定は、区画の各断片がゾーン・リストの異なるゾーンに配置されることを示します。例えば、排他設定を使用する場合、3 つの断片 (プライマリーと 2 つの同期レプリカ) がある場合は、ゾーン・リストに 3 つのゾーンがなければならないということです。

各断片は、1 つのゾーン・ルールに関連付けることができます。ゾーン・ルールは、2 つの断片間で共有できます。ルールが共有される場合、含ままたは排他フラグは、1 つのルールを共有するすべてのタイプの断片に拡張されます。

## 例

さまざまなシナリオおよびそのシナリオを実装するためのデプロイメント構成を示す一連の例は、以下のとおりです。

### ゾーン間でプライマリーとレプリカをストライピングする

3 つのブレード・シャーシがあり、3 つすべてにプライマリーを分散し、1 つの同期レプリカをプライマリー以外のシャーシに配置するものとします。各シャーシをシャーシ名 ALPHA、BETA、および GAMMA を持つゾーンとして定義します。デプロイメント XML の例は以下のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="library">
    <mapSet name="ms1" numberOfPartitions="37" minSyncReplicas="1"
      maxSyncReplicas="1" maxAsyncReplicas="0">
      <map ref="book" />
    <zoneMetadata>
      <shardMapping shard="P" zoneRuleRef="stripeZone"/>
      <shardMapping shard="S" zoneRuleRef="stripeZone"/>
      <zoneRule name="stripeZone" exclusivePlacement="true" >
        <zone name="ALPHA" />
        <zone name="BETA" />
        <zone name="GAMMA" />
      </zoneRule>
    </zoneMetadata>
  </mapSet>
</objectgridDeployment>
</deploymentPolicy>
```

このデプロイメント XML には、「book」という名前の 1 つのマッピングを持つ「library」という名前のグリッドが含まれます。さらに、1 つの同期レプリカを持つ 4 つの区画を使用します。zone metadata 文節は、1 つのゾーン・ルールの定義およびゾーン・ルールと断片の関連付けを示します。プライマリー断片と同期断片は、ともにゾーン・ルール「stripeZone」に関連付けられます。このゾーン・ルールでは 3 つのゾーンがすべて含まれ、排他配置が使用されるようになっています。このルールは、区画 0 のプライマリーが ALPHA に配置されると、区画 0 のレプリカは BETA か GAMMA のいずれかに配置されることとなります。他の区画のプライマリーは他のゾーンに配置され、レプリカが同様に配置されます。

### 非同期レプリカがプライマリーや同期レプリカと異なるゾーンにある

この例では、2 つのビルがあり、その間の接続は待ち時間が長いものとして扱います。すべてのシナリオでデータ損失のない高可用性が保たれるようにします。ただし、ビル間の同期レプリカ生成によるパフォーマンス・インパクトのためトレードオフが生じます。このため、一方のビルに同期レプリカがあり、他方のビルに非同期レプリカがあるようなプライマリーが必要になります。通常では、障害は、大規模な問題ではなく、JVM の破損やコンピューター障害です。このトポロジーを使用すると、データ損失なしに通常の障害を切り抜けることができます。ビルがなくなることは非常にまれなことであるため、その場合でもある程度のデータ損失は許容範囲内に収まります。それぞれのビルに 1 つずつ、合計 2 つのゾーンを作成できます。デプロイメント XML ファイルは以下のようになります。

```

<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

<objectgridDeployment objectgridName="library">
  <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="1"
    maxSyncReplicas="1" maxAsyncReplicas="1">
    <map ref="book" />
    <zoneMetadata>
      <shardMapping shard="P" zoneRuleRef="primarySync"/>
      <shardMapping shard="S" zoneRuleRef="primarySync"/>
      <shardMapping shard="A" zoneRuleRef="aysnc"/>
      <zoneRule name="primarySync" exclusivePlacement="false" >
        <zone name="B1dA" />
        <zone name="B1dB" />
      </zoneRule>
      <zoneRule name="aysnc" exclusivePlacement="true">
        <zone name="B1dA" />
        <zone name="B1dB" />
      </zoneRule>
    </zoneMetadata>
  </mapSet>
</objectgridDeployment>
</deploymentPolicy>

```

プライマリーと同期レプリカは、排他フラグ設定 `false` で `primarySync` ゾーン・ルールを共有します。このため、プライマリーか同期のいずれかがゾーンに配置されると、他方も同じゾーンに配置されます。非同期レプリカは、`primarySync` ゾーン・ルールと同じゾーンで第 2 のゾーン・ルールを使用しますが、`true` に設定された **exclusivePlacement** 属性を使用します。この属性は、断片を同じ区画の別の断片があるゾーンには配置できないことを示します。結果的に、非同期レプリカは、プライマリーまたは同期レプリカと同じゾーンに配置されません。

### すべてのプライマリーを 1 つのゾーンに配置し、すべてのレプリカを別のゾーンに配置する

ここでは、すべてのプライマリーが特定のゾーンに配置され、すべてのレプリカが別のゾーンに配置されます。これにより、1 つのプライマリーと 1 つの非同期レプリカを持つこととなります。すべてのレプリカはゾーン A に、プライマリーはゾーン B に配置されます。

```

<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

<objectgridDeployment objectgridName="library">
  <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="0"
    maxSyncReplicas="0" maxAsyncReplicas="1">
    <map ref="book" />
    <zoneMetadata>
      <shardMapping shard="P" zoneRuleRef="primaryRule"/>
      <shardMapping shard="A" zoneRuleRef="replicaRule"/>
      <zoneRule name="primaryRule">
        <zone name="A" />
      </zoneRule>
      <zoneRule name="replicaRule">
        <zone name="B" />
      </zoneRule>
    </zoneMetadata>
  </mapSet>
</objectgridDeployment>
</deploymentPolicy>

```

ここには、プライマリー用に 1 つ (P)、レプリカ用に 1 つ (A) という 2 つのルールがあります。



## 広域ネットワーク (WAN) 上のゾーン

低速のネットワーク相互接続を使用する複数のビルまたはデータ・センターにまたがって、1 つの eXtreme Scale をデプロイしたい場合があります。ネットワーク接続が低速になれば、それだけ処理能力が低下し、接続待ち時間が長くなります。このモードでは、ネットワーク輻輳やその他の要因のために、ネットワーク分割の可能性も増します。eXtreme Scale は、以下の方法でこの厳しい環境に対処します。

### ゾーン間のハートビート処理の制限

コア・グループにグループ化された Java 仮想マシンは、互いにハートビートを実行します。カタログ・サービスが Java 仮想マシンをグループに編成した場合、こうしたグループはゾーンをまたぎません。そのグループ内のリーダーがメンバーシップ情報をカタログ・サービスにプッシュします。カタログ・サービスは報告された障害を検証してから、アクションを実行します。問題のある Java 仮想マシンとの接続を試みて、この処理を実行します。カタログ・サービスは、誤障害検出を認めた場合、何のアクションも実行しません。コア・グループ区画が短時間で回復するためです。

またカタログ・サービスは、定期的にコア・グループ・リーダーに低速でハートビートを行い、コア・グループ分離の症状を処理します。

### 優先ゾーン・ルーティング

優先ゾーン・ルーティングを使用して、WebSphere eXtreme Scale がトランザクションをゾーンに送信する方法を定義できます。

データ・グリッドの断片が配置される場所を制御します。いくつかの基本的なシナリオ、および、それに合わせたデプロイメント・ポリシーの構成方法について詳しくは、271 ページの『レプリカ配置のためのゾーンの構成』を参照してください。

優先ゾーン・ルーティングは、特定のゾーンあるいはゾーンのセットを優先する機能を WebSphere eXtreme Scale クライアントに付与します。結果として、クライアント・トランザクションを優先ゾーンに送付してから、他のゾーンにも送信することを試みます。

### 優先ゾーン・ルーティングの要件

優先ゾーン・ルーティングを試みる前に、アプリケーションがシナリオの要件を満たすことができることを確認してください。

優先ゾーン・ルーティングを使用するには、コンテナごとの区画の配置が必要です。この配置戦略はセッション・データを ObjectGrid に保管しようとしているアプリケーションには最適です。WebSphere eXtreme Scale におけるデフォルトの区画配置戦略は、fixed-partition です。トランザクションのコミット時にキーがハッシュされて、固定区画配置を使用する際に、マップのキー値ペアがどの区画に納められるかが決まります。

コンテナごとの配置により、データはトランザクション・コミット時に SessionHandle オブジェクトを介してランダムに区画に割り当てられます。データ・グリッドからデータを取得するには、この SessionHandle オブジェクトを再構成できなければなりません。



ゾーンを使用すると、プライマリー断片とレプリカ断片がドメインに配置される場所に対する制御を強化できます。デプロイメントで複数ゾーンを使用すると、データが複数の物理ロケーションに存在する場合に有利です。地理的にプライマリーとレプリカを分離させることは、1つのデータ・センターの壊滅的な損失でも、データの可用性に影響を与えないようにする1つの方法です。

データが複数ゾーンに散在される場合、クライアントもそのトポロジーに散在されると考えられます。クライアントをそれぞれのローカル・ゾーンあるいはデータ・センターにルーティングすることには、ネットワーク待ち時間の削減という明確なパフォーマンス上の利点があります。クライアントをローカル・ゾーン、または、可能であればデータ・センターにルーティングしてください。

## 優先ゾーン・ルーティングのトポロジーの構成

次のシナリオを考えてみます。データ・センターが2つ、Chicago と London にあります。クライアントの応答時間を最小にするために、クライアントはローカル・データ・センターに対してデータの読み取りと書き込みを行うようにします。

トランザクションが各ロケーションでローカルに書き込みが行われるためには、プライマリー断片は各データ・センターに配置される必要があります。クライアントは、ローカル・ゾーンへ経路指定するためにゾーンについて把握している必要があります。

コンテナごとの配置により、新しいプライマリー断片は、開始された各コンテナに配置されます。レプリカは、デプロイメント・ポリシーにより指定されたゾーンと配置のルールに従って配置されます。デフォルトで、レプリカは、プライマリー断片とは異なるゾーンに配置されます。このシナリオでは、以下のデプロイメント・ポリシーを考えてみます。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="universe">
    <mapSet name="mapSet1" placementStrategy="PER_CONTAINER"
      numberOfPartitions="3" maxAsyncReplicas="1">
      <map ref="planet" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

デプロイメント・ポリシーを使用して開始される各コンテナで、3つの新規プライマリーを受け取ります。各プライマリーには、非同期レプリカが1つ作成されます。各コンテナを適切なゾーン名で開始します。コンテナを **startOgServer** スクリプトを使用して開始する場合、**-zone** パラメーターを使用します。

Chicago のコンテナ・サーバーの場合、以下のようにします。

- **UNIX** **Linux**  

```
startOgServer.sh s1 -objectGridFile ../xml/universeGrid.xml
-deploymentPolicyFile ../xml/universeDp.xml
-catalogServiceEndpoints MyServer1.company.com:2809
-zone Chicago
```
- **Windows**

```

startOgServer.bat s1 -objectGridFile ../xml/universeGrid.xml
-deploymentPolicyFile ../xml/universeDp.xml
-catalogServiceEndPoints MyServer1.company.com:2809
-zone Chicago

```

ご使用のコンテナが WebSphere Application Server で実行されている場合、ノード・グループを作成してそれに「ReplicationZone」という接頭部を付けた名前を指定します。これらのノード・グループのノード上で実行中のサーバーは、適切なゾーンに配置されます。例えば、Chicago ノードで実行中のサーバーは、ReplicationZoneChicago という名前のノード・グループに含まれる可能性があります。

詳しくは、271 ページの『レプリカ配置のためのゾーンの構成』を参照してください。

Chicago ゾーンにプライマリ断片があるものについては、そのレプリカが London ゾーンにあります。London ゾーンにプライマリ断片があるものについては、そのレプリカが Chicago ゾーンにあります。

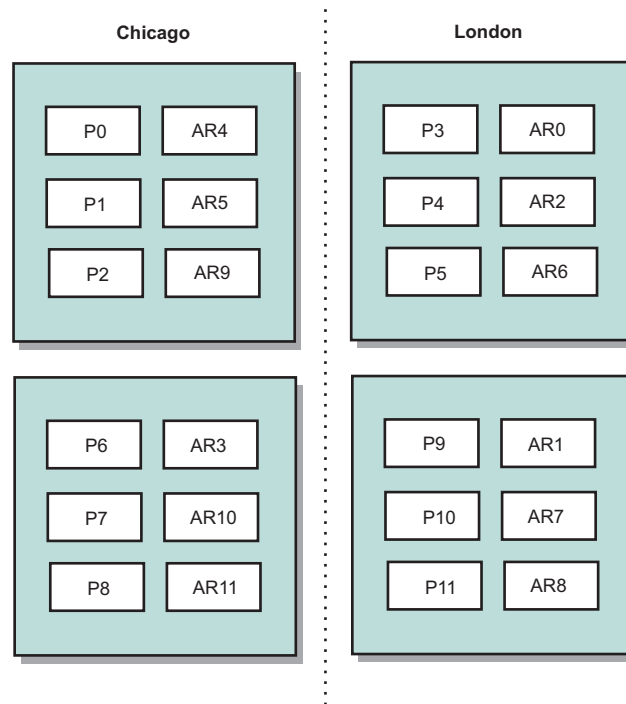


図 29. ゾーン内のプライマリとレプリカ

クライアントの優先ゾーンを設定します。クライアント・プロパティ・ファイルをクライアント Java 仮想マシン (JVM) に提供します。objectGridClient.properties という名前のファイルを作成し、このファイルが確実にクラスパスに入るようにします。

このファイルに **preferZones** プロパティを組み込みます。このプロパティ値を適切なゾーンに設定します。Chicago のクライアントは、objectGridClient.properties ファイルに以下の値を持っている必要があります。

```
preferZones=Chicago
```

London のクライアントのプロパティ・ファイルには、以下の値が含まれていなければなりません。

```
preferZones=London
```

このプロパティにより、各クライアントがトランザクションを可能な限りそのローカル・ゾーンに経路指定するように指示します。トポロジーは、ローカル・ゾーンのプライマリー断片に挿入されるデータを非同期で外部のゾーンに複製します。

## SessionHandle インスタンスを使用してローカル・ゾーンに経路指定する

コンテナごとの配置ストラテジーでは、データ・グリッド内のキー値ペアのロケーションを決定する場合にハッシュ・ベースのアルゴリズムを使用しません。この配置ストラテジーの使用時は、トランザクションが正しいロケーションに確実に経路指定されるように `SessionHandle` オブジェクトを使用する必要があります。トランザクションがコミットされると、`SessionHandle` オブジェクトがセッションにバインドされます (まだ設定されていない場合)。また、トランザクションをコミットする前に `Session.getSessionHandle` メソッドを呼び出すことによって `SessionHandle` オブジェクトをセッションにバインドすることができます。以下のコード・スニペットは、トランザクションのコミット前に `SessionHandle` がバインドされる場合を示しています。

```
Session ogSession = objectGrid.getSession();

// binding the SessionHandle
SessionHandle sessionHandle = ogSession.getSessionHandle();

ogSession.begin();
ObjectMap map = ogSession.getMap("planet");
map.insert("planet1", "mercury");

// tran is routed to partition specified by SessionHandle
ogSession.commit();
```

前のコードは、Chicago データ・センターのクライアントで実行されていたと想定してください。このクライアントの `preferZones` 属性は、Chicago に設定されています。結果として、このデプロイメントは、Chicago ゾーンのプライマリー区画 0、1、2、6、7、または 8 のいずれかにトランザクションを経路指定します。

`SessionHandle` オブジェクトは、このコミット済みデータを保管している区画に戻るパスを提供します。コミット済みデータを含む区画に戻るには、`SessionHandle` オブジェクトを再使用または再構成して、`Session` に対して設定する必要があります。

```
ogSession.setSessionHandle(sessionHandle);
ogSession.begin();

// value returned will be "mercury"
String value = map.get("planet1");
ogSession.commit();
```

このコードのトランザクションは、挿入トランザクション中に作成された `SessionHandle` オブジェクトを再使用します。次に、`get` トランザクションにより、挿入されたデータを保持する区画に経路指定されます。`SessionHandle` オブジェクトがないと、トランザクションは挿入されたデータを取得できません。

## コンテナおよびゾーン障害がゾーン・ベースのルーティングにどのように影響するか

一般に、**preferZones** プロパティが設定されているクライアントでは、すべてのトランザクションを指定されたゾーン (複数可) に経路指定します。ただし、コンテナの損失があると、レプリカ断片がプライマリ断片にプロモートします。ローカル・ゾーンの区画に既に経路指定されていたクライアントは、以前に挿入されたデータをリモート・ゾーンから取得する必要があります。

次のシナリオを考えてみます。Chicago ゾーンの 1 コンテナが損失したとします。このコンテナには区画 0、1、および 2 のプライマリが既に格納されています。London ゾーンでこれらの区画のレプリカをホストしたため、これらの区画の新しいプライマリ断片は London ゾーンに配置されます。

フェイルオーバーになった区画のいずれかを指す **SessionHandle** オブジェクトを使用している Chicago のクライアントは、今度は London に経路指定します。新規 **SessionHandle** オブジェクトを使用している Chicago のクライアントは、Chicago ベースのプライマリに経路指定します。

同様に、Chicago ゾーン全体が損失した場合、London ゾーンのすべてのレプリカがプライマリにプロモートされます。このシナリオでは、すべての Chicago クライアントがそのトランザクションを London に経路指定します。

## コンテナ・サーバーのゾーンの定義

ゾーンは、コンテナ・サーバーの集合です。コンテナ・サーバーは、1 つのゾーンにのみ所属できます。コンテナ・サーバーは、始動するときにゾーンに割り当てられます。

## このタスクについて

コンテナ・サーバーは始動時にゾーン・メンバーシップを定義するため、コンテナ・サーバーを始動する前にゾーンを計画する必要があります。コンテナ・サーバーのゾーン・メンバーシップを変更する場合は、新しいゾーン情報を使用してそのコンテナ・サーバーを再始動する必要があります。

## 手順

- **スタンドアロン・コンテナ・サーバーのゾーンを定義します。**
  1. **startOgServer** スクリプトの **-zone** パラメーターを使用して、始動したサーバーの中のすべてのコンテナのゾーンを指定します。サーバーの開始方法について詳しくは、457 ページの『**startOgServer** スクリプト』を参照してください。
  2. また、組み込みサーバー API を使用してプログラマチックにコンテナ・サーバーを始動するときに、ゾーン名を指定することができます。詳しくは、468 ページの『組み込みサーバー API を使用したサーバーの開始と停止』を参照してください。
- **WebSphere Application Server** 内で実行中のコンテナ・サーバーのゾーンを定義します。

ノード・グループを使用して、特定のゾーンにコンテナ・サーバーを配置することができます。「ReplicationZone<identifier>」という構文を使用して、ゾーンを割り当てるノード・グループに名前を付けます。デプロイメント・ポリシー内にゾーンを定義するときに、ノード・グループに付けたとおりの名前をゾーンに付ける必要があります。ノード・グループ名と、デプロイメント・ポリシー記述子 XML ファイル内のゾーン名は同じでなければなりません。

**重要:** WebSphere Application Server では、ノードが複数のノード・グループに存在してもかまいません。コンテナ・サーバーは 1 つのゾーン内にしか存在できないので、ノードは必ず 1 つの ReplicationZone ノード・グループ内に存在します。

例えば、4 つのノードを、A と B の 2 つのゾーンに分ける場合は次のようになります。

1. node1、node2、node3、node4 の 4 つのノードを構成します。各ノードには 2 つのサーバーがあります。
2. ReplicationZoneA という名前のノード・グループと ReplicationZoneB という名前のノード・グループを作成します。
3. node1 と node2 を ReplicationZoneA に追加し、node3 と node4 を ReplicationZoneB に追加します。
4. デプロイメント・ポリシー記述子 XML ファイル内に ReplicationZoneA と ReplicationZoneB を定義します。例については、284 ページの『例: WebSphere Application Server 環境内のゾーン』を参照してください。
5. node1 と node2 のサーバーが始動すると、それらのサーバーは ReplicationZoneA (WebSphere eXtreme Scale 構成内のゾーン A) に加わります。node3 と node4 のサーバーは、WebSphere eXtreme Scale 構成内のゾーン B として ReplicationZoneB に加わります。

### 例: デプロイメント・ポリシー記述子 XML ファイルを使用したゾーン定義

デプロイメント・ポリシー記述子 XML ファイルを使用して、ゾーンおよびゾーン・ルールを指定できます。

### 例: 異なるゾーンでのプライマリ断片およびレプリカ断片

この例は、1 つの非同期レプリカで、プライマリ断片を 1 つのゾーンに、レプリカ断片を別のゾーンに配置するものです。すべてのプライマリ断片は、DC1 ゾーンで開始します。レプリカ断片はゾーン DC2 で開始します。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd" xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="library">
    <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="0"
      maxSyncReplicas="0" maxAsyncReplicas="1">
      <map ref="book" />
      <zoneMetadata>
        <shardMapping shard="P" zoneRuleRef="primaryRule"/>
        <shardMapping shard="A" zoneRuleRef="replicaRule"/>
        <zoneRule name="primaryRule">
          <zone name="DC1" />
        </zoneRule>
      </zoneMetadata>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

```

    <zoneRule name="replicaRule">
    </zoneRule>
</zoneMetadata>
</mapSet>
</objectgridDeployment>
</deploymentPolicy>

```

1 つの非同期レプリカが ms1 mapSet エレメントの中で定義されます。したがって、各区画に、1 つのプライマリー断片と 1 つの非同期レプリカ断片の、2 つの断片が存在します。zoneMetadata エレメントの中で、断片ごとに shardMapping エレメントが定義されます。つまり、プライマリーには P が、非同期レプリカには DC1 が定義されます。primaryRule 属性はプライマリー断片のゾーン・セット (これがまさにゾーン DC1 です) を定義し、このルールはプライマリー断片の配置に使用されます。非同期レプリカは DC2 ゾーンに配置されます。

しかし、DC2 ゾーンが失われると、レプリカ断片は使用不可になります。DC1 ゾーンでコンテナ・サーバーが失われるか失敗すると、たとえレプリカが指定されていてもデータが損失する可能性があります。

この可能性に対処するには、次のセクションで説明しているように、ゾーンを追加するか、レプリカを追加します。

### 例: ゾーンの追加、断片のストライピング

次のコードは、新しいゾーンを構成します。

```

<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd" xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="library">
    <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="0"
      maxSyncReplicas="0" maxAsyncReplicas="1">
      <map ref="book" />
      <zoneMetadata>
        <shardMapping shard="P" zoneRuleRef="stripeRule"/>
        <shardMapping shard="A" zoneRuleRef="stripeRule"/>
        <zoneRule name="stripeRule" exclusivePlacement="true">
          <zone name="A" />
          <zone name="B" />
          <zone name="C" />
        </zoneRule>
      </zoneMetadata>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>

```

このコードでは、全部で 3 つのゾーン (A、B、および C) が定義されました。プライマリーとレプリカで別々のゾーン・ルールではなく、stripeRule という共有ゾーン・ルールが定義されています。このルールには、すべてのゾーンが含まれ、exclusivePlacement 属性は true に設定されています。eXtreme Scale 配置ポリシーによって、プライマリー断片とレプリカ断片は確実に別々のゾーンに配置されます。この配置のストライピングによって、プライマリー断片とレプリカ断片が、このポリシーに従って両方のゾーンに拡散することになります。3 目目のゾーン C を追加することで、いずれか 1 つのゾーンが失われてもデータは損失されず、各区画のプライマリー断片とレプリカ断片は依然として残ることになります。ゾーンが失敗すると、プライマリー断片かレプリカ断片のどちらかが失われるか、どちらも



失われぬという結果になります。失われた断片は、残っているゾーンにある残存断片で置き換えられ、もう一方の残っているゾーンに配置されます。

### 例: レプリカの追加および複数のデータ・センターの定義

古典的な 2 つのデータ・センターのシナリオは、各データ・センター内では高速で待ち時間が短いネットワークですが、データ・センター間の待ち時間は長くなります。同期レプリカは各データ・センター内で使用され、待ち時間が短いことにより、レプリカ生成が応答時間に与える影響が最小に抑えられます。データ・センター間では非同期レプリカ生成が使用されるため、ネットワークの長い待ち時間が応答時間に影響を及ぼすことはありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd" xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
<objectgridDeployment objectgridName="library">
  <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="1"
    maxSyncReplicas="1" maxAsyncReplicas="1">
    <map ref="book" />
    <zoneMetadata>
      <shardMapping shard="P" zoneRuleRef="primarySync"/>
      <shardMapping shard="S" zoneRuleRef="primarySync"/>
      <shardMapping shard="A" zoneRuleRef="async"/>
      <zoneRule name="primarySync" exclusivePlacement="false">
        <zone name="DC1" />
        <zone name="DC2" />
      </zoneRule>
      <zoneRule name="async" exclusivePlacement="true">
        <zone name="DC1" />
        <zone name="DC2" />
      </zoneRule>
    </zoneMetadata>
  </mapSet>
</objectgridDeployment>
</deploymentPolicy>
```

プライマリーと同期レプリカは、exclusivePlacement 属性設定 false で primarySync ルールを共有します。exclusivePlacement 属性が false に設定されていると、各区画のプライマリー断片と同期レプリカ断片を同じゾーンに配置する構成が作成されます。非同期レプリカ断片は、primarySync ゾーン・ルールとゾーンがほとんど同じである第 2 のゾーン・ルールを使用します。しかし、非同期レプリカは、true に設定された exclusivePlacement 属性を使用します。exclusivePlacement 属性は true に設定されると、断片を同じ区画の別の断片があるゾーンには配置できないことを意味します。結果的に、非同期レプリカ断片は、プライマリーまたは同期レプリカ断片と同じゾーンに配置されません。この mapSet には区画ごとに 3 つの断片 (プライマリー、同期レプリカ、および非同期レプリカ) があるため、3 つの shardMapping エレメント (各断片に 1 つ) があります。

ゾーンが失われると、非同期レプリカも失われます。非同期レプリカには独立したゾーンがないため、失われた非同期レプリカは再生成されません。プライマリー断片とレプリカ断片が失われると、残存した非同期レプリカがプライマリーにプロモートされ、ゾーン内に新しい同期レプリカが作成されます。プライマリーとレプリカは、各ゾーンの間でストライピングされます。

排他的配置の場合、各断片は独自のゾーンを持ちます。つまり、これらの断片をすべて独自のゾーンに配置できるだけの十分なゾーンを用意する必要があります。ル

ールに 1 つのゾーンがある場合は、そのゾーンに断片を 1 つしか配置できません。2 つのゾーンがあれば、最大 2 つの断片をそのゾーンに配置できます。

## 例: WebSphere Application Server 環境内のゾーン

次のコードは、新しいゾーンを構成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd" xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="library">
    <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="0"
      maxSyncReplicas="0" maxAsyncReplicas="1">
      <map ref="book" />
      <zoneMetadata>
        <shardMapping shard="P" zoneRuleRef="stripeRule"/>
        <shardMapping shard="A" zoneRuleRef="stripeRule"/>
        <zoneRule name="stripeRule" exclusivePlacement="true">
          <zone name="ReplicationZoneA" />
          <zone name="ReplicationZoneB" />
          <zone name="ReplicationZoneC" />
        </zoneRule>
      </zoneMetadata>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

この例では、ReplicationZoneA、ReplicationZoneB、ReplicationZoneC の 3 つのノード・グループが WebSphere Application Server 環境の中で定義されます。ノード・グループ名と、デプロイメント・ポリシー記述子 XML ファイル内のゾーン名は同じでなければならず、テキスト ReplicationZone<identifier> を含んでいなければなりません。このファイルは、断片のストライピングの例と類似した構成を定義しますが、WebSphere Application Server 構成の必要な命名を示します。

## xscmd ユーティリティーによるゾーン情報の表示

**xscmd** サンプル・ユーティリティーを使用して、断片配置データも含めた現在のゾーン・デプロイメントについての情報を表示できます。

### 始める前に

- 複数のデータ・センターがある分散データ・グリッドをデプロイします。詳しくは、276 ページの『優先ゾーン・ルーティング』を参照してください。

### このタスクについて

製品に付属する **xscmd** ユーティリティーを使用して、ゾーン設定に関連する構成についての情報を判別できます。

### 手順

**xscmd** ユーティリティーを使用して、データの断片についての情報を判別します。以下のコマンドを実行します。

```
xscmd -c showPlacement -z zone_name
```

## 例

開始用 (getting started) サンプル `wxs_install_root/ObjectGrid/gettingstarted` を使用して、さらに単純なシナリオを実行することもできます。詳しくは、1 ページの『チュートリアル: WebSphere eXtreme Scale 入門』を参照してください。

1. カタログ・サーバーを始動します。

```
runcat.bat
```

2. 例えば次のようなコマンドを使用して、必要なレプリカの数、ゾーン・ルール、コンテナ、その他の設定を決定します。`startOgServer.bat serverA0 -objectgridFile xml¥objectgrid.xml -deploymentPolicyFile xml¥deployment.xml -zone zoneA`
3. データ・グリッド内の障害をシミュレートするために、コンテナ・プロセスを停止できます。`stopOgServer.bat serverA0,serverA1,serverB0 -catalogServiceEndPoints localhost:2809`

区画の最後の断片を含むサーバーが停止した場合、eXtreme Scale は、新しいプライマリー断片を割り振ります。データ損失を確認することができます。

- **runclient** スクリプトは、データ・グリッドに項目を挿入したり、データ・グリッドから項目を読み取ったりします。
  - **xscmd -c showMapSizes** コマンドは、データ・グリッド内の項目数を示します。
4. 次のコマンドを使用して、アクティブなコンテナ・サーバーを表示します。

```
xscmd -c showPlacement -z zone_name
```

---

## カタログ・サーバーおよびコンテナ・サーバーの構成

WebSphere eXtreme Scale には、カタログ・サーバーとコンテナ・サーバーの 2 タイプのサーバーがあります。カタログ・サーバーは、断片の配置を制御し、コンテナ・サーバーの検出とモニターをします。複数のカタログ・サーバーがカタログ・サービス・ドメインと結合して、環境に高可用性を提供することができます。コンテナ・サーバーは、データ・グリッドのアプリケーション・データを保管する Java 仮想マシン (JVM) です。

## カタログ・サーバーおよびカタログ・サービス・ドメインの構成

カタログ・サービスは、一般的に定常状態でアイドルになるロジックをホストします。その結果として、カタログ・サービスがスケラビリティに与える影響はごくわずかです。サービスは、同時に使用可能になる多数のコンテナ・サーバーにサービスを提供するために作成されています。高可用性のために、カタログ・サービスをカタログ・サービス・ドメインに構成します。

### 始める前に

カタログ・サービス・ドメインが開始されると、データ・グリッドのメンバーは相互にバインドされます。カタログ・サービス・ドメイン構成を実行時に変更することはできないので、カタログ・サービス・ドメインのトポロジは慎重に計画してください。エラー防止のため、データ・グリッドはできるだけ広範囲に分散させてください。

カタログ・サービス・ドメインで単一障害点を回避するためのベスト・プラクティスは、3 つの異なるノード上で少なくとも 3 つのカタログ・サーバーを始動することです。

ノードを 2 つしか使用していない場合、2 つのノード上にカタログ・サーバーを 2 つずつ、合計 4 つのカタログ・サーバー・プロセスを構成します。この構成を作成すると、ノードが 1 つしか開始されない場合でも、必要な 2 台のカタログ・サーバーが実行されるようになります。少なくとも 2 台のカタログ・サーバーを同時に始動する必要があります。カタログ・サーバーは、始動すると、構成内の他のカタログ・サーバーを探し、少なくとももう 1 つのカタログ・サーバーが検出されるまで正常に始動しません。

## このタスクについて

### スタンドアロン・カタログ・サーバーおよびカタログ・サービス・ドメイン

**startOgServer** コマンドまたは組み込みサーバー API に渡すパラメーターおよびプロパティ・ファイルを使用して、スタンドアロン・カタログ・サーバーとカタログ・サービス・ドメインを構成します。

- 『例: カタログ・サービス・ドメインの構成』
- 451 ページの『スタンドアロン・カタログ・サービスの開始』
- カタログ・サーバー・プロパティ
- 457 ページの『**startOgServer** スクリプト』

### WebSphere Application Server カタログ・サーバーおよびカタログ・サービス・ドメイン

WebSphere Application Server 内で稼働するカタログ・サーバーは、WebSphere Application Server 管理コンソール、管理用タスク、およびサーバー・プロパティ・ファイルを使用して構成します。サーバーのライフサイクルは、WebSphere Application Server 内のプロセス・ライフサイクルによって制御されます。プロセスが WebSphere Application Server 内で開始または停止すると、そのプロセスで実行されているカタログ・サーバーも開始または停止します。

- 288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』
- 288 ページの『WebSphere Application Server でのカタログ・サービスの構成』

### 例: カタログ・サービス・ドメインの構成

カタログ・サービスを使用する場合、単一障害点を回避するには少なくとも 2 台のカタログ・サーバーが必要です。少なくとも 2 台のカタログ・サーバーが常に実行されているようにするために、環境内のノード数に応じてさまざまな構成を作成することができます。

### 例: スタンドアロン環境の 2 つのノード上での 4 台のカタログ・サーバーの始動

以下のスクリプトにより、host1 ノード上でカタログ・サーバー cs0 および cs1 を始動し、host2 ノード上でカタログ・サーバー cs2 および cs3 を始動します。

```
./startOgServer.sh|bat cs0 -listenerPort 2809 -catalogServiceEndPoints  
cs0:host1:6601:6602,cs1:host1:6603:6604,cs2:host2:6601:6602,cs3:host2:6603:6604  
-quorum true -jvmArgs -Xmx256m
```

```
./startOgServer.sh|bat cs1 -listenerPort 2810 -catalogServiceEndPoints  
cs0:host1:6601:6602,cs1:host1:6603:6604,cs2:host2:6601:6602,cs3:host2:6603:6604  
-quorum true -jvmArgs -Xmx256m
```

```
./startOgServer.sh|bat cs2 -listenerPort 2809 -catalogServiceEndPoints  
cs0:host1:6601:6602,cs1:host1:6603:6604,cs2:host2:6601:6602,cs3:host2:6603:6604  
-quorum true -jvmArgs -Xmx256m
```

```
./startOgServer.sh|bat cs3 -listenerPort 2810 -catalogServiceEndPoints  
cs0:host1:6601:6602,cs1:host1:6603:6604,cs2:host2:6601:6602,cs3:host2:6603:6604  
-quorum true -jvmArgs -Xmx256m
```

**要確認:** ノード上で実行中のカタログ・サーバーにはそれぞれ固有のポート番号が必要であるため、**-listenerPort** オプションを使用する必要があります。

### 例: WebSphere Application Server 環境での複数のカタログ・サーバーの始動

カタログ・サーバーは、WebSphere Application Server 環境で自動的に始動します。カタログ・サービス・ドメインを作成することにより、始動する複数のカタログ・サーバーを定義できます。カタログ・サービス・ドメインで複数のエンドポイントを指定した後に、カタログ・サーバーが並行して始動するように、含まれているアプリケーション・サーバーを再始動します。

- **WebSphere Application Server Network Deployment:** 複数の既存のアプリケーション・サーバーをカタログ・サービス・ドメインのメンバーにするためにセルから選択できます。
- **基本 WebSphere Application Server:** 複数のスタンドアロン・ノード上のカタログ・サービスを開始できます。プロファイル管理ツールを使用して、複数のプロファイルを同じインストール・イメージ上に定義することによって、それぞれに固有のポートが割り当てられた、一連のスタンドアロン・ノードを作成できます。各アプリケーション・サーバーで、カタログ・サービス・ドメインを定義します。リモート・サーバーをこの構成に追加することによって、他のすべてのアプリケーション・サーバーを指定できます。この構成をすべてのスタンドアロン・サーバー上で作成したら、**startServer** スクリプトを実行するか、Windows サービスを使用してサーバーを始動することにより、一連の基本アプリケーション・サーバーを並行して始動できます。

### WebSphere eXtreme Scale と WebSphere Application Server の構成

WebSphere Application Server でカタログ・サービスおよびコンテナ・サーバー・プロセスを実行できます。これらのサーバーを構成するプロセスは、スタンドアロン構成の場合とは異なります。カタログ・サービスは、WebSphere Application Server サーバーまたはデプロイメント・マネージャーで自動的に開始できます。eXtreme Scale アプリケーションが WebSphere Application Server 環境にデプロイされて、開始されるときに、コンテナ・プロセスは開始されます。

## このタスクについて

**重要:** 実稼働環境では、コンテナ・サーバーをカタログ・サーバーと連結しないようにしてください。カタログ・サービスを、複数のノード・エージェント・プロセス、または eXtreme Scale アプリケーションをホスティングしていないアプリケーション・サーバーに組み込んでください。

### WebSphere Application Server でのカタログ・サービスの構成:

カタログ・サービス・プロセスは、WebSphere Application Server 内で実行できません。WebSphere Application Server 内でのサーバーのライフサイクルに従って、いつカタログ・サービスが開始または停止するかが決まります。

#### 手順

1. WebSphere eXtreme Scale プロファイルの拡張に使用する WebSphere Application Server プロセスを 1 つ以上選択します。詳しくは、225 ページの『WebSphere eXtreme Scale のプロファイルの作成および拡張』を参照してください。  
WebSphere Application Server Network Deployment のデプロイメント・マネージャーでカタログ・サービスを自動的に開始するには、WebSphere eXtreme Scale をデプロイメント・マネージャー・ノードにインストールし、デプロイメント・マネージャー・プロファイルを拡張してください。
2. WebSphere Application Server プロセスのサーバー・プロパティ・ファイルを構成し、ノードのクラスパスに追加します。詳しくは、サーバー・プロパティ・ファイルを参照してください。
3. カタログ・サービス・ドメインを構成します。カタログ・サービス・ドメインは、環境内にあるカタログ・サーバーのグループです。詳しくは、『WebSphere Application Server でのカタログ・サービス・ドメインの作成』を参照してください。
4. カタログ・サーバーをホスティングしている WebSphere Application Server プロセスを開始します。詳しくは、467 ページの『WebSphere Application Server 環境でのサーバーの開始と停止』を参照してください。

### WebSphere Application Server でのカタログ・サービス・ドメインの作成:

カタログ・サービス・ドメインは、断片の配置を管理し、データ・グリッド内のコンテナ・サーバーのヘルスをモニターするカタログ・サーバーのグループを定義します。

#### 始める前に

- WebSphere eXtreme Scale を WebSphere Application Server にインストールします。詳しくは、205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。

## このタスクについて

カタログ・サービス・ドメインを作成することで、カタログ・サーバーの高可用性コレクションが定義されます。



これらのカタログ・サーバーは、単一のセルおよびコア・グループ内の WebSphere Application Server で実行できます。カタログ・サービス・ドメインは、異なる Java SE プロセスまたは他の WebSphere Application Server セルで実行されるサーバーのリモート・グループも定義できます。

**セル内の既存のアプリケーション・サーバーで稼働するカタログ・サーバーの場合:**  
セル内のアプリケーション・サーバーにカタログ・サーバーを配置するカタログ・サービス・ドメインを定義した場合には、WebSphere Application Server のコア・グループ・メカニズムが使用されます。セル内のアプリケーション・サーバーで、カタログ・サービスが自動的に開始されます。結果として、単一のカタログ・サービス・ドメインのメンバーがコア・グループの境界にまたがることできないため、カタログ・サービス・ドメインは複数のセルにまたがることはできません。ただし、WebSphere eXtreme Scale コンテナ・サーバーおよびクライアントは、セル境界を越えてカタログ・サーバー (スタンドアロン・カタログ・サービス・ドメインや別のセルに組み込まれたカタログ・サービス・ドメインなど) に接続することで、複数のセルにまたがることはできます。

**リモート・カタログ・サーバーの場合:** 別の WebSphere Application Server セル内で稼働中か、スタンドアロン・プロセスとして稼働中のカタログ・サービス・ドメインに、WebSphere eXtreme Scale コンテナおよびクライアントを接続できます。リモートで構成されたカタログ・サーバーはセルの中で自動的に始動しないため、リモートで構成されたカタログ・サーバーは、すべて手動で始動する必要があります。リモート・カタログ・サービス・ドメインを構成する場合、ドメイン名は、リモート・カタログ・サーバーの始動時に指定するドメイン名と一致している必要があります。スタンドアロン・カタログ・サーバーのカタログ・サービスのデフォルトのドメイン名は、DefaultDomain です。カタログ・サービスのドメイン名は、**startOgServer** コマンド **-domain** パラメーター、サーバー・プロパティ・ファイル、または組み込まれたサーバー API を使用して指定します。リモート・ドメイン内の各リモート・カタログ・サーバー・プロセスは、同じドメイン名を使用して始動する必要があります。カタログ・サーバーの始動について詳しくは、451 ページの『スタンドアロン・カタログ・サービスの開始』を参照してください。

**重要:** 実稼働環境では、カタログ・サービスを WebSphere eXtreme Scale コンテナ・サーバーと連結しないようにしてください。カタログ・サービスを、複数のノード・エージェント・プロセス、または WebSphere eXtreme Scale アプリケーションをホストしていないアプリケーション・サーバーに組み込んでください。

## 手順

1. カタログ・サービス・ドメインを作成します。
  - a. WebSphere Application Server 管理コンソールで、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」 > 「新規」をクリックします。
  - b. カタログ・サービス・ドメインの名前、デフォルト値、JMX 認証資格情報を定義します。カタログ・サービス・ドメインのリモート・エンドポイントを構成する場合、カタログ・サービス・ドメインの名前は、カタログ・サーバーの始動時に指定するカタログ・サービス・ドメインの名前と一致している必要があります。

- c. カタログ・サーバー・エンドポイントを追加します。既存のアプリケーション・サーバーを選択するか、カタログ・サービスを実行しているリモート・サーバーを追加することができます。
2. カタログ・サービス・ドメイン内のカタログ・サーバーへの接続をテストします。既存のアプリケーション・サーバーの場合、カタログ・サーバーは、関連するアプリケーション・サーバーを開始する際に始動します。リモート・アプリケーション・サーバーの場合、**startOgServer** コマンドまたは組み込まれたサーバー API を使用して、手動でサーバーを始動する必要があります。
    - a. WebSphere Application Server 管理コンソールで、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」をクリックします。
    - b. テストするカタログ・サービス・ドメインを選択して、「テスト接続」をクリックします。このボタンをクリックすると、すべての定義されたカタログ・サービス・ドメイン・エンドポイントが 1 つずつ照会されます。いずれかのエンドポイントが使用可能であれば、カタログ・サービス・ドメインへの接続が成功したことを示すメッセージが返されます。

#### カタログ・サービス・ドメイン管理用タスク:

Jacl または Jython スクリプト言語を使用して、WebSphere Application Server 構成内のカタログ・サービス・ドメインを管理できます。

#### 要件

WebSphere Application Server 環境に WebSphere eXtreme Scale クライアントをインストールしている必要があります。

#### すべての管理用タスクのリスト

カタログ・サービス・ドメインに関連したすべての管理用タスクのリストを取得するには、`wsadmin` で以下のコマンドを実行します。

```
wsadmin>$AdminTask help XSDomainManagement
```

#### コマンド

カタログ・サービス・ドメインの管理用タスクには、以下のコマンドが含まれます。

- 291 ページの『createXSDomain』
- 294 ページの『deleteXSDomain』
- 294 ページの『getDefaultXSDomain』
- 295 ページの『listXSDomains』
- 295 ページの『modifyXSDomain』
- 301 ページの『testXSDomainConnection』
- 301 ページの『testXSSTestServerConnection』

## createXSDomain

**createXSDomain** コマンドは、新規カタログ・サービス・ドメインを登録します。

表 16. *createXSDomain* コマンド引数

引数	説明
<b>-name</b> (必須)	作成するカタログ・サービス・ドメインの名前を指定します。
<b>-default</b>	カタログ・サービス・ドメインがセルでデフォルトかどうかを指定します。デフォルト値は <code>true</code> です。(ブール値: <code>true</code> または <code>false</code> に設定)
<b>-properties</b>	カタログ・サービス・ドメインのカスタム・プロパティを指定します。

表 17. *defineDomainServers* ステップ引数

引数	説明
<i>name_of_endpoint</i>	カタログ・サービス・エンドポイントの名前を指定します。 <ul style="list-style-type: none"><li>• <b>既存のアプリケーション・サーバーの場合:</b> エンドポイントの名前は、<code>cell_name%node_name%server_name</code> の形式でなければなりません。</li><li>• <b>リモート・サーバーの場合:</b> リモート・サーバーのホスト名を指定します。複数のエンドポイントを同じ名前にすることができますが、クライアント・ポートの値はそれぞれのエンドポイントで固有でなければなりません。</li></ul>
<i>custom_properties</i>	カタログ・サービス・ドメイン・エンドポイントのカスタム・プロパティを指定します。カスタム・プロパティがない場合、この引数には一組の二重引用符 (") を使用します。

表 17. *defineDomainServers* ステップ引数 (続き)

引数	説明
<i>endpoint_ports</i>	<p>カタログ・サービス・ドメイン・エンドポイントのポート番号を指定します。ポートは、 &lt;<i>client_port</i>&gt;,&lt;<i>listener_port</i>&gt; の順序で指定する必要があります。</p> <p><b>クライアント・ポート</b>            カatalog・サービス・ドメイン内のカタログ・サーバー間の通信に使用するポートを指定します。この値は、WebSphere Application Server プロセスのみで稼働するカタログ・サーバーに必要で、他で使用されていないどのポートにも設定できます。</p> <p><b>リスナー・ポート</b>            クライアントとの通信に使用するポートを指定します。この値はリモート・エンドポイントに必要で、カタログ・サービスが開始されたときに使用された値と一致している必要があります。リスナー・ポートは、カタログ・サービスとの通信のために、クライアントおよびコンテナによって使用されます。</p> <p><b>WebSphere eXtreme Scale のリモート・エンドポイントの場合:</b> コンテナおよびクライアントがオブジェクト・リクエスト・ブローカー (ORB) を介してカタログ・サービスと通信するための ORB リスナー・ポートを定義します。WebSphere Application Server エンドポイントの場合、リスナー・ポート値は、<b>BOOTSTRAP_ADDRESS</b> ポート構成から継承されるためオプションです。</p>

表 18. *configureClientSecurity* ステップ引数

引数	説明
<b>-securityEnabled</b>	<p>カタログ・サーバーのクライアント・セキュリティーが使用可能なことを指定します。選択したカタログ・サーバーに関連付けられたサーバー・プロパティ・ファイルには、一致する <b>securityEnabled</b> 設定がなければなりません。これらの設定が一致しない場合、例外が発生します。(ブール値: true または false に設定)</p>

表 18. *configureClientSecurity* ステップ引数 (続き)

引数	説明
<b>-credentialAuthentication</b> (オプション)	資格情報の認証を実施するか、またはサポートするかを示します。  <b>常になし</b> クライアント証明書認証は実施されません。  <b>必須</b> 資格情報の認証は必ず実施されます。サーバーが資格情報の認証をサポートしない場合、クライアントはサーバーに接続できません。  <b>サポートされる</b> (デフォルト) 資格情報の認証は、クライアントとサーバーの両方が資格情報の認証をサポートする場合のみ実施されます。
<b>-authenticationRetryCount</b> (オプション)	資格情報の有効期限が切れている場合に認証を再試行できる回数を指定します。  認証の再試行を望まない場合は、値を 0 に設定します。デフォルト値は 0 です。
<b>-credentialGeneratorClass</b>	クライアントがスレッドからセキュリティー・トークンを取得するよう、 <code>com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator</code> 実装クラスを指示します。
<b>-credentialGeneratorProps</b>	<code>CredentialGenerator</code> 実装クラスのプロパティを指定します。プロパティは、 <code>setProperties(String)</code> メソッドを使用してオブジェクトに送信されます。資格情報生成プログラムのプロパティ値は、「資格情報生成プログラム・クラス」フィールドに値が指定されている場合のみ使用されます。

### 戻り値:

#### バッチ・モードの使用例

バッチ・モードの場合、コマンド項目が正しくフォーマットされていることが必要です。入力する値が適切に処理されるように、対話モードの使用を検討してください。バッチ・モードを使用する場合、特定のプロパティ配列を使用して **-defineDomainServers** のステップ引数を定義する必要があります。このプロパティ配列のフォーマットは、`name_of_endpoint custom_properties endpoint_ports` です。 `endpoint_ports` 値は、`<client_port>,<listener_port>` の順序で指定する必要があるポートのリストです。

- `Jacl` を使用した、リモート・エンドポイントのカatalog・サービス・ドメインの作成:

```
$AdminTask createXSDomain {-name TestDomain -default true -defineDomainServers
{{xhost1.ibm.com "" ,2809}} -configureClientSecurity {-securityEnabled false
-credentialAuthentication Required -authenticationRetryCount 0 -credentialGeneratorClass
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
-credentialGeneratorProps "manager manager1"}}
```

- `Jython` ストリングを使用した、リモート・エンドポイントのカatalog・サービス・ドメインの作成:

```
AdminTask.createXSDomain('[ -name TestDomain -default true
-defineDomainServers [[xhost1.ibm.com "" ,2809]
[xhost2.ibm.com "" ,2809]] -configureClientSecurity [-securityEnabled false
-credentialAuthentication Required -authenticationRetryCount 0 -credentialGeneratorClass
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
-credentialGeneratorProps "manager manager1" ]')
```

- Jacl を使用した、既存のアプリケーション・サーバー・エンドポイントのカタログ・サービス・ドメインの作成:

```
$AdminTask createXSDomain { -name TestDomain -default true -defineDomainServers
{{cellName/nodeName/serverName "" 1109}}
```

#### 対話モードの使用例

- Jacl を使用:

```
$AdminTask createXSDomain {-interactive}
```

- Jython スtringを使用:

```
AdminTask.createXSDomain ('[-interactive]')
```

#### deleteXSDomain

**deleteXSDomain** コマンドは、カタログ・サービス・ドメインを削除します。

必須パラメーター:

##### -name

削除するカタログ・サービス・ドメインの名前を指定します。

戻り値:

#### バッチ・モードの使用例

- Jacl を使用:

```
$AdminTask deleteXSDomain { -name TestDomain }
```

- Jython スtringを使用:

```
AdminTask.deleteXSDomain ('[-name TestDomain ]')
```

#### 対話モードの使用例

- Jacl を使用:

```
$AdminTask deleteXSDomain {-interactive}
```

- Jython スtringを使用:

```
AdminTask.deleteXSDomain ('[-interactive]')
```

#### getDefaultXSDomain

**getDefaultXSDomain** コマンドは、セルのデフォルト・カタログ・サービス・ドメインを返します。

必須パラメーター: なし

戻り値: デフォルト・カタログ・サービス・ドメインの名前。

#### バッチ・モードの使用例

- Jacl を使用:

```
$AdminTask getDefaultXSDomain
```



- Jython ストリングを使用:  
AdminTask.getDefaultXSDomain

#### 対話モードの使用例

- Jacl を使用:  
\$AdminTask getDefaultXSDomain {-interactive}
- Jython ストリングを使用:  
AdminTask.getDefaultXSDomain ('[-interactive]')

#### listXSDomains

**listXSDomains** コマンドは、既存のカタログ・サービス・ドメインのリストを返します。

**必須パラメーター:** なし

**戻り値:** セル内のすべてのカタログ・サービス・ドメインのリスト。

#### バッチ・モードの使用例

- Jacl を使用:  
\$AdminTask listXSDomains
- Jython ストリングを使用:  
AdminTask.listXSDomains

#### 対話モードの使用例

- Jacl を使用:  
\$AdminTask listXSDomains {-interactive}
- Jython ストリングを使用:  
AdminTask.listXSDomains ('[-interactive]')

#### modifyXSDomain

**modifyXSDomain** コマンドは、既存のカタログ・サービス・ドメインを変更します。

バッチ・モードの場合、コマンド項目が正しくフォーマットされていることが必要です。入力する値が適切に処理されるように、対話モードの使用を検討してください。バッチ・モードを使用する場合、特定のプロパティ配列を使用して **-modifyEndpoints**、**-addEndpoints**、および **-removeEndpoints** ステップ引数を定義する必要があります。このプロパティ配列のフォーマットは、*name\_of\_endpoint host\_name custom\_properties endpoint\_ports* です。 *endpoint\_ports* 値は、*<client\_port>,<listener\_port>* の順序で指定する必要があるポートのリストです。

表 19. *modifyXSDomain* コマンド引数

引数	説明
<b>-name</b> (必須)	編集するカタログ・サービス・ドメインの名前を指定します。

表 19. *modifyXSDomain* コマンド引数 (続き)

引数	説明
<b>-default</b>	true に設定した場合、選択したカタログ・サービス・ドメインがセルのデフォルトであることを指定します。(ブール値)
<b>-properties</b>	カタログ・サービス・ドメインのカスタム・プロパティを指定します。

表 20. *modifyEndpoints* ステップ引数

引数	説明
<i>name_of_endpoint</i>	<p>カタログ・サービス・エンドポイントの名前を指定します。</p> <ul style="list-style-type: none"> <li>• <b>既存のアプリケーション・サーバーの場合:</b> エンドポイントの名前は、<i>cell_name</i>¥<i>node_name</i>¥<i>server_name</i> の形式でなければなりません。</li> <li>• <b>リモート・サーバーの場合:</b> リモート・サーバーのホスト名を指定します。複数のエンドポイントを同じ名前にすることができますが、リスナー・ポートの値はそれぞれのエンドポイントで固有でなければなりません。</li> </ul>

表 20. `modifyEndpoints` ステップ引数 (続き)

引数	説明
<code>endpoint_ports</code>	<p>カタログ・サービス・ドメイン・エンドポイントのポート番号を指定します。エンドポイントは、<code>&lt;client_port&gt;</code>,<code>&lt;listener_port&gt;</code> の順序で指定する必要があります。</p> <p><b>クライアント・ポート</b>            カatalog・サービス・ドメイン内のカタログ・サーバー間の通信に使用するポートを指定します。この値は、WebSphere Application Server プロセスのみで稼働するカタログ・サーバーに必要で、他で使用されていないどのポートにも設定できません。</p> <p><b>リスナー・ポート</b>            クライアントとの通信に使用するポートを指定します。この値はリモート・エンドポイントに必要で、カタログ・サービスが開始されたときに使用された値と一致している必要があります。リスナー・ポートは、カタログ・サービスとの通信のために、クライアントおよびコンテナによって使用されます。</p> <p><b>WebSphere eXtreme Scale のリモート・エンドポイントの場合:</b> コンテナおよびクライアントがオブジェクト・リクエスト・ブローカー (ORB) を介してカタログ・サービスと通信するための ORB リスナー・ポートを定義します。WebSphere Application Server エンドポイントの場合、値が <code>BOOTSTRAP_ADDRESS</code> ポート構成から継承されるため、リスナー・ポート値の指定はオプションです。</p>

表 21. *addEndpoints* ステップ引数

引数	説明
<i>name_of_endpoint</i>	<p>カタログ・サービス・エンドポイントの名前を指定します。</p> <ul style="list-style-type: none"> <li>• <b>既存のアプリケーション・サーバーの場合:</b> エンドポイントの名前は、 <i>cell_name</i>¥<i>node_name</i>¥<i>server_name</i> の形式でなければなりません。</li> <li>• <b>リモート・サーバーの場合:</b> リモート・サーバーのホスト名を指定します。複数のエンドポイントを同じ名前にすることができますが、リスナー・ポートの値はそれぞれのエンドポイントで固有でなければなりません。</li> </ul>
<i>custom_properties</i>	<p>カタログ・サービス・ドメイン・エンドポイントのカスタム・プロパティを指定します。カスタム・プロパティがない場合、この引数には一組の二重引用符 (") を使用します。</p>

表 21. *addEndpoints* ステップ引数 (続き)

引数	説明
<i>endpoint_ports</i>	<p>カタログ・サービス・ドメイン・エンドポイントのポート番号を指定します。エンドポイントは、<i>&lt;client_port&gt;</i>,<i>&lt;listener_port&gt;</i> の順序で指定する必要があります。</p> <p><b>クライアント・ポート</b>            カatalog・サービス・ドメイン内のカタログ・サーバー間の通信に使用するポートを指定します。この値は、WebSphere Application Server プロセスのみで稼働するカタログ・サーバーに必要で、他で使用されていないどのポートにも設定できます。</p> <p><b>リスナー・ポート</b>            クライアントとの通信に使用するポートを指定します。この値はリモート・エンドポイントに必要で、カタログ・サービスが開始されたときに使用された値と一致している必要があります。リスナー・ポートは、カタログ・サービスとの通信のために、クライアントおよびコンテナによって使用されます。</p> <p><b>WebSphere eXtreme Scale のリモート・エンドポイントの場合:</b> コンテナおよびクライアントがオブジェクト・リクエスト・ブローカー (ORB) を介してカタログ・サービスと通信するための ORB リスナー・ポートを定義します。WebSphere Application Server エンドポイントの場合、値が <i>BOOTSTRAP_ADDRESS</i> ポート構成から継承されるため、リスナー・ポート値の指定はオプションです。</p>

表 22. *removeEndpoints* ステップ引数

引数	説明
<i>name_of_endpoint</i>	削除するカタログ・サービス・エンドポイントの名前を指定します。

表 23. `configureClientSecurity` ステップ引数

引数	説明
<code>-securityEnabled</code>	カタログ・サーバーのクライアント・セキュリティーが使用可能なことを指定します。選択したカタログ・サーバーに関連付けられたサーバー・プロパティー・ファイルには、一致する <code>securityEnabled</code> 設定がなければなりません。これらの設定が一致しない場合、例外が発生します。 (プール値: true または false に設定)
<code>-credentialAuthentication</code> (オプション)	資格情報の認証を実施するか、またはサポートするかを示します。 <b>常になし</b> クライアント証明書認証は実施されません。 <b>必須</b> 資格情報の認証は必ず実施されます。サーバーが資格情報の認証をサポートしない場合、クライアントはサーバーに接続できません。 <b>サポートされる</b> (デフォルト) 資格情報の認証は、クライアントとサーバーの両方が資格情報の認証をサポートする場合のみ実施されます。
<code>-authenticationRetryCount</code> (オプション)	資格情報の有効期限が切れている場合に認証を再試行できる回数を指定します。  認証の再試行を望まない場合は、値を 0 に設定します。デフォルト値は 0 です。
<code>-credentialGeneratorClass</code>	クライアントがスレッドからセキュリティー・トークンを取得するよう、 <code>com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator</code> 実装クラスを指示します。
<code>-credentialGeneratorProps</code>	<code>CredentialGenerator</code> 実装クラスのプロパティーを指定します。プロパティーは、 <code>setProperties(String)</code> メソッドを使用してオブジェクトに送信されます。資格情報生成プログラムのプロパティー値は、「資格情報生成プログラム・クラス」フィールドに値が指定されている場合のみ使用されます。

戻り値:

#### バッチ・モードの使用例

- Jacl を使用:

```
$AdminTask modifyXSDomain {-name TestDomain -default true -modifyEndpoints
  {{xhost1.ibm.com "" ,2809}} -addEndpoints {{xhost2.ibm.com "" ,2809}}
  -removeEndpoints {{xhost3.ibm.com}}}
```

- Jython スtringを使用:

```
AdminTask.modifyXSDomain('[-name TestDomain
  -default false -modifyEndpoints [[xhost1.ibm.com "" ,2809]]
  -addEndpoints [[xhost3.ibm.com "" ,2809]]
  -removeEndpoints [[xhost2.ibm.com]]]')
```

- 変更コマンドでのクライアント・セキュリティー仕様の使用:



```
$AdminTask modifyXSDomain {-name myDomain -default false
-configureClientSecurity {-securityEnabled true -
Supported -authenticationRetryCount 1 -credentialGeneratorClass
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
-credentialGeneratorProps "manager manager1"}}
```

### 対話モードの使用例

- Jacl を使用:

```
$AdminTask modifyXSDomain {-interactive}
```

- Jython スtringを使用:

```
AdminTask.modifyXSDomain ('[-interactive]')
```

### testXSDomainConnection

**testXSDomainConnection** コマンドは、カタログ・サービス・ドメインへの接続をテストします。

#### 必須パラメーター:

##### -name

接続をテストするカタログ・サービス・ドメインの名前を指定します。

#### オプション・パラメーター

##### -timeout

接続されるまで待機する最大時間を秒数で指定します。

戻り値: 接続できた場合、true が返されます。接続できなかった場合は、接続エラー情報が返されます。

### バッチ・モードの使用例

- Jacl を使用:

```
$Admintask testXSDomainConnection
```

- Jython スtringを使用:

```
AdminTask.testXSDomainConnection
```

### 対話モードの使用例

- Jacl を使用:

```
$AdminTask testXSDomainConnection {-interactive}
```

- Jython スtringを使用:

```
AdminTask.testXSDomainConnection ('[-interactive]')
```

### testXSServerConnection

**testXSServerConnection** コマンドは、カタログ・サーバーへの接続をテストします。このコマンドは、スタンドアロン・サーバーと、カタログ・サービス・ドメインに属するサーバーの両方で機能します。

#### 必須パラメーター:

##### ホスト (host)

カタログ・サーバーが存在するホストを指定します。

## listenerPort

カタログ・サーバーのリスナー・ポートを指定します。

## オプション・パラメーター

### timeout

カタログ・サーバーに接続されるまで待機する最大時間を秒数で指定します。

### domain

カタログ・サービス・ドメインの名前を指定します。このパラメーターの値を定義した場合は、指定されたカタログ・サービス・ドメインのクライアント・セキュリティ・プロパティを使用して接続がテストされます。そうでなければ、指定されたホストとリスナー・ポートのカタログ・サービス・ドメインを見つけるために検索が実行されます。カタログ・サービス・ドメインが見つかった場合は、そのカタログ・サービス・ドメインに定義されているクライアント・セキュリティ・プロパティを使用してサーバーがテストされます。そうでなければ、テスト時にクライアント・セキュリティ・プロパティは使用されません。

## 戻り値:

### バッチ・モードの使用例

- Jacl を使用:  

```
$Admintask testXSSTestServerConnection {-host xhost1.ibm.com -listenerPort 2809}
```
- Jython スtringを使用:  

```
AdminTask.testXSSTestServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
```

### 対話モードの使用例

- Jacl を使用:  

```
$AdminTask testXSSTestServerConnection {-interactive}
```
- Jython スtringを使用:  

```
AdminTask.testXSSTestServerConnection ('[-interactive]')
```

## カタログ・サービス・ドメイン・コレクション:

このページを使用すると、カタログ・サービス・ドメインを管理できます。カタログ・サービス・ドメインは、断片の配置を管理し、データ・グリッド内のコンテナ・サーバーのヘルスをモニターするカタログ・サーバーのグループを定義します。

この管理コンソール・ページを表示するには、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」の順にクリックします。新規カタログ・サービス・ドメインを作成するには、「新規」をクリックします。カタログ・サービス・ドメインを削除するには、削除したいカタログ・サービス・ドメインを選択して、「削除」をクリックします。

## テスト接続:

「テスト接続」ボタンをクリックすると、定義されたカタログ・サービス・ドメイン・エンドポイントのすべてが 1 つずつ照会され、使用可能なエンドポイントがあった場合、カタログ・サービス・ドメインへの接続が成功したことを示すメッセー

ジが返されます。このボタンを使用すると、接続情報およびセキュリティー情報が正しく構成されているかをテストすることができます。

#### **デフォルトの設定:**


デフォルトとして使用するカタログ・サービス・ドメインを定義します。1つのカタログ・サービス・ドメインをデフォルトとして選択し、「**デフォルトの設定**」を選択します。1つのカタログ・サービス・ドメインのみをデフォルトとして選択できます。

#### **名前:**

カタログ・サービス・ドメインの名前を指定します。

#### **デフォルト:**

リスト内のどのカタログ・サービス・ドメインがデフォルトであるかを指定しま

す。デフォルトのカタログ・サービス・ドメインは、次のアイコン  で示されます。

#### **カタログ・サービス・ドメイン設定:**

このページを使用すると、特定のカタログ・サービス・ドメインの設定を管理できます。カタログ・サービス・ドメインは、断片の配置を管理し、データ・グリッド内のコンテナ・サーバーのヘルスをモニターするカタログ・サーバーのグループを定義します。デプロイメント・マネージャーと同じセルにあるカタログ・サービス・ドメインを定義できます。WebSphere eXtreme Scale 構成が異なるセルにある場合、またはデータ・グリッドが Java SE プロセスから構成される場合は、リモート・カタログ・サービス・ドメインも定義できます。

この管理コンソール・ページを表示するには、「**システム管理**」 > 「**WebSphere eXtreme Scale**」 > 「**カタログ・サービス・ドメイン**」 > 「`catalog_service_domain_name`」の順にクリックします。

#### **テスト接続:**

「**テスト接続**」ボタンをクリックすると、定義されたカタログ・サービス・ドメイン・エンドポイントのすべてが1つずつ照会され、使用可能なエンドポイントがあった場合、カタログ・サービス・ドメインへの接続が成功したことを示すメッセージが返されます。このボタンを使用すると、接続情報およびセキュリティー情報が正しく構成されているかをテストすることができます。

#### **名前:**

カタログ・サービス・ドメインの名前を指定します。

#### **別のカタログ・サービス・ドメインが明示的に指定されない限り、このカタログ・サービス・ドメインをデフォルトとして使用可能にする:**

このチェック・ボックスを選択すると、選択されたカタログ・サービス・ドメインがそのセルのデフォルトのカタログ・サービス・ドメインになります。WebSphere

eXtreme Scale プロファイルで拡張されているセル内の各サーバー・プロファイルは、選択したカタログ・サービス・ドメインに属しています。

WebSphere eXtreme Scale について、Java EE アプリケーション・モジュールに組み込まれているすべての eXtreme Scale コンテナは、デフォルトのドメインに接続します。クライアントは、`ServerFactory.getServerProperties()`、`.getCatalogServiceBootstrap()` API を使用してデフォルトのドメインに接続して、`ObjectGridManager.connect()` API を呼び出すときに使用するカタログ・サービス・エンドポイントを取得できます。

異なるカタログ・サーバーのセットを指すようにデフォルトのドメインを変更すると、すべてのコンテナおよびクライアントが、再始動後に新規ドメインを参照します。

### カタログ・サーバー:

このカタログ・サービス・ドメインに属するカタログ・サーバーのリストを指定します。

リストにカタログ・サーバーを追加するには、「**新規**」をクリックします。このカタログ・サーバーは、eXtreme Scale 構成に事前に存在する必要があります。エンドポイントを選択して、「**編集**」または「**削除**」をクリックすると、リストでサーバーを編集または削除することもできます。各カタログ・サーバー・エンドポイントについて以下のプロパティを定義します。

### カタログ・サーバー・エンドポイント

カタログ・サービスが実行している既存のアプリケーション・サーバーまたはリモート・サーバーの名前を指定します。1 つのカタログ・サービス・ドメインに、既存のアプリケーション・サーバーとリモート・サーバーのエンドポイントの混合を含めることはできません。

- **既存のアプリケーション・サーバー:** セル内のアプリケーション・サーバー、ノード・エージェント、またはデプロイメント・マネージャーのパスを指定します。選択されたサーバーでカタログ・サービスが自動的に開始します。既存のアプリケーション・サーバーのリストから選択します。カタログ・サービス・ドメイン内で定義したすべてのアプリケーション・サーバーは、同じコア・グループになければなりません。
- **リモート・サーバー:** リモート・カタログ・サーバーのホスト名を指定します。

**WebSphere eXtreme Scale リモート・エンドポイントの場合:** リモート・カタログ・サーバー・プロセスのホスト名を指定します。リモート・サーバーは、`startOgServer` スクリプトまたは組み込みサーバー API を使用して始動する必要があります。

### クライアント・ポート

カタログ・サービス・ドメイン内のカタログ・サーバー間の通信に使用するポートを指定します。WebSphere Application Server プロセスで実行中のカタログ・サーバーの場合、この値は必須です。別のプロセスによって使用されていないどのポートにも値を設定できます。

## リスナー・ポート

クライアントとの通信に使用するポートを指定します。この値はリモート・エンドポイントに必要で、カタログ・サービスが開始されたときに使用された値と一致している必要があります。リスナー・ポートは、カタログ・サービスとの通信のために、クライアントおよびコンテナによって使用されます。

**WebSphere eXtreme Scale のリモート・エンドポイントの場合:** コンテナおよびクライアントがオブジェクト・リクエスト・ブローカー (ORB) を介してカタログ・サービスと通信するための ORB リスナー・ポートを定義します。WebSphere Application Server エンドポイントの場合、リスナー・ポート値は、BOOTSTRAP\_ADDRESS ポート構成から継承されます。

## 状況

表 24. カタログ・サーバー・エンドポイント状況

アイコン	定義
	不明
	始動済み
	停止済み

## クライアント・セキュリティー・プロパティー:

このページを使用して、カタログ・サービス・ドメインのクライアント・セキュリティーを構成します。この設定は、カタログ・サービス・ドメイン内のすべてのサーバーに適用されます。これらのプロパティーは、`com.ibm.websphere.xs.sessionFilterProps` カスタム・プロパティーを使用して `splicer.properties` ファイルを指定するか、アプリケーション EAR ファイルを接合してオーバーライドできます。

この管理コンソール・ページを表示するには、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」 >

「`catalog_service_domain_name`」 > 「クライアント・セキュリティー・プロパティー」の順にクリックします。

## クライアント・セキュリティーの使用可能化:

カタログ・サーバーのクライアント・セキュリティーが使用可能なことを指定します。選択したカタログ・サーバーに関連付けられたサーバー・プロパティー・ファイルには、一致する `securityEnabled` 設定がなければなりません。これらの設定が一致しない場合、例外が発生します。

## 資格情報の認証:

資格情報の認証を実施するか、またはサポートするかを示します。

### 常になし

クライアント資格情報の認証は実施されません。

**必須** 資格情報の認証は必ず実施されます。サーバーが資格情報の認証をサポートしない場合、クライアントはサーバーに接続できません。

#### **サポートされる**

資格情報の認証は、クライアントとサーバーの両方が資格情報の認証をサポートする場合のみ実施されます。

#### **認証の再試行回数:**

資格情報の有効期限が切れている場合に認証を再試行できる回数を指定します。

認証の再試行を望まない場合は、値を 0 に設定します。

#### **資格情報生成プログラムクラス:**

クライアントが `CredentialGenerator` オブジェクトから資格情報を取得するよう、`com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator` 実装クラスを指定します。

2 つの事前定義資格情報生成プログラム・クラスから選択することも、カスタム資格情報生成プログラムを指定することもできます。カスタム資格情報生成プログラムを選択した場合は、資格情報生成プログラム・クラス名を指定する必要があります。

- `com.ibm.websphere.objectgrid.security.plugins. UserPasswordCredentialGenerator`
- `com.ibm.websphere.objectgrid.security.plugins. UserPasswordCredentialGenerator`
- カスタム資格情報生成プログラム

#### **サブジェクト・タイプ:**

J2EE 呼び出し元を使用するか、J2EE `runAs` サブジェクト・タイプを使用するかを指定します。この値は、`WSTokenCredentialGenerator` 資格情報生成プログラムを選択するときに指定する必要があります。

- **runAs:** サブジェクトには、J2EE `run as ID` および J2EE `run as` 資格情報のプリンシパルが含まれています。
- **呼び出し元:** このサブジェクトには、J2EE 呼び出し元および J2EE 呼び出し元資格情報のプリンシパルが含まれています。

#### **ユーザー ID:**

`UserPasswordCredentialGenerator` 資格情報生成プログラム実装を使用する場合、ユーザー ID を指定します。

#### **パスワード:**

`UserPasswordCredentialGenerator` 資格情報生成プログラムの実装を使用する場合、パスワードを指定します。

#### **資格情報生成プログラム・プロパティ:**

カスタム `CredentialGenerator` 実装クラスのプロパティを指定します。このプロパティは、`setProperties(String)` メソッドを使用してオブジェクトに設定されます。資



格情報生成プログラムのプロパティ値は、「資格情報生成プログラム・クラス (Credential generator class)」フィールドに値が指定されている場合のみ使用されません。

### カタログ・サービス・ドメインのカスタム・プロパティ:

カスタム・プロパティを定義すると、カタログ・サービス・ドメインの構成をさらに編集できます。

この管理コンソール・ページを表示するには、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」 > 「カスタム・プロパティ」の順にクリックします。新規カスタム・プロパティを作成するには、「新規」をクリックします。

#### 名前:

カタログ・サービス・ドメインのカスタム・プロパティの名前を指定します。

#### 値:

カタログ・サービス・ドメインのカスタム・プロパティの値を指定します。

## クォーラム・メカニズムの構成

クォーラム・メカニズムはカタログ・サービスごとに構成します。カタログ・サービス・ドメイン内のすべてのカタログ・サーバーでクォーラム・メカニズムを使用可能にする必要があります。

### 始める前に

クォーラム・メカニズムを使用可能にするには、このタイプの構成をサポートするトポロジを構成しなければなりません。構成は次の要件に対応している必要があります。

- **フラット IP アドレス・スペース:** ネットワーク上のアドレス可能なすべてのエレメントが、ネットワーク上のアドレス可能な他のすべてのエレメントにスムーズに接続できなければなりません。フラット IP アドレス名前空間を使用する必要があります。構成内のすべてのファイアウォールが、カタログ・サーバーとコンテナ・サーバーをホスティングするために使用される IP アドレスとポート間ですべてのトラフィックのフローを許可しなければなりません。
- **カタログ・サーバーの数:** 構成内のデータ・センターごとに少なくとも 1 つのカタログ・サーバーを開始する必要があります。
- **ハートビート間隔設定:** ハートビート間隔を定義しない場合、デフォルト値は 30 秒です。WebSphere eXtreme Scale は、定義された間隔で単一ゾーン内の JVM をチェックします。例えば、コンテナ・サーバーのハートビートがなかった場合に、クォーラムが確立されているとフェイルオーバー・イベントが発生して、新しいコンテナ・サーバーが設定されます。詳しくは、310 ページの『フェイルオーバー検出のためのハートビート間隔設定のチューニング』を参照してください。
- **トランスポート・セキュリティー:** データ・センターは、通常、地理的に異なるロケーションにデプロイされるため、セキュリティー上の理由からデータ・セン



ター間のトランスポート・セキュリティーを使用可能にすることが推奨されます。「管理ガイド」のトランスポート層セキュリティーを参照してください。

## このタスクについて

デフォルトではクォーラム・メカニズムが使用不可になっています。次のようなシナリオでは、クォーラム・メカニズムを使用可能にしてください。

- カタログ・サービス・ドメインが、予測不能または不安定なネットワークに広がっている場合。このタイプのネットワークは複数のデータ・センターに広がっている可能性があります。
- ネットワーク・ブラウン・アウトがあった際、データ・グリッドが自己修復せず、代わりにデータ・グリッドの操作を一時的に休止することが望ましい場合。

カタログ・サービス・ドメインが単一データ・センター内に含まれている場合、またはローカル・エリア・ネットワーク (LAN) 上にある場合は、クォーラム・メカニズムを使用不可にしたままでかまいません。このタイプの構成では、デフォルトのハートビート設定が使用され、ブラウン・アウトの時間は 10 秒未満であると想定されます。検出期間は約 30 秒ごとであるため、ブラウン・アウトが短時間発生しても、それによってデータ・グリッド内の配置変更が起こることはありません。

クォーラムを使用可能にする場合は、配置操作を実行するため、すべてのカタログ・サーバーが使用可能で、データ・グリッドと通信する必要があります。ネットワーク・ブラウン・アウトが発生した場合、すべてのカタログ・サーバーが使用可能になるまで配置は休止されます。データ・センターの障害が発生した場合は、障害があったカタログ・サーバーをクォーラムから除去するため、管理者による手動の処置が必要です。

## 手順

1. **カタログ・サーバーのクォーラムを使用可能にします。** WebSphere Application Server の場合、サーバー・プロパティー・ファイルを使用してクォーラムを構成する必要があります。スタンドアロン環境の場合、プロパティー方式を使用するか、サーバーの開始時にクォーラムを使用可能にすることができます。
  - **サーバー・プロパティー・ファイル内に `enableQuorum=true` プロパティーを設定する。**

この構成は、WebSphere Application Server でもスタンドアロン環境でも使用できます。

```
catalogClusterEndpoints=cat0:cat0.domain.com:6600:6601,  
cat1:cat1.domain.com:6600:6601  
catalogServiceEndpoints= cat0.domain.com:2809, cat1.domain.com:2809  
enableQuorum=true
```

図 30. `objectGridServer.properties` ファイル

プロパティー・ファイルの構成の詳細については、サーバー・プロパティー・ファイルを参照してください。

- **`startOgServer` コマンドで `-quorum` 使用可能フラグを渡す。**

この構成方式は、スタンドアロン・サーバーの開始時にのみ使用できます。

```
# bin/startOgServer cat0 -serverProps objectGridServer.properties -quorum true
```

**start0gServer** コマンドの詳細については、457 ページの『**start0gServer** スクリプト』を参照してください。

## 2. 同一ゾーン内のコンテナ・サーバーを開始します。

複数のデータ・センターにまたがるデータ・グリッドを実行する場合、サーバーはゾーン情報を使用して、そのサーバーが存在するデータ・センターを識別する必要があります。コンテナ・サーバーにゾーンを設定することにより、WebSphere eXtreme Scale が、データ・センターのスコープ内にあるコンテナ・サーバーのヘルスをモニターでき、データ・センター間のトラフィックを最小化できます。コア・グループ内のコンテナ・サーバー JVM は、(広域ネットワークのような) リンクで接続された複数の LAN にまたがってはなりません。コンテナ・サーバーのゾーンを定義する方法の詳細については、280 ページの『コンテナ・サーバーのゾーンの定義』を参照してください。

コンテナ・サーバー JVM はゾーン ID でタグ付けされます。コンテナ JVM のデータ・グリッドは JVM から成る小さいコア・グループに自動的に分割されます。1 つのコア・グループには同じゾーンからの JVM のみが含まれます。いかなる時点でも、異なるゾーンからの JVM が同じコア・グループに存在することはありません。

コア・グループはそのメンバー JVM の障害の検出を活発に試みます。

## タスクの結果

カタログ・サービス・ドメイン内のカタログ・サーバーでクォーラム・メカニズムを使用可能に設定すると、データ・グリッドの配置操作を実行するために、すべてのカタログ・サーバーが使用可能でなければなりません。ネットワーク・ブラウン・アウトが短時間発生した場合、クォーラム内のすべてのカタログ・サーバーが使用可能になるまで配置操作は一時的に停止されます。

これらのステップを繰り返して、追加のカタログ・サーバーをクォーラムに追加できます。

## 次のタスク

- 構成で要求される管理方式を使用してカタログ・サーバーを停止することで、カタログ・サーバーをクォーラムから除去できます。管理用アクションを使用してカタログ・サーバーを停止すると、残りのカタログ・サーバーを使用してクォーラムが自動的に再確立され、配置を継続できます。このトピックで説明したステップを使用してカタログ・サーバーを再始動すると、カタログ・サーバーはクォーラムに再加入できます。
- 現在定義されているクォーラム内に存在するカタログ・サーバーで長期または永続的な障害が発生した場合、配置を継続できるようにクォーラム・メカニズムをオーバーライドする必要があります。クォーラム・メカニズムのオーバーライドの詳細については、480 ページの『データ・センター障害の管理』を参照してください。

## フェイルオーバー検出のためのハートビート間隔設定のチューニング

ハートビート間隔設定で、障害の起きたサーバーがないかを調べるシステム・チェックの間の時間を構成できます。 **-heartbeat** オプションは **start0Gserver** コマンドのパラメーターで、カタログ・サーバーにのみ適用されます。

### このタスクについて

フェイルオーバーの構成は、使用している環境のタイプによって異なります。スタンドアロン環境を使用している場合は、コマンド行でフェイルオーバーを構成できます。 WebSphere Application Server Network Deployment 環境を使用している場合は、WebSphere Application Server Network Deployment 管理コンソールでフェイルオーバーを構成する必要があります。

### 手順

- スタンドアロン環境のフェイルオーバーを構成します。

ハートビート間隔は、**start0GServer** スクリプト・ファイルの **-heartbeat** パラメーターを使用してコマンド行で構成できます。このパラメーターは以下のいずれかの値に設定します。

表 25. ハートビート間隔

値	アクション	説明
0	標準 (デフォルト)	通常、30 秒以内にフェイルオーバーが検出されます。
-1	高速	通常、5 秒以内にフェイルオーバーが検出されます。
1	低速	通常、180 秒以内にフェイルオーバーが検出されます。

高速のハートビート間隔は、プロセスおよびネットワークが安定している場合に役立ちます。ネットワークまたはプロセスが最適に構成されていないと、ハートビートを見逃す可能性があり、そうなった場合は誤って障害検出が示されることがあります。

- WebSphere Application Server 環境のフェイルオーバーを構成します。

WebSphere Application Server Network Deployment バージョン 6.0.2 以降は、WebSphere eXtreme Scale のフェイルオーバーを高速で行えるように構成できます。ハード障害の場合のデフォルトのフェイルオーバー時間は、約 200 秒です。ハード障害は、物理的なコンピューターまたはサーバーの破損、ネットワーク・ケーブルの切断、オペレーティング・システム・エラーのことです。プロセスの異常終了やソフト障害による障害は、一般的に 1 秒未満でフェイルオーバーされます。ソフト障害の障害検出は、デッド・プロセスのネットワーク・ソケットがそのプロセスをホスティングするサーバーのオペレーティング・システムにより自動的にクローズされるときに発生します。

### コア・グループのハートビート構成

WebSphere Application Server プロセスで実行されている WebSphere eXtreme Scale は、アプリケーション・サーバーのコア・グループ設定のフェイルオーバー特性を継承します。以下のセクションでは、以下のようなさまざまなバージョンの WebSphere Application Server Network Deployment のコア・グループ・ハートビート設定を構成する方法について説明します。

- **WebSphere Application Server Network Deployment バージョン 6.x または 7.x のコア・グループ設定を更新します。**

ハートビート間隔は、WebSphere Application Server のバージョン 6.0 からバージョン 6.1.0.12 までは秒単位、バージョン 6.1.0.13 からはミリ秒単位で指定します。また、欠落ハートビートの数も指定する必要があります。この値は、ピア Java 仮想マシン (JVM) に障害が起きたと見なされるまでに、容認される欠落ハートビートの数を示します。ハード障害の検出時間は、ほぼハートビート間隔と欠落ハートビート数の積です。

これらのプロパティは、WebSphere 管理コンソールで、コア・グループに対してカスタム・プロパティを使用して指定します。構成について詳しくは、コア・グループ・カスタム・プロパティを参照してください。アプリケーションによって使用されるすべてのコア・グループに対して、以下のプロパティを指定する必要があります。

- ハートビート間隔は、`IBM_CS_FD_PERIOD_SEC` カスタム・プロパティ (秒単位) または `IBM_CS_FD_PERIOD_MILLIS` カスタム・プロパティ (ミリ秒単位、バージョン 6.1.0.13 以降が必要) を使用して指定します。
- 欠落ハートビート数は、`IBM_CS_FD_CONSECUTIVE_MISSED` カスタム・プロパティを使用して指定します。

`IBM_CS_FD_PERIOD_SEC` プロパティのデフォルト値は 20 で、`IBM_CS_FD_CONSECUTIVE_MISSED` プロパティのデフォルト値は 10 です。`IBM_CS_FD_PERIOD_MILLIS` プロパティを指定すると、設定されている `IBM_CS_FD_PERIOD_SEC` カスタム・プロパティがオーバーライドされます。これらのプロパティの値は、正の整数値です。

WebSphere Application Server Network Deployment 6.x サーバーで 1500 ミリ秒の障害検出時間を実現するには、以下の設定を使用します。

- `IBM_CS_FD_PERIOD_MILLIS = 750` を設定 (WebSphere Application Server Network Deployment バージョン 6.1.0.13 以降)
- `IBM_CS_FD_CONSECUTIVE_MISSED = 2` を設定

- **WebSphere Application Server Network Deployment バージョン 7.0 のコア・グループ設定を更新します。**

バージョン 7.0 の WebSphere Application Server Network Deployment は、フェイルオーバー検出を増減するために調整できる以下の 2 つのコア・グループ設定を提供します。

- ハートビート伝送期間。 デフォルト値は 30000 ミリ秒です。
- ハートビート・タイムアウト期間。 デフォルト値は 180000 ミリ秒です。

これらの設定を変更する方法については詳しくは、WebSphere Application Server Network Deployment インフォメーション・センター: ディスカバリーおよび障害検出の設定を参照してください。

WebSphere Application Server Network Deployment バージョン 7 サーバーで 1500 ミリ秒の障害検出時間を実現するには、以下の設定を使用します。

- ハートビート伝送期間を 750 ミリ秒に設定します。

- ハートビート・タイムアウト期間を 1500 ミリ秒に設定します。

## 次のタスク

短いフェイルオーバー時間を指定するようにこれらの設定を変更すると、注意すべきシステム・チューニング上の問題が生じます。まず Java はリアルタイム環境ではありません。JVM に長期のガーベッジ・コレクション時間が発生すると、スレッドが遅延する可能性があります。JVM をホスティングするマシンの負荷が大きくなった (JVM 自身またはマシンで実行中の他のプロセスが原因) 場合にも、スレッドが遅延する可能性があります。スレッドが遅延された場合、ハートビートが正確な時間で送信されない可能性があります。最悪の場合、必要なフェイルオーバー時間で遅延が生じる可能性があります。スレッドが遅延すると、誤障害検出が発生します。実動環境で誤障害検出が発生しないように、システムを調整し、サイズ設定する必要があります。これを確実にするには、適切な負荷テストが最善の策です。

注: eXtreme Scale の現行バージョンは、WebSphere Real Time をサポートします。

## コンテナ・サーバーの構成

コンテナ・サーバーは、データ・グリッドのアプリケーション・データを保管します。通常、このデータは区画と呼ばれるパーツに分割され、複数のコンテナ・サーバーでホストされます。これを受けて各コンテナ・サーバーは、完全なデータのサブセットをホストします。

### このタスクについて

- **スタンドアロン・コンテナ・サーバー:**

スタンドアロン・コンテナ・サーバーは、サーバー・プロパティ・ファイルとデプロイメント・ポリシー XML ファイルを使用して構成します。コンテナ・サーバーのライフサイクルは、**startOgServer** および **stopOgServer** スクリプト、または組み込みサーバー API を使用して制御します。

- **WebSphere Application Server 内で開始するコンテナ・サーバー:**

WebSphere Application Server 内のコンテナ・サーバーは、サーバー・プロパティ・ファイルと Java EE アプリケーション・モジュールに組み込まれるデプロイメント・ポリシー XML ファイルを使用して構成します。コンテナ・サーバーのライフサイクルは、アプリケーションによって制御されます。コンテナ・サーバーは、アプリケーションと一緒に開始または停止します。

### コンテナ・サーバーの再始動

コンテナ・サーバーを再始動するためのシステム・プロパティを追加できます。

### 手順

- `com.ibm.websphere.objectgrid.container.reconnect.block.reconnect.time` 他のコンテナ再接続呼び出しを遮断する時間長 (ミリ秒) を定義します。コンテナ・サーバーが、製品オファリング WebSphere eXtreme Scale for WebSphere Application Server 上で実行される場合に限り有効です。デフォルトは 30000 ミリ秒です。



- com.ibm.websphere.objectgrid.container.reconnect.min.successful.heartbeats  
 コンテナが停止できるまでの、正常なハートビートの最小数を定義します。コンテナ・サーバーが、製品オフリング WebSphere eXtreme Scale for WebSphere Application Server 上で実行される場合に限り有効です。デフォルトは 10 です。
- com.ibm.websphere.objectgrid.container.reconnect.restart コンテナの再接続で JVM が再始動できるかどうかを定義します。コンテナが、スタンドアロン環境の製品オフリング WebSphere eXtreme Scale 上で実行される場合に限り有効です。デフォルトは true です。
- com.ibm.websphere.objectgrid.container.reconnect.restart.delay JVM を再始動するとき、親が停止してから新たに作成された子コンテナの開始を続行するまでに遅延する時間 (ミリ秒) を定義します。コンテナが、スタンドアロン環境の製品オフリング WebSphere eXtreme Scale 上で実行される場合に限り有効です。デフォルトは 2000 ミリ秒です。
- com.ibm.websphere.objectgrid.container.reconnect.restart.parent.timeout JVM を再始動するときに、新たに作成された子コンテナが親の停止を待つ時間 (ミリ秒) を定義します。この時間を過ぎるとタイムアウトになります。コンテナが、スタンドアロン環境の製品オフリング WebSphere eXtreme Scale 上で実行される場合に限り有効です。デフォルトは 180000 ミリ秒です。
- com.ibm.websphere.objectgrid.container.reconnect.retry.forever コンテナが、コンテナの再接続を永久に再試行するかどうかを定義します。コンテナ・サーバーが、製品オフリング WebSphere eXtreme Scale for WebSphere Application Server 上で実行される場合に限り有効です。デフォルトは false です。

## WebSphere Application Server のコンテナ・サーバーの構成

WebSphere Application Server 内のコンテナ・サーバーは、サーバー・プロパティ・ファイルと Java EE アプリケーション・モジュールに組み込まれるデプロイメント・ポリシー XML ファイルを使用して構成します。アプリケーションが停止または開始するときに、コンテナ・サーバーも停止または開始します。

### 始める前に

カタログ・サービス・ドメインを構成します。詳しくは、288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』を参照してください。

### このタスクについて

WebSphere Application Server 内のコンテナ・サーバーを作成するには、コンテナ・サーバーを作成する WebSphere eXtreme Scale 構成 XML ファイルをアプリケーション・モジュール内に組み込む必要があります。

### 手順

1. WebSphere eXtreme Scale コンテナ・サーバー定義を含んでいる Java EE アプリケーションをデプロイするアプリケーション・サーバーを指定します。ターゲット・アプリケーション・サーバー・プロファイルが、WebSphere eXtreme Scale プロファイルで拡張されていることを確認します。 実稼働環境では、コンテナ

ー・サーバー用のサーバーをカタログ・サーバーと連結しないでください。詳しくは、225 ページの『WebSphere eXtreme Scale のプロファイルの作成および拡張』を参照してください。

2. サーバー・プロパティ・ファイルを構成し、サーバー・プロパティ・ファイルを各ターゲット・アプリケーション・サーバー・ノードのクラスパスに追加します。詳しくは、サーバー・プロパティ・ファイルを参照してください。
3. ObjectGrid 記述子 XML ファイルとデプロイメント・ポリシー XML ファイルをアプリケーション・モジュールに追加します。詳しくは、『コンテナ・サーバーの自動始動のための WebSphere Application Server アプリケーションの構成』を参照してください。

### コンテナ・サーバーの自動始動のための WebSphere Application Server アプリケーションの構成:

WebSphere Application Server 環境内のコンテナ・サーバーは、eXtreme Scale XML ファイルを組み込んだモジュールが開始されると自動的に開始されます。

#### 始める前に

WebSphere Application Server および WebSphere eXtreme Scale をインストールする必要があります。さらに、WebSphere Application Server 管理コンソールにアクセスできなければなりません。

#### このタスクについて

Java Platform, Enterprise Edition アプリケーションのクラス・ローダー規則は複雑であるため、Java EE サーバー内で共有データ・グリッドを使用しているときは、ロード元クラスが非常に複雑になります。A Java EE アプリケーションは通常、単一のエンタープライズ・アーカイブ (EAR) ファイルです。EAR ファイルには、1 つ以上のデプロイされた Enterprise JavaBeans (EJB) モジュールまたは Web アーカイブ (WAR) モジュールが含まれます。

WebSphere eXtreme Scale は各モジュールの開始を監視し、eXtreme Scale XML ファイルを検査します。カタログ・サービスが、XML ファイルによるモジュールの開始を検出すると、アプリケーション・サーバーはコンテナ・サーバー Java 仮想マシン (JVM) として登録されます。コンテナ・サーバーをカタログ・サービスに登録することにより、異なるデータ・グリッドに同じアプリケーションをデプロイできますが、カタログ・サービスで単一データ・グリッドとして使用されます。カタログ・サービスは、セル、グリッド、または動的グリッドとの関連はありません。必要に応じて、単一のデータ・グリッドを複数のセルに分散させることができます。

#### 手順

1. META-INF フォルダに eXtreme Scale XML ファイルが含まれるモジュールを持つように EAR ファイルをパッケージ化します。WebSphere eXtreme Scale は、EJB および WEB モジュールが開始されると、これらのモジュールの META-INF フォルダで objectGrid.xml および objectGridDeployment.xml ファイルの存在を検出します。objectGrid.xml ファイルのみが検出されると、



JVM はクライアントと見なされます。その他の場合は、この JVM が `objectGridDeployment.xml` ファイルで定義されているデータ・グリッドのコンテナーであると見なされます。

これらの XML ファイルの正しい名前を使用しなければなりません。ファイル名には大/小文字の区別があります。これらのファイルが存在しないと、コンテナーは開始されません。断片が配置されることを示すメッセージは `systemout.log` ファイルで確認することができます。eXtreme Scale を使用する EJB モジュールまたは WAR モジュールの META-INF ディレクトリーに eXtreme Scale XML ファイルがなければなりません。

eXtreme Scale XML ファイルには以下のものがあります。

- `objectGrid.xml` という ObjectGrid 記述子 XML ファイル。詳しくは、ObjectGrid 記述子 XML ファイルを参照してください。
- `objectGridDeployment.xml` というデプロイメント記述子 XML ファイル。詳しくは、デプロイメント・ポリシー記述子 XML ファイルを参照してください。
- (オプション) エンティティーが使用されている場合、エンティティー・メタデータ記述子 XML ファイル。`entity.xml` ファイル名は、`objectGrid.xml` ファイルに指定されている名前と一致しなければなりません。詳しくは、エンティティー・メタデータ記述子 XML ファイルを参照してください。

ランタイムはこれらのファイルを検出し、カタログ・サービスに連絡して、別のコンテナーを使用してこの eXtreme Scale の断片をホストできることを通知します。

**ヒント:** アプリケーションにエンティティーがあり、1 つのコンテナー・サーバーを使用する計画であれば、デプロイメント記述子 XML ファイルの中の `minSyncReplicas` 値を 0 に設定します。そうしないと、別のサーバーが `minSyncReplica` ポリシーを満たしはじめるまで配置を行えないため、`SystemOut.log` ファイル内に以下のメッセージのいずれかが記録される可能性があります。

```
CWPRJ1005E: Error resolving entity association. Entity=entity_name,
association=association_name.
(エンティティー関連の解決中にエラーが発生しました。
エンティティー=entity_name、関連=association_name)
```

```
CW0BJ3013E: The EntityMetadata repository is not available. Timeout
threshold reached when trying to register the entity: entity_name.
(EntityMetadata リポジトリーを使用できません。
エンティティー entity_name の登録試行中にタイムアウトしきい値に到達しました)
```

## 2. アプリケーションをデプロイして開始します。

モジュールが開始されると、コンテナーが自動的に開始されます。カタログ・サービスが開始され、できるだけ速やかに区画のプライマリーおよびレプリカ (断片) が配置されます。配置を遅らせるように環境を構成していない限り、この配置は直ちに行われます。詳しくは、475 ページの『配置の制御』を参照してください。

## 次のタスク

コンテナと同じセル内にあるアプリケーションは、`ObjectGridManager.connect(null, null)` メソッドを使用してこれらのデータ・グリッドに接続した後、`getObjectGrid(ccc, "object grid name")` メソッドを呼び出すことができます。 `connect` または `getObjectGrid` メソッドはコンテナが断片の配置を完了するまでブロックされることがありますが、このブロックはデータ・グリッドが開始されるときだけ問題となります。

### ClassLoader

eXtreme Scale に格納されたプラグインまたはオブジェクトは、特定のクラス・ローダーにロードされます。ロードされたオブジェクトは、同じ EAR 内の 2 つの EJB モジュールに含めることができます。これらのオブジェクトは同じですが、別の `ClassLoader` を使用してロードされています。アプリケーション A が、サーバーに対してローカルなマップに `Person` オブジェクトを保管した場合、アプリケーション B がこのオブジェクトを読み取ろうとすると、`ClassCastException` を受け取ります。この例外は、アプリケーション B が `Person` オブジェクトを別のクラス・ローダーにロードしたために発生します。

この問題を解決する方法の 1 つは、eXtreme Scale に格納された必要なプラグインおよびオブジェクトをルート・モジュールが含むようにすることです。 eXtreme Scale を使用する各モジュールは、ルート・モジュール内でクラスを参照する必要があります。もう 1 つの解決方法は、これらの共有オブジェクトを、モジュールとアプリケーションの両方が共有する共通クラス・ローダー上のユーティリティ JAR ファイル内に配置することです。オブジェクトは、`WebSphere` クラスまたは `lib/ext` ディレクトリーにも配置できますが、この配置ではデプロイメントが複雑になります。

EAR ファイル内の EJB モジュールは通常、同じ `ClassLoader` を共有するため、この問題の影響を受けません。各 WAR モジュールには独自の `ClassLoader` があり、この問題の影響を受けます。

### データ・グリッド・クライアントのみに接続

`catalog.services.cluster` プロパティーがセル、ノード、またはサーバー・カスタム・プロパティーで定義されている場合は、EAR ファイル内のすべてのモジュールが `ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null)` メソッドを呼び出して `ClientClusterContext` を取得できます。このモジュールはまた `ObjectGridManager.getObjectGrid(ccc, "grid name")` メソッドを呼び出して、データ・グリッドの参照を取得できます。アプリケーション・オブジェクトがマップに格納されている場合は、それらのオブジェクトが共通の `ClassLoader` に存在することを確認してください。

Java クライアントまたはセル外のクライアントは、カタログ・サービスのブートストラップ IIOP ポートと接続できます。`WebSphere Application Server` の中で、デプロイメント・マネージャーは、デフォルトでカタログ・サービスをホストします。そして、クライアントは、`ClientClusterContext` と指定されたデータ・グリッドを取得できます。

### エンティティー・マネージャー

エンティティ・マネージャーを使用すると、タプルはアプリケーション・オブジェクトではなくマップに保管され、クラス・ローダーが少なくなるという問題が発生します。ただし、プラグインが問題になることがあります。また、エンティティ (ObjectGridManager.connect("host:port[,host:port], null, objectGridOverride) または ObjectGridManager.connect(null, objectGridOverride)) が定義されているデータ・グリッドに接続するときは、クライアント・オーバーライド ObjectGrid 記述子 XML ファイルが常に必要となることにも注意してください。

---

## IBM eXtremeMemory および IBM eXtremeIO の構成

eXtremeMemory を構成することにより、オブジェクトを Java ヒープでなくネイティブ・メモリーに保管できます。eXtremeMemory を構成すると、新しいトランスポート・メカニズムである eXtremeIO が使用可能になります。オブジェクトを Java ヒープから移動することで、ガーベッジ・コレクションに伴う一時停止を回避でき、より安定したパフォーマンスを得られるうえ、応答時間も予測可能になります。

### 始める前に

- **Linux** eXtremeIO と eXtremeMemory は、64 ビット SDK を使用する x86 64 ビット Linux システムのみでサポートされます。
- 使用するマップ・セットに含まれるマップはすべて COPY\_TO\_BYTES または COPY\_TO\_BYTES\_RAW コピー・モードで構成されている必要があります。マップ・セット内のいずれかのマップがこれらのコピー・モードを使用していない場合、オブジェクトは Java ヒープに保管され、オブジェクト・リクエスト・ブローカー (ORB) が使用されます。
- 次の構成シナリオでは、eXtremeIO と eXtremeMemory は使用できません。
  - カスタム evictor プラグインを使用する場合
  - 複合索引を使用する場合
  - 組み込みの後書きローダーを使用する場合

### このタスクについて

JVM は、プロセス・メモリーを収集、圧縮、および拡張するために使用量ヒューリスティックに依存します。これらの操作はガーベッジ・コレクターによって実行されます。しかし、ガーベッジ・コレクションの実行には関連のコストが付随します。Java ヒープのサイズとデータ・グリッド内のオブジェクトの数が増すにつれ、ガーベッジ・コレクションの実行にかかるコストも増えます。JVM は、異なるユース・ケースや目標 (最適なスループット、最適な一時停止時間、世代別、バランス、およびリアルタイムのガーベッジ・コレクション) に応じたそれぞれのヒューリスティックを提供します。完璧なヒューリスティックは存在しません。単一ヒューリスティックがすべての可能な構成に適合するわけではありません。

WebSphere eXtreme Scale はデータ・キャッシングと分散マップを使用します。分散マップのエントリーには、既知のライフサイクルが入っています。このライフサイクルには、次の操作、GET、INSERT、DELETE、および UPDATE が含まれます。これら既知のマップ・ライフサイクルを使用することで、eXtremeMemory および eXtremeIO は、JVM 使用量ヒューリスティックよりも効率的にメモリーを使用できます。

次の図は、eXtremeMemory の使用が、どのように環境内でより一貫性のある相対応答時間をもたらすかを示しています。相対応答時間が高いパーセンタイルに近づくにつれ、eXtremeMemory を使用する要求のほうが相対応答時間が少なくなります。図は 95 から 100 パーセンタイルを表示しています。

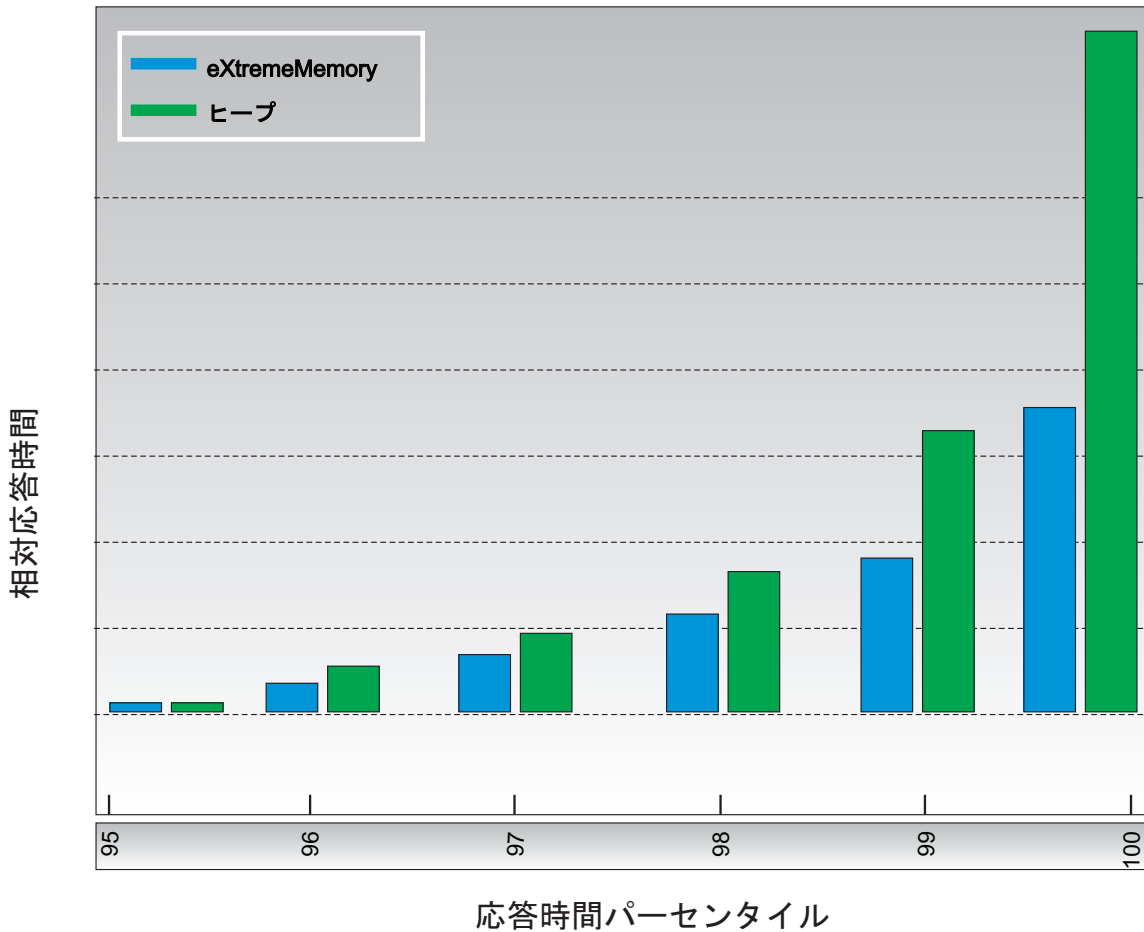


図 31. eXtremeMemory とヒープ・ストレージの応答時間の比較

eXtremeMemory を使用すると、コンテナ・サーバー間の通信に eXtremeIO が使用されます。オブジェクトは、コンテナ・サーバー上でバイト列にシリアライズされます。eXtremeIO と eXtremeMemory を使用可能にするには、データ・グリッド内のすべてのコンテナ・サーバーで必要なサーバー・プロパティを設定し、サーバーを再始動します。

### 手順

1. オプション: 使用する適切な **maxXMSize** プロパティ値を決定します。
  - a. 既存の構成内で、エントリーごとのサイズを判定します。 **xscmd -c showMapSizes** コマンドを実行して、このサイズを判定します。
  - b. **maxXMSize** の値を計算します。 エントリーの最大合計サイズ (*maximum\_total\_size*) を求めるには、*size\_per\_entry* と

*maximum\_number\_of\_entries* を乗算します。メタデータ処理に使用する割合が **maxXMSize** の 60% を超えないようにしてください。*maximum\_total\_size* \* 1.65 を乗算して、**maxXMSize** の値を求めます。

2. 構成内の各コンテナ・サーバーのサーバー・プロパティ・ファイルを更新して、新しいトランスポートを使用可能にします。次のサーバー・プロパティにより、新しいトランスポートが使用可能になります。

#### 必須プロパティ

##### **enableXM**

true に設定すると、サーバー上の IBM eXtremeMemory を使用可能にし、同期および非同期のレプリカ生成用に IBM eXtremeIO を使用するようにサーバーを構成します。キャッシュ・エントリは、Java ヒープではなく、ネイティブ・メモリーに保管されます。データ・グリッドのすべてのコンテナ・サーバーは、**enableXM** プロパティに同じ値を使用しなければなりません。

デフォルト: false

#### 推奨されるプロパティ

##### **maxXMSize**

eXtremeMemory ストレージ用にサーバーが使用するメモリーの最大量を、メガバイト単位で設定します。

デフォルト: システム上の合計メモリーの 25%

#### オプション・プロパティ

##### **maxXIONetworkThreads**

eXtremeIO トランスポート・ネットワーク・スレッド・プールに割り振るスレッドの最大数を設定します。

デフォルト:50

##### **minXIONetworkThreads**

eXtremeIO トランスポート・ネットワーク・スレッド・プールに割り振るスレッドの最小数を設定します。

デフォルト:50

##### **maxXIOWorkerThreads**

eXtremeIO トランスポート要求処理スレッド・プールに割り振るスレッドの最大数を設定します。

デフォルト:128

##### **minXIOWorkerThreads**

eXtremeIO トランスポート要求処理スレッド・プールに割り振るスレッドの最小数を設定します。

デフォルト:128

##### **xioChannel.xioContainerTCPNonSecure.Port**

サーバー上の eXtremeIO の非セキュア・リスナー・ポート番号を指定します。値を設定しなければ、一時ポートが使用されます。**transportType** プロパティが TCP/IP に設定されている場合のみこのプロパティが使用されます。

**制約事項:** xioChannel.xioContainerTCPNonSecure.Port プロパティは Liberty プロファイルではサポートされません。

#### **xioChannel.xioContainerTCPSecure.Port**

サーバー上の eXtremeIO の SSL ポート番号を指定します。

**transportType** プロパティが SSL-Supported または SSL-Required に設定されている場合のみこのプロパティが使用されます。

3. コンテナ・サーバーを再始動して、新しいトランスポート・メカニズムの使用を開始します。詳しくは、451 ページの『スタンドアロン・サーバーの始動と停止』および 467 ページの『WebSphere Application Server 環境でのサーバーの開始と停止』を参照してください。

---

## 複数データ・センター・トポロジーの構成

マルチマスター非同期レプリカ生成の場合、一連のカタログ・サービス・ドメイン同士をリンクします。そうすると、接続されたカタログ・サービス・ドメインは、リンクを介したレプリカ生成を使用して同期化されます。リンクを定義するには、プロパティ・ファイルを使用するか、実行時に Java Management Extensions (JMX) プログラムを使用するか、またはコマンド行ユーティリティを使用できます。ドメインの現在のリンク・セットは、カタログ・サービス内に保管されます。データ・グリッドをホスティングするカタログ・サービス・ドメインを再始動せずにリンクを追加および削除できます。

### 始める前に

- マルチマスター・レプリカ生成トポロジーおよび設計上の考慮事項の詳細については、37 ページの『複数データ・センター・トポロジーの計画』を参照してください。カタログ・サービス・ドメイン間のリンクは、サーバー・プロパティ・ファイルを使用して構成でき、サーバー開始時にトポロジーを形成できます。リンクは実行時にも構成できます。
- マルチマスター・レプリカ生成トポロジーでローダーを使用する場合は、データ・センター間で正確なデータをどのように維持していくか計画する必要があります。使用できるアプローチは、使用するトポロジーによって異なります。詳しくは、43 ページの『マルチマスター・トポロジーでのローダーについての考慮事項』を参照してください。

### 手順

- ブートストラップのために、トポロジー内の各カタログ・サービス・ドメインのカタログ・サーバーのサーバー・プロパティ・ファイル内にリンクを定義します。

カタログ・サーバーのこのファイルを定義する方法の詳細については、サーバー・プロパティ・ファイルを参照してください。

**重要:** プロパティ名では大/小文字が区別されます。

#### **ローカル・ドメイン・ネーム:**

現行カタログ・サーバーのカタログ・サービス・ドメインの名前 (例えば、ドメイン A) を指定します。

```
domainName=A
```



### 外部ドメイン・ネームのオプション・リスト:

マルチマスター・レプリカ生成トポロジー内のリンク先のカタログ・サービス・ドメインの名前 (例えば、ドメイン B) を指定します。

```
foreignDomains=B
```

### 外部ドメイン・ネームのエンドポイントのオプション・リスト:

外部ドメイン (ドメイン B など) のカタログ・サーバーの接続情報を指定します。

```
B.endPoints=hostB1:2809, hostB2:2809
```

外部ドメインに複数のカタログ・サーバーがある場合には、すべてを指定します。

- **xscmd** ユーティリティーまたは **JMX** プログラミングを使用して、実行時にリンクを追加または削除します。

ドメインのリンクは、複製メモリー内のカタログ・サービスで保持されます。このリンク・セットは、このドメインまたはその他のドメインを再始動することなく、いつでも管理者が変更できます。 **xscmd** ユーティリティーには、リンクを処理するためのいくつかのオプションが用意されています。

**xscmd** ユーティリティーは特定のカタログ・サービス、すなわち単一カタログ・サービス・ドメインに接続します。そのため、**xscmd** ユーティリティーを使用して、接続先のドメインと任意のその他のドメインとの間のリンクを作成および破棄できます。

例えば以下のように、コマンド行を使用して、リンクを作成します。

```
xscmd -c establishLink -cep host:2809 -fd dname -fe fdHostA:2809,fdHostB:2809
```

このコマンドは、ローカル・ドメインと **dname** という外部ドメインの間に新規リンクを確立します。 **dname** カタログ・サービスは、**fdHostA:2809** および **fdHostB:2809** で実行されています。ローカル・カタログ・サービス・ドメインには、カタログ・サービス・リスナー・ホストと **host:2809** のポートがあります。外部ドメインからのすべてのカタログ・サービス・エンドポイントを指定して、ドメインへのフォールト・トレランス接続を可能にします。外部カタログ・サービス・ドメインのカタログ・サービスの場合、使用する **host:port** ペアを単一にしないでください。

**xscmd** では **-cep** オプションを指定して、任意のローカル・カタログ・サービス **JVM** を使用できます。カタログ・サーバーが **WebSphere Application Server** デプロイメント・マネージャー内でホスティングされている場合、ポートは通常 9809 です。

外部ドメインに指定するポートは、非 **JMX** ポートです。 **eXtreme Scale** クライアントで使用する通常のポートです。

新規リンクを追加するコマンドの発行後に、カタログ・サービスは外部ドメインへの複製を開始するように、管理下のすべてのコンテナに指示します。リンクは、両側には必要ありません。リンクは片側で作成するだけで十分です。

例えば以下のように、コマンド行を使用して、リンクを削除します。

```
xscmd -c dismissLink -cep host:2809 -fd dname
```

このコマンドはドメインのカタログ・サービスに接続して、特定のドメインへの複製を停止するように指示します。リンクの解除は片側からのみ行う必要があります。

## 2 つのカタログ・サービス・ドメイン間のリンク

カタログ・サービス・ドメイン A と B という 2 つドメインがセットアップされた構成を作成するとします。

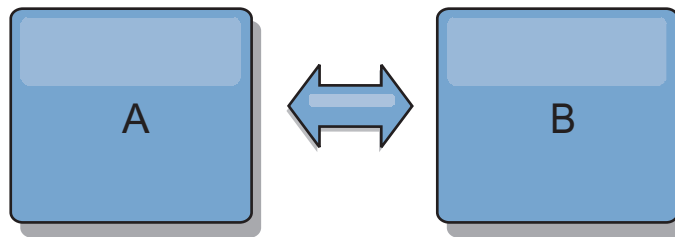


図 32. カタログ・サービス・ドメイン間のリンク

ドメイン A 内のカタログ・サーバーのサーバー・プロパティ・ファイルを以下のようにします。

```
domainName=A  
foreignDomains=B  
B.endPoints=hostB1:2809, hostB2:2809
```

ドメイン B 内のカタログ・サーバーのサーバー・プロパティ・ファイルを以下のようにします。2 つのプロパティ・ファイルが似ていることに注目してください。

```
domainName=B  
foreignDomains=A  
A.endPoints=hostA1:2809,hostA2:2809
```

2 つのドメインを始動すると、以下の特性を備えたすべてのデータ・グリッドがドメイン間で複製されます。

- 固有のドメイン・名前を持つ専用カタログ・サービスがある
- ドメイン内の他のデータ・グリッドと同じグリッド名である
- ドメイン内の他のデータ・グリッドと同じ数の区画がある
- FIXED\_PARTITION データ・グリッドである (PER\_CONTAINER データ・グリッドは複製不可)
- 
- ドメイン内の他のデータ・グリッドと同じデータ・タイプが複製される
- ドメイン内の他のデータ・グリッドと同じマップ・セット名、マップ名、および動的マップ・テンプレートがある

カタログ・サービス・ドメインのレプリカ生成ポリシーは無視されます。

前の例は、他のドメインへのリンクを持つように各ドメインを構成する方法を示していますが、リンクは一方向で定義するだけで十分です。これは、ハブおよびスポ

ーク・トポロジーでは特に便利であり、構成を大幅に単純化できます。スポークの追加時にハブ・プロパティ・ファイルを更新する必要はありません。各スポーク・ファイルにハブの情報を含めるだけで十分です。同様に、リング・トポロジーでは、各ドメインに、リング内の前と次のドメインへのリンクを含めるだけで十分です。

例: ハブおよびスポーク・トポロジー

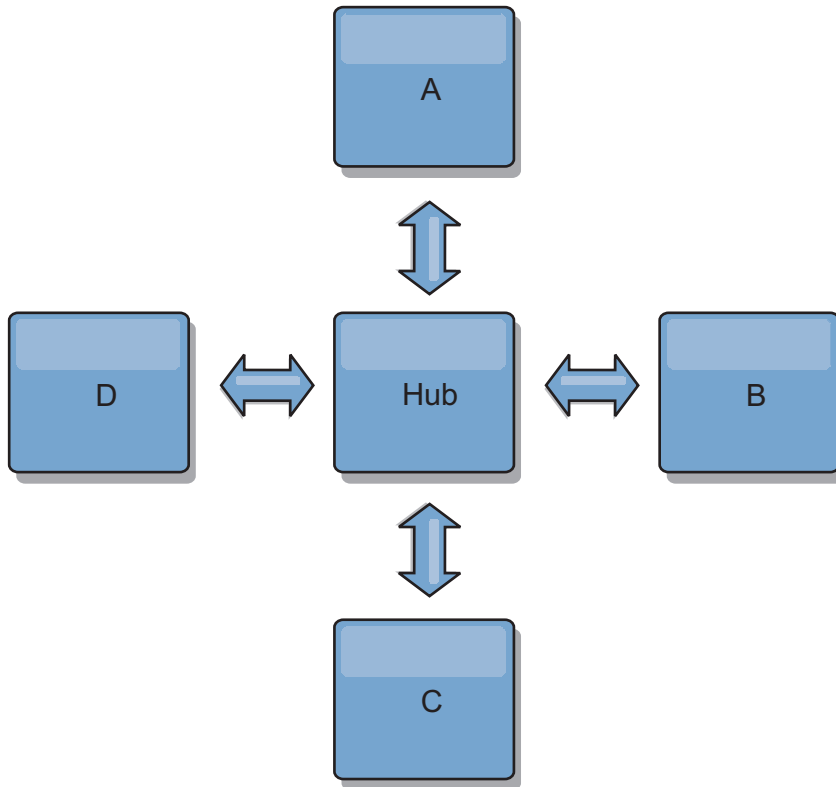


図 33. ハブおよびスポーク・トポロジー

ハブと 4 つのスポーク (ドメイン A、B、C、D) の場合、以下の例のようなサーバー・プロパティ・ファイルになります。

```
domainName=Hub
```

スポーク A のサーバー・プロパティは次のとおりです。

```
domainName=A  
foreignDomains=Hub  
Hub.endPoints=hostH1:2809, hostH2:2809
```

スポーク B のサーバー・プロパティは次のとおりです。

```
domainName=B  
foreignDomains=Hub  
Hub.endPoints=hostH1:2809, hostH2:2809
```

スポーク C のサーバー・プロパティは次のとおりです。

```
domainName=C  
foreignDomains=Hub  
Hub.endPoints=hostH1:2809, hostH2:2809
```

スポーク D のサーバー・プロパティは次のとおりです。

```
domainName=D  
foreignDomains=Hub  
Hub.endPoints=hostH1:2809, hostH2:2809
```

## 次のタスク

カタログ・サービス・ドメイン間の競合を解決するカスタム競合アービターを指定できます。詳しくは、マルチマスター・レプリカ生成のためのカスタム・アービターの作成を参照してください。

---

## ポートの構成

WebSphere eXtreme Scale は、分散キャッシュであり、Java 仮想マシン (JVM) や他のサーバー間でオブジェクト・リクエスト・ブローカー (ORB) および伝送制御プロトコル (TCP) スタックと通信するためにオープン・ポートを必要とします。

## スタンドアロン・モードでのポートの構成

eXtreme Scale デプロイメント内のサーバーおよびクライアントに必要なポートは、コマンド行パラメーターまたはプロパティ・ファイルを使用して構成するか、プログラムで構成できます。以降のセクションに記載するほとんどの例は、**startOgServer** スクリプトへのコマンド行パラメーターについての説明です。組み込みのサーバー API またはクライアント API を使用してプロパティ・ファイル内に同等の構成オプションを設定することもできます。

### 手順

1. カタログ・サービス・エンドポイントを開始します。

WebSphere eXtreme Scale は、Java 仮想マシン間の通信に IIOP を使用します。カタログ・サービス JVM は、IIOP サービス・ポートとグループ・サービス・ポートに関してポートの明示的な構成を必要とする唯一のプロセスです。他のプロセスはポートを動的に割り振ります。

- a. クライアント・ポートとピア・ポートを指定します。カタログ・サービス・ドメイン内のカタログ・サービス間の通信には、クライアント・ポートとピア・ポートが使用されます。クライアント・ポートとピア・ポートを指定するには、次のコマンド行オプションを使用します。

**-catalogServiceEndpoints <serverName:hostName:clientPort:peerPort>**

コンテナでは、カタログ・サービス上のオブジェクト・リクエスト・ブローカー (ORB) のホストとポートを参照します。各属性の定義は次のとおりです。

#### **serverName**

起動しようとしているプロセスを識別する名前を指定します。

#### **hostName**

サーバーを起動するコンピューターのホスト名を指定します。

#### **clientPort**

ピア・カタログ・サービス通信に使用されるポートを指定します。

## peerPort

この値は、haManagerPort と同じです。ピア・カタログ・サービス通信に使用されるポートを指定します。

次の例は、cs1 カatalog・サーバーを始動するものです。このサーバーは、cs2 および cs3 サーバーと同じカatalog・サービス・ドメイン内にあります。

```
startOgServer.bat|sh cs1 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

catalogServiceEndPoints サーバー・プロパティを使用してカatalog・サービス・エンドポイントを設定することもできます。

- b. オブジェクト・リクエスト・ブローカー (ORB) リスナー・ポートを設定します。カatalog・サービス・ドメイン内のカatalog・サービス間の通信、またカatalog・サービスとコンテナ・サーバーおよびクライアントとの通信にもオブジェクト・リクエスト・ブローカー (ORB) リスナー・ポートが使用されます。リスナー・ポートとリスナー・ホストを指定するには、次のコマンド行オプションを使用します。

### -listenerHost <host name>

Internet Inter-ORB Protocol (IIOP) との通信用に、オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のホスト名を指定します。この値は、完全修飾ドメイン名または IP アドレスである必要があります。ご使用の構成に複数のネットワーク・カードが含まれている場合は、リスナー・ホストとリスナー・ポートを設定し、バインドする IP アドレスを JVM 内のオブジェクト・リクエスト・ブローカーに通知します。使用する IP アドレスを指定しなければ、接続タイムアウトや異常な API 障害、クライアントがハングしたように見える状態などの症状が現れることがあります。デフォルト: localhost

### -listenerPort <port>

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポート番号を指定します。この設定は、コンテナおよびクライアントが ORB を介してカatalog・サービスと通信するように構成します。

WebSphere Application Server では、listenerPort は BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパティは、コンテナ・サーバーとカatalog・サービスの両方に適用されます。デフォルト: 2809

listenerHost サーバー・プロパティおよび listenerPort サーバー・プロパティを使用してリスナー・ポートとリスナー・ホストを設定することもできます。

- c. オプション: JMX サービス・ポートを設定します。

JMX クライアントからの通信には、JMX サービス・ポートが使用されます。JMX サービス・ポートを指定するには、次のコマンド行オプションを使用します。

#### **-JMXServicePort <port>**

MBean サーバーが Java Management Extensions (JMX) との通信を listen するポート番号を指定します。JMXServicePort プロパティは、JMX 用の非 SSL ポートを指定します。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**JMXServicePort** とポート番号を明示的に指定してください。このプロパティは、コンテナー・サーバーとカタログ・サービスの両方に適用されます。(スタンドアロン環境の場合のみ必須。)

**デフォルト:** 1099 (カタログ・サーバーの場合)

JMXServicePort サーバー・プロパティを使用して JMX サービス・ポートを設定することもできます。

- d. オプション: JMX コネクター・ポートを設定します。

JMX クライアントからの通信には、JMX コネクター・ポートが使用されます。JMX コネクター・ポートを指定するには、次のコマンド行オプションを使用します。

#### **-JMXConnectorPort <port>**

Java Management Extensions (JMX) サービスのバインド先の Secure Sockets Layer (SSL) ポートを定義します。

JMXConnectorPort サーバー・プロパティを使用して JMX コネクター・ポートを設定することもできます。

- e. Secure Sockets Layer (SSL) ポートを設定します。

セキュリティが有効になっている場合、SSL ポートも必要になります。SSL ポートを指定するには、次のコマンド行オプションを使用します。

```
-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>
```

```
./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809  
-catalogServiceEndpoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
```

図 34. コマンド行の使用例: hostA で最初のカatalog・サーバーを始動します。コマンドの例は以下のとおりです。

hostB で 2 番目のカatalog・サーバーを始動します。コマンドの例は以下のとおりです。

```
./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809  
-catalogServiceEndpoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
```

- 2. コンテナー・サーバー・エンドポイントを開始します。

次のコマンドは、サンプル・カatalog・サービスで使用するコンテナー JVM を開始します。

```
./startOgServer.sh c0 -catalogServiceEndpoints hostA:2809,hostB:2809
```

コンテナー・サーバー Java 仮想マシンは 2 つのポートを使用します。ピア・コンテナー・サーバーとカatalog・サーバー間の内部通信には、HA マネージャー・ポートが使用されます。ピア・コンテナー・サーバー、カatalog・サーバ



一、およびクライアント間の IIOP 通信には、リスナー・ポートが使用されま  
す。リスナー・ホストは、ORB を特定のネットワーク・アダプターにバインド  
するときに使用されます。これらのポートを指定しない場合、両方のポートが動  
的に選択されます。しかし、ファイアウォール環境などで、ポートを明示的に構  
成する必要がある場合は、コマンド行オプションを使用して ORB ポートを指定  
できます。

- a. リスナー・ホストとリスナー・ポートを指定します。リスナー・ポートとリス  
ナー・ホストを指定するには、次のコマンド行オプションを使用します。

**-listenerHost <host name>**

Internet Inter-ORB Protocol (IIOP) との通信用に、オブジェクト・リクエ  
スト・ブローカー (ORB) がバインドする先のホスト名を指定します。こ  
の値は、完全修飾ドメイン名または IP アドレスである必要があります。  
ご使用の構成に複数のネットワーク・カードが含まれている場合は、リス  
ナー・ホストとリスナー・ポートを設定し、バインドする IP アドレスを  
JVM 内のオブジェクト・リクエスト・ブローカーに通知します。使用す  
る IP アドレスを指定しなければ、接続タイムアウトや異常な API 障  
害、クライアントがハングしたように見える状態などの症状が現れること  
があります。**デフォルト: localhost**

**-listenerPort <port>**

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポー  
ト番号を指定します。この設定は、コンテナおよびクライアントが  
ORB を介してカタログ・サービスと通信するように構成します。  
WebSphere Application Server では、listenerPort は  
BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパ  
ティは、コンテナ・サーバーとカタログ・サービスの両方に適用され  
ます。**デフォルト: 2809**

listenerHost サーバー・プロパティおよび listenerPort サーバー・プロパテ  
ィーを使用してリスナー・ポートとリスナー・ホストを設定することもでき  
ます。

- b. HA マネージャー・ポートを指定します。HA マネージャー・ポートを指定  
するには、次のコマンド行オプションを使用します。

**-haManagerPort <port>**

HA マネージャーが使用するポート番号を指定します。このプロパティ  
が設定されていない場合、空きポートが選択されます。このプロパティ  
は、WebSphere Application Server 環境では無視されます。

HAManagerPort サーバー・プロパティを使用して HA マネージャー・ポー  
トを設定することもできます。

- c. オプション: SSL ポートを指定します。

セキュリティが有効になっている場合、Secure Sockets Layer (SSL) ポート  
も必要になります。SSL ポートを指定するには、次のコマンド行オプション  
を使用します。

`-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>`

- d. オプション: JMX サービス・ポートを指定します。

#### **-JMXServicePort <port>**

MBean サーバーが Java Management Extensions (JMX) との通信を listen するポート番号を指定します。JMXServicePort プロパティーは、JMX 用の非 SSL ポートを指定します。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**JMXServicePort** とポート番号を明示的に指定してください。このプロパティーは、コンテナー・サーバーとカタログ・サービスの両方に適用されます。(スタンドアロン環境の場合のみ必須。)

**デフォルト:** 1099 (カタログ・サーバーの場合)

JMXServicePort サーバー・プロパティーを使用して JMX サービス・ポートを設定することもできます。

- e. オプション: JMX コネクター・ポートを設定します。

JMX クライアントからの通信には、JMX コネクター・ポートが使用されます。JMX コネクター・ポートを指定するには、次のコマンド行オプションを使用します。

#### **-JMXConnectorPort <port>**

Java Management Extensions (JMX) サービスのバインド先の Secure Sockets Layer (SSL) ポートを定義します。

JMXConnectorPort サーバー・プロパティーを使用して JMX コネクター・ポートを設定することもできます。

3. クライアント・エンドポイントを開始します。

クライアントが必要とするのは、カタログ・サービスのリスナー・エンドポイントのみです。クライアントは、カタログ・サービスから自動的にコンテナー・サーバー Java 仮想マシン (すなわち、データを保管している Java 仮想マシン) のエンドポイントを取得します。前述の例でカタログ・サービスに接続するには、クライアントは、以下の `host:port` のペアのリストを接続 API に渡さなければなりません。

```
hostA:2809,hostB:2809
```

クライアントは、DataGrid API を使用する場合、コンテナー・サーバーからコールバックを受け取ることもできます。これらのコールバックは、IIOP を使用して ORB リスナー・ポートと通信します。コールバックを受け取るポートおよびネットワーク・アダプターを指定するには、クライアント・プロパティー・ファイル内で **listenerHost** および **listenerPort** プロパティーを設定します。

セキュリティーが有効になっている場合、Secure Sockets Layer (SSL) ポートも必要になります。SSL ポートを指定するには、クライアント・プロセスの開始時に次のシステム・プロパティーを使用します。

```
-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>
```

## WebSphere Application Server 環境でのポートの構成

WebSphere eXtreme Scale カタログ・サービス、コンテナ・サーバー、およびクライアントは、WebSphere Application Server プロセス内で実行される時、プロセスに既に定義されているポートとサービスを使用します。

### このタスクについて

次のセクションでは、デプロイメント内でのポートの使用についての詳細を説明します。

#### 1. カタログ・サービス・エンドポイント

WebSphere eXtreme Scale カタログ・サービスは、WebSphere Application Server プロセス内で稼働し、管理コンソールまたは管理用タスクを使用して構成されず。ポートはすべてプロセスから継承しますが、クライアント・ポートは例外で、このポートは明示的に構成されます。カタログ・サービスが使用するポートについて詳しくは、53 ページの『ネットワーク・ポートの計画』を参照してください。カタログ・サービス・ドメインの構成について詳しくは、高可用性カタログ・サービスを参照してください。

#### 2. コンテナ・サーバー・エンドポイント

WebSphere eXtreme Scale コンテナ・サーバーは、Java EE モジュール内でホスティングされます。コンテナ・サーバーは、アプリケーション・サーバー・プロセス用に定義されているポートを使用します。コンテナ・サービスが使用するポートについて詳しくは、53 ページの『ネットワーク・ポートの計画』を参照してください。Enterprise JavaBeans™ (EJB) や Web モジュールなどの Java EE モジュール内でコンテナを開始する方法について詳しくは、314 ページの『コンテナ・サーバーの自動始動のための WebSphere Application Server アプリケーションの構成』を参照してください。

#### 3. クライアント・エンドポイント

WebSphere eXtreme Scale クライアントは、Java EE Web モジュールまたは EJB モジュール内でホスティングされます。

クライアントは、ObjectGridManager.connect() API を使用してカタログ・サービス・ドメインにプログラマチックに接続します。接続先が同一セル内でホスティングされるカタログ・サービス・ドメインである場合、クライアント接続は、ObjectGridManager に対し次の API 呼び出しを使用して、デフォルトのカタログ・サービス・ドメインを自動的に検出します。

```
connect(securityProps, overrideObjectGridXML)
```

デフォルトのカタログ・サービス・ドメインがリモート側 (セルの外部) でホスティングされている場合は、ObjectGridManager API の次のメソッドを使用して、カタログ・サービス・エンドポイントを指定する必要があります。

```
connect(catalogServerEndpoints, securityProps, overrideObjectGridXML)
```

デフォルトのカタログ・サービス・ドメインがセル内に定義されている場合は、CatalogServerProperties API を使用して、カタログ・サーバーのアドレスを取得できません。XSDomainManagement 管理用タスクを使用しても、構成済みカタログ・サービス・ドメイン・エンドポイントを取得できません。

## 複数のネットワーク・カードを含むサーバー

複数のネットワーク・カードを含むサーバー上で eXtreme Scale プロセスを実行できます。

サーバーまたはクライアントが複数のネットワーク・カードを含むサーバー上で実行されている場合は、ネットワーク・カードとホスト名を eXtreme Scale 構成で指定して、指定したカードをバインドする必要があります。この構成を指定しないと、eXtreme Scale ランタイムが自動的に 1 つを選択し、接続の失敗またはパフォーマンスの低下が生じる可能性があります。

カタログ・サーバーまたはコンテナ・サーバーの場合は、以下のいずれかの方法でリスナー・ホストとリスナー・ポートを設定する必要があります。

- サーバー・プロパティ
- startOgServer.shlbat スクリプトでのコマンド行パラメーター。

クライアントでは、コマンド行を使用できず、クライアント・プロパティを使用しなければなりません。

---

## トランスポートの構成

トランスポートは、構成内の異なるサーバー・プロセス間でオブジェクトとデータの交換を可能にします。

### このタスクについて

メインになるトランスポート・メカニズムはオブジェクト・リクエスト・ブローカー (ORB) です。このメカニズムはキャッシュ・エントリーを Java ヒープに保管します。

次の構成シナリオでは、トランスポート・メカニズムとして ORB を使用する必要があります。

- x86 64 ビット Linux 以外のシステムを使用する場合
- WebSphere Application Server 環境内で稼働するコンテナ・サーバーを使用する場合
- evictor プラグインまたは複合索引を使用する場合

現在 eXtremeMemory を使用している場合は、eXtremeIO という新しいトランスポートが使用されます。eXtremeMemory では、キャッシュ・エントリーはネイティブ・メモリー内に保管されます。ネイティブ・メモリーはガーベッジ・コレクションの対象にならないため、より安定したパフォーマンスを得られるうえ、応答時間も予測可能になります。オブジェクトは、コンテナ・サーバー上でバイト列にシリアルライズされます。詳しくは、317 ページの『IBM eXtremeMemory および IBM eXtremeIO の構成』を参照してください。

## オブジェクト・リクエスト・ブローカーの構成

オブジェクト・リクエスト・ブローカー (ORB) は、TCP スタックを介した通信のために WebSphere eXtreme Scale によって使用されます。orb.properties ファイルを使用して ORB が使用するプロパティを受け渡し、データ・グリッドのトラン

サポート動作を変更します。WebSphere eXtreme Scale または WebSphere Application Server で提供される ORB を WebSphere eXtreme Scale サーバーに使用する場合、何のアクションも必要ありません。

## WebSphere Application Server 環境でのオブジェクト・リクエスト・ブローカーの構成

WebSphere Application Server または WebSphere Application Server Network Deployment 環境内でオブジェクト・リクエスト・ブローカー (ORB) を直接使用するアプリケーションに WebSphere eXtreme Scale を使用できます。

### 手順

1. アプリケーション・サーバーに適切な名前を付けます。

サーバーが ORB を使用して互いに通信する場合には、WebSphere Application Server 環境内に同じ名前のサーバーが存在することはできません。同じ名前のプロセスでシステム・プロパティー

**-Dcom.ibm.websphere.orb.uniqueServerName=true** を指定することで、この制限を解決できます。例えば、各ノードにある `server1` という名前のサーバーがカタログ・サービス・ドメインとして使用される場合や、複数のノード・エージェントを使用してカタログ・サービス・ドメインが形成される場合などです。

2. WebSphere Application Server 構成内の ORB プロパティーを調整します。

調整可能なプロパティーの詳細については、560 ページの『ORB プロパティー』を参照してください。プロパティーによって、設定の変更は管理コンソールか `was_rootproperties/orb.properties` ファイルで行います。

3. 複数のネットワーク・インターフェース・カードを使用する場合、WebSphere Application Server 管理コンソールの「ポート」パネルで ORB\_LISTENER\_ADDRESS の値を設定する必要があります。構成内のアプリケーション・サーバーごとに、このステップを繰り返します。

- a. 「サーバー」 > 「アプリケーション・サーバー」 > 「`server_name`」をクリックし、対象のアプリケーション・サーバーを選択します。「通信」の下の「ポート」をクリックします。指定されたサーバーの「ポート」パネルが表示されます。

- b. 「詳細」をクリックし、ORB\_LISTENER\_ADDRESS の値を編集します。

- c. 「ホスト」フィールドに IP アドレスを入力します。マルチネットワーク・インターフェース環境の場合、この値は専用アドレスでなければなりません。

注: DNS ホスト名は、ORB\_LISTENER\_ADDRESS の値としてはサポートされません。

- d. 「ポート」フィールドに、ポート番号を入力します。ポート番号は、クライアント要求を受け入れるようにサービスが構成されているポートを指定します。ポート値は、ホスト名とともに使用します。

### 次のタスク

**wxsLogAnalyzer** ツールを使用して、環境内の ORB 設定を検査できます。詳しくは、622 ページの『ログおよびトレース・データの分析』を参照してください。



## オブジェクト・リクエスト・ブローカーとスタンドアロン WebSphere eXtreme Scale プロセスの構成

WebSphere Application Server または WebSphere Application Server Network Deployment を含まない環境で、オブジェクト・リクエスト・ブローカー (ORB) を直接使用するアプリケーションを使用して WebSphere eXtreme Scale を使用することができます。

### 始める前に

eXtreme Scale に組み込まれていないアプリケーション、あるいは他のコンポーネントやフレームワークを実行中に、eXtreme Scale と同じプロセス内で ORB を使用する場合は、追加のタスクを実行して、ご使用の環境で eXtreme Scale が正しく実行されていることを確認する必要があります。

### このタスクについて

ご使用の環境で ORB の使用を初期化するには、orb.properties ファイルに **ObjectGridInitializer** プロパティを追加します。ORB を使用して、ご使用の環境にある eXtreme Scale プロセスと別のプロセスの間の通信を使用可能にします。

### 手順

1. スタンドアロン・インストールでは、orb.properties ファイルが組み込まれません。orb.properties ファイルを java/jre/lib ディレクトリー内に配置する必要があります。プロパティと設定の説明は、560 ページの『ORB プロパティ』を参照してください。
2. orb.properties ファイルに次の行を入力し、変更を保存します。

```
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
```

### タスクの結果

eXtreme Scale は ORB を正しく初期化し、その ORB が使用可能に設定されている他のアプリケーションと共存できます。

eXtreme Scale でカスタム・バージョンの ORB を使用する場合は、『カスタム・オブジェクト・リクエスト・ブローカーの構成』を参照してください。

### 次のタスク

xsLogAnalyzer ツールを使用して、環境内の ORB 設定を検査できます。詳しくは、622 ページの『ログおよびトレース・データの分析』を参照してください。

### カスタム・オブジェクト・リクエスト・ブローカーの構成

WebSphere eXtreme Scale は、プロセス間の通信を使用可能にするために Object Request Broker (ORB) を使用します。WebSphere eXtreme Scale または WebSphere Application Server が提供する Object Request Broker (ORB) を、ご使用の WebSphere eXtreme Scale サーバーに使用する際は、何のアクションも必要ありません。必要な作業はほとんどなく、ご使用の WebSphere eXtreme Scale クライアント用に同じ ORB を使用することができます。しかし、カスタム ORB を使用する必要がある場合は、選択肢として IBM SDK に付属の ORB をお勧めしますが、その



ORB を使用するには構成が必要です。他のベンダーから提供される ORB を使用することも可能です。この場合も構成が必要になります。

### 始める前に

WebSphere eXtreme Scale または WebSphere Application Server で提供される ORB、IBM SDK で提供される ORB、外部ベンダーの ORB のうち、どれを使用するかを決定します。

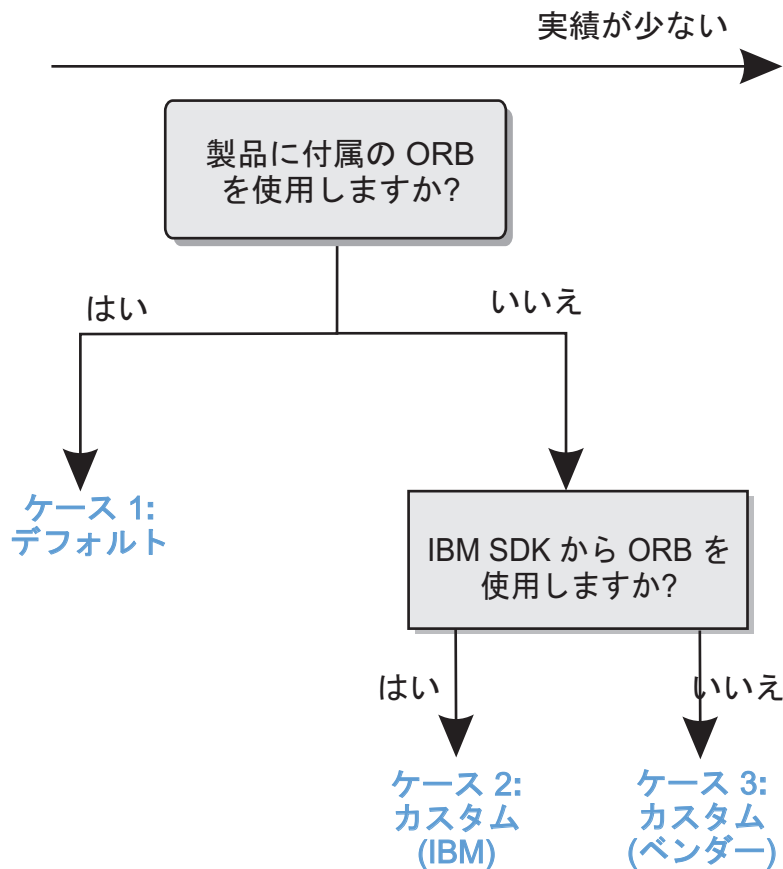


図 35. ORB の選択

WebSphere eXtreme Scale サーバー・プロセス用と WebSphere eXtreme Scale クライアント・プロセス用に、別々に決定することができます。eXtreme Scale は、ほとんどのベンダーの開発キットをサポートしていますが、サーバー・プロセスとクライアント・プロセスの両方において、eXtreme Scale で提供される ORB を使用することを推奨します。eXtreme Scale は、Oracle Java Development Kit (JDK) で提供される ORB はサポートしません。

### このタスクについて

選択した ORB の使用に必要な構成を、よく理解してください。

#### ケース 1: デフォルトの ORB

- WebSphere eXtreme Scale サーバー・プロセスでは、WebSphere eXtreme Scale または WebSphere Application Server で提供される ORB を使用するために必要とされる構成はありません。
- WebSphere eXtreme Scale クライアント・プロセスでは、WebSphere eXtreme Scale または WebSphere Application Server で提供される ORB を使用するために、最小限のクラスパス構成が必要となります。

### ケース 2: カスタム ORB (IBM)

IBM SDK で提供される ORB を使用するために WebSphere eXtreme Scale のクライアント・プロセスを構成するには、このトピックで後述する説明を参照してください。IBM SDK を使用する場合も、他の開発キットを使用する場合も、IBM ORB を使用できます。IBM SDK バージョン 5 以降を使用できます。

### ケース 3: カスタム ORB (外部ベンダー提供)

WebSphere eXtreme Scale のクライアント・プロセスにベンダーの ORB を使用するケースは、実施されたテストが最も少ないオプションです。独立系ソフトウェア・ベンダーの ORB を使用して発生した問題は、IBM ORB で再現可能で、JRE との互換性がないとサポートに問い合わせることはできません。

Oracle Java Development Kit (JDK) で提供される ORB は、サポート対象外です。

## 手順

- デフォルト ORB の 1 つを使用するようクライアント・プロセスを構成します (ケース 1)。 次の JVM 引数を使用します。 :

```
-jvmArgs -Djava.endorsed.dirs=default_ORB_directory${pathSeparator}JRE_HOME/lib/endorsed
```

デフォルトの ORB ディレクトリーは、`wxs_home/lib/endorsed` です。  
`orb.properties` ファイルにある次のプロパティーの更新が必要な場合もあります。

```
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
```

- IBM SDK バージョン 5 を使用するようクライアント・プロセスまたはサーバー・プロセスを構成します (ケース 2)。
  1. ORB Java アーカイブ (JAR) ファイルを空のディレクトリーまたは `custom_ORB_directory` にコピーします。
    - `ibmorb.jar`
    - `ibmorbapi.jar`
  2. `custom_ORB_directory` ディレクトリーを、Java コマンドを開始するスクリプト内の承認済みディレクトリーとして指定します。

**ヒント:** Java コマンドで既に承認済みディレクトリーを指定している場合、`custom_ORB_directory` ディレクトリーをその既存の承認済みディレクトリーの下に配置するというオプションもあります。`custom_ORB_directory` ディレクトリーを既存の承認済みディレクトリーの下に配置すると、スクリプトを更新する必要がなくなります。いずれにせよ、このスクリプトを更新する場合は、既存の引数を完全に置き換えるのではなく、必ず既存の

-Djava.endorsed.dirs= 引数に接頭部として *custom\_ORB\_directory* ディレクトリーを追加するようにしてください。

- スタンドアロンの eXtreme Scale 環境用にスクリプトを更新します。

```
setupCmdLine.bat|sh ファイルの OBJECTGRID_ENDORSED_DIRS 変数の  
パスを編集して、custom_ORB_directory を指定します。変更を保存します。
```

- eXtreme Scale が WebSphere Application Server 環境に組み込まれている場合、スクリプトを更新します。

```
startOgServer スクリプトに以下のシステム・プロパティーとパラメータ  
を追加します。
```

```
-jvmArgs -Djava.endorsed.dirs=custom_ORB_directory
```

- クライアント・アプリケーション・プロセスまたはサーバー・プロセスの開始に使用するカスタム・スクリプトを更新します。

```
-Djava.endorsed.dirs=custom_ORB_directory
```

---

## クライアントの構成

スタンドアロン環境で実行する WebSphere eXtreme Scale を構成することも、WebSphere Application Server を使用する環境で実行する eXtreme Scale を構成することもできます。グリッドのサーバー・サイドの構成変更を WebSphere eXtreme Scale デプロイメントに反映するには、動的に適用するのではなく、プロセスを再始動して変更を有効にする必要があります。一方、クライアント・サイドでは、既存のクライアント・インスタンスの構成設定は変更できませんが、XML ファイルを使用するか、プログラムで、新しいクライアント・インスタンスを必要な設定で作成できます。クライアントの作成時には、現行のサーバー構成からのデフォルト設定をオーバーライド可能です。

次の方法で、eXtreme Scale クライアントを構成することができます。いずれの方法も、クライアント・オーバーライド XML ファイルを使用して、もしくはプログラムで実行することができます。

- XML 構成
- プログラマチック構成
- Spring Framework 構成
- ニア・キャッシュの使用不可化

クライアント上で以下のプラグインをオーバーライドできます。

- **ObjectGrid** プラグイン
  - TransactionCallback プラグイン
  - ObjectGridEventListener プラグイン
- **BackingMap** プラグイン
  - Evictor プラグイン
  - MapEventListener プラグイン
  - numberOfBuckets 属性
  - ttlEvictorType 属性
  - timeToLive 属性

## XML 構成を使用したクライアントの構成

ObjectGrid 構成 XML ファイルを使用して、クライアント・サイドの設定を変更できます。

### このタスクについて

WebSphere eXtreme Scale クライアントの設定を変更するには、コンテナ・サーバーに使用したファイルと似た構造の ObjectGrid XML ファイルを作成する必要があります。

以下のクライアントの設定をオーバーライドできます。

1. クライアント固有の ObjectGrid インスタンスを作成します。
2. サーバーの開始に使用された ObjectGrid XML ファイルをコピーします。
3. その新規ファイルを編集してクライアント・サイドでカスタマイズします。
  - クライアントの属性を設定または更新するには、新しい値を指定するか、あるいは既存の値を変更します。
  - クライアントからプラグインを除去するには、className 属性の値として空ストリングを使用します。
  - 既存のプラグインを変更するには、className 属性に新しい値を指定します。
  - クライアント・オーバーライドでサポートされるプラグイン (TRANSACTION\_CALLBACK、OBJECTGRID\_EVENT\_LISTENER、EVICTOR、MAP\_EVENT\_LISTENER) を追加することもできます。
4. 新規に作成されたクライアント・オーバーライド XML ファイルを使用して、クライアントを作成します。

### 手順

1. コンテナ・サーバー用のファイルと似た構造の ObjectGrid 構成 XML ファイルをクライアント用に作成します。

以下の XML ファイルがデプロイメント・ポリシー XML ファイルと対になっており、これらのファイルを使用してコンテナ・サーバーが開始されたものと想定します。

#### companyGridServerSide.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
    <objectGrid name="CompanyGrid">
      <bean id="TransactionCallback"
        className="com.company.MyTxCallback" />
      <bean id="ObjectGridEventListener"
        className="com.company.MyOgEventListener" />
      <backingMap name="Customer"
        pluginCollectionRef="customerPlugins" />
      <backingMap name="Item" />
      <backingMap name="OrderLine" numberOfBuckets="1049"
        timeToLive="1600" ttlEvictorType="LAST_ACCESS_TIME" />
      <backingMap name="Order" lockStrategy="PESSIMISTIC"
        pluginCollectionRef="orderPlugins" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="customerPlugins">
      <bean id="Evictor" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

```

        className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor" />
        <bean id="MapEventListener"
            className="com.company.MyMapEventListener" />
    </backingMapPluginCollection>
</backingMapPluginCollection id="orderPlugins">
    <bean id="MapIndexPlugin"
        className="com.company.MyMapIndexPlugin" />
</backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>

```

コンテナ・サーバーでは、CompanyGrid という名前の ObjectGrid インスタンスが companyGridServerSide.xml ファイルによる定義に従って動作します。デフォルトでは CompanyGrid クライアントの設定は、サーバーで実行している CompanyGrid インスタンスの設定と同じです。

以下の ObjectGrid XML ファイルを使用すると、CompanyGrid クライアントの属性およびプラグインのいくつかを指定できます。

companyGridClientSide.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyClientTxCallback" />
            <bean id="ObjectGridEventListener" className="" />
            <backingMap name="Customer" numberOfBuckets="1429"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" numberOfBuckets="701"
                timeToLive="800" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor" />
            <bean id="MapEventListener" className="" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>

```

定義されたオーバーライドを要約すると次のようになります。

- クライアントの TransactionCallback は、サーバー・サイドで設定されている com.company.MyTxCallback ではなく、com.company.MyClientTxCallback になります。
- className 値が空ストリングであるため、クライアントには ObjectGridEventListener プラグインがありません。
- クライアントは Customer backingMap に対して numberOfBuckets を 1429 に設定し、その Evictor プラグインを保持して、MapEventListener プラグインを除去します。
- OrderLine backingMap の numberOfBuckets 属性および timeToLive 属性は変更されます。
- 異なる lockStrategy 属性が指定されていても、lockStrategy 属性はクライアント・オーバーライドでサポートされていないため、影響はありません。

2. XML ファイルを使用してクライアントを作成します。

companyGridClientSide.xml ファイルを使用して CompanyGrid クライアントを作成するには、ObjectGrid XML ファイルを URL として、ObjectGridManager インターフェースのいずれかの connect メソッドに渡します。

```
ObjectGridManager ogManager =  
    ObjectGridManagerFactory.ObjectGridManager();  
ClientClusterContext clientClusterContext =  
    ogManager.connect("MyServer1.company.com:2809", null, new URL(  
        "file:xml/companyGridClientSide.xml"));
```

## クライアント無効化メカニズムの使用可能化

分散 WebSphere eXtreme Scale 環境では、optimistic ロック・ストラテジーが使用されているか、ロックが無効にされている場合、クライアント・サイドにはデフォルトでニア・キャッシュがあります。ニア・キャッシュにはそれ独自のローカル・キャッシュ・データがあります。eXtreme Scale クライアントが更新をコミットすると、この更新はクライアントのニア・キャッシュとサーバーに送られます。しかし、他の eXtreme Scale クライアントはこの更新情報を受け取らないので、それらのデータは古くなっていることがあります。

### ニア・キャッシュ

アプリケーションは、eXtreme Scale クライアントのこの失効データの問題を認識しておく必要があります。Java Message Service (JMS) ベースの組み込み

ObjectGridEventListener

com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener クラスを使用して、分散 eXtreme Scale 環境内でクライアント無効化メカニズムを使用可能にすることができます。

クライアント無効化メカニズムは、分散 eXtreme Scale 環境におけるクライアントのニア・キャッシュ内の失効データの問題に対する解決方法です。このメカニズムにより、クライアントのニア・キャッシュはサーバーまたは他のクライアントと同期します。ただし、この JMS ベースのクライアント無効化メカニズムを使用しても、クライアントのニア・キャッシュが即時に更新されるわけではありません。ランタイムが更新を公開するときに遅延が発生します。

分散 eXtreme Scale 環境におけるクライアント無効化メカニズムには、次の 2 つのモデルがあります。

- クライアント/サーバー・モデル: このモデルでは、すべてのサーバー・プロセスは、指定された JMS の宛先にすべてのトランザクション変更を公開する publisher ロールにあります。すべてのクライアント・プロセスは受信側ロールにあり、指定された JMS の宛先からすべてのトランザクション変更を受け取ります。
- 二重ロール・モデルとしてのクライアント: このモデルでは、すべてのサーバー・プロセスは JMS の宛先とは無関係です。すべてのクライアント・プロセスは JMS publisher ロールであり、かつ receiver ロールです。クライアントで発生するトランザクション変更は JMS の宛先に公開され、すべてのクライアントがこれらのトランザクション変更を受け取ります。

詳しくは、265 ページの『JMS イベント・リスナー』を参照してください。



## クライアント/サーバー・モデル

クライアント/サーバー・モデルでは、サーバーは JMS publisher ロールにあり、クライアントは JMS receiver ロールにあります。

### client-server model XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener">
        <className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
          <property name="invalidationModel" type="java.lang.String" value="CLIENT_SERVER_MODEL" description="" />
          <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
          <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
          <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
          <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
          <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
          <property name="jms_userid" type="java.lang.String" value="" description="" />
          <property name="jms_password" type="java.lang.String" value="" description="" />
          <property name="jndi_properties" type="java.lang.String"
            value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;
            java.naming.provider.url=
            tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
            description="jndi properties" />
        </bean>

        <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
          lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
          timeToLive="28800" />
        <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
          lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
          timeToLive="2700" />
        <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
          lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
          timeToLive="2700" />
        <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
          lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
          timeToLive="2700" />
        <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
          lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
          timeToLive="2700" />
      </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
      <backingMapPluginCollection id="agent">
        <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
      </backingMapPluginCollection>
      <backingMapPluginCollection id="profile">
        <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
          <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
          <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
          <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
        </bean>
      </backingMapPluginCollection>

      <backingMapPluginCollection id="pessimisticMap" />
      <backingMapPluginCollection id="excludedMap1" />
      <backingMapPluginCollection id="excludedMap2" />
    </backingMapPluginCollections>
  </objectGridConfig>
```

## 二重ロール・モデルとしてのクライアント

二重ロール・モデルとしてのクライアントでは、各クライアントは JMS publisher ロールと receiver ロールの両方を持っています。クライアントは、コミットされたすべてのトランザクション変更を、指定された JMS 宛先に公開し、コミットされたすべてのトランザクション変更を他のクライアントから受け取ります。このモデルでは、サーバーは JMS とは無関係です。

### dual-roles model XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
```

```

<objectGrid name="AgentObjectGrid">
  <bean id="ObjectGridEventListener"
    className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
    <property name="invalidationModel" type="java.lang.String" value="CLIENT_AS_DUAL_ROLES_MODEL" description="" />
    <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
    <property name="mapsToPublish" type="java.lang.String" value="agent;profile;peessimisticMap" description="" />
    <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
    <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
    <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
    <property name="jms_userid" type="java.lang.String" value="" description="" />
    <property name="jms_password" type="java.lang.String" value="" description="" />
    <property name="jndi_properties" type="java.lang.String"
      value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;java.naming.provider.url=
tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
      description="jndi properties" />
    </bean>

  <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
    lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
    timeToLive="28800" />
  <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
    lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
    timeToLive="2700" />
  <backingMap name="peessimisticMap" readOnly="false" pluginCollectionRef="peessimisticMap" preloadMode="false"
    lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
    timeToLive="2700" />
  <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
    lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
    timeToLive="2700" />
  <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
    lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
    timeToLive="2700" />
</objectGrid>
</objectGrids>

<backingMapPluginCollections>
  <backingMapPluginCollection id="agent">
    <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
  </backingMapPluginCollection>
  <backingMapPluginCollection id="profile">
    <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
      <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
      <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
    </bean>
  </backingMapPluginCollection>

  <backingMapPluginCollection id="peessimisticMap" />
  <backingMapPluginCollection id="excludedMap1" />
  <backingMapPluginCollection id="excludedMap2" />
</backingMapPluginCollections>
</objectGridConfig>

```

## 要求再試行タイムアウト値の構成

信頼できるマップの場合、トランザクション要求の再試行タイムアウト値をミリ秒単位で WebSphere eXtreme Scale に指定できます。

### このタスクについて

タイムアウト値はクライアント・プロパティ・ファイルまたはセッション内で構成できます。セッションの値がクライアント・プロパティ設定に優先します。ゼロより大きい値に設定された場合、タイムアウト条件が満たされるか、永続的な障害が起こるまで、要求は試行されます。永続的な障害とは、DuplicateKeyException 例外などです。値ゼロはフェイル・ファースト・モード設定を表し、eXtreme Scale は、どのようなタイプのトランザクションの後であっても、トランザクションを再試行しません。

実行時は、トランザクション・タイムアウト値が再試行タイムアウト値と一緒に使用され、再試行タイムアウトがトランザクション・タイムアウトを超えないようにします。

トランザクションには 2 つのタイプがあります。自動コミット・トランザクションと明示的に `begin` メソッドと `commit` メソッドを使用するトランザクションです。再試行の有効な例外は、これら 2 つのタイプのトランザクション間で異なります。

- セッション内で呼び出されるトランザクションの場合、`CORBA SystemException` および `eXtreme Scale TargetNotAvailable` 例外であれば、トランザクションは再試行されます。
- 自動コミット・トランザクションの場合、`CORBA SystemException` および `eXtreme Scale` アベイラビリティ例外であれば、トランザクションは再試行されます。これらの例外には、`ReplicationVotedToRollbackTransactionException`、`TargetNotAvailable`、および `AvailabilityException` 例外が含まれます。

アプリケーション障害やその他の永続障害はすぐに再発するので、クライアントはトランザクションを再試行しません。このような永続障害には `DuplicateKeyException` や `KeyNotFoundException` 例外があります。例外の後はトランザクションを再試行せず、すべての例外を返すようにするには、フェイル・ファースト設定を使用します。

#### クライアントがトランザクションを再試行する例外

- `ReplicationVotedToRollbackTransactionException` (自動コミットの場合のみ)
- `TargetNotAvailable`
- `org.omg.CORBA.SystemException`
- `AvailabilityException` (自動コミットの場合のみ)
- `LockTimeoutException` (自動コミットの場合のみ)
- `UnavailableServiceException` (自動コミットの場合のみ)

#### トランザクションが再試行されない永続的な例外

- `DuplicateKeyException`
- `KeyNotFoundException`
- `LoaderException`
- `TransactionAffinityException`
- `LockDeadlockException`
- `OptimisticCollisionException`

#### 手順

- タイムアウト値をクライアント・プロパティ・ファイル内に設定します。

クライアントの `requestRetryTimeout` 値を設定するには、クライアント・プロパティ・ファイル内の `requestRetryTimeout` プロパティを追加または変更します。クライアント・プロパティは、デフォルトでは `objectGridClient.properties` ファイルです。`requestRetryTimeout` プロパティはミリ秒単位で指定されます。ゼロより大きい値に設定すると、再試行可能な例外が起こったときに要求は再試行されます。値を 0 に設定すると、例外が起こったときに再試行なしで失敗します。デフォルトの動作を使用するには、このプロパティを除去するか、値を -1 に設定します。`objectGridClient.properties` ファイル内の値の例を次に示します。

```
requestRetryTimeout = 30000
```

requestRetryTimeout 値はミリ秒で指定されます。この例の場合、値が ObjectGrid インスタンスで使用されると、requestRetryTimeout 値は 30 秒です。

- タイムアウト値をプログラマチックに設定します。

クライアント・プロパティをプログラマチックに設定するには、まず最初にアプリケーションの適切な <ロケーション>にクライアント・プロパティ・ファイルを作成します。以下の例では、クライアント・プロパティ・ファイルは、前のセクションの objectGridClient.properties スニペットを参照します。

ObjectGridManager インスタンスに接続した後、前述のようにクライアント・プロパティを設定します。その後、ObjectGrid インスタンスを取得すると、ファイルで定義したクライアント・プロパティがインスタンスに設定されています。クライアント・プロパティ・ファイルを変更する場合は、そのたびに新しい ObjectGrid インスタンスを明示的に取得してください。

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
String objectGridName = "testObjectGrid";
URL clientXML = null;
ClientContext ccc = manager.connect("localhost:2809", null, clientXML);
File file = new File("<location>/objectGridClient.properties");
URL url = file.toURI().toURL();
ccc.setClientProperties(objectGridName, url);
ObjectGrid objectGrid = ogManager.getObjectGrid(ccc, objectGridName);
```

- セッション・コミット時に指定変更ファイルを設定します。

要求再試行タイムアウトをセッションに設定するか、requestRetryTimeout クライアント・プロパティをオーバーライドするには、Session インターフェースの setRequestRetryTimeout(long) メソッドを呼び出します。

```
Session sessionA = objectGrid.getSession();
sessionA.setRequestRetryTimeout(30000);
ObjectMap mapA = sessionA.getMap("payroll");
String key = "key:" + j;
mapA.insert(key, "valueA");
```

クライアント・プロパティ・ファイルに設定されている値に関係なく、このセッションでは現在 30000 ms または 30 秒という requestRetryTimeout 値が使用されています。セッション・インターフェースの詳細については、セッションを使用したグリッド内データへのアクセスを参照してください。

---

## eXtreme Scale 接続ファクトリーの構成

eXtreme Scale 接続ファクトリーは、Java EE アプリケーションがリモート WebSphere eXtreme Scale データ・グリッドに接続できるようにします。カスタム・プロパティを使用してリソース・アダプターを構成してください。

### 始める前に

接続ファクトリーを作成する前に、リソース・アダプターをインストールする必要があります。

### このタスクについて

リソース・アダプターをインストールした後、リモート・データ・グリッドへの eXtreme Scale クライアント接続を表す 1 つ以上のリソース・アダプター接続ファ

クトリーを作成することができます。以下のステップを実行して、リソース・アダプター接続ファクトリーを構成し、アプリケーション内でこれを使用するようにします。

スタンドアロン・リソース・アダプターの場合はノード・スコープに、組み込みリソース・アダプターの場合はアプリケーション内に、eXtreme Scale 接続ファクトリーを作成することができます。WebSphere Application Server 内に接続ファクトリーを作成する方法については、関連トピックを参照してください。

## 手順

1. WebSphere Application Server 管理コンソールを使用して、eXtreme Scale クライアント接続を表す eXtreme Scale 接続ファクトリーを作成します。「管理コンソールにおける Java EE コネクター接続ファクトリーの構成」を参照してください。「一般プロパティ」パネルで接続ファクトリーのプロパティを指定した後、「適用」をクリックして、「カスタム・プロパティ」リンクをアクティブにする必要があります。
2. 管理コンソールの「カスタム・プロパティ」をクリックします。以下のカスタム・プロパティを設定して、リモート・データ・グリッドへのクライアント接続を構成します。

表 26. 接続ファクトリーを構成するためのカスタム・プロパティ

プロパティ名	タイプ	説明
ConnectionName	ストリング	eXtreme Scale クライアント接続の名前。  ConnectionName は、Managed Bean として公開されたとき、接続を識別するのに役立ちます。このプロパティはオプションです。ConnectionName は、指定されないと未定義になります。
CatalogServiceEndpoints	ストリング	カタログ・サービス・ドメインのエンドポイントで、形式は次のとおりです。 <host>:<port>[,<host><port>]。詳しくは、303 ページの『カタログ・サービス・ドメイン設定』を参照してください。  このプロパティは、カタログ・サービス・ドメインが設定されない場合に必須となります。
CatalogServiceDomain	ストリング	WebSphere Application Server で定義されるカタログ・サービス・ドメイン名。詳しくは、285 ページの『カタログ・サーバーおよびカタログ・サービス・ドメインの構成』を参照してください。  このプロパティは、CatalogServiceEndpoints プロパティが設定されない場合に必須となります。
ObjectGridName	ストリング	(オプション) この接続ファクトリーの接続先であるデータ・グリッドの名前。このプロパティが指定されない場合は、アプリケーションが、接続ファクトリーから接続を取得するときに、名前を提供する必要があります。
ObjectGridURL	ストリング	(オプション) クライアント・データ・グリッド・オーバーライド XML ファイルの URL。このプロパティは、同時に ObjectGridResource が指定された場合には無効となります。詳しくは、335 ページの『クライアントの構成』を参照してください。
ObjectGridResource	ストリング	クライアント・データ・グリッド・オーバーライド XML ファイルのリソース・パス。このプロパティはオプションであり、同時に ObjectGridURL が指定された場合は無効となります。詳しくは、335 ページの『クライアントの構成』を参照してください。
ClientPropertiesURL	ストリング	(オプション) クライアント・プロパティ・ファイルの URL。このプロパティは、同時に ClientPropertiesResource が指定された場合には無効となります。詳しくは、クライアント・プロパティ・ファイルを参照してください。
ClientPropertiesResource	ストリング	(オプション) クライアント・プロパティ・ファイルのリソース・パス。このプロパティは、同時に ClientPropertiesURL が指定された場合には無効となります。詳しくは、クライアント・プロパティ・ファイルを参照してください。

WebSphere Application Server では、接続プールを調整したりセキュリティーを管理したりする、その他の構成オプションを使用することもできます。

WebSphere Application Server インフォメーション・センターのトピックへのリンクについては、関連情報を参照してください。

## 次のタスク

アプリケーション内に eXtreme Scale 接続ファクトリー参照を作成します。詳しくは、345 ページの『eXtreme Scale に接続するためのアプリケーションの構成』を参照してください。

## eXtreme Scale 接続ファクトリーを使用するための Eclipse 環境の構成

eXtreme Scale リソース・アダプターには、カスタム接続ファクトリーが含まれています。これらのインターフェースを eXtreme Scale Java Platform, Enterprise Edition (Java EE) アプリケーションで使用するためには、`wxsra.rar` ファイルをワークスペースにインポートし、それをアプリケーション・プロジェクトにリンクする必要があります。

### 始める前に

- Rational® Application Developer バージョン 7 以降または Eclipse Java EE IDE for Web Developers バージョン 1.4 以降をインストールする必要があります。
- サーバー・ランタイム環境を構成する必要があります。

### 手順

1. `wxsra.rar` ファイルをプロジェクトにインポートします。そのためには、「ファイル」 > 「インポート」を選択します。「インポート」ウィンドウが表示されます。
2. 「Java EE」 > 「RAR ファイル」を選択します。「コネクター・インポート」ウィンドウが表示されます。
3. コネクター・ファイルを指定するには、「参照」をクリックして、`wxsra.rar` ファイルを見つけます。リソース・アダプターをインストールすると、`wxsra.rar` ファイルがインストールされています。リソース・アダプター・アーカイブ (RAR) ファイルは次のロケーションにあります。
  - WebSphere Application Server インストールの場合: `wxs_install_root/optionalLibraries/ObjectGrid`
  - スタンドアロン・インストールの場合: `wxs_install_root/ObjectGrid/lib directory`
4. 新しいコネクター・プロジェクトの名前を「コネクター・プロジェクト」フィールドに作成します。デフォルト名の `wxsra` を使用することができます。
5. Java EE サーバー・ランタイム環境を参照するターゲット・ランタイムを選択します。
6. オプションで、「プロジェクトを EAR に追加」を選択し、RAR を既存の EAR プロジェクトに組み込みます。



## タスクの結果

これで、RAR ファイルが Eclipse ワークスペースにインポートされます。

## 次のタスク

次のステップを使用して、他の Java EE プロジェクトから RAR プロジェクトを参照することができます。

1. プロジェクトを右クリックして、「プロパティ」をクリックします。
2. 「Java ビルド・パス」を選択します。
3. 「プロジェクト」タブを選択します。
4. 「追加」をクリックします。
5. `wxsra` コネクター・プロジェクトを選択して、「OK」をクリックします。
6. 再び「OK」をクリックして、「プロパティ」ウィンドウを閉じます。

これで、eXtreme Scale リソース・アダプター・クラスがクラスパスに存在するようになりました。Eclipse コンソールを使用して製品ランタイム JAR ファイルをインストールするための詳細は、スタンドアロン開発環境のセットアップを参照してください。

---

## eXtreme Scale に接続するためのアプリケーションの構成

アプリケーションは、eXtreme Scale 接続ファクトリーを使用して、eXtreme Scale クライアント接続への接続ハンドルを作成します。このタスクを使用して、リソース・アダプター接続ファクトリー参照を構成できます。

### 始める前に

Java Platform, Enterprise Edition (Java EE) アプリケーション・コンポーネント (Enterprise JavaBeans (EJB) コンテナまたはサーブレットなど) を作成します。

### 手順

アプリケーション・コンポーネントに `javax.resource.cci.ConnectionFactory` リソース参照を作成します。リソース参照は、アプリケーション・プロバイダーによりデプロイメント記述子で宣言されます。接続ファクトリーは、カタログ・サービス・ドメインで使用可能な 1 つ以上の指定されたデータ・グリッドと通信するために使用できる eXtreme Scale クライアント接続を表します。

---

## キャッシュ統合の構成

WebSphere eXtreme Scale を他のキャッシュ関連製品と統合することができます。また、WebSphere eXtreme Scale 動的キャッシュ・プロバイダーを使用して、WebSphere eXtreme Scale を WebSphere Application Server 内の動的キャッシュ・コンポーネントに接続することもできます。WebSphere Application Server に対する拡張としてもう 1 つ考えられるのは、HTTP セッションをキャッシュに入れる操作を支援する WebSphere eXtreme Scale HTTP セッション・マネージャーです。

## HTTP セッション・マネージャーの構成

HTTP セッション・マネージャーは、関連するアプリケーションのセッション・レプリカ生成機能を提供します。セッション・マネージャーは Web コンテナと連動して、HTTP セッションを作成し、アプリケーションに関連付けられた HTTP セッションのライフサイクルを管理します。

### WebSphere Application Server での HTTP セッション・マネージャーの構成

WebSphere Application Server はセッション管理機能を備えていますが、要求の数が増えるとパフォーマンスが低下します。WebSphere eXtreme Scale には、セッション・レプリカ生成、高可用性、優れたスケーラビリティ、および堅固な構成オプションを備えたセッション管理実装がバンドルされています。

#### 始める前に

- eXtreme Scale セッション・マネージャーを使用するには、WebSphere eXtreme Scale は、WebSphere Application Server または WebSphere Application Server Network Deployment セルにインストールする必要があります。詳しくは、205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。
- カタログ・サービス・ドメイン内のカタログ・サーバーの Secure Sockets Layer (SSL) が使用可能な場合、または SSL がサポートされるカタログ・サービス・ドメインで SSL を使用する必要がある場合は、WebSphere Application Server 管理コンソールでグローバル・セキュリティを有効にする必要があります。サーバー・プロパティ・ファイルで、transportType 属性を SSL-Required に設定して、カタログ・サーバーに SSL を要求します。グローバル・セキュリティの構成に関して詳しくは、グローバル・セキュリティの設定を参照してください。

#### このタスクについて

WebSphere eXtreme Scale HTTP セッション・マネージャーは、キャッシング用に、組み込みサーバーとリモート・サーバーの両方をサポートします。

##### • 組み込みシナリオ

組み込みシナリオでは、サーブレットが実行される同じプロセス内で WebSphere eXtreme Scale サーバーが相互に連結されています。セッション・マネージャーはローカルの ObjectGrid インスタンスと直接通信できるため、コストのかかるネットワーク遅延を回避することができます。

WebSphere Application Server を使用している場合は、提供された `wxs_home/session/samples/objectGrid.xml` および `wxs_home/session/samples/objectGridDeployment.xml` ファイルを、ご使用の Web アーカイブ (WAR) ファイルの META-INF ディレクトリに配置してください。アプリケーションが始動して、セッション・マネージャーと同じプロセス内の eXtreme Scale コンテナを自動的に始動すると、eXtreme Scale がこれらのファイルを自動的に検出します。

使用するレプリカ生成が同期レプリカ生成か非同期レプリカ生成かということ、および構成するレプリカの数に応じて、`objectGridDeployment.xml` ファイルを変更できます。

#### • リモート・サーバー・シナリオ

リモート・サーバー・シナリオでは、サーブレットとは異なるプロセスでコンテナ・サーバーが実行されます。セッション・マネージャーはリモートのコンテナ・サーバーと通信します。リモートのネットワーク接続のコンテナ・サーバーを使用するためには、カタログ・サービス・ドメインのホスト名およびポート番号によってセッション・マネージャーを構成する必要があります。そうすると、セッション・マネージャーは、eXtreme Scale クライアント接続を使用して、カタログ・サーバーおよびコンテナ・サーバーと通信します。

コンテナ・サーバーを独立したスタンドアロン・プロセス内で開始する場合は、セッション・マネージャーのサンプル・ディレクトリーで提供される `objectGridStandAlone.xml` ファイルと `objectGridDeploymentStandAlone.xml` ファイルを使用して、eXtreme Scale コンテナを開始します。

#### 手順

1. アプリケーションを接合することで、アプリケーションがセッション・マネージャーを使用できるようにします。セッション・マネージャーを使用するためには、適切なフィルター宣言をアプリケーションの `Web` デプロイメント記述子に追加する必要があります。さらに、セッション・マネージャー構成パラメーターが、デプロイメント記述子内のサーブレット・コンテキスト初期化パラメーターという形式でセッション・マネージャーに渡されます。この情報は、以下に示す複数の方法でアプリケーションに導入することができます。

#### • WebSphere Application Server との自動接合

アプリケーションのインストール時に、WebSphere eXtreme Scale HTTP セッション・マネージャーを使用するようにアプリケーションを構成できます。また、アプリケーションまたはサーバーの構成を編集して、WebSphere eXtreme Scale HTTP セッション・マネージャーを使用することもできます。詳しくは、350 ページの『WebSphere Application Server の HTTP セッション管理のためのアプリケーションの自動接続』を参照してください。

#### • カスタム・プロパティを使用したアプリケーションの自動接合

WebSphere Application Server または WebSphere Application Server Network Deployment でアプリケーションを実行している場合には、アプリケーションを手動で接合する必要はありません。

カスタム・プロパティをセルまたはサーバーに追加して、そのスコープにあるすべての `Web` アプリケーションに `splicer.properties` ファイルを設定します。次のステップを実行して、カスタム・プロパティを構成します。

- a. WebSphere Application Server 管理コンソールで、カスタム・プロパティを設定する正しいパスにナビゲートし、`splicer.properties` ファイルの場所を指示します。

- カスタム・プロパティをすべてのアプリケーションまたは特定のアプリケーションに設定するには、「システム管理」 > 「セル」 > 「カスタム・プロパティ」をクリックします。
  - 特定のアプリケーション・サーバー上のすべてのアプリケーションに適用するカスタム・プロパティを設定するには、「アプリケーション・サーバー」 > 「<server\_name>」 > 「管理」 > 「カスタム・プロパティ」をクリックします。プロパティ名は `com.ibm.websphere.xs.sessionFilterProps` で、その値はアプリケーションが必要とする `splicer.properties` ファイルの場所です。ファイルの場所のパスは、例えば `/opt/splicer.properties` です。
- b. `com.ibm.websphere.xs.sessionFilterProps` カスタム・プロパティを追加します。このカスタム・プロパティの値には、編集する `splicer.properties` ファイルのロケーションが指定されています。このファイルは、デプロイメント・マネージャーに存在します。セル・レベルのカスタム・プロパティを使用して特定のアプリケーション用の `splicer.properties` ファイルを指示する必要がある場合は、カスタム・プロパティの名前を `<application_name>,com.ibm.websphere.xs.sessionFilterProps` のように入力します。ここで、`application_name` は、カスタム・プロパティを適用するアプリケーションの名前を示します。

**重要:** 更新済み `splicer.properties` ファイルが、セッション・レプリカ生成のために接合されるアプリケーションをホスティングしているアプリケーション・サーバーを含んでいるすべてのノードで、同じパス上に存在することを確認してください。

使用可能なスコープはセル、サーバー、およびアプリケーションであり、デプロイメント・マネージャーで実行している場合にのみ使用可能です。別のスコープが必要な場合は、Web アプリケーションを手動で接合してください。

**要確認:** 自動接合オプションは、アプリケーションを実行しているすべてのノードの同じパスに `splicer.properties` ファイルが存在する場合にのみ機能する点にも注意してください。Windows ノードと UNIX ノードがともに存在する混合環境では、このオプションは使用できないため、アプリケーションを手動で接合する必要があります。

#### • **addObjectGridFilter** スクリプトによるアプリケーションの接合

eXtreme Scale とともに提供されるコマンド行スクリプトを使用して、フィルター宣言と構成によってアプリケーションをサブレット・コンテキスト初期化パラメーターの形式で接合します。WebSphere Application Server デプロイメントの場合、このスクリプトは `<was_home>/optionalLibraries/ObjectGrid/session/bin/addObjectGridFilter.bat/sh` にあります。スタンドアロン・デプロイメントの場合、スクリプトは `WXS_HOME/ObjectGrid/session/bin/addObjectGridFilter.sh/bat` にあります。

**oddObjectGridFilter** スクリプトは 2 つのパラメーターを使用します。

- アプリケーション - 接合するエンタープライズ・アーカイブ・ファイルへの絶対パス

- 各種構成プロパティが入ったスプライサー・プロパティ・ファイルへの絶対パス

このスクリプトの使用形式は次のとおりです。

#### Windows

```
addObjectGridFilter.bat [ear_file] [splicer_properties_file]
```

#### UNIX

```
addObjectGridFilter.sh [ear_file] [splicer_properties_file]
```

#### UNIX

**UNIX 上の WebSphere Application Server にインストールされている eXtreme Scale の使用例:**

- cd *wxs\_home*/optionalLibraries/ObjectGrid/session/bin
- addObjectGridFilter.sh /tmp/mySessionTest.ear *was\_root*/optionalLibraries/ObjectGrid/session/samples/splicer.properties

#### UNIX

**UNIX 上のスタンドアロン・ディレクトリーにインストールされている eXtreme Scale の使用例:**

- cd *was\_root*/session/bin
- addObjectGridFilter.sh /tmp/mySessionTest.ear *was\_root*/session/samples/splicer.properties

接合されるサブレット・フィルターは構成値のデフォルトを保持します。これらのデフォルト値は、2 番目の引数にあるプロパティ・ファイルで指定する構成オプションでオーバーライドできます。使用できるパラメーターのリストについては、368 ページの『サブレット・コンテキスト初期化パラメーター』を参照してください。

eXtreme Scale インストールとともに提供されるサンプルの `splicer.properties` ファイルを変更して使用することができます。また、各サブレットを拡張することによってセッション・マネージャーを挿入する、**addObjectGridServlets** スクリプトも使用できます。ただし、推奨スクリプトは **addObjectGridFilter** スクリプトです。

#### • Ant ビルド・スクリプトによるアプリケーションの手動接合

WebSphere eXtreme Scale には Apache Ant で使用できる `build.xml` ファイルが同梱されています。このファイルは WebSphere Application Server インストールの `was_root/bin` フォルダーに含まれています。 `build.xml` ファイルを変更して、セッション・マネージャー構成プロパティを変更できます。構成プロパティは `splicer.properties` ファイル内のプロパティ名と同一です。 `build.xml` を変更し、次のコマンドを実行して Ant プロセスを呼び出します。

- **UNIX** ant.sh、ws\_ant.sh

- **Windows** ant.bat、ws\_ant.bat

(UNIX) または (Windows)

## • Web 記述子の手動更新

Web アプリケーションに同梱されている web.xml ファイルを編集して、フィルター宣言、そのサーブレット・マッピング、およびサーブレット・コンテキスト初期化パラメーターが組み込まれるようにします。この方法はエラーを起こしやすいため、使用しないようにしてください。

使用できるパラメーターのリストについては、368 ページの『サーブレット・コンテキスト初期化パラメーター』を参照してください。

2. アプリケーションをデプロイします。サーバーやクラスターに対して通常使用する手順に従ってアプリケーションをデプロイしてください。アプリケーションをデプロイした後、アプリケーションを始動することができます。
3. アプリケーションにアクセスします。これで、セッション・マネージャーおよび WebSphere eXtreme Scale と対話するアプリケーションにアクセスすることができます。

## 次のタスク

アプリケーションの装備時にセッション・マネージャーの構成属性の大多数を変更して、セッション・マネージャーを使用するようにすることができます。これらの属性には、同期または非同期のレプリカ生成、メモリー内セッション・テーブル・サイズなどがあります。アプリケーションの装備時に変更できる属性を別にすれば、アプリケーションのデプロイメント後に変更できるその他の構成属性は、WebSphere eXtreme Scale サーバー・クラスター・トポロジと、それらのクラスターのクライアント (セッション・マネージャー) がそれらのクラスターに接続する方法に関する属性のみです。

**リモート・シナリオの動作:** アプリケーション・セッション・データをホスティングしている全データ・グリッドに Web コンテナ・クライアントから到達できない場合、クライアントは、代わりに WebSphere Application Server の基本 Web コンテナをセッション管理に使用します。次のようなシナリオでは、データ・グリッドに到達できないことがあります。

- Web コンテナとリモート・コンテナ・サーバー間のネットワークの問題
- リモート・コンテナ・サーバーのプロセスが停止した場合

**sessionTableSize** パラメーターによって指定される、メモリー内に保持されるセッション参照の数は、セッションが基本 Web コンテナ内に保管されている場合、そのまま維持されます。セッション数が **sessionTableSize** の値を超えると、最長未使用時間を基にセッションが Web コンテナ・セッション・キャッシュで無効化されます。リモート・データ・グリッドが使用可能になると、Web コンテナ・キャッシュで無効化されたセッションは、リモート・データ・グリッドからデータを取得し、データを新規セッションにロードできます。リモート・データ・グリッド全体が使用不可なまま、セッションがセッション・キャッシュで無効化されると、ユーザー・セッション・データは失われます。このような問題があるため、負荷の下でシステムを実行する場合、実動リモート・データ・グリッド全体をシャットダウンすることはしないでください。

**WebSphere Application Server の HTTP セッション管理のためのアプリケーションの自動接続:**



データ・グリッドへのセッションをパーシストするように WebSphere Application Server アプリケーションを構成できます。このデータ・グリッドは、WebSphere Application Server 内で実行される組み込みコンテナ・サーバー内またはリモート・データ・グリッド内に配置できます。

## 始める前に

WebSphere Application Server で構成を変更する前に、以下の情報が必要です。

- 使用するセッション・データ・グリッドの名前。セッション・データ・グリッドの作成については、346 ページの『WebSphere Application Server での HTTP セッション・マネージャーの構成』を参照してください。
- セッションを管理するために使用するカタログ・サービスが、セッション・アプリケーションをインストールするセルの外側にある場合には、カタログ・サービス・ドメインを作成する必要があります。詳しくは、288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』を参照してください。
- カatalog・サービス・ドメインを構成する際、コンテナ・サーバーが認証を要求する場合は、カタログ・サービス・ドメインのクライアント・セキュリティーを有効にする必要がある可能性があります。これらの設定は、どの CredentialGenerator 実装を使用するかをランタイムに通知します。この実装は、リモート・データ・グリッドに渡す資格情報を生成します。これらの設定の構成について詳しくは、599 ページの『カタログ・サービス・ドメインのクライアント・セキュリティーの構成』を参照してください。
- 次のいずれかのシナリオに対応するようにしたい場合、WebSphere Application Server 管理コンソールでグローバル・セキュリティーが有効になっていること。
  - カatalog・サービス・ドメイン内のカタログ・サーバーの Secure Sockets Layer (SSL) が使用可能である。
  - SSL がサポートされるカタログ・サービス・ドメインの SSL を使用する必要がある。

サーバー・プロパティ・ファイルで、**transportType** 属性を SSL-Required に設定して、カタログ・サーバーに SSL を要求します。グローバル・セキュリティーの構成に関して詳しくは、グローバル・セキュリティーの設定を参照してください。

- バージョン 7.1.0.3 以降を使用している場合は、データ・グリッドへのセッション・トラッキング・メカニズムとして URL 再書き込みまたは Cookie を使用してセッションを保持できます。バージョン 7.1.0.3 より前のリリースでは、セッション・トラッキング・メカニズムとして URL 再書き込みを使用するセッションは、保持できません。URL 再書き込みを使用するセッションのパーシスタンスを有効にするには、自動的にアプリケーションを接合した後に、`splicer.properties` ファイル内の **useURLEncoding** プロパティを `true` に設定します。
- WebSphere Application Server での HTTP セッション管理のためにアプリケーションを自動的に接合する場合、Web アプリケーションをホストしているすべてのアプリケーション・サーバーの **HttpSessionIdReuse** Web コンテナ・カスタム・プロパティが `true` に設定されていること。このプロパティによって、あるアプリケーション・サーバーから別のアプリケーション・サーバーにフェイルオーバーしたセッションや、リモート・シナリオでメモリー内のセッション・

キャッシュから無効にされたセッションは、そのセッション ID を要求間で保持できません。この動作を望まない場合は、アプリケーションのセッション管理を構成する前に、該当するすべてのアプリケーション・サーバー上で Web コンテナー・カスタム・プロパティを false に設定します。このカスタム・プロパティについて詳しくは、629 ページの『キャッシュ統合のトラブルシューティング』を参照してください。

## 手順

- アプリケーションのインストール時にセッション管理を構成するには、以下の手順を完了します。
  1. WebSphere Application Server の管理コンソールで、「アプリケーション」 > 「新規アプリケーション」 > 「新規エンタープライズ・アプリケーション」をクリックします。「詳細」パスを選択してアプリケーションを作成し、初期のウィザード・ステップを完了します。
  2. ウィザードの「eXtreme Scale セッション管理設定」ステップで、使用するデータ・グリッドを構成します。「リモート eXtreme Scale データ・グリッド」または「組み込み eXtreme Scale データ・グリッド」のいずれかを選択します。
    - 「リモート eXtreme Scale データ・グリッド」オプションの場合、セッション・データ・グリッドを管理するカタログ・サービス・ドメインを選択して、アクティブ・セッション・データ・グリッドのリストからデータ・グリッドを選択します。
    - 「組み込み eXtreme Scale データ・グリッド」オプションの場合、デフォルト ObjectGrid 構成を選択するか、ObjectGrid 構成ファイルの特定の場所を指定します。
  3. ウィザードのステップを完了して、アプリケーションのインストールを終了します。

wsadmin スクリプトを使用して、アプリケーションをインストールすることもできます。次の例では、**-SessionManagement** パラメーターにより、管理コンソールで作成するものと同じ構成を作成できます。

### リモート eXtreme Scale データ・グリッド構成の場合:

```
AdminApp.install('C:/A.ear', '[ -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*.dll=755#.*.so=755#.*.a=755#.*.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement cs0!:grid0]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdglapp.war,WEB-INF/web.xml
default_host] [MicroDG2App microdgg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]')
```

### デフォルト構成を使用した eXtreme Scale 組み込みシナリオの場合:

```
AdminApp.install('C:/A.ear', '[ -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*.dll=755#.*.so=755#.*.a=755#.*.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement !:!:default]] -MapWebModToVH [[MicroWebApp microwebapp.war,
```

```
WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdglapp.war,WEB-INF/web.xml
default_host] [MicroDG2App microdgd2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]')
```

### カスタム構成を使用した eXtreme Scale 組み込みシナリオの場合:

```
AdminApp.install('C:/A.ear', '[ -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*%.dll=755#.*%.so=755#.*%.a=755#.*%.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement ::::custom:::c:%XS%objectgrid.xml:::c:%XS%objectgriddeployment.xml]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdglapp.war,WEB-INF/web.xml
default_host] [MicroDG2App microdgd2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]')
```

- **WebSphere Application Server** の管理コンソールで既存のアプリケーション上にセッション管理を構成する場合は以下を行います。

1. WebSphere Application Server の管理コンソールで、「アプリケーション」 > 「アプリケーション・タイプ」 > 「WebSphere エンタープライズ・アプリケーション」 > 「application\_name」 > 「Web モジュール・プロパティ」 > 「セッション管理」 > 「eXtreme Scale セッション管理設定」をクリックします。
2. フィールドを更新して、データ・グリッドへのセッション・パーシスタンスを使用可能にします。

wsadmin スクリプトを使用して、アプリケーションを更新することもできます。次の例では、**-SessionManagement** パラメーターにより管理コンソールで作成するものと同じ構成を作成できます。

### リモート eXtreme Scale データ・グリッド構成の場合:

```
AdminApp.edit('DefaultApplication', '[-SessionManagement[[true
XSRemoteSessionManagement cs0:::grid0]]')
```

渡される **:::** 文字は、区切り文字として使用されます。渡される値は次のとおりです。

```
catalogServiceName:::gridName
```

### デフォルト構成を使用した eXtreme Scale 組み込みシナリオの場合:

```
AdminApp.edit('DefaultApplication', '[-SessionManagement[[true
XSEmbeddedSessionManagement :::::default]]')
```

渡される **:::** 文字は、区切り文字として使用されます。渡される値は次のとおりです。

```
catalogServiceName:::gridName:::default:::
absolutePath_to_objectGridXmlfile:::absolutePath_to_DeploymentXmlfile
```

### カスタム構成を使用した eXtreme Scale 組み込みシナリオの場合:

```
AdminApp.edit('DefaultApplication', '[-SessionManagement[[true
XSEmbeddedSessionManagement
:::custom:::c:%XS%objectgrid.xml:::c:%XS%objectgriddeployment.xml]]')
```

渡される **:::** 文字は、区切り文字として使用されます。渡される値は次のとおりです。

```
catalogServiceName:::gridName:::custom:::
absolutePath_to_objectGridXmlfile:::absolutePath_to_DeploymentXmlfile
```

変更を保存する場合、アプリケーションはアプライアンス上のセッション・パーシスタンスに構成済みのデータ・グリッド (data grid)を使用します。

- 既存のサーバー上でセッション管理を構成するには、以下を行います。
  1. WebSphere Application Server の管理コンソールで、「サーバー」 > 「サーバー・タイプ」 > 「WebSphere アプリケーション・サーバー」 > 「*server\_name*」 > 「セッション管理」 > 「eXtreme Scale セッション管理設定」をクリックします。
  2. フィールドを更新して、セッション・パーシスタンスを使用可能にします。また、以下の wsadmin ツール・コマンドで、既存のサーバーでのセッション管理を構成できます。

#### リモート eXtreme Scale データ・グリッド構成の場合:

```
AdminTask.configureServerSessionManagement(['-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSRemoteSessionManagement -XSRemoteSessionManagement
[-catalogService cs0 -csGridName grid0]'])
```

#### eXtreme Scale 組み込み構成の場合:

– デフォルト構成 (デフォルト XML ファイルを使用する場合):

```
AdminTask.configureServerSessionManagement(['-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSEmbeddedSessionManagement
-XSEmbeddedSessionManagement [-embeddedGridType default -objectGridXML -objectGridDeploymentXML ]'])
```

– カスタム構成 (カスタマイズした XML ファイルを使用する場合):

```
AdminTask.configureServerSessionManagement(['-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSEmbeddedSessionManagement
-XSEmbeddedSessionManagement
[-embeddedGridType custom -objectGridXML c:%XS%objectgrid.xml -objectGridDeploymentXML
c:%XS%objectgriddeployment.xml]'])
```

変更を保存すると、今後サーバーはサーバー上で稼働するすべてのアプリケーションでセッション・パーシスタンスの構成済み データ・グリッド (data grid)を使用します。

- HTTP セッション構成の他の局面を編集する場合は、`splicer.properties` ファイルを編集します。 `sessionFilterProps` カスタム・プロパティーを見つけて、`splicer.properties` ファイルのパスの場所を取得できます。サーバー・レベルでセッション・パーシスタンスを構成した場合、カスタム・プロパティーの名前は `com.ibm.websphere.xs.sessionFilterProps` です。アプリケーション・レベルでセッション・パーシスタンスを構成した場合、カスタム・プロパティーの名前は `<application_name>.com.ibm.websphere.xs.sessionFilterProps` です。これらのカスタム・プロパティーは、おそらく、次のいずれかの場所にあります。
  - WebSphere Application Server Network Deployment 環境の場合:  
`splicer.properties` ファイルはデプロイメント・マネージャー・プロファイル・パス上にあります。
  - スタンドアロン WebSphere Application Server 環境の場合: アプリケーション・サーバー上のカスタム・プロパティー

示されたファイルを開き、変更し、ノードを同期できます。それによって、更新されたプロパティー・ファイルが、構成内の他のノードに伝搬されます。セッションを適切に存続させるために、すべてのアプリケーション・サーバー・ノードは、`splicer.properties` ファイルが指定されたパスにあることを必要とします。

**重要:** URL 再書き込みを使用するセッションのパーシスタンスを有効にするには、`splicer.properties` ファイル内の `useURLEncoding` プロパティを `true` に設定します。

`splicer.properties` ファイルのプロパティに関して詳しくは、371 ページの『`splicer.properties` ファイル』を参照してください。

## タスクの結果

データ・グリッド (data grid)へのセッションをパーシストするように HTTP セッション・マネージャーが構成されました。セッションがタイムアウトになると、項目はデータ・グリッドから除去されます。WebSphere Application Server 管理コンソールでのセッション・タイムアウト値の更新について詳しくは、セッション管理設定を参照してください。

### *eXtreme Scale* セッション管理設定:

セッション・パーシスタンスのために WebSphere eXtreme Scale または WebSphere DataPower® XC10 Appliance を使用するように、WebSphere Application Server アプリケーションを構成できます。

これらの設定は、エンタープライズ・アプリケーション・インストール・ウィザード、あるいはアプリケーションまたはサーバー詳細ページで編集することができます。

- バージョン 6.1: 「アプリケーション」 > 「新規アプリケーションのインストール」
- バージョン 6.1: 「アプリケーション」 > 「エンタープライズ・アプリケーション」 > 「`application_name`」
- バージョン 6.1: 「サーバー」 > 「アプリケーション・サーバー」 > 「`server_name`」 > 「Web コンテナ設定」 > 「セッション管理」
- バージョン 7.0: 「アプリケーション」 > 「新規アプリケーション」 > 「新規エンタープライズ・アプリケーション」とクリックして、アプリケーション作成のための詳細パスを選択します。
- バージョン 7.0: 「アプリケーション」 > 「アプリケーション・タイプ」 > 「WebSphere エンタープライズ・アプリケーション」 > 「`application_name`」 > 「Web モジュール・プロパティ」 > 「セッション管理」 > 「セッション管理設定」
- バージョン 7.0: 「サーバー」 > 「サーバー・タイプ」 > 「WebSphere Application Server」 > 「`server_name`」 > 「コンテナ設定」 > 「セッション管理設定」

### セッション管理使用可能:

セッション管理を使用可能にし、セッション・パーシスタンスのために WebSphere eXtreme Scale の組み込みデータ・グリッドまたはリモート・データ・グリッド、あるいは WebSphere DataPower XC10 アプライアンスを使用できるようにします。

### セッション・パーシスタンスの管理:



セッション・パーシスタンスがどのように管理されるかを指定します。以下のオプションのいずれかを選択できます。

- WebSphere DataPower XC10 アプライアンス
- リモート eXtreme Scale データ・グリッド
- 組み込み eXtreme Scale データ・グリッド

構成する残りの設定は、選択したセッション・パーシスタンス・メカニズムによります。

#### **WebSphere DataPower XC10 アプライアンス固有の設定:**

以下の設定は、セッション・パーシスタンスのための WebSphere DataPower XC10 アプライアンスの構成に固有の設定です。

#### **WebSphere DataPower XC10 Appliance の IP またはホスト名:**

セッションのパーシストのために使用されるアプライアンスの IP またはホスト名を指定します。

#### **IBM WebSphere DataPower XC10 アプライアンス管理資格情報:**

DataPower XC10 アプライアンスのユーザー・インターフェースにログインするために使用するユーザー名とパスワードを指定します。「テスト接続...」をクリックして、アプライアンスへの接続をテストします。

#### **セッション・パーシスタンス設定:**

セッションがパーシストされるデータ・グリッドを指定します。以下のオプションのいずれかを選択できます。

- **IBM WebSphere DataPower XC10 アプライアンスの新規のデータ・グリッドにおけるセッション・パーシスト。**ここで「データ・グリッド名」を指定できます。
- **IBM WebSphere DataPower XC10 アプライアンスの既存のデータ・グリッドにおけるセッション・パーシスト。**ここで「既存のデータ・グリッド名」を入力または参照できます。

#### **リモート eXtreme Scale データ・グリッド構成:**

以下の設定は、セッション・パーシスタンスのためのリモート eXtreme Scale グリッドの構成に固有の設定です。

#### **リモート・セッション・データ・グリッドを管理するカタログ・サービス・ドメイン:**

セッションの管理に使用するカタログ・サービス・ドメインを指定します。

カタログ・サービス・ドメインが表示されていない場合や、新規カタログ・サービス・ドメインを作成したい場合は、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」の順にクリックします。

#### **セッション情報が保管されるリモート・データ・グリッド:**



カタログ・サービス・ドメイン内でセッション情報が保管されるデータ・グリッドの名前を指定します。カタログ・サービスを選択すると、アクティブ・リモート・グリッドのリストが取り込まれます。リモート・データ・グリッドは、eXtreme Scale 構成に事前に存在している必要があります。

#### 組み込み eXtreme Scale データ・グリッド構成:

以下の設定は、組み込み eXtreme Scale の構成に固有の設定です。組み込み eXtreme Scale シナリオでは、eXtreme Scale プロセスは、WebSphere Application Server プロセスによってホスティングされます。

#### eXtreme Scale 組み込みデータ・グリッド構成:

- デフォルトの ObjectGrid 構成を使用
- カスタム ObjectGrid 構成ファイルを指定

構成にコピーする `objectgrid.xml` ファイルの絶対パス

使用する構成の `objectgrid.xml` ファイルの絶対パスを指定します。

構成にコピーされる `objectgriddeployment.xml` ファイルの絶対パス

使用する構成の `objectgriddeployment.xml` ファイルの絶対パスを指定します。

## WebSphere eXtreme Scale を使用した SIP セッション管理

Session Initiation Protocol (SIP) セッション・レプリカ生成用のデータ・レプリカ生成サービス (DRS) の代わりに、WebSphere eXtreme Scale を、信頼できる SIP レプリカ生成メカニズムとして使用できます。

### SIP セッション管理の構成

WebSphere eXtreme Scale を SIP レプリカ生成メカニズムとして使用するには、`com.ibm.sip.ha.replicator.type` カスタム・プロパティを設定します。このカスタム・プロパティを追加する各サーバーごとに、管理コンソールで、「アプリケーション・サーバー」 > `my_application_server` > 「SIP コンテナ」 > 「カスタム・プロパティ」を選択します。「名前」には `com.ibm.sip.ha.replicator.type` と入力し、「値」には `OBJECTGRID` と入力します。

以下のプロパティを使用して、SIP セッションの保管に使用する ObjectGrid の振る舞いをカスタマイズします。このカスタム・プロパティを追加する各サーバーごとに、管理コンソールで、「アプリケーション・サーバー」 > `my_application_server` > 「SIP コンテナ」 > 「カスタム・プロパティ」をクリックします。「名前」および「値」を入力します。各サーバーは、機能のプロパティに設定されているものと同じプロパティを所有する必要があります。

表 27. ObjectGrid を使用した SIP セッション管理のためのカスタム・プロパティ

プロパティ	値	デフォルト
<code>com.ibm.sip.ha.replicator.type</code>	OBJECTGRID: SIP セッション・ストアとして ObjectGrid を使用	
<code>min.synchronous.replicas</code>	同期レプリカの最小数	0
<code>max.synchronous.replicas</code>	同期レプリカの最大数	0
<code>max.asynchronous.replicas</code>	非同期レプリカの最大数	1

表 27. ObjectGrid を使用した SIP セッション管理のためのカスタム・プロパティ (続き)

プロパティ	値	デフォルト
auto.replace.lost.shards	詳しくは、268 ページの『分散デプロイメントの構成』を参照してください。	true
development.mode	<ul style="list-style-type: none"> <li>• true - プライマリーと同じノード上でレプリカをアクティブにできる</li> <li>• false - レプリカはプライマリーと異なるノード上になければならない</li> </ul>	false

## WebSphere Portal での HTTP セッション・マネージャーの構成

WebSphere Portal の HTTP セッションをデータ・グリッドに保持できます。

### 始める前に

WebSphere eXtreme Scale と WebSphere Portal 環境が、次の要件を満たしている必要があります。

- WebSphere eXtreme Scale のインストール方法は、デプロイメント・シナリオによって異なります。データ・グリッドをホスティングするコンテナ・サーバーは、WebSphere Application Server セルの内部と外部のいずれでも実行できます。
  - コンテナ・サーバーを WebSphere Application Server セル内部で実行する場合 (組み込みシナリオ): WebSphere eXtreme Scale クライアントとサーバーの両方を WebSphere Application Server および WebSphere Portal ノードにインストールします。
  - コンテナ・サーバーを WebSphere Application Server セルの外部で実行する場合 (リモート・シナリオ): WebSphere eXtreme Scale クライアントを WebSphere Application Server および WebSphere Portal ノードにインストールします。

詳しくは、205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。

- WebSphere Portal バージョン 7 以降。
- カスタム・ポートレットは、WebSphere Portal 内で構成しなければなりません。現在、WebSphere Portal に付属の管理ポートレットは、データ・グリッドと統合できません。

### このタスクについて

WebSphere eXtreme Scale を WebSphere Portal 環境に導入することは、以下のシナリオでメリットがあります。

**重要:** 以下のシナリオでは、メリットについて紹介しますが、WebSphere eXtreme Scale を環境に導入することにより、WebSphere Portal 層でのプロセッサ使用量が増える場合もあります。

- セッション・パーシスタンスが必要な場合

例えば、WebSphere Portal Server の障害時でもカスタム・ポートレットのセッション・データを使用可能な状態で維持する必要がある場合は、HTTP セッション

を WebSphere eXtreme Scale データ・グリッドに保持できます。複数のサーバーにデータを複製しておく、データの可用性が高まります。

- **複数データ・センター・トポロジー**

ロケーションが物理的に異なる複数のデータ・センターにまたがるトポロジーの場合、WebSphere Portal HTTP セッションを WebSphere eXtreme Scale データ・グリッドに保持できます。セッションは、複数あるデータ・センター内のデータ・グリッドに複製されます。あるデータ・センターで障害が起こると、セッションは、データ・グリッドのデータのコピーを保持する別のデータ・センターにロールオーバーされます。

- **WebSphere Portal Server 層のメモリー所要量を低下させる場合**

セッション・データをコンテナ・サーバーのリモート層にオフロードすることで、セッションのサブセットが WebSphere Portal Server 上に存在することになります。このデータのオフロードにより、WebSphere Portal Server 層のメモリー所要量が低下します。

### 手順

1. wps WebSphere Portal アプリケーションと任意のカスタム・ポートレットを接合し、セッションをデータ・グリッドに格納できるようにします。

アプリケーションのデプロイ時に HTTP セッション管理を構成して、アプリケーションを接合するか、カスタム・プロパティを使用して、自動的にアプリケーションを接合することができます。アプリケーションの接合については、346 ページの『WebSphere Application Server での HTTP セッション・マネージャーの構成』を参照してください。

2. コンテナ・サーバーを WebSphere Application Server の外部に置くりモート・シナリオを使用する場合、リモート HTTP セッション・パーシスタンス・シナリオに必要なリモート eXtreme Scale コンテナを明示的に開始します。

XS/ObjectGrid/session/samples/objectGridStandAlone.xml 構成ファイルと objectGridDeploymentStandAlone.xml 構成ファイルを使用して、コンテナを開始します。例えば、以下のようなコマンドを使用できます。

```
startOgServer.sh xsContainer1 -catalogServiceEndpoints <host>:<port>
-objectgridFile XS/ObjectGrid/session/samples/objectGridStandAlone.xml -deploymentPolicyFile
XS/ObjectGrid/session/samples/objectGridDeploymentStandAlone.xml
```

コンテナ・サーバーの開始方法については、454 ページの『コンテナ・サーバーの始動』を参照してください。組み込みシナリオを使用する場合、コンテナ・サーバーの構成および開始方法については、313 ページの『WebSphere Application Server のコンテナ・サーバーの構成』を参照してください。

3. WebSphere Portal Server を再始動します。詳しくは、『WebSphere Portal Version 7: Starting and stopping servers, deployment managers, and node agents (WebSphere Portal バージョン 7: サーバー、デプロイメント・マネージャー、およびノード・エージェントの開始と停止)』を参照してください。

### タスクの結果

WebSphere Portal Server にアクセスでき、構成されているカスタム・ポートレットの HTTP セッション・データはデータ・グリッドに保持されます。

アプリケーション・セッション・データをホスティングしている全データ・グリッドに Web コンテナ・クライアントから到達できない場合、クライアントは、代わりに WebSphere Application Server の基本 Web コンテナをセッション管理に使用します。次のようなシナリオでは、データ・グリッドに到達できないことがあります。

- Web コンテナとリモート・コンテナ・サーバー間のネットワークの問題
- リモート・コンテナ・サーバーのプロセスが停止した場合

**sessionTableSize** パラメーターによって指定される、メモリー内に保持されるセッション参照の数は、セッションが基本 Web コンテナ内に保管されている場合、そのまま維持されます。セッション数が **sessionTableSize** の値を超えると、最長未使用時間を基にセッションが Web コンテナ・セッション・キャッシュで無効化されます。リモート・データ・グリッドが使用可能になると、Web コンテナ・キャッシュで無効化されたセッションは、リモート・データ・グリッドからデータを取得し、データを新規セッションにロードできます。リモート・データ・グリッド全体が使用不可なまま、セッションがセッション・キャッシュで無効化されると、ユーザーのセッション・データは失われます。このような問題があるため、負荷の下でシステムを実行する場合、実動リモート・データ・グリッド全体をシャットダウンすることはしないでください。

## 各種アプリケーション・サーバー用の HTTP セッション・マネージャーの構成

WebSphere eXtreme Scale には、Web コンテナのデフォルト・セッション・マネージャーをオーバーライドするセッション管理実装がバンドルされています。この実装は、セッション・レプリカ生成、高可用性、より優れたスケーラビリティと構成オプションを提供します。WebSphere eXtreme Scale セッション・レプリカ生成マネージャーおよび汎用組み込み ObjectGrid コンテナの開始を有効にします。

### このタスクについて

WebSphere Application Server Community Edition などの WebSphere Application Server を実行していない他のアプリケーション・サーバーで HTTP セッション・マネージャーを使用できます。データ・グリッドを使用するように他のアプリケーション・サーバーを構成するには、アプリケーションを接合して、WebSphere eXtreme Scale Java アーカイブ (JAR) ファイルをアプリケーションに取り込む必要があります。

### 手順

1. アプリケーションを接合することで、アプリケーションがセッション・マネージャーを使用できるようにします。セッション・マネージャーを使用するためには、適切なフィルター宣言をアプリケーションの Web デプロイメント記述子に追加する必要があります。さらに、セッション・マネージャー構成パラメーターが、デプロイメント記述子内のサブレット・コンテキスト初期化パラメーターという形式でセッション・マネージャーに渡されます。この情報は、以下に示す 3 とおりの方法でアプリケーションに導入することができます。

- **addObjectGridFilter** スクリプト:

eXtreme Scale とともに提供されるコマンド行スクリプトを使用して、フィルター宣言と構成によってアプリケーションをサブレット・コンテキスト初期

化パラメーターの形式で接合します。 `wxs_home/session/bin/addObjectGridFilter.sh|bat` スクリプトは、2 つのパラメーターを使用します。1 つは接合するエンタープライズ・アーカイブ (EAR) ファイルまたは Web アーカイブ (WAR) ファイルへの絶対パスで、もう 1 つは各種構成プロパティが含まれたスライサー・プロパティ・ファイルへの絶対パスです。このスクリプトの使用形式は以下の通りです。

#### Windows

```
addObjectGridFilter.bat <ear_or_war_file> <splicer_properties_file>
```

#### UNIX

```
addObjectGridFilter.sh <ear_or_war_file> <splicer_properties_file>
```

#### UNIX

UNIX 上のスタンドアロン・ディレクトリーにインストールされている eXtreme Scale の使用例:

- a. `cd wxs_home/session/bin`
- b. `addObjectGridFilter.sh /tmp/mySessionTest.ear wxs_home/session/samples/splicer.properties`

接合されるサブレット・フィルターは構成値のデフォルトを保持します。これらのデフォルト値は、2 番目の引数にあるプロパティ・ファイルで指定する構成オプションでオーバーライドできます。使用できるパラメーターのリストについては、368 ページの『サブレット・コンテキスト初期化パラメーター』を参照してください。

eXtreme Scale インストールとともに提供されるサンプルの `splicer.properties` ファイルを変更して使用することができます。また、各サブレットを拡張することによってセッション・マネージャーを挿入する、**addObjectGridServlets** スクリプトも使用できます。ただし、推奨スクリプトは **addObjectGridFilter** スクリプトです。

- Ant ビルド・スクリプト:

WebSphere eXtreme Scale には Apache Ant で使用できる `build.xml` ファイルが同梱されています。このファイルは WebSphere Application Server インストールの `was_root/bin` フォルダーに含まれています。`build.xml` ファイルを変更して、セッション・マネージャー構成プロパティを変更できます。構成プロパティは `splicer.properties` ファイル内のプロパティ名と同一です。`build.xml` ファイルの変更後に、`ant.sh`、`ws_ant.sh` (UNIX) または `ant.bat`、`ws_ant.bat` (Windows) を実行することで、Ant プロセスを呼び出します。

- 手動による Web 記述子の更新:

Web アプリケーションに同梱されている `web.xml` ファイルを編集して、フィルター宣言、そのサブレット・マッピング、およびサブレット・コンテキスト初期化パラメーターが組み込まれるようにします。この方法はエラーを起こしやすいため、使用しないようにしてください。

使用できるパラメーターのリストについては、368 ページの『サブレット・コンテキスト初期化パラメーター』を参照してください。



2. WebSphere eXtreme Scale セッション・レプリカ生成マネージャーの JAR ファイルをアプリケーションに取り込みます。ファイルは、アプリケーション・モジュールの WEB-INF/lib ディレクトリーまたはアプリケーション・サーバーのクラスパスに組み込むことができます。必要な JAR ファイルは、以下のよう  
に、使用しているコンテナのタイプによって異なります。
  - リモート・コンテナ・サーバー: ogclient.jar と sessionobjectgrid.jar
  - 組み込みコンテナ・サーバー: objectgrid.jar と sessionobjectgrid.jar
3. オプション: リモート・コンテナ・サーバーを使用する場合、コンテナ・サーバーを開始します。詳しくは、454 ページの『コンテナ・サーバーの始動』を参照してください。
4. アプリケーションをデプロイします。サーバーやクラスターに対して通常使用する手順に従ってアプリケーションをデプロイしてください。アプリケーションをデプロイした後、アプリケーションを始動することができます。
5. アプリケーションにアクセスします。これで、セッション・マネージャーおよび WebSphere eXtreme Scale と対話するアプリケーションにアクセスすることができます。

## 次のタスク

アプリケーションの装備時にセッション・マネージャーの構成属性の大多数を変更して、セッション・マネージャーを使用するようにすることができます。これらの属性には、レプリカ生成タイプ (同期または非同期) のバリエーション、メモリー内セッション・テーブルのサイズなどがあります。アプリケーションの装備時に変更できる属性を別にすれば、アプリケーションのデプロイメント後に変更できるその他の構成属性は、WebSphere eXtreme Scale サーバー・クラスター・トポロジーと、それらのクラスターのクライアント (セッション・マネージャー) がそれらのクラスターに接続する方法に関係する属性のみです。

リモート・シナリオの動作: アプリケーション・セッション・データをホスティングしている全データ・グリッドに Web コンテナ・クライアントから到達できない場合、クライアントは、代わりにアプリケーション・サーバーの基本 Web コンテナをセッション管理に使用します。次のようなシナリオでは、データ・グリッドに到達できないことがあります。

- Web コンテナとリモート・コンテナ・サーバー間のネットワークの問題
- リモート・コンテナ・サーバーのプロセスが停止した場合

**sessionTableSize** パラメーターによって指定される、メモリー内に保持されるセッション参照の数は、セッションが基本 Web コンテナ内に保管されている場合、そのまま維持されます。セッション数が **sessionTableSize** の値を超えると、最長未使用時間を基にセッションが Web コンテナ・セッション・キャッシュで無効化されます。リモート・データ・グリッドが使用可能になると、Web コンテナ・キャッシュで無効化されたセッションは、リモート・データ・グリッドからデータを取得し、データを新規セッションにロードできます。リモート・データ・グリッド全体が使用不可なまま、セッションがセッション・キャッシュで無効化されると、ユーザー・セッション・データは失われます。このような問題があるため、負荷の下でシステムを実行する場合、実動リモート・データ・グリッド全体をシャットダウンすることはしないでください。



## HTTP セッション・マネージャー構成のための XML ファイル

HTTP セッション・データを保管するコンテナ・サーバーを始動するときは、デフォルトの XML ファイルを使用することもできるし、カスタマイズされた XML ファイルを指定することもできます。これらのファイルは、特定の ObjectGrid 名、レプリカ数などを作成します。

### サンプル・ファイルの場所

これらの XML ファイルは、スタンドアロンのインストール済み環境の場合は `wxs_install_root/ObjectGrid/session/samples` の中に、WebSphere Application Server セルにインストールされた WebSphere eXtreme Scale の場合は `was_root/optionalLibraries/ObjectGrid/session/samples` の中にパッケージされています。

### 組み込み XML パッケージ

組み込みシナリオを構成する場合、コンテナ・サーバーは Web コンテナ層で始動します。デフォルトで提供される、`objectGrid.xml` ファイルと `objectGridDeployment.xml` ファイルを使用します。これらのファイルを更新して HTTP セッション・マネージャーの振る舞いをカスタマイズすることができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd" xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="session" txTimeout="30">
      <bean id="ObjectGridEventListener" className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
      <backingMap name="objectgridSessionMetadata" pluginCollectionRef="objectgridSessionMetadata" readOnly="false"
        lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="NO_COPY"/>
      <backingMap name="objectgridSessionAttribute.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
        ttlEvictorType="NONE" copyMode="NO_COPY"/>
      <backingMap name="objectgridSessionTTL.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
        ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="NO_COPY"/>
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="objectgridSessionMetadata">
      <bean id="MapEventListener" className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

図 36. `objectGrid.xml` ファイル

#### 変更可能な値:

##### ObjectGrid name 属性

値は、他の構成ファイル内の次の値と一致していなければなりません。

- Web アプリケーションの接続に使用される `splicer.properties` ファイル内の `objectGridName` プロパティ。
- `objectGridDeployment.xml` ファイル内の `objectgridName` 属性。

複数のアプリケーションを持っていて、セッション・データを異なるデータ・グリッドに保管したい場合は、それらのアプリケーションが異なる ObjectGrid name 属性値を持つ必要があります。

##### ObjectGrid txTimeout 属性

この値は、トランザクションがオープン状態を持続できる秒数を決定します。この時間を過ぎると、コンテナ・サーバーがトランザクションのタイムアウトをトリガーします。デフォルトは 30 秒で、環境に応じて変更でき

ます。HTTP セッション・パーシスタンスが、**replicationInterval** サブレット・コンテキスト初期化パラメーター値がゼロより大きい値に設定されて構成されている場合、トランザクションはスレッド上でバッチ処理されます。**replicationInterval** プロパティが 0 に設定されている場合、トランザクションは、通常、Web アプリケーションが有効な **HttpSession** オブジェクトを取得したときに開始します。トランザクションは、Web アプリケーション要求の終了時にコミットされます。使用している環境に 30 秒より長くかかる要求がある場合は、それに合わせてこの値を設定してください。

#### 変更不可の値:

##### **ObjectGridEventListener**

**ObjectGridEventListener** 行は変更不可で、内部で使用されます。

##### **objectgridSessionMetadata**

**objectgridSessionMetadata** 行は HTTP セッション・メタデータが保管されるマップを参照します。このマップには、データ・グリッドに保管されている HTTP セッションごとに 1 つのエントリーがあります。

##### **objectgridSessionTTL.\***

この値は変更できません。また、将来使用されるものです。

##### **objectgridSessionAttribute.\***

**objectgridSessionAttribute.\*** テキストは動的マップを定義します。この値は、**splicer.properties** ファイル内で **fragmentedSession** パラメーターが **true** に設定されている場合に、HTTP セッションの属性が保管されるマップの作成に使用されます。この動的マップは、**objectgridSessionAttribute** と呼ばれます。このテンプレートに基づいて、**objectgridSessionAttributeEvicted** と呼ばれる別のマップが作成されます。このマップには、タイムアウトになったが Web コンテナが無効にしなかったセッションが保管されます。

**MapEventListener** 行は内部用で、変更はできません。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="session">
    <mapSet name="sessionMapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="0"
maxAsyncReplicas="1" developmentMode="false" placementStrategy="PER_CONTAINER">
      <map ref="objectgridSessionMetadata"/>
      <map ref="objectgridSessionAttribute.*"/>
      <map ref="objectgridSessionTTL.*"/>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

図 37. *objectGridDeployment.xml* ファイル

#### 変更可能な値:

##### **ObjectGrid name** 属性

値は、他の構成ファイル内の次の値と一致していなければなりません。

- Web アプリケーションの接続に使用される **splicer.properties** ファイル内の **objectGridName** プロパティ。

- objectGrid.xml ファイル内の ObjectGrid **name** 属性。

複数のアプリケーションを持っていて、セッション・データを異なるデータ・グリッドに保管したい場合は、それらのアプリケーションが異なる ObjectGrid name 属性値を持つ必要があります。

### **mapSet** エlement属性

placementStrategy 属性を除くすべての mapSet プロパティは変更することができます。

**Name** 任意の値に更新できます。

### **numberOfPartitions**

Web アプリケーションをホスティングしている各サーバーで開始されるプライマリー区画の数を指定します。区画を追加すると、フェイルオーバー時にデータがより散らばった状態になります。デフォルト値は 5 区画であり、これはほとんどのアプリケーションで問題のない値です。

### **minSyncReplicas**、**maxSyncReplicas**、および **maxAsyncReplicas**

HTTP セッション・データを格納するレプリカの数とタイプを指定します。デフォルトは 1 つの非同期レプリカであり、これはほとんどのアプリケーションで問題のない値です。同期レプリカ生成は要求パス中に行われますが、これによってご使用の Web アプリケーションの応答時間が長くなる場合があります。

### **developmentMode**

区画のレプリカ断片をそのプライマリー断片と同じノードに配置することができるかどうかを、eXtreme Scale 配置サービスに通知します。開発環境ではこの値を TRUE に設定できますが、ノード障害がセッション・データの損失を引き起こす場合があるため、実稼働環境ではこの機能を使用不可に設定してください。

### **placementStrategy**

この属性の値は変更しないでください。

ファイルの残りの部分は objectGrid.xml ファイル内と同じマップ名を参照します。これらの名前は変更できません。

### **変更不可の値:**

- mapSet エlementの placementStrategy 属性

### **リモート XML パッケージ**

コンテナがスタンドアロン・プロセスとして実行されるリモート・モードを使用しているときは、これらのプロセスを開始するのに objectGridStandAlone.xml ファイルおよび objectGridDeploymentStandAlone.xml ファイルを使用する必要があります。これらのファイルを更新することで構成を変更することができます。

```

<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
<objectGrid name="session" txTimeout="30">
<bean id="ObjectGridEventListener" className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
<backingMap name="objectgridSessionMetadata" pluginCollectionRef="objectgridSessionMetadata"
readOnly="false" lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600"
copyMode="COPY_TO_BYTES"/>
<backingMap name="objectgridSessionAttribute.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
ttlEvictorType="NONE" copyMode="COPY_TO_BYTES"/>
<backingMap name="objectgridSessionTTL.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="COPY_TO_BYTES"/>
</objectGrid>
</objectGrids>
<backingMapPluginCollections>
<backingMapPluginCollection id="objectgridSessionMetadata">
<bean id="MapEventListener" className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
</backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>

```

図 38. *objectGridStandAlone.xml* ファイル

### 変更可能な値:

#### ObjectGrid name 属性

値は、他の構成ファイル内の次の値と一致していなければなりません。

- Web アプリケーションの接続に使用される `splicer.properties` ファイル内の **objectGridName** プロパティ。
- `objectGridStandAlone.xml` ファイル内の **objectgridName** 属性。

複数のアプリケーションを持っていて、セッション・データを異なるデータ・グリッドに保管したい場合は、それらのアプリケーションが異なる ObjectGrid name 属性値を持つ必要があります。

#### ObjectGrid txTimeout 属性

この値は、トランザクションがオープン状態を持続できる秒数を決定します。この時間を過ぎると、コンテナ・サーバーがトランザクションのタイムアウトをトリガーします。デフォルトは 30 秒で、環境に応じて変更できます。HTTP セッション・パーシスタンスが、**replicationInterval** サーブレット・コンテキスト初期化パラメーター値がゼロより大きい値に設定されて構成されている場合、トランザクションはスレッド上でバッチ処理されます。**replicationInterval** プロパティが 0 に設定されている場合、トランザクションは、通常、Web アプリケーションが有効な HttpSession オブジェクトを取得したときに開始します。トランザクションは、Web アプリケーション要求の終了時にコミットされます。使用している環境に 30 秒より長くかかる要求がある場合は、それに合わせてこの値を設定してください。

### 変更不可の値:

#### ObjectGridEventListener

ObjectGridEventListener 行は変更不可で、内部で使用されます。

#### objectgridSessionMetadata

objectgridSessionMetadata 行は HTTP セッション・メタデータが保管されるマップを参照します。このマップには、データ・グリッドに保管されている HTTP セッションごとに 1 つのエントリーがあります。

#### objectgridSessionTTL.\*

この値は変更できません。また、将来使用されるものです。

### **objectgridSessionAttribute.\***

objectgridSessionAttribute.\* テキストは動的マップを定義します。この値は、splicer.properties ファイル内で **fragmentedSession** パラメーターが true に設定されている場合に、HTTP セッションの属性が保管されるマップの作成に使用されます。この動的マップは、objectgridSessionAttribute と呼ばれます。このテンプレートに基づいて、objectgridSessionAttributeEvicted と呼ばれる別のマップが作成されます。このマップには、タイムアウトになったが Web コンテナが無効にできなかったセッションが保管されます。

**MetadataMapListener** 行は内部用で、変更はできません。

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="session">
    <mapSet name="sessionMapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="0"
      maxAsyncReplicas="1" developmentMode="false" placementStrategy="PER_CONTAINER">
      <map ref="objectgridSessionMetadata"/>
      <map ref="objectgridSessionAttribute.*"/>
      <map ref="objectgridSessionTTL.*"/>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

図 39. objectGridDeploymentStandAlone.xml ファイル

### 変更可能な値:

#### **objectgridName** 属性

値は、他の構成ファイル内の次の値と一致していなければなりません。

- Web アプリケーションの接続に使用される splicer.properties ファイル内の **objectGridName** プロパティ。
- objectGrid.xml ファイル内の **ObjectGrid name** 属性。

複数のアプリケーションを持っていて、セッション・データを異なるデータ・グリッドに保管したい場合は、それらのアプリケーションが異なる ObjectGrid name 属性値を持つ必要があります。

#### **mapSet** エレメント属性

placementStrategy 属性を除くすべての mapSet プロパティは変更することができます。

**Name** 任意の値に更新できます。

#### **numberOfPartitions**

Web アプリケーションをホスティングしている各サーバーで開始されるプライマリー区画の数を指定します。区画を追加すると、フェイルオーバー時にデータがより散らばった状態になります。デフォルト値は 5 区画であり、これはほとんどのアプリケーションで問題のない値です。

#### **minSyncReplicas、maxSyncReplicas、および maxAsyncReplicas**

HTTP セッション・データを格納するレプリカの数とタイプを指定

します。デフォルトは 1 つの非同期レプリカであり、これはほとんどのアプリケーションで問題のない値です。同期レプリカ生成は要求パス中に行われますが、これによってご使用の Web アプリケーションの応答時間が長くなる場合があります。

#### **developmentMode**

区画のレプリカ断片をそのプライマリー断片と同じノードに配置することができるかどうかを、eXtreme Scale 配置サービスに通知します。開発環境ではこの値を TRUE に設定できますが、ノード障害がセッション・データの損失を引き起こす場合があるため、実稼働環境ではこの機能を使用不可に設定してください。

#### **placementStrategy**

この属性の値は変更しないでください。

ファイルの残りの部分は objectGrid.xml ファイル内と同じマップ名を参照します。これらの名前は変更できません。

#### **変更不可の値:**

- mapSet エLEMENTの placementStrategy 属性

### **サブレット・コンテキスト初期化パラメーター**

以下に示すサブレット・コンテキスト初期化パラメーターのリストは、選択した接続メソッドに必要なスプライサー・プロパティ・ファイルに指定できるものです。

#### **パラメーター**

##### **objectGridType**

REMOTE または EMBEDDED のいずれかのストリング値。デフォルトは REMOTE です。

このパラメーターが REMOTE に設定されている場合、セッション・データは Web アプリケーションが実行されているサーバーの外に保管されます。

このパラメーターが EMBEDDED に設定されている場合は、Web アプリケーションが実行されているアプリケーション・サーバー・プロセス内で組み込みの eXtreme Scale コンテナが開始されます。

##### **objectGridName**

特定の Web アプリケーションで使用する ObjectGrid インスタンスの名前を定義するストリング値。デフォルトの名前は「session」です。

eXtreme Scale コンテナ・サーバーの始動に使用する ObjectGrid XML ファイルとデプロイメント XML ファイルの両方で、このプロパティは objectGridName を反映する必要があります。

##### **catalogHostPort**

カタログ・サーバーに接続して、クライアント・サイドの ObjectGrid インスタンスを取得できます。値の形式は、host:port<,host:port> でなければなりません。host は、カタログ・サーバーが実行されているリスナー・ホストです。port は、そのカタログ・サーバー・プロセスのリスナー・ポートです。このリストは任意の長さにすることができ、ブートストラッピングにのみ使用されま



す。最初の実行可能なアドレスが使用されます。このパラメーターは、**catalog.services.cluster** プロパティーが構成されている場合は、WebSphere Application Server 内でオプションです。

### **replicationInterval**

更新されたセッションを ObjectGrid に書き込む時間間隔を秒数で定義する整数値。デフォルトは 10 秒です。0 から 60 までの値を設定できます。0 の場合は、更新されたセッションを ObjectGrid に書き込むのは、各要求のサーブレット・サービス・メソッド呼び出しの最後ということになります。

**replicationInterval** 値が高いほどパフォーマンスは向上します。これは、データ・グリッドに書き込まれるアップデートの数が少ないためです。ただし、値が高いとそれだけ構成のフォールト・トレラント性が低くなります。

この設定は、objectGridType が REMOTE に設定されている場合のみ適用されます。

### **sessionTableSize**

メモリー内に保持するセッション参照数を定義する整数値。デフォルトは 1000 です。

EMBEDDED トポロジーは既に Web コンテナと同じ層にセッション・データを持っているため、この設定は REMOTE トポロジーのみに関するものです。

セッションは、最長未使用時間 (LRU) ロジックに基づいて、メモリー内のテーブルから除去されます。メモリー内のテーブルから除去されたセッションは Web コンテナから無効化されます。しかし、データ自体はグリッドから削除されないため、そのセッションに対する後続の要求は引き続きデータを検索することができます。この値は、Web コンテナの最大スレッド・プール値よりも高く設定されなければなりません。そうすると、セッション・キャッシュでの競合を削減できます。

### **fragmentedSession**

true または false のいずれかのストリング値。デフォルト値は true です。この設定を使用して、製品がセッション・データをエンタリー全体として保管するか、それぞれの属性を個別に保管するかを制御します。

Web アプリケーションのセッションが持っている属性の数が多き場合や属性のサイズが大きき場合は、fragmentedSession パラメーターを true に設定してください。すべての属性はデータ・グリッド内の同じキーに保管されるため、セッションが持っている属性の数が少ない場合は fragmentedSession を false に設定してください。

以前のフィルター・ベースの実装環境では、このプロパティーは

「persistenceMechanism」と呼ばれており、設定可能な値は ObjectGridStore (フラグメント化されている場合) および ObjectGridAtomicSessionStore (フラグメント化されていない場合) でした。

### **securityEnabled**

true または false のいずれかのストリング値。デフォルト値は false です。この設定により、eXtreme Scale クライアント・セキュリティーを使用可能にす

ることができます。この設定は、eXtreme Scale サーバー・プロパティ・ファイルの **securityEnabled** 設定と一致していなければなりません。これらの設定が一致しないと、例外が発生します。

#### **credentialGeneratorClass**

com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator インターフェースを実装するクラスの名前。このクラスを使用して、クライアントの資格情報が取得されます。

#### **credentialGeneratorProps**

CredentialGenerator 実装クラスのプロパティ。このプロパティが、setProperties(String) メソッドを使用してオブジェクトに設定されます。credentialGeneratorProps 値は、**credentialGeneratorClass** プロパティの値が非ヌルの場合にのみ使用されます。

#### **objectGridXML**

objectgrid.xml ファイルが置かれているロケーション。  
objectGridType=EMBEDDED および **objectGridXML** プロパティが指定されていない場合、eXtreme Scale ライブラリーにパッケージされた組み込み XML ファイルが自動的にロードされます。

#### **objectGridDeploymentXML**

objectGrid デプロイメント・ポリシーの XML ファイルが置かれているロケーションを指定します。objectGridType=EMBEDDED および **objectGridDeploymentXML** プロパティが指定されていない場合、eXtreme Scale ライブラリーにパッケージされた組み込み XML ファイルが自動的にロードされます。

#### **traceSpec**

ストリング値として設定される、IBM WebSphere のトレース仕様を指定します。この設定は、WebSphere Application Server以外のアプリケーション・サーバーに使用してください。

#### **traceFile**

トレース・ファイルの場所をストリング値として指定します。この設定は、WebSphere Application Server以外のアプリケーション・サーバーに使用してください。

#### **cookieDomain**

ホストをまたいだセッションへのアクセスが必要かどうかを指定します。値を、ホスト間に共通のドメインの名前に設定してください。

#### **reuseSessionID**

基礎になっている Web コンテナが異なるホストへの要求でセッション ID を再利用する場合は、true に設定します。デフォルト値は false です。このプロパティの値は、Web コンテナの中での値と同じでなければなりません。WebSphere Application Server を使用していて、管理コンソールまたは wsadmin ツール・スクリプトを使用して eXtreme Scale HTTP セッション・パーシスタンスを構成している場合、デフォルトで、Web コンテナ・カスタム・プロパティ HttpSessionIdReuse=true が追加されます。**reuseSessionID** も true に設定されます。セッション ID の再利用を望まない場合は、eXtreme Scale セッ

セッション・パーシスタンスを構成する前に、Web コンテナ・カスタム・プロパティに `HttpSessionIdReuse=false` カスタム・プロパティを設定します。

### shareSessionsAcrossWebApps

セッションが Web アプリケーション間で共有されるかどうかを指定します。true または false のいずれかのストリング値として指定されます。デフォルトは false です。サーブレット仕様では、HTTP セッションを Web アプリケーション間で共有できないとされています。この共有を可能にするため、サーブレット仕様の拡張が提供されます。

### useURLEncoding

URL 再書き込みを使用可能にする場合は、true に設定します。デフォルト値は false です。これは、セッション・データの保管に Cookie が使用されることを示します。このパラメーターの値は、セッション管理の Web コンテナ設定と同じでなければなりません。

## splicer.properties ファイル

splicer.properties ファイルには、サーブレット・フィルター・ベースのセッション・マネージャーを構成するための、すべての構成オプションが含まれます。

## サンプル・スプライサー・プロパティ

このファイルに説明されている追加プロパティを使用する場合は、有効にするプロパティの行のコメントを外してください。

```
# サーブレット・フィルター・ベース ObjectGrid
# セッション・マネージャーが使用するように構成できる、
# すべての構成オプションが含まれたプロパティ・ファイル。
#
# このプロパティ・ファイルで、これらの構成設定に割り当てるすべての
# デフォルト値を保持できます。また、このプロパティ・ファイルを
# filtersplicer ANT タスクと組み合わせて使用する場合には、ANT タスク・
# プロパティを使用して、個々の設定をオーバーライドできます。

# ストリング値「REMOTE」または「EMBEDDED」。デフォルトは REMOTE です。
# If it is set to "REMOTE", the session data will be stored outside of
# the server on which the web application is running. If it is set to
# "EMBEDDED", an embedded WebSphere eXtreme Scale container will start
# in the application server process on which the web application is running.

objectGridType = REMOTE

# A string value that defines the name of the ObjectGrid
# instance used for a particular web application. The default name
# is session. This property must reflect the objectGridName in both
# the objectgrid xml and deployment xml files used to start the eXtreme
# Scale containers.

objectGridName = session

# カタログ・サーバーに接続して、クライアント・サイドの ObjectGrid
# インスタンスを取得できます。The value needs to be of the
# form "host:port<,host:port>", where the host is the listener host
# on which the catalog server is running, and the port is the listener
# port for that catalog server process.
# このリストは任意の長さにする事ができ、ブートストラッピングにのみ使用
# されます。最初の実行可能なアドレスが使用されます。It is optional inside WebSphere
# if the catalog.services.cluster property is configured.

# catalogHostPort = host:port<,host:port>
```

```

# An integer value (in seconds) that defines the time in seconds between
# writing of updated sessions to ObjectGrid. The default is 10. This property
# is only used when objectGridType is set to REMOTE. Possible values are
# from 0 to 60. 0 means updated sessions are written to the ObjectGrid
# at the end of servlet service method call for each request.

replicationInterval = 10

# メモリー内に保持するセッション参照数を定義する整数値。
# デフォルトは 1000 です。This property is only used when
# objectGridType is set to REMOTE. When the number of sessions stored
# in memory in the web container exceeds this value, the least recently
# accessed session is invalidated from the web container. If a request
# comes in for that session after it's been invalidated, a new session
# will be created (with a new session ID if reuseSessionId=false),
# populated with the invalidated session's attributes. This value should
# always be set to be higher than the maximum size of the web container
# thread pool to avoid contention on this session cache.

sessionTableSize = 1000

# A string value of either "true" or "false", default is "true".
# It is to control whether we store session data as a whole entry
# or store each attribute separately.
# This property was referred to as persistenceMechanism in the
# previous filter-based implementation, with the possible values
# of ObjectGridStore (fragmented) and ObjectGridAtomicSessionStore
# (not fragmented).

fragmentedSession = true

# A string value of either "true" or "false", default is "false".
# Enables eXtreme Scale client security. This setting needs to match
# the securityEnabled setting in the eXtreme Scale server properties
# file. これらの設定が一致しないと、例外が発生します。

securityEnabled = false

# Specifies the client credential authentication support.
# The possible values are:
# Never - The client does not support credential authentication.
# Supported* - The client supports the credential authentication if and only if the server
# supports too.
# Required - The client requires the credential authentication.
# The default value is Supported.

# credentialAuthentication =

# Specifies the retry count for authentication if the credential
# is expired. If the value is set to 0, there will not be
# any authentication retry.

# authenticationRetryCount =

# Specifies the name of the class that implements the
# com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator
# interface. This class is used to get credentials for clients.

# credentialGeneratorClass =

# Specifies the properties for the CredentialGenerator implementation
# class. The properties are set to the object with the setProperties(String)
# method. The credentialGeneratorProps value is used only if the value of the
# credentialGeneratorClass property is not null.

# credentialGeneratorProps =

```

```

# The file location of the objectgrid xml file.
# The built-in xml file packaged in the eXtreme Scale library
# will automatically be loaded if this property
# is not specified and if objectGridType=EMBEDDED

# objectGridXML =

# The file location of the objectGrid deployment policy xml file.
# The built-in xml file packaged in the eXtreme Scale library
# will automatically be loaded if this property
# is not specified and if objectGridType=EMBEDDED

# objectGridDeploymentXML =

# IBM WebSphere トレース仕様を示すストリング。
# useful for all other application servers besides WebSphere.

# traceSpec =

# トレース・ファイルの場所を示すストリング。
# useful for all other application servers besides WebSphere.

# traceFile=

# This property should be set if you require sessions to be
# accessible across hosts. The value will be the name of the
# common domain between the hosts.

# cookieDomain=

# Set to true if the underlying web container will reuse
# session ID's across requests to different hosts. Default is
# false. The value of this should be the same as what is set
# in the web container.

# reuseSessionId=

# ストリング値「true」または「false」。The default is
# "false". Per the servlet specification, HTTP Sessions cannot
# be shared across web applications. An extension to the servlet
# specification is provided to allow this sharing.

# shareSessionsAcrossWebApps = false

# Set to true if you want to enable urlRewriting. Default is
# false, which means cookies will be used to store data. The
# value of this should reflect what is set in the web container
# settings for session management.

# useURLEncoding = false

```

## WebSphere eXtreme Scale の動的キャッシュ・プロバイダーの構成

eXtreme Scale の動的キャッシュ・プロバイダーのインストールと構成は、要件と、セットアップした環境によって異なります。

### 始める前に

- 動的キャッシュ・プロバイダーを使用するには、WebSphere Application Server ノードの各デプロイメント (デプロイメント・マネージャー・ノードを含む) 上に WebSphere eXtreme Scale をインストールしておく必要があります。詳しくは、205 ページの

205 ページの『WebSphere Application Server での WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール』を参照してください。

- カタログ・サービス・ドメイン内のカタログ・サーバーの Secure Sockets Layer (SSL) が使用可能な場合、または SSL がサポートされるカタログ・サービス・ドメインで SSL を使用する必要がある場合は、WebSphere Application Server 管理コンソールでグローバル・セキュリティを有効にする必要があります。サーバー・プロパティ・ファイルで、transportType 属性を SSL-Required に設定して、カタログ・サーバーに SSL を要求します。グローバル・セキュリティの構成に関して詳しくは、グローバル・セキュリティの設定を参照してください。

## このタスクについて

IBM WebSphere Commerce での eXtreme Scale 動的キャッシュ・プロバイダーの使用については、IBM WebSphere Commerce 資料の以下のトピックを参照してください。

- Enabling the dynamic cache service and servlet caching
- Enabling WebSphere Commerce data cache

定義したオブジェクト・キャッシュまたはサーブレット・キャッシュ・インスタンスにキャッシュを特別に送信しない場合には、動的キャッシュ API 呼び出しは、恐らく baseCache によって処理されます。eXtreme Scale 動的キャッシュ・プロバイダーを JSP、Web サービス、あるいは、コマンド・キャッシングのために使用する場合、eXtreme Scale 動的キャッシュ・プロバイダーを使用するように baseCache インスタンスを設定する必要があります。baseCache インスタンスの構成には同じ構成プロパティが使用されます。これらの構成プロパティは、Java 仮想マシン (JVM) のカスタム・プロパティとして設定する必要がありますので注意してください。この注意は、サーブレット・キャッシングを除き、このセクションで説明するすべてのキャッシュ構成プロパティに該当します。サーブレット・キャッシングに動的キャッシュ・プロバイダーを使用して eXtreme Scale を使用する場合には、必ず、カスタム・プロパティではなくシステム・プロパティで使用可能化を構成してください。

## 手順

1. eXtreme Scale 動的キャッシュ・プロバイダーを使用可能にします。

- **WebSphere Application Server バージョン 7.0 以降:**

管理コンソールで、eXtreme Scale 動的キャッシュ・プロバイダーを使用するよう動的キャッシュ・サービスを構成できます。eXtreme Scale をインストールすると、eXtreme Scale 動的キャッシュ・プロバイダーが、管理コンソールの「キャッシュ・プロバイダー」オプションとして即時に使用可能になります。詳しくは、WebSphere Application Server バージョン 7.0 インフォメーション・センター: キャッシュ・サービス・プロバイダーの選択を参照してください。

- **WebSphere Application Server バージョン 6.1:**



カスタム・プロパティを使用して、eXtreme Scale 動的キャッシュ・プロバイダーを使用するよう動的キャッシュ・サービスを構成します。これらのカスタム・プロパティは、WebSphere Application Server バージョン 7.0 以降でも使用できます。キャッシュ・インスタンスにカスタム・プロパティを作成するには、「リソース」 > 「キャッシュ・インスタンス」 >

「cache\_instance\_type」 > 「cache\_instance\_name」 > 「カスタム・プロパティ」 > 「新規」をクリックします。ベース・キャッシュ・インスタンスを使用する場合は、JVM にカスタム・プロパティを作成してください。

#### **com.ibm.ws.cache.CacheConfig.cacheProviderName**

eXtreme Scale 動的キャッシュ・プロバイダーを使用するには、値を `com.ibm.ws.objectgrid.dynacache.CacheProviderImpl` に設定します。このカスタム・プロパティは、動的キャッシュ・インスタンスまたはベース・キャッシュ・インスタンスに対して作成できます。このカスタム・プロパティをベース・キャッシュ・インスタンスに設定するよう選択すると、サーバー上のその他すべてのキャッシュ・インスタンスが、デフォルトで eXtreme Scale プロバイダーを使用するようになります。baseCache に設定されるすべての eXtreme Scale 動的キャッシュ・プロバイダー構成プロパティは、eXtreme Scale でサポートされるすべてのキャッシュ・インスタンスのデフォルトの構成プロパティになります。ベース・キャッシュ・インスタンスをオーバーライドして、特定の動的キャッシュ・インスタンスではデフォルトの動的キャッシュ・プロバイダーを使用するには、その動的キャッシュ・インスタンスに

`com.ibm.ws.cache.CacheConfig.cacheProviderName` カスタム・プロパティを作成し、値を `default` に設定してください。

2. オプション: 複製キャッシュ・インスタンスを使用する場合、キャッシュのレプリカ生成設定を構成します。

eXtreme Scale 動的キャッシュ・プロバイダーでは、ローカル・キャッシュ・インスタンスまたは複製キャッシュ・インスタンスを使用できます。ローカル・キャッシュ・インスタンスのみを使用する場合、このステップはスキップできません。

次のいずれかの方式を使用して、複製キャッシュを構成します。

- 管理コンソールでキャッシュ・レプリカ生成を有効にする。 WebSphere Application Server バージョン 7.0 では、いつでもキャッシュ・レプリカ生成を有効にできます。 WebSphere Application Server バージョン 6.1 では、DRS レプリカ生成ドメインを作成する必要があります。
- `com.ibm.ws.cache.CacheConfig.enableCacheReplication` カスタム・プロパティを使用してキャッシュ・レプリカ生成を有効にして、DRS レプリカ生成ドメインが割り当てられていない場合でも、それが複製されたキャッシュであることを報告するようキャッシュに強制する。このカスタム・プロパティの値を `true` に設定します。このカスタム・プロパティは、オブジェクト・キャッシュまたはサーブレット・キャッシュを使用する場合はキャッシュ・インスタンスに設定し、baseCache インスタンスを使用する場合は JVM に設定してください。

- オプション: eXtreme Scale を JSP フラグメント・キャッシュとして使用する場  
合、`com.ibm.ws.cache.CacheConfig.disableTemplateInvalidation` カスタム・プロパテ  
ィーを `true` に設定して、JSP 再ロード時のテンプレート・ベースの無効化を使  
用不可に設定します。
- 動的キャッシュ・サービスのトポロジを構成します。

eXtreme Scale 動的キャッシュ・プロバイダーで唯一必須の構成パラメーター  
が、キャッシュ・トポロジです。このカスタム・プロパティは、キャッシ  
ュ・インスタンスに設定するか、`baseCache` インスタンスを使用する場合は動的  
キャッシュ・サービスに設定してください。カスタム・プロパティの名前を  
`com.ibm.websphere.xs.dynacache.topology` と入力します。

このプロパティに許可される 3 つの値は次のとおりです。許可される値のい  
ずれかを使用する必要があります。

- `embedded`
- `embedded_partitioned`
- `remote`

組み込みトポロジまたは組み込み区画化トポロジを使用する場合は、  
`com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent` カスタム・プ  
ロパティを `true` に設定して、シリアライゼーションのコストをいくらかでも  
節約することをお勧めします。このカスタム・プロパティは、キャッシュ・イ  
ンスタンスに設定するか、`baseCache` インスタンスを使用する場合は `JVM` に設  
定してください。

- オプション: 組み込み区画化トポロジを使用する場合、動的キャッシュ・サー  
ビスの初期コンテナ数を構成します。

初期コンテナの数を構成することで、組み込み区画化トポロジを使用するキ  
ャッシュのパフォーマンスを最大化することができます。WebSphere  
Application Server Java 仮想マシンで、変数をシステム・プロパティとして設  
定します。

プロパティの名前を

`com.ibm.websphere.xs.dynacache.num_initial_containers` と入力します。

この構成プロパティの推奨値は、この分散キャッシュ・インスタンスにアクセ  
スする WebSphere Application Server インスタンスの総数に等しい、またはそれ  
よりわずかに少ない整数です。例えば、データ・グリッド・メンバー間で動的キ  
ャッシュ・サービスが共有される場合、この値にはデータ・グリッド・メンバ  
ーの数を設定します。

組み込みトポロジまたは組み込み区画化トポロジの場合は、WebSphere  
Application Server のバージョン 7.0 を使用する必要があります。JVM プロセ  
スで以下のカスタム・プロパティを設定して、初期コンテナをすぐに使用で  
きるようにします。

`com.ibm.ws.cache.CacheConfig.createCacheAtServerStartup=true`

- eXtreme Scale カタログ・サービス・グリッドを構成します。

分散キャッシュ・インスタンスの動的キャッシュ・プロバイダーとして eXtreme Scale を使用している場合、eXtreme Scale カタログ・サービス・ドメインを構成する必要があります。

単一のカタログ・サービス・ドメインで、eXtreme Scale でサポートされる複数の動的キャッシュ・サービス・プロバイダーに対応することができます。

カタログ・サービスは、WebSphere Application Server プロセスの内部または外部で実行されます。eXtreme Scale バージョン 7.1 から、管理コンソールを使用してカタログ・サービス・ドメインを構成すると、動的キャッシュでその設定が使用されます。追加の手順を実行してカタログ・サービスをセットアップする必要はありません。詳しくは、288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』を参照してください。

#### 7. カスタム・キー・オブジェクトを構成します。

キーとしてカスタム・オブジェクトを使用している場合、オブジェクトは `Serializable` インターフェースまたは `Externalizable` インターフェースを実装している必要があります。組み込みトポロジーあるいは組み込み区画化トポロジーを使用している場合、まるでデフォルトの動的キャッシュ・プロバイダーで使っているかのようにオブジェクトを WebSphere 共有ライブラリー・パスに配置する必要があります。詳しくは、WebSphere Application Server Network Deployment インフォメーション・センターの動的キャッシュ用の `DistributedMap` および `DistributedObjectCache` インターフェースの使用を参照してください。

リモート・トポロジーを使用している場合は、カスタム・キー・オブジェクトをスタンドアロン eXtreme Scale コンテナの `CLASSPATH` に配置する必要があります。詳しくは、454 ページの『コンテナ・サーバーの始動』を参照してください。

#### 8. オプション: リモート・トポロジーを使用する場合、eXtreme Scale コンテナ・サーバーを構成します。

##### • 組み込みトポロジーまたは組み込み区画化トポロジー:

キャッシュ・データは、WebSphere eXtreme Scale コンテナ・サーバーに保管されます。コンテナ・サーバーは、WebSphere Application Server プロセスの内部または外部で実行できます。キャッシュ・インスタンスに組み込みトポロジーまたは組み込み区画化トポロジーを使用している場合、eXtreme Scale プロバイダーは、WebSphere プロセスの内部に自動的にコンテナを作成します。これらのトポロジーの場合、それ以上の構成は必要ありません。

##### • リモート・トポロジー:

リモート・トポロジーを使用する場合、キャッシュ・インスタンスにアクセスする WebSphere Application Server インスタンスを開始する前に、スタンドアロン eXtreme Scale コンテナ・サーバーを開始しておく必要があります。詳しくは、「管理ガイド」のスタンドアロン・コンテナ・サーバーを開始するステップを参照してください。特定の動的キャッシュ・サービスのすべてのコンテナ・サーバーが、同じカタログ・サービス・エンドポイントを指していることを確認してください。

スタンドアロン eXtreme Scale 動的キャッシュ・プロバイダー・コンテナの XML 構成ファイルは、`was_root/optionalLibraries/ObjectGrid/dynacache/etc` ディレクトリー (WebSphere Application Server 上のインストールの場合)、または `wxs_install_root/ObjectGrid/dynacache/etc` ディレクトリー (スタンドアロン・インストールの場合) にあります。このファイルの名前は、`dynacache-remote-objectgrid.xml` および `dynacache-remote-definition.xml` です。これらのファイルのコピーを作成して編集し、eXtreme Scale 動的キャッシュ・プロバイダーのスタンドアロン・コンテナを開始するときに使用してください。 **dynacache-remote-deployment.xml** ファイルの **numInitialContainers** パラメーターは、実行中のコンテナ・プロセスの数に一致しなければなりません。なお、`dynacache-remote-objectgrid.xml` ファイルの **numberOfPartitions** 属性のデフォルト値は 47 です。

**注:** 一連のコンテナ・サーバー・プロセスには、リモート・トポロジーを使用するよう構成されたすべての動的キャッシュ・インスタンスを処理するために十分な空きメモリーが必要です。 `catalog.services.cluster` カスタム・プロパティーの値が同じ、または同等の値を共有する WebSphere Application Server プロセスはすべて、同じスタンドアロン・コンテナのセットを使用しなければなりません。コンテナの数とそれらのコンテナが常駐するサーバーの数は、適切に見積もる必要があります。詳細情報については、69 ページの『動的キャッシュのキャパシティー・プランニング』を参照してください。

eXtreme Scale 動的キャッシュ・プロバイダーのスタンドアロン・コンテナを開始するコマンド行エントリーを次に示します。

#### UNIX

```
startOgServer.sh container1 -objectGridFile
../dynacache/etc/dynacache-remote-objectgrid.xml -deploymentPolicyFile
../dynacache/etc/dynacache-remote-deployment.xml -catalogServiceEndPoints
MyServer1.company.com:2809
```

9. 分散トポロジーまたは組み込みトポロジーでは、サイジング・エージェントを使用可能にして、メモリー消費量の推定精度を向上させます。

サイジング・エージェントは、メモリー消費量を推定します (`usedBytes` 統計)。このエージェントでは、Java 5 以上の JVM が必要です。

以下の引数を JVM コマンド行に追加することで、エージェントをロードします。

```
-javaagent:WXS lib directory/wxssizeagent.jar
```

組み込みトポロジーでは、WebSphere Application Server プロセスのコマンド行に引数を追加します。

分散トポロジーでは、eXtreme Scale プロセス (コンテナ) および WebSphere Application Server プロセスのコマンド行に引数を追加します。

## JPA レベル 2 (L2) キャッシュ・プラグイン

WebSphere eXtreme Scale には、OpenJPA と Hibernate Java Persistence API (JPA) プロバイダーの両方に対するレベル 2 (L2) のキャッシュ・プラグインが組み込まれ

ています。これらのプラグインのいずれかを使用すると、アプリケーションは JPA API を使用します。データ・グリッドがアプリケーションとデータベースとの間に導入されて、応答時間が短縮されます。

eXtreme Scale を L2 キャッシュ・プロバイダーとして使用することにより、データ読み取りおよび照会時のパフォーマンスが向上し、データベースに対する負荷が軽減します。WebSphere eXtreme Scale ではキャッシュがすべてのプロセスで自動的に複製されるので、組み込みキャッシュ実装をしのぐ利点があります。あるクライアントが値をキャッシュすると、他のすべてのクライアントが、そのキャッシュされた値をローカルのメモリー内で使用できるようになります。

L2 キャッシュ・プロバイダーのトポロジおよびプロパティは、`persistence.xml` ファイルで構成できます。これらのプロパティの構成について詳しくは、386 ページの『JPA キャッシュ構成プロパティ』を参照してください。

**ヒント:** JPA L2 キャッシュ・プラグインは、JPA API を使用するアプリケーションを必要とします。WebSphere eXtreme Scale API を使用して JPA データ・ソースにアクセスする場合は、JPA ロードーを使用します。詳しくは、JPA ロードーを参照してください。

## JPA L2 キャッシュ・トポロジに関する考慮事項

次の要因が、構成するトポロジのタイプに影響を与えます。

### 1. キャッシュに入れられる予想データ量

- データが単一 JVM ヒープに収まる場合は、381 ページの『組み込みトポロジ』または 380 ページの『イントラドメイン・トポロジ』を使用します。
- データが単一 JVM ヒープに収まらない場合は、382 ページの『組み込みの区画化トポロジ』、または 384 ページの『リモート・トポロジ』を使用します。

### 2. 予想される読み取り/書き込み率

読み取り/書き込み率は、L2 キャッシュのパフォーマンスに影響します。トポロジごとに、読み取り操作および書き込み操作の処理は異なります。

- 381 ページの『組み込みトポロジ』: ローカル読み取り、リモート書き込み
- 380 ページの『イントラドメイン・トポロジ』: ローカル読み取り、ローカル書き込み
- 382 ページの『組み込みの区画化トポロジ』: 区画化: リモート読み取り、リモート書き込み
- 384 ページの『リモート・トポロジ』: リモート読み取り、リモート書き込み

ほとんどが読み取りのみのアプリケーションの場合、可能であれば、組み込みトポロジおよびイントラドメイン・トポロジを使用します。書き込みの方が多いアプリケーションの場合、イントラドメイン・トポロジを使用します。

### 3. データのキーによる検索に対する照会のパーセンテージ

JPA 照会キャッシュが使用可能に設定されている場合、照会操作は JPA 照会キャッシュを使用します。読み取り/書き込み率が高いアプリケーションに対しての



み JPA 照会キャッシュを使用可能に設定します。例えば、読み取り操作が 99% に迫る場合です。JPA 照会キャッシュを使用する場合、381 ページの『組み込みトポロジー』または『イントラドメイン・トポロジー』を使用する必要があります。

キーによる検索操作では、ターゲット・エンティティにリレーションシップがなければ、ターゲット・エンティティが取り出されます。ターゲット・エンティティに EAGER フェッチ・タイプとのリレーションシップがあれば、これらのリレーションシップがターゲット・エンティティと一緒に取り出されます。JPA データ・キャッシュの中でこれらのリレーションシップを取り出すと、少数のキャッシュのヒットですべてのリレーションシップ・データを取得することになります。

#### 4. 容認されるデータの失効性レベル

JVM がほとんどないシステムでは、書き込み操作でデータのレプリカ生成の待ち時間が発生します。キャッシュの目的は、同期化された最終データ・ビューをすべての JVM にわたって保守することです。イントラドメイン・トポロジーを使用している場合、書き込み操作ではデータのレプリカ生成で遅延が発生します。このトポロジーを使用しているアプリケーションの場合、データを上書きする可能性のある失効読み取りと同時書き込みを容認する必要があります。

### イントラドメイン・トポロジー

イントラドメイン・トポロジーを使用すると、プライマリー断片がトポロジー内のすべてのコンテナ・サーバーに置かれます。これらのプライマリー断片には、区画のデータの全セットが含まれています。これらのプライマリー断片はすべて、キャッシュ書き込み操作も実行できます。この構成を使用すると、すべてのキャッシュ書き込み操作が単一のプライマリー断片を経由しなければならない組み込みトポロジーのボトルネックが解決します。

イントラドメイン・トポロジーでは、構成ファイルでレプリカを定義していたとしても、レプリカ断片は作成されません。それぞれの冗長プライマリー断片にはデータの全コピーが含まれているため、それぞれのプライマリー断片をレプリカ断片とみなすこともできます。この構成は、単一区画を使用し、組み込みトポロジーと似ています。



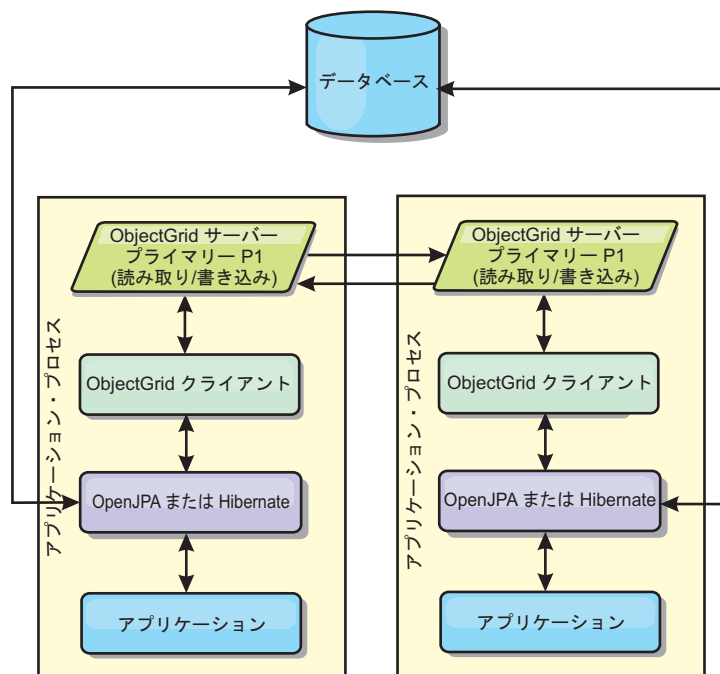


図 40. JPA インtradメイン・トポロジー

イントラドメイン・トポロジーでの関連 JPA キャッシュ構成プロパティ:

ObjectGridName=objectgrid\_name,ObjectGridType=EMBEDDED,PlacementScope=CONTAINER\_SCOPE,PlacementScopeTopology=HUB | RING

利点:

- キャッシュ読み取りおよび更新がローカルで行われる。
- 構成が簡単である。

制約:

- このトポロジーは、コンテナ・サーバーに区画データの全セットを含めることができる場合に最適です。
- すべてのコンテナ・サーバーはプライマリー断片をホストするため、レプリカ断片は、構成済みであったとしても配置されることはありません。ただし、すべてのプライマリー断片は、他のプライマリー断片を使用して複製されることになるため、これらのプライマリー断片は互いのレプリカとなります。

## 組み込みトポロジー

**ヒント:** 最高のパフォーマンスのためにイントラドメイン・トポロジーを使用することを検討してください。

組み込みトポロジーでは、各アプリケーションのプロセス・スペース内にコンテナ・サーバーを作成します。OpenJPA および Hibernate が、キャッシュのメモリ内コピーで直接読み取りを行い、他のすべてのコピーに書き込みを行います。非同期レプリカ生成を使用することによって、書き込みのパフォーマンスを向上させることができます。このデフォルト・トポロジーは、キャッシュ・データの量が少なく、1 つのプロセスに十分収まる場合に最も良く機能します。組み込みトポロジーを使用して、データの単一区画を作成します。

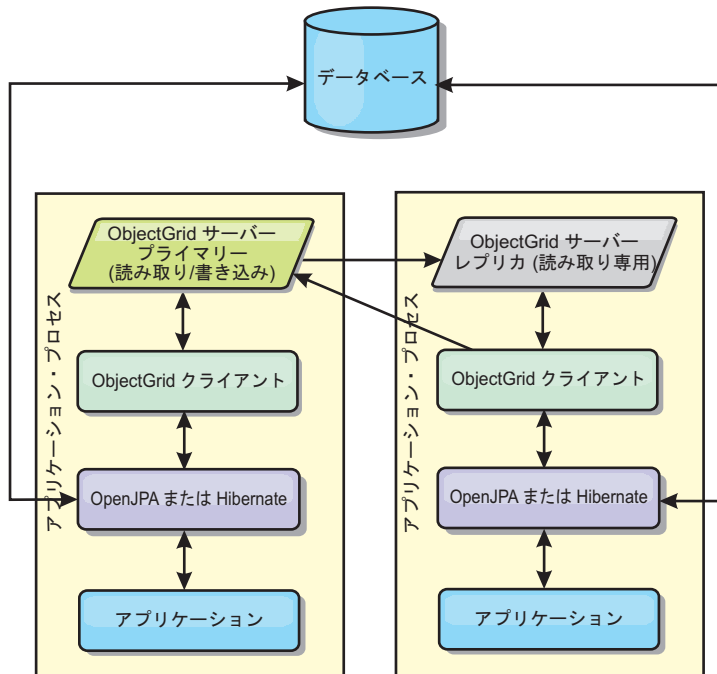


図 41. JPA 組み込みトポロジー

組み込みトポロジーでの関連 JPA キャッシュ構成プロパティ:

ObjectGridName=objectgrid\_name,ObjectGridType=EMBEDDED,MaxNumberOfReplicas=num\_replicas,ReplicaMode=SYNC | ASYNC | NONE

利点:

- すべてのキャッシュ読み取りが高速のローカル・アクセスである。
- 構成が簡単である。

制約:

- データ量が、プロセスのサイズに限られる。
- すべてのキャッシュ更新は、1 つのプライマリー断片を通じて送信される。これにより、ボトルネックが発生する。

### 組み込みの区画化トポロジー

**ヒント:** 最高のパフォーマンスのためにイントラドメイン・トポロジーを使用することを検討してください。

注意:

組み込み区画化トポロジーで JPA 照会キャッシュを使用しないでください。照会キャッシュは、エンティティ・キーのコレクションである照会結果を保管します。照会キャッシュはデータ・キャッシュからすべてのエンティティ・データを取り出します。データ・キャッシュは複数のプロセス間で分割されるため、これらの追加呼び出しによって L2 キャッシュの利点が失われる可能性があります。

キャッシュ・データの量が多すぎて 1 つのプロセスに収まらないときは、組み込み区画化トポロジーを使用できます。このトポロジーでは、データを複数のプロセス間で分割します。データはプライマリー断片間で分割されるため、それぞれのプライマリー断片にデータのサブセットが含まれます。データベース待ち時間が大きい

場合でも、このオプションを使用できます。

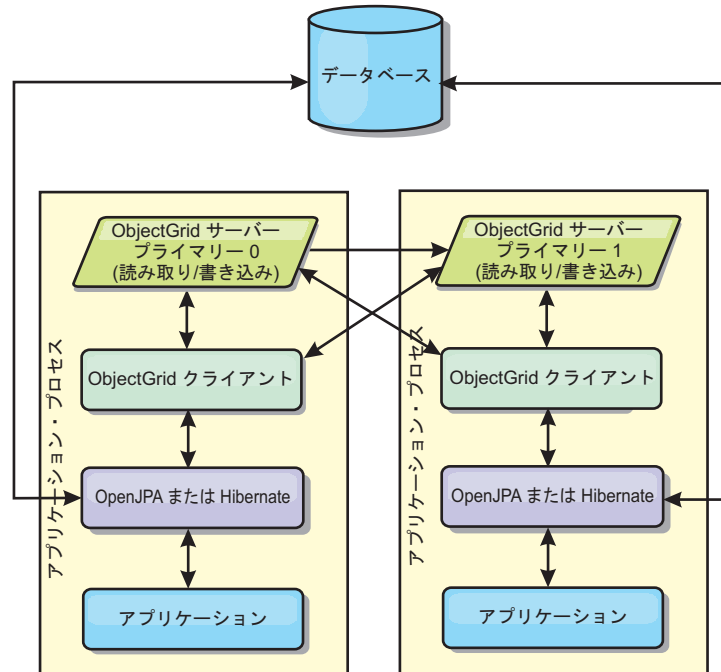


図 42. JPA 組み込み区画化トポロジー

組み込みの区画化トポロジーでの関連 JPA キャッシュ構成プロパティ:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED_PARTITION,ReplicaMode=SYNC | ASYNC | NONE,  
NumberOfPartitions=num_partitions,ReplicaReadEnabled=TRUE | FALSE
```

利点:

- 大容量のデータを保管できる。
- 構成が簡単である。
- キャッシュ更新が複数のプロセスに分散される。

制約:

- ほとんどのキャッシュ読み取りおよび更新がリモートで行われる。

例えば、JVM あたり最大 1 GB で 10 GB のデータをキャッシュに入れる場合、Java 仮想マシンが 10 必要になります。したがって、区画数は 10 以上に設定されます。理想的には、区画数を素数に設定しなければなりません。そうすると、各断片が適当な量のメモリーを保管するようになります。通常、`numberOfPartitions` は、Java 仮想マシンの数に等しくなるようにします。このように設定すれば、各 JVM に 1 つの区画が格納されます。レプリカ生成を使用可能にする場合、システム内の Java 仮想マシンの数を増やす必要があります。そうでないと、各 JVM ごとに 1 つのレプリカ区画も格納することになり、この区画はプライマリー区画と同量のメモリーを消費します。

選択された構成のパフォーマンスを最大化するには、「管理ガイド」の『メモリー・サイズ設定および区画数の計算』を参照してください。

例えば、4 つの Java 仮想マシン があるシステムで `numberOfPartitions` 設定値が 4 の場合、各 JVM は 1 つのプライマリー区画をホストします。読み取り操作では、25 パーセントの確率でローカルで使用可能な区画からデータを取り出すことができ、これは、リモート JVM からデータを取得する場合と比較してはるかに速くなります。照会の実行などの読み取り操作で 4 つの区画が均等に関わるデータ・コレクションを取り出す必要がある場合、呼び出しの 75 パーセントがリモートで、呼び出しの 25 パーセントがローカルです。`ReplicaMode` が `SYNC` または `ASYNC` に設定され、`ReplicaReadEnabled` が `true` に設定されている場合、4 つのレプリカ区画が作成され、これが 4 つの Java 仮想マシン に分散されます。各 JVM は、1 つのプライマリー区画と 1 つのレプリカ区画をホストします。読み取り操作をローカルで実行する確率は、50 パーセントに増えます。4 つの区画が均等に関わるデータ・コレクションを取り出す読み取り操作では、50 パーセントのリモート呼び出しと 50 パーセントのローカル呼び出しがあります。ローカル呼び出しは、リモート呼び出しよりはるかに高速です。リモート呼び出しが行われると、パフォーマンスは落ちます。

## リモート・トポロジー

### 注意:

リモート・トポロジーで JPA 照会キャッシュを使用しないでください。照会キャッシュは、エンティティ・キーのコレクションである照会結果を保管します。照会キャッシュはデータ・キャッシュからすべてのエンティティ・データを取り出します。データ・キャッシュはリモートであるため、これらの追加呼び出しによって L2 キャッシュの利点が失われる可能性があります。

**ヒント:** 最高のパフォーマンスのためにイントラドメイン・トポロジーを使用することを検討してください。

リモート・トポロジーでは、すべてのキャッシュ・データを 1 つ以上の個別のプロセスに保管し、アプリケーション・プロセスのメモリー使用を減らします。区画化され、複製された eXtreme Scale データ・グリッドをデプロイすることによって、個別のプロセスへデータ配布するという利点が生かされます。前のセクションで説明した組み込み構成および組み込み区画化構成とは対照的に、リモート・データ・グリッドを管理する場合は、アプリケーションおよび JPA プロバイダーから独立して管理する必要があります。

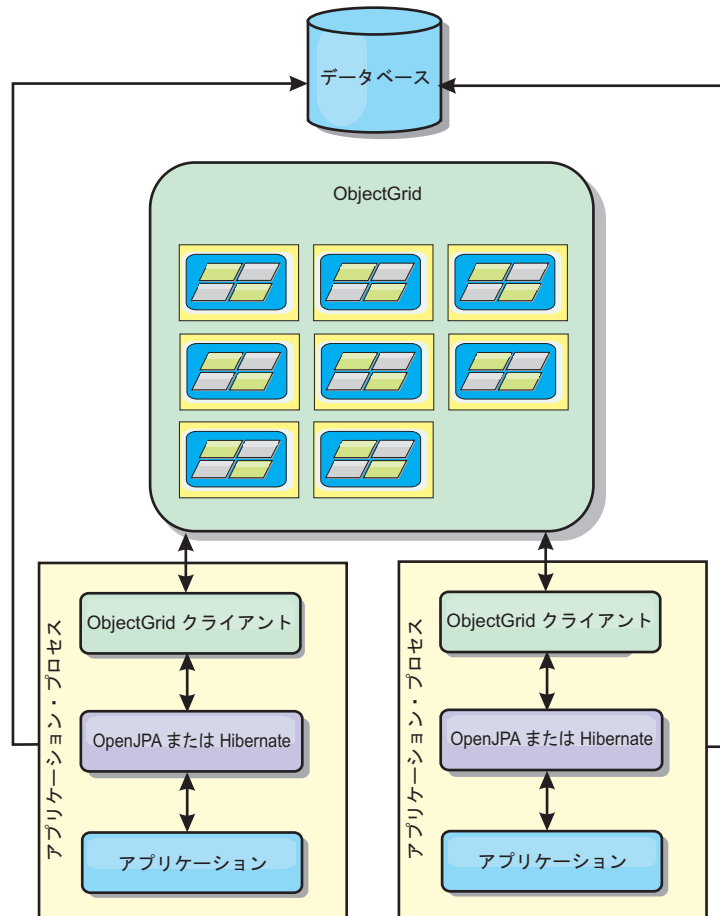


図 43. JPA リモート・トポロジー

リモート・トポロジーでの関連 JPA キャッシュ構成プロパティ:

`ObjectGridName=objectgrid_name,ObjectGridType=REMOTE`

REMOTE ObjectGrid タイプでは、ObjectGrid およびデプロイメント・ポリシーが JPA アプリケーションとは別に定義されるため、プロパティ設定は必要ありません。JPA キャッシュ・プラグインは、リモートで、既存のリモート ObjectGrid に接続します。

ObjectGrid とのすべての対話のリモートで行われるため、このトポロジーは、すべての ObjectGrid タイプの中で、パフォーマンスが最も遅くなります。

利点:

- 大容量のデータを保管できる。
- アプリケーション・プロセスがキャッシュ・データから解放される。
- キャッシュ更新が複数のプロセスに分散される。
- 柔軟な構成オプションがある。

制約:

- すべてのキャッシュ読み取りおよび更新がリモートで行われる。

## JPA キャッシュ構成プロパティ

WebSphere eXtreme Scale には、OpenJPA と Hibernate Java Persistence API (JPA) プロバイダーの両方に対するレベル 2 のキャッシュ・プラグインが組み込まれています。L2 キャッシュ・プラグインを構成するには、`persistence.xml` ファイル内のプロパティを更新する必要があります。

**ヒント:** JPA L2 キャッシュ・プラグインは、JPA API を使用するアプリケーションを必要とします。WebSphere eXtreme Scale API を使用して JPA データ・ソースにアクセスする場合は、JPA ロードーを使用します。詳しくは、405 ページの『JPA ロードーの構成』を参照してください。

### プロパティの場所

これらのプロパティは、`persistence.xml` ファイルの中で構成できます。これらのプロパティをこのファイルの中で指定するための構文は、OpenJPA を使用しているか Hibernate を使用しているかによって異なります。

- **OpenJPA:** DataCache または QueryCache でプロパティを設定できます。

```
<property name="openjpa.DataCache"
  value="<object_grid_datacache_class(<property>=<value>,...)" />
```

または

```
<property name="openjpa.QueryCache"
  value="<object_grid_querycache_class(<property>=<value>,...)" />
```

- **Hibernate:**

```
<property name="objectgrid.configuration"
  value="<property>=<value>,..." />
```

### デフォルトのトポロジーおよびプロパティ

構成で何の値も指定されない場合、次のデフォルト・プロパティ値が使用されます。

- **ObjectGridName:** パーシスタンス・ユニット名
- **ObjectGridType:** EMBEDDED
- **NumberOfPartitions:** 1 ( ObjectGrid タイプが EMBEDDED の場合、変更不可)
- **ReplicaMode:** SYNC
- **ReplicaReadEnabled:** TRUE (ObjectGrid タイプが EMBEDDED の場合、変更不可)
- **MaxUsedMemory:** TRUE
- **MaxNumberOfReplicas:** 47 (分散システムでは、Java 仮想マシンの数以下でなければなりません)

### プロパティ

次のプロパティを使用して、JPA キャッシュ・プラグインを構成できます。

#### ObjectGridName

固有の ObjectGrid 名を指定します。デフォルト値は、定義済みのパーシスタンス・ユニット名になります。パーシスタンス・ユニット名が JPA プロバイダーから使用できない場合、生成名が使用されます。



## ObjectGridType

ObjectGrid のタイプを指定します。

有効な値:

### EMBEDDED

デフォルトおよび推奨の構成タイプ。そのデフォルト設定には、NumberOfPartitions=1、ReplicaMode=SYNC、ReplicaReadEnabled=true および MaxNumberOfReplicas=47 が含まれます。 **ReplicaMode** パラメーターはレプリカ生成モードの設定に、 **MaxNumberOfReplicas** パラメーターはレプリカの最大数の設定に使用します。システムに 47 を超える Java 仮想マシンがある場合、 **MaxNumberOfReplicas** 値を Java 仮想マシンの数に等しくなるように設定します。

### EMBEDDED\_PARTITION

システムが、分散システムで大量のデータをキャッシュに入れる必要がある場合に使用するタイプ。デフォルトの区画数は 47 でレプリカ・モードは NONE です。少数の Java 仮想マシンのみを持つ小規模のシステムでは、 **NumberOfPartitions** を Java 仮想マシンの数以下に設定します。システムを調整するために、 **ReplicaMode**、 **NumberOfPartitions**、および **ReplicaReadEnabled** 値を指定できます。

**REMOTE** キャッシュは、カタログ・サービスからリモートの分散 ObjectGrid に接続しようとしています。

## MaxNumberOfReplicas

キャッシュに使用されるレプリカの最大数を指定します。この値は、EMBEDDED タイプのみに適用されます。この数値は、システム内の Java 仮想マシン 数以上にする必要があります。デフォルト値は 47 です。

有効な値: 1 以上

## MaxUsedMemory

**有効な値:** TRUE または FALSE メモリーに余裕がなくなった場合にキャッシュ・エントリーの除去を使用可能にします。デフォルト値は TRUE で、JVM ヒープ使用率しきい値が 70 % を超えると、データの除去が開始されます。デフォルトの JVM ヒープ使用率しきい値の比率 (%) は、objectGridServer.properties ファイルの memoryThresholdPercentage プロパティを設定し、このファイルをクラスパスに配置することによって変更することができます。Evictor について詳しくは、キャッシュ・オブジェクトの除去のためのプラグイン「製品概要」で Evictor に関する情報を参照してください。サーバー・プロパティ・ファイルに関して詳しくは、サーバー・プロパティ・ファイルを参照してください。

## NumberOfPartitions

**有効な値:** 1 以上キャッシュに使用される区画の数を指定します。このプロパティは、ObjectGridType の値が EMBEDDED\_PARTITION に設定されている場合に適用されます。デフォルト値は 47 です。EMBEDDED タイプの場合、 **NumberOfPartitions** 値は常に 1 です。

## PlacementScope

マップ・セットの単一インスタンスの細分度を示します。

有効な値:

### DOMAIN\_SCOPE

(デフォルト) 区画ごとに 1 つのプライマリー断片をカタログ・サービス・ドメイン内のコンテナ・サーバーに配置します。各区画のレプリカ断片は、カタログ・サービス・ドメイン内の他のコンテナ・サーバーに配置されます。

### CONTAINER\_SCOPE

カタログ・サービス・ドメイン内の各コンテナ・サーバーに、1 つのプライマリー断片を配置します。

## PlacementScopeTopology

カタログ・サービス・ドメイン内のコンテナ・サーバーのリンク・トポロジーを定義します。この値は、PlacementScope 値が DOMAIN\_SCOPE 以外の値に設定されたときにのみ使用します。

有効な値:

**HUB** (デフォルト) ハブ・トポロジーが選択されると、1 つのデータ・グリッドがハブとして選択されます。その他のすべてのデータ・グリッドは、ハブに接続します。スポークの接続が 1 つであるため、このトポロジーは、極めて拡張が容易です。ハブは、ボトルネックおよび一時的な単一障害点となる可能性があります。ハブに障害が起これば、別のコンテナ・サーバーに再配置されます。この構成の利点は、単一ポイントであるハブがすべての競合に対処できる、さらに複雑なアービトレーション・コードを作成できることです。

**RING** リング・トポロジーが選択されると、各データ・グリッドは他の 2 つのデータ・グリッドとリンクされます。リンクの順序は保証されません。しかし、開始する各コンテナは、リングに追加される最初のコンテナと最後のコンテナとにリンクされることが十分考えられます。このトポロジーが最もスケラブルですが、2 つのリンクに障害が起これば一時的に切断されてしまいます。コンテナ・サーバーに障害が起これば、障害が検出された後、残った正常なサーバーの間で、リンクが確立されます。

## ReplicaMode

有効な値: SYNC/ASync/NONE キャッシュをレプリカにコピーする場合に使用するメソッドを指定します。このプロパティは、ObjectGridType の値を EMBEDDED または EMBEDDED\_PARTITION に設定している場合に適用されます。デフォルト値は、EMBEDDED\_PARTITION タイプの場合が NONE で、EMBEDDED タイプの場合は SYNC です。ReplicaMode が、EMBEDDED ObjectGridType の場合に NONE に設定されても、EMBEDDED タイプではやはり SYNC の ReplicaMode が使用されます。

## ReplicaReadEnabled

有効な値: TRUE または FALSE 使用可能にする場合、クライアントはレプリカから読み込みます。このプロパティは、EMBEDDED\_PARTITION タイプに

適用されます。デフォルト値は、EMBEDDED\_PARTITION タイプの場合 FALSE です。EMBEDDED タイプの場合は、常に **ReplicaReadEnabled** の値は TRUE に設定されます。

### writeBehind

**Hibernate プロバイダーの場合のみ:** writeBehind が使用可能になっていると、writeBehindInterval 条件または writeBehindMaxBatchSize 条件のどちらかが満たされるまで、更新は一時的に JVM の有効範囲データ・ストレージに保管されます。

**重要:** writeBehind が使用可能でない場合、他の後書き構成設定は無視されます。

**重要:** 後書き機能を使用する場合は注意してください。後書き構成では、JVM 全体でデータを同期する待ち時間はさらに長くなり、更新が失われる可能性がより高くなります。4 つ以上の JVM を使用可能にした後書き構成を持つシステムでは、1 つの JVM で実行された更新は、約 15 秒遅れてから、他の JVM で使用可能になります。任意の 2 つの JVM が同じエントリーを更新する場合は、最初に更新をフラッシュする 1 つの JVM はその更新を失います。

**有効な値:** TRUE または FALSE

**デフォルト値:** FALSE

### writeBehindInterval

**Hibernate プロバイダーの場合のみ:** 更新をキャッシュにフラッシュする時間間隔 (ミリ秒) を指定します。

**有効な値:** 1 以上

**デフォルト値:** 5000 (5 秒)

### writeBehindPoolSize

**Hibernate プロバイダーの場合のみ:** 更新をキャッシュにフラッシュするとき使用するスレッド・プールの最大サイズを指定します。

**有効な値:** 1 以上

**デフォルト値:** 5

### writeBehindMaxBatchSize

**Hibernate プロバイダーの場合のみ:** 更新をキャッシュにフラッシュするときの、領域キャッシュごとの最大バッチ・サイズを指定します。例えば、サイズが 1000 に設定されていて、領域キャッシュの後書きストレージに保管された更新が 1000 エントリーを超えた場合は、指定された

writeBehindInterval 条件が満たされなくても、更新はキャッシュにフラッシュされます。おおよそ writeBehindInterval 値に指定された秒ごとか、または各領域キャッシュの後書きストレージのサイズが 1000 エントリーを超えたときか、そのいずれかで、更新はキャッシュにフラッシュされます。

writeBehindMaxBatchSize 条件が一致する場合は注意してください。この条件に一致する領域キャッシュのみが後書きストレージの更新をキャッシュにフラッシュします。領域キャッシュは、通常、エンティティーまたは照会に対応します。

有効な値: 1 以上

デフォルト値: 1000

## OpenJPA キャッシュ・プラグインの構成

OpenJPA の場合、DataCache 実装と QueryCache 実装の両方を構成できます。

### 始める前に

- 使用する JPA キャッシュ・プラグイン・トポロジーを決定する必要があります。各種構成や各トポロジーで設定すべきプロパティの詳細については、378 ページの『JPA レベル 2 (L2) キャッシュ・プラグイン』を参照してください。
- JPA API を使用するアプリケーションが必要です。WebSphere eXtreme Scale API を使用して JPA のデータにアクセスする必要がある場合は、JPA ロードラーを使用してください。詳しくは、405 ページの『JPA ロードラーの構成』を参照してください。

### 手順

1. persistence.xml ファイル内のプロパティを設定して、OpenJPA キャッシュ・プラグインを構成します。DataCache 実装または QueryCache 実装でこれらのプロパティを設定できます。

DataCache および QueryCache 構成は、互いに独立しています。いずれかの構成を使用可能にすることができます。ただし、両方の構成を使用可能にした場合、QueryCache 構成では、DataCache 構成と同じ構成が使用され、その構成プロパティは廃棄されます。

```
<property name="openjpa.DataCache"
  value="<object_grid_datacache_class(<property>=<value>,...)" />
```

または

```
<property name="openjpa.QueryCache"
  value="<object_grid_querycache_class(<property>=<value>,...)" />
```

**注:** QueryCache 構成を使用可能にできるのは、組み込みトポロジーと組み込みイントラドメイン・トポロジーの場合のみです。

ObjectGrid キャッシュ・クラスのプロパティ・リストに ObjectGridName プロパティ、ObjectGridType プロパティ、その他の単純なデプロイメント・ポリシー関連のプロパティを指定すれば、キャッシュ構成をカスタマイズすることができます。次に例を挙げます。

```
<property name="openjpa.DataCache"
  value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(
  ObjectGridName=BasicTestObjectGrid,ObjectGridType=EMBEDDED,
  maxNumberOfReplicas=4)" />
<property name="openjpa.QueryCache"
  value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()" />
<property name="openjpa.RemoteCommitProvider" value="sjvm" />
```

設定できるプロパティのリストについては、386 ページの『JPA キャッシュ構成プロパティ』を参照してください。

2. persistence.xml ファイル内では、openjpa.RemoteCommitProvider プロパティを sjvm に設定する必要もあります。

```
<property name="openjpa.RemoteCommitProvider" value="sjvm" />
```

3. オプション: キャッシュで使用するデータ・グリッドをさらにカスタマイズするには、XML ファイルを使用して追加設定を指定できます。

ほとんどのシナリオでは、キャッシュ・プロパティーを設定すれば十分です。キャッシュによって使用される ObjectGrid をさらにカスタマイズするには、OpenJPA ObjectGrid 構成 XML ファイルを、persistence.xml ファイルと同様に META-INF ディレクトリーに提供することができます。初期化時に、キャッシュはこれらの XML ファイルを検索し、検出されるとそれらのファイルを処理します。

OpenJPA ObjectGrid 構成 XML ファイルには次の 3 つのタイプがあります。

- openjpa-objectGrid.xml (ObjectGrid 構成)

ファイル・パス: META-INF/openjpa-objectGrid.xml

このファイルは、EMBEDDED および EMBEDDED\_PARTITION タイプの両方に対して ObjectGrid 構成をカスタマイズする場合に使用されます。REMOTE タイプの場合、このファイルは無視されます。デフォルトでは、各エンティティー・クラスは、ObjectGrid 構成内のエンティティー・クラス名として名付けられた独自の BackingMap 構成にマップされます。例えば、com.mycompany.Employee エンティティー・クラスは、com.mycompany.Employee BackingMap にマップされます。デフォルト BackingMap 構成は、readOnly="false"、copyKey="false"、lockStrategy="NONE"、および copyMode="NO\_COPY" です。一部の BackingMap を選択された構成でカスタマイズできます。ALL\_ENTITY\_MAPS 予約キーワードを使用すれば、openjpa-objectGrid.xml ファイルにリストされた他のカスタマイズ・マップを除くすべてのマップを示すことができます。この openjpa-objectGrid.xml ファイルにリストされていない BackingMap は、デフォルト構成を使用します。カスタマイズされた BackingMaps が BackingMaps 属性またはプロパティーを指定しておらず、それらの属性がデフォルト構成で指定されている場合は、デフォルト構成の属性値が適用されます。例えば、エンティティー・クラスが timeToLive=30 でアノテーションを付けられている場合、そのエンティティーのデフォルトの BackingMap 構成には timeToLive=30 が設定されます。カスタム openjpa-objectGrid.xml ファイルにもその BackingMap が含まれているが、timeToLive 値を指定していない場合、カスタマイズされた BackingMap には、デフォルトで、timeToLive=30 値が設定されます。openjpa-objectGrid.xml ファイルは、デフォルト構成をオーバーライドまたは拡張しようとします。

- openjpa-objectGridDeployment.xml (デプロイメント・ポリシー)

ファイル・パス: META-INF/openjpa-objectGridDeployment.xml

このファイルは、デプロイメント・ポリシーのカスタマイズに使用されます。デプロイメント・ポリシーをカスタマイズする場合、openjpa-objectGridDeployment.xml ファイルが提供されている場合は、デフォルトのデプロイメント・ポリシーは廃棄されます。デプロイメント・ポリシー属性値は、すべて openjpa-objectGridDeployment.xml ファイルから提供されます。

- openjpa-objectGrid-client-override.xml (クライアント ObjectGrid オーバーライド構成)



ファイル・パス: META-INF/openjpa-objectGrid-client-override.xml

このファイルは、クライアント・サイド ObjectGrid のカスタマイズに使用されます。デフォルトでは、ObjectGrid キャッシュは、ニア・キャッシュを使用不可にするデフォルト・クライアント・オーバーライド ObjectGrid 構成を適用します。アプリケーションがニア・キャッシュを必要とする場合、アプリケーションはこのファイルを提供し、numberOfBuckets="xxx" を指定することができます。デフォルトのクライアント・オーバーライドでは、numberOfBuckets="0" を設定し、ニア・キャッシュを使用不可にします。ニア・キャッシュは、openjpa-objectGrid-client-override.xml ファイルによって numberOfBuckets の値をゼロより大きい値に再設定すれば、アクティブにすることができます。openjpa-objectGrid-client-override.xml ファイルの機能は、openjpa-objectGrid.xml ファイルに似ています。このファイルは、デフォルトのクライアント ObjectGrid オーバーライド構成をオーバーライドまたは拡張します。

構成された eXtreme Scale トポロジーに応じて、これらの 3 つの XML ファイルのいずれか任意のファイルを提供してそのトポロジーをカスタマイズすることができます。

EMBEDDED と EMBEDDED\_PARTITION のタイプの場合、3 つの XML ファイルの任意のいずれかを提供し、ObjectGrid、デプロイメント・ポリシー、およびクライアント ObjectGrid オーバーライド構成をカスタマイズできます。

REMOTE ObjectGrid の場合、ObjectGrid キャッシュは動的 ObjectGrid を作成しません。代わりにキャッシュは、カタログ・サービスからクライアント・サイドの ObjectGrid を取得するだけです。openjpa-objectGrid-client-override.xml ファイルを提供して、クライアント ObjectGrid オーバーライド構成をカスタマイズできるだけです。

4. オプション: (リモート構成のみ) REMOTE ObjectGrid タイプのキャッシュを構成する場合、外部 eXtreme Scale システムをセットアップします。

REMOTE ObjectGrid タイプでキャッシュを構成する場合、外部 eXtreme Scale システムをセットアップする必要があります。外部システムをセットアップするためには、persistence.xml ファイルに基づく ObjectGrid および ObjectGridDeployment 構成 XML ファイルの両方が必要になります。これらの構成ファイルの例については、393 ページの『例: OpenJPA ObjectGrid XML ファイル』を参照してください。

## タスクの結果

### EMBEDDED、EMBEDDED\_PARTITION、またはドメイン内構成:

アプリケーションを開始すると、プラグインが自動的にカタログ・サービスを検出または開始し、コンテナ・サーバーを開始し、コンテナ・サーバーをカタログ・サービスに接続します。それから、プラグインは、クライアント接続を使用して別のアプリケーション・サーバー・プロセスで実行されている ObjectGrid コンテナおよびそのピアと通信します。

### REMOTE 構成:



デプロイメント・ポリシーは JPA アプリケーションとは別に指定されます。外部 ObjectGrid システムには、カタログ・サービスおよびコンテナ・サーバー・プロセスの両方があります。コンテナ・サーバーを始動する前に、カタログ・サービスを始動する必要があります。詳しくは、451 ページの『スタンドアロン・サーバーの始動』と 454 ページの『コンテナ・サーバーの始動』を参照してください。

## 次のタスク

- この構成を使用する OpenJPA アプリケーションを開発します。詳しくは、例: ObjectGrid キャッシュにデータをプリロードするための Hibernate プラグインの使用を参照してください。
- 実稼働環境の場合、EMBEDDED または EMBEDDED\_PARTITION 構成用に自動的に作成されるプロセスのカタログ・サービス・ドメインを作成します。
  - スタンドアロン環境:

WebSphere Application Server プロセス内でサーバーを実行しない場合、カタログ・サービス・ドメインのホストおよびポートは、`objectGridServer.properties` というプロパティ・ファイルを使用して指定されます。このファイルは、アプリケーションのクラスパスに保管し、**catalogServiceEndPoints** プロパティを定義しておく必要があります。カタログ・サービス・ドメインは、アプリケーション・プロセスとは別に開始され、アプリケーション・プロセスが開始される前に開始されなければなりません。

`objectGridServer.properties` ファイルのフォーマットは次のとおりです。

```
catalogServiceEndPoints=<hostname1>:<port1>,<hostname2>:<port2>
```

- WebSphere Application Server 環境:

WebSphere Application Server プロセス内で実行する場合、JPA キャッシュ・プラグインでは、WebSphere Application Server セルに定義されたカタログ・サービスまたはカタログ・サービス・ドメインに自動的に接続します。

- EMBEDDED または EMBEDDED\_PARTITION ObjectGridType を Java SEJava SE 環境で使用する場合、組み込み eXtreme Scale サーバーを停止するため、プログラムの末尾に `System.exit(0)` メソッドを使用してください。そうしないと、プログラムが応答しなくなる可能性があります。

## 例: OpenJPA ObjectGrid XML ファイル:

OpenJPA ObjectGrid XML ファイルは、パーシスタンス・ユニットの構成に基づいて作成する必要があります。

### persistence.xml ファイル

パーシスタンス・ユニットの構成を表す `persistence.xml` ファイルの例を以下に示します。

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistibleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
```

```

<class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
<class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
<class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
<class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
<class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>
<exclude-unlisted-classes>true</exclude-unlisted-classes>

<properties>
<!-- Database setting -->

<!-- enable cache -->
<property name="openjpa.DataCache"
value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(objectGridName=Annuity,
objectGridType=EMBEDDED, maxNumberOfReplicas=4)" />
<property name="openjpa.RemoteCommitProvider" value="sjvm" />
<property name="openjpa.QueryCache"
value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()" />
</properties>
</persistence-unit>
</persistence>

```

## openjpa-objectGrid.xml ファイル

openjpa-objectGrid.xml ファイルは、EMBEDDED および EMBEDDED\_PARTITION タイプの両方に対して ObjectGrid 構成をカスタマイズする場合に使用されます。以下に示すのは、この persistence.xml ファイルに適合する openjpa-objectGrid.xml ファイルです。

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
<objectGrid name="Annuity">
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistibleObject"
readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistibleObject" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
<backingMap name="ObjectGridQueryCache" readOnly="false" copyKey="false"
lockStrategy="NONE" copyMode="NO_COPY" pluginCollectionRef="ObjectGridQueryCache"
evictionTriggers="MEMORY_USAGE_THRESHOLD" />
</objectGrid>
</objectGrids>
<backingMapPluginCollections>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
<bean id="ObjectTransformer"
className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
<bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
</bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address">
<bean id="ObjectTransformer"
className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
<bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
</bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
<bean id="ObjectTransformer"
className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
<bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
</bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">

```

```

    <bean id="ObjectTransformer"
        className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
    <bean id="ObjectTransformer"
        className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection
    id="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistibleObject">
    <bean id="ObjectTransformer"
        className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
    <bean id="ObjectTransformer"
        className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
    <bean id="ObjectTransformer"
        className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="ObjectGridQueryCache">
    <bean id="MapIndexPlugin" className="com.ibm.websphere.objectgrid.plugins.index.HashIndex" >
        <property name="Name" type="java.lang.String"
            value="QueryCacheKeyIndex" description="name of index"/>
        <property name="POJOKeyIndex" type="boolean" value="true" description="POJO Key Index" />
    </bean>
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>

```

### 重要:

1. 各エンティティは、完全修飾エンティティ・クラス名として名付けられた BackingMap にマップされます。

デフォルトでは、エンティティは第 2 レベル・キャッシュの一部です。キャッシングから除外する必要がある Entity クラスで、L2 キャッシュから除外するエンティティ・クラスに @DataCache(enabled=false) アノテーションを含めることができます。

```

import org.apache.openjpa.persistence.DataCache;
@Entity
@DataCache(enabled=false)
public class OpenJPACacheTest { ... }

```

2. エンティティ・クラスが、継承の階層にある場合、子クラスは親の BackingMap にマップされます。継承の階層は単一の BackingMap を共有しません。
3. QueryCache のサポートには ObjectGridQueryCache マップが必要です。
4. 各エンティティ・マップの backingMapPluginCollection には、com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer クラスを使用する ObjectTransformer がなければなりません。
5. ObjectGridQueryCache マップの backingMapPluginCollection には、サンプルに示されているように QueryCacheKeyIndex と名付けられたキー索引がなければなりません。
6. 各マップで、Evictor はオプションです。

## openjpa-objectGridDeployment.xml ファイル

openjpa-objectGridDeployment.xml ファイルは、デプロイメント・ポリシーのカスタマイズに使用されます。以下に示すのは、この persistence.xml ファイルに適合する openjpa-objectGridDeployment.xml ファイルです。

### openjpa-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1"
      minSyncReplicas="0" maxSyncReplicas="4" maxAsyncReplicas="0"
      replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistibleObject" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="ObjectGridQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

注: QueryCache のサポートには ObjectGridQueryCache マップが必要です。

## Hibernate キャッシュ・プラグインの構成

プロパティ・ファイルを指定することで、Hibernate キャッシュ・プラグインを使用するキャッシュを使用可能にできます。

### 始める前に

- 使用する JPA キャッシュ・プラグイン・トポロジーを決定する必要があります。各構成の詳細については、378 ページの『JPA レベル 2 (L2) キャッシュ・プラグイン』を参照してください。
- JPA API を使用するアプリケーションが必要です。WebSphere eXtreme Scale API を使用して JPA のデータにアクセスする必要がある場合は、JPA ロードーを使用してください。詳しくは、405 ページの『JPA ロードーの構成』を参照してください。

### 手順

1. WebSphere Application Server を使用する場合、Java アーカイブ (JAR) ファイルを構成内の適切な場所に配置します。

Hibernate キャッシュ・プラグインは、oghibernate-cache.jar ファイル内にパッケージ化されていて、was\_root/optionalLibraries/ObjectGrid ディレクトリにインストールされます。Hibernate キャッシュ・プラグインを使用するには、Hibernate ライブラリーに oghibernate-cache.jar ファイルを組み込む必要があります。例えば、使用するアプリケーションに Hibernate ライブラリーを組み込む場合は、oghibernate-cache.jar ファイルも組み込まなければなりません。共有ライブラリーを定義して Hibernate ライブラリーを組み込む場合は、oghibernate-cache.jar ファイルをその共有ライブラリー・ディレクトリに追加しなければなりません。

eXtreme Scale は、cglib.jar ファイルを WebSphere Application Server 環境にインストールしません。cglib.jar に依存する既存アプリケーションまたは共

有ライブラリー (Hibernate など) がある場合、cglib.jar ファイルを見つけ、クラスパスに組み込んでください。例えば、アプリケーションに Hibernate ライブラリーのすべての JAR ファイルが組み込まれていながら、Hibernate で使用可能な cglib.jar ファイルが除外されている場合、Hibernate の cglib.jar ファイルをアプリケーションに組み込む必要があります。

2. persistence.xml ファイル内のプロパティを設定して、Hibernate キャッシュ・プラグインを構成します。

persistence.xml ファイル内のプロパティを設定する構文は次のとおりです。

```
<property name="hibernate.cache.provider_class"
    value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
<property name="hibernate.cache.use_query_cache" value="true"/>
<property name="objectgrid.configuration" value="<property>=<value>,..." />
<property name="objectgrid.hibernate.regionNames" value="<regionName>,..." />
```

- **hibernate.cache.provider\_class: provider\_class** プロパティの値は com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider クラスです。
- **hibernate.cache.use\_query\_cache:** 照会キャッシュを使用可能にするには、**use\_query\_cache** プロパティの値を true に設定します。

注: 照会キャッシュを使用可能にできるのは、組み込みトポロジーと組み込みイントラドメイン・トポロジーの場合のみです。

- **objectgrid.configuration:** objectgrid.configuration プロパティを使用して、eXtreme Scale キャッシュ構成プロパティを指定します。これにはデータ・グリッドに断片を配置する方法を指定する ObjectGridType 属性も含まれます。

名前が競合する可能性をなくすため、固有の ObjectGridName プロパティ値を指定する必要があります。他の eXtreme Scale キャッシュ構成プロパティはオプションです。

後書きキャッシングを使用可能にするには、次のように objectgrid.configuration プロパティの後書き属性を使用します。後書きキャッシングを使用可能になると、データがキャッシュにフラッシュされたとき、更新は writeBehindInterval 条件または writeBehindMaxBatchSize 条件が満たされるまで、JVM スコープのデータ・ストレージに一時的に保管されます。

```
writeBehind=true, writeBehindInterval=5000, writeBehindPoolSize=10, writeBehindMaxBatchSize=1000
```

**重要:** writeBehind が使用可能でない場合、他の後書き構成設定は無視されません。

**objectgrid.configuration** プロパティに設定できる値の詳細については、386 ページの『JPA キャッシュ構成プロパティ』を参照してください。

- **objectgrid.hibernate.regionNames:** objectgrid.hibernate.regionNames プロパティはオプションですが、eXtreme Scale キャッシュの初期化後に regionNames の値を定義する場合は指定する必要があります。regionName にマップされるエンティティ・クラスで、そのエンティティ・クラスが persistence.xml ファイルに指定されていないか、または Hibernate マッピング・ファイルに含まれていない例を考えます。さらに、これにエンティティ・アノテーションが設定されているとします。これにより、このエンティティ・クラスの regionName は、eXtreme Scale キャッシュが初期化される場



合、クラス・ロード時に解決されます。別の例としては、eXtreme Scale キャッシュ初期化後に実行される `Query.setCacheRegion(String regionName)` メソッドがあります。こうした状態では、動的決定の可能性のあるすべての `regionNames` を `objectgrid.hibernate.regionNames` プロパティーに組み入れ、eXtreme Scale キャッシュがすべての `regionNames` の `BackingMaps` を準備できるようにします。

3. オプション: キャッシュで使用するデータ・グリッドをさらにカスタマイズするには、XML ファイルを使用して追加設定を指定できます。

ほとんどのシナリオでは、キャッシュ・プロパティーを設定すれば十分です。キャッシュによって使用される `ObjectGrid` をさらにカスタマイズするには、Hibernate `ObjectGrid` 構成 XML ファイルを、`persistence.xml` ファイルと同様に `META-INF` ディレクトリーに提供することができます。初期化時に、キャッシュはこれらの XML ファイルを検索し、検出されるとそれらのファイル进行处理します。

Hibernate `ObjectGrid` 構成 XML ファイルには次の 3 つのタイプがあります。

- `hibernate-objectGrid.xml` (`ObjectGrid` 構成)

**ファイル・パス:** `META-INF/hibernate-objectGrid.xml`

デフォルトでは、各エンティティー・クラスには、`regionName` が関連付けられ (エンティティー・クラス名にデフォルト設定)、その `regionName` が、`ObjectGrid` 構成内の `regionName` として名付けられた `BackingMap` 構成にマップされます。例えば、`com.mycompany.Employee` エンティティー・クラスには、`com.mycompany.Employee BackingMap` とデフォルト設定される `regionName` が関連付けられます。デフォルト `BackingMap` 構成は、`readOnly="false"`、`copyKey="false"`、`lockStrategy="NONE"`、および `copyMode="NO_COPY"` です。一部の `BackingMap` を選択された構成でカスタマイズできます。予約キーワード「`ALL_ENTITY_MAPS`」を使用すれば、`hibernate-objectGrid.xml` ファイルにリストされた他のカスタマイズ・マップを除くすべてのマップを示すことができます。この `hibernate-objectGrid.xml` ファイルにリストされていない `BackingMap` は、デフォルト構成を使用します。

- `hibernate-objectGridDeployment.xml` (デプロイメント・ポリシー)

**ファイル・パス:** `META-INF/hibernate-objectGridDeployment.xml`

このファイルは、デプロイメント・ポリシーのカスタマイズに使用されます。デプロイメント・ポリシーをカスタマイズする場合、`hibernate-objectGridDeployment.xml` が提供されている場合は、デフォルトのデプロイメント・ポリシーは廃棄されます。すべてのデプロイメント・ポリシー属性値は、この提供された `hibernate-objectGridDeployment.xml` ファイルから得られます。

- `hibernate-objectGrid-client-override.xml` (クライアント `ObjectGrid` オーバーライド構成)

**ファイル・パス:** `META-INF/hibernate-objectGrid-client-override.xml`



このファイルは、クライアント・サイド ObjectGrid のカスタマイズに使用されます。デフォルトでは、ObjectGrid キャッシュは、ニア・キャッシュを使用不可にするデフォルト・クライアント・オーバーライド構成を適用します。アプリケーションがニア・キャッシュを必要とする場合、アプリケーションはこのファイルを提供し、numberOfBuckets="xxx" を指定することができます。デフォルトのクライアント・オーバーライドでは、numberOfBuckets="0" を設定し、ニア・キャッシュを使用不可にします。ニア・キャッシュは、hibernate-objectGrid-client-override.xml ファイルによって numberOfBuckets 属性の値をゼロより大きい値に再設定すれば、アクティブにすることができます。hibernate-objectGrid-client-override.xml ファイルの機能は、hibernate-objectGrid.xml に似ています。このファイルは、デフォルトのクライアント ObjectGrid オーバーライド構成をオーバーライドまたは拡張します。

構成された eXtreme Scale トポロジーに応じて、これらの 3 つの XML ファイルのいずれか任意のファイルを提供してそのトポロジーをカスタマイズすることができます。

EMBEDDED と EMBEDDED\_PARTITION の両方のタイプの場合、3 つの XML ファイルの任意のいずれかを提供し、ObjectGrid、デプロイメント・ポリシー、およびクライアント ObjectGrid オーバーライド構成をカスタマイズできます。

REMOTE ObjectGrid の場合、キャッシュは動的 ObjectGrid を作成しません。キャッシュは、カタログ・サービスからクライアント・サイドの ObjectGrid を取得するだけです。hibernate-objectGrid-client-override.xml ファイルを提供して、クライアント ObjectGrid オーバーライド構成をカスタマイズできるだけです。

4. オプション: (リモート構成のみ) REMOTE ObjectGrid タイプのキャッシュを構成する場合、外部 eXtreme Scale システムをセットアップします。

REMOTE ObjectGrid タイプでキャッシュを構成する場合、外部 eXtreme Scale システムをセットアップする必要があります。外部システムをセットアップするためには、persistence.xml ファイルに基づく ObjectGrid および ObjectGridDeployment 構成 XML ファイルの両方が必要になります。これらの構成ファイルの例については、400 ページの『例: Hibernate ObjectGrid XML ファイル』を参照してください。

## タスクの結果

### EMBEDDED または EMBEDDED\_PARTITION 構成:

アプリケーションを開始すると、プラグインが自動的にカタログ・サービスを検出または開始し、コンテナ・サーバーを開始し、コンテナ・サーバーをカタログ・サービスに接続します。それから、プラグインは、クライアント接続を使用して別のアプリケーション・サーバー・プロセスで実行されている ObjectGrid コンテナおよびそのピアと通信します。

各 JPA エンティティには、エンティティのクラス名を使用して独立したバックアップ・マップが割り当てられます。各 BackingMap には、次の属性があります。

- readOnly="false"

- copyKey="false"
- lockStrategy="NONE"
- copyMode="NO\_COPY"

#### REMOTE 構成:

デプロイメント・ポリシーは JPA アプリケーションとは別に指定されます。外部 ObjectGrid システムには、カタログ・サービスおよびコンテナ・サーバー・プロセスの両方があります。コンテナ・サーバーを始動する前に、カタログ・サービスを始動する必要があります。詳しくは、451 ページの『スタンドアロン・サーバーの始動』と 454 ページの『コンテナ・サーバーの始動』を参照してください。

#### 次のタスク

- この構成を使用する Hibernate アプリケーションを開発します。詳しくは、例: ObjectGrid キャッシュにデータをプリロードするための Hibernate プラグインの使用を参照してください。
- 実稼働環境の場合、EMBEDDED または EMBEDDED\_PARTITION 構成用に自動的に作成されるプロセスのカタログ・サービス・ドメインを作成します。
  - スタンドアロン環境:

WebSphere Application Server プロセス内でサーバーを実行しない場合、カタログ・サービス・ドメインのホストおよびポートは、objectGridServer.properties というプロパティ・ファイルを使用して指定されます。このファイルは、アプリケーションのクラスパスに保管し、catalogServiceEndPoints プロパティを定義しておく必要があります。カタログ・サービス・ドメインは、アプリケーション・プロセスとは別に開始され、アプリケーション・プロセスが開始される前に開始されなければなりません。

objectGridServer.properties ファイルのフォーマットは次のとおりです。

```
catalogServiceEndPoints=<hostname1>:<port1>,<hostname2>:<port2>
```

- WebSphere Application Server 環境:

WebSphere Application Server プロセス内で実行する場合、JPA キャッシュ・プラグインでは、WebSphere Application Server セルに定義されたカタログ・サービスまたはカタログ・サービス・ドメインに自動的に接続します。

- EMBEDDED または EMBEDDED\_PARTITION ObjectGridType を Java SEJava SE 環境で使用する場合、組み込み eXtreme Scale サーバーを停止するため、プログラムの末尾に System.exit(0) メソッドを使用してください。そうしないと、プログラムが応答しなくなる可能性があります。

#### 例: Hibernate ObjectGrid XML ファイル:

Hibernate ObjectGrid XML ファイルは、パーシスタンス・ユニットの構成に基づいて作成する必要があります。

#### persistence.xml ファイル

パーシスタンス・ユニットの構成を表す persistence.xml ファイルの例を以下に示します。

```

<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>

    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistibleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>

    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
      <property name="hibernate.show_sql" value="false" />
      <property name="hibernate.connection.url" value="jdbc:db2:Annuity" />
      <property name="hibernate.connection.driver_class" value="com.ibm.db2.jcc.DB2Driver" />
      <property name="hibernate.default_schema" value="EJB30" />

      <!-- Cache -->
      <property name="hibernate.cache.provider_class"
        value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
      <property name="hibernate.cache.use_query_cache" value="true" />
      <property name="objectgrid.configuration" value="ObjectGridType=EMBEDDED,
        ObjectGridName=Annuity, MaxNumberOfReplicas=4" />
    </properties>
  </persistence-unit>
</persistence>

```

## hibernate-objectGridDeployment.xml ファイル

デプロイメント・ポリシーをカスタマイズする必要がある場合は、hibernate-objectGridDeployment.xml ファイルを使用します。このファイルをMETA-INF/hibernate-objectGridDeployment.xml ディレクトリに置くと、デフォルトのデプロイメント・ポリシーは、このファイルの中の構成によってオーバーライドされます。

```

<?xml version="1.0" encoding="UTF-8" ?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1" minSyncReplicas="0"
      maxSyncReplicas="4" maxAsyncReplicas="0" replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="defaultCacheMap" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="org.hibernate.cache.UpdateTimestampsCache" />
      <map ref="org.hibernate.cache.StandardQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>

```

## hibernate-objectGrid.xml ファイル

Java Persistence API (JPA) で Hibernate を使用していない場合、次のhibernate-objectGrid.xml の例を使用して、Hibernate 構成を作成します。

```

<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Annuity">
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <backingMap name="defaultCacheMap" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="defaultCacheMap" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"

```

```

        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
<backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
<backingMap name="org.hibernate.cache.UpdateTimestampsCache" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="org.hibernate.cache.UpdateTimestampsCache" />
<backingMap name="org.hibernate.cache.StandardQueryCache" readOnly="false" copyKey="false"
        lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
        pluginCollectionRef="org.hibernate.cache.StandardQueryCache" />
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="defaultCacheMap">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="org.hibernate.cache.UpdateTimestampsCache">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
<backingMapPluginCollection id="org.hibernate.cache.StandardQueryCache">
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" >
    </bean>
</backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>

```

注: org.hibernate.cache.UpdateTimestampsCache、  
org.hibernate.cache.StandardQueryCache および defaultCacheMap マップが必要です。

## Spring キャッシュ・プロバイダーの構成

Spring Framework バージョン 3.1 では、新しいキャッシュの抽象化が導入されています。この新しい抽象化を使用すると、既存の Spring アプリケーションに対してキャッシュをトランスペアレントに追加できます。キャッシュ抽象化のためのキャッシュ・プロバイダーとして WebSphere eXtreme Scale を使用できます。

### 始める前に

- Spring Framework バージョン 3.1 以降を使用するアプリケーションが必要です。

- ご使用のアプリケーションはアノテーションを使用してキャッシュのメソッドを宣言する必要があります。キャッシュ抽象化のためにご使用のアプリケーションを更新する方法について、詳しくは [Spring Framework Reference Documentation : Cache abstraction](#) を参照してください。
- `ogclient.jar` ファイルが Spring アプリケーションのクラスパスにあることを確認してください。
- アプリケーションを実行中の JVM が、WebSphere eXtreme Scale クライアントによってインストールされる JVM でない場合、次の JVM 引数を追加して IBM Object Request Broker (ORB) が使用されるようにする必要があります。  
`-Djava.endorsed.dirs=wxs_root/lib/endorsed`
- カタログ・サーバーを開始する必要があります。詳しくは、451 ページの『スタンドアロン・カタログ・サービスの開始』を参照してください。

## このタスクについて

Spring Framework でキャッシュ抽象化を使用することで、Java メソッドの実行回数を減らすことができます。構成後は、特定のメソッドの結果はキャッシュに入れられます。同じ引数を使用してメソッドを再実行すると、抽象化はキャッシュをチェックし、メソッドの結果が既にキャッシュに入っているかを確認します。結果がキャッシュに入っていれば、メソッドを再実行する必要はありません。抽象化を実装すると、コストがかかるメソッドの実行回数を減らすと共に、ご使用のアプリケーションの平均応答時間を短くすることができます。

## 手順

1. Spring 用の構成ファイルを使用するようにコンテナ・サーバーを構成します。

キャッシュにアクセスする Spring アプリケーションを開始する前に、コンテナ・サーバーを開始する必要があります。コンテナ・サーバーを開始するには、451 ページの『スタンドアロン・サーバーの始動』を参照してください。

eXtreme Scale Spring キャッシュ・プロバイダーのコンテナ・サーバーを開始するためのデフォルトの XML 構成ファイルは、以下のいずれかのロケーションにあります。

- スタンドアロン・インストールの場合: `wxs_install_root/ObjectGrid/spring/etc`
- WebSphere Application Server 上のインストールの場合: `was_root/optionalLibraries/ObjectGrid/spring/etc`

このファイルの名前は、`spring-remote-objectgrid.xml` および `spring-remote-deployment.xml` です。これらのファイルをそのまま使用することも、カスタマイズすることも、あるいは独自に構成ファイルを作成することもできます。

eXtreme Scale Spring キャッシュ・プロバイダーのスタンドアロン・コンテナ・サーバーを開始するには、以下のコマンドを実行します。以下のコマンド

は、`wxs_home/ObjectGrid/bin` ディレクトリーから実行します。 Windows

```
startOgServer.bat container1 -objectGridFile ../spring/etc/spring-remote-objectgrid.xml
-deploymentPolicyFile ../spring/etc/spring-remote-deployment.xml
```

```
startOgServer.sh container1 -objectGridFile ../spring/etc/spring-remote-objectgrid.xml
-deploymentPolicyFile ../spring/etc/spring-remote-deployment.xml
```

2. WebSphere eXtreme Scale をキャッシュ・プロバイダーとして使用するよう、Spring Inversion of Control (IoC) コンテナを構成します。WebSphere eXtreme Scale のキャッシュ実装環境は、`com.ibm.websphere.objectgrid.spring` パッケージ内にあります。Spring IoC コンテナ構成で以下の Bean を定義します。

```
<bean id="wxsCSDomain" class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean"
  p:catalog-service-endpoints="CATALOG_SERVICE_ENDPOINTS"
  p:client-override-xml="CLIENT_OVERRIDE_XML (optional)"
  p:client-security-config="CLIENT_SECURITY_CONFIG (optional)" />

<bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
  p:object-grid-name="OBJECT_GRID_NAME (optional)"
  p:catalog-service-domain-ref="wxsCSDomain" />

<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
  <property name="caches">
    <set>
      <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
        p:name="CACHE_NAME"
        p:map-name="MAP_NAME (optional)"
        p:object-grid-client-ref="wxsGridClient" />
    </set>
  </property>
</bean>
```

#### CATALOG\_SERVICE\_ENDPOINTS

オブジェクト・リクエスト・ブローカー (ORB) のホストおよびポート番号を指定します。

#### CLIENT\_OVERRIDE\_XML (オプション)

クライアント・サイドで設定を変更する ObjectGrid XML ファイルの絶対パスまたは相対パスを、Spring リソースとして指定します。Spring 内のリソースの指定について詳しくは、Spring Framework Reference Documentation: Resources を参照してください。

例: `p:client-override-xml="file:/path/to/objectgrid.xml"`

例: `p:client-override-xml="classpath:com/example/app/override-objectgrid.xml"`

例: `p:client-override-xml="http://myserver/override-objectgrid.xml"`

例: `p:client-override-xml="ftp://myserver/override-objectgrid.xml"`

#### CLIENT\_SECURITY\_CONFIG (オプション)

`client.properties` ファイルの絶対パスまたは相対パスを、Spring リソースとして指定します。Spring 内のリソースの指定について詳しくは、Spring Framework Reference Documentation: Resources を参照してください。

例: `p:client-security-config="file:/path/to/client.properties"`

#### OBJECT\_GRID\_NAME (オプション)

ObjectGrid 名を指定します。提供された XML 構成ファイルを使用してコンテナ・サーバーを開始する場合、このパラメーターは不要です。このパラメーターは、コンテナ・サーバーの開始に使用する XML 構成ファイルと整合している必要があります。



## CACHE\_NAME

Spring キャッシング・アプリケーションで指定されるキャッシュの名前を指定します。

## MAP\_NAME (オプション)

キャッシュのバックアップ・マップの名前を指定します。提供された XML 構成ファイルを使用してコンテナ・サーバーを開始する場合、このパラメーターは不要です。このパラメーターは、コンテナ・サーバーの開始に使用する XML 構成ファイルと整合している必要があります。提供された XML 構成ファイルを使用する場合は、MAP\_NAME 値は必要ありません。データ・グリッドのマップは、Spring アプリケーションが実行されると自動的に作成されます。動的マップの名前は、IBM\_SPRING\_PARTITIONED\_ で始まります。例:  
IBM\_SPRING\_PARTITIONED\_1、IBM\_SPRING\_PARTITIONED\_2、などです。

## 例

以下のスニペットは、localhost:2809 でカタログ・サービス・ドメインがホスティングする、default と books という名前の 2 つのキャッシュを作成します。

```
<bean id="wxsCSDomain" class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean"
  p:catalog-service-endpoints="localhost:2809" />
<bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
  p:catalog-service-domain-ref="wxsCSDomain" />
<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
  <property name="caches">
    <set>
      <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
        p:name="default"
        p:object-grid-client-ref="wxsGridClient" />
      <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
        p:name="books"
        p:object-grid-client-ref="wxsGridClient" />
    </set>
  </property>
</bean>
```

---

## データベース統合の構成

WebSphere eXtreme Scale を使用すると、データベースの負荷を軽減できます。WebSphere eXtreme Scale とデータベース間には Java Persistence API (JPA) を使用でき、変更をローダーとして統合できます。

### 始める前に

データベースで作成できるさまざまなトポロジーの要約については、19 ページの『データベース統合: 後書き、インライン、およびサイド・キャッシング』を参照してください。

## JPA ローダーの構成

Java Persistence API (JPA) ローダーは、JPA を使用してデータベースと対話するプラグイン実装です。

### 始める前に

- Hibernate または OpenJPA などの JPA 実装が必要です。
- データベースには、選択された JPA プロバイダーがサポートする任意のバックエンドを使用できます。

- JPALoader プラグインと JPAEntityLoader プラグインのどちらを使用するか決定します。ObjectMap API を使用してデータを保管する場合、JPALoader プラグインを使用します。EntityManager API を使用してデータを保管する場合、JPAEntityLoader プラグインを使用します。

注: JPA API を使用して JPA データ・ソースにアクセスする場合は、JPA L2 キャッシュ・プラグインを使用してください。キャッシュ・プラグインは、JPA アプリケーションを使用したままでも、アプリケーションと JPA データ・ソースの間にデータ・グリッドを導入できます。詳しくは、378 ページの『JPA レベル 2 (L2) キャッシュ・プラグイン』を参照してください。

## このタスクについて

Java Persistence API (JPA) Loader の機能について詳しくは、JPA ローダーを参照してください。

### 手順

1. データベースと対話するために JPA が必要とする必須パラメーターを構成します。

次のパラメーターが必要です。これらのパラメーターは、JPALoader または JPAEntityLoader Bean、および JPATxCallback Bean 内で構成されます。

- **persistenceUnitName:** パーシスタンス・ユニット名を指定します。このパラメーターは、JPA EntityManagerFactory の作成、および persistence.xml ファイル内の JPA エンティティ・メタデータの検索という 2 つの目的のために必要です。この属性は、JPATxCallback に設定されます。
- **JPAPropertyFactory:** パーシスタンス・プロパティ・マップを作成し、デフォルトのパーシスタンス・プロパティをオーバーライドするためのファクトリーを指定します。この属性は、JPATxCallback に設定されます。この属性を設定するために、Spring スタイルの構成が必要です。
- **entityClassName:** JPA メソッド (EntityManager.persist、EntityManager.find など) を使用する場合に必要エンティティ・クラス名を指定します。JPALoader にはこのパラメーターが必要ですが、JPAEntityLoader ではこのパラメーターはオプションです。JPAEntityLoader プラグインの場合、**entityClassName** パラメーターが構成されていないと、ObjectGrid エンティティ・マップに構成されているエンティティ・クラスが使用されます。eXtreme Scale EntityManager と JPA プロバイダーに同じクラスを使用する必要があります。この属性は、JPALoader または JPAEntityLoader に設定されます。
- **preloadPartition:** マップ・プリロードが開始される区画を指定します。プリロード区画がゼロより小さいか、または「区画総数マイナス 1」よりも大きい場合、マップ・プリロードは開始されません。デフォルト値は -1 ですから、デフォルトではプリロードは開始されません。この属性は、JPALoader または JPAEntityLoader に設定されます。

この 4 つの JPA パラメーターが eXtreme Scale に構成されるほか、JPA エンティティからキーを取得するため、JPA メタデータが使用されます。JPA メ

タデータは、アノテーションとして構成するか、 persistence.xml ファイル内に指定される orm.xml ファイルとして構成できます。これは、eXtreme Scale 構成には含まれません。

## 2. JPA 構成用に XML ファイルを構成します。

JPALoader または JPAEntityLoader を構成する場合は、データベースとの通信のためのプラグインを参照してください。

ローダー構成と一緒に、JPATxCallback トランザクション・コールバックを構成します。以下に、JPAEntityLoader と JPATxCallback が構成されている ObjectGrid XML 記述子ファイル (objectgrid.xml) の例を示します。

### コールバックを含むローダーの構成 - XML の例

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="JPAEM" entityMetadataXMLFile="jpaEMD.xml">
      <bean id="TransactionCallback"
        className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
        <property
          name="persistenceUnitName"
          type="java.lang.String"
          value="employeeEMPU" />
        </bean>
      <backingMap name="Employee" pluginCollectionRef="Employee" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="Employee">
      <bean id="Loader"
        className="com.ibm.websphere.objectgrid.jpa.JPAEntityLoader">
      <property
        name="entityClassName"
        type="java.lang.String"
        value="com.ibm.ws.objectgrid.jpa.test.entity.Employee"/>
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

JPAPropertyFactory を構成する場合は、Spring スタイルの構成を使用する必要があります。以下に示すのは、Spring Bean が eXtreme Scale 構成に使用されるように構成している XML 構成ファイル・サンプル例 (JPAEM\_spring.xml) です。

### JPA プロパティ・ファクトリーを含むローダーの構成 - XML の例

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

  <objectgrid:JPAEntityLoader id="jpaLoader"
    entityClassName="com.ibm.ws.objectgrid.jpa.test.entity.Employee"/>
  <objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU" />
</beans>
```

次に Objectgrid.xml 構成 XML ファイルを示します。ObjectGrid の名前は JPAEM であり、これは、JPAEM\_spring.xml Spring 構成ファイルの ObjectGrid 名に一致しています。

#### JPAEM ローダー構成 - XML の例

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="JPAEM" entityMetadataXMLFile="jpaEMD.xml">
      <bean id="TransactionCallback"
        className="{spring}jpaTxCallback"/>
      <backingMap name="Employee" pluginCollectionRef="Employee"
        writeBehind="T4"/>
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="Employee">
      <bean id="Loader" className="{spring}jpaLoader" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

エンティティには、JPA アノテーションおよび eXtreme Scale エンティティ・マネージャー・アノテーションの両方でアノテーションを付けることができます。各アノテーションには、使用可能な等価 XML があります。そのため、eXtreme Scale は Spring 名前空間を追加しています。この Spring 名前空間サポートを使用してこれらを構成することもできます。詳しくは、Spring Framework の概要を参照してください。

## JPA 時間ベース・データ・アップデーターの構成

時間ベース・データベース更新は、ローカルまたは分散 eXtreme Scale 構成の場合、XML を使用して構成することができます。ローカル構成はプログラムでも構成できます。

### このタスクについて

Java Persistence API (JPA) 時間ベース・データ・アップデーターがどのように機能するかについて詳しくは、JPA 時間ベース・データ・アップデーターを参照してください。

### 手順

timeBasedDBUpdate 構成を作成します。

#### • XML ファイルを使用:

次に示すのは、timeBasedDBUpdate 構成を含む objectgrid.xml ファイルの例です。

#### JPA 時間ベース・アップデーター - XML の例

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="changeOG"
      entityMetadataXMLFile="userEMD.xml">
      <backingMap name="user" >
        <timeBasedDBUpdate timestampField="rowChgTs"
          persistenceUnitName="userderby"
          entityClass="com.test.UserClass"
          mode="INVALIDATE_ONLY"
        />
      </backingMap>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

```
        </backingMap>
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
</objectGridConfig>
```

この例では、マップ "user" が、時間ベース・データベース更新で構成されています。データベースの更新モードは、INVALIDATE\_ONLY、タイム・スタンプ・フィールドは、rowChgTs です。

分散 ObjectGrid "changeOG" が、コンテナ・サーバーで開始されると、時間ベース・データベース更新スレッドが、区画 0 で自動的に始動されます。

- **プログラムで:**

ローカル ObjectGrid を作成する場合、TimeBasedDBUpdateConfig オブジェクトを作成し、これを BackingMap インスタンスに設定することもできます。

```
public void setTimeBasedDBUpdateConfig(TimeBasedDBUpdateConfig dbUpdateConfig);
```

BackingMap インスタンスへのオブジェクトの設定に関して詳しくは、API 資料にある BackingMap インターフェースに関する情報を参照してください。

また、com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp アノテーションを使用して、エンティティ・クラスのタイム・スタンプ・フィールドにアノテーションを付けることができます。このクラスの値を構成すれば、XML 構成の timestampField を構成する必要はありません。

## 次のタスク

JPA 時間ベース・データ・アップデーターを開始します。詳しくは、JPA 時間ベース・アップデーターの開始を参照してください。

---

## REST データ・サービスの構成

WebSphere eXtreme Scale REST データ・サービスは、WebSphere Application Server バージョン 7.0、WebSphere Application Server Community Edition、および Apache Tomcat で使用できます。

### このタスクについて

付属のサンプルには、区画化されたデータ・グリッドを実行するためのソース・コードおよびコンパイルされたバイナリーが含まれています。このサンプルでは、単純データ・グリッドを作成する方法と、エンティティを使用してデータをモデル化する方法を示します。また、Java または C# を使用してエンティティの追加および照会を可能にする 2 つのコマンド行クライアント・アプリケーションを提供します。

サンプル Java クライアントは、Java EntityManager API を使用して、データ・グリッド内のデータを永続化および照会します。このクライアントは、Eclipse またはコマンド行スクリプトを使用して実行できます。なお、サンプル Java クライアントでは REST データ・サービスについては説明されませんが、グリッド内のデータの更新は可能であるため、Web ブラウザーなどのクライアントでデータを読み取ることができます。

サンプル Microsoft WCF Data Services C# クライアントは、.NET Framework を使用した REST データ・サービスを介して、eXtreme Scale データ・グリッドと通信します。WCF Data Services クライアントは、データ・グリッドを更新および照会するために使用できます。

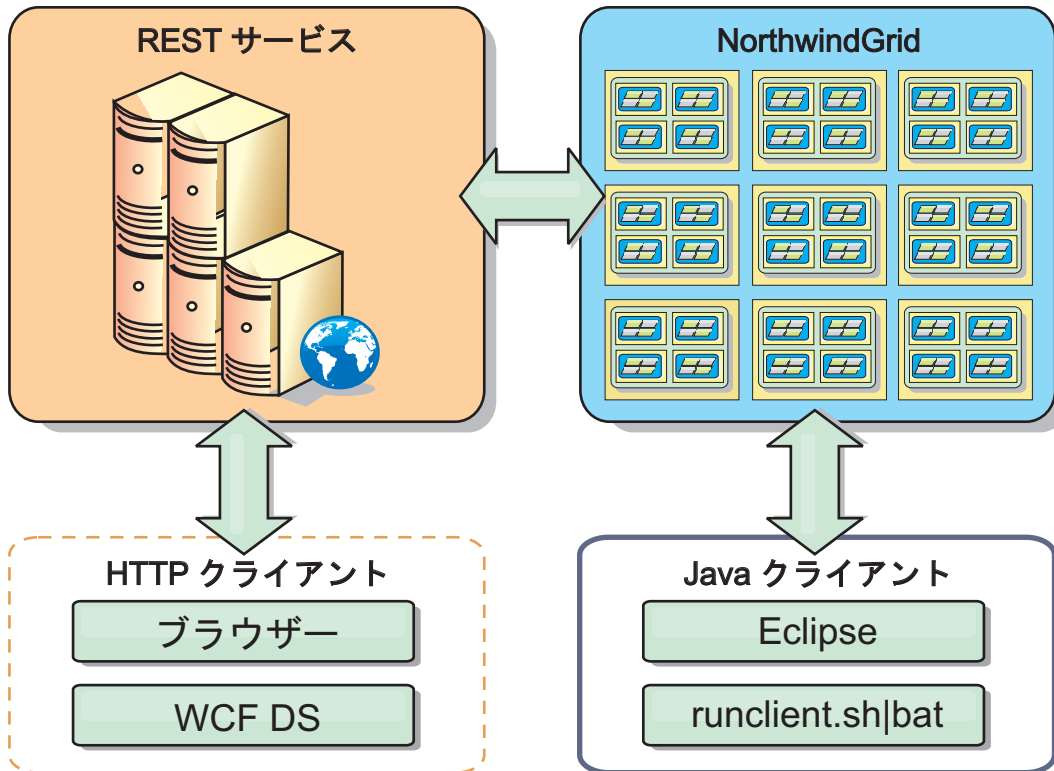


図 44. 開始用 (getting started) サンプル・トポロジー: REST データ・サービスを使用する HTTP クライアントと Java クライアントは同じデータ・グリッドにアクセスできます。

## 手順

1. eXtreme Scale データ・グリッドを構成および開始します。 411 ページの『REST データ・サービスの使用可能化』を参照してください。
2. Web ブラウザーで REST データ・サービスを構成および開始します。 420 ページの『REST データ・サービス用のアプリケーション・サーバーの構成』を参照してください。
3. クライアントを実行して、REST データ・サービスと対話します。以下の 2 つのオプションを使用できます。
  - a. サンプル Java クライアントを実行して、EntityManager API でグリッドにデータを追加し、Web ブラウザーおよび eXtreme Scale REST データ・サービスでグリッド内のデータを照会します。 440 ページの『REST データ・サービスでの Java クライアントの使用』を参照してください。
  - b. サンプル WCF Data Services C# クライアントを実行します。 442 ページの『REST データ・サービスでの Visual Studio 2008 WCF クライアント』を参照してください。



## REST データ・サービスの使用可能化

REST データ・サービスは、WebSphere eXtreme Scale エンティティ・メタデータを表し、各エンティティを EntitySet として表すことができます。

### サンプル eXtreme Scale データ・グリッドの開始

一般的に、REST データ・サービスを開始する前に、eXtreme Scale データ・グリッドを開始します。以下のステップで、1 つの eXtreme Scale カタログ・サービス・プロセスおよび 2 つのコンテナ・サーバー・プロセスを開始します。

WebSphere eXtreme Scale は、3 つの異なる方法を使用してインストールできます。

- 試用インストール
- スタンドアロン・デプロイメント
- WebSphere Application Server 統合デプロイメント

### eXtreme Scale のスケーラブル・データ・モデル

Microsoft Northwind サンプルは、Order Detail 表を使用して多対多のアソシエーションを Order と Product の間で確立します。

ADO.NET Entity Framework および Java Persistence API (JPA) などのオブジェクト・リレーショナル・マッピング仕様 (ORMs) は、エンティティを使用する表やリレーションシップをマップすることができます。ただし、このアーキテクチャーは拡張されません。すべてが同一マシン上、あるいはうまく機能するような高価なマシンのクラスター上に存在していなければなりません。

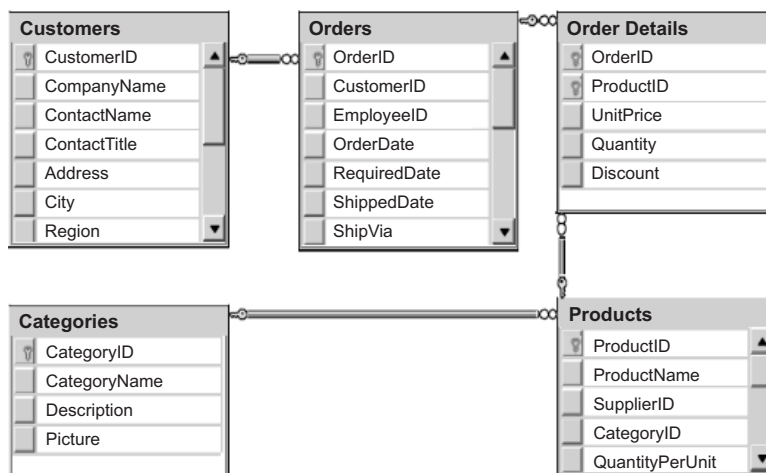


図 45. Microsoft SQL Server Northwind サンプルのスキーマ図

拡張が容易な種類のサンプルを作成するためには、各エンティティまたは関連エンティティのグループが単一キーを基にして区画に分割できるように、エンティティをモデル化する必要があります。単一キーに基づいて区画を作成することで、複数の独立したサーバーに要求を分散させることができます。この構成を実現させるために、エンティティを 2 つのツリーに分割しました。Customer および Order のツリーと、Product および Category のツリーです。このモデルでは、各ツリーは個別に区画に分割することができるため、異なる速度で、スケーラビリティ

ーを高めながら成長できます。

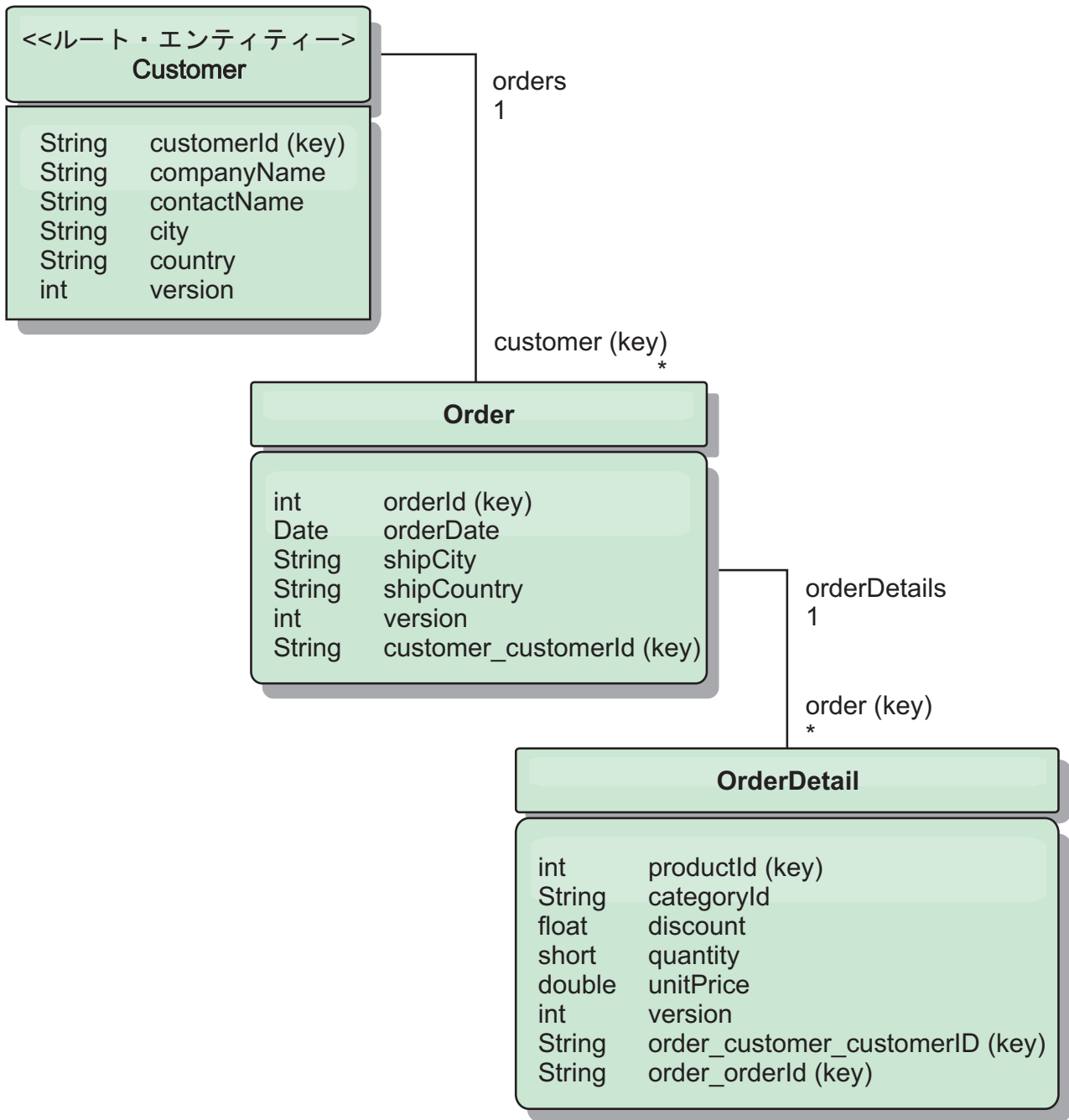


図 46. Customer および Order エンティティのスキーマ図

例えば、Order と Product はいずれも固有な、別の整数をキーとして持っています。つまり、Order 表と Product 表は本当にお互いに独立しています。例えば、カタログのサイズや販売する製品数の、オーダー総数への影響を考えてみます。直感的に、多くの製品を持てば多くのオーダーを受けることを意味するようにも思えますが、これは必ずしもそうとは限りません。これが真実であれば、カタログにより多くの製品を追加するだけで、簡単に売り上げを伸ばすことができるでしょう。オーダーと製品には、それぞれ独自の独立した表があります。オーダーと製品がそれぞれ独自に別々のデータ・グリッドを持つように、この概念をさらに拡張するこ

とができます。独立したデータ・グリッドを使用すると、アプリケーションが拡張できるように、各データ・グリッドの個別のサイズの他に、区画数およびサーバー数を制御することができます。カタログのサイズを 2 倍にすると、製品グリッドを 2 倍にする必要がありますが、オーダー・データ・グリッドはおそらく変わりません。オーダーの急増、または予測されるオーダーの急増に関しては、その逆が真実となります。

スキーマでは、Customer にはゼロ以上の Order があり、Order は 1 つの特定製品それぞれに明細行 (OrderDetail) があります。Product は、各 OrderDetail で ID (製品キー) によって識別されます。単一データ・グリッドは、Customer をデータ・グリッドのルート・エンティティとして使用し、Customer、Order、および OrderDetails を保管します。Customer を ID で取得することができますが、Customer ID から始めて Order を取得しなければなりません。そのため、Customer ID は Order にそのキーの一部として追加されます。同様に、カスタマー ID およびオーダー ID は OrderDetail ID の一部です。

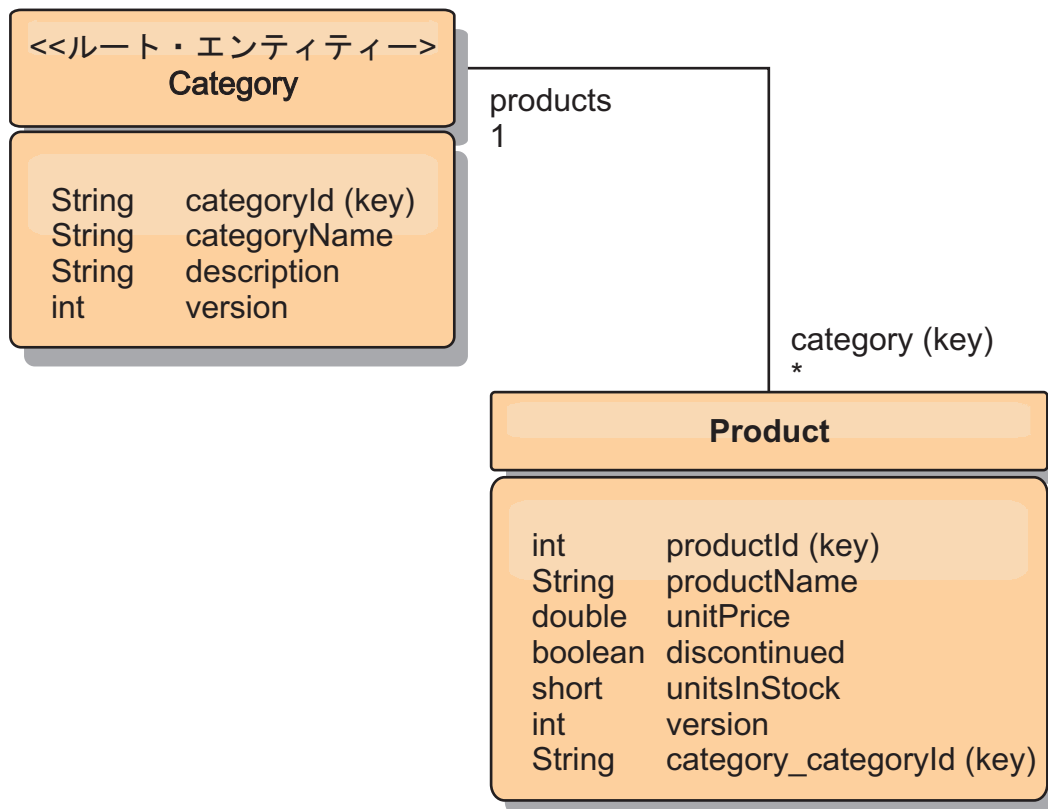


図 47. Category および Product エンティティのスキーマ図

Category および Product スキーマでは、Category がスキーマ・ルートです。このスキーマにより、カスタマーはカテゴリーごとに製品を照会することができます。キー・アソシエーションおよびその重要性のさらに詳細な情報については、414 ページの『REST を使用したデータの取得および更新』を参照してください。

## REST を使用したデータの取得および更新

OData プロトコルは、すべてのエンティティが正規化形式でアドレス指定できることを要求します。つまり、各エンティティは区画に分割されたルート・エンティティ、スキーマ・ルートのキーを含んでいなければなりません。

以下は、子エンティティをアドレス指定するためのルート・エンティティからのアソシエーションの使用法の例です。

```
/Customer('ACME')/order(100)
```

WCF Data Services では、子エンティティは直接アドレス可能でなければなりません。つまり、スキーマ・ルートのキーは、次のように子のキーの一部でなければなりません。/Order(customer\_customerId='ACME', orderId=100) これは、ルート・エンティティへのアソシエーションの作成により実現され、ルート・エンティティへの 1 対 1 または多対 1 のアソシエーションもキーとして識別されます。エンティティがキーの一部として組み込まれる場合、親エンティティの属性はキー・プロパティとして公開されます。

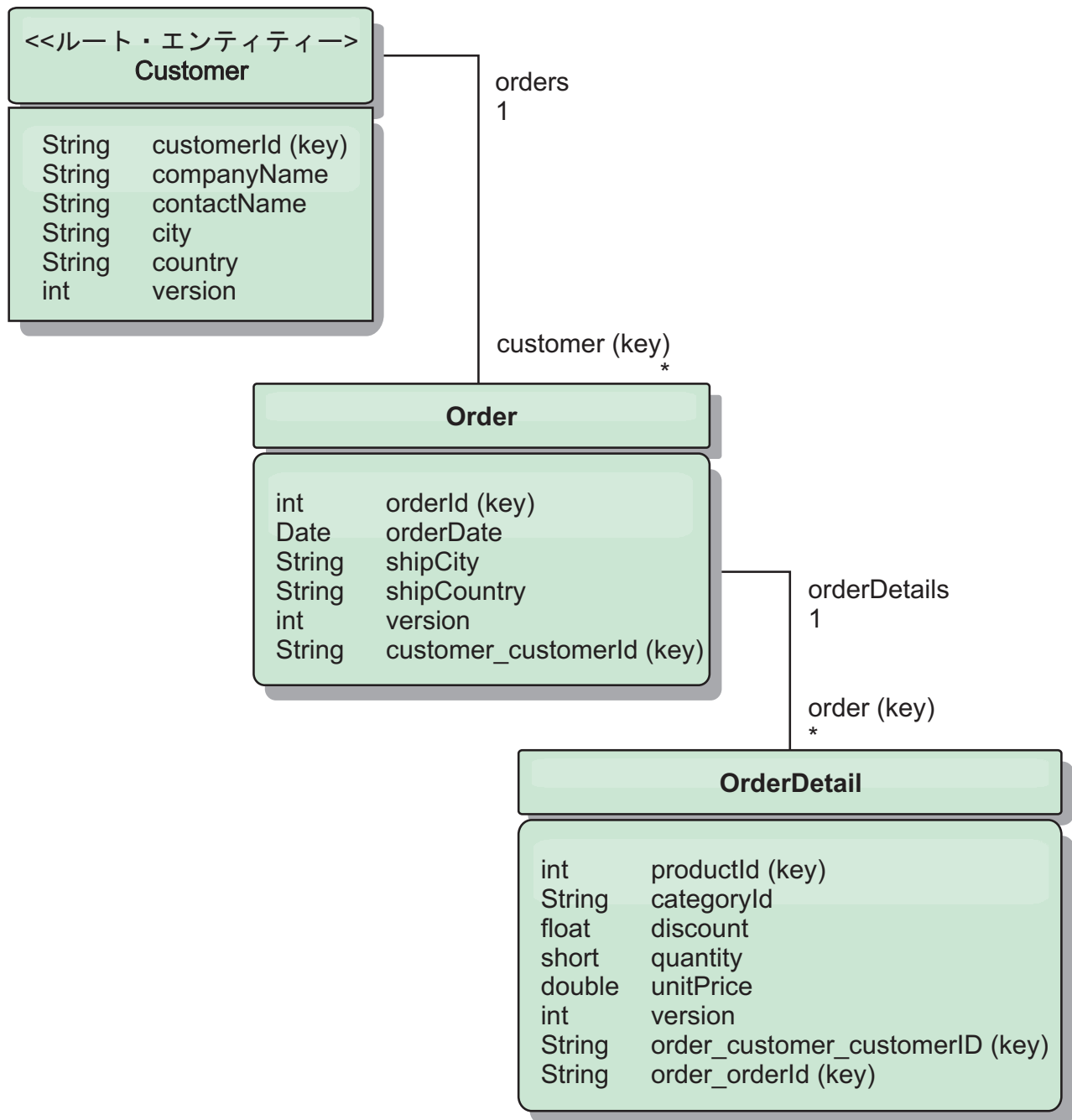


図 48. Customer および Order エンティティのスキーマ図

Customer/Order エンティティのスキーマ図で、どのように各エンティティが Customer を使用して区画に分割されているかを示しています。Order エンティティにはキーの一部として Customer が組み込まれるため、直接アクセスすることができます。REST データ・サービスは、すべてのキー・アソシエーションを個別のプロパティとして公開します。Order には customer\_customerId があり、OrderDetail には order\_customer\_customerId および order\_orderId があります。

EntityManager API を使用すると、Customer とオーダー ID を使用して Order を検索することができます。

```

transaction.begin();
// Look-up the Order using the Customer. We only include the Id
// in the Customer class when building the OrderId key instance.
Order order = (Order) em.find(Order.class,
    new OrderId(100, new Customer('ACME')));
...
transaction.commit();

```

REST データ・サービスを使用する場合、どちらかの URL を使用して Order を取得することができます。

- /Order(orderId=100, customer\_customerId='ACME')
- /Customer('ACME')/orders?\$filter=orderId eq 100

カスタマー・キーは Customer エンティティの属性名 (下線文字とカスタマー ID の属性名 customer\_customerId) を使用してアドレス指定されています。

エンティティには、非ルート・エンティティのすべての上位エンティティがルートへのキー・アソシエーションを持つ場合は、キーの一部として非ルート・エンティティを組み込むこともできます。この例では、OrderDetail には Order へのキー・アソシエーションがあり、Order にはルートの Customer エンティティへのキー・アソシエーションがあります。EntityManager API の使用は、次のとおりです。

```

transaction.begin();
// Construct an OrderDetailId key instance. It includes
// The Order and Customer with only the keys set.
Customer customerACME = new Customer("ACME");
Order order100 = new Order(100, customerACME);
OrderDetailId orderDetailKey =
    new OrderDetailId(order100, "COMP");
OrderDetail orderDetail = (OrderDetail)
    em.find(OrderDetail.class, orderDetailKey);
...

```

REST データ・サービスは、OrderDetail に直接アドレスを指定することができます。

```

/OrderDetail(productId=500, order_customer_customerId='ACME', order_orderId =100)

```

OrderDetail エンティティから Product エンティティへのアソシエーションは分割され、Orders および Product インベントリを個別に区画に分割することができます。OrderDetail エンティティは、強いリレーションシップの代わりにカテゴリと製品 ID を保管します。2 つのエンティティ・スキーマを切り離すと、一度に 1 つの区画だけがアクセスされます。

Category および Product スキーマは、図で示すように、ルート・エンティティが Category で、各 Product には Category エンティティへのアソシエーションがあることを表しています。Category エンティティは Product ID に組み込まれています。REST データ・サービスは、キー・プロパティを公開します。

category\_categoryId で Product に直接アドレスを指定できます。

Category はルート・エンティティなので、区画に分割された環境で Product を検索するには、Category が認識されている必要があります。EntityManager API を使用すると、Product の検索前にトランザクションは Category エンティティに pinned されなければなりません。



EntityManager API の使用は、次のとおりです。

```
transaction.begin();
// Create the Category root entity with only the key. This
// allows us to construct a ProductId without needing to find
// The Category first. The transaction is now pinned to the
// partition where Category "COMP" is stored.
Category cat = new Category("COMP");
Product product = (Product) em.find(Product.class,
    new ProductId(500, cat));
...
```

REST データ・サービスは、Product に直接アドレスを指定することができます。

```
/Product(productId=500, category_categoryId='COMP')
```

## REST データ・サービスのスタンドアロン・データ・グリッドの開始

以下のステップに従って、スタンドアロン eXtreme Scale デプロイメントの WebSphere eXtreme Scale REST サービス・サンプル・データ・グリッドを開始します。

### 始める前に

以下のように、WebSphere eXtreme Scale の試用版または完全な製品をインストールします。

- スタンドアロン・バージョンの製品をインストールして、後続フィックスがある場合にはすべて適用します。
- WebSphere eXtreme Scale REST データ・サービスが含まれている WebSphere eXtreme Scale バージョン 7.1 以上の試用版をダウンロードし、解凍します。

### このタスクについて

WebSphere eXtreme Scale サンプル・データ・グリッドを開始します。

### 手順

1. カタログ・サービス・プロセスを開始します。コマンド行または端末ウィンドウを開いて、以下のように、JAVA\_HOME 環境変数を設定します。

- `Linux` `UNIX` `export JAVA_HOME=java_home`

- `Windows` `set JAVA_HOME=java_home`

2. `cd restservice_home/gettingstarted`

3. カタログ・サービス・プロセスを開始します。eXtreme Scale セキュリティーなしでサービスを開始するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcat.sh`

- `Windows` `runcat.bat`

eXtreme Scale セキュリティーありでサービスを開始するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcat_secure.sh`

- `Windows` `runcat_secure.bat`

4. 以下のように、2 つのコンテナ・サーバー・プロセスを開始します。もう 1 つコマンド行または端末ウィンドウを開いて、以下のように、JAVA\_HOME 環境変数を設定します。

- **Linux** **UNIX** `export JAVA_HOME=java_home`
- **Windows** `set JAVA_HOME=java_home`

5. `cd restservice_home/gettingstarted`

6. 以下のように、コンテナ・サーバー・プロセスを開始します。

eXtreme Scale セキュリティーなしでサーバーを始動するには、以下のコマンドを使用します。

- **Linux** **UNIX** `./runcontainer.sh container0`
- **Windows** `runcontainer.bat container0`

eXtreme Scale セキュリティーありでサーバーを始動するには、以下のコマンドを使用します。

- **Linux** **UNIX** `./runcontainer_secure.sh container0`
- **Windows** `runcontainer_secure.bat container0`

7. もう 1 つコマンド行または端末ウィンドウを開いて、以下のように、JAVA\_HOME 環境変数を設定します。

- **Linux** **UNIX** `export JAVA_HOME=java_home`
- **Windows** `set JAVA_HOME=java_home`

8. `cd restservice_home/gettingstarted`

9. 以下のように、2 番目のコンテナ・サーバー・プロセスを開始します。

eXtreme Scale セキュリティーなしでサーバーを始動するには、以下のコマンドを使用します。

- **Linux** **UNIX** `./runcontainer.sh container1`
- **Windows** `runcontainer.bat container1`

eXtreme Scale セキュリティーありでサーバーを始動するには、以下のコマンドを使用します。

- **Linux** **UNIX** `./runcontainer_secure.sh container1`
- **Windows** `runcontainer_secure.bat container1`

## タスクの結果

eXtreme Scale コンテナの準備ができるまで待機してから、次のステップに進みます。以下のメッセージが端末ウィンドウに表示されると、コンテナ・サーバーの準備ができています。

```
CWOBJ1001I: ObjectGrid サーバー container_name は要求を処理する用意ができています。
```

ここで、`container_name` は、開始したコンテナの名前です。

## WebSphere Application Server での REST データ・サービスのデータ・グリッドの開始

以下のステップに従って、WebSphere Application Server に統合された WebSphere eXtreme Scale デプロイメントのスタンドアロン WebSphere eXtreme Scale REST サービス・サンプル・データ・グリッドを開始します。WebSphere eXtreme Scale は WebSphere Application Server に統合されていますが、以下のステップでは、スタンドアロン WebSphere eXtreme Scale カタログ・サービス・プロセスおよびコンテナが開始されます。

### 始める前に

セキュリティーが使用不可に設定された WebSphere Application Server バージョン 7.0.0.5 以上のインストール・ディレクトリーに製品をインストールします。少なくとも 1 つのアプリケーション・サーバー・プロファイルを拡張します。

### このタスクについて

WebSphere eXtreme Scale サンプル・データ・グリッドを開始します。

### 手順

1. カタログ・サービス・プロセスを開始します。コマンド行または端末ウィンドウを開いて、以下のように、`JAVA_HOME` 環境変数を設定します。

- `Linux` `UNIX` `export JAVA_HOME=java_home`
- `Windows` `set JAVA_HOME=java_home`

```
cd restservice_home/gettingstarted
```

2. カタログ・サービス・プロセスを開始します。

eXtreme Scale セキュリティーなしでサーバーを始動するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcat.sh`
- `Windows` `runcat.bat`

eXtreme Scale セキュリティーありでサーバーを始動するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcat_secure.sh`
- `Windows` `runcat_secure.bat`

3. 以下のように、2 つのコンテナ・サーバー・プロセスを開始します。もう 1 つコマンド行または端末ウィンドウを開いて、以下のように、`JAVA_HOME` 環境変数を設定します。

- `Linux` `UNIX` `export JAVA_HOME=java_home`
- `Windows` `set JAVA_HOME=java_home`

4. 以下のように、コンテナ・サーバー・プロセスを開始します。

eXtreme Scale セキュリティーなしでサーバーを始動するには、以下のコマンドを使用します。

- a. コマンド行ウィンドウを開きます。
- b. `cd restservice_home/gettingstarted`
- c. eXtreme Scale セキュリティーなしでサーバーを始動するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcontainer.sh container0`

- `Windows` `runcontainer.bat container0`

- d. eXtreme Scale セキュリティーありでサーバーを始動するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcontainer_secure.sh container0`

- `Windows` `runcontainer_secure.bat container0`

5. 以下のように、2 番目のコンテナ・サーバー・プロセスを開始します。

- a. コマンド行ウィンドウを開きます。
- b. `cd restservice_home/gettingstarted`
- c. eXtreme Scale セキュリティーなしでサーバーを始動するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcontainer.sh container1`

- `Windows` `runcontainer.bat container1`

- d. eXtreme Scale セキュリティーありでサーバーを始動するには、以下のコマンドを使用します。

- `Linux` `UNIX` `./runcontainer_secure.sh container1`

- `Windows` `runcontainer_secure.bat container1`

## タスクの結果

コンテナ・サーバーの準備ができるまで待機してから、次のステップに進みます。以下のメッセージが表示されると、コンテナ・サーバーの準備ができています。

```
CWOBJ1001I: ObjectGrid サーバー container_name は要求を処理する用意ができています。
```

ここで、*container\_name* は前のステップで開始したコンテナの名前です。

## REST データ・サービス用のアプリケーション・サーバーの構成

さまざまなアプリケーション・サーバーを構成して、REST データ・サービスを使用できます。

## WebSphere Application Server への REST データ・サービスのデプロイ

このトピックでは、WebSphere Application Server または WebSphere Application Server Network Deployment バージョン 6.1.0.25 以上で WebSphere eXtreme Scale REST データ・サービスを構成する方法について説明します。これらの説明は、WebSphere eXtreme Scale が WebSphere Application Server デプロイメントに統合されるデプロイメントにも適用されます。

### 始める前に

WebSphere eXtreme Scale の REST データ・サービスを構成およびデプロイするには、システムに以下の環境のいずれかが必要です。

- スタンドアロン WebSphere eXtreme Scale クライアントを備えた WebSphere Application Server:
  - REST データ・サービスが付属した WebSphere eXtreme Scale 試用版バージョン 7.1 をダウンロードして解凍するか、累積フィックス 2 を適用した WebSphere eXtreme Scale バージョン 7.1.0.0 製品をスタンドアロン・ディレクトリにインストールします。
  - WebSphere Application Server バージョン 6.1.0.25 または 7.0.0.5 以上をインストールして実行します。
- WebSphere eXtreme Scale と統合された WebSphere Application Server。

累積フィックス 2 以上を適用した WebSphere eXtreme Scale バージョン 7.1.0.0 を WebSphere Application Server バージョン 6.1.0.25 または 7.0 以上の上にインストールします。

**ヒント:** WebSphere eXtreme Scale REST データ・サービスには、WebSphere eXtreme Scale クライアント・オプションのインストールのみが必要です。プロファイルを拡張する必要はありません。

WebSphere Application Server インフォメーション・センターで、Java 2 セキュリティを使用可能にする方法について参照してください。

### 手順

1. データ・グリッドを構成および開始します。
  - a. REST データ・サービスとともに使用するためのデータ・グリッドの構成の詳細については、419 ページの『WebSphere Application Server での REST データ・サービスのデータ・グリッドの開始』を参照してください。
  - b. クライアントがデータ・グリッド内のエンティティに接続およびアクセスできることを検査します。例えば、1 ページの『チュートリアル: WebSphere eXtreme Scale 入門』を参照してください。
2. eXtreme Scale REST サービス構成 JAR またはディレクトリをビルドします。200 ページの『REST データ・サービスのインストール』の REST サービスのパッケージ化およびデプロイに関する情報を参照してください。
3. REST データ・サービス構成 JAR またはディレクトリをアプリケーション・サーバー・クラスパスに追加します。
  - a. WebSphere Application Server 管理コンソールを開きます。

- b. 「環境」 > 「共有ライブラリー」にナビゲートします。
  - c. 「新規」をクリックします。
  - d. 以下の項目を該当するフィールドに追加します。
    - 名前: extremescale\_rest\_configuration
    - クラスパス: <REST サービス構成 JAR またはディレクトリー>
  - e. 「OK」をクリックします。
  - f. 変更をマスター構成に保存します。
4. 以下のように、WebSphere eXtreme Scale クライアント・ランタイム JAR の wsogclient.jar および REST データ・サービス構成 JAR またはディレクトリーをアプリケーション・サーバー・クラスパスに追加します。WebSphere eXtreme Scale が WebSphere Application Server インストール済み環境と統合されている場合、このステップは必要ありません。
    - a. WebSphere Application Server 管理コンソールを開きます。
    - b. 「環境」 > 「共有ライブラリー」にナビゲートします。
    - c. 「新規」をクリックします。
    - d. 以下の項目をフィールドに追加します。
      - 名前: extremescale\_client\_v71
      - クラスパス: wxs\_home/lib/wsogclient.jar

**要確認:** それぞれのパスを別個の行に追加します。

    - e. 「OK」をクリックします。
    - f. 変更をマスター構成に保存します。
  5. 管理コンソールを使用して、REST データ・サービスの EAR ファイル wxsrestservice.ear を WebSphere Application Server にインストールします。
    - a. WebSphere Application Server 管理コンソールを開きます。
    - b. 「アプリケーション」 > 「新規アプリケーション」をクリックします。
    - c. ファイル・システム上の /lib/wxsrestservice.ear ファイルを参照して選択し、「次へ」をクリックします。
      - WebSphere Application Server バージョン 7.0 を使用している場合は、「次へ」をクリックします。
      - WebSphere Application Server バージョン 6.1 を使用している場合は、名前 /wxsrestservice でコンテキスト・ルート値を入力して、次のステップに進みます。
    - d. 詳細インストール・オプションを選択して、「次へ」をクリックします。
    - e. アプリケーション・セキュリティー警告画面で、「続行」をクリックします。
    - f. デフォルト・インストール・オプションを選択して、「次へ」をクリックします。
    - g. アプリケーションのマップ先サーバーを選択して、「次へ」をクリックします。
    - h. JSP 再ロード・ページで、デフォルトを使用し、「次へ」をクリックします。



- i. 共有ライブラリー・ページで、wxsrestservice.war モジュールを、以下の定義済み共有ライブラリーにマップします。
  - extremescale\_rest\_configuration
  - extremescale\_client\_v71

**ヒント:** この共有ライブラリーは、WebSphere eXtreme Scale が WebSphere Application Server と統合されていない場合にのみ必要です。

- j. マップ共有ライブラリー・リレーションシップ・ページで、デフォルトを使用し、「次へ」をクリックします。
  - k. マップ仮想ホスト・ページで、デフォルトを使用し、「次へ」をクリックします。
  - l. マップ・コンテキスト・ルート・ページで、コンテキスト・ルートを wxsrestservice に設定します。
  - m. 要約画面で、「終了」をクリックして、インストールを完了します。
  - n. 変更をマスター構成に保存します。
6. wxsrestservice REST データ・サービス・アプリケーションを開始します。
    - a. 管理コンソールで、アプリケーションに移動します。
      - WebSphere Application Server バージョン 7.0: 管理コンソールで、「アプリケーション」 > 「アプリケーション・タイプ」 > 「WebSphere アプリケーション」をクリックします。
      - WebSphere Application Server バージョン 6.1: 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」をクリックします。
    - b. wxsrestservice アプリケーションの隣にあるチェック・ボックスにチェック・マークを付け、「開始」をクリックします。
    - c. アプリケーション・サーバー・プロファイルの SystemOut.log を確認します。REST データ・サービスが正常に開始すると、サーバー・プロファイルの SystemOut.log ファイル内に、以下のメッセージが表示されます。

CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

7. 以下のように、REST データ・サービスが動作していることを確認します。ポート番号は、アプリケーション・サーバー・プロファイル・ログ・ディレクトリー内にある SystemOut.log ファイルで、メッセージ ID の SRVE0250I で表示されている最初のポートを見ることで確認できます。デフォルト・ポートは 9080 です。

例: <http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/> 結果: AtomPub サービス文書が表示されます。

例: [http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/\\$metadata](http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/$metadata). Entity Model Data Extensions (EDMX) 文書が表示されます。

8. データ・グリッド・プロセスを停止するには、それぞれのコマンド・ウィンドウで CTRL+C を使用します。

## WebSphere Application Server 7.0 に統合された WebSphere eXtreme Scale での REST データ・サービスの開始:

このトピックでは、WebSphere eXtreme Scale で統合および拡張されている WebSphere Application Server バージョン 7.0 を使用して eXtreme Scale REST データ・サービスを構成および開始する方法について説明します。

### 始める前に

サンプル・スタンドアロン eXtreme Scale データ・グリッドが開始されていることを確認します。データ・グリッドの開始方法の詳細については、411 ページの『REST データ・サービスの使用可能化』を参照してください。

### このタスクについて

WebSphere Application Server を使用して WebSphere eXtreme Scale REST データ・サービスを開始するには、以下のステップに従います。

### 手順

1. 以下のように、WebSphere eXtreme Scale REST データ・サービスのサンプル構成 JAR をクラスパスに追加します。
  - a. WebSphere 管理コンソールを開きます。
  - b. 「環境」->「共有ライブラリー」にナビゲートします。
  - c. 「新規」をクリックします。
  - d. 以下の項目を該当するフィールドに追加します。
    - 1) 名前: extremescale\_gettingstarted\_config
    - 2) クラスパス
      - restservice\_home/gettingstarted/restclient/bin
      - restservice\_home/gettingstarted/common/bin

**要確認:** 各パスは異なる行に記述する必要があります。
  - e. 「OK」をクリックします。
  - f. 変更をマスター構成に保存します。
2. WebSphere 管理コンソールを使用して、REST データ・サービスの EAR ファイル wxsrestservice.ear を WebSphere Application Server にインストールします。
  - a. WebSphere 管理コンソールを開きます。
  - b. 「アプリケーション」->「新規アプリケーション」にナビゲートします。
  - c. ファイル・システム上の restservice\_home/lib/wxsrestservice.ear ファイルを参照します。ファイルを選択して、「次へ」をクリックします。
  - d. 詳細インストール・オプションを選択して、「次へ」をクリックします。
  - e. アプリケーション・セキュリティー警告画面で、「続行」をクリックします。
  - f. デフォルト・インストール・オプションを選択して、「次へ」をクリックします。

- g. wxsrestservice.war モジュールのマップ先サーバーを選択して、「次へ」をクリックします。
  - h. JSP 再ロード・ページで、デフォルトを使用し、「次へ」をクリックします。
  - i. 共有ライブラリー・ページで、「wxsrestservice.war」モジュールを、ステップ 1 で定義した共有ライブラリー `extremescale_gettingstarted_config` にマップします。
  - j. マップ共有ライブラリー・リレーションシップ・ページで、デフォルトを使用し、「次へ」をクリックします。
  - k. マップ仮想ホスト・ページで、デフォルトを使用し、「次へ」をクリックします。
  - l. マップ・コンテキスト・ルート・ページで、コンテキスト・ルートを `wxsrestservice` に設定します。
  - m. 要約画面で、「終了」をクリックして、インストールを完了します。
  - n. 変更をマスター構成に保存します。
3. eXtreme Scale セキュリティーを使用可能にして eXtreme Scale データ・グリッドを始動した場合には、`restservice_home/gettingstarted/restclient/bin/wxsRestService.properties` ファイルで以下のプロパティーを設定します。

`ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties`

- 4. アプリケーション・サーバーおよび「wxsrestservice」eXtreme Scale REST データ・サーバー・アプリケーションを開始します。

アプリケーションの開始後に、アプリケーション・サーバーの `SystemOut.log` を確認して、以下のメッセージが表示されていることを確認します。CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

- 5. 以下のように、REST データ・サービスが動作していることを確認します。
  - a. ブラウザーを開いて、以下にナビゲートします。

`http://localhost:9080/wxsrestservice/restservice/NorthwindGrid`

NorthwindGrid のサービス文書が表示されます。

- b. 以下にナビゲートします。

`http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/$metadata`

Entity Model Data Extensions (EDMX) 文書が表示されます。

- 6. データ・グリッド・プロセスを停止するには、それぞれのコマンド・ウィンドウで CTRL+C を使用して、プロセスを停止します。

## WebSphere Application Server Community Edition への REST データ・サービスのデプロイ

WebSphere Application Server Community Edition バージョン 2.1.1.3 以降で eXtreme Scale REST データ・サービスを構成できます。

## 始める前に

- IBM (推奨) もしくは Oracle の JRE または JDK バージョン 5 以上がインストールされ、JAVA\_HOME 環境変数が設定されています。
- WebSphere Application Server Community Edition バージョン 2.1.1.3 以降をダウンロードして、wasce\_root ディレクトリー (例えば、/opt/IBM/wasce ディレクトリー) にインストールします。バージョン 2.1.1 またはその他のバージョンについて、インストールの指示を参照してください。

## 手順

1. データ・グリッドを構成および開始します。
  - a. REST データ・サービスとともに使用するための eXtreme Scale データ・グリッドの構成の詳細については、417 ページの『REST データ・サービスのスタンドアロン・データ・グリッドの開始』を参照してください。
  - b. eXtreme Scale クライアントがデータ・グリッド内のエンティティーに接続およびアクセスできることを検査します。例えば、1 ページの『チュートリアル: WebSphere eXtreme Scale 入門』を参照してください。
2. eXtreme Scale REST サービス構成 JAR またはディレクトリーをビルドします。詳細については、200 ページの『REST データ・サービスのインストール』のトピックのパッケージ化とデプロイメントの情報を参照してください。
3. WebSphere Application Server Community Edition サーバーを始動します。
  - a. Java SE セキュリティーを使用可能にせずにサーバーを始動するには、以下のコマンドを実行します。

```
UNIX Linux wasce_root/bin/startup.sh
```

```
Windows wasce_root/bin/startup.bat
```

- b. Java SE セキュリティーを使用可能にしてサーバーを始動するには、以下のステップに従います。

```
UNIX Linux
```

  - 1) コマンド行または端末ウィンドウを開いて、以下のコピー・コマンドを実行 (または、指定したポリシー・ファイルを既存ポリシーにコピー) します。

```
cp restservice_home/gettingstarted/wasce/geronimo.policy wasce_root/bin
```
  - 2) wasce\_root/bin/setenv.sh ファイルを編集します。
  - 3) 「WASCE\_JAVA\_HOME=」が含まれている行の後に、以下を追加します。

```
export JAVA_OPTS="-Djava.security.manager -Djava.security.policy=geronimo.policy"
```

```
Windows
```

- 1) コマンド行ウィンドウを開いて、以下のコピー・コマンドを実行、または、指定したポリシー・ファイルを既存ポリシーにコピーします。

```
copy restservice_home%gettingstarted%wasce%geronimo.policy%bin
```
- 2) wasce\_root%bin%setenv.bat ファイルを編集します。
- 3) 「set WASCE\_JAVA\_HOME=」が含まれている行の後に、以下を追加します。

```
set JAVA_OPTS="-Djava.security.manager
-Djava.security.policy=geronimo.policy"
```

4. 以下のように、ObjectGrid クライアント・ランタイム JAR を WebSphere Application Server Community Edition リポジトリに追加します。
  - a. WebSphere Application Server Community Edition 管理コンソールを開いてログインします。デフォルト URL は `http://localhost:8080/console` で、デフォルトのユーザー ID は `system`、パスワードは `manager` です。
  - b. コンソール・ウィンドウの左側で、「サービス」フォルダー内の「リポジトリ」リンクをクリックします。
  - c. 「リポジトリへのアーカイブの追加」セクションで、入力テキスト・ボックスに以下を入力します。

表 28. リポジトリへのアーカイブの追加

テキスト・ボックス	値
ファイル	<code>wxs_home/lib/ogclient.jar</code>
グループ	<code>com.ibm.websphere.xs</code>
成果物	<code>ogclient</code>
バージョン	<code>7.1</code>
タイプ	<code>JAR</code>

- d. 「インストール」ボタンをクリックします。

クラスおよびライブラリーの依存関係を構成するさまざまな方法の詳細については、技術情報 [Specifying external dependencies to applications running on WebSphere Application Server Community Edition](#) を参照してください。

5. REST データ・サービス・モジュール `wxsrestservice.war` ファイルを WebSphere Application Server Community Edition サーバーにデプロイして開始します。
  - a. サンプル・デプロイメント・プラン XML ファイル `restservice_home/gettingstarted/wasce/geronimo-web.xml` をコピーして、REST データ・サービス構成 JAR またはディレクトリへのパス依存関係を組み込むように編集します。`wxsRestService.properties` ファイルおよび他の構成ファイルならびにメタデータ・クラスを組み込むようにするクラスパス設定の例については、セクションを参照してください。
  - b. WebSphere Application Server Community Edition 管理コンソールを開いてログインします。
 

**ヒント:** デフォルト URL は `http://localhost:8080/console` です。デフォルト・ユーザー ID は `system` で、パスワードは `manager` です。
  - c. コンソール・ウィンドウの左側にある「新規デプロイ」リンクをクリックします。
  - d. 「新規アプリケーションのインストール」ページで、テキスト・ボックスに以下の値を入力します。

表 29. 新規アプリケーションのインストール

テキスト・ボックス	値
アーカイブ	restservice_home/lib/wxsrestservice.war
プラン	restservice_home/gettingstarted/wasce/geronimo-web.xml

**ヒント:** ステップ 3 でコピーして編集した `geronimo-web.xml` ファイルへのパスを使用します。

- e. 「インストール」ボタンをクリックします。コンソール・ページに、アプリケーションが正常にインストールされて開始されたことが示されます。
- f. WebSphere Application Server Community Edition システム出力ログまたはコンソールを調べて、REST データ・サービスが正常に開始されたことを確認します。次のメッセージが表示されている必要があります。

CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

- 6. 以下のコマンドを実行して、WebSphere Application Server Community Edition サーバーを始動します。

- UNIX Linux `wasce_root/bin/startup.sh`
- Windows `wasce_root/bin/startup.bat`

- 7. 以下のように、eXtreme Scale REST データ・サービスおよび提供サンプルを WebSphere Application Server Community Edition サーバーにインストールします。

- a. 以下のように、ObjectGrid クライアント・ランタイム JAR を WebSphere Application Server Community Edition リポジトリに追加します。
  - 1) WebSphere Application Server Community Edition 管理コンソールを開いてログインします。デフォルト URL は `http://localhost:8080/console` です。デフォルト・ユーザー ID は `system` で、パスワードは `manager` です。
  - 2) コンソール・ウィンドウの左側で、「サービス」フォルダー内の「リポジトリ」リンクをクリックします。
  - 3) 「リポジトリへのアーカイブの追加」セクションで、入力テキスト・ボックスに以下を入力します。

表 30. リポジトリへのアーカイブの追加

テキスト・ボックス	値
ファイル	wxs_home/lib/ogclient.jar
グループ	com.ibm.websphere.xs
成果物	ogclient
バージョン	7.1
タイプ	JAR

- 4) 「インストール」ボタンをクリックします。



**ヒント:** クラスおよびライブラリーの依存関係を構成するさまざまな方法の詳細については、技術情報 *Specifying external dependencies to applications running on WebSphere Application Server Community Edition* を参照してください。

b. REST データ・サービス・モジュール `wxsrestservice.war` を WebSphere Application Server Community Edition サーバーにデプロイします。

1) 開始用 (getting started) サンプル・クラスパス・ディレクトリーへのパス依存関係を組み込むように、サンプルの `restservice_home/gettingstarted/wasce/geronimo-web.xml` デプロイメント XML ファイルを編集します。

- 2 つの開始用 (getting started) クライアント GBean の「classesDirs」を変更します。

GettingStarted\_Client\_SharedLib GBean の「classesDirs」パスを `restservice_home/gettingstarted/restclient/bin` に設定する必要があります。

GettingStarted\_Common\_SharedLib GBean の「classesDirs」パスを `restservice_home/gettingstarted/common/bin` に設定する必要があります。

- 2) WebSphere Application Server Community Edition 管理コンソールを開いてログインします。
- 3) コンソール・ウィンドウの左側にある「**新規デプロイ**」リンクをクリックします。
- 4) 「**新規アプリケーションのインストール**」ページで、テキスト・ボックスに以下の値を入力します。

表 31. 新規アプリケーションのインストール

テキスト・ボックス	値
アーカイブ	<code>restservice_home/lib/wxsrestservice.war</code>
プラン	<code>restservice_home/gettingstarted/wasce/geronimo-web.xml</code>

5) 「**インストール**」ボタンをクリックします。

コンソール・ページに、アプリケーションが正常にインストールされて開始されたことが示されます。

6) WebSphere Application Server Community Edition システム出力ログで以下のメッセージが存在することを確認して、REST データ・サービスが正常に開始されていることを検査します。

CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

8. 以下のように、REST データ・サービスが動作していることを確認します。

Web ブラウザーを開いて、URL `http://<host>:<port>/<context root>/restservice/<Grid Name>` にナビゲートします。

WebSphere Application Server Community Edition のデフォルト・ポートは 8080 で、`/var/config/config-substitutions.properties` ファイルで「HTTPPort」プロパティを使用して定義されます。

例: `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/`

## タスクの結果

AtomPub サービス文書が表示されます。

## WebSphere Application Server Community Edition での REST データ・サービスの開始:

このトピックでは、WebSphere Application Server Community Edition を使用して eXtreme Scale REST データ・サービスを構成および開始する方法について説明します。

### 始める前に

サンプル・データ・グリッドが開始されていることを確認します。グリッドの開始方法の詳細については、411 ページの『REST データ・サービスの使用可能化』を参照してください。

### 手順

1. WebSphere Application Server Community Edition バージョン 2.1.1.3 以降をダウンロードして、`wasce_root` (`/opt/IBM/wasce` など) にインストールします。
2. 以下のコマンドを実行して、WebSphere Application Server Community Edition サーバーを始動します。
  - **Linux** **UNIX** `wasce_root/bin/startup.sh`
  - **Windows** `wasce_root/bin/startup.bat`
3. eXtreme Scale セキュリティーを使用可能にして eXtreme Scale グリッドを始動した場合には、`restservice_home/gettingstarted/restclient/bin/wxsRestService.properties` ファイルで以下のプロパティを設定します。

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties  
loginType=none
```

4. 以下のように、eXtreme Scale REST データ・サービスおよび提供サンプルを WebSphere Application Server Community Edition サーバーにインストールします。
  - a. 以下のように、ObjectGrid クライアント・ランタイム JAR を WebSphere Application Server Community Edition リポジトリに追加します。
    - 1) WebSphere Application Server Community Edition 管理コンソールを開いてログインします。

**ヒント:** デフォルト URL は `http://localhost:8080/console` です。デフォルト・ユーザー ID は `system` で、パスワードは `manager` です。
    - 2) 「サービス」フォルダー内の「リポジトリ」をクリックします。
    - 3) 「リポジトリへのアーカイブの追加」セクションで、入力テキスト・ボックスに以下を入力します。

表 32. リポジトリへのアーカイブ

テキスト・ボックス	値
ファイル	wxs_home/lib/ogclient.jar
グループ	com.ibm.websphere.xs
成果物	ogclient
バージョン	7.0
タイプ	jar

4) 「インストール」 ボタンをクリックします。

**ヒント:** 構成クラスおよびライブラリーの依存関係のさまざまな方法の詳細については、技術情報 [Specifying external dependencies to applications running on WebSphere Application Server Community Edition](#) を参照してください。

b. REST データ・サービス・モジュール `wxsrestservice.war` ファイルを WebSphere Application Server Community Edition サーバーにデプロイします。

1) 開始用 (getting started) サンプル・クラスパス・ディレクトリへのパス依存関係を組み込むように、サンプルの `restservice_home/gettingstarted/wasce/geronimo-web.xml` デプロイメント XML ファイルを編集します。

2 つの開始用 (getting started) クライアント GBean の `classesDirs` パスを変更します。

- `GettingStarted_Client_SharedLib` GBean の「`classesDirs`」パスを `restservice_home/gettingstarted/restclient/bin` に設定する必要があります。
- `GettingStarted_Common_SharedLib` GBean の「`classesDirs`」パスを `restservice_home/gettingstarted/common/bin` に設定する必要があります。

2) WebSphere Application Server Community Edition 管理コンソールを開いてログインします。

**ヒント:** デフォルト URL は `http://localhost:8080/console` です。デフォルト・ユーザー ID は `system` で、パスワードは `manager` です。

3) 「新規デプロイ」をクリックします。

4) 「新規アプリケーションのインストール」 ページで、テキスト・ボックスに以下の値を入力します。

表 33. インストール値

テキスト・ボックス	値
アーカイブ	<code>restservice_home/lib/wxsrestservice.war</code>
プラン	<code>restservice_home/gettingstarted/wasce/geronimo-web.xml</code>

- 5) 「インストール」 ボタンをクリックします。

コンソール・ページに、アプリケーションが正常にインストールされて開始されたことが示されます。

- 6) WebSphere Application Server Community Edition システム出力ログまたはコンソールで以下のメッセージが存在することを確認して、REST データ・サービスが正常に開始されていることを検査します。

CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

5. 以下のように、REST データ・サービスが動作していることを確認します。
  - a. ブラウザー・ウィンドウで、リンク `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid` を開きます。NorthwindGrid グリッドのサービス文書が表示されます。
  - b. ブラウザー・ウィンドウで、リンク `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata` を開きます。Entity Model Data Extensions (EDMX) 文書が表示されます。
6. グリッド・プロセスを停止するには、それぞれのコマンド・ウィンドウで CTRL+C を使用して、プロセスを停止します。
7. WebSphere Application Server Community Edition を停止するには、以下のコマンドを使用します。
  - **UNIX**   **Linux**   `wasce_root/bin/shutdown.sh`
  - **Windows**   `wasce_root%bin%shutdown.bat`

**ヒント:** デフォルト・ユーザー ID は `system` で、パスワードは `manager` です。カスタム・ポートを使用する場合は、`-port` オプションを使用します。

## Apache Tomcat への REST データ・サービスのデプロイ

このトピックでは、WebSphere eXtreme Scale REST データ・サービスを Apache Tomcat バージョン 5.5 以上で構成する方法について説明します。

### このタスクについて

- IBM もしくは Oracle の JRE または JDK バージョン 5 以上がインストールされ、`JAVA_HOME` 環境変数が指定されています。
- Apache Tomcat バージョン 5.5 以上をインストールします。Tomcat のインストール方法の詳細については、Apache Tomcat を参照してください。
- WebSphere eXtreme Scale のスタンドアロン・インストール。

### 手順

1. Oracle JRE または JDK を使用している場合は、IBM ORB を Tomcat にインストールします。
  - a. Tomcat バージョン 5.5:

すべての JAR ファイルを以下のようにコピーします。

`wxs_home/lib/endorsed` ディレクトリー

から

`tomcat_root/common/endorsed` ディレクトリーに

b. Tomcat バージョン 6.0:

以下のように、「endorsed」ディレクトリーを作成します。

```
UNIX Linux mkdir tomcat_root/endorsed
```

```
Windows md tomcat_root/endorsed
```

すべての JAR ファイルを以下のようにコピーします。

`wxs_home/lib/endorsed`

から

`tomcat_root/common/endorsed`

2. データ・グリッドを構成および開始します。
  - a. REST データ・サービスとともに使用するためのデータ・グリッドの構成の詳細については、253 ページの『第 6 章 構成』を参照してください。
  - b. eXtreme Scale クライアントがグリッド内のエンティティーに接続およびアクセスできることを検査します。例えば、409 ページの『REST データ・サービスの構成』を参照してください。
3. eXtreme Scale REST サービス構成 JAR またはディレクトリーをビルドします。詳細については、200 ページの『REST データ・サービスのインストール』のパッケージ化とデプロイメントの情報を参照してください。
4. REST データ・サービス・モジュール `wxsrestservice.war` を Tomcat サーバーにデプロイします。

`wxsrestservice.war` ファイルを以下のようにコピーします。

`restservice_home/lib`

から

`tomcat_root/webapps`

5. ObjectGrid クライアント・ランタイム JAR およびアプリケーション JAR を Tomcat の共有クラスパスに追加します。
  - a. `tomcat_root/conf/catalina.properties` ファイルを編集します。
  - b. 以下の各パス名をコンマで区切って、`shared.loader` プロパティーの末尾に追加します。
    - `wxs_home/lib/ogclient.jar`
    - `restservice_home/gettingstarted/restclient/bin`
    - `restservice_home/gettingstarted/common/bin`
6. Java 2 セキュリティーを使用している場合は、以下のように、Tomcat ポリシー・ファイルにセキュリティ許可を追加します。
  - Tomcat バージョン 5.5 を使用している場合:

`restservice_home/gettingstarted/tomcat/catalina-5_5.policy` にあるサンプルの 5.5 catalina ポリシー・ファイルの内容を `tomcat_root/conf/catalina.policy` ファイルにマージします。

- Tomcat バージョン 6.0 を使用している場合:

`restservice_home/gettingstarted/tomcat/catalina-6_0.policy` にあるサンプルの 6.0 catalina ポリシー・ファイルの内容を `tomcat_root/conf/catalina.policy` ファイルにマージします。

7. 以下のように、Tomcat サーバーを始動します。

- **Tomcat 5.5 を UNIX または Windows で、あるいは Tomcat 6.0 ZIP 配布版を使用する場合:**

a. `cd tomcat_root/bin`

b. 以下のように、サーバーを始動します。

- Java 2 セキュリティーを使用可能にしていない場合:

```
UNIX Linux ./catalina.sh run
```

```
Windows catalina.bat run
```

- Java 2 セキュリティーを使用可能にしている場合:

```
UNIX Linux ./catalina.sh run -security
```

```
Windows catalina.bat run -security
```

c. Apache Tomcat のログは、コンソールに表示されます。REST データ・サービスが正常に開始すると、管理コンソールに以下のメッセージが表示されます。

```
CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。
```

- **Windows インストーラー用配布版を使用して、Tomcat 6.0 を Windows で使用する場合:**

a. `cd /bin`

b. 以下のように、Apache Tomcat 6 構成ツールを開始します。

```
tomcat6w.exe
```

c. Java 2 セキュリティーを使用可能にする場合は、以下のようにします (オプション)。

Apache Tomcat 6 のプロパティ・ウィンドウの Java タブの Java オプションに以下の項目を追加します。

```
-Djava.security.manager
```

```
-Djava.security.policy=%conf%catalina.policy
```

d. Apache Tomcat 6 のプロパティ・ウィンドウの「Start」ボタンをクリックして、Tomcat サーバーを始動します。



- e. 以下のログを確認して、Tomcat サーバーが正常に始動していることを確認します。

– `tomcat_root/bin/catalina.log`

Tomcat サーバー・エンジンの状況を表示します。

– `tomcat_root/bin/stdout.log`

システム出力ログを表示します。

- f. REST データ・サービスが正常に開始すると、システム出力ログに以下のメッセージが表示されます。

CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

8. 以下のように、REST データ・サービスが動作していることを確認します。

Web ブラウザーを開いて、以下の URL にナビゲートします。

`http://host:port/context_root/restservice/grid_name`

Tomcat のデフォルト・ポートは 8080 であり、`tomcat_root/conf/server.xml` ファイル内の `<Connector>` エレメントで構成されます。

例:

`http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/`

## タスクの結果

AtomPub サービス文書が表示されます。

### Apache Tomcat での REST データ・サービスの開始:

このトピックでは、Apache Tomcat バージョン 5.5 以上を使用して、eXtreme Scale REST データ・サービスを構成および開始する方法について説明します。

#### 始める前に

サンプル eXtreme Scale データ・グリッドが開始されていることを確認します。データ・グリッドの開始方法の詳細については、411 ページの『REST データ・サービスの使用可能化』を参照してください。

#### 手順

1. `tomcat_root` に Apache Tomcat バージョン 5.5 以上をダウンロードしてインストールします。例: `/opt/tomcat`
2. 以下のように、eXtreme Scale REST データ・サービスおよび提供サンプルを Tomcat サーバーにインストールします。
  - a. Oracle JRE または JDK を使用している場合は、IBM ORB を Tomcat にインストールする必要があります。
    - Tomcat バージョン 5.5 の場合

すべての JAR ファイルを以下のようにコピーします。

wxs\_home/lib/endorsed

to

tomcat\_root/common/endorsed

- Tomcat バージョン 6.0 の場合

- 1) 「endorsed」ディレクトリーを作成します。

– **UNIX** **Linux** mkdir tomcat\_root/endorsed

– **Windows** md tomcat\_root/endorsed

- 2) すべての JAR ファイルを以下のようにコピーします。

wxs\_home/lib/endorsed

to

tomcat\_root/endorsed

- b. REST データ・サービス・モジュール wxsrestservice.war を Tomcat サーバーにデプロイします。

wxsrestservice.war ファイルを以下のようにコピーします。

restservice\_home/lib

から

tomcat\_root/webapps

- c. ObjectGrid クライアント・ランタイム JAR およびアプリケーション JAR を Tomcat の共有クラスパスに追加します。

- 1) tomcat\_root/conf/catalina.properties ファイルを編集します。

- 2) コンマ区切りリストの形式で、以下のパス名を shared.loader プロパティの末尾に追加します。

- wxs\_home/lib/ogclient.jar
- restservice\_home/gettingstarted/restclient/bin
- restservice\_home/gettingstarted/common/bin

**重要:** パス分離文字は、スラッシュにする必要があります。

3. eXtreme Scale セキュリティーを使用可能にして eXtreme Scale データ・グリッドを始動した場合には、restservice\_home/gettingstarted/restclient/bin/wxsRestService.properties ファイルで以下のプロパティを設定します。

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties  
loginType=none
```

4. REST データ・サービスが含まれた Tomcat サーバーを始動します。

- Tomcat 5.5 を UNIX または Windows で、あるいは Tomcat 6.0 を UNIX で使用する場合:

- a. cd tomcat\_root/bin

- b. 以下のように、サーバーを始動します。

- **UNIX** **Linux** ./catalina.sh run

- **Windows** catalina.bat run

- c. コンソールに、Apache Tomcat のログが表示されます。REST データ・サービスが正常に開始すると、管理コンソールに以下のメッセージが表示されます。

CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

- Tomcat 6.0 を Windows で使用する場合:

a. cd tomcat\_root/bin

b. tomcat6w.exe コマンドで、Apache Tomcat 6 構成ツールを開始します。

c. Apache Tomcat 6 のプロパティ・ウィンドウの「Start」ボタンをクリックして、Tomcat サーバーを始動します。

d. 以下のログを確認して、Tomcat サーバーが正常に始動していることを確認します。

- tomcat\_root/bin/catalina.log

Tomcat サーバー・エンジンの状況を表示します。

- tomcat\_root/bin/stdout.log

システム出力ログを表示します。

e. REST データ・サービスが正常に開始されていると、以下のメッセージがシステム出力ログに表示されます。CW0BJ4000I: WebSphere eXtreme Scale REST データ・サービスが開始されました。

5. 以下のように、REST データ・サービスが動作していることを確認します。

a. ブラウザーを開いて、以下にナビゲートします。

<http://localhost:8080/wxsrestservice/restservice/NorthwindGrid>

NorthwindGrid のサービス文書が表示されます。

b. 以下にナビゲートします。

[http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/\\$metadata](http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata)

Entity Model Data Extensions (EDMX) 文書が表示されます。

6. データ・グリッド・プロセスを停止するには、それぞれのコマンド・ウィンドウで CTRL+C を使用します。

7. Tomcat を停止するには、Tomcat を開始したウィンドウで CTRL+C を使用します。

## REST データ・サービス ATOM フィードにアクセスする Web ブラウザーの構成

eXtreme Scale REST データ・サービスは、Web ブラウザーを使用した場合に、ATOM フィードをデフォルトで作成します。ATOM フィード・フォーマットは、古いブラウザーでは互換性がないことがあり、また、XML として表示できないデー

タとして解釈されることがあります。Internet Explorer バージョン 8 または Firefox バージョン 3 を構成して、ブラウザ内で ATOM フィードおよび XML を表示できます。

## このタスクについて

eXtreme Scale REST データ・サービスは、Web ブラウザーを使用した場合に、ATOM フィードをデフォルトで作成します。ATOM フィード・フォーマットは、古いブラウザでは互換性がないことがあり、また、XML として表示できないデータとして解釈されることがあります。古いブラウザでは、ファイルをディスクに保存するように求められます。ファイルをダウンロードしてから、お好みの XML リーダーを使用して、ファイルを参照してください。生成された XML は表示用にフォーマット設定されていないため、すべてが 1 行で出力されます。ほとんどの XML 読み取りプログラム (Eclipse など) では、XML を可読フォーマットに再設定することができます。

最新のブラウザ (Microsoft Internet Explorer バージョン 8 や Firefox バージョン 3 など) では、ATOM XML ファイルをブラウザでネイティブ表示できます。以下のトピックでは、ブラウザ内で ATOM フィードおよび XML を表示するように Internet Explorer バージョン 8 および Firefox バージョン 3 を構成する方法を詳細に説明します。

## 手順

### Internet Explorer バージョン 8 の構成

- REST データ・サービスが生成する ATOM フィードを Internet Explorer で表示できるようにするには、以下のステップを使用します。
  1. 「ツール」 > 「インターネット オプション」をクリックします。
  2. 「コンテンツ」タブを選択します。
  3. 「フィードと Web スライス」セクションの「設定」ボタンをクリックします。
  4. 「フィードの読み取りビューを有効にする」ボックスのチェック・マークを外します。
  5. 「OK」をクリックして、ブラウザに戻ります。
  6. Internet Explorer を再始動します。

### Firefox バージョン 3 の構成

- Firefox では、コンテンツ・タイプが application/atom+xml のページは自動的に表示されません。ページの初回表示時に、Firefox でファイルを保存するように求められます。ページを表示するには、以下のように、Firefox でファイル自体を開きます。
  1. アプリケーション選択ダイアログ・ボックスで、「アプリケーションで開く」ラジオ・ボタンを選択して、「参照」ボタンをクリックします。
  2. Firefox インストール・ディレクトリーにナビゲートします。例: C:\Program Files\Mozilla Firefox
  3. firefox.exe を選択して、「OK」ボタンをクリックします。
  4. 「今後この種類のファイルは同様に処理する」チェック・ボックスにチェック・マークを付けます。

5. 「OK」 ボタンをクリックします。
  6. Firefox で、ATOM XML ページが新しいブラウザー・ウィンドウまたはタブで表示されます。
- Firefox は、ATOM フィードを可読フォーマットで自動的にレンダリングします。ただし、REST データ・サービスが作成するフィードには XML が含まれません。Firefox では、フィード・レンダラーを使用不可にしない限り、XML を表示できません。Internet Explorer とは異なり、Firefox では、ATOM フィード・レンダリング・プラグインを明示的に編集する必要があります。ATOM フィードを XML ファイルとして読み取るように Firefox を構成するには、以下のステップに従います。

1. ファイル <firefoxInstallRoot>%components%FeedConverter.js をテキスト・エディターで開きます。このパスで、<firefoxInstallRoot> は、Firefox がインストールされているルート・ディレクトリーです。

Windows オペレーティング・システムの場合、デフォルト・ディレクトリーは C:%Program Files%Mozilla Firefox です。

2. 以下のようなスニペットを探します。

```
// show the feed page if it wasn't sniffed and we have a document,
// or we have a document, title, and link or id
if (result.doc && (!this._sniffed ||
    (result.doc.title && (result.doc.link || result.doc.id)))) {
```

3. if および result で開始している 2 行の行頭に // (2 つのスラッシュ) を追加して、コメント化します。
4. スニペットにステートメント if(0) { を追加します。
5. 結果のテキストは以下のようになります。

```
// show the feed page if it wasn't sniffed and we have a document,
// or we have a document, title, and link or id
//if (result.doc && (!this._sniffed ||
//    (result.doc.title && (result.doc.link || result.doc.id)))) {
if(0) {
```

6. ファイルを保存します。
  7. Firefox を再始動します。
  8. これで、Firefox のブラウザーで、すべてのフィードを自動的に表示できるようになりました。
- いくつかの URL を試して、セットアップをテストします。

## 例

このセクションでは、REST に付属の開始用 (getting started) サンプルで追加されたデータを表示するために使用できる一部のサンプル URL について説明します。以下の URL を使用する前に、サンプル Java クライアントまたはサンプル Visual Studio WCF Data Services クライアントを使用して、デフォルト・データ・セットを eXtreme Scale サンプル・データ・グリッドに追加します。

以下の例では、ポートは 8080 であると想定しています (ポートは可変)。別のアプリケーション・サーバーで REST データ・サービスを構成する方法の詳細については、セクションを参照してください。

- 「ACME」という ID の単一の顧客を表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')
```

- 「ACME」という顧客のすべてのオーダーを表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')/orders
```

- 顧客「ACME」およびオーダーを表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')?$expand=orders
```

- 顧客「ACME」のオーダー 1000 を表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Order(orderId=1000,customer_customerId='ACME')
```

- 顧客「ACME」のオーダー 1000 および関連付けられた Customer を表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Order(orderId=1000,customer_customerId='ACME')?$expand=customer
```

- 顧客「ACME」のオーダー 1000 および関連付けられた Customer および OrderDetails を表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Order(orderId=1000,customer_customerId='ACME')?$expand=customer,orderDetails
```

- 顧客「ACME」の 2009 年 10 月 (GMT) のすべてのオーダーを表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer(customerId='ACME')/orders?$filter=orderDate ge datetime'2009-10-01T00:00:00' and orderDate lt datetime'2009-11-01T00:00:00'
```

- 顧客「ACME」の 2009 年 10 月 (GMT) の最初の 3 つのオーダーおよび orderDetails を表示:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer(customerId='ACME')/orders?$filter=orderDate ge datetime'2009-10-01T00:00:00' and orderDate lt datetime'2009-11-01T00:00:00'&&$orderby=orderDate&$top=3&$expand=orderDetails
```

## REST データ・サービスでの Java クライアントの使用

Java クライアント・アプリケーションは eXtreme Scale EntityManager API を使用して、データをグリッドに挿入します。

### このタスクについて

前のセクションでは、eXtreme Scale データ・グリッドを作成して、eXtreme Scale REST データ・サービスを構成および開始する方法について説明しました。Java クライアント・アプリケーションは eXtreme Scale EntityManager API を使用して、データをグリッドに挿入します。REST インターフェースの使用方法については説明されていません。このクライアントの目的は、EntityManager API を使用して eXtreme Scale データ・グリッドと対話して、グリッド内のデータを変更できるようにする方法を説明することです。REST データ・サービスを使用してグリッド内のデータを表示するには、Web ブラウザーを使用するか、Visual Studio 2008 クライアント・アプリケーションを使用します。

### 手順

eXtreme Scale データ・グリッドにコンテンツを迅速に追加するには、以下のコマンドを実行します。

1. コマンド行または端末ウィンドウを開いて、以下のように、JAVA\_HOME 環境変数を設定します。

- `Linux` `UNIX` `export JAVA_HOME=java_home`



- `Windows` `set JAVA_HOME=java_home`
2. `cd restservice_home/gettingstarted`
  3. グリッドに何らかのデータを挿入します。挿入したデータは、後から Web ブラウザーおよび REST データ・サービスを使用して取得します。

eXtreme Scale セキュリティーなしで データ・グリッドを始動した場合には、以下のコマンドを使用します。

- `UNIX` `Linux` `./runclient.sh load default`
- `Windows` `runclient.bat load default`

eXtreme Scale セキュリティーありで データ・グリッドを始動した場合には、以下のコマンドを使用します。

- `UNIX` `Linux` `./runclient_secure.sh load default`
- `Windows` `runclient_secure.bat load default`

Java クライアントの場合は、以下のコマンド構文を使用します。

- `UNIX` `Linux` `runclient.sh command`
- `Windows` `runclient.bat command`

以下のコマンドが使用可能です。

- `load default`

Customer、Category、および Product の各エンティティの事前定義セットをデータ・グリッドにロードして、顧客ごとに Orders のランダム・セットを作成します。

- `load category categoryId categoryName firstProductId num_products`

製品 Category および固定数の Product エンティティをデータ・グリッドに作成します。firstProductId パラメーターには、最初の製品の ID 番号を指定し、指定数の製品が作成されるまで、それ以降の製品に次の ID を割り当てます。

- `load customer companyCode contactNamecompanyName numOrders firstOrderIdshipCity maxItems discountPct`

新規 Customer をデータ・グリッドにロードして、現在グリッドにロードされている任意のランダム製品の Order エンティティの固定セットを作成します。Order の数は、<numOrders> パラメーターを設定することで決定します。各 Order には、ランダム数の OrderDetail エンティティが含まれます (最大数は <maxItems>)。

- `display customer companyCode`

Customer エンティティ、および関連付けられた Order エンティティと OrderDetail エンティティを表示します。

- `display category categoryId`

製品 Category エンティティおよび関連付けられた Product エンティティを表示します。

## タスクの結果

- `runclient.bat load default`
- `runclient.bat load customer IBM "John Doe" "IBM Corporation" 5 5000 Rochester 5 0.05`
- `runclient.bat load category 5 "Household Items" 100 5`
- `runclient.bat display customer IBM`
- `runclient.bat display category 5`

## Eclipse でのサンプル・データ・グリッドおよび Java クライアントの実行およびビルド

REST データ・サービスの開始用 (getting started) サンプルは、Eclipse を使用して更新および拡張できます。Eclipse 環境のセットアップ方法の詳細については、テキスト資料 `restservice_home/gettingstarted/ECLIPSE_README.txt` を参照してください。

WXSRestGettingStarted プロジェクトを Eclipse をインポートして正常にビルドすると、サンプルが自動的に再コンパイルされて、コンテナ・サーバーおよびクライアントを開始するために使用するスクリプト・ファイルでクラス・ファイルおよび XML ファイルが自動的に選択されます。Web サーバーが Eclipse ビルド・ディレクトリを自動的に読み取るように構成されているため、変更が行われると、REST データ・サービスでもその変更が自動的に検出されます。

**重要:** ソースまたは構成ファイルを変更する際には、eXtreme Scale コンテナ・サーバーと REST データ・サービス・アプリケーションの両方を再始動する必要があります。eXtreme Scale コンテナ・サーバーは、REST データ・サービス Web アプリケーションの前に始動する必要があります。

## REST データ・サービスでの Visual Studio 2008 WCF クライアント

eXtreme Scale REST データ・サービス開始用 (getting started) サンプルには、eXtreme Scale REST データ・サービスと対話できる WCF Data Services クライアントが含まれています。サンプルは、C# のコマンド行アプリケーションとして作成されています。

### ソフトウェア要件

WCF Data Services C# サンプル・クライアントには、以下が必要です。

- オペレーティング・システム
  - Microsoft Windows XP
  - Microsoft Windows Server 2003
  - Microsoft Windows Server 2008
  - Microsoft Windows Vista
- Microsoft Visual Studio 2008 (Service Pack 1 適用済み)

**ヒント:** 追加のハードウェアおよびソフトウェアの要件については、上のリンクを参照してください。

- Microsoft .NET Framework 3.5 Service Pack 1
- Microsoft Support: .NET Framework 3.5 Service Pack 1 の更新が使用可能です

## 開始用 (getting started) クライアントのビルドおよび実行

WCF Data Services サンプル・クライアントには、サンプルを実行するための、Visual Studio 2008 のプロジェクトとソリューションおよびソース・コードが含まれています。サンプルを実行するには、Visual Studio 2008 にロードして、Windows 実行可能プログラムにコンパイルする必要があります。サンプルをビルドして実行するには、テキスト資料 `restservice_home/gettingstarted/VS2008_README.txt` を参照してください。

## WCF Data Services C# クライアントのコマンド構文

```
Windows WXSRESTGettingStarted.exe <サービス URL> <コマンド>
```

<サービス URL> は、セクションで構成された eXtreme Scale REST データ・サービスの URL です。

以下のコマンドが使用可能です。

- `load default`

Customer、Category、および Product の各エンティティの事前定義セットをデータ・グリッドにロードして、顧客ごとに Orders のランダム・セットを作成します。

- `load category <categoryId> <categoryName> <firstProductId> <numProducts>`

製品 Category および固定数の Product エンティティをデータ・グリッドに作成します。 `firstProductId` パラメーターには、最初の製品の ID 番号を指定し、指定数の製品が作成されるまで、それ以降の製品に次の ID を割り当てます。

- `load customer <companyCode> <contactName> <companyName> <numOrders> <firstOrderId> <shipCity> <maxItems> <discountPct>`

新規 Customer をデータ・グリッドにロードして、現在データ・グリッドにロードされている任意のランダム製品の Order エンティティの固定セットを作成します。 Order の数は、`<numOrders>` パラメーターを設定することで決定します。各 Order には、ランダム数の OrderDetail エンティティが含まれます (最大数は `<maxItems>`)。

- `display customer <companyCode>`

Customer エンティティ、および関連付けられた Order エンティティと OrderDetail エンティティを表示します。

- `display category <categoryId>`

製品 Category エンティティおよび関連付けられた Product エンティティを表示します。

- `unload`

「default load」コマンドを使用してロードされたすべてのエンティティを削除します。

次に、さまざまなコマンドの例を示します。

- WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid load default
- WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid load customer
- IBM "John Doe" "IBM Corporation" 5 5000 Rochester 5 0.05
- WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid load category 5 "Household Items" 100 5
- WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid display customer IBM
- WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid display category 5

---

## OSGi 用サーバーの構成

WebSphere eXtreme Scale にはサーバー OSGi バンドルが組み込まれていて、OSGi フレームワーク内でサーバーとコンテナを開始および構成できます。構成トピックでは、eXtreme Scale サーバー・バンドル、OSGi Blueprint サービス、および eXtreme Scale 構成を使用して、Eclipse Equinox OSGi フレームワーク内で eXtreme Scale サーバーを実行する方法を説明します。

### このタスクについて

Eclipse Equinox 内で eXtreme Scale サーバーを開始するには、次のタスクが必要です。

#### 手順

1. eXtreme Scale プラグインを保管する OSGi バンドルを作成し、それらをサービスとして公開し、それらのサービスを参照するよう ObjectGrid 記述子 XML ファイルを更新します。
2. OSGi を構成して eXtreme Scale コンテナ・サーバーを開始します。
3. eXtreme Scale サーバー・バンドルを OSGi フレームワーク内にインストールし、開始します。
4. eXtreme Scale プラグインを含んでいる OSGi バンドルをインストールし、開始します。

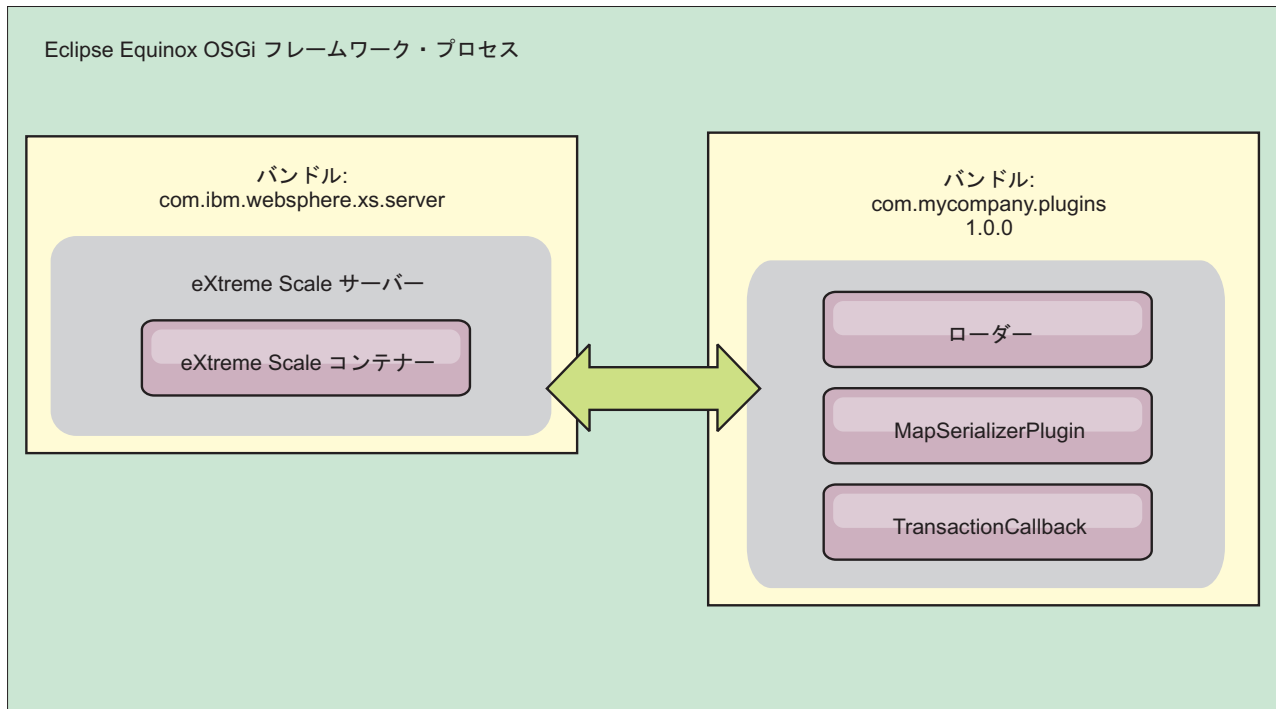


図 49. eXtreme Scale プラグインを含んでいる OSGi バンドルをインストールおよび開始する Eclipse Equinox プロセス

## OSGi Blueprint での eXtreme Scale プラグインの構成

eXtreme Scale ObjectGrid および BackingMap プラグインはすべて、Eclipse Gemini または Apache Aries で使用可能な OSGi Blueprint サービスを使用して OSGi Bean およびサービスとして定義できます。

### 始める前に

プラグインを OSGi サービスとして構成するには、プラグインを OSGi バンドルにパッケージ化し、必要なプラグインの基本原則を理解する必要があります。バンドルは、WebSphere eXtreme Scale サーバー・パッケージまたはクライアント・パッケージに加えてプラグインが必要とするその他の従属パッケージをインポートするか、eXtreme Scale サーバー・バンドルまたはクライアント・バンドルへのバンドル依存関係を作成しなければなりません。このトピックでは、Blueprint XML を構成して、プラグイン Bean を作成し、それらを eXtreme Scale で使用できるように OSGi サービスとして公開する方法を説明します。

### このタスクについて

Bean とサービスは Blueprint XML ファイル内に定義します。そうすると、Blueprint コンテナによって Bean が検出および作成され、Bean 同士がワイヤリングされ、サービスとして公開されます。このプロセスにより、eXtreme Scale サーバー・バンドルとクライアント・バンドルを含め、その他の OSGi バンドルで Bean が使用可能になります。

eXtreme Scale で使用するカスタム・プラグイン・サービスを作成する場合、プラグインをホスティングするバンドルは、Blueprint を使用するよう構成しなければな

りません。さらに、Blueprint XML ファイルを作成し、そのファイルをバンドル内に保管しなければなりません。Blueprint Container 仕様の全般的な知識を得るには、Blueprint Container 仕様による OSGi アプリケーションの構築を参照してください。

## 手順

1. Blueprint XML ファイルを作成します。ファイルには任意の名前を付けることができます。ただし、次のように blueprint 名前空間を含める必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  ...
</blueprint>
```

2. eXtreme Scale プラグインごとに Bean 定義を Blueprint XML ファイル内に作成します。

Bean は <bean> エレメントを使用して定義し、他の Bean 参照にワイヤリングでき、初期化パラメーターを組み込むことができます。

**重要:** Bean の定義時は、正しいスコープを使用する必要があります。Blueprint は singleton スコープとプロトタイプ・スコープをサポートします。eXtreme Scale はカスタム断片スコープもサポートします。

すべての Bean は、関連付けられる各 ObjectGrid 断片または BackingMap インスタンスで固有でなければならぬため、ほとんどの eXtreme Scale プラグインはプロトタイプ・スコープまたは断片スコープの Bean として定義します。正しいインスタンスの取得を可能にするために Bean を他のコンテキストで使用する場合、断片スコープの Bean が便利です。

プロトタイプ・スコープの Bean を定義するには、Bean の scope="prototype" 属性を使用します。

```
<bean id="myPluginBean" class="com.mycompany.MyBean" scope="prototype">
  ...
</bean>
```

断片スコープの Bean を定義するには、objectgrid 名前空間を XML スキーマに追加し、Bean の scope="objectgrid:shard" 属性を使用してください。

```
<?xml version="1.0" encoding="UTF-8"?>

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"

  xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
    http://www.ibm.com/schema/objectgrid/objectgrid.xsd">

  <bean id="myPluginBean" class="com.mycompany.MyBean"
    scope="objectgrid:shard">
    ...
  </bean>

  ...
```

3. 各プラグイン Bean の PluginServiceFactory Bean 定義を作成します。正しい Bean スコープを適用できるように、すべての eXtreme Scale Bean に PluginServiceFactory Bean を定義する必要があります。eXtreme Scale には、ユーザーが使用できる BlueprintServiceFactory が組み込まれています。それには設



定が必要な 2 つのプロパティがあります。blueprintContainer プロパティには blueprintContainer 参照を設定し、beanId プロパティには Bean ID 名を設定する必要があります。eXtreme Scale が適切な Bean のインスタンスを生成するためにサービスを検索すると、サーバーは Blueprint コンテナを使用して Bean コンポーネント・インスタンスを検索します。

```
bean id="myPluginBeanFactory"
  class="com.ibm.websphere.objectgrid.plugins.osgi.BluePrintServiceFactory">
  <property name="blueprintContainer" ref="blueprintContainer" />
  <property name="beanId" value="myPluginBean" />
</bean>
```

4. 各 PluginServiceFactory Bean のサービス・マネージャーを作成します。各サービス・マネージャーは、<service> エレメントを使用して PluginServiceFactory Bean を公開します。サービス・エレメントは、OSGi に公開する名前、PluginServiceFactory Bean への参照、公開するインターフェース、およびサービスのランキングを識別します。eXtreme Scale はサービス・マネージャー・ランキングを使用して、eXtreme Scale グリッドがアクティブなときにサービス・アップグレードを実行します。ランキングが指定されない場合、OSGi フレームワークはランキング 0 を想定します。詳細については、サービス・ランキングの更新を参照してください。

Blueprint には、サービス・マネージャーを構成するためのオプションがいくつかあります。PluginServiceFactory Bean の単純なサービス・マネージャーを定義するには、PluginServiceFactory Bean ごとに <service> エレメントを作成します。

```
<service ref="myPluginBeanFactory"
  interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory"
  ranking="1">
</service>
```

5. Blueprint XML ファイルをプラグイン・バンドル内に保管します。Blueprint XML ファイルは OSGI-INF/blueprint ディレクトリー内に保管し、Blueprint コンテナが検出されるようにしなければなりません。

Blueprint XML ファイルを他のディレクトリーに保管するには、次の Bundle-Blueprint マニフェスト・ヘッダーを指定する必要があります。

```
Bundle-Blueprint: OSGI-INF/blueprint.xml
```

## タスクの結果

これで、OSGi Blueprint コンテナ内に公開される eXtreme Scale プラグインが構成されました。さらに、OSGi Blueprint サービスを使用してプラグインを参照するように ObjectGrid 記述子 XML ファイルも構成されました。

## OSGi Blueprint でのサーバーの構成

OSGi Blueprint XML ファイルを使用して WebSphere eXtreme Scale コンテナ・サーバーを構成できます。この方法によりパッケージ化が簡単になるほか、自己完結型サーバー・バンドルの作成が可能になります。

### 始める前に

このトピックは、以下のタスクが完了していることを前提としています。

- Eclipse Gemini または Apache Aries の Blueprint コンテナを使用する Eclipse Equinox OSGi フレームワークをインストールし、開始していること。
- eXtreme Scale サーバー・バンドルをインストールし、開始していること。
- eXtreme Scale 動的プラグイン・バンドルの作成が完了していること。
- eXtreme Scale ObjectGrid 記述子 XML ファイルとデプロイメント・ポリシー XML ファイルの作成が完了していること。

## このタスクについて

このタスクでは、Blueprint XML ファイルを使用して eXtreme Scale サーバーとコンテナを構成する方法を説明します。この手順の結果として、コンテナ・バンドルが作成されます。コンテナ・バンドルが開始されると、eXtreme Scale サーバー・バンドルはそのバンドルを追跡し、サーバー XML を解析し、サーバーとコンテナを開始します。

コンテナ・バンドルは、動的プラグイン更新が必要でない場合またはプラグインが動的更新をサポートしない場合に、オプションでアプリケーションおよび eXtreme Scale プラグインと結合できます。

## 手順

1. objectgrid 名前空間が組み込まれた Blueprint XML ファイルを作成します。ファイルには任意の名前を付けることができます。ただし、blueprint 名前空間を含める必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
  xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
    http://www.ibm.com/schema/objectgrid/objectgrid.xsd">
  ...
</blueprint>
```

2. 適切なサーバー・プロパティを使用して eXtreme Scale サーバーの XML 定義を追加します。すべての使用可能な構成プロパティの詳細については、Spring 記述子 XML ファイルを参照してください。次の XML 定義の例を参照してください。

```
<objectgrid:server id="xsServer" tracespec="ObjectGridOSGi=all=enabled"
  tracefile="logs/osgi/wxsserver/trace.log" jmxport="1199" listenerPort="2909">
  <objectgrid:catalog host="catserver1.mycompany.com" port="2809" />
  <objectgrid:catalog host="catserver2.mycompany.com" port="2809" />
</objectgrid:server>
```

3. サーバー定義への参照と、バンドルに組み込まれている ObjectGrid 記述子 XML ファイルと ObjectGrid デプロイメント XML ファイルを使用して eXtreme Scale コンテナの XML 定義を追加します。例えば、次のようになります。

```
<objectgrid:container id="container"
  objectgridxml="/META-INF/objectGrid.xml"
  deploymentxml="/META-INF/objectGridDeployment.xml"
  server="xsServer" />
```

4. Blueprint XML ファイルをコンテナ・バンドル内に保管します。Blueprint XML は OSGI-INF/blueprint ディレクトリ内に保管し、Blueprint コンテナが検出されるようにしなければなりません。

Blueprint XML を他のディレクトリーに保管するには、Bundle-Blueprint マニフェスト・ヘッダーを指定する必要があります。例えば、次のようにします。

```
Bundle-Blueprint: OSGI-INF/blueprint.xml
```

5. ファイルを単一バンドル JAR ファイルにパッケージ化します。次のバンドル・ディレクトリー階層の例を参照してください。

```
MyBundle.jar
  /META-INF/manifest.mf
  /META-INF/objectGrid.xml
  /META-INF/objectGridDeployment.xml
  /OSGI-INF/blueprint/blueprint.xml
```

## タスクの結果

これで eXtreme Scale コンテナ・バンドルが作成されたので、Eclipse Equinox にインストールできます。コンテナ・バンドルが開始されると、eXtreme Scale サーバー・バンドル内の eXtreme Scale サーバー・ランタイム環境が、バンドルに定義されているパラメーターを使用して singleton eXtreme Scale サーバーを自動的に開始し、コンテナ・サーバーも開始します。バンドルは停止したり開始したりでき、それを受けてコンテナも停止または開始されます。サーバーは singleton であり、バンドルがはじめて開始されたときは停止しません。

## OSGi 構成管理でのサーバーの構成

OSGi 構成管理サービスを使用して、WebSphere eXtreme Scale コンテナ・サーバーを構成できます。

### このタスクについて

サーバーを構成するには、ManagedService 永続 ID (PID)

、com.ibm.websphere.xs.server がファイル・システム上の ObjectGrid サーバー・プロパティ・ファイルを参照するように設定します。コンテナを構成するには、ManagedServiceFactory PID、com.ibm.websphere.xs.container がファイル・システム上の ObjectGrid デプロイメント XML ファイルと ObjectGrid デプロイメント・ポリシー XML ファイルを参照するように設定します。

2 つの PID が構成管理サービス内で設定されると、eXtreme Scale サーバー・サービスがサーバーを自動的に初期化し、指定された構成ファイルを使用してコンテナを開始します。構成管理 PID は OSGi 構成ディレクトリーに保持されます。構成がクリアされなければ、フレームワークの再始動後もこれらの設定は保存されます。

構成管理プロパティを設定できるサード・パーティー・ユーティリティーがいくつか存在します。次のユーティリティーは本製品がサポートするツールの例です。

- Luminis OSGi Configuration Admin command line client では、コマンド行構成が可能です。
- Apache Felix File Install では、標準プロパティ・ファイル内に構成管理 PID 設定を指定できます。

Luminis の OSGi Configuration Administration command-line client を使用して、eXtreme Scale コンテナ・サーバーを構成するには、次のステップを実行します。

## 手順

1. OSGi コンソールで次のコマンドを実行して、ObjectGrid サーバー・プロパティ・ファイルの管理サービス PID を作成します。

```
osgi> cm create com.ibm.websphere.xs.server
osgi> cm put com.ibm.websphere.xs.server objectgrid.server.props /mypath/server.properties
```

2. OSGi コンソールで次のコマンドを実行して、ObjectGrid コンテナの管理サービス・ファクトリー永続 ID (PID) を作成します。

**重要:** **createf** 構成管理コマンドによって作成される PID を使用してください。次のコード・スニペット内で使用している PID は、あくまで例です。

```
osgi> cm createf com.ibm.websphere.xs.container
PID: com.ibm.websphere.xs.container-123456789-0
osgi> cm put com.ibm.websphere.xs.container-123456789-0 objectgridFile /mypath/objectGrid.xml
osgi> cm put com.ibm.websphere.xs.container-123456789-0 deploymentPolicyFile /mypath/deployment.xml
```

## タスクの結果

これで、Eclipse Equinox OSGi フレームワーク内で開始される eXtreme Scale コンテナ・サーバーが構成されました。

## 次のタスク

コンテナ・サーバーは、ServerFactory API と OSGi バンドル・アクティベーターを使用してプログラマチックに作成することもできます。ServerFactory API の使用については詳しくは、API 資料を参照してください。

---

## 第 7 章 管理



製品環境の管理と運用には、サーバーの開始と停止、データ・グリッドの可用性の管理、データ・センター障害からの復旧のシナリオなどがあります。カタログ・サーバーとコンテナ・サーバーの構成が終了したら、さまざまな方式を使用してサーバーを開始および停止できます。サーバーの開始および停止に使用する方式は、組み込みトポロジーを使用するか、スタンドアロン・トポロジーを使用するか、または WebSphere Application Server 内で稼働するトポロジーを使用するかによって異なります。

---

### スタンドアロン・サーバーの始動と停止

スタンドアロンのカタログ・サーバーおよびコンテナ・サーバーの始動と停止は、**startOgServer** スクリプトと **stopOgServer** スクリプト、または組み込みのサーバー API を使用して行うことができます。

#### 始める前に

外部のクライアント・セキュリティー・プロバイダーを使用しているスタンドアロン環境でサーバーを始動または停止する場合、**startOgServer** スクリプトまたは **stopOgServer** スクリプトを実行する前に、*CLIENT\_AUTH\_LIB* 環境変数を設定する必要があります。この環境変数の設定について詳しくは、601 ページの『スタンドアロン環境でのセキュア・サーバーの始動』を参照してください。

### スタンドアロン・サーバーの始動

スタンドアロン構成を実行しているとき、環境はカタログ・サーバー、コンテナ・サーバー、およびクライアント・プロセスで構成されています。また、組み込みのサーバー API を使用すれば、WebSphere eXtreme Scale サーバーを既存の Java アプリケーション内に組み込むことができます。これらのプロセスは手動で構成して開始する必要があります。

#### 始める前に

WebSphere Application Server がインストールされていない環境で WebSphere eXtreme Scale サーバーを始動できます。WebSphere Application Server を使用している場合は、287 ページの『WebSphere eXtreme Scale と WebSphere Application Server の構成』を参照してください。

### スタンドアロン・カタログ・サービスの開始

WebSphere Application Server で実行されていない分散 WebSphere eXtreme Scale 環境を使用している場合は、カタログ・サービスを手動で開始する必要があります。

#### 始める前に

- WebSphere Application Server を使用している場合、カタログ・サービスは既存のプロセス内で自動的に開始します。詳しくは、WebSphere Application Server でのカタログ・サービスの開始を参照してください。

## このタスクについて

**startOgServer** スクリプトを使用してカタログ・サービスを開始します。開始コマンドを呼び出すには、UNIX プラットフォームでは **startOgServer.sh** スクリプトを、また、Windows では **startOgServer.bat** を使用します。

カタログ・サービスは単一のプロセスで実行することができます。また、複数のカタログ・サーバーを組み込んでカタログ・サービス・ドメインを形成することもできます。実稼働環境では、高可用性のためにカタログ・サービス・ドメインが必要です。カタログ・サービス・ドメインの詳細については、「製品概要」のカタログ・サービス・ドメインに関する情報を参照してください。また、このスクリプトに追加のパラメーターを指定することで、オブジェクト・リクエスト・ブローカー (ORB) を特定のホストおよびポートにバインドしたり、ドメインを指定したり、セキュリティを使用可能にしたりできます。

### 手順

- 単一カタログ・サーバー・プロセスを開始します。

単一のカタログ・サーバーを始動するには、コマンド行から以下のコマンドを入力します。

1. bin ディレクトリーに移動します。

```
cd objectgridRoot/bin
```

2. **startOgServer** コマンドを実行します。

```
startOgServer.bat|sh catalogServer
```

使用可能なすべてのコマンド行パラメーターのリストについては、457 ページの『**startOgServer** スクリプト』を参照してください。実稼働環境では、単一の Java 仮想マシン (JVM) を使用してカタログ・サービスを実行しないようにしてください。カタログ・サービスが失敗すると、新規クライアントをデプロイ済みの eXtreme Scale に経路指定することも、新規 ObjectGrid インスタンスをドメインに追加することもできません。これらの理由により、Java 仮想マシンのセットを始動してカタログ・サービス・ドメインを実行するようにしてください。

- 複数のエンドポイントで構成されるカタログ・サービス・ドメインを開始します。

サーバーのセットを開始してカタログ・サービスを実行するには、**startOgServer** スクリプトで **-catalogServiceEndPoints** オプションを使用する必要があります。この引数は、*serverName:hostname:clientPort:peerPort* の形式のカタログ・サービス・エンドポイントのリストを受け入れます。以下の例は、カタログ・サービスをホストする 3 つの Java 仮想マシンのうち、最初のものを始動する方法を示しています。

1. bin ディレクトリーに移動します。

```
cd wxs_install_root/bin
```

2. **startOgServer** コマンドを実行します。

```
startOgServer.bat|sh cs1 -catalogServiceEndPoints  
cs1:MyServer1.company.com:6601:6602,  
cs2:MyServer2.company.com:6601:6602,  
cs3:MyServer3.company.com:6601:6602
```



この例では、MyServer1.company.com ホスト上の cs1 サーバーが始動されます。このサーバーの名前は、スクリプトに渡される最初の引数です。cs1 サーバーの初期化時に、catalogServiceEndpoints パラメーターが検査されて、このプロセスに割り振られるポートが決定されます。このリストは、cs1 サーバーが他のサーバー (cs2 および cs3) からの接続を受け入れることができるようにするためにも使用されます。

- リスト内の残りのカタログ・サーバーを開始するには、以下の引数を startOgServer スクリプトに渡します。MyServer2.company.com ホスト上の cs2 サーバーを始動します。

```
startOgServer.bat|sh cs2 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

MyServer3.company.com 上の cs3 を始動します。

```
startOgServer.bat|sh cs3 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

**重要:** 少なくとも 2 つのカタログ・サーバーを同時に始動してください。

データ・グリッドに入っているカタログ・サーバーは、それぞれのサーバーが、他のカタログ・サーバーがコア・グループに参加するのを休止して待つため、同時に始動する必要があります。データ・グリッド用に構成されているカタログ・サーバーは、グループ内の他のメンバーを識別するまで始動しません。カタログ・サーバーは、他のサーバーがいずれも使用可能にならないと、最終的にタイムアウトになります。

- ORB を特定のホストおよびポートにバインドします。

**catalogServiceEndpoints** 引数で定義されたポートを別にすれば、各カタログ・サービスも、オブジェクト・リクエスト・ブローカー (ORB) を使用して、クライアントおよびコンテナからの接続を受け入れます。デフォルトでは、ORB はローカル・ホストのポート 2809 で listen します。カタログ・サービス JVM で特定のホストおよびポートに ORB をバインドする場合は、**-listenerHost** および **-listenerPort** 引数を使用します。次の例は、ORB が

MyServer1.company.com のポート 7000 にバインドされた単一の JVM カタログ・サーバーを始動する方法を示しています。

```
startOgServer.sh catalogServer -listenerHost MyServer1.company.com
-listenerPort 7000
```

各 eXtreme Scale コンテナおよびクライアントにカタログ・サービス ORB エンドポイント・データが提供されるようにする必要があります。クライアントにはこのデータのサブセットのみが必要ですが、高可用性のために少なくとも 2 つのエンドポイントを使用するようにしてください。

- オプション: カタログ・サービス・ドメインに名前を付けます。

カタログ・サービスを開始するとき、カタログ・サービス・ドメイン・ネームは必要ではありません。しかし、マルチマスター・レプリカ生成を使用する場合、または同一プロセス・セット内で複数のカタログ・サービス・ドメインを使用する場合は、固有のカタログ・サービス・ドメイン・ネームを定義する必要があります。

ます。デフォルトのドメイン・ネームは `DefaultDomain` です。ドメインに名前を付けるには、**-domain** オプションを使用します。次の例は、ドメイン・ネーム `myDomain` を持つ単一カタログ・サービス JVM の開始方法を示しています。

```
startOgServer.sh catalogServer -domain myDomain
```

マルチマスター・レプリカ生成の構成の詳細については、320 ページの『複数データ・センター・トポロジーの構成』を参照してください。

- **セキュア・カタログ・サービスを開始します。** 詳しくは、601 ページの『スタンダード環境でのセキュア・サーバーの始動』を参照してください。
- **カタログ・サービスをプログラマチックに開始します。**

`CatalogServerProperties.setCatalogServer` メソッドによりフラグが立てられた JVM 設定は、eXtreme Scale のカタログ・サービスをホストできます。このメソッドは、eXtreme Scale サーバー・ランタイムに対して、サーバーの始動時にカタログ・サービスをインスタンス化することを指示します。以下のコードは、eXtreme Scale カatalog・サーバーをインスタンス化する方法を示しています。

```
CatalogServerProperties catalogServerProperties =  
    ServerFactory.getCatalogProperties();  
catalogServerProperties.setCatalogServer(true);  
  
//The getInstance() method will start the catalog service.  
Server server = ServerFactory.getInstance();
```

サーバーをプログラマチックに開始する方法について詳しくは、468 ページの『組み込みサーバー API を使用したサーバーの開始と停止』を参照してください。

## コンテナ・サーバーの始動

コンテナ・サーバーは、デプロイメント・トポロジーまたは `server.properties` ファイルを使用して、コマンド行から始動できます。

### このタスクについて

コンテナ・プロセスを開始するには、ObjectGrid XML ファイルが必要です。ObjectGrid XML ファイルは、コンテナがホストする eXtreme Scale サーバーを指定します。コンテナに渡される XML の各 ObjectGrid をホストするようにコンテナが装備されていることを確認してください。これらの ObjectGrid が必要とするクラスは、すべてコンテナのクラスパスになければなりません。ObjectGrid XML ファイルに関して詳しくは、`objectGrid.xsd` ファイルを参照してください。

### 手順

- **コマンド行からコンテナ・プロセスを開始します。**
  1. コマンド行から、`bin` ディレクトリーに移動します。

```
cd wxs_install_root/bin
```

2. 以下のコマンドを実行します。

```
startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml  
-catalogServiceEndpoints MyServer1.company.com:2809
```

**重要:** コンテナでは、**-catalogServiceEndpoints** オプションを使用して、カタログ・サービス上のオブジェクト・リクエスト・ブローカー (ORB) のホストと

ポートを参照します。カタログ・サービスでは、**-listenerHost** および **-listenerPort** オプションを使用して ORB バインディング用のホストとポートを指定するか、またはデフォルトのバインディングを受け入れます。コンテナを開始する場合は、**-catalogServiceEndPoints** オプションを使用して、カタログ・サービスで **-listenerHost** および **-listenerPort** オプションに渡される値を参照します。カタログ・サービスの開始時に **-listenerHost** および **-listenerPort** オプションが使用されなかった場合、ORB は、カタログ・サービスのローカル・ホストでポート 2809 にバインドします。カタログ・サービスで **-catalogServiceEndPoints** オプションに渡されたホストとポートを参照する場合に、**-catalogServiceEndPoints** オプションを使用しないでください。カタログ・サービスでは、**-catalogServiceEndPoints** オプションを使用して、静的サーバー構成に必要なポートを指定します。

このプロセスは、スクリプトに渡される最初の引数 `c0` によって識別されます。`companyGrid.xml` を使用してコンテナを開始してください。カタログ・サーバー ORB が、コンテナとは異なるホストで実行されているか、またはデフォルト以外のポートを使用している場合は、**-catalogServiceEndPoints** 引数を使用してその ORB に接続する必要があります。この例では、単一のカタログ・サービスが `MyServer1.company.com` のポート 2809 で実行されているものと仮定します。

- **デプロイメント・ポリシーを使用してコンテナを開始します。**

必須ではありませんが、コンテナの開始時には、デプロイメント・ポリシーが推奨されます。デプロイメント・ポリシーは、eXtreme Scale の区画化およびレプリカ生成をセットアップするために使用されます。デプロイメント・ポリシーは、配置方法に影響を与えるためにも使用されます。前述の例では、デプロイメント・ポリシー・ファイルが提供されなかったため、レプリカ生成、区画化、および配置に関して、すべてのデフォルト値を受け取ります。したがって、`CompanyGrid` にあるマップは 1 つの `mapSet` 内に入ります。この `mapSet` は区画化も複製もされません。デプロイメント・ポリシー・ファイルについて詳しくは、デプロイメント・ポリシー記述子 XML ファイルを参照してください。次の例では、`companyGridDpReplication.xml` ファイルを使用してコンテナ JVM (`c0 JVM`) を開始します。

1. コマンド行から、`bin` ディレクトリーに移動します。

```
cd wxs_install_root/bin
```

2. 以下のコマンドを実行します。

```
startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml  
-deploymentPolicyFile ../xml/companyGridDpReplication.xml  
-catalogServiceEndPoints MyServer1.company.com:2809
```

**注:** Java クラスが特定のディレクトリーに保管されている場合、あるいはローダーまたはエージェントを使用している場合は、`StartOgServer` スクリプトを変更する代わりに、`-jvmArgs-cp C:¥ . . . ¥DirectoryP0J0s¥P0J0s.jar` というように引数を指定してサーバーを起動することができます。

`companyGridDpReplication.xml` ファイルでは、単一のマップ・セットにすべてのマップが含まれています。この `mapSet` は 10 個の区画に分割されます。各区画には 1 つの同期レプリカが存在し、非同期レプリカは存在しません。また、`companyGrid.xml` ObjectGrid XML ファイルとペアになる

companyGridDpReplication.xml デプロイメント・ポリシーを使用するコンテナーは、CompanyGrid 断片をホストできます。別のコンテナー JVM (c1 JVM) を開始します。

1. コマンド行から、bin ディレクトリーに移動します。

```
cd wxs_install_root/bin
```

2. 以下のコマンドを実行します。

```
startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplication.xml
-catalogServiceEndPoints MyServer1.company.com:2809
```

各デプロイメント・ポリシーには、1 つ以上の objectgridDeployment エレメントが含まれています。コンテナーが開始されると、コンテナーは、デプロイメント・ポリシーをカタログ・サービスに公開します。カタログ・サービスは各 objectgridDeployment エレメントを検査します。objectgridName 属性が、前に受信された objectgridDeployment エレメントの objectgridName 属性と一致する場合、最新の objectgridDeployment エレメントは無視されます。特定の objectgridName 属性用に受信された最初の objectgridDeployment エレメントがマスターとして使用されます。例えば、c2 JVM が、mapSet を異なる数の区画に分割するデプロイメント・ポリシーを使用するとします。

#### companyGridDpReplicationModified.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="CompanyGrid">
    <mapSet name="mapSet1" numberOfPartitions="5"
      minSyncReplicas="1" maxSyncReplicas="1"
      maxAsyncReplicas="0">
      <map ref="Customer" />
      <map ref="Item" />
      <map ref="OrderLine" />
      <map ref="Order" />
    </mapSet>
  </objectgridDeployment>

</deploymentPolicy>
```

これで、3 番目の JVM である c2 JVM を開始できます。

1. コマンド行から、bin ディレクトリーに移動します。

```
cd wxs_install_root/bin
```

2. 以下のコマンドを実行します。

```
startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
```

c2 JVM 上のコンテナーが、mapSet1 に 5 つの区画を指定するデプロイメント・ポリシーで開始されます。しかし、カタログ・サービスは、CompanyGrid の objectgridDeployment のマスター・コピーを既に保持しています。c0 JVM は開始されたときに、この mapSet に 10 個の区画を指定しました。c0 が、デプロイメント・ポリシーを開始および公開する最初のコンテナーであったため、c0 のデ

プロイメント・ポリシーがマスターになりました。したがって、後続のデプロイメント・ポリシー内の CompanyGrid に等しい objectgridDeployment 属性値はすべて無視されます。

- **サーバー・プロパティ・ファイルを使用してコンテナを開始します。**

サーバー・プロパティ・ファイルを使用して、コンテナでのトレースをセットアップし、セキュリティーを構成することができます。次のコマンドを実行し、サーバー・プロパティ・ファイルを使用してコンテナ c3 を開始します。

1. コマンド行から、bin ディレクトリーに移動します。

```
cd wxs_install_root/bin
```

2. 以下のコマンドを実行します。

```
startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml  
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml  
-catalogServiceEndPoints MyServer1.company.com:2809  
-serverProps ../serverProps/server.properties
```

server.properties ファイルの例を次に示します。

```
server.properties  
workingDirectory=  
traceSpec==all=disabled  
systemStreamToFileEnabled=true  
enableMBeans=true  
memoryThresholdPercentage=50
```

これは、セキュリティーを有効にしていない基本的なサーバー・プロパティ・ファイルです。server.properties ファイルに関して詳しくは、サーバー・プロパティ・ファイルを参照してください。

- **コンテナ・サーバーをプログラマチックに始動します。**

コンテナ・サーバーをプログラマチックに始動する方法について詳しくは、468 ページの『組み込みサーバー API を使用したサーバーの開始と停止』を参照してください。

## startOgServer スクリプト

**startOgServer** スクリプトはコンテナ・サーバーとカタログ・サーバーを始動します。サーバーの始動時に各種パラメーターを使用して、トレースを使用可能にしたり、ポート番号を指定するなど、さまざまな設定を行うことができます。

### 目的

**startOgServer** スクリプトを使用してサーバーを始動することができます。

### ロケーション

**startOgServer** スクリプトは、ルート・ディレクトリーの bin ディレクトリーにあります。例えば、次のとおりです。

```
cd wxs_install_root/bin
```

**注:** Java クラスが特定のディレクトリーに保管されている場合、あるいはローダーまたはエージェントを使用している場合は、startOgServer スクリプトを変更する代

わりに、`-jvmArgs-cp C:¥ . . . ¥DirectoryPOJ0s¥POJ0s.jar` というように引数を指定してサーバーを起動することができます。

## カタログ・サーバーの場合の使用法

カタログ・サーバーを始動する場合:

### Windows

```
startOgServer.bat <server> [options]
```

### UNIX

```
startOgServer.sh <server>[options]
```

デフォルトの構成済みカタログ・サーバーを始動するには、以下のコマンドを使用します。

### Windows

```
startOgServer.bat catalogServer
```

### UNIX

```
startOgServer.sh catalogServer
```

## カタログ・サーバーの始動のオプション

次のパラメーターはすべてオプションです。

カタログ・サーバーの始動のためのパラメーター:

**-catalogServiceEndPoints <serverName:hostName:clientPort:peerPort>**

コンテナでは、カタログ・サービス上のオブジェクト・リクエスト・ブローカー (ORB) のホストとポートを参照します。各属性の定義は次のとおりです。

### serverName

起動しようとしているプロセスを識別する名前を指定します。

### hostName

サーバーを起動するコンピューターのホスト名を指定します。

### clientPort

ピア・カタログ・サービス通信に使用されるポートを指定します。

### peerPort

この値は、`haManagerPort` と同じです。ピア・カタログ・サービス通信に使用されるポートを指定します。

次の例は、`cs1` カタログ・サーバーを始動するものです。このサーバーは、`cs2` および `cs3` サーバーと同じカタログ・サービス・ドメイン内にあります。

```
startOgServer.bat|sh cs1 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

**-clusterSecurityFile <cluster security xml file>**

ハード・ディスク上の `objectGridSecurity.xml` ファイルを指定します。このフ



ファイルは、すべてのサーバー (カタログ・サーバーおよびコンテナ・サーバーを含む) に共通するセキュリティー・プロパティーを記述します。プロパティー例の 1 つは、ユーザー・レジストリーおよび認証メカニズムを表すオーセンティケーター構成です。

例:/opt/xs/ogsecurity.xml

**-clusterSecurityUrl <cluster security xml URL>**

objectGridSecurity.xml ファイルを、ハード・ディスクまたはネットワーク上のこのファイルへの URL として指定します。このファイルは、すべてのサーバー (カタログ・サーバーおよびコンテナ・サーバーを含む) に共通するセキュリティー・プロパティーを記述します。プロパティー例の 1 つは、ユーザー・レジストリーおよび認証メカニズムを表すオーセンティケーター構成です。

例: file:///opt/xs/ogsecurity.xml

**-domain <domain name>**

このカタログ・サーバーのカタログ・サービス・ドメインの名前を指定します。カタログ・サービス・ドメインには、可用性の高いカタログ・サーバーのグループが含まれます。単一ドメインの各カタログ・サーバーは、**-domain** パラメーターに同じ値を指定する必要があります。

**-JMXConnectorPort <port>**

Java Management Extensions (JMX) サービスのバインド先の Secure Sockets Layer (SSL) ポートを定義します。

**-haManagerPort <port>**

HA マネージャーが使用するポート番号を指定します。このプロパティーが設定されていない場合、空きポートが選択されます。このプロパティーは、WebSphere Application Server 環境では無視されます。

**-JMXServicePort <port>**

MBean サーバーが Java Management Extensions (JMX) との通信を listen するポート番号を指定します。JMXServicePort プロパティーは、JMX 用の非 SSL ポートを指定します。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**JMXServicePort** とポート番号を明示的に指定してください。このプロパティーは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。(スタンドアロン環境の場合のみ必須。)

デフォルト: 1099 (カタログ・サーバーの場合)

**-jvmArgs <JVM arguments>**

JVM 引数 (複数可) を指定します。 **-jvmArgs** オプションより後にあるオプションは、すべてサーバー Java 仮想マシン (JVM) を始動するために使用されるものです。 **-jvmArgs** パラメーターを使用するときは、最後に指定されるスクリプト引数 (オプション) になるようにしてください。

例:-jvmArgs -Xms256M -Xmx1G

**-listenerHost <host name>**

Internet Inter-ORB Protocol (IIOP) との通信用に、オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のホスト名を指定します。この値は、完全修飾ドメイン名または IP アドレスである必要があります。ご使用の構成に複数のネットワーク・カードが含まれている場合は、リスナー・ホストとリスナー・

ポートを設定し、バインドする IP アドレスを JVM 内のオブジェクト・リクエスト・ブローカーに通知します。使用する IP アドレスを指定しなければ、接続タイムアウトや異常な API 障害、クライアントがハングしたように見える状態などの症状が現れることがあります。**デフォルト: localhost**

**-listenerPort <port>**

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポート番号を指定します。この設定は、コンテナおよびクライアントが ORB を介してカタログ・サービスと通信するように構成します。WebSphere Application Server では、listenerPort は BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。**デフォルト: 2809**

**-quorum true|false**

カタログ・サービスのクォラムを使用可能にします。使用可能なコンテナ・サーバー上の区画を移動する前に、クォラムを使用して、大半のカタログ・サービス・ドメインを確実に使用可能にします。クォラムを使用可能にするには、この値を true または enabled に設定します。デフォルト値は disabled です。このプロパティは、カタログ・サービスにのみ適用されます。詳しくは、カタログ・サーバー・クォラムを参照してください。

**-script <script file>**

カタログ・サーバーまたはコンテナを始動し、その後必要に応じてパラメータ化または編集を行うために指定するコマンドのカスタム・スクリプトの場所を指定します。

**-serverProps <server properties file>**

サーバー固有のセキュリティー・プロパティが含まれているサーバー・プロパティ・ファイル指定します。このプロパティに対して指定されるファイル名の形式は、単なるプレーン・ファイル・パス形式です。例えば、  
c:/tmp/og/catalogserver.props などです。

**-traceSpec <trace specification>**

コンテナ・サーバーのトレースおよびトレース仕様ストリングを使用可能にします。トレースは、デフォルトで使用不可に設定されています。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。例:

- ObjectGrid=all=enabled
- ObjectGrid\*=all=enabled

**-traceFile <trace file>**

トレース情報を書き込むファイル名を指定します。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。例:

../logs/c4Trace.log

**-timeout <seconds>**

サーバーの始動がタイムアウトになる秒数を指定します。

**コンテナ・サーバーの場合の使用法** Windows

```
startOgServer.bat <server> -objectgridFile <xml file>  
-deploymentPolicyFile <xml file> [options]
```

#### Windows

```
startOgServer.bat <server> -objectgridUrl <xml URL>  
-deploymentPolicyUrl <xml URL> [options]
```

#### UNIX

```
startOgServer.sh <server> -objectgridFile <xml file>  
-deploymentPolicyFile <xml file> [options]
```

#### UNIX

```
startOgServer.sh <server> -objectgridUrl <xml URL>  
-deploymentPolicyUrl <xml URL> [options]
```

## コンテナ・サーバーのオプション

### **-catalogServiceEndpoints<hostName:port,hostName:port>**

カタログ・サービス上のオブジェクト・リクエスト・ブローカー (ORB) ホストおよびポートを指定します。

デフォルト: localhost:2809

### **-deploymentPolicyFile <deployment policy xml file>**

ハード・ディスク上のデプロイメント・ポリシー・ファイルへのパスを指定します。デプロイメント・ポリシーは、区画化およびレプリカ生成をセットアップするために使用されます。デプロイメント・ポリシーは、配置方法に影響を与えるためにも使用されます。

例: ../xml/SimpleDP.xml

### **-deploymentPolicyUrl <deployment policy url>**

ハード・ディスクまたはネットワーク上のデプロイメント・ポリシー・ファイルの URL を指定します。デプロイメント・ポリシーは、区画化およびレプリカ生成をセットアップするために使用されます。デプロイメント・ポリシーは、配置方法に影響を与えるためにも使用されます。

例: file://xml/SimpleDP.xml

### **-JMXConnectorPort <port>**

Java Management Extensions (JMX) サービスのバインド先の Secure Sockets Layer (SSL) ポートを定義します。

### **-JMXServicePort <port>**

MBean サーバーが Java Management Extensions (JMX) との通信を listen するポート番号を指定します。JMXServicePort プロパティは、JMX 用の非 SSL ポートを指定します。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**JMXServicePort** とポート番号を明示的に指定してください。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。(スタンドアロン環境の場合のみ必須。)デフォルト: 1099

### **-jvmArgs <JVM arguments>**

JVM 引数 (複数可) を指定します。 **-jvmArgs** オプションより後にあるオプションは、すべてサーバー Java 仮想マシン (JVM) を始動するために使用される

ものです。**-jvmArgs** パラメーターを使用するときは、最後に指定されるスクリプト引数 (オプション) になるようにしてください。

**例:-jvmArgs -Xms256M -Xmx1G**

**-listenerHost <host name>**

Internet Inter-ORB Protocol (IIOP) との通信用に、オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のホスト名を指定します。この値は、完全修飾ドメイン名または IP アドレスである必要があります。ご使用の構成に複数のネットワーク・カードが含まれている場合は、リスナー・ホストとリスナー・ポートを設定し、バインドする IP アドレスを JVM 内のオブジェクト・リクエスト・ブローカーに通知します。使用する IP アドレスを指定しなければ、接続タイムアウトや異常な API 障害、クライアントがハングしたように見える状態などの症状が現れることがあります。**デフォルト: localhost**

**-listenerPort <port>**

オブジェクト・リクエスト・ブローカー (ORB) がバインドする先のポート番号を指定します。この設定は、コンテナおよびクライアントが ORB を介してカタログ・サービスと通信するように構成します。WebSphere Application Server では、listenerPort は BOOTSTRAP\_ADDRESS ポート構成によって継承されます。このプロパティーは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。**デフォルト: 2809**

**-objectgridFile <ObjectGrid descriptor xml file>**

ObjectGrid 記述子ファイルへのパスを指定します。ObjectGrid XML ファイルは、コンテナがホストする eXtreme Scale サーバーを指定します。

**-objectgridUrl <ObjectGrid descriptor url>**

ObjectGrid 記述子ファイルの URL を指定します。ObjectGrid XML ファイルは、コンテナがホストする eXtreme Scale サーバーを指定します。

**-script <script file>**

カタログ・サーバーまたはコンテナを始動し、その後必要に応じてパラメータ化または編集を行うために指定するコマンドのカスタム・スクリプトの場所を指定します。

**-serverProps <server properties file>**

サーバー・プロパティー・ファイルへのパスを指定します。

**例:../security/server.props**

**-timeout <seconds>**

サーバーの始動がタイムアウトになる秒数を指定します。

**-traceFile <trace file>**

トレース情報を書き込むファイル名を指定します。このプロパティーは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。**例:**

**../logs/c4Trace.log**

**-traceSpec <trace specification>**

コンテナ・サーバーのトレースおよびトレース仕様ストリングを使用可能にします。トレースは、デフォルトで使用不可に設定されています。このプロパティーは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。**例:**

• ObjectGrid=all=enabled

- ObjectGrid\*=all=enabled

#### -zone <zone name>

サーバー内のすべてのコンテナのために使用するゾーンを指定します。ゾーンの構成について詳しくは、276 ページの『優先ゾーン・ルーティング』 「製品概要」でゾーンについての情報を参照してください。

## スタンドアロン・サーバーの停止

**stopOgServer** スクリプトを使用して、eXtreme Scale サーバー・プロセスを停止できます。

### このタスクについて

bin ディレクトリーに移動して、**stopOgServer** スクリプトを実行します。

```
cd wxs_install_root/bin
```

### 手順

- 単一のコンテナ・サーバーを停止します。

**stopOgServer** スクリプトを実行してコンテナ・サーバーを停止します。単一のコンテナ・サーバーを停止するときは、このコマンドのみを使用します。複数のコンテナ・サーバー上で連続して単一カタログ・サーバーの停止コマンドを実行すると、断片配置についてパフォーマンスおよびチャーンの問題が発生する可能性があります。

```
stopOgServer containerServer -catalogServiceEndPoints MyServer1.company.com:2809
```

**重要:** **-catalogServiceEndPoints** オプションは、コンテナの開始に使用された **-catalogServiceEndPoints** オプションの値と一致している必要があります。コンテナの開始に **-catalogServiceEndPoints** を使用しなかった場合には、恐らく、デフォルト値は localhost またはホスト名と 2809 (カタログ・サービスに接続するための ORB ポート) です。それ以外の場合は、カタログ・サービスで **-listenerHost** および **-listenerPort** に渡した値を使用します。カタログ・サービスの開始時に **-listenerHost** および **-listenerPort** オプションが使用されなかった場合、ORB はカタログ・サービスのために localhost のポート 2809 にバインドします。

- 複数のコンテナ・サーバーを停止します。

同時に複数のコンテナ・サーバーを停止するときに断片配置のチャーンおよびパフォーマンスの問題を回避するために、次のコマンド形式を使用します。コンテナ・サーバーのリストはコンマで区切ります。

```
stopOgServer containerServer0,containerServer1,containerServer2  
-catalogServiceEndPoints MyServer1.company.com:2809
```

特定のゾーンまたはホスト上のすべてのコンテナを停止する場合は、**-teardown** パラメーターを使用します。詳しくは、466 ページの『**xscmd** ユーティリティーによるサーバーの正常停止』を参照してください。

- カタログ・サーバーを停止します。

**stopOgServer** スクリプトを実行してカタログ・サーバーを停止します。

```
stopOgServer.sh catalogServer -catalogServiceEndPoints MyServer1.company.com:2809
```

**重要:** カタログ・サービスを停止するときは、**-catalogServiceEndpoints** オプションを使用して、カタログ・サービス上のオブジェクト・リクエスト・ブローカー (ORB) ホストおよびポートを参照します。カタログ・サービスでは、**-listenerHost** および **-listenerPort** オプションを使用して ORB バインディング用のホストとポートを指定するか、またはデフォルトのバインディングを受け入れます。カタログ・サービスの開始時に **-listenerHost** および **-listenerPort** オプションが使用されなかった場合、ORB はカタログ・サービスのために localhost のポート 2809 にバインドします。**-catalogServiceEndpoints** オプションは、カタログ・サービスを停止するときは、カタログ・サービスを開始したときと異なります。

デフォルト・ポートを使用しなかった場合には、カタログ・サービスを開始するには、ピア・アクセス・ポートおよびクライアント・アクセス・ポートが必要です。カタログ・サービスの停止には、ORB ポートのみが必要になります。

- **Web コンソール・サーバーを停止します。** Web コンソール・サーバーを停止するには、**stopConsoleServer.bat|sh** スクリプトを実行します。このスクリプトは、インストール済み環境の `wxs_install_root/ObjectGrid/bin` ディレクトリにあります。詳しくは、507 ページの『Web コンソールの開始とログオン』を参照してください。
- **サーバー停止プロセスのトレースを使用可能にします。**

コンテナを停止できない場合は、トレースを使用可能にして問題のデバッグに役立てることができます。サーバーの停止時にトレースを使用可能にするには、**-traceSpec** および **-traceFile** パラメーターを停止コマンドに追加します。**-traceSpec** パラメーターは使用可能にするトレースのタイプを指定し、**-traceFile** パラメーターはそのトレース・データのために作成して使用するファイルのパスと名前を指定します。

1. コマンド行から、bin ディレクトリに移動します。

```
cd wxs_install_root/bin
```

2. トレースを使用可能にして **stopOgServer** スクリプトを実行します。

```
stopOgServer.sh c4 -catalogServiceEndpoints MyServer1.company.com:2809  
-traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
```

トレースが取得されたら、ポート競合、欠落クラス、欠落または正しくない XML ファイルに関連したエラーがないか、またはスタック・トレースがないか調べます。推奨される開始トレース仕様は、以下のとおりです。

- ObjectGrid=all=enabled
- ObjectGrid\*=all=enabled

すべてのトレース仕様オプションについては、620 ページの『トレース・オプション』を参照してください。

- **組み込みサーバーをプログラムで停止します。**

組み込みサーバーをプログラムで停止することについて詳しくは、468 ページの『組み込みサーバー API を使用したサーバーの開始と停止』を参照してください。



## stopOgServer スクリプト

**stopOgServer** スクリプトは、カタログ・サーバーとコンテナ・サーバーを停止します。

### 目的

**stopOgServer** スクリプトを使用してサーバーを停止します。サーバーの名前と、サーバーのカタログ・サービス・エンドポイントを指定する必要があります。

### ロケーション

**stopOgServer** スクリプトは、root ディレクトリーの bin ディレクトリーにあります。例えば、次のとおりです。

```
cd wxs_install_root/bin
```

### 使用法

カタログ・サーバーまたはコンテナ・サーバーを停止する場合: Windows

```
stopOgServer.bat <server_name> -catalogServiceEndPoints  
<csHost:csListenerPort,csHost:csListenerPort> [options]
```

UNIX

```
stopOgServer.sh <server_name> -catalogServiceEndPoints  
<csHost:csListenerPort,csHost:csListenerPort> [options]
```

### オプション

**-catalogServiceEndPoints <csHost:csListenerPort, csHost:csListenerPort...>**

オブジェクト・リクエスト・ブローカー (ORB) ホストおよびポート番号を指定します。

**コンテナ・サーバーの場合:** カタログ・サービス・エンドポイントのリストは、コンテナ・サーバーの始動に使用されたリストと同じである必要があります。コンテナ・サーバーを始動するときはこのオプションを指定しなかった場合は、デフォルト値の localhost:2809 を使用してください。

**カタログ・サーバーの場合:** カタログ・サービスを停止する場合は、カタログ・サービスを開始したときに **-listenerHost** オプションおよび **-listenerPort** オプションに指定した値を使用してください。カタログ・サーバーを始動するときこれらのオプションを指定しなかった場合は、デフォルト値の localhost:2809 を使用してください。カタログ・サービスを停止するときに使用する **-catalogServiceEndPoints** 値は、カタログ・サービスを開始するときのものとは異なります。

**-clientSecurityFile <security properties file>**

クライアントのセキュリティー・プロパティーを定義するクライアント・プロパティー・ファイルへのパスを指定します。このファイルのセキュリティー設定について詳しくは、クライアント・プロパティー・ファイルを参照してください。

**-traceSpec <trace specification>**

コンテナ・サーバーのトレースおよびトレース仕様ストリングを使用可能にします。トレースは、デフォルトで使用不可に設定されています。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。例:

- ObjectGrid=all=enabled
- ObjectGrid\*=all=enabled

#### **-traceFile <trace file>**

トレース情報を書き込むファイル名を指定します。このプロパティは、コンテナ・サーバーとカタログ・サービスの両方に適用されます。例:

```
../logs/c4Trace.log
```

#### **-jvmArgs <JVM arguments>**

JVM 引数 (複数可) を指定します。 **-jvmArgs** オプションより後にあるオプションは、すべてサーバー Java 仮想マシン (JVM) を始動するために使用されるものです。 **-jvmArgs** パラメーターを使用するときは、最後に指定されるスクリプト引数 (オプション) になるようにしてください。

例:**-jvmArgs -Xms256M -Xmx1G**

### **xscmd ユーティリティーによるサーバーの正常停止**

**xscmd** ユーティリティーと **-c teardown** コマンドを使用して、カタログおよびコンテナ・サーバーのリストまたはグループを停止できます。このコマンドは、プロセスが停止または強制終了されるときに通常発生するカタログ・サービスによる不要な配置およびリカバリーのアクションを回避して、データ・グリッドのすべてまたは一部のシャットダウンを単純にします。

#### **手順**

- 特定のリストのサーバーを停止します。

**-teardown** パラメーターの後に、サーバーのリストを指定します。

```
xscmd -c teardown
```

- 特定のゾーン内のすべてのサーバーを停止します。

**-z** パラメーターを使用し、ゾーンの名前を指定します。カタログ・サーバーはゾーン内で実行中のサーバーを判別し、**xscmd** ユーティリティーは、サーバーをシャットダウンする前に、選択されたゾーン内にあるサーバーのリストを表示してプロンプトを出します。

```
xscmd -c teardown -z zone_name
```

- 特定のホスト上のすべてのサーバーを停止します。

**-hf** パラメーターを使用し、ホストの名前を指定します。例えば、**myhost.mycompany.com** 上のすべてのサーバーをシャットダウンするには、**-hf myhost.mycompany.com** と入力します。カタログ・サーバーはホスト上で実行中のサーバーを判別し、**xscmd** ユーティリティーは、サーバーをシャットダウンする前に、選択されたホスト上にあるサーバーのリストを表示してプロンプトを出します。

```
xscmd -teardown -hf <host_name>
```

---

## WebSphere Application Server 環境でのサーバーの開始と停止

WebSphere Application Server または WebSphere Application Server Network Deployment 環境では、カタログ・サーバーとコンテナ・サーバーを自動的に開始できます。

### 始める前に

WebSphere Application Server 上で実行するカタログ・サーバーとコンテナ・サーバーを構成します。

- 288 ページの『WebSphere Application Server でのカタログ・サービスの構成』
- 313 ページの『WebSphere Application Server のコンテナ・サーバーの構成』

### このタスクについて

WebSphere Application Server 内のカタログ・サーバーとコンテナ・サーバーのライフサイクルは、これらのサーバーを実行するプロセスとリンクします。

### 手順

- **WebSphere Application Server でのカタログ・サービスの開始:**

カタログ・サーバーのライフサイクルは、WebSphere Application Server プロセスに連動します。WebSphere Application Server のカタログ・サービス・ドメインの構成が終了したら、カタログ・サービス・ドメインのメンバーとして定義した各サーバーを再始動してください。カタログ・サービス・ドメインに関連付けたサーバー上のカタログ・サービスが自動的に開始します。カタログ・サービスは、WebSphere Application Server のエディションに応じて、次のいずれかのシナリオに従い自動的に開始することもできます。

- **基本 WebSphere Application Server:** コンテナ・サーバーとカタログ・サービスを自動的に開始するようアプリケーションを構成できます。このフィーチャーにより、カタログ・サービスを明示的に開始する必要がなくなるため、Rational Application Developer などの開発環境での単体テストが簡略化されます。詳しくは、314 ページの『コンテナ・サーバーの自動始動のための WebSphere Application Server アプリケーションの構成』を参照してください。
- **WebSphere Application Server Network Deployment:** デプロイメント・マネージャー・ノードに WebSphere eXtreme Scale がインストールされていて、デプロイメント・マネージャー・プロファイルが拡張されている場合、カタログ・サービスはデプロイメント・マネージャー・プロセス内で自動的に開始します。詳しくは、288 ページの『WebSphere Application Server でのカタログ・サービスの構成』を参照してください。

- **WebSphere Application Server でのコンテナ・サーバーの始動:**

コンテナ・サーバーのライフサイクルは、WebSphere Application Server アプリケーションに連動します。構成済みアプリケーションを開始すると、コンテナ・サーバーも開始します。

- **サーバーのデータ・グリッド全体の停止:**

アプリケーションおよび関連のアプリケーション・サーバーを停止することでカタログ・サーバーとコンテナ・サーバーを停止することもできますが、**xscmd** ユーティリティーまたは **MBean** を使用してデータ・グリッド全体を停止することもできます。

– **xscmd** ユーティリティーの場合

データ・グリッド全体を停止する方法については、466 ページの『**xscmd** ユーティリティーによるサーバーの正常停止』を参照してください。

– **MBean** の場合

`PlacementServiceMBean` `MBean` の `tearDownServers` オペレーションを使用します。

---

## 組み込みサーバー API を使用したサーバーの開始と停止

WebSphere eXtreme Scale では、組み込みサーバーおよびコンテナのライフサイクルの管理にプログラマチック API を使用できます。コマンド行オプションやファイル・ベースのサーバー・プロパティでも構成可能な任意のオプションを使用して、プログラムでサーバーを構成できます。コンテナ・サーバー、カタログ・サービス、またはその両方として、組み込みサーバーの構成が可能です。

### 始める前に

既存の Java 仮想マシン内からコードを実行するためのメソッドが必要です。eXtreme Scale クラスが、クラス・ローダー・ツリーから利用可能でなければなりません。

### このタスクについて

管理 API を使用して多くの管理タスクを実行できます。API の一般的な使用法の 1 つとして、Web アプリケーションの状態を保管する内部サーバーとしての使用があります。Web サーバーは、組み込み WebSphere eXtreme Scale サーバーを始動し、コンテナ・サーバーをカタログ・サービスに報告することができ、サーバーは、より幅広い分散グリッドのメンバーとして追加されます。この使用方法では、本来は揮発性のデータ・ストアにスケラビリティと高可用性が提供されます。

組み込み eXtreme Scale サーバーの全ライフサイクルをプログラムで制御できます。例は、できる限り汎用的にして、概要を説明したステップの直接的なコードの例のみを示しています。

### 手順

1. `ServerFactory` クラスから `ServerProperties` オブジェクトを取得し、必要なオプションを構成します。

すべての eXtreme Scale サーバーに、一連の構成可能なプロパティがあります。コマンド行からサーバーが始動されると、それらのプロパティはデフォルトに設定されますが、外部ソースまたはファイルを指定することによって、複数のプロパティをオーバーライドすることができます。組み込み有効範囲では、`ServerProperties` オブジェクトでプロパティを直接設定できます。これらのプロパティは、`ServerFactory` クラスからサーバー・インスタンスを取得する前に設

定する必要があります。以下の例のスニペットでは、`ServerProperties` オブジェクトを取得して `CatalogServiceBootstrap` フィールドを設定し、複数のオプション・サーバー設定を初期化します。構成可能な設定のリストについては、API 資料を参照してください。

```
ServerProperties props = ServerFactory.getServerProperties();
props.setCatalogServiceBootstrap("host:port"); // required to connect to specific catalog service
props.setServerName("ServerOne"); // name server
props.setTraceSpecification("com.ibm.ws.objectgrid=all=enabled"); // Sets trace spec
```

2. サーバーをカタログ・サービスにする場合には、`CatalogServerProperties` オブジェクトを取得します。

すべての組み込みサーバーが、カタログ・サービスまたはコンテナ・サーバー、あるいはその両方になることができます。以下の例では、`CatalogServerProperties` オブジェクトを取得し、カタログ・サービス・オプションを使用可能にし、さまざまなカタログ・サービス設定を構成します。

```
CatalogServerProperties catalogProps = ServerFactory.getCatalogProperties();
catalogProps.setCatalogServer(true); // false by default, it is required to set as a catalog service
catalogProps.setQuorum(true); // enables / disables quorum
```

3. `ServerFactory` クラスから `Server` インスタンスを取得します。 `Server` インスタンスは、グリッド内のメンバーシップの管理に参与するプロセス・スコープの singleton です。このインスタンスが初期化された後、このプロセスが接続され、グリッド内の他のサーバーで高度に利用可能になります。以下の例は、`Server` インスタンスの作成方法を示しています。

```
Server server = ServerFactory.getInstance();
```

上記の例において、`ServerFactory` クラスは、`Server` インスタンスを返す静的メソッドを提供します。`ServerFactory` クラスは、`Server` インスタンスを取得するための唯一のインターフェースとして意図されています。そのため、このクラスは必ず、インスタンスが singleton であるか、または各 JVM または独立したクラス・ローダーの 1 つインスタンスであるようにします。`getInstance` メソッドで `Server` インスタンスを初期化します。インスタンスを初期化する前にすべてのサーバー・プロパティの構成が必要です。`Server` クラスは、新規の `Container` インスタンスの作成に参与します。`ServerFactory` クラスと `Server` クラスの両方を使用して、組み込み `Server` インスタンスのライフサイクルを管理できます。

4. `Server` インスタンスを使用して `Container` インスタンスを開始します。

断片を組み込みサーバーに配置するには、その前に、サーバーにコンテナを作成する必要があります。`Server` インターフェースの `createContainer` メソッドは、`DeploymentPolicy` 引数を使用します。以下の例では、取得したサーバー・インスタンスを使用して、作成した `DeploymentPolicy` ファイルでコンテナを作成します。`Container` は、シリアライゼーションのためにアプリケーション・バイナリーが使用可能になっているクラス・ローダーを必要とします。これらのバイナリーは、使用するクラス・ローダーを `Thread` コンテキスト・クラス・ローダーに設定して `createContainer` メソッドを呼び出すことによって、使用可能になります。

```
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(new
    URL("file://urltodeployment.xml"),
    new URL("file://urltoobjectgrid.xml"));
Container container = server.createContainer(policy);
```

5. コンテナを除去してクリーンアップします。

コンテナ・サーバーを除去してクリーンアップするには、取得した Container インスタンスで `teardown` メソッドを実行します。コンテナで `teardown` メソッドを実行すると、コンテナを適切にクリーンアップし、組み込みサーバーからコンテナを除去します。

コンテナのクリーンアップ処理には、そのコンテナ内のすべての断片の移動と終了処理が含まれます。各サーバーには、多くのコンテナと断片が含まれます。コンテナをクリーンアップしても、親の Server インスタンスのライフサイクルには影響しません。以下の例は、サーバーで `teardown` メソッドを実行する方法を示しています。`teardown` メソッドは、ContainerMBean インターフェースを通して使用可能になります。ContainerMBean インターフェースを使用することによって、このコンテナに対するプログラムによるアクセスがもうなくても、その MBean でコンテナを除去してクリーンアップすることができます。また、`terminate` メソッドが Container インターフェースに存在しますが、どうしても必要でない限り、このメソッドは使用しないでください。このメソッドは強制力が強く、断片の適切な移動とクリーンアップの調整は行いません。

```
container.teardown();
```

#### 6. 組み込みサーバーを停止します。

組み込みサーバーを停止するときには、そのサーバーで実行されているコンテナと断片も停止します。組み込みサーバーの停止時には、開いているすべての接続をクリーンアップして、すべての断片を移動または終了処理する必要があります。以下の例では、サーバーの停止方法と、Server インターフェースで `waitFor` メソッドを使用して Server インスタンスが確実に完全にシャットダウンする方法を示しています。コンテナの例と同様に、`stopServer` メソッドは、ServerMBean インターフェースを通して使用可能になります。このインターフェースでは、該当の Managed Bean (MBean) によりサーバーを停止できます。

```
ServerFactory.stopServer(); // Uses the factory to kill the Server singleton
// or
server.stopServer(); // Uses the Server instance directly
server.waitFor(); // Returns when the server has properly completed its shutdown procedures
```

全コードの例:

```
import java.net.MalformedURLException;
import java.net.URL;

import com.ibm.websphere.objectgrid.ObjectGridException;
import com.ibm.websphere.objectgrid.deployment.DeploymentPolicy;
import com.ibm.websphere.objectgrid.deployment.DeploymentPolicyFactory;
import com.ibm.websphere.objectgrid.server.Container;
import com.ibm.websphere.objectgrid.server.Server;
import com.ibm.websphere.objectgrid.server.ServerFactory;
import com.ibm.websphere.objectgrid.server.ServerProperties;

public class ServerFactoryTest {

    public static void main(String[] args) {

        try {

            ServerProperties props = ServerFactory.getServerProperties();
            props.setCatalogServiceBootstrap("catalogservice-hostname:catalogservice-port");
            props.setServerName("ServerOne"); // name server
            props.setTraceSpecification("com.ibm.ws.objectgrid=all=enabled"); // TraceSpec

            /*
             * In most cases, the server will serve as a container server only
             * and will connect to an external catalog service. This is a more
             * highly available way of doing things. The commented code excerpt
             * below will enable this Server to be a catalog service.
            */
        }
    }
}
```



```

*
*
* CatalogServerProperties catalogProps =
* ServerFactory.getCatalogProperties();
* catalogProps.setCatalogServer(true); // enable catalog service
* catalogProps.setQuorum(true); // enable quorum
*/

Server server = ServerFactory.getInstance();

DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy
(new URL("url to deployment xml"), new URL("url to objectgrid xml file"));
Container container = server.createContainer(policy);

/*
* Shard will now be placed on this container if the deployment requirements are met.
* This encompasses embedded server and container creation.
*
* The lines below will simply demonstrate calling the cleanup methods
*/

container.teardown();
server.stopServer();
int success = server.waitFor();

} catch (ObjectGridException e) {
// Container failed to initialize
} catch (MalformedURLException e2) {
// invalid url to xml file(s)
}
}
}
}

```

## 組み込みサーバー API

WebSphere eXtreme Scale には、既存の Java アプリケーション内に eXtreme Scale サーバーおよびクライアントを組み込む、アプリケーション・プログラミング・インターフェース (API) およびシステム・プログラミング・インターフェースが含まれています。以下のトピックで、使用可能な組み込みサーバー API について説明します。

### eXtreme Scale サーバーのインスタンス化

eXtreme Scale サーバー・インスタンスの構成には、いくつかのプロパティを使用できますが、これは、`ServerFactory.getServerProperties` メソッドから取得できます。`ServerProperties` オブジェクトは singleton のため、`getServerProperties` メソッドの各呼び出しでは同じインスタンスが取得されます。

次のコードを使用して、新規サーバーを作成することができます。

```
Server server = ServerFactory.getInstance();
```

`getInstance` の最初の呼び出しの前に設定されたすべてのプロパティは、サーバーの初期化に使用されます。

### サーバー・プロパティの設定

サーバー・プロパティは、`ServerFactory.getInstance` が最初に呼び出されるまで設定できます。`getInstance` メソッドの最初の呼び出しで、eXtreme Scale サーバーがインスタンス化され、構成されたすべてのプロパティが読み取られます。作成後にプロパティを設定しても効果はありません。以下の例は、`Server` インスタンスをインスタンス化する前のプロパティの設定方法を示しています。

```

// Get the server properties associated with this process.
ServerProperties serverProperties = ServerFactory.getServerProperties();

// Set the server name for this process.
serverProperties.setServerName("EmbeddedServerA");

// Set the name of the zone this process is contained in.
serverProperties.setZoneName("EmbeddedZone1");

// Set the end point information required to bootstrap to the catalog service.
serverProperties.setCatalogServiceBootstrap("localhost:2809");

// Set the ORB listener host name to use to bind to.
serverProperties.setListenerHost("host.local.domain");

// Set the ORB listener port to use to bind to.
serverProperties.setListenerPort(9010);

// Turn off all MBeans for this process.
serverProperties.setMBeansEnabled(false);

Server server = ServerFactory.getInstance();

```

## カタログ・サービスの組み込み

`CatalogServerProperties.setCatalogServer` メソッドによりフラグが立てれた JVM 設定は、eXtreme Scale のカタログ・サービスをホストできます。このメソッドは、eXtreme Scale サーバー・ランタイムに対して、サーバーの始動時にカタログ・サービスをインスタンス化することを指示します。以下のコードは、eXtreme Scale カタログ・サーバーをインスタンス化する方法を示しています。

```

CatalogServerProperties catalogServerProperties =
    ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();

```

## eXtreme Scale コンテナの組み込み

JVM が複数の eXtreme Scale コンテナをホストするようにするには、`Server.createContainer` メソッドを発行します。以下のコードは、eXtreme Scale コンテナをインスタンス化する方法を示しています。

```

Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);

```

## 自己完結型のサーバー・プロセス

すべてのサービスは、まとめて開始することができ、これは開発に便利で、実行中にも実用的です。サービスをまとめて開始することにより、1 つのプロセスで、カタログ・サービスの開始、コンテナ・セットの開始、クライアント接続ロジックの実行をすべて行うことができます。このような方法でサービスを開始すると、分散環境にデプロイする前にプログラム上の問題を整理することができます。以下のコードは、自己完結型の eXtreme Scale サーバーをインスタンス化する方法を示しています。

```

CatalogServerProperties catalogServerProperties =
    ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);

```

## WebSphere Application Server における eXtreme Scale の組み込み

eXtreme Scale の構成は、eXtreme Scale を WebSphere Application Server 環境にインストールすると、自動的にセットアップされます。サーバーにアクセスしてコンテナを作成する前にプロパティを設定する必要はありません。以下のコードは、eXtreme Scale サーバーを WebSphere Application Server でインスタンス化する方法を示しています。

```

Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);

```

組み込みカタログ・サービスおよびコンテナをプログラマチックに開始する方法に関する段階的な例は、468 ページの『組み込みサーバー API を使用したサーバーの開始と停止』を参照してください。

---

## xscmd ユーティリティによる管理

**xscmd** を使用して、マルチマスター・レプリカ生成リンクの確立、クォーラムの上書き、ティアダウン・コマンドを使用したサーバー・グループの停止などの管理タスクを環境内で実行することができます。

### 始める前に

- カタログ・サーバーおよびコンテナ・サーバーが始動済みでなければなりません。カタログ・サーバーがカタログ・サービス・ドメインにある場合、少なくとも 2 つのカタログ・サーバーが始動済みでなければなりません。
- 製品と一緒にインストールされたランタイム環境を使用するように `JAVA_HOME` 環境変数が設定されていることを確認してください。製品の試用版を使用している場合は、`JAVA_HOME` 環境変数を設定する必要があります。

### このタスクについて

**xscmd** ユーティリティは、完全にサポートされたモニターおよび管理のツールとして、**xsadmin** サンプル・ユーティリティに取って代わります。**xsadmin** ツールを使用して同様の操作を完了できる可能性もありますが、このツールはサポートされていません。**xsadmin** サンプルは現在のデプロイメント・データの解析とディスクカバリーの方法を提供するもので、カスタム・ユーティリティの作成基盤として使用することができます。以前、モニターおよび管理のために **xsadmin** ツールを使用していた場合は、スクリプトを **xscmd** ユーティリティを使用するように更新することを検討してください。**xsadmin** コマンドを新しい **xscmd** コマンドにマッピングすることについて詳しくは、245 ページの『**xsadmin** ツールから **xscmd** ツール

へのマイグレーション』を参照してください。

## 手順

1. オプション: クライアント認証が使用可能な場合: コマンド行ウィンドウを開きます。コマンド行で、適切な環境変数を設定します。

a. `CLIENT_AUTH_LIB` 環境変数を設定します。

- `Windows` `set CLIENT_AUTH_LIB=<path_to_security_JAR_or_classes>`
- `UNIX` `set CLIENT_AUTH_LIB=<path_to_security_JAR_or_classes>`  
`export CLIENT_AUTH_LIB`

2. `wxs_home/bin` ディレクトリーに移動します。

```
cd wxs_home/bin
```

3. さまざまな `xscmd` オプションのヘルプを表示します。

• 一般ヘルプを表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -h
```

```
- Windows xscmd.bat -h
```

• すべてのコマンドのリストを表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -lc
```

```
- Windows xscmd.bat -lc
```

• 特定のコマンドのヘルプを表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -h command_name
```

```
- Windows xscmd.bat -h command_name
```

• コマンド・グループのリストを表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -lcg
```

```
- Windows xscmd.bat -lcg
```

• コマンド・グループ内のコマンドのリストを表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -lc command_group_name
```

```
- Windows xscmd.bat -lc command_group_name
```

4. 特定のカタログ・サーバーに接続するコマンドを実行します。デフォルトでは、`xscmd` は、ホスト名およびポート `localhost:2809` を使用して、ローカル・ホスト上のカタログ・サーバーに接続します。ホスト名およびポートのリストをコマンドに指定して、他のホスト上のカタログ・サーバーに接続することもできます。リストから、`xscmd` ユーティリティーの 1 つが、ランダム・ホストに接続します。指定するホストのリストは、同じカタログ・サービス・ドメイン内になければなりません。

• 接続するスタンドアロンのカタログ・サーバーのリストを指定します。

```
- UNIX ./xscmd.sh -c <command_name> -cep hostname:port  
(,hostname:port)
```

```
- Windows xscmd.bat -c <command_name> -cep hostname:port  
(,hostname:port)
```

上記のコマンドで、*command\_name* は実行しようとしているコマンドの名前です。*hostname:port* 値は、カタログ・サーバーのホスト名とリスナー・ポートです。スタンドアロンのカタログ・サーバーのリスナー・ポート値は、**start0gServer** コマンドを実行するときに指定されます。

- 接続する WebSphere Application Server カタログ・サーバーのリストを指定します。デフォルトのローカル・ホスト値では、WebSphere Application Server で稼働しているカタログ・サーバーに接続することはできません。

```
- UNIX ./xscmd.sh -c <command_name> -cep was_hostname:port  
(,hostname:port)
```

```
- Windows xscmd.bat -c <command_name> -cep was_hostname:port  
(,hostname:port)
```

上記のコマンドで、*command\_name* は実行しようとしているコマンドの名前です。*was\_hostname* 値は、WebSphere Application Server セル内のカタログ・サーバーのホスト名です。*port* 値は、リスナー・ポートです。WebSphere Application Server では、リスナー・ポート値は **BOOTSTRAP\_ADDRESS** ポート構成によって継承されます。カタログ・サーバーがデプロイメント・マネージャーで稼働している場合、デフォルト値は 9809 です。カタログ・サーバーがアプリケーション・サーバーで稼働している場合、アプリケーション・サーバーの **BOOTSTRAP\_ADDRESS** ポート構成を調べて、ポート番号を確認してください。

---

## 配置の制御

いくつかの異なるオプションを使用して、構成内のさまざまなコンテナ・サーバー上にいつ断片を配置するかを制御できます。始動時は、断片の配置を遅らせるよう選択できます。すべてのコンテナ・サーバーを実行中の場合は、サーバーを保守する間、配置を中断、再開、または変更する必要がある可能性があります。

### 手順

#### 始動時の配置の制御

環境の始動時、断片の配置をいつ開始するかを制御できます。デフォルトである種の制御は実施されます。配置を制御するアクションを何も取らなければ、断片の配置は即時に開始されます。断片がすぐに配置されると、後続のコンテナ・サーバーが開始するにつれ断片の配置が均等でなくなる可能性があり、配分のバランスを取るために追加の配置操作が実行されます。

- コンテナ・サーバーの開始時、断片のบาลランシングを一時的に中断して、断片の即時配置を回避する。

コンテナ・サーバーを開始する前に、**xscmd -c suspendBalancing** コマンドを使用して、特定のデータ・グリッドおよびマップ・セットの断片のบาลランシングを停止します。コンテナ・サーバーが開始されたら、**xscmd -c resumeBalancing** コマンドを使用して、コンテナ・サーバーでの断片の配置を開始できます。

- **placementDeferralInterval** プロパティを構成する。

**placementDeferralInterval** プロパティは、コンテナ・サーバーでの断片配置のサイクル数を最小にします。断片配置は定義された時間間隔で起動されません。

カタログ・サーバーのサーバー・プロパティ・ファイル内で

**placementDeferralInterval** プロパティを設定します。組み込みサーバー API を使用している場合は、CatalogServerProperties インターフェースの `setPlacementDeferralInterval` メソッドを使用してください。このプロパティは、コンテナ・サーバー上に断片を配置するまでの時間 (ミリ秒数) を設定します。このプロパティのデフォルト値は 15 秒です。デフォルト値では、コンテナ・サーバーが開始しても、プロパティに指定された時間が経過するまで配置は開始されません。複数のコンテナ・サーバーが連続して開始する場合、所定の間隔内に新しいコンテナ・サーバーが開始すると、遅延インターバル・タイマーはリセットされます。例えば、最初のコンテナ・サーバーの 10 秒後に 2 番目のコンテナ・サーバーが開始すれば、2 番目のコンテナ・サーバーの開始後 15 秒経過するまで配置は開始されません。しかし、2 番目のコンテナ・サーバーの 20 秒後に 3 番目のコンテナ・サーバーが開始した場合、配置は既に最初の 2 つのコンテナ・サーバーで開始されています。

コンテナ・サーバーが使用不可になると、カタログ・サーバーがそのイベントを認識し次第、配置が起動され、できるだけ速やかにリカバリーできるようにします。

次のヒントを基に、配置の遅延時間が正しい値に設定されているかどうか判断できます。

- 複数のコンテナ・サーバーを同時に開始し、各コンテナ・サーバーの SystemOut.log ファイルで CWOBJ1001 メッセージを確認します。各コンテナ・サーバーのログ・ファイルにあるこれらのメッセージのタイム・スタンプは、コンテナ・サーバーの実際の開始時刻を表します。場合によっては、**placementDeferralInterval** プロパティを調整して、この間隔により多くのコンテナ・サーバーの開始を含めることを検討してください。例えば、最初のコンテナ・サーバーが最後のコンテナ・サーバーの 90 秒前に開始している場合、プロパティを 90 秒に設定します。
- CWOBJ1001 メッセージの後、どのくらいの時間で CWOBJ1511 メッセージが発生しているかに注目します。この時間の長さから、正常に遅延が発生したかどうか分かることがあります。
- 開発環境を使用している場合、アプリケーションのテスト時に間隔の長さを調整します。

- **numInitialContainers** 属性を構成する。

以前 **numInitialContainers** 属性を使用していた場合、引き続きこの属性を使用できます。しかし、配置を制御する目的では、**numInitialContainers** 属性よりも **xscmd -c suspendBalancing** および **xscmd -c resumeBalancing** コマンド、その次には **placementDeferralInterval** を使用するほうが推奨されます。

**numInitialContainers** 属性は、この mapSet エレメント内の断片の初期配置を行うために必要とするコンテナ・サーバーの数を指定します。

**numInitialContainers** 属性は、デプロイメント・ポリシー記述子 XML ファイ



ル内にあります。**numInitialContainers** と **placementDeferralInterval** を両方設定した場合、**placementDeferralInterval** プロパティの値に関係なく、**numInitialContainers** 値が満たされるまで配置は発生しないので注意してください。

初期始動後の配置の制御

- 配置を強制的に行う。

**xscmd -c triggerPlacement -g my\_OG -ms my\_Map\_Set** コマンドを使用して、強制しなければ配置は発生しない可能性がある一時点で、強制的に配置を行うことができます。ここで、**my\_OG** と **my\_Map\_Set** は、実際のデータ・グリッドとマップ・セットの値に設定してください。例えば、**placementDeferralInterval** プロパティで指定された時間がまだ経過していないときや、balancingが中断状態のときにこのコマンドを実行できます。

- プライマリー断片を再割り当てする。

**xscmd -c swapShardWithPrimary** コマンドを使用して、新しいプライマリー断片になるレプリカ断片を割り当てます。前のプライマリー断片はレプリカになります。

- プライマリー断片とレプリカ断片を再balancingする。

**xscmd -c balanceShardTypes** コマンドを使用して、構成内で実行中のコンテナ・サーバー間でプライマリー断片とレプリカ断片の比率が衡平になるよう調整します。各コンテナ・サーバーでの比率は、断片 1 つの差の範囲内に保たれます。

- 配置を中断または再開する。

**xscmd -c suspendBalancing** コマンドまたは **xscmd -c resumeBalancing** コマンドを使用して、特定のデータ・グリッドおよびマップ・セットの断片のbalancingを停止または開始します。balancingが中断状態になっても、次の配置アクションはそのまま実行されます。

- コンテナ・サーバーに障害が起こった際の断片のプロモーション。
- **xscmd -c swapShardWithPrimary** コマンドによる断片の役割の交換。
- **xscmd -c triggerPlacement -g myOG -ms myMapSet** コマンドによる断片配置で起動されたbalancing。

## 次のタスク

環境内の配置を **xscmd -c placementServiceStatus** コマンドでモニターできます。

---

## ObjectGrid の可用性の管理

ObjectGrid インスタンスの可用性状態は、特定の時点でどの要求を処理できるかを決定します。StateManager インターフェースを使用して、ObjectGrid インスタンスの状態の設定および取得を行います。

### このタスクについて

任意の ObjectGrid インスタンスには、4 つの可用性状態があります。

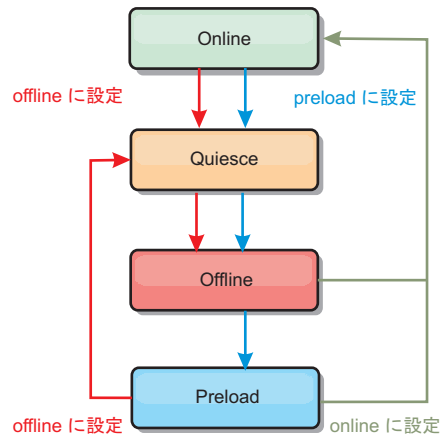


図 50. ObjectGrid インスタンスの可用性状態

### ONLINE

ONLINE 状態は、ObjectGrid のデフォルトの可用性状態です。ONLINE 状態の ObjectGrid は、標準 eXtreme Scale クライアントからのどの要求でも処理できます。ただし、プリロード・クライアントからの要求は、ObjectGrid が ONLINE である間は拒否されます。

### QUIESCE

QUIESCE 状態は、遷移的状态です。QUIESCE 状態にある ObjectGrid は、すぐに OFFLINE 状態に移行します。QUIESCE 状態にある ObjectGrid は、未解決のトランザクションを処理できます。ただし新規トランザクションは拒否されます。ObjectGrid が QUIESCE 状態でいられるのは、30 秒までです。この時間を過ぎると、可用性状態は OFFLINE に移行します。

### OFFLINE

OFFLINE 状態では、ObjectGrid に送信されたすべてのトランザクションは拒否されます。

### PRELOAD

PRELOAD 状態は、プリロード・クライアントからデータを ObjectGrid にロードする場合に使用できます。ObjectGrid が PRELOAD 状態である間は、プリロード・クライアントしか、ObjectGrid に対してトランザクションをコミットできません。他のすべてのトランザクションは拒否されます。

ObjectGrid が要求をサポートするのに適切な可用性状態にない場合、その要求は拒否されます。要求が拒否されると、必ず AvailabilityException 例外が発生します。

### 手順

1. ObjectGrid 構成 XML ファイルを使用して、ObjectGrid の初期状態を設定します。

**initialState** 属性は、ObjectGrid でその始動時の状態を示すのに使用できません。通常、初期化が完了した ObjectGrid は、ルーティングに使用可能になります。トラフィックが ObjectGrid にルーティングされないように、後で状態を変更できます。ObjectGrid を初期化する必要があるが、すぐには使用可能でない場合、**initialState** 属性を使用できます。

`initialState` 属性は、ObjectGrid の構成 XML ファイルで設定されます。デフォルト状態は ONLINE です。有効な値には、次のものが含まれます。

- ONLINE (デフォルト)
- PRELOAD
- OFFLINE

`initialState` 属性について詳しくは、ObjectGrid 記述子 XML ファイルを参照してください。

ObjectGrid で `initialState` 属性が設定されている場合、その状態を明示的にオンラインに戻す必要があります。さもないと、ObjectGrid は使用不可のままになります。ObjectGrid が ONLINE 状態でなければ、AvailabilityException 例外になります。

詳しくは、AvailabilityState API 資料を参照してください。

### プリロードのための `initialState` 属性の使用

ObjectGrid にデータがプリロードされる場合、ObjectGrid が使用可能になる時点と、クライアント・トラフィックをブロックするためにプリロード状態に切り替わる時点との間に、しばらく時間があくことがあります。この時間を回避するため、ObjectGrid の初期状態を PRELOAD に設定できます。この場合にも、ObjectGrid は必要な初期化をすべて完了しますが、状態が変更され、プリロードを実行できるようになるまで、トラフィックをブロックします。

PRELOAD 状態も OFFLINE 状態もトラフィックをブロックしますが、プリロードを開始する場合は PRELOAD 状態を使用する必要があります。

### フェイルオーバーおよびバランシングの振る舞い

レプリカ・データ・グリッドがプライマリー・データ・グリッドにプロモートされると、レプリカは、`initialState` 設定を使用しません。プライマリー・データ・グリッドが再バランシングのために移動された場合、移動が完了する前に、データがプライマリーの新しい場所にコピーされるため、`initialState` 設定は使用されません。レプリカ生成が構成されていない場合、フェイルオーバーが発生すると、プライマリーは `initialState` 設定になり、新規プライマリーを配置する必要があります。

2. StateManager インターフェースを使用して可用性状態を変更します。

ObjectGrid の可用性状態を設定するには、StateManager を使用します。サーバーで実行中の ObjectGrid の可用性状態を設定するには、対応する ObjectGrid クライアントを StateManager インターフェースに渡します。以下のコードは、ObjectGrid の可用性状態を変更する方法を示したものです。

```
ClientClusterContext client = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

StateManager で `setObjectGridState` メソッドが呼び出されると、ObjectGrid の各断片が要求状態に遷移します。メソッドが戻ると、ObjectGrid 内のすべての断片は、適切な状態になります。

サーバー・サイド ObjectGrid の可用性状態を変更するには、ObjectGridEventListener プラグインを使用します。サーバー・サイド ObjectGrid の可用性状態の変更は、その ObjectGrid の区画が 1 つだけの場合にのみ行ってください。ObjectGrid が複数の区画を持っている場合、各プライマリーで shardActivated メソッドが呼び出され、結果として ObjectGrid の状態変更のために必要以上の呼び出しが行われることになります。

```
public class OGListener implements ObjectGridEventListener,
    ObjectGridEventGroup.ShardEvents {
    public void shardActivated(ObjectGrid grid) {
        StateManager stateManager = StateManagerFactory.getStateManager();
        stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

QUIESCE は遷移的状态であるため、ObjectGrid を QUIESCE 状態にするのに StateManager インターフェースを使用することはできません。ObjectGrid は、この状態を経てから OFFLINE 状態に移行します。

### 3. 可用性状態を取得します。

特定の ObjectGrid の可用性状態を取得するには、StateManager の getObjectGridState メソッドを使用します。

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

getObjectGridState メソッドは、ObjectGrid 内のランダム・プライマリーを選択して、その AvailabilityState を返します。ObjectGrid のすべての断片は同じ可用性状態になっているか、同じ可用性状態に遷移中である必要があるため、このメソッドにより、ObjectGrid の現在の可用性状態に関する妥当な結果がもたらされません。

---

## データ・センター障害の管理

データ・センターで障害シナリオが発生した場合、コンテナ・サーバー・イベントが無視されないようにクォーラムのオーバーライドを検討してください。xscmd ユーティリティを使用して、クォーラムについて照会したり（クォーラムの状況など）、クォーラムのタスク（クォーラムのオーバーライドなど）を実行したりできます。

### 始める前に

- すべてのカタログ・サーバーでクォーラム・メカニズムが同じ設定になるように構成します。詳しくは、307 ページの『クォーラム・メカニズムの構成』を参照してください。
- クォーラムとは、データ・グリッドの配置操作を実行するために必要なカタログ・サーバーの最小数のことです。また、より少ない数を構成している場合を除いて、カタログ・サーバーのフル・セットでもあります。WebSphere eXtreme Scale は、以下の理由によってクォーラムの損失を予想します。
  - カatalog・サービス JVM メンバーの障害
  - ネットワークのブラウン・アウト
  - データ・センター損失

次のメッセージはクォーラムが失われたことを示しています。このメッセージはご使用のカタログ・サービス・ログに入っています。

CW0BJ1254W: カタログ・サービスがクォーラムを待機しています。

## このタスクについて

データ・センターで障害シナリオが発生した場合のみ、クォーラムをオーバーライドしてください。クォーラムをオーバーライドすると、残存しているカタログ・サーバー・インスタンスがすべて使用されます。ある残存物に対してクォーラムをオーバーライドする指示が出されると、それがすべての残存物に通知されます。

## 手順

- **xscmd** ユーティリティを使用してクォーラムの状況を照会します。

```
xscmd -c showQuorumStatus -cep cathost:2809
```

このオプションを使用して、カタログ・サービス・インスタンスのクォーラムの状況を表示します。次のいずれかの結果が表示されます。

- クォーラムが使用不可である: カタログ・サーバーがクォーラム使用不可モードで実行されています。クォーラム使用不可モードは、開発モードまたは単一データ・センター・モードです。複数データ・センター構成の場合、クォーラム使用不可モードは使用しないでください。
- クォーラムが使用可能で、かつカタログ・サーバーがクォーラムを持っている: クォーラムが使用可能で、しかもシステムが正常に機能しています。
- クォーラムは使用可能だが、カタログ・サーバーがクォーラムを待機している: クォーラムが使用可能で、かつクォーラムが失われています。
- クォーラムが使用可能で、かつクォーラムがオーバーライドされている: クォーラムが使用可能で、しかもクォーラムがオーバーライドされました。
- クォーラム状況が禁止である: ブラウン・アウトが発生したとき、カタログ・サーバーが 2 つの区画 (A および B) に分割され、カタログ・サーバー A のクォーラムがオーバーライドされました。ネットワーク区画は分解され、B 区画にあるサーバーが禁止されているため、JVM の再始動が必要です。この状況は、ブラウン・アウト中に B にあるカタログ JVM が再始動されてから、ブラウン・アウトが解消された場合にも起こります。

- **xscmd** ユーティリティを使用してクォーラムをオーバーライドします。

```
xscmd -c overrideQuorum -cep cathost:2809
```

このコマンドを実行すると、残存しているカタログ・サーバーによるクォーラムの再確立が強制されます。

- **xscmd** ユーティリティを使用してクォーラムを診断します。

- コア・グループのリストを表示する。

-c **listCoreGroups** オプションを使用すると、カタログ・サーバーのすべてのコア・グループのリストを表示できます。

```
xscmd -c listCoreGroups -cep cathost:2809
```

- サーバーを削除する。

**-c teardown** オプションを使用すると、データ・グリッドから手動でサーバーを除去できます。通常は、グリッドからサーバーを除去する必要はありません。障害サーバーとして検出されたサーバーは自動的に除去されます。このコマンドは IBM サポートのガイダンスのもとで使用するために提供されています。このコマンドの使用法についての詳細は、466 ページの『**xscmd** ユーティリティーによるサーバーの正常停止』を参照してください。

```
xscmd -c teardown server1,server2,server3 -cep cathost:2809 -g Grid
```

– 経路テーブルを表示する。

**-c routetable** オプションを使用すると、データ・グリッドへの新しいクライアント接続をシミュレートすることによって、現在の経路テーブルを表示できます。また、すべてのコンテナ・サーバーが経路テーブル内でのそれぞれのロール（どの区画のどのタイプの断片であるかなど）を認識していることを確認することによって、経路テーブルの妥当性検査も行います。

```
xscmd -c routetable -cep cathost:2809 -g myGrid
```

– マップ・サイズを確認する。

**-c showMapSizes** オプションを使用すると、キーの分布がキー内の断片間で均等か確認できます。一部のコンテナ・サーバーのキーが他のコンテナ・サーバーより多い場合は、キー・オブジェクトのハッシュ関数の分布に問題がある可能性があります。

```
xscmd -c showMapSizes -cep cathost:2809 -g myGrid -ms myMapSet
```

– トレース・ストリングを設定する。

**-c setTraceSpec** オプションを使用すると、**xscmd** コマンドに指定されたフィルターと一致するすべての JVM にトレース設定を指定できます。この設定は、別のコマンドが使用されるまで、または変更された JVM が障害を起こすか停止するまで、トレース設定のみを変更します。

```
xscmd -c setTraceSpec -spec ObjectGrid*=event=enabled -cep cathost:1099 -g myGrid -hf host1
```

このストリングは、指定されたホスト名（この場合は host1）を持つサーバー上にあるすべての JVM のトレースを使用可能にします。

– 未割り当ての断片を表示する。

**-c showPlacement -sf U** オプションを使用すると、データ・グリッド上に配置できない断片のリストを表示できます。配置サービスに配置を妨げる制約があると、断片は配置できません。例えば、実動モードのとき、単一物理サーバー上の JVM を開始した場合は、プライマリー断片のみを配置できます。2 番目の物理サーバーで JVM が開始されるまで、レプリカは未割り当てとなります。配置サービスは、プライマリー断片をホスティングしている JVM とは異なる IP アドレスを持つ JVM にのみレプリカを配置します。ゾーンに JVM が存在しない場合も、断片が未割り当てとなることがあります。

```
xscmd -c showPlacement -sf U -cep cathost:2809 -g myGrid
```



## データの照会と無効化

モニター・コンソールおよび `xscmd` ユーティリティで QUERY インターフェースを使用して、マップから小規模なキー・セットを取得し、一連のデータを無効化することができます。

### 始める前に

- Web コンソールを使用してデータの照会および無効化を行う場合は、まずモニター・コンソールを構成します。詳しくは、507 ページの『Web コンソールによるモニター』を参照してください。
- `xscmd` を使用してデータの照会および無効化を行う場合は、`xscmd` ユーティリティをセットアップします。詳しくは、473 ページの『`xscmd` ユーティリティによる管理』を参照してください。

### このタスクについて

データ・グリッドの内容を照会するには、コンソールまたは `xscmd` ユーティリティを使用できます。データ・キーで正規表現を実行することで、データを照会できます。その後、同じ照会を使用してデータを無効にすることができます。

正規表現は、他のストリングとのマッチング使用される構造化ストリングです。一般的な正規表現には以下のようなものがあります。



正規表現	説明
.	任意の 1 文字を検索します。
.*	任意の 1 文字を 0 回以上検索します。
A.*	「A」で始まるすべてのキーを検索します。
.*A	「A」で終わるすべてのキーを検索します。
A?	「A」を 0 回または 1 回検索します。
A*	「A」を 0 回または複数回検索します。
A+	「A」を 1 回または複数回検索します。
A?B*C+	後に 0 個または 1 個の A が続き、その後に 0 個または複数個の B が続き、その後に 1 個または複数個の C が続く任意の 1 文字を検索します
[ABC].*	「A」、「B」、「C」のいずれかの文字で始まるキーを検索します。
.*ABC.*	ストリングのどこかに「ABC」というストリングを含むキーを検索します。
(ABC DEF).*	ストリング「ABC」または「DEF」で始まるキーを検索します。

詳しい正規表現の構文については `java.util.regex.Pattern` API の資料を参照してください。

### 手順

- コンソールでデータの照会または無効化を行います。

1. コンソールで、照会ページに進みます。 Web コンソールで、「管理」 > 「データ・グリッド内容の照会」とクリックします。フィルター処理をするマップを選択します。
2. マップ内のデータを検索またはフィルター処理します。データの検索またはフィルター処理を行うには、以下のいずれかのオプションを使用します。

- フィールドに正規表現を入力し、「検索」ボタン (  ) をクリックします。その正規表現に一致するキーのリストが表示されます。このデータのリストは、一致するすべてのデータのサブセットである場合があります。
  - パーティション・セットで結果のフィルター処理をするには、「フィルター」ボタン (  ) をクリックします。次に正規表現を入力して、結果のフィルター処理をするパーティションの範囲を選択します。
3. データを無効化します。データを無効化すると、データは、データ・グリッドから永続的に削除されます。

#### 選択したキー

無効化するキーをテーブルから選択することができます。エントリーを個々にクリックすることも、あるいは、「すべて選択」チェック・ボックスをクリックする (これによりテーブル内の最大 500 個のエントリーが選択される) こともできます。削除したいエントリーを選択したら、「無効化」 > 「選択したキー」とクリックします。

#### 照会に一致するすべてのキー

指定した正規表現に一致するすべてのデータを無効化することもできます。このオプションを使用すると、コンソールに表示される最大 500 のエントリーだけでなく、正規表現に一致する、データ・グリッド内のすべてのデータが削除されます。選択した正規表現を使用してエントリーを無効化するには、「無効化」 > 「照会に一致するすべてのキー」とクリックします。

- **xscmd** ユーティリティでデータの照会または無効化を行います。

#### データの照会:

```
xscmd.sh -c findbykey -g <data_grid> -m <map>
-fs <find_string> [-fp <partitionid>]
```

検索ストリング値のデータ・グリッド、マップ、および正規表現を含める必要があります。区画 ID でフィルタリングすることもできます。結果は照会全体のサブセットを返します。

#### データの無効化:

照会によって選択されたデータを無効化するには、コマンドに **-inv** 引数を含めます。

```
xscmd -c findbykey -g <data_grid> -m <map>
-fs <find_string> [-fp <partitionid>] -inv
```

検索ストリング値のデータ・グリッド、マップ、および正規表現を含める必要があります。区画 ID でフィルタリングすることもできます。無効化を実行する際、照会によって返される小規模なセットだけではなく、一致するすべての値が無効化されます。

**重要:** **UNIX** **Linux** 使用する正規表現が文字 `.*` で始まる場合、コマンドを実行する際にその文字は正常に処理されない可能性があります。この問題を解決するには、ご使用の正規表現を次のいずれかの方法でフォーマットしてください。

- 正規表現をアポストロフィ文字で囲みます: `-fs '.*'`
- 円記号 (¥) を使用してアスタリスク文字をエスケープします: `-fs .¥*`

**例:**

次の例は、Grid データ・グリッドと Map1 マップにあるすべてのエントリーを検索します。

```
xscmd -c findbykey -g Grid -m Map1 -fs ".*"
```

このコマンドは、次の結果を返します。

```
3 matching keys were found.
Partition Key
-----
2          keyghi
4          keydef
6          keyabc
```

## xscmd ユーティリティを使用した eXtreme Scale 環境情報の取得

**xscmd** ユーティリティを `-c showinfo` コマンドと一緒に使用して、WebSphere eXtreme Scale 環境で実行されているサーバーに関する重要な詳細を表示することができます。これらのサーバーには、WebSphere eXtreme Scale サーバー、Java 仮想マシン、および (該当する場合) WebSphere Application Server と一緒に実行されるサーバーが含まれます。このコマンドを実行して、名前およびバージョン情報、ホスト名と IP アドレス、これらのサーバーのインストール・ディレクトリを取得します。`-c showinfo` コマンドを使用すると、これらの詳細を取得でき、ログ・ファイルやディレクトリを調べたり、サード・パーティー・アプリケーションを使用したりする必要がありません。

### 手順

- カタログ・サーバーのうち少なくとも 1 つが稼働中であることを確認します。eXtreme Scale ドメイン全体の環境詳細を取得する場合は、すべてのサーバーが稼働中であることを確認してください。

eXtreme Scale ドメイン全体の環境情報を取得するには、次のコマンドを実行します。

**Windows**  
`xscmd.bat -c showinfo`

**UNIX**  
`./xscmd.sh -c showinfo`

このコマンドは、ご使用の環境内で稼働しているサーバーに関するすべての情報を返します。

- 特定のサーバーに関する情報を取得するには、**-s** パラメーターを使用し、そのサーバーの名前を指定します。

Windows

```
xscmd.bat -c showinfo -s <server_name>
```

UNIX

```
./xscmd.sh -c showinfo -s <server_name>
```

- サーバーのリストを表示するには、**-sl** パラメーターを使用します。

Windows

```
xscmd.bat -c showinfo -sl <server_name>[,<server_name>]
```

UNIX

```
./xscmd.sh -c showinfo -sl <server_name>[,<server_name>]
```

- 特定のホストで稼働しているサーバーの特定のセットについて環境情報を取得するには、**-hf** パラメーターを使用し、そのホストの名前を指定します。

Windows

```
xscmd.bat -c showinfo -hf <host_name>
```

UNIX

```
./xscmd.sh -c showinfo -hf <host_name>
```

---

## Eclipse Equinox OSGi フレームワークを使用した eXtreme Scale サーバーの始動

WebSphere eXtreme Scale コンテナ・サーバーは、いくつかの方法を使用して、Eclipse Equinox OSGi フレームワークの中で始動することができます。

### 始める前に

eXtreme Scale コンテナを開始する前に、次のタスクを完了していなければなりません。

1. WebSphere eXtreme Scale サーバー・バンドルが Eclipse Equinox にインストールされていなければなりません。
2. アプリケーションは OSGi バンドルとしてパッケージされていなければなりません。
3. WebSphere eXtreme Scale プラグインがある場合は、OSGi バンドルとしてパッケージされていなければなりません。これらのプラグインは、アプリケーションと同じバンドルにバンドルすることも、別々のバンドルとしてバンドルすることもできます。

## このタスクについて

このタスクでは、Eclipse Equinox OSGi フレームワークの中で eXtreme Scale コンテナ・サーバーを始動する方法を説明します。Eclipse Equinox 実装を使用してコンテナ・サーバーを始動するには、次のいずれかの方法を使用することができます。

- OSGi Blueprint サービス

OSGi バンドルの中に、すべての構成およびメタデータを含めることができます。次の図を参考にして、この方法の Eclipse Equinox プロセスを理解してください。

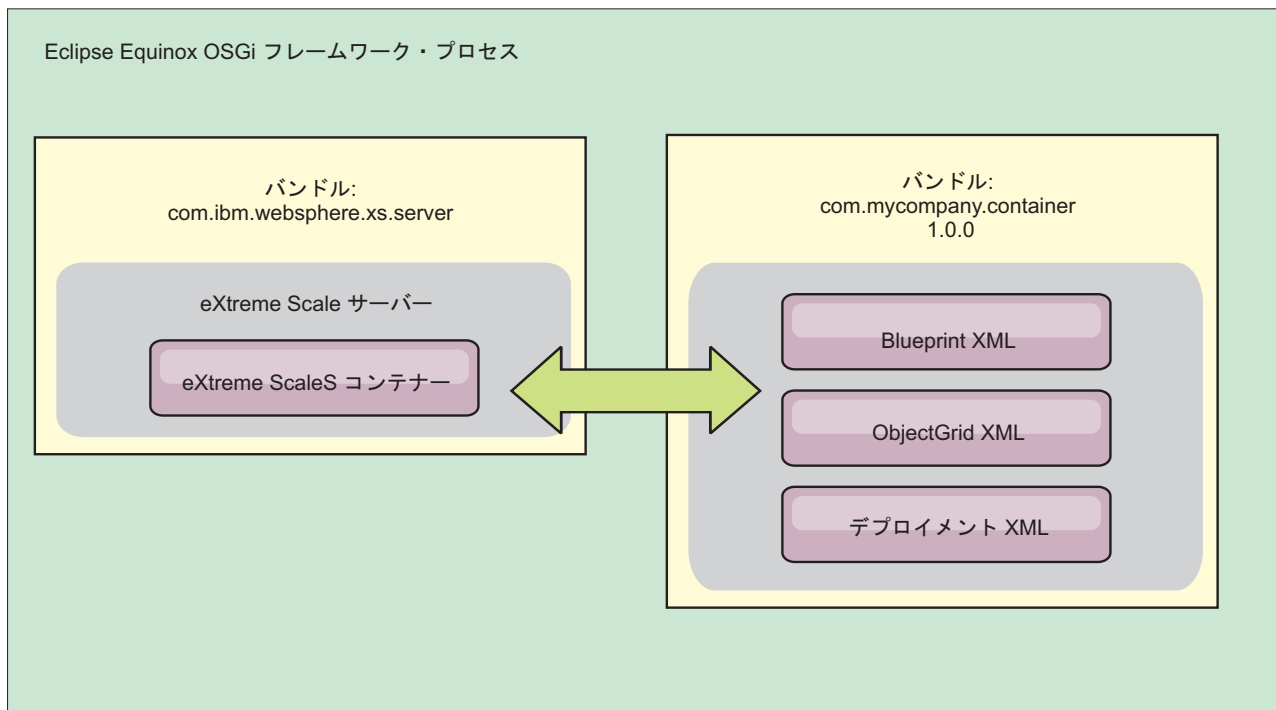


図 51. OSGi バンドルにすべての構成およびメタデータを含めるための Eclipse Equinox プロセス

- OSGi Configuration Admin サービス

OSGi バンドルの外部で構成およびメタデータを指定できます。次の図を参考にして、この方法の Eclipse Equinox プロセスを理解してください。

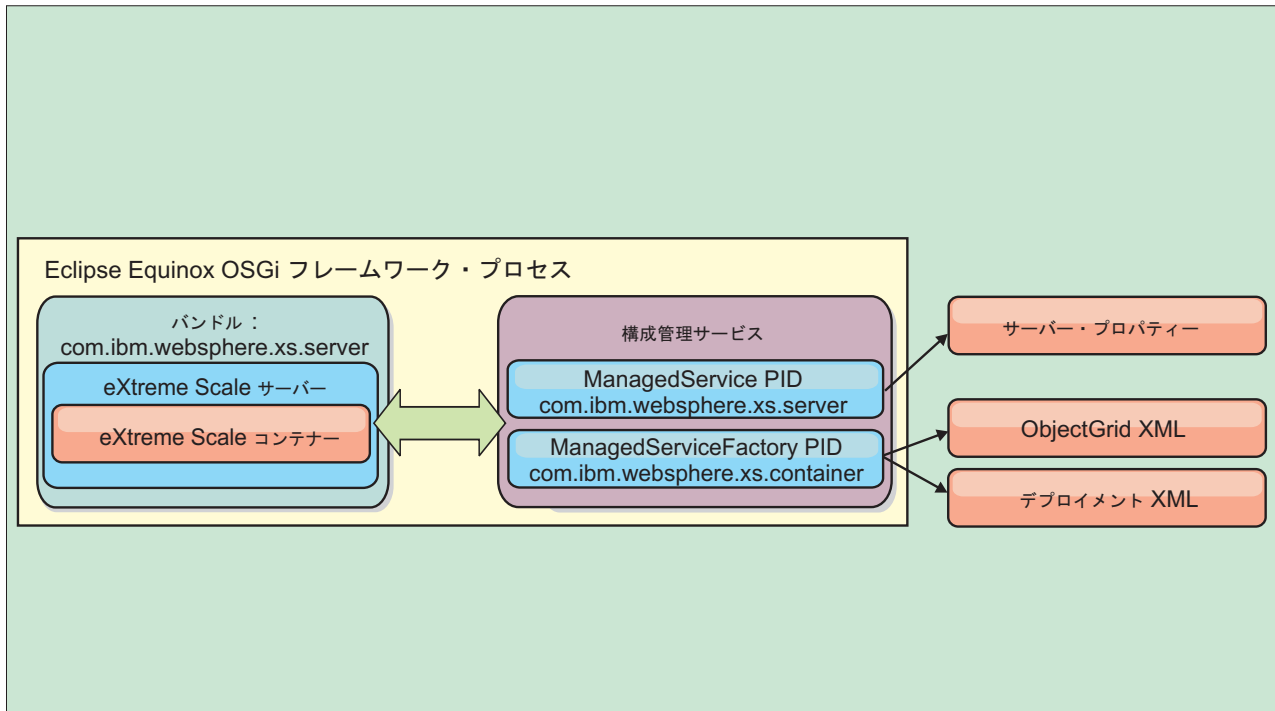


図 52. OSGi バンドルの外部で構成およびメタデータを指定するための Eclipse Equinox プロセス

- プログラムによる構成

カスタマイズされた構成ソリューションをサポートします。

いずれの場合にも、eXtreme Scale サーバーの singleton が構成され、1 つ以上のコンテナが構成されます。

eXtreme Scale サーバー・バンドル objectgrid.jar には、OSGi フレームワークの中で eXtreme Scale グリッド・コンテナを開始して実行するのに必要なすべてのライブラリーが含まれます。サーバー・ランタイム環境は、OSGi サービス・マネージャーを使用して、ユーザー提供のプラグインおよびデータ・オブジェクトと対話します。

**重要:** eXtreme Scale サーバー・バンドルが始動され、eXtreme Scale サーバーが初期化された後に、eXtreme Scale サーバーを再始動することはできません。eXtreme Scale サーバーを再始動するには、Eclipse Equinox プロセスを再開する必要があります。

Spring 名前空間に対する eXtreme Scale サポートを使用して、Blueprint XML ファイルで eXtreme Scale コンテナ・サーバーを構成できます。サーバーおよびコンテナの XML エレメントが Blueprint XML ファイルに追加されると、eXtreme Scale 名前空間ハンドラーが、バンドルの始動時に Blueprint XML ファイルで定義されるパラメーターを使用して、コンテナ・サーバーを自動的に始動します。バンドルが停止されると、バンドルはコンテナを停止します。

Blueprint XML で eXtreme Scale コンテナ・サーバーを構成するには、次のステップを実行します。



## 手順

- OSGi Blueprint を使用して、eXtreme Scale コンテナ・サーバーを始動します。
  1. コンテナ・バンドルを作成します。
  2. コンテナ・バンドルを Eclipse Equinox OSGi フレームワークにインストールします。『OSGi 対応プラグインのインストールと開始』を参照してください。
  3. コンテナ・バンドルを開始します。
- OSGi Configuration Admin を使用して、eXtreme Scale コンテナ・サーバーを始動します。
  1. Config Admin を使用して、サーバーおよびコンテナを構成します。
  2. eXtreme Scale サーバー・バンドルが開始されるか、Config Admin によって永続 ID が作成されると、サーバーおよびコンテナは自動的に始動します。
- ServerFactory API を使用して、eXtreme Scale コンテナ・サーバーを始動します。サーバー API 資料を参照してください。
  1. OSGi バンドル・アクティベーター・クラスを作成し、eXtreme Scale ServerFactory API を使用してサーバーを始動します。

---

## OSGi 対応プラグインのインストールと開始

このタスクでは、動的プラグイン・バンドルを OSGi フレームワークにインストールします。その後、そのプラグインを開始します。

### 始める前に

このトピックは、以下のタスクが完了していることを前提としています。

- eXtreme Scale サーバーまたはクライアント・バンドルを Eclipse Equinox OSGi フレームワークにインストール済みである。203 ページの『eXtreme Scale バンドルのインストール』を参照してください。
- 1 つ以上の動的 BackingMap または ObjectGrid プラグインを実装済みである。eXtreme Scale 動的プラグインのビルドを参照してください。
- 動的プラグインを OSGi バンドル内に OSGi サービスとしてパッケージ化済みである。

### このタスクについて

このタスクでは、Eclipse Equinox コンソールを使用してバンドルをインストールする方法を説明します。バンドルはいくつかの異なる方式 (config.ini 構成ファイルを変更するなど) を使用してインストールできます。Eclipse Equinox を組み込む製品には、バンドルを管理するための代替の方式があります。Eclipse Equinox で config.ini ファイルにバンドルを追加する方法については、「Eclipse runtime options」を参照してください。

OSGi では、重複サービスを持つバンドルの開始が許可されます。WebSphere eXtreme Scale は最新のサービス・ランキングを使用します。1 つの eXtreme Scale データ・グリッド内で複数の OSGi フレームワークを開始する場合は、各サーバーで正しいサービス・ランキングが開始されるようにしなければなりません。そうしないと、いろいろなバージョンが混在したグリッドが開始されます。

データ・グリッドで使用中のバージョンを確認するには、xscmd ユーティリティーを使用して、現在のランキングと使用可能なランキングを確認します。使用可能なサービス・ランキングの詳細については、494 ページの『xscmd による eXtreme Scale プラグインの OSGi サービスの更新』を参照してください。

## 手順

OSGi コンソールを使用してプラグイン・バンドルを Eclipse Equinox OSGi フレームワークにインストールします。

1. コンソールを有効にするよう指定して Eclipse Equinox フレームワークを開始します。例えば、次のようにします。

```
<java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

2. Equinox コンソールで、プラグイン・バンドルをインストールします。

```
osgi> install file:///<path to bundle>
```

Equinox が、新しくインストールされたバンドルのバンドル ID を表示します。

```
Bundle id is 17
```

3. Equinox コンソールで、次の行を入力してバンドルを開始します。ここで、<id> は、バンドルのインストール時に割り当てられたバンドル ID です。

```
osgi> start <id>
```

4. Equinox コンソールで、サービス状況を取得して、バンドルが開始したことを確認します。

```
osgi> ss
```

バンドルが正常に開始した場合、バンドルは ACTIVE 状態を表示します。例えば、次のとおりです。

```
17      ACTIVE      com.mycompany.plugin.bundle_VRM
```

config.ini ファイルを使用して、プラグイン・バンドルを Eclipse Equinox OSGi フレームワークにインストールします。

5. プラグイン・バンドルを次の例のような Eclipse Equinox プラグイン・ディレクトリーにコピーします。

```
<equinox_root>/plugins
```

6. Eclipse Equinox config.ini 構成ファイルを編集し、バンドルを osgi.bundles プロパティーに追加します。例えば、次のとおりです。

```
osgi.bundles=¥  
org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, ¥  
org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, ¥  
org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, ¥  
com.mycompany.plugin.bundle_VRM.jar@1:start
```

**重要:** 最後のバンドル名の後に空白行が存在することを確認してください。各バンドルはコンマで区切ります。

7. コンソールを有効にするよう指定して Eclipse Equinox フレームワークを開始します。例えば、次のようにします。

```
<java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

8. Equinox コンソールで、サービス状況を取得して、バンドルが開始したことを確認します。例えば、次のようにします。

```
osgi> ss
```

バンドルが正常に開始した場合、バンドルは **ACTIVE** 状態を表示します。例えば、次のとおりです。

```
17      ACTIVE      com.mycompany.plugin.bundle_VRM
```

## タスクの結果

これでプラグイン・バンドルがインストールされ、開始されました。今度は、eXtreme Scale コンテナまたはクライアントを開始できます。eXtreme Scale プラグインの作成法の詳細については、システム API とプラグインのトピックを参照してください。

---

## xscmd ユーティリティによる OSGi 対応サービスの管理

**xscmd** ユーティリティを使用して、各コンテナが使用しているサービスとサービス・ランキングを表示したり、バンドルの新しいバージョンを使用するようランタイム環境を更新したりするなど、管理者用タスクを実行できます。

### このタスクについて

Eclipse Equinox OSGi フレームワークでは、同一バンドルの複数バージョンをインストールでき、それらのバンドルを実行時に更新できます。WebSphere eXtreme Scale は、多数の OSGi フレームワーク・インスタンス内でコンテナ・サーバーを実行する分散環境です。

手動で OSGi フレームワークにバンドルをコピーしたり、インストールしたり、それらのバンドルを開始したりする作業は管理者の担当です。eXtreme Scale には、ObjectGrid 記述子 XML ファイル内で eXtreme Scale プラグインとして識別されたサービスを追跡する OSGi ServiceTrackerCustomizer が組み込まれています。**xscmd** ユーティリティを使用すると、プラグインのどのバージョンが使用されているか、どのようなバージョンが使用可能かを確認でき、バンドル・アップグレードも実行できます。

eXtreme Scale はサービス・ランキング番号を使用して、各サービスのバージョンを識別します。参照先が同じサービスが 2 つ以上ロードされると、eXtreme Scale は、ランキングが最も高いサービスを自動的に使用します。

### 手順

- **osgiCurrent** コマンドを実行して、各 eXtreme Scale サーバーが正しいサービス・ランキングのプラグインを使用していることを確認します。

eXtreme Scale はランキングが最も高いサービス参照を自動的に選択するため、複数ランキングのプラグイン・サービスが設定されたデータ・グリッドが開始される可能性があります。

コマンドがランキングの不一致を検出するか、サービスを検出できない場合は、ゼロ以外のエラー・レベルが設定されます。コマンドが正常に完了した場合、エラー・レベルは 0 に設定されます。

次の例は、4つのサーバーの同一グリッドに2つのプラグインがインストールされている場合の **osgiCurrent** コマンドの出力を示します。loaderPlugin プラグインはランキング 1 を使用し、txCallbackPlugin プラグインはランキング 2 を使用しています。

```
OSGi Service Name Current Ranking ObjectGrid Name MapSet Name Server Name
-----
loaderPlugin      1           MyGrid      MapSetA     server1
loaderPlugin      1           MyGrid      MapSetA     server2
loaderPlugin      1           MyGrid      MapSetA     server3
loaderPlugin      1           MyGrid      MapSetA     server4
txCallbackPlugin  2           MyGrid      MapSetA     server1
txCallbackPlugin  2           MyGrid      MapSetA     server2
txCallbackPlugin  2           MyGrid      MapSetA     server3
txCallbackPlugin  2           MyGrid      MapSetA     server4
```

次の例は、新しいランキングの loaderPlugin が設定された server2 を開始した場合の **osgiCurrent** コマンドの出力を示します。

```
OSGi Service Name Current Ranking ObjectGrid Name MapSet Name Server Name
-----
loaderPlugin      1           MyGrid      MapSetA     server1
loaderPlugin      2           MyGrid      MapSetA     server2
loaderPlugin      1           MyGrid      MapSetA     server3
loaderPlugin      1           MyGrid      MapSetA     server4
txCallbackPlugin  2           MyGrid      MapSetA     server1
txCallbackPlugin  2           MyGrid      MapSetA     server2
txCallbackPlugin  2           MyGrid      MapSetA     server3
txCallbackPlugin  2           MyGrid      MapSetA     server4
```

- **osgiAll** コマンドを実行して、各 eXtreme Scale コンテナ・サーバーで正しいプラグイン・サービスが開始されたことを確認します。

ObjectGrid 構成が参照しているサービスを含んでいるバンドルが開始すると、eXtreme Scale ランタイム環境はプラグインを自動的に追跡します。ただし、すぐには使用しません。**osgiAll** コマンドは、各サーバーで使用可能なプラグインを表示します。

パラメーターを指定せずに実行すると、すべてのグリッドおよびサーバーのすべてのサービスが表示されます。追加のフィルター (**-serviceName <service\_name>** フィルターなど) を指定して、単一サービスやデータ・グリッドのサブセットなどに出力を制限できます。

次の例は、2つのサーバーで2つのプラグインが開始された場合の **osgiAll** コマンドの出力を示します。loaderPlugin はランキング 1 と 2 の両方が開始済みで、txCallbackPlugin はランキング 1 が開始済みです。出力の最後にあるサマリー・メッセージから、両方のサーバーが同じサービス・ランキングを認識していることを確認できます。

```
Server: server1
  OSGi Service Name Available Rankings
  -----
  loaderPlugin      1, 2
  txCallbackPlugin  1

Server: server2
  OSGi Service Name Available Rankings
  -----
  loaderPlugin      1, 2
```

```
txCallbackPlugin 1
```

Summary - All servers have the same service rankings.

次の例は、server1 でランキング 1 の loaderPlugin を含んでいるバンドルが停止した場合の **osgiAll** コマンドの出力を示します。出力の下部にあるサマリー・メッセージから、現在 server1 にはランキング 1 の loaderPlugin がないことを確認できます。

```
Server: server1
OSGi Service Name Available Rankings
-----
loaderPlugin      2
txCallbackPlugin  1
```

```
Server: server2
OSGi Service Name Available Rankings
-----
loaderPlugin      1, 2
txCallbackPlugin  1
```

Summary - The following servers are missing service rankings:

```
Server OSGi Service Name Missing Rankings
-----
server1 loaderPlugin      1
```

次の例は、**-sn** 引数を使用してサービス名が指定されたときに、そのサービスが存在しない場合の出力を示します。

```
Server: server2
OSGi Service Name Available Rankings
-----
invalidPlugin     No service found
```

```
Server: server1
OSGi Service Name Available Rankings
-----
invalidPlugin     No service found
```

Summary - All servers have the same service rankings.

- **osgiCheck** コマンドを実行して、プラグイン・サービスとランキングのセットをチェックし、それらが使用可能かどうか確認します。

**osgiCheck** コマンドは、**-serviceRankings <serviceName>;<ranking>[,<serviceName>;<ranking>]** の形式で、サービス・ランキングのセットを 1 つ以上受け入れます。

ランキングがすべて使用可能な場合、メソッドはエラー・レベル 0 を返します。使用不可のランキングが 1 つ以上ある場合は、ゼロ以外のエラー・レベルが設定されます。指定されたサービス・ランキングを含んでいないすべてのサーバーの表が表示されます。追加フィルターを使用して、eXtreme Scale ドメイン内の使用可能なサーバーのサブセットにサービス・チェックを制限できます。

例えば、指定されたランキングまたはサービスがない場合は、次のメッセージが表示されます。

```
Server OSGi Service Unavailable Rankings
-----
server1 loaderPlugin 3
server2 loaderPlugin 3
```

- **osgiUpdate** コマンドを実行して、単一 ObjectGrid および MapSet 内のすべてのサーバーを対象に 1 つ以上のプラグインのランキングを 1 回の操作で更新します。

コマンドは、`-serviceRankings <serviceName>;<ranking>[,<serviceName>;<ranking>] -g <grid name> -ms <mapset name>` の形式で、サービス・ランキングのセットを 1 つ以上受け入れます。

このコマンドでは、以下の操作を実行できます。

- 指定されたサービスが各サーバーで更新のために使用可能なことを確認する。
- **StateManager** インターフェースを使用してグリッドの状態をオフラインに変更する。詳しくは、477 ページの『ObjectGrid の可用性の管理』を参照してください。このプロセスはグリッドを静止し、実行中のトランザクションがすべて完了するまで待機し、新規トランザクションを開始できないようにします。またこのプロセスは、トランザクション・アクティビティーを停止するように **ObjectGridLifecycleListener** プラグインと **BackingMapLifecycleListener** プラグインにシグナル通知します。イベント・リスナー・プラグインの詳細については、イベント・リスナーの指定のためのプラグインを参照してください。
- 新しいサービス・バージョンを使用するように OSGi フレームワーク内で実行中の各 **eXtreme Scale** コンテナを更新する。
- グリッドの状態をオンラインに変更し、トランザクションを継続できるようにする。

更新処理はべき等のプロセスであるため、クライアントがいずれかのタスクを完了できない場合は、操作がロールバックされることとなります。クライアントがロールバックを実行できない場合またはクライアントが更新処理中に中断された場合は、同じコマンドを再実行でき、クライアントは適切なステップから続行します。

クライアントがプロセスを続行できず、別のクライアントからプロセスが再始動された場合、`-force` オプションを使用すると、クライアントが更新を実行できるようになります。**osgiUpdate** コマンドは、複数のクライアントが同一マップ・セットを同時に更新しないようにします。**osgiUpdate** コマンドの詳細については、『**xscmd** による eXtreme Scale プラグインの OSGi サービスの更新』を参照してください。

## **xscmd による eXtreme Scale プラグインの OSGi サービスの更新**

WebSphere eXtreme Scale は、グリッドがアクティブであってもコンテナ・サーバー・プラグイン・バンドルをアップグレードできるようサポートします。このサポートにより、管理者はグリッド・プロセスを再始動しなくともアプリケーションの更新や追加を完了できます。

### **始める前に**

eXtreme Scale OSGi バンドルを新しいバージョンに更新する前に、次のステップを完了してください。



1. サポートされる OSGi フレームワーク内で eXtreme Scale サーバーを開始します。
2. すべての eXtreme Scale プラグインをバンドルに分割します。各バンドルは、サービス・ランキングを使用して、それぞれのプラグインのバージョンを識別する必要があります。
3. キャッシュ・オブジェクトを Java プリミティブ型 (byte[], Integer, String など) として指定するか、MapSerializerPlugin プラグインを使用して保管しなければなりません。データ・オブジェクトは eXtreme Scale バンドル内に保管され、アップグレードされません。データと相互作用するプラグインのみが更新されません。
4. バージョン互換のあるキャッシュ・オブジェクト・データを設計します。新しいプラグインは、古いプラグインによって作成されたデータと相互作用できなければなりません。
5. ObjectGridLifecycle および BackingMapLifecycle イベントを listen するようプラグインを設計して、それらのプラグイン内にある他のプラグインまたはメタデータへの参照をリフレッシュするような設計にしてください。このようなアプローチを使用すると、メインのプラグインが更新されたとき、それらの参照プラグインもリフレッシュされます。
6. eXtreme Scale OSGi 更新処理はサーバーのみに影響します。プラグインを使用しているクライアントがある場合、別途それらのクライアントを更新しなければなりません。

## このタスクについて

OSGi が使用可能でないと、管理者がアプリケーション・プラグインまたはキャッシュ・オブジェクトを更新する必要がある場合、各グリッド・ノードを 1 つずつアップグレードしなければならず、ネットワーク、メモリー、および CPU の利用にストレスを与えます。プラグインとキャッシュ Java オブジェクトは直接グリッド内に保管されるため、このような操作が必要になります。各クラスの ClassLoader は異なるため、プロセスを再始動せずにクラスを更新すると、グリッド・プラグインに矛盾が生じます。

eXtreme Scale 製品には xscmd ユーティリティーと MBean が組み込まれています。管理者はそれらを使用して、各グリッド・コンテナをホスティングしている OSGi フレームワークにインストールされているすべてのプラグイン・バンドルを表示し、どのバージョンを使用するか選択できます。xscmd を使用してプラグインを新しいランキングに更新すると、グリッドが静止してすべてのトランザクションが排出され、プラグインが更新された後、グリッドが再度アクティブ化されます。更新処理中にエラーが発生すると、プロセスはロールバックされ、古いランキングが復元されます。

## 手順

1. バンドルのバージョンを作成し、バンドル・マニフェスト内のバージョン番号を上げ、各 eXtreme Scale プラグイン・サービスのランキングも大きくします。元のバンドル・バージョンが Bundle-Version: 1.0.0 であれば、次のバージョンは Bundle-Version: 1.1.0 と定義できます。

元のサービス・ランキングが `ranking="1"` であれば、次のランキングは `ranking="2"` と定義できます。

**重要:** OSGi サービス・ランキングは整数でなければなりません。

2. 新規バンドルを eXtreme Scale コンテナ・サーバーをホスティングしている各 OSGi フレームワーク・ノードにコピーします。

3. 新規バンドルを OSGi フレームワークにインストールします。バンドルには、バンドル ID が割り当てられます。例えば、次のようになります。

```
osgi> install <URL to bundle>
```

4. 割り当てられたバンドル ID を使用して、新規バンドルを開始します。例えば、次のようになります。

```
osgi> start <id>
```

新規バンドルが開始されると、eXtreme Scale OSGi サービス・トラッカーがバンドルを検出し、更新可能な状態にします。

5. **xscmd -c osgiAll** コマンドを使用して、各コンテナ・サーバーが新規バンドルを認識していることを確認します。**osgiAll** コマンドは、グリッド内のすべてのコンテナに ObjectGrid 記述子 XML ファイル内で参照されているすべてのサービスについて照会し、使用可能なすべてのランキングを表示します。例えば、次のようになります。

```
xscmd -c osgiAll
```

```
Server: server1
  OSGi Service Name      Available Rankings
  -----
  myLoaderServiceFactory 1, 2
  mySerializerServiceFactory 1, 2
```

```
Server: server2
  OSGi Service Name      Available Rankings
  -----
  myLoaderServiceFactory 1, 2
  mySerializerServiceFactory 1, 2
```

Summary - All servers have the same service rankings.

6. **xscmd -c osgiCheck** コマンドを使用して、1 つ以上のサービス・ランキングが更新の有効なターゲットであるか確認します。例えば、次のようになります。

```
xscmd -c osgiCheck -sr
mySerializerServiceFactory;2,myLoaderServiceFactory;2
```

CWXSIO040I: The command osgiCheck has completed successfully.

7. **osgiCheck** コマンドの結果に何もエラーがなければ、更新処理中の障害に備えて配置サービスのバランサーを中断して、断片移動を回避します。配置を中断するには、更新の影響を受けるすべてのオブジェクト・グリッドとマップ・セットに対して **xscmd -c suspendBalancing** コマンドを使用します。例えば、次のようになります。

```
xscmd -c suspendBalancing -g MyGrid -ms MyMapSet
```

8. 各オブジェクト・グリッドとマップ・セットのバランシングが中断状態になったら、再度 **xscmd -c osgiCheck** コマンドを使用して、1 つ以上のサービス・ランキングが更新の有効なターゲットであるか確認します。例えば、次のようになります。

```
xscmd -c osgiCheck -sr
mySerializerServiceFactory;2,myLoaderServiceFactory;2
```

CWXS10040I: The command osgiCheck has completed successfully.

- オブジェクト・グリッドとマップ・セットのバランシングが中断状態になったら、**osgiUpdate** コマンドを使用して、オブジェクト・グリッドとマップ・セットのすべてのサーバーでサービスを更新します。例えば、次のようになります。

```
xscmd -c osgiUpdate -sr
mySerializerServiceFactory;2,myLoaderServiceFactory;2 -g MyGrid -ms MyMapSet
```

- アップグレードが成功したことを確認します。例えば、次のようになります。

Update succeeded for the following service rankings:

Service	Ranking
-----	-----
mySerializerServiceFactory	2
myLoaderServiceFactory	2

- ランキングが正常に更新されたことを確認したら、**xscmd -c resumeBalancing** コマンドを使用して、バランシングを再度使用可能にします。例えば、次のようになります。

```
xscmd -c resumeBalancing -g MyGrid -ms MyMapSet
```

- eXtreme Scale コンテナをホスティングしている各 OSGi フレームワーク内の古いバンドルを停止し、アンインストールします。例えば、次のコードを Eclipse Equinox コンソールで入力します。

```
osgi> stop <id>
osgi> uninstall <id>
```

## タスクの結果

eXtreme Scale バンドルが新しいバージョンに更新されました。

---

## Managed Beans (MBeans) を使用した管理

デプロイメントを管理およびモニターするには、さまざまなタイプの Java Management Extensions (JMX) MBeans を使用できます。各 MBean は、マップ、データ・グリッド、サーバー、サービスなどの特定のエンティティを参照します。

### JMX MBean インターフェースおよび WebSphere eXtreme Scale

各 MBean には、属性値を表す `get` メソッドがあります。この `get` メソッドは、プログラムから直接呼び出すことはできません。JMX 仕様では、属性の扱い方が操作のときと異なります。ベンダー JMX コンソールを使用して属性を表示し、プログラムまたはベンダー JMX コンソールで操作を実行することができます。

### パッケージ `com.ibm.websphere.objectgrid.management`

使用可能なすべての MBean の概要と詳細なプログラミング仕様については、次の API 資料を参照してください。パッケージ `com.ibm.websphere.objectgrid.management`

## wsadmin ツールを使用した Managed Beans (MBeans) へのアクセス

WebSphere Application Server で提供される wsadmin ユーティリティを使用し、Managed Bean (MBean) 情報にアクセスすることができます。

### 手順

WebSphere Application Server インストール内の bin ディレクトリーから wsadmin ツールを実行します。次の例は、動的 eXtreme Scale における現在の断片配置のビューを取得するものです。wsadmin ツールは、eXtreme Scale が稼働している任意のインストール済み環境から実行できます。wsadmin ツールをカタログ・サービスで実行する必要はありません。

```
$ wsadmin.sh -lang jython
wsadmin>placementService = AdminControl.queryNames
("com.ibm.websphere.objectgrid:*,type=PlacementService")
wsadmin>print AdminControl.invoke(placementService,
"listObjectGridPlacement","library ms1")

<objectGrid name="library" mapSetName="ms1">
  <container name="container-0" zoneName="DefaultDomain"
    hostName="host1.company.org" serverName="server1">
    <shard type="Primary" partitionName="0"/>
    <shard type="SynchronousReplica" partitionName="1"/>
  </container>
  <container name="container-1" zoneName="DefaultDomain"
    hostName="host2.company.org" serverName="server2">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="Primary" partitionName="1"/>
  </container>
  <container name="UNASSIGNED" zoneName=" ibm SYSTEM"
    hostName="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
  </container>
</objectGrid>
```

## Managed Bean (MBean) へのプログラマチックなアクセス

MBean への接続に Java アプリケーションを使用できます。これらのアプリケーションは、com.ibm.websphere.objectgrid.management パッケージ内のインターフェースを使用します。

### このタスクについて

MBean にアクセスするためのプログラマチックな方式は、接続先のサーバーのタイプによって異なります。

- スタンドアロン・カタログ・サービス MBean サーバーへの接続
- コンテナ MBean サーバーへの接続
- WebSphere Application Server 内でホスティングされるカタログ・サービス MBean サーバーへの接続
- セキュリティーが使用可能なカタログ・サービス MBean サーバーへの接続

### 手順

- スタンドアロン・カタログ・サービス MBean サーバーへの接続

次のサンプル・プログラムは、スタンドアロン・カタログ・サービス MBean サーバーに接続し、指定された ObjectGrid および MapSet の各コンテナ・サーバーとコンテナ・サーバーそれぞれに割り振られた断片のリストを XML フォーマットのストリングで返します。

```
package com.ibm.websphere.sample.xs.admin;

import java.util.Set;

import javax.management.MBeanServerConnection;
import javax.management.ObjectName;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;

/**
 * Collects the placement information from the Catalog Server for a given ObjectGrid.
 */
public final class CollectPlacementPlan {
    private static String hostName = "localhost";

    private static int port = 1099;

    private static String objectGridName = "library";

    private static String mapSetName = "ms1";

    /**
     * Connects to the ObjectGrid Catalog Service to retrieve placement information and
     * prints it out.
     *
     * @param args
     * @throws Exception
     *         If there is a problem connecting to the catalog service MBean server.
     */
    public static void main(String[] args) throws Exception {
        String serviceURL = "service:jmx:rmi:///jndi/rmi://" + hostName + ":" + port +
            "/objectgrid/MBeanServer";
        JMXServiceURL jmxUrl = new JMXServiceURL(serviceURL);
        JMXConnector jmxCon = JMXConnectorFactory.connect(jmxUrl);

        try {
            MBeanServerConnection catalogServerConnection = jmxCon.getMBeanServerConnection();

            Set placementSet = catalogServerConnection.queryNames(new
                ObjectName("com.ibm.websphere.objectgrid"
                    + ".*,type=PlacementService"), null);
            ObjectName placementService = (ObjectName) placementSet.iterator().next();
            Object placementXML = catalogServerConnection.invoke(placementService,
                "listObjectGridPlacement", new Object[] {
                    objectGridName, mapSetName }, new String[] { String.class.getName(),
                String.class.getName() });
            System.out.println(placementXML);
        } catch (Exception e) {
            if(jmxCon != null) {
                jmxCon.close();
            }
        }
    }
}
```

図 53. *CollectPlacementPlan.java*

サンプル・プログラムに関する注記:

- カタログ・サービスの **JMXServiceURL** の値は、常に次のフォームです。  
`service:jmx:rmi:///jndi/rmi://<host>:<port>/objectgrid/MBeanServer`。ここで、`<host>` はカタログ・サービスを実行しているホストで、`<port>` はカタログ・サービスを開始するとき **-JMXServicePort** オプションで指定した JMX サービス・ポートです。ポートが指定されない場合、デフォルトは 1099 です。
- ObjectGrid またはマップの統計を使用可能にするには、ObjectGrid コンテナの開始時、サーバー・プロパティー・ファイル内にプロパティー `statsSpec=all=enabled` が指定されている必要があります。
- コンテナ・サーバー内で稼働する MBean を使用不可にするには、サーバー・プロパティー・ファイル内にプロパティー `enableMBeans=false` を指定してください。

出力の例を次に示します。この出力から、2 つのコンテナ・サーバーがアクティブなことがわかります。Container-0 コンテナ・サーバーは 4 つのプライマリ断片をホスティングしています。Container-1 コンテナ・サーバーは、Container-0 コンテナ・サーバー上の各プライマリ断片の同期レプリカをホスティングしています。この構成では、2 つの同期レプリカと 1 つの非同期レプリカが構成されています。結果として、Unassigned コンテナ・サーバーに残りの断片が残されています。さらに 2 つのコンテナ・サーバーが開始すると、Unassigned コンテナ・サーバーは表示されなくなります。

```
<objectGrid name="library" mapSetName="ms1">
  <container name="Container-1" zoneName="DefaultZone"
    hostname="myhost.mycompany.com" serverName="ogserver">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="SynchronousReplica" partitionName="1"/>
    <shard type="SynchronousReplica" partitionName="2"/>
    <shard type="SynchronousReplica" partitionName="3"/>
  </container>
  <container name="Container-0" zoneName="DefaultZone"
    hostname="myhost.mycompany.com" serverName="ogserver">
    <shard type="Primary" partitionName="0"/>
    <shard type="Primary" partitionName="1"/>
    <shard type="Primary" partitionName="2"/>
    <shard type="Primary" partitionName="3"/>
  </container>
  <container name="library:ms1:_UnassignedContainer_" zoneName="_ibm_SYSTEM"
    hostname="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="SynchronousReplica" partitionName="1"/>
    <shard type="SynchronousReplica" partitionName="2"/>
    <shard type="SynchronousReplica" partitionName="3"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="1"/>
    <shard type="AsynchronousReplica" partitionName="2"/>
    <shard type="AsynchronousReplica" partitionName="3"/>
  </container>
</objectGrid>
```

#### • コンテナ MBean サーバーへの接続

コンテナ・サーバーは MBean をホスティングして、コンテナ・サーバー内で実行している個々のマップや ObjectGrid インスタンスに関する情報を照会します。次のサンプル・プログラムは、JMX アドレスが `localhost:1099` のカタログ・サーバーによってホスティングされる各コンテナ・サーバーの状況をプリントします。



```

package com.ibm.websphere.sample.xs.admin;

import java.util.List;
import java.util.Set;

import javax.management.MBeanServerConnection;
import javax.management.ObjectInstance;
import javax.management.ObjectName;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;

/**
 * Collects placement status from each of the available containers directly.
 */
public final class CollectContainerStatus {
    private static String hostName = "localhost";

    private static int port = 1099;

    /**
     * @param args
     */
    public static void main(String[] args) throws Exception {
        String serviceURL = "service:jmx:rmi:///jndi/rmi://" + hostName + ":" + port + "/objectgrid/MBeanServer";
        JMXServiceURL jmxUrl = new JMXServiceURL(serviceURL);
        JMXConnector jmxCon = JMXConnectorFactory.connect(jmxUrl);

        try {
            MBeanServerConnection catalogServerConnection = jmxCon.getMBeanServerConnection();

            Set placementSet = catalogServerConnection.queryNames(new ObjectName("com.ibm.websphere.objectgrid"
                + ".*:*,type=PlacementService"), null);

            ObjectName placementService = (ObjectName) placementSet.iterator().next();
            List<String> containerJMXAddresses = (List<String>) catalogServerConnection.invoke(placementService,
                "retrieveAllServersJMXAddresses", new Object[0], new String[0]);
            for (String address : containerJMXAddresses) {
                JMXServiceURL containerJMXURL = new JMXServiceURL(address);
                JMXConnector containerConnector = JMXConnectorFactory.connect(containerJMXURL);
                MBeanServerConnection containerConnection = containerConnector.getMBeanServerConnection();
                Set<ObjectInstance> containers = containerConnection.queryMBeans(
                    new ObjectName("*:*,type=ObjectGridContainer"), null);
                for (ObjectInstance container : containers) {
                    System.out.println(containerConnection.getAttribute(container.getObjectName(), "Status"));
                }
            }
        } finally {
            if(jmxCon != null) {
                jmxCon.close();
            }
        }
    }
}

```

図 54. *CollectContainerStatus.java*

サンプル・プログラムが各コンテナのコンテナ・サーバーの状況をプリントします。出力の例を次に示します。

```

<container name="Container-0" zoneName="DefaultZone" hostName="descartes.rchland.ibm.com"
  serverName="ogserver">
  <shard type="Primary" partitionName="1"/>
  <shard type="Primary" partitionName="0"/>
  <shard type="Primary" partitionName="3"/>
  <shard type="Primary" partitionName="2"/>
</container>

```

- **WebSphere Application Server 内でホスティングされるカタログ・サービス MBean サーバーへの接続**

WebSphere Application Server 内で MBean にプログラマチックにアクセスする方式は、スタンドアロン構成内で MBean にアクセスするときと少し違います。

1. MBean サーバーに接続する Java プログラムを作成してコンパイルします。  
以下にサンプル・プログラムを示します。

```

package com.ibm.websphere.sample.xs.admin;

import java.util.Set;

import javax.management.MBeanServerConnection;
import javax.management.ObjectName;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;

/**
 * Collects the placement information from the catalog server running in a deployment manager for a given ObjectGrid.
 */
public final class CollectPlacementPlanWAS {
    private static String hostName = "localhost";

    private static int port = 9809;

    private static String objectGridName = "library";

    private static String mapSetName = "ms1";

    /**
     * Connects to the catalog service to retrieve placement information and prints it out.
     *
     * @param args
     * @throws Exception
     *         If there is a problem connecting to the catalog service MBean server.
     */
    public static void main(String[] args) throws Exception {

        // connect to bootstrap port of the deployment manager
        String serviceURL = "service:jmx:iiop://" + hostName + ":" + port + "/jndi/JMXConnector";
        JMXServiceURL jmxUrl = new JMXServiceURL(serviceURL);
        JMXConnector jmxCon = JMXConnectorFactory.connect(jmxUrl);

        try {
            MBeanServerConnection catalogServerConnection = jmxCon.getMBeanServerConnection();

            Set placementSet =
            catalogServerConnection.queryNames(new ObjectName("com.ibm.websphere.objectgrid"
                + ".*:type=PlacementService"), null);

            ObjectName placementService = (ObjectName) placementSet.iterator().next();
            Object placementXML = catalogServerConnection.invoke(placementService,
                "listObjectGridPlacement", new Object[] {
                    objectGridName, mapSetName }, new String[] { String.class.getName(),
                    String.class.getName() });
            System.out.println(placementXML);
        } finally {
            if(jmxCon != null) {
                jmxCon.close();
            }
        }
    }
}

```

図 55. *CollectPlacementPlan.java*

2. 以下のコマンドを実行します。

```

"$JAVA_HOME/bin/java" "$WAS_LOGGING" -Djava.security.auth.login.config="$app_server_root/properties/wsjaas_client.conf" ¥
-Djava.ext.dirs="$JAVA_HOME/jre/lib/ext:$WAS_EXT_DIRS:$WAS_HOME/plugins:$WAS_HOME/lib/WMQ/java/lib" ¥
-Djava.naming.provider.url=<an_IIOP_URL_or_a_corbaloc_URL_to_your_application_server_machine_name> ¥
-Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory ¥
-Dserver.root="$WAS_HOME" "$CLIENTSAS" "$CLIENTSSL" $USER_INSTALL_PROP ¥
-classpath "$WAS_CLASSPATH":<list_of_your_application_jars_and_classes> ¥
<fully_qualified_class_name_to_run> <your_application_parameters>

```

このコマンドは、変数を適切に設定する `was_root/bin/setupCmdLine.sh` スクリプトが実行済みであると想定しています。`java.naming.provider.url` プロパティ値のフォーマットの例は、`corbaloc:iiop:1.0@<host>:<port>/NameService` です。

- **セキュリティーが使用可能なカタログ・サービス MBean サーバーへの接続**

セキュリティーが使用可能なカタログ・サービス MBean への接続の詳細については、588 ページの『Java Management Extensions (JMX) セキュリティー』を参照してください。

## 次のタスク

MBean を使用して統計を表示したり、管理操作を実行したりする方法の例については、`xsadmin` サンプル・アプリケーションを参照してください。`xsadmin` サンプル・アプリケーションのソース・コードは、スタンドアロン・インストールの場合は `wxs_home/samples/xsadmin.jar` ファイルで確認でき、WebSphere Application Server インストールの場合は `wxs_home/xsadmin.jar` ファイルで確認できます。`xsAdmin` サンプル・アプリケーションで実行できる操作の詳細については、サンプル: `xsadmin` ユーティリティーを参照してください。

MBean の詳細については、`com.ibm.websphere.objectgrid.management` パッケージ内でも参照できます。

---

## J2C クライアント接続の管理

WebSphere eXtreme Scale 接続ファクトリーには、アプリケーション間で共有でき、かつアプリケーションの再始動を通じて永続化できる eXtreme Scale クライアント接続が含まれています。

### このタスクについて

クライアント接続には、接続状況情報およびライフサイクル管理操作を提供する管理 Bean が含まれています。

### 手順

クライアント接続を維持します。 `XSCConnectionFactory` 接続ファクトリー・オブジェクトから最初の接続が取得されると、リモート・データ・グリッドへの eXtreme Scale クライアント接続が確立され、`ObjectGridJ2CConnection` MBean が作成されます。クライアント接続は、プロセスが存続する限り維持されます。クライアント接続を終了するには、次のいずれかのイベントを呼び出します。

- リソース・アダプターを停止する。リソース・アダプターを停止することができます。例えば、リソース・アダプターがアプリケーションに組み込まれ、アプリケーションが停止されたときなどです。
- `resetConnection` MBean 操作を `ObjectGridJ2CConnection` MBean で呼び出す。接続がリセットされると、すべての接続が無効にされ、トランザクションが完了し、`ObjectGrid` クライアント接続が破棄されます。その後、接続ファクトリーで `getConnection` メソッドが呼び出されると、新しいクライアント接続が確立されません。

WebSphere Application Server は、J2C 接続の管理、接続プールのモニター、およびパフォーマンスのための追加の管理 Bean も提供します。

## 第 8 章 モニター



付属モニター・コンソール、API、MBean、ログ、およびユーティリティーを使用して、アプリケーション環境のパフォーマンスをモニターできます。

### 統計の概説

WebSphere eXtreme Scale での統計は、内部統計ツリーに作成されます。内部ツリーからは、StatsAccessor API、Performance Monitoring Infrastructure (PMI) モジュール、および MBean API が作成されます。

次の図は、WebSphere eXtreme Scale の統計の一般的なセットアップを示しています。

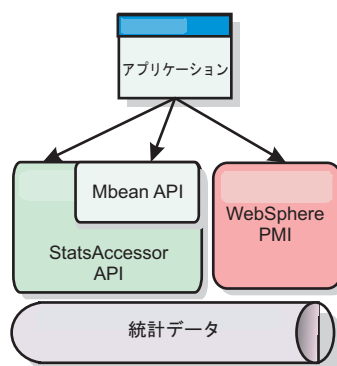


図 56. 統計の概説

これらの API のそれぞれは統計ツリーに対するビューを提供しますが、使用される理由は異なります。

- **統計 API:** 統計 API により、開発者はカスタム MBean やロギングのような柔軟でカスタマイズ可能な統計統合ソリューションのために、統計に直接アクセスできます。
- **MBean API:** MBean API は、モニター用の仕様ベースのメカニズムです。MBean API は、Statistics API を使用し、サーバー Java 仮想マシン (JVM) に対してローカルに実行します。API および MBean の構造は、他のベンダーのユーティリティーと容易に統合できるように設計されています。分散オブジェクト・グリッドを実行中の場合は、MBean API を使用します。
- **WebSphere Application Server Performance Monitoring Infrastructure (PMI) モジュール:** PMI は、WebSphere Application Server 内で WebSphere eXtreme Scale を実行中の場合に使用します。これらのモジュールは、内部統計ツリーのビューを提供します。

## 統計 API

ツリー・マップに非常によく似ており、特定のモジュールを取得するための対応するパスおよびキー、すなわちこの場合は細分度または集約レベルがあります。例えば、ツリー内に常に任意のルート・ノードがあって、「accounting」という名前の ObjectGrid に属している「payroll」という名前のマップに関して統計が収集されると想定します。例えば、マップの集約レベルまたは細分度についてモジュールにアクセスするには、パスの String[] を渡すことができます。この場合、各 String がノードのパスを表わすので、これは String[] {root, "accounting", "payroll"} と同等です。この構造の利点は、ユーザーがパス内のどのノードにも配列を指定でき、そのノードの集約レベルを取得できるという点です。このため、String[] {root, "accounting"} を渡すと、マップ統計が取得されますが、これは「accounting」というグリッド全体に対するものです。これにより、ユーザーは、モニターする統計のタイプを、アプリケーションに必要ななどのような集約レベルでも指定できます。

## WebSphere Application Server PMI モジュール

WebSphere eXtreme Scale には、WebSphere Application Server PMI で使用するための統計モジュールが組み込まれています。WebSphere Application Server プロファイルが WebSphere eXtreme Scale で拡張されると、拡張スクリプトにより WebSphere eXtreme Scale モジュールが自動的に WebSphere Application Server 構成ファイルに統合されます。PMI を使用すると、統計モジュールを使用可能および使用不可にしたり、さまざまな細分度で統計を自動的に集約したり、また組み込み Tivoli Performance Viewer を使用してデータをグラフ化することもできます。詳しくは、529 ページの『WebSphere Application Server PMI によるモニター』を参照してください。

## ベンダー製品と Managed Bean (MBean) との統合

eXtreme Scale API および Managed Bean は、サード・パーティーのモニタリング・アプリケーションと簡単に統合できるように設計されています。eXtreme Scale トポロジに関する情報を分析するために使用できる単純な Java Management Extensions (JMX) コンソールの例のいくつかとして、JConsole や MC4J があります。またプログラマチック API を使用して、eXtreme Scale パフォーマンスのスナップショットを作成するか、そのパフォーマンスを追跡するアダプター実装を作成することもできます。WebSphere eXtreme Scale には、すぐに使用可能なモニター機能を持つサンプルのモニター・アプリケーションが含まれており、拡張したカスタム・モニター・ユーティリティを作成するためのテンプレートとしてこれを使用できます。



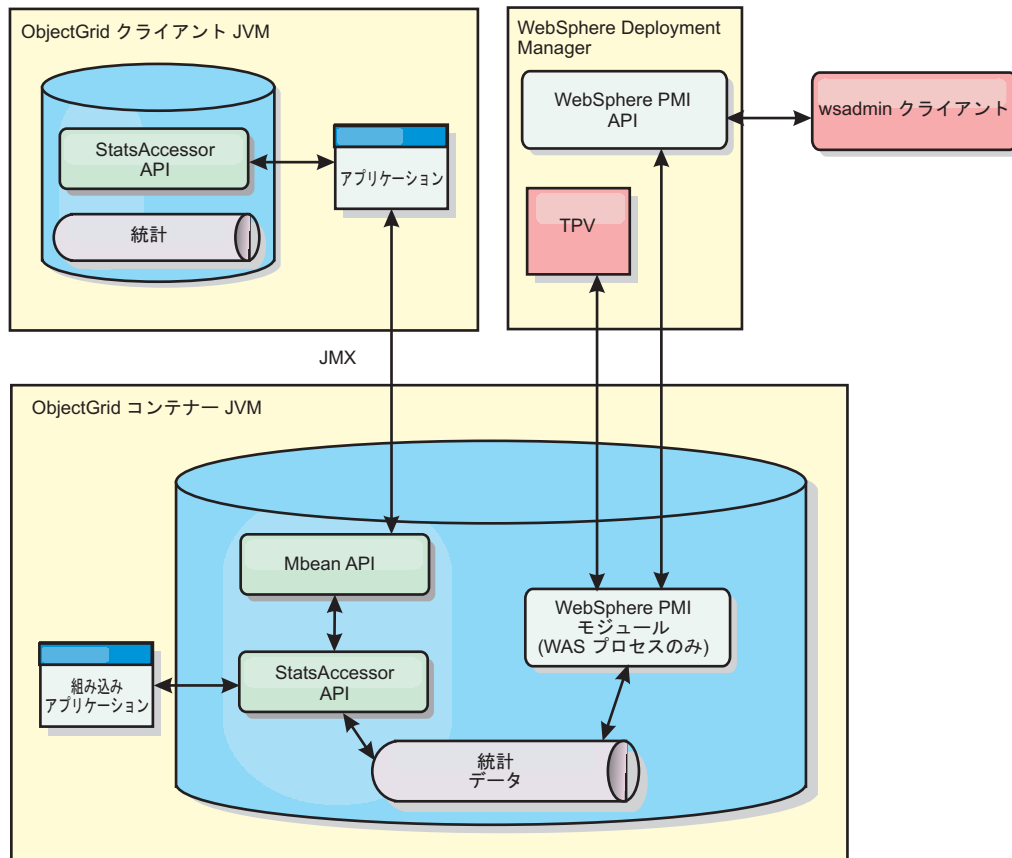


図 57. MBean の概説

詳しくは、サンプル: **xsadmin** ユーティリティを参照してください。特定のベンダー・アプリケーションとの統合について詳しくは、以下のトピックを参照してください。

- IBM Tivoli モニター・エージェントでの eXtreme Scale のモニター
- 553 ページの『Hyperic HQ による eXtreme Scale のモニター』
- 550 ページの『CA Wily Introscope による eXtreme Scale アプリケーションのモニター』

## Web コンソールによるモニター

Web コンソールでは、現在と過去の統計をグラフにできます。このコンソールには、概要を表示するように事前構成されたグラフがいくつか用意されているほか、使用可能な統計からグラフを作成できるカスタム・レポート・ページもあります。WebSphere eXtreme Scale のモニター・コンソールのグラフ機能を使用して、環境内のデータ・グリッドの全体的なパフォーマンスを表示できます。

## Web コンソールの開始とログオン

**startConsoleServer** コマンドを実行してコンソール・サーバーを始動し、デフォルトのユーザー ID とパスワードを使用してサーバーにログオンします。

## 始める前に

### • Web ブラウザー要件

Web コンソールで、次のいずれかのブラウザを使用します。

- Mozilla Firefox、バージョン 3.5.x 以降
- Mozilla Firefox、バージョン 3.6.x 以降
- Microsoft Internet Explorer バージョン 7 または 8

## 手順

1. オプション: デフォルト・ポート以外のポートでコンソール・サーバーを実行する場合は、`wxs_install_root/ObjectGrid/console/config/zero.config` ファイルを編集します。コンソール・サーバーのデフォルト・ポートは、HTTP では 7080、HTTPS では 7443 です。次のプロパティを編集してデフォルト値を変更できます。

```
/config/http/port = 7080  
/config/https/port = 7443
```

既にコンソール・サーバーが始動した後にこれらの値を編集した場合は、新規のポート番号を使用するために、サーバーを再始動します。

2. コンソール・サーバーを始動します。コンソール・サーバーを始動する **startConsoleServer.bat|sh** スクリプトは、インストール済み環境の `wxs_install_root/ObjectGrid/bin` ディレクトリーにあります。
3. コンソールにログオンします。
  - a. Web ブラウザーから、`https://your.console.host:7443` に進み、`your.console.host` を、コンソールをインストールしたサーバーのホスト名に置き換えます。
  - b. コンソールにログオンします。
    - **ユーザー ID:** admin
    - **パスワード:** adminコンソールのウェルカム・ページが表示されます。
4. コンソール構成を編集します。「設定」 > 「構成」をクリックして、コンソール構成を確認します。コンソール構成には、以下のような情報があります。
  - WebSphere eXtreme Scale クライアントのトレース・ストリング (\*=`all=disabled` など)
  - 管理者の名前とパスワード
  - 管理者の E メール・アドレス

## 次のタスク

- 統計の追跡を開始するために、カタログ・サーバーを Web コンソール・サーバーに接続します。詳しくは、509 ページの『Web コンソールのカタログ・サーバーへの接続』を参照してください。
- Web コンソール・サーバーを停止する必要がある場合は、**stopConsoleServer.bat|sh** スクリプトを実行します。このスクリプトは、インストール済み環境の `wxs_install_root/ObjectGrid/bin` ディレクトリーにあります。

## Web コンソールのカタログ・サーバーへの接続

Web コンソールで統計の表示を開始するには、最初に、モニターするカタログ・サーバーに接続する必要があります。カタログ・サーバーのセキュリティーが使用可能になっている場合は、追加のステップが必要です。

### 始める前に

- Web コンソール・サーバーが実行中でなければなりません。詳しくは、507 ページの『Web コンソールの開始とログオン』を参照してください。
- 接続先のカタログ・サーバーの内、少なくとも 1 つが実行中でなければなりません。詳しくは、451 ページの『スタンドアロン・カタログ・サービスの開始』を参照してください。

### 手順

1. カタログ・サーバーの Secure Sockets Layer (SSL) が使用可能である場合は、鍵ストア、トラストストア、およびクライアント・プロパティ・ファイルを構成する必要があります。サーバー・プロパティ・ファイルの中の `transportType` 属性を `SSL-Required` に設定して、カタログ・サーバーの SSL を使用可能にします。
  - a. 鍵ストアおよびトラストストアを構成し、次に、公開証明書を交換するか、クロス・インポートします。例えば、トラストストアおよび鍵ストアを、Web コンソールを実行しているサーバー上の場所にコピーします。
  - b. Web コンソール・サーバー上のクライアント・プロパティ・ファイルを編集して、SSL 構成のプロパティが含まれるようにします。例えば、`wxs_install_root/ObjectGridProperties/sampleclient.properties` ファイルを編集します。Web コンソールからのアウトバウンド SSL 接続には次のプロパティが必要です。

```
#-----  
# SSL Configuration  
#  
# - contextProvider      (IBMJSSE2, IBMJSSE, IBMJSSEFIPS, etc.)  
# - protocol             (SSL, SSLv2, SSLv3, TLS, TLSv1, etc.)  
# - keyStoreType         (JKS, JCEK, PKCS12, etc.)  
# - trustStoreType       (JKS, JCEK, PKCS12, etc.)  
# - keyStore              (fully qualified path to key store file)  
# - trustStore            (fully qualified path to trust store file)  
# - alias                 (string specifying ssl certificate alias to use from keyStore)  
# - keyStorePassword     (string specifying password to the key store - encoded or not)  
# - trustStorePassword   (string specifying password to the trust store - encoded or not)  
#  
# Uncomment these properties to set the SSL configuration.  
#-----  
#alias=clientprivate  
#contextProvider=IBMJSSE  
#protocol=SSL  
#keyStoreType=JKS  
#keyStore=etc/test/security/client.private  
#keyStorePassword={xor}PDM20jErLyy#  
#trustStoreType=JKS  
#trustStore=etc/test/security/server.public  
#trustStorePassword={xor}Lyo9MzY8
```

**重要:** Windows Windows を使用している場合、パス内の円記号 ( ¥ ) はすべてエスケープする必要があります。例えば、パス `C:¥opt¥ibm` を使用する場合、プロパティ・ファイルに `C:¥¥opt¥¥ibm` と入力します。

2. モニター対象のカタログ・サーバーへの接続を確立して維持します。次のステップを繰り返して、それぞれのカタログ・サーバーを構成に追加します。
  - a. 「設定」 > 「eXtreme Scale カタログ・サーバー」をクリックします。
  - b. 新規カタログ・サーバーを追加します。



- 1) 「追加」アイコン ( ) をクリックして、既存のカタログ・サーバーを登録します。
  - 2) ホスト名、リスナー・ポートなどの情報を指定します。ポートの構成およびデフォルトについて詳しくは、53 ページの『ネットワーク・ポートの計画』を参照してください。
  - 3) 「OK」をクリックします。
  - 4) カatalog・サーバーがナビゲーション・ツリーに追加されていることを確認します。
3. カatalog・サービス・ドメインの中に作成するカタログ・サーバーをグループにします。カatalog・サービス・ドメインでセキュリティー設定が構成されているため、カatalog・サーバーでセキュリティーが使用可能にされているときはカatalog・サービス・ドメインを作成する必要があります。
- a. 「設定」 > 「eXtreme Scale ドメイン」ページをクリックします。
  - b. 新規カatalog・サービス・ドメインを追加します。



- 1) 「追加」アイコン ( ) をクリックして、カatalog・サービス・ドメインを登録します。カatalog・サービス・ドメインの名前を入力します。
- 2) カatalog・サービス・ドメインを作成した後、プロパティを編集できます。カatalog・サービス・ドメインのプロパティは次のとおりです。

**Name** 管理者によって割り当てられた、ドメインのホスト名を示します。

#### カatalog・サーバー

選択したドメインに属する 1 つ以上のカatalog・サーバーをリストします。前のステップで作成したカatalog・サーバーを追加できます。

#### 生成プログラム・クラス

**CredentialGenerator** インターフェースを実装するクラスの名前を指定します。このクラスを使用して、クライアントの資格情報が取得されます。このフィールドに値を指定すると、`client.properties` ファイルにある **credentialGeneratorClass** プロパティが、指定した値でオーバーライドされます。

#### 生成プログラム・プロパティ

**CredentialGenerator** 実装クラスのプロパティを指定します。このプロパティが、`setProperty(String)` メソッドを使用してオブジェクトに設定されます。 **credentialGeneratorProps** 値は、**credentialGeneratorClass** プロパティの値が非ヌルの場合にのみ使用されます。このフィールドに値を指定すると、`client.properties` ファイルにある **credentialGeneratorProps** プロパティが、指定した値でオーバーライドされます。

### eXtreme Scale クライアント・プロパティ・パス

前のステップでセキュリティー・プロパティを含める編集をしたクライアント・プロパティ・ファイルへのパスを指定します。例えば、`c:\%ObjectGridProperties%sampleclient.properties` ファイルを示します。コンソールがセキュア接続を使用しないようにする場合は、このフィールドの値を削除できます。パスを設定した後、コンソールは非セキュアな接続を使用します。

- 3) 「OK」をクリックします。
- 4) ドメインがナビゲーション・ツリーに追加されていることを確認します。

既存のカタログ・サービス・ドメインに関する情報を表示するには、「設定」 > 「eXtreme Scale ドメイン」ページのナビゲーション・ツリーの中で、カタログ・サービス・ドメインの名前をクリックします。

4. 接続状況を表示します。「**現行ドメイン**」フィールドは、Web コンソールの中で情報を表示するために現在使用されているカタログ・サービス・ドメインの名前を示します。接続状況が、カタログ・サービス・ドメインの名前の隣に表示されます。

## Web コンソールでの統計の表示

統計やその他のパフォーマンス情報を Web コンソールでモニターできます。

### 始める前に

Web コンソールで統計を表示するには、次のタスクを完了する必要があります。

1. Web コンソール・サーバーを開始します。詳しくは、507 ページの『Web コンソールの開始とログオン』を参照してください。
2. カタログ・サーバーを Web コンソール・サーバーに接続します。詳しくは、509 ページの『Web コンソールのカタログ・サーバーへの接続』を参照してください。
3. カタログ・サービス・ドメインの管理下にあるサーバー内で、アクティブなデータ・グリッドおよびアプリケーションを実行します。

### このタスクについて

データ・グリッドを作成し、データ・グリッドを使用するようにアプリケーションを構成したら、統計が使用可能になるまで少し時間を置きます。例えば、動的キャッシュでは、動的キャッシュを実行している WebSphere Application Server が動的キャッシュ・データ・グリッドに接続されるまで、統計は使用可能になりません。一般的には、統計における変化を見るために、主要な構成変更の後で最大で 1 分待ちます。

**ヒント:** グラフ内の任意のデータ・ポイントに関するより具体的な情報を表示するには、そのデータ・ポイントの上にマウス・ポインターを移動させてください。

### 手順

- 現在のサーバー統計を表示するには、「モニター」 > 「サーバー概要」をクリックします。

- すべてのデータ・グリッドのパフォーマンスを表示するには、「**モニター**」 > 「**データ・グリッド・ドメインの概要**」をクリックします。
- 個々のデータ・グリッドを表示するには、「**モニター**」 > 「**データ・グリッドの概説**」 > 「**data\_grid\_name**」をクリックします。このページでは、キャッシュ・エントリー数、平均トランザクション時間、および平均スループットを含むサマリーが表示されます。
- 特定のデータ・グリッドに関するより詳細な情報を表示するには、「**モニター**」 > 「**データ・グリッドの詳細**」をクリックします。ツリーにはご使用の構成におけるすべてのデータ・グリッドが表示されています。特定のデータ・グリッドをドリルダウンして、そのデータ・グリッドの一部であるマップを表示します。データ・グリッド名またはマップをクリックすることで、詳細情報を確認できます。
- カスタム・レポートに含める統計を選択するには、「**モニター**」 > 「**カスタム・レポート**」をクリックします。

このビューを使用して、各種統計の詳細データ・グラフを構成します。ツリーを使用して、使用可能なデータ・グリッドとサーバーおよびその関連統計を探索します。グラフ化できるデータを参照するノード上でクリックするか **Enter** を押すと、メニューが開きます。統計を含んだ新規グラフを作成するか、互換性のある統計で統計を既存のグラフに追加します。詳しくは、518 ページの『カスタム・レポートによるモニター』を参照してください。

## Web コンソール統計

Web コンソールで使用しているビューに応じて、構成に関するさまざまな統計を表示できます。これらの統計値には、使用メモリ、上位使用データ・グリッド、およびキャッシュ・エントリー数などがあります。

- 『データ・グリッド・ドメインの概要』
- 513 ページの『データ・グリッドの概説』
- 513 ページの『データ・グリッドの詳細』
- 514 ページの『サーバーの概要』
- 514 ページの『カスタム・レポート: カタログ・サービス・ドメイン統計』
  - 515 ページの『カスタム・レポート: コンテナ・サーバー統計』
  - 516 ページの『カスタム・レポート: データ・グリッド統計』
  - 516 ページの『カスタム・レポート: マップ統計』

## データ・グリッド・ドメインの概要

データ・グリッド・ドメインの概要についての統計は、「**モニター**」 > 「**データ・グリッド・ドメインの概要**」ページに表示されます。データ・グリッド・ドメインの詳細情報を表示するには、次のタブの 1 つをクリックします。

### 「使用容量」タブ

「**現行データ・グリッド使用容量の分布**」チャートには、「**合計プール**」のピクチャー、および「**最大使用容量コンシューマー**」が表示されます。上位 25 のデータ・グリッドのみが表示されます。「**一定時間の使用容量**」チャートに、データ・グリッドが消費したバイト数が表示されます。



### 「平均スループット」タブ

「トランザクション平均時間 (ミリ秒) によるトップ 5 のアクティブなデータ・グリッド」チャートには、平均トランザクション時間で編成された、トップ 5 のデータ・キャッシュのリストが含まれています。「一定時間の平均スループット」チャートは、最新の週、日、時間内の、平均、最大、および最小のスループットを表示します。

### 「トランザクション平均時間」タブ

「最も遅い 5 データ・グリッド」チャートは、最も遅いデータ・グリッドに関するデータを表示します。「一定時間のトランザクション平均時間」チャートは、最新の週、日、時間内の、平均、最大、および最小のトランザクション時間を表示します。

## データ・グリッドの概説

個々のデータ・グリッドの統計を表示するには、「モニター」 > 「データ・グリッドの概説」 > `data_grid_name` をクリックします。

### 過去 30 秒間についての現在の要約

選択されたデータ・グリッドの現在のキャッシュ・エントリー数、平均トランザクション時間、平均スループット、およびキャッシュのヒット率を表示します。

### 「使用容量」タブ

「過去 30 秒間についての現在の要約」チャートは、指定された時刻範囲のキャッシュ・エントリー数と使用容量 (バイト) を表示します。

### 「キャッシュの使用」タブ

「キャッシュの使用」チャートは、キャッシュに対する成功した照会の数の視覚化に便利で、指定された時刻範囲のキャッシュの試行数、キャッシュのヒット数、およびキャッシュのヒット率を表示します。

### 「平均スループット」タブ

「平均スループット対トランザクション平均時間」チャートは、指定された時刻範囲のトランザクション時間とスループットを表示します。

## データ・グリッドの詳細

データ・グリッド統計は、「モニター」 > 「データ・グリッドの詳細」ページに表示されます。選択されたグリッドのデータと、そのグリッド内のマップを見ることができます。

### 過去 30 秒間についての現在の要約

選択されたデータ・グリッドの現在の使用容量、キャッシュ・エントリー数、平均スループット、およびトランザクション平均時間を表示します。

### 現行 eXtreme Scale オブジェクト・グリッド・マップ使用容量の分布

合計プール (ゾーン別の容量および各ゾーンの合計容量を含む) を表示します。上位 25 の ObjectGrid マップのみが表示されます。また、マップごとの最大使用容量コンシューマーも表示できます。

### 現行ゾーン使用容量の分布

合計プールを表示します。これには、選択されたデータ・グリッドのゾーン

内の合計プールおよび使用容量コンシューマー上位が含まれます。また、ゾーンごとの最大使用容量コンシューマーも表示できます。

#### マップ統計:

##### 過去 30 秒間についての現在の要約

選択されたマップの現在の使用容量、キャッシュ・エントリー数、平均スループット、およびトランザクション平均時間を表示します。

##### 現行区画使用容量の分布

区画を表示します。区画には、合計プールおよび使用容量コンシューマー上位が含まれます。上位 25 の区画のみが表示されます。また、区画ごとの最大使用容量コンシューマーも表示できます。

#### サーバーの概要

サーバー統計は、「モニター」 > 「サーバーの概要」ページに表示されます。

##### 現行サーバー使用メモリーの分布

このチャートは、2 つのビューで構成されます。「合計プール」は、サーバー実行時における現在の使用 (実) メモリー量を表示します。「最大使用メモリー・コンシューマー」は、サーバーごとの使用メモリーを表示します。ただし、メモリー使用量が多い順に上位 25 のサーバーのみが表示されます。

##### 一定時間の合計メモリー

サーバー実行時における実メモリー使用量を表示します。

##### 一定時間の使用メモリー

サーバー実行時における使用メモリー量を表示します。

#### カスタム・レポート: カタログ・サービス・ドメイン統計

カスタム・レポートを作成することで、カタログ・サービス・ドメイン統計を表示できます。「モニター」 > 「カスタム・レポート」をクリックします。

##### 平均トランザクション時間 (ミリ秒)

このドメインでトランザクションを完了するために必要な平均時間を表示します。

##### 平均トランザクション・スループット (トランザクション/秒)

このドメイン内の 1 秒当たりのトランザクションの平均数を表示します。

##### 最大トランザクション時間 (ミリ秒)

このドメイン内で最も時間がかかった トランザクションで費やした時間を表示します。

##### 最小トランザクション時間 (ミリ秒)

このドメイン内で最も時間がかからなかった トランザクションで費やした時間を表示します。

##### 合計トランザクション時間 (ミリ秒)

このドメインでトランザクションに費やした、ドメインの初期設定時からの合計時間を表示します。

## カスタム・レポート: コンテナ・サーバー統計

カスタム・レポートを作成することで、コンテナ・サーバー統計を表示できます。「モニター」 > 「カスタム・レポート」をクリックします。

### 平均トランザクション時間 (ミリ秒)

このカタログ・サーバーでトランザクションを完了するために必要な平均時間を表示します。

### 平均トランザクション・スループット (トランザクション/秒)

このカタログ・サーバーの 1 秒当たりのトランザクションの平均数を表示します。

### 最大トランザクション時間 (ミリ秒)

このカタログ・サーバーで最も時間がかかった トランザクションで費やした時間を表示します。

### 最小トランザクション時間 (ミリ秒)

このカタログ・サーバーで最も時間がかからなかった トランザクションで費やした時間を表示します。

### 合計トランザクション時間 (ミリ秒)

このカタログ・サーバーでトランザクションに費やした、このカタログ・サーバーの初期設定時からの合計時間を表示します。

### キャッシュ内の合計エントリー

このカタログ・サーバーで監視されるグリッド内にキャッシュされたオブジェクトの現在の数を表示します。

### ヒット・レート (パーセンテージ)

選択したデータ・グリッドのヒット・レート (ヒット率) を表示します。高ヒット・レートが望ましい状態です。ヒット・レートは、永続ストアへのアクセスの回避にグリッドがどのくらい役立っているかを示します。

### 使用バイト

このマップによるメモリー消費量を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

### 最小使用バイト

このカタログ・サービスおよびそのマップによるメモリー消費量の最低点を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

### 最大使用バイト

このカタログ・サービスおよびそのマップによるメモリー消費量の最高点を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

### 合計ヒット数

要求データがマップ内で検出され、永続ストアにアクセスする必要がなかった回数の合計を表示します。

### 合計 GET 数

データを取得するためにマップが永続ストアにアクセスする必要があった回数の合計を表示します。

#### 空きヒープ (MB)

カタログ・サーバーによって使用されている JVM で使用可能なヒープの実際の容量を表示します。

#### 合計ヒープ

このカタログ・サーバーによって使用されている JVM で使用可能なヒープの容量を表示します。

#### 使用可能なプロセッサの数

このカタログ・サービスおよびそのマップで使用可能なプロセッサの数を表示します。最高の安定度を実現するために、60% のプロセッサ負荷でサーバー、さらに 60% のヒープ負荷で JVM ヒープを稼働してください。スパイクでプロセッサ使用量を 80% から 90% の間に引き上げることができますが、通常はこのレベルより高いレベルでサーバーを稼働しないようにしてください。

#### 最大ヒープ・サイズ (MB)

このカタログ・サーバーによって使用されている JVM で使用可能なヒープの最大容量を表示します。

#### 使用メモリー

このカタログ・サーバーによって使用されている JVM の使用メモリーを表示します。

### カスタム・レポート: データ・グリッド統計

カスタム・レポートを作成することで、データ・グリッド統計を表示できます。「モニター」 > 「カスタム・レポート」をクリックします。

#### 平均トランザクション時間 (ミリ秒)

このグリッドに関係するトランザクションを完了するために必要な平均時間を表示します。

#### 平均トランザクション・スループット (トランザクション/秒)

このグリッドが完了した 1 秒当たりのトランザクションの平均数を表示します。

#### 最大トランザクション時間 (ミリ秒)

このグリッドが完了した中で最も時間がかかった トランザクションで費やした時間を表示します。

#### 最小トランザクション時間 (ミリ秒)

このグリッドが完了した中で最も時間がかからなかった トランザクションで費やした時間を表示します。

#### 合計トランザクション時間 (ミリ秒)

このグリッドのトランザクション処理の合計時間を表示します。

### カスタム・レポート: マップ統計

カスタム・レポートを作成することで、マップ統計を表示できます。「モニター」 > 「カスタム・レポート」をクリックします。

#### キャッシュ内の合計エントリー

このマップでキャッシュされているオブジェクトの現在の数を表示します。

### ヒット・レート (パーセンテージ)

選択したマップのヒット・レート (ヒット率) を表示します。高ヒット・レートが望ましい状態です。ヒット・レートは、永続ストアへのアクセスの回避にマップがどのくらい役立っているかを示します。

### 使用バイト

このマップによるメモリー消費量を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

### 最小使用バイト

このマップの最小消費量 (バイト単位) を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

### 最大使用バイト

このマップの最大消費量 (バイト単位) を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

### 合計ヒット数

要求データがマップ内で検出され、永続ストアにアクセスする必要がなかった回数の合計を表示します。

### 合計 GET 数

データを取得するためにマップが永続ストアにアクセスする必要があった回数の合計を表示します。

### 空きヒープ (MB)

カタログ・サーバーによって使用されている JVM 内の、このマップで使用可能なヒープの現在の容量を表示します。

### 合計ヒープ (MB)

カタログ・サーバーによって使用されている JVM 内の、このマップで使用可能なヒープの合計容量を表示します。最高の安定度を実現するために、60% のプロセッサ負荷でサーバー、さらに 60% のヒープ負荷で JVM ヒープを稼働してください。スパイクでプロセッサ使用量を 80% から 90% の間に引き上げることができますが、通常はこのレベルより高いレベルでサーバーを稼働しないようにしてください。

### 使用可能なプロセッサの数

このマップで使用可能なプロセッサの数を表示します。最高の安定度を実現するために、60% のプロセッサ負荷でサーバー、さらに 60% のヒープ負荷で JVM ヒープを稼働してください。スパイクでプロセッサ使用量を 80% から 90% の間に引き上げることができますが、通常はこのレベルより高いレベルでサーバーを稼働しないようにしてください。

### 最大ヒープ・サイズ (MB)

カタログ・サーバーによって使用されている JVM 内の、このマップで使用可能なヒープの最大容量を表示します。

### 使用メモリー (MB)

このマップの使用メモリー容量を表示します。

## カスタム・レポートによるモニター


カスタム・レポートを作成して、環境のカタログ・サービス・ドメイン、データ・グリッド、およびコンテナ・サーバーについての統計が含まれたさまざまなチャートを保存することができます。カスタム・レポートを保存すると、後でロードして再表示することができます。

### 始める前に

Web コンソールで統計を表示するには、次のタスクを完了する必要があります。

1. Web コンソール・サーバーを開始します。詳しくは、507 ページの『Web コンソールの開始とログオン』を参照してください。
2. カタログ・サーバーを Web コンソール・サーバーに接続します。詳しくは、509 ページの『Web コンソールのカタログ・サーバーへの接続』を参照してください。
3. カタログ・サービス・ドメインの管理下にあるサーバー内で、アクティブなデータ・グリッドおよびアプリケーションを実行します。

### 手順

- カスタム・レポートを作成します。
  1. 「モニター」 > 「カスタム・レポート」をクリックします。定義した eXtreme Scale ドメインのリストが、ツリー形式でリストされます。各ドメインを展開すると、カスタム・レポートに追加できる使用可能な統計を表示できます。
  2. 追跡する統計が入ったチャートを追加します。使用可能な統計は、「チャート」アイコン (  ) によって示されます。追跡する統計の 1 つをクリックします。「新しいチャートに追加」または「既存のチャートに追加」を選択します。選択に応じて、選択された統計が新規チャート・タブまたは選択した既存のチャートに表示されます。チャート上に既にあるメトリックと新規のメトリックが同じ単位を使用する場合にのみ、既存のチャートにメトリックを追加できます。
- カスタム・レポートを保存します。カスタム・レポートを保存すると、作成したすべてのタブの中に統計が保存されます。レポートを保存するには、「保存」をクリックします。
- カスタム・レポートをロードします。「ロード」をクリックし、表示する、保存されたカスタム・レポートを選択します。

---

## CSV ファイルによるモニター

モニター・データを CSV ファイルに書き込むことができます。これらの CSV ファイルには、JVM、マップ、または ObjectGrid インスタンスに関する情報を含めることができます。

### このタスクについて

モニター・データを CSV ファイルに書き込むことで、個々のコンテナ・サーバーまたはカタログ・サーバーの履歴データをダウンロードして分析できます。CSV



ファイルを使用可能にするサーバー・プロパティが指定されたサーバーを開始すると、データの収集が始まります。その後は、この CSV ファイルをいつでもダウンロードでき、自由にファイルを使用できます。

## 手順

1. CSV ファイルの使用可能化に関連した次のプロパティを指定してサーバー・プロパティ・ファイルを更新します。

```
parameter=default value
jvmStatsLoggingEnabled=true
maxJVMStatsFiles=5
maxJVMStatsFileSize=100
jvmStatsFileName=jvmstats
jvmStatsWriteRate=10
```

```
mapStatsLoggingEnabled=true
maxMapStatsFiles=5
maxMapStatsFileSize=100
mapStatsFileName=mapstats
mapStatsWriteRate=10
```

```
ogStatsLoggingEnabled=true
maxOGStatsFiles=5
maxOGStatsFileSize=100
ogStatsFileName=ogstats
ogStatsWriteRate=10
```

これらのプロパティについては詳しくは、サーバー・プロパティ・ファイルを参照してください。

2. サーバーを再始動して、サーバー・プロパティ・ファイルの変更を反映します。
3. CSV ファイルをダウンロードします。CSV ファイルは `server_name/logs` ディレクトリーに書き込まれます。
4. データの処理に使用するプログラム (スプレッドシートなど) に CSV ファイルをインポートします。

## 次のタスク

CSV ファイルに含まれるデータの詳細については、『CSV ファイルの統計定義』を参照してください。

## CSV ファイルの統計定義

サーバーにダウンロード可能な CSV ファイルには、ヒストリカル・チャートを作成するために使用可能な統計やその他の情報が含まれています。

### Java 仮想マシン (JVM) の統計ログ

#### TimeStamp (列 1)

Java 仮想マシン (JVM) に対して取られた統計スナップショットの日時を示します。

#### ServerName (列 2)

JVM のサーバー名を指定します。

#### Hostname (列 3)

JVM のホスト名を指定します。

**DomainName (列 4)**

JVM が属しているカタログ・サービス・ドメインを指定します。

**FreeMemory (列 5)**

JVM で使用可能なバイト数を指定します。

**MaxMemory (列 6)**

JVM に割り振り可能な最大バイト数を指定します。

**TotalMemory (列 7)**

サーバー実行時における実メモリー使用量を表示します。

**AvailProcs (列 8)**

このカタログ・サービスおよびそのマップで使用可能なプロセッサの数を表示します。最高の安定度を実現するために、60% のプロセッサ負荷でサーバー、さらに 60% のヒープ負荷で JVM ヒープを稼働してください。スパイクでプロセッサ使用量を 80% から 90% の間に引き上げることができますが、通常はこのレベルより高いレベルでサーバーを稼働しないようにしてください。

**マップの統計ログ****TimeStamp (列 1)**

マップに対して取られた統計スナップショットの日時を指定します。

**MapName (列 2)**

マップの名前を指定します。

**PartitionId (列 3)**

区画 ID を指定します。

**OgName (列 4)**

このマップが属しているデータ・グリッドの名前を指定します。

**MapSetName (列 5)**

このマップが属しているマップ・セットの名前を指定します。

**HitRate (列 6)**

選択したマップのヒット・レート (ヒット率) を表示します。高ヒット・レートが望ましい状態です。ヒット・レートは、永続ストアへのアクセスの回避にデータ・グリッドがどのくらい役立っているかを示します。

**Count (列 7)**

サーバーの始動以降に収集されたデータ・サンプルの数を示します。例えば、100 の値は、その項目がサーバーの始動以降収集された 100 番目のサンプル項目であることを示します。

**TotalGetCount (列 8)**

データを取得するためにマップが永続ストアにアクセスする必要があった回数の合計を表示します。

**TotalHitCount (列 9)**

要求データがマップ内で検出され、永続ストアにアクセスする必要がなかった回数の合計を表示します。

**StartTime (列 10)**

最後のリセット呼び出しからカウンターが開始する時刻を示します。リセットは、サーバーが始動または再始動したときに行われます。

**LastCount (列 11)**

最後のデータ・サンプルが取られてから経過した時間を示します。

**LastTotalGetCount (列 12)**

現在のキャッシュからの取得操作の総数から、前の期間の取得操作の数を引いた数を示します。

**LastTotalHitCount (列 13)**

現在のキャッシュからのヒット総数から、前の期間のヒット数を引いた数を示します。

**UsedBytes (列 14)**

このマップによるメモリー消費量を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

**MinUsedBytes (列 15)**

このカタログ・サービスおよびそのマップによるメモリー消費量の最低点を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

**MaxUsedBytes (列 16)**

このカタログ・サービスおよびそのマップによるメモリー消費量の最高点を表示します。使用バイト統計は、単純なオブジェクトまたは COPY\_TO\_BYTES コピー・モードを使用している場合に限り正確です。

**LastUsedBytes (列 17)**

現在の UsedBytes 値から以前の統計収集期間の UsedBytes 値を引いた数を示します。

**SampleLen (列 18)**

データのサンプルが取られた期間の長さをミリ秒単位で示します。

**ObjectGrid 統計ログ****TimeStamp (列 1)**

データ・グリッドに対して取られた統計スナップショットの日時を指定します。

**OgName (列 2)**

データ・グリッドの名前を指定します。

**Count (列 3)**

サーバーの始動以降に収集されたデータ・サンプルの数を示します。例えば、100 の値は、その項目がサーバーの始動以降収集された 100 番目のサンプル項目であることを示します。

**PartitionId (列 4)**

区画 ID を指定します。

**Hostname (列 5)**

ホスト名を指定します。

**DomainName (列 6)**

このデータ・グリッドが属しているカタログ・サービス・ドメインを指定します。

**MaxTime (列 7)**

このサーバーで最も時間がかかった トランザクションで費やした時間を表示します。

**MinTime (列 8)**

このサーバーで最も時間がかからなかった トランザクションで費やした時間を表示します。

**MeanTime (列 9)**

トランザクションにかかった平均時間を示します。

**TotalTime (列 10)**

このサーバーでトランザクションに費やした、このサーバーの初期設定時からの合計時間を表示します。

**AvgTransTime (列 11)**

このサーバーでトランザクションを完了するために必要な平均時間を表示します。

**AvgThroughPut (列 12)**

このサーバーの 1 秒当たりのトランザクションの平均数を表示します。

**SumOfSquares (列 13)**

トランザクション時間の二乗値の合計を示します。この値は、ある時点での平均からの偏差を測定します。

**SampleLen (列 14)**

データのサンプルが取られた期間の長さをミリ秒単位で示します。

**LastDataSample (列 15)**

最後のサンプルが取られてから経過した時間を示します。

**LastTotalTime (列 16)**

現在のデータ・サンプルの合計時間から前の合計時間を引いた値を示します。

**StartTime (列 17)**

データの最後のリセット以降に統計の収集が開始された時刻を示します。サーバーが再始動すると、データはリセットされます。

---

## 統計の使用可能化

WebSphere eXtreme Scale は、内部統計モデルを使用して、データの追跡およびフィルター処理を行います。このモデルは、すべてのデータ・ビューで統計のスナップショットを収集するために使用される基礎となる構造です。統計モジュールから情報を取得するには、いくつかの方法を使用できます。

### このタスクについて

統計を使用可能にできるすべてのモジュールのリストについては、StatsSpec クラスを参照してください。

## 手順

- サーバー・プロパティ・ファイルを使用して、統計を使用可能にします。コンテナ・サーバーのサーバー・プロパティ・ファイルの中の **statsSpec** プロパティを使用して、サーバーの始動時に統計仕様を設定できます。詳しくは、サーバー・プロパティ・ファイルを参照してください。
- **xscmd** ユーティリティーを使用して統計を使用可能にします。 **-c setStatsSpec** コマンドを使用して、実行時に統計仕様を設定できます。詳しくは、473 ページの『**xscmd** ユーティリティーによる管理』を参照してください。
- **StatsSpec** インターフェースを使用してプログラマチックに統計を使用可能にします。詳しくは、525 ページの『統計 API によるモニター』を参照してください。
- **DynamicServerMBean** の **setStatsSpec** オペレーションを使用して、**JMX** で統計を使用可能にします。詳しくは、インターフェース **DynamicServerMBean** を参照してください。

## 例

次に、プロパティ・ファイル、**xscmd** ユーティリティー、または **StatsSpec** インターフェースを使用して指定することがある **statsSpec** スtring の例をいくつか挙げます。

すべてのモジュールのすべての統計を使用可能にします。

```
all=enabled
```

すべてのモジュールのすべての統計を使用不可にします。

```
all=disabled
```

**OGStatsModule** 内のすべての統計について、統計を使用可能にします。

```
og.all=enabled
```

**OGStatsModule** 内および **MapStatsModule** 内のすべての統計について、統計を使用可能にします。

```
og.all=enabled;map.all=enabled
```

「マップの使用バイト数」統計についてのみ統計を使用可能にし、それ以外はすべて使用不可にします。

```
all=disabled;map.usedbytes=enabled
```

## 統計モジュール

**WebSphere eXtreme Scale** は、内部統計モデルを使用して、データの追跡およびフィルター処理を行います。このモデルは、すべてのデータ・ビューで統計のスナップショットを収集するために使用される基礎となる構造です。

### 概要

**WebSphere eXtreme Scale** での統計は、**StatsModules** コンポーネント内で追跡され、収容されます。統計モデルには、以下のいくつかのタイプの統計モジュールが存在します。

### **OGStatsModule**

トランザクション応答時間など、ObjectGrid インスタンスの統計を提供します。

### **MapStatsModule**

エントリー数やヒット率など、単一マップの統計を提供します。

### **QueryStatsModule**

計画作成や実行時間など、照会の統計を提供します。

### **AgentStatsModule**

シリアライズ時間や実行時間など、DataGrid API エージェントの統計を提供します。

### **HashIndexStatsModule**

HashIndex 照会および保守の実行時間の統計を提供します。

### **SessionStatsModule**

HTTP セッション・マネージャー・プラグインの統計を提供します。

統計モジュールについて詳しくは、API 資料の `com.ibm.websphere.objectgrid.stats` パッケージを参照してください。

## **ローカル環境での統計**

モデルは、前のリストで説明したすべての StatsModule タイプから構成される n 進ツリー (すべてのノードについて同じ次数を持つツリー構造) に似た編成になっています。この編成構造のため、ツリー内のすべてのノードは、StatsFact インターフェースで表現されます。StatsFact インターフェースは、集約の目的で個別のモジュールまたはモジュールのグループを表わすことができます。例えば、ツリー内のいくつかのリーフ・ノードが特定の MapStatsModule オブジェクトを表わす場合、これらのノードの親 StatsFact ノードには、すべての子モジュールについて集約された統計が含まれます。StatsFact オブジェクトのフェッチ後には、インターフェースを使用して対応する StatsModule を取得することができます。

ツリー・マップに非常によく似ており、対応するパスまたはキーを使用して特定の StatsFact を取得することができます。パスは、要求されたファクトへのパスに沿ったすべてのノードから構成される String[] 値です。例えば、MapA と MapB という 2 つのマップを含む ObjectGridA という名前の ObjectGrid を作成したとします。MapA の StatsModule へのパスは、[ObjectGridA, MapA] のようになります。両方のマップの集約統計へのパスは、[ObjectGridA] となります。

## **分散環境での統計**

分散環境では、統計モジュールは異なるパスを使用して取得されます。サーバーには複数の区画を入れることができるため、統計ツリーは、各モジュールが属する区画を追跡する必要があります。結果として、特定の StatsFact オブジェクトをルックアップするためのパスは異なります。前の例を使用しますが、マップが区画 1 に存在するという点を付け加えると、MapA の StatsFact オブジェクトを取得するためのパスは [1,ObjectGridA, MapA] となります。



## 統計 API によるモニター

統計 API は、内部統計ツリーに直接接続するインターフェースです。統計はデフォルトでは使用不可になっていますが、StatsSpec インターフェースを設定することで使用可能にすることができます。StatsSpec インターフェースは、WebSphere eXtreme Scale がどのように統計をモニターするかを定義します。

### このタスクについて

ローカルの StatsAccessor API を使用して、実行中のコードと同じ Java 仮想マシン (JVM) にある ObjectGrid インスタンス上のデータおよびアクセス統計を照会することができます。個々のインターフェースについて詳しくは、API 資料を参照してください。次の手順で、内部統計ツリーのモニターを使用可能にします。

### 手順

1. StatsAccessor オブジェクトを検索します。StatsAccessor インターフェースは singleton パターンに従います。したがって、クラス・ローダーに関連する問題を別にすれば、JVM ごとに 1 つの StatsAccessor インスタンスが存在するはずで、このクラスはすべてのローカル統計操作のメイン・インターフェースとして機能します。以下のコードは、accessor クラスの検索方法の例です。この操作は、他のすべての ObjectGrid 呼び出しより前に呼び出します。

```
public class LocalClient
{
    public static void main(String[] args) {
        // retrieve a handle to the StatsAccessor
        StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();
    }
}
```

2. データ・グリッド StatsSpec インターフェースを設定します。すべての統計を ObjectGrid レベルでのみ収集するように、この JVM を設定します。トランザクションを開始する前に、必要と思われるすべての統計をアプリケーションが使用可能にするようにする必要があります。次の例は、static 定数フィールドと spec スtring の両方を使用して StatsSpec インターフェースを設定するものです。static 定数フィールドは既に仕様が定義されているため、このフィールドを使用する方が簡単です。ただし、spec スtring を使用すれば、必要な統計のどんな組み合わせでも使用可能にすることができます。

```
public static void main(String[] args) {
    // retrieve a handle to the StatsAccessor
    StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

    // Set the spec via the static field
    StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
    accessor.setStatsSpec(spec);

    // Set the spec via the spec String
    StatsSpec spec = new StatsSpec("og.all=enabled");
    accessor.setStatsSpec(spec);
}
```

3. トランザクションをグリッドに送信して、モニター用のデータが収集されるようにします。統計用に有効なデータを収集するには、トランザクションをデータ・グリッドに送る必要があります。次のコード抜粋は、ObjectGridA 内の MapA にレコードを挿入するものです。統計は、ObjectGrid レベルであるため、ObjectGrid 内のマップはいずれも同じ結果を示します。

```
public static void main(String[] args) {  
  
    // retrieve a handle to the StatsAccessor  
    StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();  
  
    // Set the spec via the static field  
    StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);  
    accessor.setStatsSpec(spec);  
  
    ObjectGridManager manager =  
    ObjectGridManagerFactory.getObjectGridManager();  
    ObjectGrid grid = manager.getObjectGrid("ObjectGridA");  
    Session session = grid.getSession();  
    Map map = session.getMap("MapA");  
  
    // Drive insert  
    session.begin();  
    map.insert("SomeKey", "SomeValue");  
    session.commit();  
}
```

4. StatsAccessor API を使用して StatsFact を照会します。すべての統計パスは StatsFact インターフェースに関連付けられます。StatsFact インターフェースは、StatsModule オブジェクトを編成して組み込むために使用される汎用プレースホルダーです。実際の統計モジュールにアクセスするためには、前もって StatsFact オブジェクトを検索する必要があります。

```
public static void main(String[] args)  
{  
  
    // retrieve a handle to the StatsAccessor  
    StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();  
  
    // Set the spec via the static field  
    StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);  
    accessor.setStatsSpec(spec);  
  
    ObjectGridManager manager =  
    ObjectGridManagerFactory.getObjectGridManager();  
    ObjectGrid grid = manager.getObjectGrid("ObjectGridA");  
    Session session = grid.getSession();  
    Map map = session.getMap("MapA");  
  
    // Drive insert  
    session.begin();  
    map.insert("SomeKey", "SomeValue");  
    session.commit();  
  
    // Retrieve StatsFact  
  
    StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},  
    StatsModule.MODULE_TYPE_OBJECT_GRID);  
  
}
```

5. StatsModule オブジェクトと対話します。StatsModule オブジェクトは StatsFact インターフェース内に含まれています。StatsFact インターフェースを使用してモジュールへの参照を取得できます。StatsFact インターフェースは汎用インタ

ーフエースであるため、戻されたモジュールを予期された `StatsModule` タイプにキャストする必要があります。このタスクは `eXtreme Scale` の統計を収集するため、戻された `StatsModule` オブジェクトは `OGStatsModule` タイプにキャストされます。モジュールがキャストされたならば、使用可能なすべての統計にアクセスすることができます。

```
public static void main(String[] args) {  
  
    // retrieve a handle to the StatsAccessor  
    StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();  
  
    // Set the spec via the static field  
    StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);  
    accessor.setStatsSpec(spec);  
  
    ObjectGridManager manager =  
    ObjectGridmanagerFactory.getObjectGridManager();  
    ObjectGrid grid = manager.getObjectGrid("ObjectGridA");  
    Session session = grid.getSession();  
    Map map = session.getMap("MapA");  
  
    // Drive insert  
    session.begin();  
    map.insert("SomeKey", "SomeValue");  
    session.commit();  
  
    // Retrieve StatsFact  
    StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},  
    StatsModule.MODULE_TYPE_OBJECT_GRID);  
  
    // Retrieve module and time  
    OGStatsModule module = (OGStatsModule)fact.getStatsModule();  
    ActiveTimeStatistic timeStat =  
    module.getTransactionTime("Default", true);  
    double time = timeStat.getMeanTime();  
  
}
```

---

## xscmd ユーティリティによるモニター

**xscmd** ユーティリティは、完全にサポートされたモニターおよび管理のツールとして、**xsadmin** サンプル・ユーティリティに取って代わります。**xscmd** ユーティリティを使用すれば、WebSphere eXtreme Scale トポロジに関するテキスト情報を表示できます。

### 始める前に

- **xscmd** ユーティリティを使用して結果を表示するには、データ・グリッド・トポロジを作成しておく必要があります。カタログ・サーバーおよびコンテナ・サーバーは、始動済みでなければなりません。詳しくは、451 ページの『スタンドアロン・サーバーの始動と停止』を参照してください。
- **xscmd** ユーティリティの開始について詳しくは、473 ページの『**xscmd** ユーティリティによる管理』を参照してください。

### このタスクについて

**xscmd** ユーティリティを使用して、マップの内容など、データ・グリッドの現在のレイアウトおよび特定の状態を表示できます。この例では、このタスクのデータ・グリッドのレイアウトは、`MapSetA` マップ・セットに属する 1 つの `MapA` マ

マップを持つ 1 つの *ObjectGridA* データ・グリッドから成ります。この例では、データ・グリッドにすべてのアクティブ・コンテナを表示する方法と、*MapA* マップのマップ・サイズに関するフィルタリングされたメトリックを印刷する方法を説明します。使用できるコマンド・オプションをすべて知りたい場合は、引数なしか、または **-help** オプションを付けて **xscmd** ユーティリティを実行してください。

## 手順

**xscmd** ユーティリティを使用して、環境をモニターします。

- すべてのサーバーの統計を使用可能にするには、以下のコマンドを実行します。

```
- UNIX ./xscmd.sh -c setStatsSpec -spec ALL=enabled -g ObjectGridA
```

```
- Windows xscmd.bat -c setStatsSpec -spec ALL=enabled -g ObjectGridA
```

- データ・グリッドのすべてのオンライン・コンテナ・サーバーを表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -c showPlacement -g ObjectGridA -ms MapSetA
```

```
- Windows xscmd.bat -c showPlacement -g ObjectGridA -ms MapSetA
```

すべてのコンテナ情報が表示されます。

**重要:** Transport Layer Security/Secure Sockets Layer (TLS/SSL) が使用可能であるときにこの情報を取得するには、JMX サービス・ポートを設定してカタログ・サーバーおよびコンテナ・サーバーを始動する必要があります。JMX サービス・ポートを設定するには、**startOgServer** スクリプトで **-JMXServicePort** オプションを使用するか、**ServerProperties** インターフェイスで **setJMXServicePort** メソッドを呼び出すことができます。

- *ObjectGridA* データ・グリッドのマップについての情報を表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA
```

```
- Windows xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA
```

- カタログ・サービスに接続して、カタログ・サービス・ドメイン全体の *MapA* マップに関する情報を表示するには、次のコマンドを実行します。

```
- UNIX ./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA -m MapA  
-cep CatalogMachine:6645
```

```
- Windows xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA -m MapA  
-cep CatalogMachine:6645
```

**xscmd** ユーティリティは、カタログ・サーバーを実行している **MBean** サーバーに接続します。単一のカタログ・サーバーに接続することによって、カタログ・サービス・ドメイン全体についての情報を取得できます。カタログ・サーバーは、スタンドアロン・プロセスまたは **WebSphere Application Server** プロセスとして実行できます。あるいは、カスタム・アプリケーション・プロセス内に組み込むこともできます。**-cep** オプションを使用して、カタログ・サービスのホスト名とポートを指定します。**-cep** オプションにカタログ・サーバーのリストを合

める場合、カタログ・サーバーは、同じカタログ・サービス・ドメイン内になければなりません。一度に 1 つのカタログ・サービス・ドメインの統計を取得できません。

- 構成内の構成されている配置とランタイムの配置を表示するには、次のいずれかのコマンドを実行します。

```
- xscmd -c placementServiceStatus
- xscmd -c placementServiceStatus -g ObjectGridA -ms MapSetA
- xscmd -c placementServiceStatus -ms MapSetA
- xscmd -c placementServiceStatus -g ObjectGridA
```

配置情報については、全体の構成、1 つのデータ・グリッド、1 つのマップ・セット、またはデータ・グリッドとマップ・セットの組み合わせを表示するそれぞれのコマンドを使用できます。

---

## WebSphere Application Server PMI によるモニター

WebSphere eXtreme Scale は、WebSphere Application Server または WebSphere Extended Deployment アプリケーション・サーバーで実行されているとき、Performance Monitoring Infrastructure (PMI) をサポートします。PMI は、ランタイム・アプリケーションでパフォーマンス・データを収集し、パフォーマンス・データをモニターするための外部アプリケーションをサポートするインターフェースを提供します。管理コンソールまたは wsadmin ツールを使用して、モニター・データにアクセスすることができます。

### 始める前に

WebSphere eXtreme Scale を WebSphere Application Server と組み合わせて使用しているとき、PMI を使用してご使用の環境をモニターすることができます。

### このタスクについて

WebSphere eXtreme Scale は、WebSphere Application Server のカスタム PMI 機能を使用して、独自の PMI 装備を追加します。この方法で、管理コンソールまたは wsadmin ツールの Java Management Extensions (JMX) インターフェースを使用して、WebSphere eXtreme Scale PMI を使用可能および使用不可にすることができます。さらに、標準 PMI、および Tivoli Performance Viewer を含むモニター・ツールによって使用される JMX インターフェースを使用して WebSphere eXtreme Scale 統計にアクセスすることができます。

### 手順

1. eXtreme Scale PMI を使用可能にします。PMI 統計を表示するには、PMI を使用可能にする必要があります。詳しくは、530 ページの『PMI の使用可能化』を参照してください。
2. eXtreme Scale PMI 統計を取得します。Tivoli Performance Viewer を使用して、eXtreme Scale アプリケーションのパフォーマンスを表示します。詳しくは、532 ページの『PMI 統計の取得』を参照してください。

## 次のタスク

wsadmin ツールについて詳しくは、498 ページの『wsadmin ツールを使用した Managed Beans (MBeans) へのアクセス』を参照してください。

## PMI の使用可能化

WebSphere Application Server Performance Monitoring Infrastructure (PMI) を使用して、任意のレベルで統計を使用可能または使用不可にすることができます。例えば、特定のマップのマップ・ヒット率統計を使用可能にするが、エントリー数統計またはローダー・バッチ更新時間統計は使用可能にしないことを選択できます。管理コンソール内またはスクリプトを使用して PMI を使用可能にすることができます。

### 始める前に

アプリケーション・サーバーを始動し、eXtreme Scale 対応アプリケーションがインストールされている必要があります。また、スクリプトを使用して PMI を使用可能にするには、wsadmin ツールにログインして使用できなければなりません。

wsadmin ツールについて詳しくは、WebSphere Application Server インフォメーション・センターの wsadmin ツールのトピックを参照してください。

### このタスクについて

WebSphere Application Server PMI を使用して、任意のレベルで統計を使用可能または使用不可にできる細かいメカニズムを提供します。例えば、特定のマップのマップ・ヒット率統計を使用可能にするが、エントリー数統計またはローダー・バッチ更新時間統計は使用可能にしないことを選択できます。このセクションでは、管理コンソールおよび wsadmin スクリプトを使用して ObjectGrid PMI を使用可能にする方法を示します。

### 手順

- 管理コンソールで PMI を使用可能にします。

1. 管理コンソールで、「モニターおよびチューニング」 > 「Performance Monitoring Infrastructure」 > 「server\_name」をクリックします。
2. 「Performance Monitoring Infrastructure (PMI) を使用可能にする」が選択されていることを確認します。この設定は、デフォルトで使用可能になっています。この設定が使用可能になっていない場合は、チェック・ボックスを選択して、サーバーを再始動します。
3. 「カスタム」をクリックします。構成ツリーで、ObjectGrid および ObjectGrid マップ・モジュールを選択します。各モジュールの統計を使用可能にします。

ObjectGrid 統計のトランザクション・タイプ・カテゴリーが実行時に作成されます。「Runtime」タブには、ObjectGrid と Map 統計のサブカテゴリーのみを表示できます。

- スクリプトを使用して PMI を使用可能にします。

1. コマンド行プロンプトを開きます。was\_root/bin ディレクトリーに移動します。wsadmin と入力して、wsadmin コマンド行ツールを開始します。
2. eXtreme Scale PMI ランタイム構成を変更します。以下のコマンドを使用して、サーバーに対して PMI が使用可能になっていることを確認します。



```
wsadmin>set s1 [$AdminConfig getid /Cell:CELL_NAME/Node:NODE_NAME/  
Server:APPLICATION_SERVER_NAME/]  
wsadmin>set pmi [$AdminConfig list PMIService $s1]  
wsadmin>$AdminConfig show $pmi.
```

PMI が使用可能になっていない場合は、以下のコマンドを実行して、PMI を使用可能にします。

```
wsadmin>$AdminConfig modify $pmi {{enable true}}  
wsadmin>$AdminConfig save
```

PMI を使用可能にする必要がある場合は、サーバーを再起動します。

3. 以下のコマンドを使用して、統計セットをカスタム・セットに変更するための変数を設定します。

```
wsadmin>set perfName [$AdminControl completeObjectName type=Perf,  
process=APPLICATION_SERVER_NAME,*]  
wsadmin>set perfOName [$AdminControl makeObjectName $perfName]  
wsadmin>set params [java::new {java.lang.Object[]} 1]  
wsadmin>$params set 0 [java::new java.lang.String custom]  
wsadmin>set sigs [java::new {java.lang.String[]} 1]  
wsadmin>$sigs set 0 java.lang.String
```

4. 以下のコマンドを使用して、統計セットをカスタムに設定します。

```
wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
```

5. 以下のコマンドを使用して、objectGridModule PMI 統計を使用可能にするための変数を設定します。

```
wsadmin>set params [java::new {java.lang.Object[]} 2]  
wsadmin>$params set 0 [java::new java.lang.String objectGridModule=1]  
wsadmin>$params set 1 [java::new java.lang.Boolean false]  
wsadmin>set sigs [java::new {java.lang.String[]} 2]  
wsadmin>$sigs set 0 java.lang.String  
wsadmin>$sigs set 1 java.lang.Boolean
```

6. 以下のコマンドを使用して、統計ストリングを設定します。

```
wsadmin>set params2 [java::new {java.lang.Object[]} 2]  
wsadmin>$params2 set 0 [java::new java.lang.String mapModule=*]  
wsadmin>$params2 set 1 [java::new java.lang.Boolean false]  
wsadmin>set sigs2 [java::new {java.lang.String[]} 2]  
wsadmin>$sigs2 set 0 java.lang.String  
wsadmin>$sigs2 set 1 java.lang.Boolean
```

7. 以下のコマンドを使用して、統計ストリングを設定します。

```
wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
```

これらのステップにより、eXtreme Scale ランタイム PMI は使用可能になりますが、PMI 構成は変更されません。アプリケーション・サーバーを再起動すると、メイン PMI の使用可能化を除いて、PMI 設定は失われます。

## 例

以下のステップを実行して、サンプル・アプリケーションの PMI 統計を使用可能にすることができます。

1. `http://host:port/ObjectGridSample` Web アドレスを使用してアプリケーションを立ち上げます。ここで、`host` および `port` は、サンプルをインストールするサーバーのホスト名および HTTP ポート番号です。

2. サンプル・アプリケーションで `ObjectGridCreationServlet` をクリックし、次にアクション・ボタン 1、2、3、4、および 5 をクリックして、ObjectGrid およびマップに対するアクションを生成します。この時点では、このサーブレット・ページを閉じないでください。
3. 管理コンソールで、「**モニターおよびチューニング**」 > 「**Performance Monitoring Infrastructure**」 > `server_name` をクリックします。「**ランタイム**」タブをクリックします。
4. 「**カスタム**」ラジオ・ボタンをクリックします。
5. ランタイム・ツリーで「ObjectGrid Maps」モジュールを展開し、「clusterObjectGrid」リンクをクリックします。「ObjectGrid マップ」グループの下に、clusterObjectGrid という名前の 1 つの ObjectGrid インスタンスが存在し、この clusterObjectGrid グループの下に、counters、employees、offices、および sites という 4 つのマップが存在します。ObjectGrids インスタンスには clusterObjectGrid インスタンスが存在し、そのインスタンスの下には、DEFAULT という名前のトランザクション・タイプがあります。
6. 興味のある統計を使用可能にすることができます。例えば、従業員マップのマップ・エントリー数、および DEFAULT トランザクション・タイプのトランザクション応答時間を使用可能にできます。

## 次のタスク

PMI が使用可能になった後、管理コンソールまたはスクリプトを介して PMI 統計を表示することができます。

## PMI 統計の取得

PMI 統計を取得することによって、eXtreme Scale アプリケーションのパフォーマンスを確認できます。

### 始める前に

- ご使用の環境の PMI 統計追跡を使用可能にします。詳しくは、530 ページの『PMI の使用可能化』を参照してください。
- このタスクにあるパスはサンプル・アプリケーションの統計を取得することを前提としたものですが、これらの統計は類似のステップを含む他のアプリケーションに対しても使用できます。
- 管理コンソールを使用して統計を取得する場合は、管理コンソールにログインできなければなりません。スクリプトを使用する場合は、wsadmin にログインできなければなりません。

### このタスクについて

管理コンソールまたはスクリプトでステップを完了すれば、PMI 統計を取得して Tivoli Performance Viewer で表示することができます。

- 管理コンソールのステップ
- スクリプトのステップ

取得できる統計についての詳細は、534 ページの『PMI モジュール』を参照してください。

## 手順

- 管理コンソールで PMI 統計を取得します。
  1. 管理コンソールで、「モニターおよびチューニング」 > 「パフォーマンス・ビューアー」 > 「現行アクティビティ」をクリックします。
  2. Tivoli Performance Viewer を使用してモニターするサーバーを選択してから、モニターを使用可能にします。
  3. サーバーをクリックして、「Performance viewer」ページを表示します。
  4. 構成ツリーを展開します。「ObjectGrid マップ」 > 「clusterObjectGrid」をクリックし、「従業員」を選択します。「ObjectGrids」 > 「clusterObjectGrid」を展開し、「DEFAULT」を選択します。
  5. ObjectGrid サンプル・アプリケーションで、ObjectGridCreationServlet サブレットに移動し、ボタン 1 をクリックしてから、マップを取り込みます。ビューアーに統計が表示されます。
- スクリプトを使用して PMI 統計を取得します。
  1. コマンド行プロンプトで、`was_root/bin` ディレクトリーに移動します。`wsadmin` と入力して `wsadmin` ツールを開始します。
  2. 以下のコマンドを使用して、環境の変数を設定します。

```
wsadmin>set perfName [$AdminControl completeObjectName type=Perf,]*
wsadmin>set perfOName [$AdminControl makeObjectName $perfName]
wsadmin>set mySrvName [$AdminControl completeObjectName type=Server,
name=APPLICATION_SERVER_NAME,]*
```
  3. 以下のコマンドを使用して、`mapModule` 統計を取得するための変数を設定します。

```
wsadmin>set params [java::new {java.lang.Object[]} 3]
wsadmin>$params set 0 [$AdminControl makeObjectName $mySrvName]
wsadmin>$params set 1 [java::new java.lang.String mapModule]
wsadmin>$params set 2 [java::new java.lang.Boolean true]
wsadmin>set sigs [java::new {java.lang.String[]} 3]
wsadmin>$sigs set 0 javax.management.ObjectName
wsadmin>$sigs set 1 java.lang.String
wsadmin>$sigs set 2 java.lang.Boolean
```
  4. 以下のコマンドを使用して、`mapModule` 統計を取得します。

```
wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
```
  5. 以下のコマンドを使用して、`objectGridModule` 統計を取得するための変数を設定します。

```
wsadmin>set params2 [java::new {java.lang.Object[]} 3]
wsadmin>$params2 set 0 [$AdminControl makeObjectName $mySrvName]
wsadmin>$params2 set 1 [java::new java.lang.String objectGridModule]
wsadmin>$params2 set 2 [java::new java.lang.Boolean true]
wsadmin>set sigs2 [java::new {java.lang.String[]} 3]
wsadmin>$sigs2 set 0 javax.management.ObjectName
wsadmin>$sigs2 set 1 java.lang.String
wsadmin>$sigs2 set 2 java.lang.Boolean
```
  6. 以下のコマンドを使用して、`objectGridModule` 統計を取得します。

```
wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
```

## タスクの結果

Tivoli Performance Viewer で統計を表示することができます。

## PMI モジュール

Performance Monitoring Infrastructure (PMI) モジュールを使用してアプリケーションのパフォーマンスをモニターすることができます。

### objectGridModule

objectGridModule は時間統計 (トランザクション応答時間) を含みます。トランザクションは、Session.begin メソッド呼び出しと Session.commit メソッド呼び出しの間の所要時間として定義されます。この所要時間は、トランザクション応答時間として追跡されます。objectGridModule のルート・エレメント (root) は、WebSphere eXtreme Scale 統計への入り口点として機能します。このルート・エレメントは ObjectGrid を子エレメントとして持ち、さらにこの子エレメントはトランザクション・タイプの子エレメントとして持ちます。応答時間統計はそれぞれのトランザクション・タイプと関連しています。

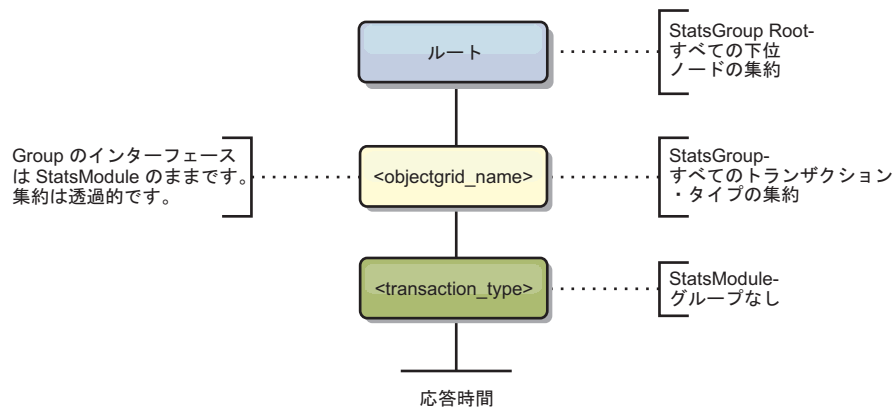


図 58. ObjectGridModule モジュールの構造

次の図は ObjectGridModule 構造の例です。この例では、2 つの ObjectGrid インスタンス (ObjectGrid A と ObjectGrid B) がシステムに存在します。ObjectGrid A インスタンスには 2 つのトランザクション・タイプ (A とデフォルト) があります。ObjectGrid B インスタンスにはデフォルトのトランザクション・タイプのみがあります。

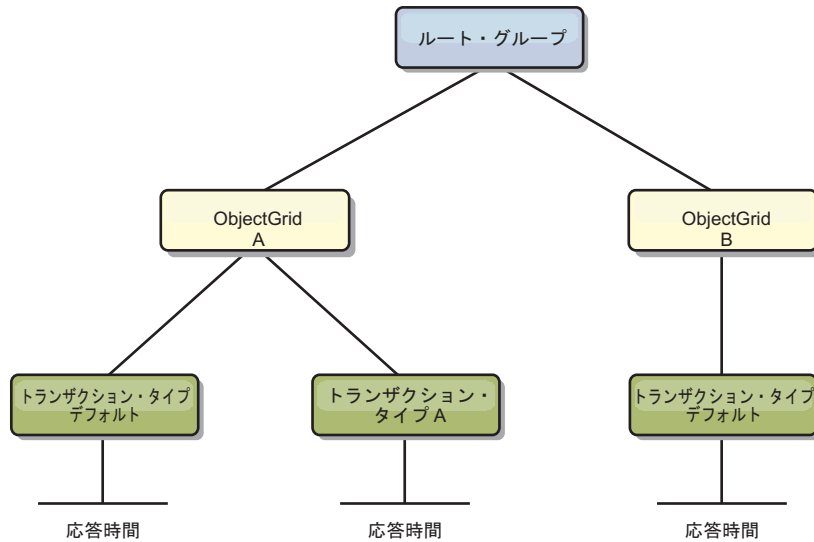


図 59. ObjectGridModule モジュール構造の例

アプリケーション開発者は、アプリケーションがどのタイプのトランザクションを使用するのかを知っているため、トランザクション・タイプはアプリケーション開発者によって定義されます。トランザクション・タイプの設定は、次の `Session.setTransactionType(String)` メソッドを使用して行われます。

```

/**
 * Sets the transaction type for future transactions.
 *
 * After this method is called, all of the future transactions have the
 * same type until another transaction type is set. If no transaction
 * type is set, the default TRANSACTION_TYPE_DEFAULT transaction type
 * is used.
 *
 * Transaction types are used mainly for statistical data tracking purpose.
 * Users can predefine types of transactions that run in an
 * application. The idea is to categorize transactions with the same characteristics
 * to one category (type), so one transaction response time statistic can be
 * used to track each transaction type.
 *
 * This tracking is useful when your application has different types of
 * transactions.
 * Among them, some types of transactions, such as update transactions, process
 * longer than other transactions, such as read-only transactions. By using the
 * transaction type, different transactions are tracked by different statistics,
 * so the statistics can be more useful.
 *
 * @param tranType the transaction type for future transactions.
 */
void setTransactionType(String tranType);

```

次の例は、`updatePrice` へのトランザクション・タイプを設定します。

```

// Set the transaction type to updatePrice
// The time between session.begin() and session.commit() will be
// tracked in the time statistic for "updatePrice".
session.setTransactionType("updatePrice");
session.begin();
map.update(stockId, new Integer(100));
session.commit();

```

最初の行は、後続のトランザクション・タイプが `updatePrice` であることを示します。`updatePrice` 統計は、例にあるセッションに対応する `ObjectGrid` インスタンスに置かれています。Java Management Extensions (JMX) インターフェースを使用して、`updatePrice` トランザクション用のトランザクション応答時間を取得できます。指定した `ObjectGrid` インスタンスで、トランザクションのすべてのタイプの集約統計も取得できます。

## mapModule

`mapModule` は、eXtreme Scale マップに関連した 3 つの統計を含んでいます。

- **マップ・ヒット率** - *BoundedRangeStatistic*: マップのヒット率を追跡します。ヒット率は 0 以上 100 以下の浮動値で、マップ取得操作に関するマップ・ヒットの比率です。
- **エントリー数** - *CountStatistic*: マップのエントリー数を追跡します。
- **ローダー・バッチ更新応答時間** - *TimeStatistic*: ローダー・バッチ更新操作に使用される応答時間を追跡します。

`mapModule` のルート・エレメント (root) は、`ObjectGrid` マップ統計への入り口点として機能します。このルート・エレメントは `ObjectGrid` を子エレメントとして持ち、さらにこの子エレメントはマップを子エレメントとして持ちます。すべてのマップ・インスタンスは、3 つのリスト統計を持っています。次の図は `mapModule` 構造です。

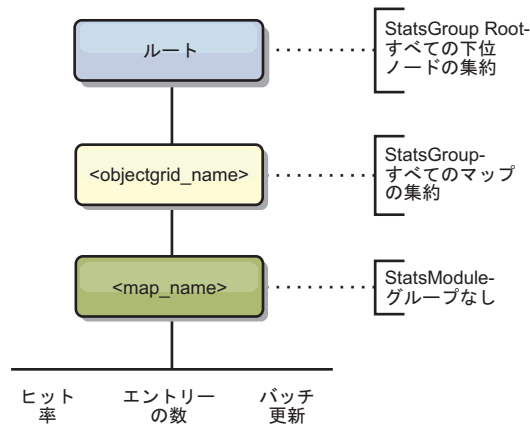
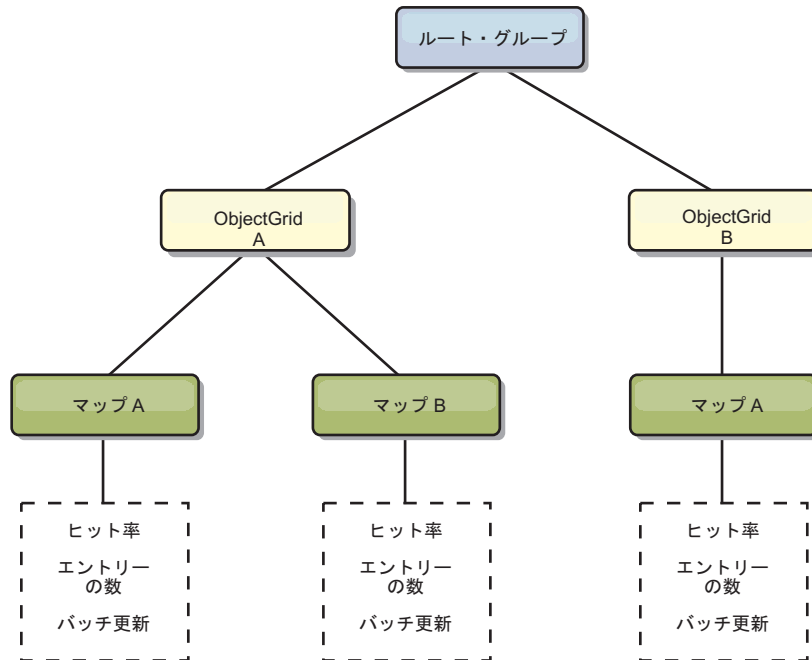


図 60. `mapModule` 構造

次の図は `mapModule` 構造の例です。

図 61. `mapModule` モジュール構造の例





## hashIndexModule

hashIndexModule は、マップ・レベルの索引に関連する次の統計を含みます。

- **検索カウント** -CountStatistic: 索引検索操作の呼び出し回数。
- **衝突カウント** -CountStatistic: 検索操作の衝突回数。
- **障害カウント** -CountStatistic: 検索操作の障害件数。
- **結果カウント** -CountStatistic: 検索操作から戻されたキーの数。
- **バッチ更新カウント** -CountStatistic: この索引に対するバッチ更新の回数。対応するマップが何らかの方法で変更されると、索引で、その doBatchUpdate() メソッドが呼び出されます。この統計からは、索引の変更または更新頻度が分かります。
- **検索操作所要時間** -TimeStatistic: 検索操作が完了するまでに要する時間。

hashIndexModule のルート・エレメント (root) は、HashIndex 統計への入り口点として機能します。このルート・エレメントは ObjectGrid を子エレメントとして持ちます。ObjectGrid はマップを子エレメントとして持ち、最終的にこの子エレメントは HashIndex を子エレメントおよびツリーのリーフ・ノードとして持ちます。すべての HashIndex インスタンスは 3 つのリスト統計を持っています。次の図は hashIndexModule の構造です。

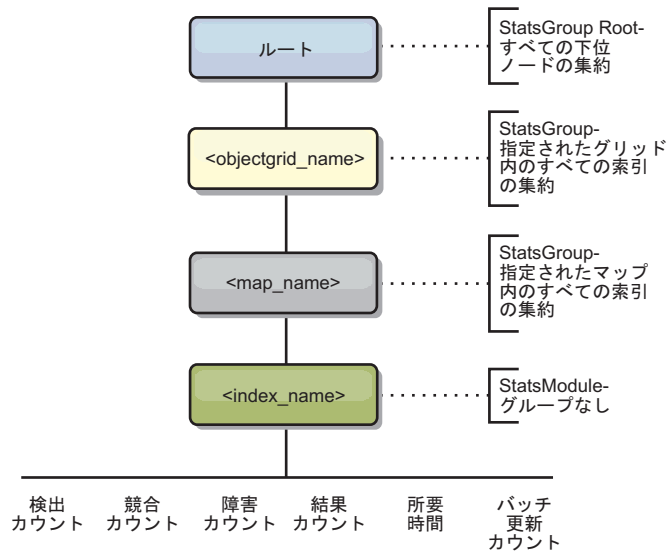


図 62. hashIndexModule モジュール構造

次の図は hashIndexModule 構造の例です。

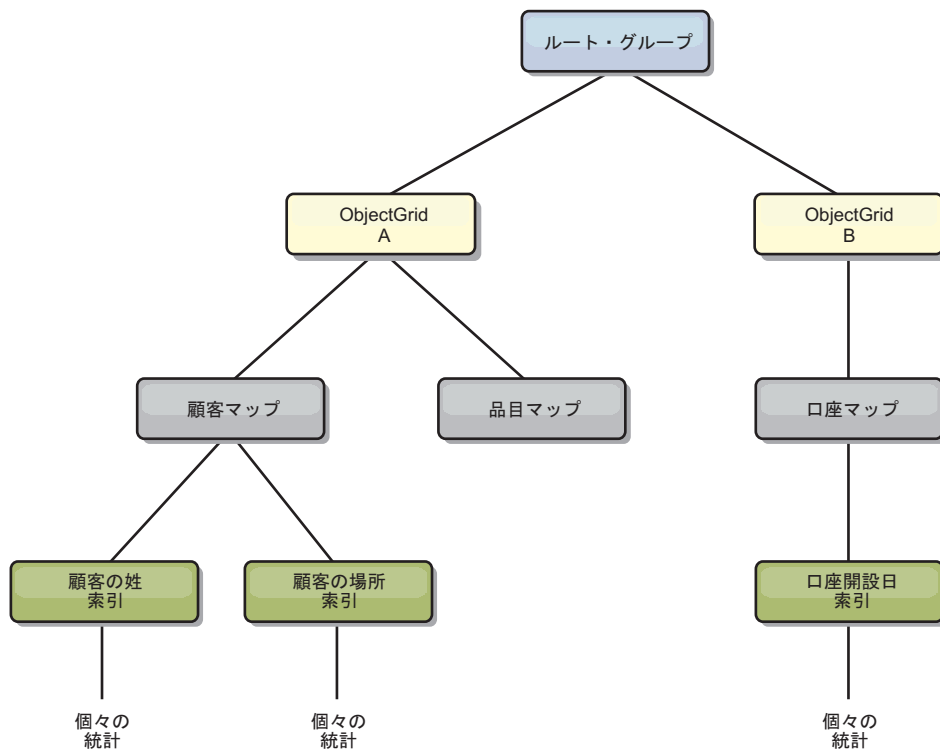


図 63. hashIndexModule モジュール構造の例

### agentManagerModule

agentManagerModule は、マップ・レベルのエージェントに関連した統計を含みます。

- 削減時間: *TimeStatistic* - エージェントが削減操作を完了するまでの時間。

- **合計所要時間:** *TimeStatistic* - エージェントがすべての操作を完了するまでの合計時間。
- **エージェント・シリアライゼーション時間:** *TimeStatistic* - エージェントをシリアライズするための時間。
- **エージェント・インフレーション時間:** *TimeStatistic* - サーバー上でエージェントをインフレーションするのに要する時間。
- **結果シリアライゼーション時間:** *TimeStatistic* - エージェントからの結果をシリアライズするための時間。
- **結果インフレーション時間:** *TimeStatistic* - エージェントからの結果をインフレーションするための時間。
- **障害カウント:** *CountStatistic* - エージェントが障害を起こした回数。
- **呼び出しカウント:** *CountStatistic* - AgentManager が呼び出された回数。
- **区画カウント:** *CountStatistic* - エージェントが送られる相手区画の数。

agentManagerModule のルート・エレメント (root) は、AgentManager 統計への入り口点として機能します。このルート・エレメントは ObjectGrid を子エレメントとして持ちます。ObjectGrid はマップを子エレメントとして持ち、最終的にこの子エレメントは AgentManager インスタンスを子エレメントおよびツリーのリーフ・ノードとして持ちます。すべての AgentManager インスタンスは統計を持っています。

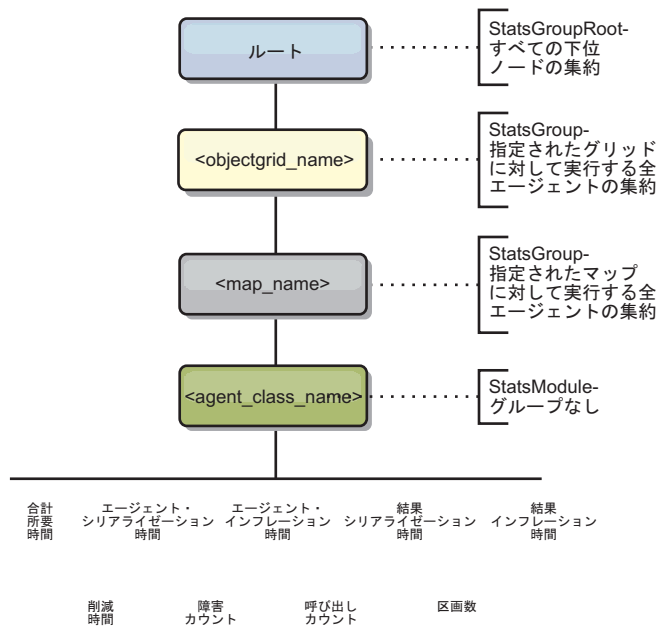


図 64. agentManagerModule 構造

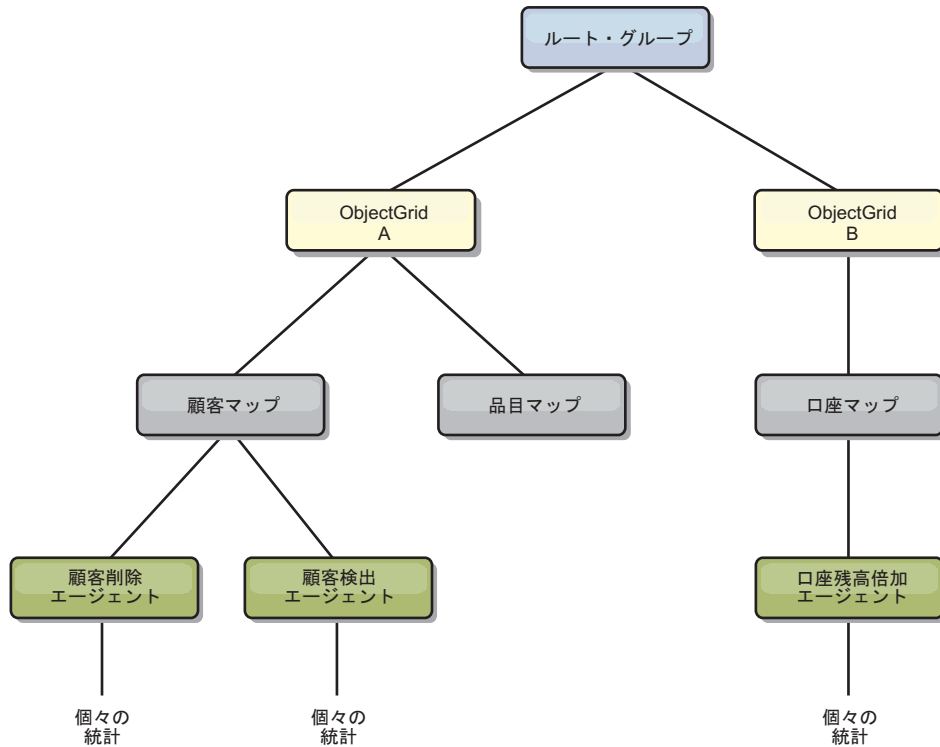


図 65. agentManagerModule 構造の例

## queryModule

queryModule は、eXtreme Scale 照会に関連した統計を含みます。

- 計画作成時間: *TimeStatistic* - 照会計画を作成するための時間。
- 実行時間: *TimeStatistic* - 照会を実行するための時間。
- 実行カウント: *CountStatistic* - 照会が実行された回数。
- 結果カウント: *CountStatistic* - 実行された各照会の結果セットごとのカウント。
- 障害カウント: *CountStatistic* - 照会が失敗した回数。

queryModule のルート・エレメント (root) は、Query 統計への入り口点として機能します。このルート・エレメントは ObjectGrid を子エレメントとして持ち、さらにこの子エレメントは照会オブジェクトを子エレメントおよびツリーのリーフ・ノードとして持ちます。すべての Query インスタンスは 3 つのリスト統計を持っています。

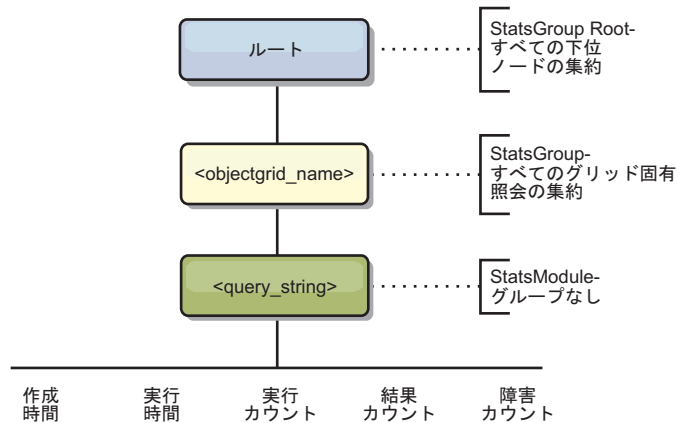


図 66. queryModule の構造

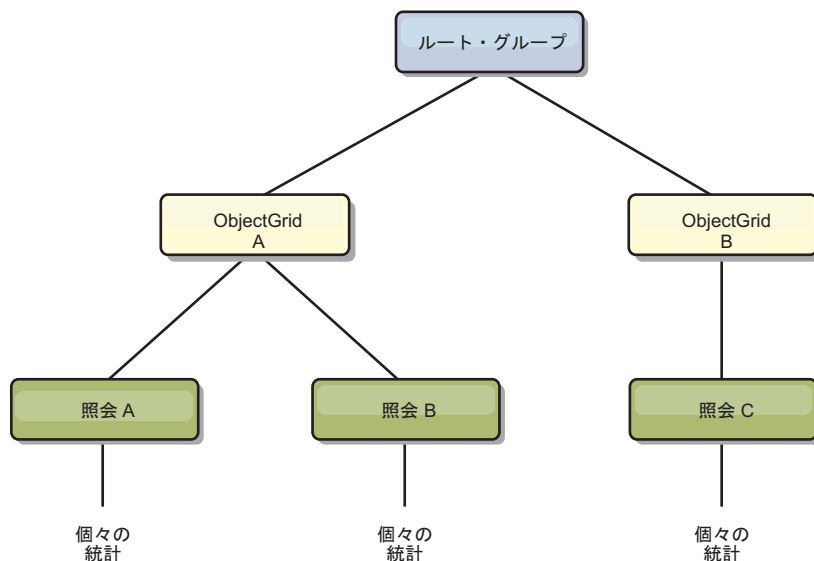


図 67. QueryStats.jpg queryModule 構造の例

## wsadmin ツールを使用した Managed Beans (MBeans) へのアクセス

WebSphere Application Server で提供される wsadmin ユーティリティを使用し、Managed Bean (MBean) 情報にアクセスすることができます。

### 手順

WebSphere Application Server インストール内の bin ディレクトリーから wsadmin ツールを実行します。次の例は、動的 eXtreme Scale における現在の断片配置のビューを取得するものです。wsadmin ツールは、eXtreme Scale が稼働している任意のインストール済み環境から実行できます。wsadmin ツールをカタログ・サービスで実行する必要はありません。

```

$ wsadmin.sh -lang jython
wsadmin>placementService = AdminControl.queryNames
("com.ibm.websphere.objectgrid:*,type=PlacementService")
wsadmin>print AdminControl.invoke(placementService,
"listObjectGridPlacement","library ms1")

<objectGrid name="library" mapSetName="ms1">
  <container name="container-0" zoneName="DefaultDomain"
  hostname="host1.company.org" serverName="server1">
    <shard type="Primary" partitionName="0"/>
    <shard type="SynchronousReplica" partitionName="1"/>
  </container>
  <container name="container-1" zoneName="DefaultDomain"
  hostname="host2.company.org" serverName="server2">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="Primary" partitionName="1"/>
  </container>
  <container name="UNASSIGNED" zoneName="_ibm_SYSTEM"
  hostname="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
  </container>
</objectGrid>

```

---

## 管理 Bean (MBean) を使用したモニター

ご使用の環境の統計をトラッキングするために Managed Bean (MBean) を使用できます。

### 始める前に

属性が記録されるようにするには、統計を使用可能に設定する必要があります。統計は、以下のいずれかの方法で使用可能にできます。

- **サーバー・プロパティ・ファイルを使用:**

サーバー・プロパティ・ファイルの `statsSpec=<StatsSpec>` というキー値エントリを設定して統計を使用可能にすることができます。次に、設定可能な例をいくつか示します。

- すべての統計を使用可能にする場合は、`statsSpec=all=enabled` を使用します。
- ObjectGrid 統計のみを使用可能にする場合は、`statsSpec=og.all=enabled` を使用します。使用可能なすべての統計の仕様の説明を確認するには、API 資料の `StatsSpec API` を参照してください。

サーバー・プロパティ・ファイルに関して詳しくは、サーバー・プロパティ・ファイルを参照してください。

- **管理 Bean を使用して:**

ObjectGrid MBean で `StatsSpec` 属性を使用することで、統計を使用可能にできます。詳しくは、API 資料の `StatsSpec API` を参照してください。

- **プログラムで:**

`StatsAccessor` インターフェースを使用して、統計をプログラムで使用可能にすることもできます。このインターフェースは、`StatsAccessorFactory` クラスを使用し



て取得されます。このインターフェースは、クライアント環境で使用するか、現行プロセスで実行中のデータ・グリッドをモニターする必要がある場合に使用します。

## 手順

- **wsadmin** ツールを使用して **MBean** 統計にアクセスします。

詳しくは、498 ページの『wsadmin ツールを使用した Managed Beans (MBeans) へのアクセス』を参照してください。

- **プログラム**で **MBean** 統計にアクセスします。

詳しくは、498 ページの『Managed Bean (MBean) へのプログラマチックなアクセス』を参照してください。

## 例

Managed Bean の使用例は、サンプル: **xsadmin** ユーティリティを参照してください。

---

## ベンダー・ツールによるモニター

一般的によく使われるいくつかのエンタープライズ・モニタリング・ソリューションを使用して、WebSphere eXtreme Scale をモニターすることができます。パブリックにアクセス可能な管理 Bean を使用して WebSphere eXtreme Scale をモニターする IBM Tivoli Monitoring および Hyperic HQ 用に、プラグイン・エージェントが組み込まれています。CA Wily Introscope は Java メソッドのインストルメンテーションを使用して、統計情報を収集します。

## IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale のモニター

IBM Tivoli Enterprise Monitoring Agent は、分散環境およびホスト環境のデータベース、オペレーティング・システム、およびサーバーをモニターするために使用できる機能の豊富なモニタリング・ソリューションです。WebSphere eXtreme Scale には、eXtreme Scale 管理 Bean をイントロスペクトする場合に使用できるカスタマイズ・エージェントが含まれています。このソリューションは、スタンドアロン eXtreme Scale および WebSphere Application Server デプロイメントの両方で効果的に機能します。

### 始める前に

- WebSphere eXtreme Scale バージョン 7.0.0 以降をインストールします。

また、WebSphere eXtreme Scale サーバーから統計データを収集するには、統計を使用可能にする必要があります。統計を使用可能にする各種オプションについては、542 ページの『管理 Bean (MBean) を使用したモニター』およびサンプル: **xsadmin** ユーティリティを参照してください。

- IBM Tivoli Monitoring バージョン 6.2.1 (フィックスパック 2 以降の適用済み) をインストールします。

- eXtreme Scale サーバーが実行される各サーバーまたはホストに Tivoli OS エージェントをインストールします。
- WebSphere eXtreme Scale エージェントをインストールします。(IBM Open Process Automation Library (OPAL) サイトから無料でダウンロードできます。)

以下の手順を実行して、Tivoli Monitoring Agent をインストールおよび構成します。

## 手順

1. WebSphere eXtreme Scale 用に Tivoli Monitoring Agent をインストールします。

Tivoli インストール・イメージをダウンロードし、そのファイルを一時ディレクトリに解凍します。

2. eXtreme Scale アプリケーション・サポート・ファイルをインストールします。

以下の各デプロイメントに eXtreme Scale アプリケーション・サポートをインストールします。

- Tivoli Enterprise Portal Server (TEPS)
  - Enterprise Desktop クライアント (TEPD)
  - Tivoli Enterprise Monitoring Server (TEMS)
- a. 作成した一時ディレクトリから、新しいコマンド・ウィンドウを開始し、ご使用のプラットフォームに合った実行可能ファイルを実行します。インストール・スクリプトが、ご使用の Tivoli デプロイメント・タイプ (TEMS、TEPD、または TEPS) を自動的に検出します。タイプによらず単一のホストでも複数のホストでもインストールできますが、デプロイメントの場合は 3 つのタイプのすべてについて eXtreme Scale エージェントのアプリケーション・サポート・ファイルをインストールする必要があります。
  - b. 「インストーラー」ウィンドウで、デプロイされた Tivoli コンポーネントの選択が正しいことを確認します。「次へ」をクリックします。
  - c. プロンプトが出されたら、ホスト名および管理資格情報をサブミットします。「次へ」をクリックします。
  - d. 「**Monitoring Agent for WebSphere eXtreme Scale**」を選択します。「次へ」をクリックします。
  - e. 実行されるインストール・アクションについて通知があります。「次へ」をクリックすると、インストールの進行状況を完了まで確認することができます。

手順を完了すると、WebSphere eXtreme Scale エージェントが必要とするすべてのアプリケーション・サポート・ファイルがインストールされます。

3. 各 eXtreme Scale ノードにエージェントをインストールします。

各コンピューターに Tivoli OS エージェントをインストールします。このエージェントを構成または開始する必要はありません。上記のステップと同じインストール・イメージを使用して、プラットフォーム固有の実行可能ファイルを実行します。

指針としては、ホストごとにエージェントを 1 つだけインストールする必要がある場合があります。1 つのエージェントで eXtreme Scale サーバーの多数の

インスタンスをサポートすることができます。1つのエージェント・インスタンスで約50のeXtreme Scaleサーバーをモニターすると、最適のパフォーマンスが得られます。

- a. 「インストール・ウィザード」初期画面で「次へ」をクリックすると、インストール・パス情報を指定する画面が開きます。
- b. 「**Tivoli Monitoring** インストール・ディレクトリー」フィールドに、C:\IBM\ITM (または /opt/IBM/ITM) と入力するか、またはこのパスを参照します。次に「インストール可能なメディアのロケーション」フィールドで、表示されている値が正しいことを確認してから「次へ」をクリックします。
- c. 追加するコンポーネント (「ソリューションのローカル・インストールの実行」など) を選択して、「次へ」をクリックします。
- d. サポートを追加する対象のアプリケーション (「**Monitoring Agent for WebSphere eXtreme Scale**」など) を選択して、「次へ」をクリックします。
- e. アプリケーション・サポートが正常に追加されるまで進行状況を確認することができます。

**注:** それぞれの eXtreme Scale ノードにこれまでのステップを繰り返します。サイレント・インストールを使用することもできます。サイレント・インストールに関して詳しくは、IBM Tivoli Monitoring インフォメーション・センターを参照してください。

#### 4. WebSphere eXtreme Scale エージェントを構成します。

インストールした各エージェントを、カタログ・サーバー、eXtreme Scale サーバー、またはその両方をモニターするように構成する必要があります。

Windows プラットフォームと UNIX プラットフォームとは構成手順が異なります。Windows プラットフォームの場合の構成は、**Tivoli Monitoring サービスの管理**ユーザー・インターフェースを使用して実行します。UNIX プラットフォームの場合の構成はコマンド行に基づいて行います。

**Windows** 以下のステップを使用して、まず Windows 上のエージェントを構成します

- a. 「**Tivoli Enterprise Monitoring サービスの管理**」ウィンドウで、「スタート」 > 「すべてのプログラム」 > 「**IBM Tivoli Monitoring**」 > 「**Tivoli Monitoring サービスの管理**」の順にクリックします。
- b. 「**Monitoring Agent for WebSphere eXtreme Scale**」を右クリックし、「**デフォルトを使用して構成**」を選択します。そうすると、エージェントに固有のインスタンスを作成するウィンドウが開きます。
- c. 固有の名前 (例えば「instance1」など) を入力し、「次へ」をクリックします。
- スタンドアロンの eXtreme Scale サーバーをモニターする場合は、次のステップを実行します。
  - a. Java パラメーターを更新し、「**Java Home**」の値が正しいことを確認します。JVM 引数は空のままでもかまいません。「次へ」をクリックします。

- b. 「MBean サーバー接続タイプ」のタイプを選択し、スタンドアロン eXtreme Scale サーバーの場合は「JSR-160 準拠サーバー」を使用します。「次へ」をクリックします。
- c. セキュリティーが有効な場合、「ユーザー ID」および「パスワード」値を更新します。「JMX サービス URL」の値は、そのままにしておきます。この値は、後でオーバーライドします。「JMX クラスパス情報」フィールドはそのままにしておきます。「次へ」をクリックします。

Windows でエージェント用にサーバーを構成するには、以下のステップを実行します。

- a. 「WebSphere eXtreme Scale グリッド・サーバー」ペインで eXtreme Scale サーバーのサブノード・インスタンスをセットアップします。ご使用のコンピューター上にコンテナ・サーバーがない場合、「次へ」をクリックして、カタログ・サービス・ペインに進みます。
  - b. ご使用のコンピューターに複数の eXtreme Scale コンテナ・サーバーがある場合、エージェントで各サーバーをモニターするように構成します。
  - c. それぞれの名前とポートが固有な場合は、「新規」をクリックし、eXtreme Scale サーバーを必要なだけいくつでも追加することができます。(eXtreme Scale サーバーが始動されるときは、JMXPort 値が指定されていなければなりません。)
  - d. コンテナ・サーバーの構成を完了したならば、「次へ」をクリックして「WebSphere eXtreme Scale カタログ・サーバー」ペインに進みます。
  - e. カタログ・サーバーがない場合は、「OK」をクリックします。カタログ・サーバーがある場合は、コンテナ・サーバーについて実行したのと同様に各サーバーに新しい構成を追加します。ここでもまた、固有の名前(できればカタログ・サービスを開始するときに使用するものと同じ名前)を選択します。「OK」をクリックして終了します。
- WebSphere Application Server プロセス内に組み込まれている eXtreme Scale サーバー上でエージェントのサーバーをモニターする場合、以下のステップを実行します。
- a. Java パラメーターを更新し、「Java Home」の値が正しいことを確認します。JVM 引数は空のままでもかまいません。「次へ」をクリックします。
  - b. 「MBean サーバー接続タイプ」を選択します。ご使用の環境に適した WebSphere Application Server バージョンを選択します。「次へ」をクリックします。
  - c. パネルの WebSphere Application Server 情報が正しいことを確認します。「次へ」をクリックします。
  - d. サブノード定義を 1 つのみ追加します。サブノード定義に名前を指定し、ポート定義は更新しないでください。WebSphere Application Server 環境内で、そのコンピューター上で実行中のノード・エージェントによって管理されるすべてのアプリケーション・サーバーからデータが収集されます。「次へ」をクリックします。
  - e. この環境にカタログ・サーバーが存在しない場合は、「OK」をクリックします。カタログ・サーバーがある場合は、コンテナ・サーバーと同様、それぞれのカタログ・サーバーについて新しい構成を追加します。カタロ

グ・サービスに固有の名前、できればカタログ・サービスを開始するときに使用するものと同じ名前を選択します。「OK」をクリックして終了します。

**注:** コンテナ・サーバーは、カタログ・サービスと連結する必要はありません。

これでエージェントとサーバーが構成されて作動可能になるので、次のウィンドウで「instance1」を右クリックしてエージェントを開始します。

**UNIX** UNIX プラットフォーム上のエージェントをコマンド行で構成する場合は、以下のステップを実行します。

次に JSR160 準拠の接続タイプを使用するスタンドアロン・サーバーの例を示します。この例では、単一ホスト (rhea00b02) 上に 3 つの eXtreme Scale コンテナがあり、JMX リスナーのアドレスは、それぞれ 15000、15001 および 15002 です。カタログ・サーバーはありません。

構成ユーティリティからの出力はモノスペースのイタリック体で、一方ユーザー応答はモノスペースの太字体で示されています。(ユーザー応答が不要であった場合は、Enter キーを押してデフォルトが選択されました。)

```
rhea00b02 # ./itcmd config -A xt
Agent configuration started...
Enter instance name (default is: ): inst1
Edit "Monitoring Agent for WebSphere eXtreme Scale" settings? [ 1=Yes, 2=No ] (default is: 1):
Edit 'Java' settings? [ 1=Yes, 2=No ] (default is: 1):
Java home (default is: C:\Program Files\IBM\Java50): /opt/OG61/java
Java trace level [ 1=Error, 2=Warning, 3=Information, 4=Minimum Debug, 5=Medium Debug, 6=Maximum Debug,
7=All ] (default is: 1):
JVM arguments (default is: ):
Edit 'Connection' settings? [ 1=Yes, 2=No ] (default is: 1):
MBean server connection type [ 1=JSR-160-Compliant Server, 2=WebSphere Application Server version 6.0,
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (default is: 1): 1
Edit 'JSR-160-Compliant Server' settings? [ 1=Yes, 2=No ] (default is: 1):
JMX user ID (default is: ):
Enter JMX password (default is: ):
Re-type : JMX password (default is: ):
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:port/objectgrid/MBeanServer):
-----
JMX Class Path Information
JMX base paths (default is: ):
JMX class path (default is: ):
JMX JAR directories (default is: ):
Edit 'WebSphere eXtreme Scale Catalog Service' settings? [ 1=Yes, 2=No ] (default is: 1): 2
Edit 'WebSphere eXtreme Scale Grid Servers' settings? [ 1=Yes, 2=No ] (default is: 1): 1
No 'WebSphere eXtreme Scale Grid Servers' settings available?
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c0
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15000/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings: WebSphere eXtreme Scale Grid Servers=ogx
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c1
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings: WebSphere eXtreme Scale Grid Servers= rhea00b02_c1
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c2
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings: WebSphere eXtreme Scale Grid Servers= rhea00b02_c2
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 5

Will this agent connect to a TEMS? [1=YES, 2=NO] (Default is: 1):
```



```

TEMS Host Name (Default is: rhea00b00):

Network Protocol [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):

    Now choose the next protocol number from one of these:
    - ip
    - sna
    - ip.spipe
    - 0 for none
Network Protocol 2 (Default is: 0):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):

Configure connection for a secondary TEMS? [1=YES, 2=NO] (Default is: 2):
Enter Optional Primary Network Name or 0 for "none" (Default is: 0):
Agent configuration completed...

```

上記の例では、「inst1」というエージェント・インスタンスが作成され、Java ホーム設定が更新されます。eXtreme Scale コンテナ・サーバーは構成されますが、カタログ・サービスは構成されません。

**注:** 上記の手順では、次の形式のテキスト・ファイルがディレクトリー <ITM\_install>/config/<host>\_xt\_<instance name>.cfg に作成されます。

**例:** rhea00b02\_xt\_inst1.cfg

自分で選んだプレーン・テキスト・エディターを使用してこのファイルを編集することをお勧めします。そうしたファイルの内容の例を以下に示します。

```

INSTANCE=inst2 [SECTION=KQZ_JAVA [ { JAVA_HOME=/opt/OG61/java } { JAVA_TRACE_LEVEL=ERROR } ]
SECTION=KQZ_JMX_CONNECTION_SECTION [ { KQZ_JMX_CONNECTION_PROPERTY=KQZ_JMX_JSR160_JSR160 } ]
SECTION=KQZ_JMX_JSR160_JSR160 [ { KQZ_JMX_JSR160_JSR160_CLASS_PATH_TITLE= }
{ KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localhost:port/objectgrid/MBeanServer } { KQZ_JMX_JSR160_JSR160_CLASS_PATH_SEPARATOR= } ]
SECTION=OGS:rhea00b02_c1 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer } ]
SECTION=OGS:rhea00b02_c0 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer } ]
SECTION=OGS:rhea00b02_c2 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer } ] ] ]

```

次に、WebSphere Application Server デプロイメント上の構成の例を示します。

```

rhea00b02 # ./itmcmd config -A xt
Agent configuration started...
Enter instance name (default is: ): inst1
Edit "Monitoring Agent for WebSphere eXtreme Scale" settings? [ 1=Yes, 2=No ] (default is: 1): 1
Edit 'Java' settings? [ 1=Yes, 2=No ] (default is: 1): 1
Java home (default is: C:\Program Files\IBM\Java50): /opt/WAS61/java
Java trace level [ 1=Error, 2=Warning, 3=Information, 4=Minimum Debug, 5=Medium Debug, 6=Maximum Debug, 7=All ] (default is: 1):
JVM arguments (default is: ):
Edit 'Connection' settings? [ 1=Yes, 2=No ] (default is: 1):
MBean server connection type [ 1=JSR-160-Compliant Server, 2=WebSphere Application Server version 6.0, 3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (default is: 1): 4
Edit 'WebSphere Application Server version 7.0' settings? [ 1=Yes, 2=No ] (default is: 1): WAS user ID (default is: ):
Enter WAS password (default is: ):
Re-type : WAS password (default is: ):
WAS host name (default is: localhost): rhea00b02
WAS port (default is: 2809):
WAS connector protocol [ 1=rmi, 2=soap ] (default is: 1):
WAS profile name (default is: ): default
-----
WAS Class Path Information
WAS base paths (default is: C:\Program Files\IBM\WebSphere\AppServer;opt/IBM/WebSphere/AppServer): /opt/WAS61
WAS class path (default is: runtimes/com.ibm.ws.admin.client_6.1.0.jar;runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar):
WAS JAR directories (default is: lib;plugins):
Edit 'WebSphere eXtreme Scale Grid Servers' settings? [ 1=Yes, 2=No ] (default is: 1):
No 'WebSphere eXtreme Scale Grid Servers' settings available?
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):

```



```
'WebSphere eXtreme Scale Grid Servers' settings: WebSphere eXtreme Scale Grid Servers=rhea00b02
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 5
Edit 'WebSphere eXtreme Scale Catalog Service' settings? [ 1=Yes, 2=No ] (default is: 1): 2
Will this agent connect to a TEMS? [1=YES, 2=NO] (Default is: 1):
TEMS Host Name (Default is: rhea00b02):

Network Protocol [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):

    Now choose the next protocol number from one of these:
    - ip
    - sna
    - ip.spipe
    - 0 for none
Network Protocol 2 (Default is: 0):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):

Configure connection for a secondary TEMS? [1=YES, 2=NO] (Default is: 2):
Enter Optional Primary Network Name or 0 for "none" (Default is: 0):
Agent configuration completed...
rhea00b02 #
```

WebSphere Application Server デプロイメントの場合、複数のサブノードを作成する必要はありません。eXtreme Scale エージェントは、ノード・エージェントに接続して、管理下のアプリケーション・サーバーからすべての情報を収集します。

SECTION=CAT はカタログ・サービス行を意味し、一方 SECTION=OGS は eXtreme Scale サーバー構成行を意味します。

- すべての eXtreme Scale コンテナ・サーバーの JMX ポートを構成します。

**-JMXServicePort** 引数が指定されないまま eXtreme Scale コンテナ・サーバーが始動されるときは、MBean サーバーに動的ポートが割り当てられます。エージェントは通信相手の JMX ポートをあらかじめ知っておく必要があります。エージェントは動的ポートとは連動しません。

サーバーの始動時、startOgServer.sh | .bat コマンドを使用して eXtreme Scale サーバーを開始する場合は **-JMXServicePort <port\_number>** 引数を指定します。このコマンドの実行により、プロセス内の JMX サーバーが静的事前定義ポートを listen するようになります。

UNIX インストールの上記の例の場合は、2 つの eXtreme Scale サーバーを次のように設定されたポートで始動する必要があります。

- "-JMXServicePort" "15000" (rhea00b02\_c0 の場合)
- "-JMXServicePort" "15001" (rhea00b02\_c1 の場合)
- eXtreme Scale エージェントを開始します。

inst1 インスタンスが作成されたとすると、上記の例のように、以下のコマンドを発行します。

- cd <ITM\_install>/bin
- itmcmd agent -o inst1 start xt
- eXtreme Scale エージェントを停止します。

「inst1」が上記の例のようにして作成されたインスタンスであったとして、以下のコマンドを発行します。

- cd <ITM\_install>/bin

- 2) `itmcmd agent -o inst1 stop xt`
6. すべての eXtreme Scale コンテナ・サーバーの統計を使用可能にします。

エージェントは、eXtreme Scale 統計 MBeans を使用して統計を記録します。以下の方式を使用して、eXtreme Scale 統計仕様を使用可能にする必要があります。

- サーバー・プロパティを構成して、コンテナ・サーバーの始動時にすべての統計を使用可能にします (all=enabled)。
- `xsadmin` サンプル・ユーティリティで、`-setstatsspec all=enabled` パラメータを使用して、すべてのアクティブ・コンテナの統計を使用可能にします。

## タスクの結果

すべてのサーバーが構成されて始動されたならば、IBM Tivoli Portal コンソールに MBean データが表示されます。事前定義ワークスペースには、グラフとデータ・メトリックがノード・レベルごとに表示されます。

モニターされるすべてのノードの「**eXtreme Scale グリッド・サーバー**」ノードについて、以下のワークスペースが定義されています。

- eXtreme Scale トランザクション・ビュー
- eXtreme Scale プライマリー断片ビュー
- eXtreme Scale メモリー・ビュー
- eXtreme Scale ObjectMap ビュー

独自のワークスペースも構成することができます。詳しくは、IBM Tivoli Monitoring インフォメーション・センターのワークスペースのカスタマイズに関する情報を参照してください。

## CA Wily Introscope による eXtreme Scale アプリケーションのモニター

CA Wily Introscope は、エンタープライズ・アプリケーション環境のパフォーマンス上の問題を検出して診断する場合に使用できるサード・パーティーの管理製品です。eXtreme Scale には、CA Wily Introscope を構成して eXtreme Scale ランタイムの選択部分をイントロスペクトし、eXtreme Scale アプリケーションを迅速に表示、検証する場合の詳細が含まれています。CA Wily Introscope は、スタンドアロン環境と WebSphere Application Server デプロイメント環境の両方で効果的に機能します。

### 概要

CA Wily Introscope を使用して eXtreme Scale アプリケーションをモニターするには、eXtreme Scale のモニター情報へのアクセスを可能にする設定で `ProbeBuilderDirective` (PBD) ファイルに作成する必要があります。

**重要:** Introscope のインスツルメンテーション・ポイントは、各フィックスパックまたはリリースで変わる可能性があります。新しいフィックスパックまたはリリースをインストールしたら、インスツルメンテーション・ポイントに変更がないか、資料を確認してください。

eXtreme Scale アプリケーションをモニターするように、CA Wily Introscope の ProbeBuilderDirective (PBD) ファイルを構成できます。CA Wily Introscope は、アプリケーション管理製品で、これを使用すると複雑な複合 Web アプリケーション環境におけるパフォーマンス上の問題を積極的に検出、選別、および診断することができます。

## カタログ・サービスのモニター用の PBD ファイル設定

カタログ・サービスをモニターするには、PBD ファイルの以下の設定を 1 つ以上を使用できます。

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl viewChangeCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl viewAboutToChange
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatCluster
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatNewServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl
importRouteInfo BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl joinPlacementGroup
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}" TraceOneMethodOfClass:
com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl classifyServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.BalanceGridEventListener shardActivated
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.BalanceGridEventListener shardDeactivate
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
```

### カタログ・サービスのモニター用のクラス

#### HAControllerImpl

HAControllerImpl クラスは、コア・グループのライフサイクル・イベントおよびフィードバック・イベントを処理します。このクラスをモニターすると、コア・グループの構造および変更を確認できます。

#### ServerAgent

ServerAgent クラスは、コア・グループ・イベントとカタログ・サービスの通信を担当します。さまざまなハートビート呼び出しをモニターして、主要なイベントを見極めることができます。

#### PlacementServiceImpl

PlacementServiceImpl クラスは、コンテナを調整します。このクラスのメソッドは、サーバーの結合イベントおよび配置イベントをモニターするために使用できます。

#### BalanceGridEventListener

BalanceGridEventListener クラスは、カタログのリーダーシップを制御します。このクラスをモニターすると、現在リーダーとして実行中のカタログ・サービスを確認できます。

## コンテナー・モニター用の PBD ファイル設定

コンテナーをモニターするには、PBD ファイルの以下の設定を 1 つ以上使用できます。

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ShardImpl processMessage
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.CommittedLogSequenceListenerProxy applyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.CommittedLogSequenceListenerProxy sendApplyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.checkpoint.CheckpointMapImpl$CheckpointIterator activateListener
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl viewChangeCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl viewAboutToChange
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent batchProcess
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeat
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatCluster
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent heartbeatNewServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
```

### コンテナー・モニター用のクラス

#### ShardImpl

ShardImpl クラスには、processMessage メソッドがあります。

processMessage メソッドは、クライアント要求のためのメソッドです。このメソッドを使用すると、サーバー・サイドの応答時間および要求数を確認できます。すべてのサーバー全体でカウントを監視し、ヒープ使用状況をモニターすることにより、グリッドのバランスが取れているかどうかを判別できます。

#### CheckpointIterator

CheckpointIterator クラスには、プライマリーをピア・モードにする activateListener メソッド呼び出しがあります。プライマリーがピア・モードになると、メソッド完了後に、レプリカがプライマリーにより更新されます。レプリカがプライマリー全体から再生成される場合、この操作に時間がかかることがあります。この操作が完了するまでは、システムの回復状態は不十分であるため、このクラスを使用してこの操作の進行状況をモニターできます。

#### CommittedLogSequenceListenerProxy

CommittedLogSequenceListenerProxy クラスには、2 つの興味深いメソッドがあります。applyCommitted メソッドは、すべてのトランザクションで実行され、sendApplyCommitted メソッドは、レプリカが情報をプルしているときに実行されます。これら 2 つのメソッドの実行頻度により、レプリカがプライマリーにどの程度後れを取らずに対応できているかがある程度分かります。

## クライアント・モニター用の PBD ファイル設定

クライアントをモニターするには、PBD ファイルの以下の設定を 1 つ以上使用できます。

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.client.ORBClientCoreMessageHandler sendMessage
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore bootstrap
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore epochChangeBootstrap
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.SessionImpl getMap
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ObjectGridImpl getSession
BlamePointTracerDifferentMethods "OGClient|{classname}|{method}"
TurnOn: ObjectMap
SetFlag: ObjectMap
IdentifyClassAs: com.ibm.ws.objectgrid.ObjectMapImpl ObjectMap
TraceComplexMethodsIfFlagged: ObjectMap BlamePointTracerDifferentMethods
"OGClient|{classname}|{method}"
```

### クライアント・モニター用のクラス

#### ORBClientCoreMessageHandler

ORBClientCoreMessageHandler クラスは、コンテナへのアプリケーション要求の送信を担当します。sendMessage メソッドでクライアントの応答時間および要求数をモニターできます。

#### ClusterStore

ClusterStore クラスには、クライアント・サイドでのルーティング情報が保持されます。

#### BaseMap

BaseMap クラスには、Evictor がマップからエントリを除去するとき呼び出される evictMapEntries メソッドがあります。

#### SelectionServiceImpl

SelectionServiceImpl クラスは、ルーティング上の決定を行います。クライアントによりフェイルオーバーに関する決定が下される場合、このクラスを使用すると、その決定から実行されるアクションを判別できます。

#### ObjectGridImpl

ObjectGridImpl クラスには、このメソッドに対する要求数を判別するためにモニターできる getSession メソッドがあります。

## Hyperic HQ による eXtreme Scale のモニター

Hyperic HQ は、サード・パーティーのモニター・ソリューションで、オープン・ソース・ソリューションあるいはエンタープライズ製品として自由に使用可能です。WebSphere eXtreme Scale には、あるプラグインが含まれていて、このプラグインにより Hyperic HQ エージェントは eXtreme Scale コンテナ・サーバーを検出し、eXtreme Scale 管理 Bean を使用して統計のレポートおよび集約を行うことができます。Hyperic HQ を使用すると、スタンドアロン eXtreme Scale デプロイメントをモニターできます。

## 始める前に

- この一連の説明は、Hyperic バージョン 4.0 を対象としています。これよりも新しいバージョンの Hyperic の場合、パス名やエージェントとサーバーの始動方法などの情報について Hyperic 資料を参照してください。
- Hyperic サーバーとエージェントのインストールをダウンロードします。サーバーのインストール済み環境が 1 つ実行中である必要があります。eXtreme Scale サーバーのすべてを検出するには、eXtreme Scale サーバーが稼働している各マシン上で Hyperic エージェントが実行中である必要があります。ダウンロード情報および資料のサポートは、Hyperic Web サイトを参照してください。
- `objectgrid-plugin.xml` および `hqplugin.jar` ファイルにアクセスする必要があります。これらのファイルは、`wxs_install_root/hyperic/etc` ディレクトリにあります。

## このタスクについて

eXtreme Scale を Hyperic HQ モニター・ソフトウェアと統合することで、ご使用の環境のパフォーマンスに関するメトリックをグラフィカルにモニターおよび表示することができます。この統合をセットアップするには、各エージェントでプラグインの実装を使用します。

## 手順

1. eXtreme Scale サーバーを始動します。Hyperic プラグインは、eXtreme Scale を実行している Java 仮想マシンに接続するローカル・プロセスを調べます。Java 仮想マシンに適切に接続する場合は、各サーバーは `-jmxServicePort` オプションを指定して開始する必要があります。 `-jmxServicePort` オプションを指定したサーバーの始動に関して詳しくは、457 ページの『`startOgServer` スクリプト』を参照してください。
2. ご使用の Hyperic の構成で、`extremescale-plugin.xml` ファイルと `wshyperic.jar` ファイルを適切なサーバーとエージェントのプラグイン・ディレクトリに置きます。Hyperic と統合するためには、エージェント・インストールとサーバー・インストールの両方でプラグインおよび Java アーカイブ (JAR) ファイルにアクセスする必要があります。サーバーは構成を動的にスワップできますが、いずれのエージェントを開始する場合もその前に統合を完了しておく必要があります。
  - a. `extremescale-plugin.xml` ファイルを、次のロケーションにあるサーバーの `plugin` ディレクトリに置きます。

```
hyperic_home/server_home/hq-engine/server/default/deploy/hq.ear/hq-plugins
```
  - b. `extremescale-plugin.xml` ファイルを、次のロケーションにあるエージェントの `plugin` ディレクトリに置きます。

```
agent_home/bundles/gent-4.0.2-939/pdk/plugins
```
  - c. `wshyperic.jar` ファイルを次のロケーションにあるエージェントの `lib` ディレクトリに置きます。

```
agent_home/bundles/gent-4.0.2-939/pdk/lib
```
3. エージェントを構成します。 `agent.properties` ファイルは、エージェント・ランタイムの構成ポイントとして機能します。このプロパティは、`agent_home/conf` ディレクトリにあります。以下のキーはオプションですが、eXtreme Scale プラグインに対しては重要です。



- `autoinventory.defaultScan.interval.millis=<time_in_milliseconds>`

エージェント・ディスカバリーの間隔をミリ秒単位に設定します。

- `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

: eXtreme Scale プラグインからの詳細のデバッグ・ステートメントを有効にします。

- `username=<username>`: セキュリティーが有効に設定されている場合に Java Management Extensions (JMX) ユーザー名を設定します。
  - `password=<password>`: セキュリティーが有効に設定されている場合に、JMX パスワードを設定します。
  - `sslEnabled=<true|false>`: プラグインに対して、Secure Sockets Layer (SSL) を使用するかどうかを指示します。デフォルトではこの値は `false` です。
  - `trustPath=<path>`: SSL 接続のトラスト・パスを設定します。
  - `trustType=<type>`: SSL 接続のトラスト・タイプを設定します。
  - `trustPass=<password>`: SSL 接続のトラスト・パスワードを設定します。
4. エージェント・ディスカバリーを開始します。Hyperic エージェントはディスカバリーおよびメトリック情報をサーバーに送ります。サーバーを使用してデータ・ビューをカスタマイズし、論理インベントリー・オブジェクトをグループ化して有用な情報を生成します。サーバーが使用可能になったならば、エージェントの起動スクリプトを実行するか、または Windows サービスを開始する必要があります。

- **Linux** `agent_home/bin/hq-agent.sh start`

- **Windows** Windows サービスを使用してエージェントを開始します。

エージェントの始動後に、サーバーが検出されて、グループが構成されます。サーバー・コンソールにログインし、サーバーのインベントリー・データベースに追加するリソースを選択することができます。サーバー・コンソールは、デフォルトで URL: `http://<server_host_name>:7080/` にあります。

5. Hyperic で統計データを収集するには、統計を使用可能にする必要があります。

eXtreme Scale の Hyperic コンソールで、**SetStatsSpec** 制御アクションを使用します。リソースにナビゲートしてから、「**制御**」タブ付きページの「**制御アクション**」ドロップダウン・リストを使用し、「**制御引数**」テキスト・ボックスに `ALL=enabled` を設定して、`SetStatsSpec` 設定を指定します。

Hyperic コンソールで設定されたフィルターによって、カタログ・サーバーは検出されません。サーバー・プロパティ・ファイルで、**statsSpec** プロパティの情報を参照してください。これにより、コンテナの始動時に統計が使用可能になります。統計を使用可能にする各種オプションについては、542 ページの『管理 Bean (MBean) を使用したモニター』およびサンプル: `xadmin` ユーティリティーを参照してください。

6. Hyperic コンソールでサーバーをモニターします。サーバーがインベントリー・モデルに追加されると、そのサービスはもはや必要なくなります。

- **ダッシュボード・ビュー:** リソース検出イベントを調べたとき、メイン・ダッシュボード・ビューにログインしました。ダッシュボード・ビューは、カスタマイズ可能なメッセージ・センターとなる汎用ビューです。このメイン・ダッシュボードにグラフやインベントリー・オブジェクトをエクスポートすることができます。
  - **リソース・ビュー:** このページからインベントリー・モデル全体を照会して調べることができます。サービスが追加されたならば、サーバー・セクションの下で正しくラベル付けされて一緒にリストされたすべての eXtreme Scale サーバーを確認することができます。個々のサーバーをクリックすると、基本メトリックを参照できます。
7. 「リソース・ビュー」のページでサーバー全体のインベントリーを表示できます。このページで、複数の ObjectGrid サーバーを選択し、それらをグループにまとめることができます。リソースのセットをグループ化すると、共通のメトリックがグラフ化されて、グループ・メンバー間のオーバーレイや相違点が表示されます。オーバーレイを表示するには、ご使用のサーバー・グループの画面でメトリックを選択します。そうすると、メトリックが図表域に表示されます。すべてのグループ・メンバーのオーバーレイを表示するには、下線の付いたメトリック名をクリックします。「ツール」メニューを使用して、必要なグラフ、ノード・ビュー、および比較オーバーレイをメイン・ダッシュボードにエクスポートすることができます。

---

## DB2 内での eXtreme Scale 情報のモニター

バックエンド・データベースとして DB2<sup>®</sup> を使用する JPAloader または JPAEntityLoader を使用する場合、eXtreme Scale 固有の情報を DB2 に渡すことができます。この情報は、DB2 Performance Expert などのパフォーマンス・モニター・ツールで表示でき、データベースにアクセスしている eXtreme Scale アプリケーションをモニターできます。

### 始める前に

トレースを設定するために使用できる各種方式の詳細については、618 ページの『トレースの収集』を参照してください。

### このタスクについて

バックエンド・データベースとして DB2 を使用するようにローダーが構成されているとき、以下の eXtreme Scale 情報をモニターのために DB2 に渡すことができます。

- **ユーザー:** eXtreme Scale で認証を受けるユーザーの名前を指定します。基本認証を使用しない場合は、認証からのプリンシパルを使用します。
- **ワークステーション名:** ホスト名、eXtreme Scale コンテナ・サーバーの IP を指定します。
- **アプリケーション名:** ObjectGrid の名前、パーシスタンス・ユニット名 (設定されている場合) を指定します。
- **アカウント情報:** スレッド ID、トランザクション・タイプ、トランザクション ID、および接続ストリングを指定します。

データベース・アクセスをモニターする方法については、DB2 Performance Expert を参照してください。

## 手順

- すべての eXtreme Scale クライアント情報を使用可能にするには、次のトレース・ストリングを設定します。

```
ObjectGridClientInfo*=event=enabled
```

- ユーザー情報以外をすべて使用可能にするには、次の設定のいずれかを使用します。

```
ObjectGridClientInfo*=event=enabled,ObjectGridClientInfoUser=event=disabled
```

または

```
ObjectGridClientInfo=event=enabled
```

## タスクの結果

トレース機能をオンにすると、DB2 Performance Expert などのパフォーマンス・モニター・ツールにデータが表示されます。

## 例

次の例では、ユーザー bob は eXtreme Scale ユーザーとして認証されています。アプリケーションは、DB2Hibernate パーシスタンス・ユニットを使用して mygrid データ・グリッドにアクセスしています。コンテナ・サーバーの名前は XS\_Server1 です。結果の情報は次のようになります。

- ユーザー=bob
- ワークステーション名=XS\_Server1,192.168.1.101
- アプリケーション名=mygrid,DB2Hibernate
- アカウント情報=1, DEFAULT,FE7954BD-0126-4000-E000-2298094151DB,com.ibm.db2.jcc.t4.b@71787178

次の例の場合、ユーザー bob は WebSphere Application Server トークンを使用して認証されています。アプリケーションは、DB2OpenJPA パーシスタンス・ユニット名を使用して mygrid データ・グリッドにアクセスしています。コンテナ・サーバーの名前は XS\_Server2 です。結果の情報は次のようになります。

- ユーザー  
=acme.principal.UserPrincipal[Bob],acme.principal.  
GroupPrincipal[admin]
- ワークステーション名=XS\_Server2,192.168.1.102
- アプリケーション名=mygrid,DB2OpenJPA
- アカウント情報=188,DEFAULT,FE72BC63-0126-4000-E000-851C092A4E33,com.ibm.ws.rsadapter.jdbc.WSJccSQLJConnection@2b432b43



---

## 第 9 章 パフォーマンスのチューニング



環境の設定をチューニングして、WebSphere eXtreme Scale 環境全体のパフォーマンスを上げることができます。

---

### オペレーティング・システムおよびネットワーク設定のチューニング

ネットワークのチューニングは、接続設定の変更によって伝送制御プロトコル (TCP) スタックの遅延を減らすことができ、TCP バッファーの変更によってスループットを改善することができます。

#### オペレーティング・システム

チューニングが最も少なくてすむのが Windows システムで、最も必要なのが Solaris システムです。以下の説明は、指定された各システムに固有のものであり、WebSphere eXtreme Scale パフォーマンスを向上させる可能性があります。ご使用の環境でのネットワークおよびアプリケーション負荷に応じて調整を行ってください。

#### Windows

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
Tcpip\Parameters
MaxFreeTcbs = dword:00011940
MaxHashTableSize = dword:00010000
MaxUserPort = dword:0000ffff
TcpTimedWaitDelay = dword:0000001e
```

#### Solaris

```
nnd -set /dev/tcp tcp_time_wait_interval 60000
fndd -set /dev/tcp tcp_keepalive_interval 15000
nnd -set /dev/tcp tcp_fin_wait_2_flush_interval 67500
nnd -set /dev/tcp tcp_conn_req_max_q 16384
nnd -set /dev/tcp tcp_conn_req_max_q0 16384
nnd -set /dev/tcp tcp_xmit_hiwat 400000
nnd -set /dev/tcp tcp_recv_hiwat 400000
nnd -set /dev/tcp tcp_cwnd_max 2097152
nnd -set /dev/tcp tcp_ip_abort_interval 20000
nnd -set /dev/tcp tcp_rexmit_interval_initial 4000
nnd -set /dev/tcp tcp_rexmit_interval_max 10000
nnd -set /dev/tcp tcp_rexmit_interval_min 3000
nnd -set /dev/tcp tcp_max_buf 4194304
```

#### AIX

```
/usr/sbin/no -o tcp_sendspace=65536
/usr/sbin/no -o tcp_recvspace=65536
/usr/sbin/no -o udp_sendspace=65536
/usr/sbin/no -o udp_recvspace=65536
/usr/sbin/no -o somaxconn=10000
/usr/sbin/no -o tcp_nodelayack=1
/usr/sbin/no -o tcp_keepinit=40
/usr/sbin/no -o tcp_keepintvl=10
```

#### LINUX

```
sysctl -w net.ipv4.tcp_timestamps=0
sysctl -w net.ipv4.tcp_tw_reuse=1
sysctl -w net.ipv4.tcp_tw_recycle=1
sysctl -w net.ipv4.tcp_fin_timeout=30
```

```
sysctl -w net.ipv4.tcp_keepalive_time=1800
sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
```

## HP-UX

```
ndd -set /dev/tcp tcp_ip_abort_cinterval 20000
```

---

## ORB プロパティ

オブジェクト・リクエスト・ブローカー (ORB) プロパティは、データ・グリッドのトランスポート動作を変更します。これらのプロパティは、`orb.properties` ファイルを使用して設定するか、WebSphere Application Server 管理コンソールで設定として設定するか、または WebSphere Application Server 管理コンソールで ORB のカスタム・プロパティとして設定することができます。

### orb.properties

`orb.properties` ファイルは、`java/jre/lib` ディレクトリにあります。WebSphere Application Server `java/jre/lib` ディレクトリにある `orb.properties` ファイルを変更すると、Java ランタイム環境 (JRE) を使用しているノード・エージェントおよびその他の Java 仮想マシン (JVM) で ORB プロパティが更新されます。この動作を望まない場合は、カスタム・プロパティまたは ORB 設定 WebSphere Application Server 管理コンソールを使用してください。

### デフォルトの WebSphere Application Server 設定

WebSphere Application Server には、デフォルトで、ORB に定義されたプロパティがいくつかあります。これらの設定は、アプリケーション・サーバー・コンテナ・サービスおよびデプロイメント・マネージャーにあります。これらのデフォルト設定は、`orb.properties` ファイルで行われた設定をオーバーライドします。説明されたそれぞれのプロパティについては、「[指定するところ](#)」のセクションを参照して、推奨値を定義する場所を決定してください。

### ファイル記述子設定

UNIX システムおよび Linux システムでは、プロセス当たり許容されるオープン・ファイルの数の制限があります。オペレーティング・システムが、許容されるオープン・ファイルの数を指定します。この値が小さすぎる場合、AIX ではメモリー割り振りエラーが発生し、多すぎるオープン・ファイルはログに記録されます。

UNIX システム端末ウィンドウで、この値をデフォルトのシステム値よりも大きく設定してください。クローンを持つ大容量 SMP マシンの場合、無限に設定してください。

AIX 構成では、コマンド `ulimit -n -1` を使用して、この値を `-1` (無限) に設定してください。

Solaris 構成の場合、コマンド `ulimit -n 16384` を使用して、この値を `16384` に設定してください。

コマンド `ulimit -a` を使用すれば、現行値を表示できます。



## ベースラインの設定

以下の設定は、適切なベースラインですが、必ずしもすべての環境に最適な設定とは限りません。ご使用の環境においてどの値が適切であるか、正しい決定をできるようにこれらの設定をよく理解してください。

```
com.ibm.CORBA.RequestTimeout=30
com.ibm.CORBA.ConnectTimeout=10
com.ibm.CORBA.FragmentTimeout=30
com.ibm.CORBA.LocateRequestTimeout=10
com.ibm.CORBA.ThreadPool.MinimumSize=256
com.ibm.CORBA.ThreadPool.MaximumSize=256
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ConnectionMultiplicity=1
com.ibm.CORBA.MinOpenConnections=1024
com.ibm.CORBA.MaxOpenConnections=1024
com.ibm.CORBA.ServerSocketQueueDepth=1024
com.ibm.CORBA.FragmentSize=0
com.ibm.CORBA.iiop.NoLocalCopies=true
com.ibm.CORBA.NoLocalInterceptors=true
```

## プロパティの説明

### タイムアウト設定

以下の設定は、ORB が要求の操作に見切りをつけるまで待機する時間に関係しています。これらの設定を使用して、異常な状況下で余分なスレッドが作られるのを防いでください。

#### 要求タイムアウト

**プロパティ名:** com.ibm.CORBA.RequestTimeout

**有効な値:** 秒数を表す整数値。

**推奨値:** 30

**指定するところ:** WebSphere Application Server 管理コンソール

**説明:** 要求 (任意) が応答を待機する秒数。その秒数を過ぎると待機を止めます。このプロパティは、ネットワーク停止の障害が発生した場合にクライアントがフェイルオーバーするまでに要する時間に影響します。このプロパティの値を極端に低く設定すると、要求が誤ってタイムアウトになる可能性があります。不用意なタイムアウトを回避するためにこのプロパティの値は慎重に考慮してください。

#### 接続タイムアウト

**プロパティ名:** com.ibm.CORBA.ConnectTimeout

**有効な値:** 秒数を表す整数値。

**推奨値:** 10

**指定するところ:** orb.properties ファイル

**説明:** ソケット接続試行で待機する秒数。その秒数を過ぎると待機を止めます。このプロパティは、要求タイムアウトと同様に、ネットワーク停止の障害が発生した場合にクライアントがフェイルオーバーするまでに要する時

間に影響します。一般に、このプロパティは要求タイムアウト値よりも小さい値に設定します。接続の確立に要する時間は比較的一定であるためです。

#### フラグメント・タイムアウト

**プロパティ名:** com.ibm.CORBA.FragmentTimeout

**有効な値:** 秒数を表す整数値。

**推奨値:** 30

**指定するところ:** orb.properties ファイル

**説明:** フラグメント要求が待機する秒数。その秒数を過ぎると待機を止めます。このプロパティは、要求タイムアウト・プロパティと類似しています。

#### スレッド・プールの設定

このプロパティは、スレッド・プール・サイズを特定のスレッド数に制約します。サーバー要求がソケットで受信されると、そのサーバー要求をスピンオフさせるために、ORB によってスレッドが使用されます。このプロパティ値を低い値に設定すると、ソケットのキュー項目数が増加して、タイムアウトになる可能性もあります。

#### 接続多重度

**プロパティ名:** com.ibm.CORBA.ConnectionMultiplicity

**有効な値:** クライアントとサーバーの間の接続数を表す整数値。デフォルト値は 1 です。これより大きい値に設定すると、複数接続にまたがる多重化の設定になります。

**推奨値:** 1

**指定するところ:** orb.properties ファイル **説明:** ORB が任意のサーバーとの複数の接続を使用できるようにします。理論的に、この値の設定は、複数接続にまたがる並列性を促進します。実際には、接続多重度の設定によるパフォーマンス上の利点はありません。このパラメーターは設定しないでください。

#### オープン接続

**プロパティ名:**

com.ibm.CORBA.MinOpenConnections、com.ibm.CORBA.MaxOpenConnections

**有効な値:** 接続数を表す整数値。

**推奨値:** 1024

**指定するところ:** WebSphere Application Server 管理コンソール **説明:** オープン接続の最小数と最大数。ORB は、クライアントとの間に確立された接続のキャッシュを保持します。この値を超えると、これらの接続は消去されます。接続の消去は、データ・グリッド内の動作の低下の原因になる可能性があります。

#### 成長可能

**プロパティ名:** com.ibm.CORBA.ThreadPool.IsGrowtable

**有効な値:** ブール値。true または false に設定します。

**推奨値:** false

**指定するところ:** orb.properties ファイル **説明:** true に設定すると、ORB が着信要求用に使用するスレッド・プールは、そのプールがサポートする以上に成長する可能性があります。プール・サイズを上回ると、要求の処理のために新規スレッドが作成されますが、そのスレッドはプールされません。値を false に設定することで、スレッド・プールの成長を防ぎます。

#### サーバー・ソケットのキュー項目数

**プロパティ名:** com.ibm.CORBA.ServerSocketQueueDepth

**有効な値:** 接続数を表す整数値。

**推奨値:** 1024

**指定するところ:** orb.properties ファイル **説明:** クライアントからの着呼接続のキューの長さを指定します。ORB は、クライアントからの着呼接続をキューに入れます。キューがフルになると、接続は拒否されます。接続の拒否は、データ・グリッド内の動作の低下の原因になる可能性があります。

#### フラグメント・サイズ

**プロパティ名:** com.ibm.CORBA.FragmentSize

**有効な値:** バイト数を指定する整数。デフォルトは 1024 です。

**推奨値:** 0

**指定するところ:** orb.properties ファイル **説明:** ORB が要求の送信時に使用する最大パケット・サイズを指定します。要求がフラグメント・サイズ制限より大きい場合、その要求は要求フラグメントに分割されて、それぞれ別々に送信されて、サーバー上で再組み立てされます。要求のフラグメント化は、パケットの再送が必要になる可能性のある不安定なネットワークで有効です。ただし、ネットワークの信頼性が高い場合、要求をフラグメントに分割すると、不必要な処理の原因になる可能性があります。

#### ローカル・コピーなし

**プロパティ名:** com.ibm.CORBA.iiop.NoLocalCopies

**有効な値:** ブール値。true または false に設定します。

**推奨値:** true

**指定するところ:** WebSphere Application Server 管理コンソール、「参照による受け渡し」設定 **説明:** ORB が参照による受け渡しをするかどうかを指定します。ORB は、デフォルトで、値の呼び出しによる受け渡しを使用します。インターフェースがローカルで開始されるとき、値の呼び出しによる受け渡しは、パスに余分なガーベッジやシリアライゼーションのコストをもたらす原因になります。この値を true に設定すると、ORB は、値の呼び出しによる受け渡しよりも効率的な参照による受け渡し方式を使用します。

#### ローカル・インターセプターなし

**プロパティ名:** com.ibm.CORBA.NoLocalInterceptors

**有効な値:** ブール値。true または false に設定します。

**推奨値:** true

**指定するところ:** orb.properties ファイル **説明:** ローカル要求 (プロセス内) を行うときにも ORB が要求インターセプターを開始するかどうかを指定します。WebSphere eXtreme Scale が使用するインターセプターは、セキュリティと経路処理を目的とし、要求がプロセス内で処理される場合には必須ではありません。プロセス間を仲介するインターセプターは、リモート・プロシージャ・コール (RPC) 操作の場合にのみ必要です。ローカル・インターセプターなしを設定すると、ローカル・インターセプターを使用することにより生じる余分な処理を回避できます。

**重要:** WebSphere eXtreme Scale セキュリティーを使用している場合は、com.ibm.CORBA.NoLocalInterceptors プロパティ値を false に設定してください。セキュリティ・インフラストラクチャーは、認証のためにインターセプターを使用します。

---

## Java 仮想マシンのチューニング

WebSphere eXtreme Scale の最善のパフォーマンスを得るために、Java 仮想マシン (JVM) チューニングの特定の局面をいくつか考慮してください。ほとんどの場合、特殊な JVM 設定はほとんどまたはまったく必要ありません。データ・グリッドに保管されるオブジェクトが多数ある場合は、ヒープ・サイズを適切なレベルに調整して、メモリー不足の状態で行われるのを避けます。

eXtremeMemory を構成することにより、オブジェクトを Java ヒープでなくネイティブ・メモリーに保管できます。eXtremeMemory を構成すると、新しいトランスポート・メカニズムである eXtremeIO が使用可能になります。オブジェクトを Java ヒープから移動することで、ガーベッジ・コレクションに伴う一時停止を回避でき、より安定したパフォーマンスを得られるうえ、応答時間も予測可能になります。詳しくは、317 ページの『IBM eXtremeMemory および IBM eXtremeIO の構成』を参照してください。

### 試験済みプラットフォーム

パフォーマンス・テストは、まず AIX (32 way)、Linux (4 way)、および Windows (8 way) のコンピューターで実行されました。ハイエンド AIX コンピューターを使用すると、大量のマルチスレッド・シナリオをテストして、競合ポイントを特定して修正できます。

### ガーベッジ・コレクション

WebSphere eXtreme Scale は、要求や応答など各トランザクション、およびログ・シーケンスに関連した一時オブジェクトを作成します。これらのオブジェクトはガーベッジ・コレクションの効率に影響するため、ガーベッジ・コレクションのチューニングは重要です。

最新の JVM はすべてパラレル・ガーベッジ・コレクション・アルゴリズムを使用しています。つまり、より多くのコアを使用することでガーベッジ・コレクションの中断を減らせるようになっています。8 個のコアを使用している物理サーバーは、4 個のコアを使用している物理サーバーよりガーベッジ・コレクションの速度が上がります。

アプリケーションにより区画ごとに大量のデータを管理する必要がある場合は、ガーベッジ・コレクションが要因になっている可能性があります。世代のコレクターが使用されている場合は、大きなヒープ (20 GB 以上) でも、ほとんどが読み取りのシナリオは正常に機能します。ただし、保有ヒープがいっぱいになった後は、コンピュータで使用可能なヒープ・サイズおよびプロセッサ数に比例して一時停止が発生します。この一時停止は、大きいヒープを持つ小さいコンピュータで大きくなる可能性があります。

## Java ガーベッジ・コレクション用の IBM 仮想マシン

IBM の Java 仮想マシンの場合、更新率が高いシナリオ (トランザクションの 100% がエントリーを変更する) には **optavgpause** コレクターを使用してください。データがほとんど更新されない (10% 以下の頻度) ようなシナリオでは、**optavgpause** コレクターより **gencon** コレクターの方がより適切に機能します。両方のコレクターを使用して実験を行い、シナリオで最も適切に機能するコレクターを確認します。詳細ガーベッジ・コレクションをオンにして実行し、ガーベッジの収集に費やされている時間の割合を確認します。チューニングで問題が修正されるまでガーベッジ・コレクションで時間の 80% が費やされたシナリオが発生しました。

ガーベッジ・コレクションのメカニズムを変更するには、**-Xgcpolicy** パラメーターを使用します。**-Xgcpolicy** パラメーターの値は、使用するガーベッジ・コレクターに応じて、**-Xgcpolicy:gencon** または **-Xgcpolicy:optavgpause** に設定できます。

- WebSphere Application Server 構成では、管理コンソールで **-Xgcpolicy** パラメーターを設定します。「サーバー」 > 「アプリケーション・サーバー」 > 「server\_name」 > 「プロセス定義」 > 「Java 仮想マシン」をクリックします。「汎用 JVM 引数」フィールドに、パラメーターを追加します。
- スタンドアロン構成では、**-jvmArgs** パラメーターを **start0gServer** スクリプトに渡して、ガーベッジ・コレクターを指定します。**-jvmArgs** パラメーターは、スクリプトに渡す最後のパラメーターでなければなりません。

## その他のガーベッジ・コレクション・オプション

**重要:** Oracle JVM を使用している場合は、デフォルトのガーベッジ・コレクションの調整とポリシーのチューニングが必要となることがあります。

WebSphere eXtreme Scale は、WebSphere Real Time Java をサポートします。WebSphere Real Time Java を一緒に使用することによって、WebSphere eXtreme Scale のトランザクション処理応答はより一貫性のある、予測可能なものになります。結果として、ガーベッジ・コレクションおよびスレッド・スケジューリングの影響は大幅に小さくなります。応答時間の標準偏差が標準 Java の 10% よりも小さくなる程度まで、影響は少なくなります。

## JVM パフォーマンス

WebSphere eXtreme Scale は、Java Platform, Standard Edition の各種バージョンで稼働します。WebSphere eXtreme Scale は Java SE バージョン 5、Java SE バージョン 6、および Java SE バージョン 7 をサポートします。開発者の生産性およびパフォーマンスを向上させるためには、Java SE バージョン 5、バージョン 6、または Java SE バージョン 7 を使用して、アノテーションおよび改良されたガーベッ



ジ・コレクションを活用してください。WebSphere eXtreme Scale は、32 ビットまたは 64 ビット版の Java 仮想マシンで動作します。

WebSphere eXtreme Scale がテストされたのは、使用可能な仮想マシンの一部ですが、サポートのリストは排他的なものではありません。Edition 5 以降のどのベンダー JVM 上でも WebSphere eXtreme Scale を実行できます。ただし、ベンダー JVM で問題が発生した場合は、その JVM ベンダーにサポートを依頼する必要があります。可能であれば、WebSphere Application Server がサポートするどのプラットフォーム上でも、WebSphere ランタイムの JVM を使用してください。

一般に、最良のパフォーマンスを得るためには Java Platform, Standard Edition の最新バージョンを使用してください。

## ヒープ・サイズ

4 コア当たり 1 JVM で 1 から 2 GB のヒープをお勧めします。最適なヒープ・サイズ値は、次の要因に基づきます。

- ヒープ内のライブ・オブジェクトの数。
- ヒープ内のライブ・オブジェクトの複雑さ。
- JVM 用に使用可能なコアの数。

例えば、10 K バイトの配列を保管するアプリケーションは、POJO の複雑なグラフを使用するアプリケーションよりもずっと大きなヒープを実行できます。

## スレッド数

スレッド数はいくつかの要因に依存します。単一の断片が管理できるスレッド数には制限があります。断片とは区画のインスタンスであり、プライマリーまたはレプリカとすることができます。JVM ごとの断片数が多いほど、それぞれ追加断片を持つスレッドが増えるので、データへの並行パスが多くなります。各断片は可能な限り並行ですが、それでも並行性について制限はあります。

## オブジェクト・リクエスト・ブローカー (ORB) 要件

IBM SDK には、WebSphere Application Server および WebSphere eXtreme Scale を使用してテスト済みの IBM ORB 実装が組み込まれています。サポート・プロセスを簡単にするため、IBM 提供の JVM を使用してください。他の JVM 実装では、異なる ORB が使用されます。IBM ORB は、IBM 提供の Java 仮想マシンと共にしか提供されていません。WebSphere eXtreme Scale には、操作する作業 ORB が必要です。WebSphere eXtreme Scale は、他のベンダーの ORB と一緒に使用できます。ただし、ベンダー ORB で問題が発生した場合は、その ORB ベンダーにサポートを依頼する必要があります。IBM ORB 実装は、サード・パーティーの Java 仮想マシンと互換性があり、必要な場合は置換できます。

## orb.properties のチューニング

研究所では、最大 1500 の JVM のデータ・グリッドで以下のファイルが使用されました。orb.properties ファイルは、ランタイム環境の lib フォルダにあります。



```

# IBM JDK properties for ORB
org.omg.CORBA.ORBClass=com.ibm.CORBA.iop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton

# WS Interceptors
org.omg.PortableInterceptor.ORBInitializerClass=com.ibm.ws.objectgrid.corba.ObjectGridInitializer

# WS ORB & Plugins properties
com.ibm.CORBA.ForceTunnel=never
com.ibm.CORBA.RequestTimeout=10
com.ibm.CORBA.ConnectTimeout=10

# Needed when lots of JVMs connect to the catalog at the same time
com.ibm.CORBA.ServerSocketQueueDepth=2048

# Clients and the catalog server can have sockets open to all JVMs
com.ibm.CORBA.MaxOpenConnections=1016

# Thread Pool for handling incoming requests, 200 threads here
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ThreadPool.MaximumSize=200
com.ibm.CORBA.ThreadPool.MinimumSize=200
com.ibm.CORBA.ThreadPool.InactivityTimeout=180000

# No splitting up large requests/responses in to smaller chunks
com.ibm.CORBA.FragmentSize=0

```

## フェイルオーバー検出のためのハートビート間隔設定のチューニング

ハートビート間隔設定で、障害の起きたサーバーがないかを調べるシステム・チェックの間の時間を構成できます。 **-heartbeat** オプションは **start0gserver** コマンドのパラメーターで、カタログ・サーバーにのみ適用されます。

### このタスクについて

フェイルオーバーの構成は、使用している環境のタイプによって異なります。スタンドアロン環境を使用している場合は、コマンド行でフェイルオーバーを構成できます。WebSphere Application Server Network Deployment 環境を使用している場合は、WebSphere Application Server Network Deployment 管理コンソールでフェイルオーバーを構成する必要があります。

### 手順

- スタンドアロン環境のフェイルオーバーを構成します。

ハートビート間隔は、**start0gServer** スクリプト・ファイルの **-heartbeat** パラメーターを使用してコマンド行で構成できます。このパラメーターは以下のいずれかの値に設定します。

表 34. ハートビート間隔

値	アクション	説明
0	標準 (デフォルト)	通常、30 秒以内にフェイルオーバーが検出されます。
-1	高速	通常、5 秒以内にフェイルオーバーが検出されます。
1	低速	通常、180 秒以内にフェイルオーバーが検出されます。

高速のハートビート間隔は、プロセスおよびネットワークが安定している場合に役立ちます。ネットワークまたはプロセスが最適に構成されていないと、ハートビートを見逃す可能性があり、そうなった場合は誤って障害検出が示されることがあります。

- WebSphere Application Server 環境のフェイルオーバーを構成します。

WebSphere Application Server Network Deployment バージョン 6.0.2 以降は、WebSphere eXtreme Scale のフェイルオーバーを高速で行えるように構成できま

す。ハード障害の場合のデフォルトのフェイルオーバー時間は、約 200 秒です。ハード障害は、物理的なコンピューターまたはサーバーの破損、ネットワーク・ケーブルの切断、オペレーティング・システム・エラーのことです。プロセスの異常終了やソフト障害による障害は、一般的に 1 秒未満でフェイルオーバーされます。ソフト障害の障害検出は、デッド・プロセスのネットワーク・ソケットがそのプロセスをホスティングするサーバーのオペレーティング・システムにより自動的にクローズされるときに発生します。

## コア・グループのハートビート構成

WebSphere Application Server プロセスで実行されている WebSphere eXtreme Scale は、アプリケーション・サーバーのコア・グループ設定のフェイルオーバー特性を継承します。以下のセクションでは、以下のようなさまざまなバージョンの WebSphere Application Server Network Deployment のコア・グループ・ハートビート設定を構成する方法について説明します。

### - WebSphere Application Server Network Deployment バージョン 6.x または 7.x のコア・グループ設定を更新します。

ハートビート間隔は、WebSphere Application Server のバージョン 6.0 からバージョン 6.1.0.12 までは秒単位、バージョン 6.1.0.13 からはミリ秒単位で指定します。また、欠落ハートビートの数も指定する必要があります。この値は、ピア Java 仮想マシン (JVM) に障害が起きたと見なされるまでに、容認される欠落ハートビートの数を示します。ハード障害の検出時間は、ほぼハートビート間隔と欠落ハートビート数の積です。

これらのプロパティは、WebSphere 管理コンソールで、コア・グループに対してカスタム・プロパティを使用して指定します。構成について詳しくは、コア・グループ・カスタム・プロパティを参照してください。アプリケーションによって使用されるすべてのコア・グループに対して、以下のプロパティを指定する必要があります。

- ハートビート間隔は、IBM\_CS\_FD\_PERIOD\_SEC カスタム・プロパティ (秒単位) または IBM\_CS\_FD\_PERIOD\_MILLIS カスタム・プロパティ (ミリ秒単位、バージョン 6.1.0.13 以降が必要) を使用して指定します。
- 欠落ハートビート数は、IBM\_CS\_FD\_CONSECUTIVE\_MISSED カスタム・プロパティを使用して指定します。

IBM\_CS\_FD\_PERIOD\_SEC プロパティのデフォルト値は 20 で、IBM\_CS\_FD\_CONSECUTIVE\_MISSED プロパティのデフォルト値は 10 です。IBM\_CS\_FD\_PERIOD\_MILLIS プロパティを指定すると、設定されている IBM\_CS\_FD\_PERIOD\_SEC カスタム・プロパティがオーバーライドされます。これらのプロパティの値は、正の整数値です。

WebSphere Application Server Network Deployment 6.x サーバーで 1500 ミリ秒の障害検出時間を実現するには、以下の設定を使用します。

- IBM\_CS\_FD\_PERIOD\_MILLIS = 750 を設定 (WebSphere Application Server Network Deployment バージョン 6.1.0.13 以降)
- IBM\_CS\_FD\_CONSECUTIVE\_MISSED = 2 を設定

- **WebSphere Application Server Network Deployment バージョン 7.0** のコア・グループ設定を更新します。

バージョン 7.0 の WebSphere Application Server Network Deployment は、フェイルオーバー検出を増減するために調整できる以下の 2 つのコア・グループ設定を提供します。

- **ハートビート伝送期間。** デフォルト値は 30000 ミリ秒です。
- **ハートビート・タイムアウト期間。** デフォルト値は 180000 ミリ秒です。

これらの設定を変更する方法については、WebSphere Application Server Network Deployment インフォメーション・センター: ディスカバリーおよび障害検出の設定を参照してください。

WebSphere Application Server Network Deployment バージョン 7 サーバーで 1500 ミリ秒の障害検出時間を実現するには、以下の設定を使用します。

- ハートビート伝送期間を 750 ミリ秒に設定します。
- ハートビート・タイムアウト期間を 1500 ミリ秒に設定します。

## 次のタスク

短いフェイルオーバー時間を指定するようにこれらの設定を変更すると、注意すべきシステム・チューニング上の問題が生じます。まず Java はリアルタイム環境ではありません。JVM に長期のガーベッジ・コレクション時間が発生すると、スレッドが遅延する可能性があります。JVM をホスティングするマシンの負荷が大きくなった (JVM 自身またはマシンで実行中の他のプロセスが原因) 場合にも、スレッドが遅延する可能性があります。スレッドが遅延された場合、ハートビートが正確な時間で送信されない可能性があります。最悪の場合、必要なフェイルオーバー時間で遅延が生じる可能性があります。スレッドが遅延すると、誤障害検出が発生します。実動環境で誤障害検出が発生しないように、システムを調整し、サイズ設定する必要があります。これを確実にするには、適切な負荷テストが最善の策です。

注: eXtreme Scale の現行バージョンは、WebSphere Real Time をサポートします。

---

## WebSphere Real Time を使用したガーベッジ・コレクションのチューニング

WebSphere eXtreme Scale を WebSphere Real Time と一緒に使用すると、標準 IBM Java™ SE Runtime Environment (JRE) で採用されているデフォルトのガーベッジ・コレクション・ポリシーと比べてパフォーマンス・スループットは犠牲にしますが、一貫性および予測可能性は高まります。費用対効果の提案が変わる可能性があります。WebSphere eXtreme Scale は、各トランザクションに関連付けられる多数の一時オブジェクトを作成します。これらの一時オブジェクトは、要求、応答、ログ・シーケンス、およびセッションを処理します。WebSphere Real Time がない場合は、トランザクションの応答時間が数百ミリ秒まで増大することがあります。しかし、WebSphere eXtreme Scale のもとで WebSphere Real Time を使用すると、ガーベッジ・コレクションの効率が上がり、応答時間がスタンドアロン構成応答時間の 10% に減少します。

## スタンドアロン環境の WebSphere Real Time

WebSphere eXtreme Scale のもとで WebSphere Real Time を使用することができます。WebSphere Real Time を使用可能にすることにより、ガーベッジ・コレクションはより予測可能になり、スタンドアロン eXtreme Scale 環境でのトランザクションの応答時間とスループットは安定した一貫性のあるものになります。

### WebSphere Real Time の利点

WebSphere eXtreme Scale は、各トランザクションに関連付けられる多数の一時オブジェクトを作成します。これらの一時オブジェクトは、要求、応答、ログ・シーケンス、およびセッションを処理します。WebSphere Real Time がない場合は、トランザクションの応答時間が数百ミリ秒まで増大することがあります。しかし、WebSphere eXtreme Scale のもとで WebSphere Real Time を使用すると、ガーベッジ・コレクションの効率が上がり、応答時間がスタンドアロン構成応答時間の 10% に減少します。

### WebSphere Real Time の使用可能化

Install eXtreme Scale を実行するコンピューターに、WebSphere Real Time とスタンドアロン WebSphere eXtreme Scale をインストールしてください。JAVA\_HOME 環境変数が標準 Java SE Runtime Environment (JRE) を指すように設定してください。

インストールされた WebSphere Real Time をポイントするよう、JAVA\_HOME 環境変数を設定します。その後、次のようにして WebSphere Real Time を使用可能にします。

1. 次の行からコメントを除去することによって、スタンドアロン・インストール objectgridroot/bin/setupCmdLine.sh | .bat ファイルを編集します。

```
WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome  
-Xgc:targetUtilization=80"
```

2. ファイルを保存します。

これで、WebSphere Real Time が使用可能になります。WebSphere Real Time を使用不可にしたい場合は、同じ行にコメントを戻します。

### ベスト・プラクティス

WebSphere Real Time によって、eXtreme Scale トランザクションの応答時間はより予測可能なものになります。その結果、eXtreme Scale トランザクションの応答時間の偏差は、WebSphere Real Time を使用すると、デフォルトのガーベッジ・コレクターを使用する標準 Java と比較して、大幅に改善されます。eXtreme Scale とともに WebSphere Real Time を使用可能にすることは、安定度および応答時間が重要なアプリケーションを使用している場合に最適です。

このセクションで説明するベスト・プラクティスは、WebSphere eXtreme Scale をより効率的にするチューニング方法と、予期される負荷に応じたコード例を示します。

- アプリケーションおよびガーベッジ・コレクター用のプロセッサ使用量を正しく設定する。

WebSphere Real Time にはプロセッサ使用量を制御する機能があり、ガーベッジ・コレクションがアプリケーションに与える影響を制御し、最小化することができます。 `-Xgc:targetUtilization=NN` パラメーターを使用して、20 秒ごとにアプリケーションが使用するプロセッサの NN パーセントを指定します。

WebSphere eXtreme Scale のデフォルトは 80% ですが、それとは異なる値 (例えば 70 など、ガーベッジ・コレクターにより多くのプロセッサ容量を提供する値) を設定するように `objectgridRoot/bin/setupCmdLine.sh` ファイル内のスクリプトを変更することができます。使用するアプリケーションのプロセッサ負荷を 80% 未満に保つことができる十分な数のサーバーをデプロイします。

- ヒープ・メモリーのサイズを大きく設定する。

WebSphere Real Time は正規の Java より多くのメモリーを使用するため、大きいヒープ・メモリーを持つ WebSphere eXtreme Scale を計画して、カタログ・サーバーおよびコンテナを開始する際、`ogStartServer` コマンドの `-jvmArgs -XmxNNNM` パラメーターでヒープ・サイズを設定してください。例えば、`-jvmArgs -Xmx500M` パラメーターを使用してカタログ・サーバーを開始し、適切なメモリー・サイズを使用してコンテナを開始します。メモリー・サイズは、JVM ごとに予想データ・サイズの 60-70% に設定することができます。この値を設定しないと、`OutOfMemoryError` エラーが発生するおそれがあります。さらに、必要ならば、`-jvmArgs -Xgc:noSynchronousGCOnOOM` パラメーターを使用して、JVM がメモリー不足になったときの非決定的振る舞いを回避することができます。

- ガーベッジ・コレクションのスレッドを調整する。

WebSphere eXtreme Scale は、各トランザクションおよびリモート・プロシージャ・コール (RPC) スレッドに関連付けられる多数の一時オブジェクトを作成します。ご使用のコンピューターに十分なプロセッサ・サイクルがある場合は、ガーベッジ・コレクションに対してパフォーマンスが有益に働きます。デフォルトのスレッド数は 1 です。スレッド数は `-Xgcthreads n` 引数によって変更できます。この引数の推奨値は、コンピューターごとの Java 仮想マシン数を考慮して使用可能となるコアの数です。

- WebSphere eXtreme Scale のもとで短時間実行されるアプリケーションのパフォーマンスを調整する。

WebSphere Real Time は、長時間実行するアプリケーション向けに調整されています。通常、信頼できるパフォーマンス・データを取得するには、WebSphere eXtreme Scale のトランザクションを連続して 2 時間実行する必要があります。 `-Xquickstart` パラメーターを使用して、短時間実行アプリケーションのパフォーマンスを最適にすることができます。このパラメーターは、JIT (Just-In-Time) コンパイラーに対して、低レベルの最適化を使用するように指示を出します。

- WebSphere eXtreme Scale クライアント・キューおよび WebSphere eXtreme Scale クライアント・リレーを最小化する。

WebSphere eXtreme Scale を WebSphere Real Time と共に使用する主な利点は、信頼性の高いトランザクション応答時間を実現できることです。通常、トランザクション応答時間の偏差について、桁が数桁も違うような改善が見られます。キューに入れられたクライアント要求、および他のソフトウェアを経由するクライアント要求リレーは応答時間に影響しますが、それは WebSphere Real Time および WebSphere eXtreme Scale の制御範囲外です。スレッドおよびソケットのパラ



メーターを変更して、顕著な遅延のない安定的かつ平滑な負荷を維持し、キュー項目数を減らすようにする必要があります。

- WebSphere Real Time スレッド化を使用する WebSphere eXtreme Scale アプリケーションを開発する。

アプリケーションを変更することなく、信頼性の高い WebSphere eXtreme Scale トランザクション応答時間を実現でき、応答時間の偏差に関して桁が数桁も違うほど改善されます。トランザクションを処理するユーザー・アプリケーションは、通常の Java スレッドではなく、スレッド優先順位およびスケジューリングの制御に優れた `RealtimeThread` を使用することで、スレッド使用の利点をさらに活用できます。

アプリケーションが現在は以下のようなコードを含んでいるとします。

```
public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
```

必要ならば、このコードを次のもので置き換えることができます。

```
public class WXSCacheAppImpl extends RealtimeThread implements  
WXSCacheAppIF
```

## WebSphere Application Server における WebSphere Real Time

WebSphere Application Server Network Deployment バージョン 7.0 環境で eXtreme Scale とともに WebSphere® Real Time を使用することができます。WebSphere Real Time を使用可能にすることにより、ガーベッジ・コレクションはより予測可能になり、トランザクションの応答時間とスループットは安定した一貫性のあるものになります。

### 利点

WebSphere eXtreme Scale を WebSphere Real Time と一緒に使用すると、標準 IBM Java™ SE Runtime Environment (JRE) で採用されているデフォルトのガーベッジ・コレクション・ポリシーと比べてパフォーマンス・スループットは犠牲にしますが、一貫性および予測可能性は高まります。いくつかの基準を基にすると、費用対効果の提案が変わる可能性があります。以下は、主要な基準の一部です。

- サーバーの機能 - 使用可能メモリー、CPU の速度とサイズ、ネットワークの速度と使用
- サーバーの負荷 - 連続的な CPU 負荷、ピークの CPU 負荷
- Java 構成 - ヒープ・サイズ、目標使用、ガーベッジ・コレクション・スレッド
- WebSphere eXtreme Scale コピー・モード構成 - バイト配列と POJO 保管の対比
- アプリケーション特性 - スレッド使用量、応答の要件と許容範囲、オブジェクト・サイズなど。

WebSphere Real Time で使用可能なこのメトロノーム・ガーベッジ・コレクション・ポリシーの他に、標準 IBM Java™ SE Runtime Environment (JRE) で使用可能なオプションのガーベッジ・コレクション・ポリシーがあります。これらのポリシー、`optthruput` (デフォルト)、`gencon`、`optavgpause`、および `subpool` は、特に異なる



アプリケーション要件と環境を解決するように設計されています。これらのポリシーについて詳しくは、564 ページの『Java 仮想マシンのチューニング』を参照してください。アプリケーションと環境の要件、資源と制約に応じて、これらのガーベッジ・コレクション・ポリシーを 1 つ以上プロトタイピングすることにより、確実に要件は満たされ、最適なポリシーを決定することができます。

## WebSphere Application Server Network Deployment との機能

1. 以下はサポートされるバージョンの一部です。
  - WebSphere Application Server Network Deployment バージョン 7.0.0.5 以降。
  - WebSphere Real Time V2 SR2 for Linux 以降。詳しくは、IBM WebSphere Real Time V2 for Linux を参照してください。
  - WebSphere eXtreme Scale バージョン 7.0.0.0 以降。
  - Linux 32 および 64 ビット・オペレーティング・システム。
2. WebSphere eXtreme Scale サーバーは、WebSphere Application Server DMgr と連結することはできません。
3. Real Time は DMgr をサポートしません。
4. Real Time は WebSphere ノード・エージェントをサポートしません。

## WebSphere Real Time の使用可能化

eXtreme Scale を実行するコンピューターに、WebSphere Real Time と WebSphere eXtreme Scale をインストールしてください。WebSphere Real Time Java を SR2 に更新します。

以下のように、WebSphere Application Server バージョン 7.0 コンソールから、各サーバーの JVM 設定を指定できます。

「サーバー」 > 「サーバー・タイプ」 > 「**WebSphere Application Server**」 > <必要なインストール済みサーバー>を選択します。

結果のページで「プロセス定義」を選択します。

次ページで、右側の列最上部の「Java 仮想マシン」をクリックします。(ここで各サーバーのヒープ・サイズ、ガーベッジ・コレクション、およびその他のフラグを設定できます。)

以下のフラグを「汎用 JVM 引数」フィールドに設定します。

```
-Xrealtime -Xgcpolicy:metronome -Xnocompressedrefs -Xgc:targetUtilization=80
```

変更内容を適用し、保存します。

eXtreme Scale サーバーが上記の JVM フラグを組み込んだ WebSphere Application Server 7.0 で Real Time を使用するには、JAVA\_HOME 環境変数を作成する必要があります。

以下のように JAVA\_HOME を設定します。

1. 「環境」を展開します。
2. 「WebSphere 変数」を選択します。

3. 「スコープの表示」の下の「すべてのスコープ」にチェック・マークが付いていることを確認します。
4. ドロップダウン・リストから必要なサーバーを選択します。(DMgr サーバーまたはノード・エージェント・サーバーは選択しないでください。)
5. JAVA\_HOME 環境変数がリストされていない場合は、「新規」を選択し、変数名に JAVA\_HOME を指定します。「値」フィールドに、Real Time への完全修飾パス名を入力します。
6. 変更内容を適用してから保存します。

## ベスト・プラクティス

一連のベスト・プラクティスについては、569 ページの『WebSphere Real Time を使用したガーベッジ・コレクションのチューニング』のベスト・プラクティスのセクションを参照してください。スタンドアロン WebSphere eXtreme Scale 環境に対するベスト・プラクティスのこのリストには、WebSphere Application Server Network Deployment 環境にデプロイする際に注意する、重要な変更がいくつかあります。

追加の JVM コマンド行パラメーターは、前のセクションで指定したガーベッジ・コレクション・ポリシーと同じロケーションに置かなければなりません。

連続的なプロセッサ負荷に対する許容できる初期目標は 50% で、短期間のピークの負荷のヒットは 75% までです。これを超えると、予測可能性と一貫性における低下が測定できるようになる前に、追加キャパシティーを追加しなければなりません。より長い応答時間が許容できれば、パフォーマンスを少し向上させることができます。80% のしきい値を超えると、一貫性と予測可能性において、重大な低下を招くことがあります。

---

## 動的キャッシュ・プロバイダーのチューニング

WebSphere eXtreme Scale 動的キャッシュ・プロバイダーは、パフォーマンスのチューニングのために以下の構成パラメーターをサポートします。

### このタスクについて

- **com.ibm.websphere.xs.dynacache.ignore\_value\_in\_change\_event:** 動的キャッシュ・プロバイダーで変更イベント・リスナーを登録し、ChangeEvent インスタンスを生成すると、値が ChangeEvent 内に入るようにするためのキャッシュ・エントリーのデシリアライズに関連したオーバーヘッドがあります。キャッシュ・インスタンスでこのオプション・パラメーターを true に設定すると、ChangeEvents の生成時にキャッシュ・エントリーのデシリアライゼーションがスキップされます。返される値は、削除操作の場合は NULL で、それ以外はシリアライズされた形式のオブジェクトを含むバイト配列になります。InvalidationEvent インスタンスには、同じようなパフォーマンスに不利な条件があり、これは `com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent` を true に設定することで回避できます。
- **com.ibm.websphere.xs.dynacache.enable\_compression:** デフォルトで、eXtreme Scale 動的キャッシュ・プロバイダーは、メモリー内のキャッシュ・エントリーを圧縮して、キャッシュ密度を上げます。これは、サーブレット・キャッシュなど

のアプリケーションで大幅なメモリーの節約になります。キャッシュ・データのほとんどが圧縮可能でないことが分かっている場合には、この値を `false` に設定することを検討してください。



---

## 第 10 章 セキュリティー



WebSphere eXtreme Scale はデータ・アクセスを保護し、外部セキュリティー・プロバイダーと統合することができます。セキュリティーには、認証、許可、トランスポート・セキュリティー、データ・グリッド・セキュリティー、ローカル・セキュリティー、JMX (Mbean) セキュリティーなどの側面があります。

---

### アプリケーション・クライアントの認証

アプリケーション・クライアントの認証は、クライアント/サーバー・セキュリティーおよび資格情報認証の使用可能化と、オーセンティケーターおよびシステム資格情報生成プログラムの構成からなります。

#### クライアント/サーバー・セキュリティーの使用可能化

ObjectGrid による認証を正常に行うには、クライアントとサーバーの両方でセキュリティーを使用可能にする必要があります。

#### クライアント・セキュリティーの使用可能化

WebSphere eXtreme Scale は、クライアント・プロパティー・サンプル・ファイル (sampleClient.properties ファイル) を WebSphere Application Server インストール済み環境では `was_root/optionalLibraries/ObjectGrid/properties` ディレクトリー内、混合サーバー・インストール済み環境では `/ObjectGrid/properties` ディレクトリー内に提供しています。このテンプレート・ファイルを、適切な値で変更することができます。objectgridClient.properties ファイル内の securityEnabled プロパティーを true に設定してください。securityEnabled プロパティーは、セキュリティーが有効かどうかを示します。クライアントがサーバーに接続されている場合、クライアント・サイドとサーバー・サイドのこの値は、両方とも true か、両方とも false である必要があります。例えば、接続されているサーバーのセキュリティーが有効な場合、クライアントがサーバーに接続するには、このプロパティー値をクライアント・サイドで true に設定する必要があります。

com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration インターフェースは、security.ogclient.props ファイルを表しています。

com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory public API を使用して、このインターフェースのインスタンスをデフォルト値で作成することができます。または ObjectGrid クライアント・セキュリティー・プロパティー・ファイルを渡して、インスタンスを作成することもできます。

security.ogclient.props ファイルには、その他のプロパティーが含まれています。詳しくは、ClientSecurityConfiguration API 資料および ClientSecurityConfigurationFactory API 資料を参照してください。

#### サーバー・セキュリティーの使用可能化

サーバー・サイドでセキュリティーを使用可能にするには、security.xml ファイル内の securityEnabled プロパティーを true に設定します。セキュリティー記述子

XML ファイルを使用してデータ・グリッドのセキュリティー構成を指定し、グリッド全体のセキュリティー構成を非セキュリティー構成から分離します。

## 資格情報認証の使用可能化

eXtreme Scale クライアントが CredentialGenerator オブジェクトを使用して Credential オブジェクトを取得すると、この Credential オブジェクトがクライアント要求とともに eXtreme Scale サーバーに送信されます。サーバーは、要求の処理前に Credential オブジェクトの認証を行います。Credential オブジェクトが正常に認証されると、この Credential オブジェクトを表す Subject オブジェクトが戻されます。その後、この Subject オブジェクトは要求の認証に使用されます。

クライアントおよびサーバーのプロパティー・ファイルで **credentialAuthentication** プロパティーを設定して、資格情報認証を使用可能にします。詳しくは、クライアント・プロパティー・ファイルおよびサーバー・プロパティー・ファイルを参照してください。

以下の 2 つの表に、さまざまな設定で、いずれの認証メカニズムが使用されるかを示します。

表 35. クライアントおよびサーバーの設定における資格情報認証

クライアント資格情報認証	サーバー資格情報認証	結果
いいえ	常になし	使用不可
いいえ	サポートされる	使用不可
いいえ	必須	Error case
サポートされる	常になし	使用不可
サポートされる	サポートされる	使用可能
サポートされる	必須	使用可能
必須	常になし	Error case
必須	サポートされる	使用可能
必須	必須	使用可能

## オーセンティケーターの構成

eXtreme Scale サーバーは、Authenticator プラグインを使用して、Credential オブジェクトの認証を行います。Authenticator インターフェースの実装では、Credential オブジェクトを取得し、Lightweight Directory Access Protocol (LDAP) サーバーなどのユーザー・レジストリーに対してこのオブジェクトを認証します。eXtreme Scale は、レジストリー構成を提供しません。ユーザー・レジストリーへの接続およびユーザー・レジストリーに対する認証は、このプラグインで実装する必要があります。

例えば、1 つの Authenticator 実装では、資格情報からユーザー ID とパスワードが抽出され、このユーザー ID とパスワードを使用して、LDAP サーバーに対する接続と検証が行われます。認証の結果として、Subject オブジェクトが作成されます。この実装では、Java 認証・承認サービス (JAAS) ログイン・モジュールを使用できます。認証の結果として、Subject オブジェクトが戻されます。



以下の例のように、セキュリティー記述子 XML ファイルでオーセンティケーターを構成できます。

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config/security">

  <security securityEnabled="true"
    loginSessionExpirationTime="300">

    <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginAuthenticator">
      </authenticator>

    </security>
</securityConfig>
```

セキュア・サーバーを開始して、セキュリティー XML ファイルを設定する場合は、**-clusterSecurityFile** オプションを使用します。詳しくは、製品概要の Java SE セキュリティーのチュートリアルを参照してください。

## システム資格情報生成プログラムの構成

このシステム資格情報生成プログラムは、システム資格情報のファクトリーを表すために使用されます。システム資格情報は、管理者資格情報に似ています。以下の例のように、カタログ・セキュリティー XML ファイル内で `SystemCredentialGenerator` エレメントを構成できます。

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.
  builtins.UserPasswordCredentialGenerator">
  <property name="properties" type="java.lang.String" value="manager manager1"
    description="username password" />
</systemCredentialGenerator>
```

デモンストレーション用のため、ユーザー名およびパスワードは平文で保管されません。実稼働環境では、ユーザー名およびパスワードは平文で保管しないでください。

WebSphere eXtreme Scale が提供するデフォルトのシステム資格情報生成プログラムは、サーバー資格情報を使用します。システム資格情報生成プログラムを明示的に指定しないと、このデフォルトのシステム資格情報生成プログラムが使用されます。

---

## アプリケーション・クライアントの許可

アプリケーション・クライアントの許可は、ObjectGrid 許可クラス、許可メカニズム、許可検査期間、および作成者限定アクセス許可から構成されます。

eXtreme Scale の許可は Subject オブジェクトおよびアクセス権に基づいています。本製品は、2 種類の許可メカニズム、つまり、Java 認証・承認サービス (JAAS) とカスタム許可をサポートしています。

### ObjectGrid 許可クラス

許可はアクセス権に基づいています。許可クラスには次の 4 つの異なるタイプがあります。

- `MapPermission` クラスは、ObjectGrid マップ内のデータへのアクセスの許可を表します。

- `ObjectGridPermission` クラスは、`ObjectGrid` へのアクセスの許可を表します。
- `ServerMapPermission` クラスは、クライアントからのサーバー・サイドの `ObjectGrid` マップへのアクセスの許可を表します。
- `AgentPermission` クラスは、サーバー・サイドのエージェントを開始する許可を表します。

API および関連許可については、*プログラミング・ガイド*のクライアント許可プログラミングに関するトピックを参照してください。

## 許可検査期間

eXtreme Scale は、パフォーマンス上の理由で、マップ許可検査結果のキャッシングをサポートしています。このメカニズムがないと、特定の許可クラスのメソッドのリストにあるメソッドが呼び出されたときに、ランタイムは、構成された許可メカニズムを呼び出してアクセスを許可します。この許可検査期間が設定されていると、許可メカニズムは、許可検査期間に基づいて定期的に呼び出されます。各許可クラスのメソッドのリストについては、*プログラミング・ガイド*のクライアント許可プログラミングに関するトピックを参照してください。

アクセス権の許可情報は `Subject` オブジェクトに基づいています。クライアントがメソッドにアクセスしようとする、eXtreme Scale ランタイムは、`Subject` オブジェクトに基づいてキャッシュ内を検索します。キャッシュ内でオブジェクトが見つからない場合、ランタイムは、この `Subject` オブジェクトに付与されている許可を確認し、この許可をキャッシュに格納します。

許可検査期間は、`ObjectGrid` が初期化される前に定義しておく必要があります。許可検査期間は、以下の 2 とおりの方法で構成できます。

`ObjectGrid XML` ファイルを使用して `ObjectGrid` を定義し、許可検査期間を設定できます。以下の例では、許可検査期間が 45 秒に設定されています。

```
<objectGrids>
  <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
    authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS"
    permissionCheckPeriod="45">
    <bean id="bean id="TransactionCallback"
      className="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
  </objectGrids>
```

API を使用して `ObjectGrid` を作成する場合、以下のメソッドを呼び出して、許可検査期間を設定します。このメソッドは、`ObjectGrid` インスタンスを初期化する前にのみ呼び出すことができます。このメソッドは、`ObjectGrid` を直接インスタンス化する場合のローカル eXtreme Scale プログラミング・モデルにのみ適用されます。

```
/**
 * This method takes a single parameter indicating how often you
 * want to check the permission used to allow a client access. If the
 * parameter is 0 then every single get/put/update/remove/evict call
 * asks the authorization mechanism, either JAAS authorization or custom
 * authorization, to check if the current subject has permission. This might be
 * prohibitively expensive from a performance point of view depending on
 * the authorization implementation, but if you need to have ever call check the
 * authorization mechanism, then set the parameter to 0.
 * Alternatively, if the parameter is > 0 then it indicates the number
 * of seconds to cache a set of permissions before returning to
 * the authorization mechanism to refresh them. This value provides much
 * better performance, but if the back-end
 * permissions are changed during this time then the ObjectGrid can
 * allow or prevent access even though the back-end security
 * provider was modified.
```

```
*
* @param period the permission check period in seconds.
*/
void setPermissionCheckPeriod(int period);
```

## 作成者限定アクセス許可

作成者限定アクセス許可を使用すると、エントリーを ObjectGrid マップに挿入したユーザーのみ (関連付けられた Principal オブジェクトによって表される) が、そのエントリーにアクセス (read、update、invalidate、および remove) できます。

既存の ObjectGrid マップの許可モデルは、アクセス・タイプに基づいていて、データ・エントリーには基づいていません。すなわち、ユーザーは、read、write、insert、delete、または invalidate などの特定のアクセス・タイプをマップ内のすべてのデータに対して保持しているか、またはどのデータに対しても保持していないかのいずれかです。しかし、eXtreme Scale は、個別のデータ・エントリーに対するユーザーの許可は行いません。この機能は、データ・エントリーに対してユーザーを許可するための新しい方法を提供します。

さまざまなユーザーが異なるデータのセットにアクセスするようなシナリオでは、このモデルが便利です。ユーザーがデータを永続ストアから ObjectGrid マップにロードするときに、永続ストアによってアクセスを許可できます。このケースでは、ObjectGrid マップ層で別の許可を実行する必要がありません。必要な処理は、作成者限定アクセスの機能を使用可能にして、データをマップにロードするユーザーが、確実にそのデータにアクセスできるようにするのみです。

以下の作成者限定モード属性値があります。

### disabled

作成者限定アクセス機能は使用不可になっています。

### complement

作成者限定アクセス機能が使用可能になり、マップ許可を補完します。すなわち、マップ許可と作成者限定アクセスの機能の両方が有効になります。結果、データに対する操作をさらに制限することができます。例えば、作成者はデータを無効化できないようにすることができます。

### supersede

作成者限定アクセス機能が使用可能になり、マップ許可を置き換えます。すなわち、作成者限定アクセス機能がマップ許可に取って代わり、マップ許可は実行されなくなります。

作成者限定アクセス・モードは、次の 2 とおりの方法で構成できます。

### XML ファイルを使用:

以下の例のように、ObjectGrid XML ファイルを使用して ObjectGrid を定義し、作成者限定アクセス・モードを disabled、complement、または supersede のいずれかに設定できます。

```
<objectGrids>
  <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
    accessByCreatorOnlyMode="supersede"
    <bean id="TransactionCallback"
      classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
  ...
</objectGrids>
```

## プログラムで:

ObjectGrid をプログラマチックに作成する場合、以下のメソッドを呼び出して作成者限定アクセス・モードを設定できます。このメソッドの呼び出しは、直接 ObjectGrid インスタンスを生成する場合のローカル eXtreme Scale プログラミング・モデルにのみ適用されます。

```
/**
 * Set the "access by creator only" mode.
 * Enabling "access by creator only" mode ensures that only the user (represented
 * by the Principals associated with it), who inserts the record into the map,
 * can access (read, update, invalidate, and remove) the record.
 * The "access by creator only" mode can be disabled, or can complement the
 * ObjectGrid authorization model, or it can supersede the ObjectGrid
 * authorization model. The default value is disabled:
 * {@link SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED}.
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_COMPLEMENT
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_SUPERSEDE
 *
 * @param accessByCreatorOnlyMode the access by creator mode.
 *
 * @since WAS XD 6.1 FIX3
 */
void setAccessByCreatorOnlyMode(int accessByCreatorOnlyMode);
```

詳細を示すために、ObjectGrid マップ・アカウントがバンキング・グリッドにあり、Manager1 と Employee1 が 2 人のユーザーであるようなシナリオを考えてみます。この場合、eXtreme Scale 許可ポリシーは、「Manager1」に対してはすべてのアクセス権を付与しますが、「Employee1」に対しては読み取りアクセス権しか付与しません。以下の例に示すのは、ObjectGrid マップ許可の JAAS ポリシーです。

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Manager1" {
        permission com.ibm.websphere.objectgrid.security.MapPermission
            "banking.account", "all"
    };
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Employee1" {
        permission com.ibm.websphere.objectgrid.security.MapPermission
            "banking.account", "read, insert"
    };
```

作成者限定アクセスの機能が、どのように許可に影響するか検討してください。

- **disabled:** 作成者限定アクセスの機能が使用不可に設定された場合、マップ許可への影響はありません。ユーザー「Manager1」は、「account」マップ内のすべてのデータにアクセスできます。ユーザー「Employee1」は、マップ内のすべてのデータの read および insert は許可されますが、マップ内のデータに対する update、invalidate、remove はできません。
- **complement:** 「complement」オプションを使用して作成者限定アクセスの機能を使用可能にした場合、マップ許可と作成者限定アクセスの機能の両方が有効になります。ユーザー「Manager1」は、自身が単独でデータをマップにロードした場合のみ、「account」マップ内のデータにアクセスできます。ユーザー「Employee1」は、自身が単独でデータをマップにロードした場合のみ、「account」マップ内のデータを読み取ることができます。(しかし、このユーザーは、マップ内のデータに対する update、invalidate、または remove は許可されません。)
- **supersede:** 「supersede」オプションを使用して作成者限定アクセスの機能を使用可能にした場合、マップ許可は実施されません。許可ポリシーは、作成者限定アクセスの許可のみになります。ユーザー「Manager1」には、「complement」モードの場合と同じ特権が与えられます。このユーザーは、自身がデータをマップに

ロードした場合のみ、「account」マップ内のデータにアクセスできます。しかし、今回はユーザー「Employee1」も、自身がデータをマップにロードすれば、「account」マップ内のデータに対する全アクセス権限が与えられます。つまり、Java 認証・承認サービス (JAAS) ポリシーに定義されている許可ポリシーは実施されません。

---

## データ・グリッドの認証

セキュア・トークン・マネージャー・プラグインを使用すると、サーバー間の認証が可能になります。そのためには、SecureTokenManager インターフェースを実装する必要があります。

generateToken(Object) メソッドは保護されるオブジェクトを取得し、外部に識別されないトークンを生成します。verifyTokens(byte[]) メソッドは逆に、トークンを元のオブジェクトに変換して戻します。

単純な SecureTokenManager 実装は XOR アルゴリズムなど単純なエンコード・アルゴリズムを使用して、オブジェクトをシリアルバイナリ形式でエンコードし、対応するデコード・アルゴリズムを使用してトークンをデコードします。この実装は保護されていないため、簡単に中断されます。

### WebSphere eXtreme Scale デフォルト実装

WebSphere eXtreme Scale には、このインターフェース用のすぐに使用可能な実装が用意されています。このデフォルト実装は、鍵ペアを使用して署名し、署名を検査します。また、秘密鍵を使用してコンテンツを暗号化します。すべてのサーバーには JCKES タイプの鍵ストアが備えられており、鍵ペア、秘密鍵と公開鍵、および秘密鍵が保管されています。鍵ストアは、秘密鍵を保管する JCKES タイプである必要があります。これらの鍵は、送信側で秘密ストリングを暗号化し、署名または検証する場合に使用されます。また、トークンは有効期限の時間に関連付けられています。受信側で、データの検証、暗号化解除、および受信側の秘密ストリングとの比較が行われます。サーバーのペアの間での認証には、Secure Sockets Layer (SSL) 通信プロトコルは必要ありません。これは、秘密鍵と公開鍵の目的が同じであるためです。ただし、サーバー通信が暗号化されていない場合は、通信時に侵入者にデータを盗まれる可能性があります。トークンの有効期限が近い場合、リプレイ・アタックの危険性は少なくなっています。この可能性は、すべてのサーバーをファイアウォールの後ろにデプロイすると、非常に小さくなります。

この方法の欠点は、WebSphere eXtreme Scale 管理者が鍵を生成し、生成した鍵をすべてのサーバーに転送する必要があるため、転送中にセキュリティー・ブリーチ (抜け穴) が発生する可能性があることです。

---

## データ・グリッド・セキュリティー

データ・グリッド・セキュリティーによって、結合サーバーの資格情報が適切であることが保証されるため、悪意のあるサーバーはデータ・グリッドを結合することができません。データ・グリッド・セキュリティーは共有秘密ストリング・メカニズムを使用します。



カタログ・サーバーを含むすべての WebSphere eXtreme Scale サーバーが、共有秘密ストリングと一致しています。サーバーがデータ・グリッドに結合する場合、秘密ストリングの表示を求められます。結合サーバーの秘密ストリングがプレジデント・サーバーまたはカタログ・サーバーのいずれかの秘密ストリングと一致する場合は、結合サーバーは受け入れられます。ストリングが一致しない場合、結合要求は拒否されます。

平文の機密事項の送信は保護されません。WebSphere eXtreme Scale セキュリティー・インフラストラクチャーには、セキュア・トークン・マネージャー・プラグインが用意されており、サーバーはこの機密事項を送信する前にセキュアにできます。セキュア操作の実装方法を決定する必要があります。WebSphere eXtreme Scale は、すぐに使用可能な実装を提供し、これによりセキュア操作が実装され、機密事項が暗号化されて署名が行われます。

秘密ストリングは `server.properties` ファイルに設定されます。`authenticationSecret` プロパティーについての詳細は、サーバー・プロパティー・ファイル を参照してください。

## SecureTokenManager プラグイン

セキュア・トークン・マネージャー・プラグインは、`com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager` インターフェースによって表されます。

SecureTokenManager プラグインについて詳しくは、SecureTokenManager API 資料を参照してください。

`generateToken(Object)` メソッドはオブジェクトを取得し、外部に識別されないトークンを生成します。`verifyTokens(byte[])` メソッドは逆に、トークンを元のオブジェクトに変換して戻します。

単純な SecureTokenManager 実装は、排他 OR (XOR) アルゴリズムなどの単純なエンコード・アルゴリズムを使用して、シリアライズ形式でオブジェクトをエンコードし、対応するデコード・アルゴリズムを使用してトークンをデコードします。この実装は保護されません。

WebSphere eXtreme Scale には、このインターフェースに対してすぐに使用可能な実装が用意されています。

デフォルトの実装では鍵ペアを使用して、署名し、シグニチャーを検査します。また、秘密鍵を使用して、コンテンツを暗号化します。すべてのサーバーには JCKES タイプの鍵ストアが備えられており、鍵ペア、秘密鍵と公開鍵、および秘密鍵が保管されています。鍵ストアは、秘密鍵を保管する JCKES タイプである必要があります。

これらの鍵は、送信側で秘密ストリングを暗号化し、署名または検証する場合に使用されます。また、トークンは有効期限の時間に関連付けられています。受信側で、データの検証、暗号化解除、および受信側の秘密ストリングとの比較が行われます。サーバーのペアの間での認証には、Secure Sockets Layer (SSL) 通信プロトコルは必要ありません。これは、秘密鍵と公開鍵の目的が同じであるためです。ただし、サーバー通信が暗号化されていない場合は、通信時に侵入者にデータを盗まれ



る可能性があります。トークンの有効期限が近いと、リプレイ・アタックの危険性は少なくなっています。この可能性は、すべてのサーバーをファイアウォールの後ろにデプロイすると、非常に小さくなります。

この方法の欠点は、WebSphere eXtreme Scale 管理者が鍵を生成し、生成した鍵をすべてのサーバーに移送する必要があるため、移送中にセキュリティー・ブリーチが発生する可能性があることです。

## セキュア・トークン・マネージャーのデフォルト・プロパティーを作成するためのサンプル・スクリプト

前のセクションで記述したように、署名とシグニチャーの検査を行う鍵ペアおよびコンテンツを暗号化する秘密鍵を含む、鍵ストアを作成することができます。

例えば、以下のように JDK 6 keytool コマンドを使用して鍵を作成できます。

```
keytool -genkeypair -alias keypair1 -keystore key1.jck -storetype JCEKS -keyalg  
rsa -dname "CN=sample.ibm.com, OU=WebSphere eXtreme Scale" -storepass key111 -keypass  
keypair1 -validity 10000
```

```
keytool -genseckey -alias seckey1 -keystore key1.jck -storetype JCEKS -keyalg  
DES -storepass key111 -keypass seckey1 -validity 1000
```

上記 2 つのコマンドは、鍵ペア「keypair1」と秘密鍵「seckey1」を作成します。次に、サーバー・プロパティー・ファイルで以下のように構成することができます。

```
secureTokenKeyStore=key1.jck  
secureTokenKeyStorePassword=key111  
secureTokenKeyStoreType=JCEKS  
secureTokenKeyPairAlias=keypair1  
secureTokenKeyPairPassword=keypair1  
secureTokenSecretKeyAlias=seckey1  
secureTokenSecretKeyPassword=seckey1  
secureTokenCipherAlgorithm=DES  
secureTokenSignAlgorithm=RSA
```

## 構成

セキュア・トークン・マネージャーの構成に使用するプロパティーについては詳しくは、サーバー・プロパティーを参照してください。

---

## トランスポート層セキュリティーおよび Secure Sockets Layer

WebSphere eXtreme Scale は、クライアントとサーバーとの間のセキュア通信に TCP/IP も、Transport Layer Security/Secure Sockets Layer (TLS/SSL) もサポートします。

### 両方向での TLS/SSL の使用可能化

TLS/SSL は、一方向で使用可能に設定されている場合があります。例えば、サーバーの公開証明書はクライアントのトラストストアにインポートされますが、クライアントの公開証明書はサーバーのトラストストアにインポートされません。しかし、WebSphere eXtreme Scale は、データ・グリッド・エージェントを広範囲にわたって使用します。データ・グリッド・エージェントの特性は、サーバーがクライアントに応答を返すとき、新しい接続を作成することです。このとき、eXtreme Scale サーバーはクライアントとして機能します。したがって、クライアントの公開証明書をサーバーのトラストストアにインポートする必要があります。

## Oracle JDK のトランスポート・セキュリティーの使用可能化

WebSphere eXtreme Scale には IBM Java Secure Sockets Extension (IBMJSSE) または IBM Java Secure Sockets Extension 2 (IBMJSSE2) が必要です。IBMJSSE プロバイダーおよび IBMJSSE2 プロバイダーには、SSL プロトコル、Transport Layer Security (TLS) プロトコル、およびアプリケーション・プログラミング・インターフェース (API) フレームワークをサポートするリファレンス実装が含まれています。

Oracle JDK は IBM JSSE プロバイダーおよび IBM JSSE2 プロバイダーを出荷しないため、Oracle JDK のトランスポート・セキュリティーを使用可能にすることはできません。この処理を行うためには、WebSphere Application Server に同梱されている Oracle JDK が必要です。WebSphere Application Server に同梱された Oracle JDK には IBM JSSE プロバイダーおよび IBM JSSE2 プロバイダーが含まれています。

WebSphere eXtreme Scale での IBM 以外の JDK の使用については、332 ページの『カスタム・オブジェクト・リクエスト・ブローカーの構成』を参照してください。-Djava.endorsed.dirs を構成する場合は、objectgridRoot/lib/endorsed ディレクトリーと JRE/lib/endorsed ディレクトリーのどちらもポイントします。ディレクトリー objectgridRoot/lib/endorsed は IBM ORB を使用するために必要で、ディレクトリー JRE/lib/endorsed は、IBM JSSE プロバイダーおよび IBM JSSE2 プロバイダーをロードするために必要です。

SSL 必須プロパティーの構成方法、鍵ストアとトラストストアの作成方法、および WebSphere eXtreme Scale でのセキュア・サーバーの開始方法について詳しくは、「製品概要」のセキュリティー・チュートリアルステップ 4 の作業を行ってください。

## セキュア・トランスポート・タイプの構成

Transport Layer Security (TLS) は、クライアントとサーバーとの間のセキュア通信を可能にします。使用される通信メカニズムは、クライアントおよびサーバーの構成ファイル内に指定された **transportType** パラメーターの値によって決まります。

### このタスクについて

Secure Sockets Layer (SSL) が使用される場合、クライアント・サイドとサーバー・サイドの両方で SSL 構成パラメーターが指定されている必要があります。Java SE 環境では、SSL 構成はクライアントまたはサーバーのプロパティー・ファイル内で構成されます。クライアントまたはサーバーが WebSphere Application Server 内にある場合は、コンテナ・サーバーおよびクライアント用の既存の WebSphere Application Server CSIV2 トランスポート設定を使用できます。詳しくは、596 ページの『WebSphere Application Server とのセキュリティー統合』を参照してください。

表 36. クライアント・トランスポートおよびサーバー・トランスポートの設定で使用されるトランスポート・プロトコル：

transportType 設定がクライアントとサーバーで異なる場合、結果のプロトコルは別のものになるかエラーになる可能性があります。

クライアントの transportType プロパティ	サーバーの transportType プロパティ	結果のプロトコル
TCP/IP	TCP/IP	TCP/IP
TCP/IP	SSL サポート	TCP/IP
TCP/IP	SSL 必須	エラー
SSL サポート	TCP/IP	TCP/IP
SSL サポート	SSL サポート	SSL (SSL が失敗した場合は TCP/IP)
SSL サポート	SSL 必須	SSL
SSL 必須	TCP/IP	エラー
SSL 必須	SSL サポート	SSL
SSL 必須	SSL 必須	SSL

## 手順

1. クライアント・セキュリティ構成に **transportType** プロパティを設定する方法については、クライアント・プロパティ・ファイルを参照してください。
2. コンテナおよびカタログ・サーバー・セキュリティ構成に **transportType** プロパティを設定する方法については、サーバー・プロパティ・ファイルを参照してください。

## クライアントまたはサーバーの Secure Sockets Layer (SSL) パラメーターの構成

クライアントとサーバーでは、SSL パラメーターの構成方法は異なります。

### このタスクについて

TLS/SSL は、一方向で使用可能に設定されている場合があります。例えば、サーバーの公開証明書はクライアントのトラストストアにインポートされますが、クライアントの公開証明書はサーバーのトラストストアにインポートされません。しかし、WebSphere eXtreme Scale は、データ・グリッド・エージェントを広範囲にわたって使用します。データ・グリッド・エージェントの特性は、サーバーがクライアントに応答を返すとき、接続を作成することです。このとき、eXtreme Scale サーバーはクライアントとして機能します。したがって、クライアントの公開証明書をサーバーのトラストストアにインポートする必要があります。

## 手順

- クライアント SSL パラメーターを構成します。

次のいずれかのオプションを使用して、クライアント上に SSL パラメーターを構成します。

- `com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory` ファクトリー・クラスを使用して、`com.ibm.websphere.objectgrid.security.config.SSLConfiguration` オブジェクトを作成します。
- `client.properties` ファイル内のパラメーターを構成します。その後、このプロパティ・ファイルを JVM クライアント・プロパティとして設定するこ

ともできれば、WebSphere eXtreme Scale API を使用することもできます。プロパティ・ファイルをクライアントの `ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String)` メソッドに渡し、返されたオブジェクトを `ObjectGridManager.connect(String, ClientSecurityConfiguration, URL)` メソッドのパラメーターとして使用します。

- サーバー SSL パラメーターを構成します。

サーバー用の SSL パラメーターは、`server.properties` ファイルを使用して構成されます。特定のプロパティ・ファイルを使用してコンテナ・サーバーまたはカタログ・サーバーを始動するには、`startOgServer` スクリプトの `-serverProps` パラメーターを使用します。eXtreme Scale サーバー用に設定できる SSL パラメーターについて詳しくは、セキュリティ・サーバー・プロパティを参照してください。

---

## Java Management Extensions (JMX) セキュリティー

分散環境での Managed Bean (MBean) 呼び出しを保護することができます。

使用可能な MBean について詳しくは、497 ページの『Managed Beans (MBeans) を使用した管理』を参照してください。

分散デプロイメント・トポロジーでは、MBean は、カタログ・サーバーおよびコンテナ・サーバーで直接ホストされます。一般に、分散トポロジーの JMX セキュリティーは、Java Management Extensions (JMX) 仕様に指定された JMX セキュリティー仕様に従います。これは、以下の 3 つのパートで構成されます。

1. 認証: リモート・クライアントは、コネクター・サーバー内で認証される必要があります。
2. アクセス制御: MBean アクセス制御は、MBean 情報にアクセスできるユーザー、および MBean 操作を実行できるユーザーを制限します。
3. セキュア・トランスポート: JMX クライアントとサーバー間のトランスポートは、TLS/SSL を使用して保護することができます。

### 認証

JMX では、コネクター・サーバーがリモート・クライアントを認証するメソッドを提供しています。RMI コネクターの場合、認証は、コネクター・サーバーが作成される場合に `JMXAuthenticator` インターフェースを実装するオブジェクトを提供することにより実行されます。eXtreme Scale は、この `JMXAuthenticator` インターフェースを実装し、`ObjectGrid Authenticator` プラグインを使用してリモート・クライアントを認証します。eXtreme Scale がどのようにしてクライアントを認証するかについて詳しくは、75 ページの『Java SE セキュリティー・チュートリアル - ステップ 2』を参照してください。

JMX クライアントは、JMX API に従って、コネクター・サーバーに接続するための資格情報を提供します。JMX フレームワークは、資格情報をコネクター・サーバーに渡し、認証のため、`JMXAuthenticator` 実装を呼び出します。前述のように、`JMXAuthenticator` 実装は、`ObjectGrid Authenticator` 実装に認証を委任します。

以下の例は、資格情報を使用してコネクタ・サーバーに接続する方法を説明していますので、参考にしてください。

```
javax.management.remote.JMXServiceURL jmxUrl = new JMXServiceURL(
    "service:jmx:rmi:///jndi/rmi://localhost:1099/objectgrid/MBeanServer");

environment.put(JMXConnector.CREDENTIALS, new UserPasswordCredential("admin", "xxxxxx"));

// Create the JMXConnectorServer
JMXConnector cntor = JMXConnectorFactory.newJMXConnector(jmxUrl, null);

// Connect and invoke an operation on the remote MBeanServer
cntor.connect(environment);
```

上記の例では、`UserPasswordCredential` オブジェクトが、ユーザー ID が `admin` に、パスワードが `xxxxx` に設定されて提供されます。この `UserPasswordCredential` オブジェクトは、環境マップに設定され、これは、`JMXConnector.connect(Map)` メソッドで使用されます。次に、この `UserPasswordCredential` オブジェクトは、JMX フレームワークによってサーバーに渡され、最終的に認証のために `ObjectGrid` 認証フレームワークに渡されます。

クライアント・プログラミング・モデルは、厳格に JMX 仕様に従います。

## アクセス制御

JMX MBean サーバーは、機密情報に対するアクセス権を持つことがあり、機密操作を実行することができる場合があります。JMX では、どのクライアントがその情報にアクセスでき、どのユーザーがそうした操作を実行できるかを識別する、必要なアクセス制御を提供しています。アクセス制御は、標準 Java セキュリティー・モデルに基づいて、MBean サーバーおよびその操作へのアクセスを制御する許可を定義することによって構築されます。

JMX 操作のアクセス制御または許可に関して、eXtreme Scale は、JMX 実装で提供される JAAS サポートに依存しています。プログラム実行の任意の時点で、実行のスレッドが保持する現行の許可セットがあります。このようなスレッドが JMX 仕様操作を呼び出すとき、これらの許可は保持許可として知られています。JMX 操作が実行されると、セキュリティ・チェックが行われ、必要な許可が保持許可によって暗黙的に示されているかどうかチェックされます。

MBean ポリシー定義は、Java ポリシー形式に従います。例えば、次のポリシーでは、すべての署名者およびすべてのコード・ベースに `PlacementServiceMBean` のサーバー JMX アドレスを取得する権限を付与します。ただし、これらの署名者およびコード・ベースは `com.ibm.websphere.objectgrid` ドメインに制限されます。

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
        [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

以下のポリシー・サンプルを使用すれば、リモート・クライアント ID に基づく許可を完了することができます。このポリシーでは、前の例に示されたものと同じ MBean 許可を付与していますが、X500Principal 名が `CN=Administrator, OU=software, O=IBM, L=Rochester, ST=MN, C=US` のユーザーだけは除きます。

```
grant principal javax.security.auth.x500.X500Principal "CN=Administrator,OU=software,O=IBM,
L=Rochester,ST=MN,C=US" {permission javax.management.MBeanPermission
"com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
[com.ibm.websphere.objectgrid:*,type=PlacementService]",
"invoke";
}
```

Java ポリシーは、セキュリティー・マネージャーがオンになっている場合に限ってチェックされます。-Djava.security.manager JVM 引数を設定してカタログ・サーバーおよびコンテナ・サーバーを始動し、MBean 操作のアクセス制御を強制するようにしてください。

## セキュア・トランスポート

JMX クライアントと JMX サーバー間のトランスポートは、TLS/SSL を使用して保護することができます。カタログ・サーバーまたはコンテナ・サーバーの transportType が SSL\_Required または SSL\_Supported に設定されている場合、SSL を使用して JMX サーバーに接続する必要があります。

SSL を使用するには、-D システム・プロパティを使用して、トラストストア、トラストストア・タイプ、およびトラストストア・パスワードを MBean クライアントで構成する必要があります。

1. -Djavax.net.ssl.trustStore=TRUST\_STORE\_LOCATION
2. -Djavax.net.ssl.trustStorePassword=TRUST\_STORE\_PASSWORD
3. -Djavax.net.ssl.trustStoreType=TRUST\_STORE\_TYPE

java\_home/jre/lib/security/java.security ファイルで SSL ソケット・ファクトリーとして com.ibm.websphere.ssl.protocol.SSLSocketFactory を使用する場合は、次のプロパティを使用します。

1. -Dcom.ibm.ssl.trustStore=TRUST\_STORE\_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST\_STORE\_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST\_STORE\_TYPE

スタンドアロン構成で Transport Layer Security/Secure Sockets Layer (TLS/SSL) が有効であるとき、この情報を取得するには、JMX サービス・ポートを設定してカタログ・サーバーおよびコンテナ・サーバーを始動する必要があります。以下のいずれかの方法を使用して、JMX サービス・ポートを設定します。

- **startOgServer** スクリプトで **-JMXServicePort** オプションを使用します。
- 組み込みサーバーを使用している場合は、ServerProperties インターフェースの setJMXServicePort メソッドを呼び出して、JMX サービス・ポートを設定します。

カタログ・サーバー上の JMX サービス・ポートのデフォルト値は 1099 です。構成の中の各 JVM に対して、異なるポート番号を使用しなければなりません。JMX/RMI を使用する場合は、たとえデフォルトのポート値を使用する場合であっても、**-JMXServicePort** オプションとポート番号を明示的に指定してください。

カタログ・サーバーからコンテナ・サーバー情報を表示するときは、JMX サービス・ポートを設定する必要があります。例えば、このポートは、**xscmd -c showMapSizes** コマンドを使用する場合に必要となります。



JMX コネクター・ポートを設定して、一時ポートが作成されないようにします。以下のいずれかの方法を使用して、JMX コネクター・ポートを設定します。

- **startOgServer** スクリプトで **-JMXConnectorPort** オプションを使用します。
- 組み込みサーバーを使用している場合は、ServerProperties インターフェースの `setJMVConnectorPort` メソッドを呼び出します。

---

## 外部プロバイダーとのセキュリティー統合

データを保護するために、いくつかのセキュリティー・プロバイダーと製品を統合できます。

WebSphere eXtreme Scale は、外部のセキュリティー実装と統合できます。この外部実装は、WebSphere eXtreme Scale に認証サービスおよび許可サービスを提供する必要があります。WebSphere eXtreme Scale には、セキュリティー実装と統合するためのプラグイン・ポイントがあります。WebSphere eXtreme Scale は、以下のコンポーネントと正常に統合されています。

- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- ObjectGrid セキュリティー
- Tivoli Access Manager
- Java 認証・承認サービス (JAAS)

eXtreme Scale は、以下のタスクにセキュリティー・プロバイダーを使用します。

- クライアントをサーバーに認証する。
- クライアントに対して、特定の eXtreme Scale 成果物へアクセスする権限、または eXtreme Scale 成果物に対して行うことができる操作を指定する権限を与える。

eXtreme Scale には、以下のタイプの許可があります。

### マップ許可

クライアントまたはグループに、マップに対する挿入、読み取り、更新、除去、および削除の操作を許可することができます。

### ObjectGrid 許可

クライアントまたはグループに、objectGrid でオブジェクト照会またはエンティティー照会を実行する許可を与えることができます。

### DataGrid エージェント許可

クライアントまたはグループに、DataGrid エージェントの ObjectGrid へのデプロイを許可することができます。

### サーバー・サイド・マップ許可

クライアントまたはグループに、サーバー・マップをクライアント・サイドに複製すること、またはサーバー・マップに動的索引を作成することを許可できます。

### 管理許可

クライアントまたはグループに、管理タスク実行の許可を与えることができます。

**注:** バックエンドに対して既にセキュリティーを有効にしている場合、こうしたセキュリティー設定ではもはや十分にデータを保護できないことに注意してください。データベースまたは他のデータ・ストアのセキュリティー設定は、決してキャッシュに転送されません。認証、許可、トランスポートのレベルのセキュリティーなど、eXtreme Scale のセキュリティー・メカニズムを使用して、現在キャッシュされているデータを個別に保護する必要があります。

**制約事項:** バージョン 1.6 以降で、WebSphere eXtreme Scale のスタンドアロン構成で SSL トランスポート層セキュリティーも使用している場合は、開発キットやランタイム環境を使用しないでください。バージョン 1.6 以降は、WebSphere eXtreme Scale バージョン 7.1 アプリケーション・プログラミング・インターフェースをサポートしません。スタンドアロンの eXtreme Scale インストール済み環境用の SSL トランスポート・セキュリティーを必要とする構成には、1.5 以前のバージョンを使用してください。この制限は、eXtreme Scale のスタンドアロン構成で SSL セキュリティーを使用している場合のみ適用されます。非 SSL のトランスポート構成においては、バージョン 1.6 はサポートされています。

---

## REST データ・サービスの保護

REST データ・サービスの複数の側面を保護します。認証および許可を使用して、eXtreme Scale REST データ・サービスへのアクセスを保護できます。また、アクセス規則として知られるサービス・スコープ構成規則によって、アクセスを制御することもできます。3 番目のセキュリティーとして、トランスポート・セキュリティーを考慮します。

### このタスクについて

認証および許可を使用して、eXtreme Scale REST データ・サービスへのアクセスを保護できます。認証および許可は、eXtreme Scale セキュリティーとの統合を伴います。

また、アクセス規則として知られるサービス・スコープ構成規則によってアクセスを制御することもできます。2 つのタイプのアクセス規則が存在します。1 つはサービスによって許可される CRUD 操作を制御するサービス操作アクセス権限で、もう 1 つは特定のエンティティー・タイプに対して許可される CRUD 操作を制御するエンティティー・アクセス権限です。

トランスポート・セキュリティーは、Web クライアントと REST サービス間の接続に対しては、ホスティングしているコンテナ構成によって提供されます。また、トランスポート・セキュリティーは、(REST サービスから eXtreme Scale データ・グリッドへの接続に対して) eXtreme Scale クライアント構成によっても提供されます。

### 手順

- 認証および許可を制御します。

認証および許可を使用して、eXtreme Scale REST データ・サービスへのアクセスを保護できます。認証および許可は、eXtreme Scale セキュリティーとの統合を伴います。

eXtreme Scale REST データ・サービスは、認証および許可のために eXtreme Scale セキュリティーを使用して、サービスにアクセスできるユーザーやユーザーがサービス経由で実行を許可される操作を制御します。 eXtreme Scale REST データ・サービスは、構成済みグローバル資格情報 (ユーザーとパスワード) を使用するか、各トランザクションで、認証および許可が実行される eXtreme Scale データ・グリッドに送信される HTTP BASIC チャレンジから得た資格情報を使用します。

1. グリッドで、eXtreme Scale クライアント認証および許可を構成します。  
eXtreme Scale クライアント認証および許可を構成する方法の詳細については、591 ページの『外部プロバイダーとのセキュリティ統合』を参照してください。
2. REST サービスによって使用される eXtreme Scale クライアントのセキュリティを構成します。

eXtreme Scale REST データ・サービスは、eXtreme Scale グリッドとの通信時に eXtreme Scale クライアント・ライブラリーを呼び出します。そのため、eXtreme Scale クライアントで eXtreme Scale セキュリティーを構成する必要があります。

eXtreme Scale クライアント認証は、objectgrid クライアント・プロパティー・ファイル内のプロパティーで使用可能にします。 REST サービスでクライアント・セキュリティを使用する場合には、最低でも以下の属性を使用可能にする必要があります。

```
securityEnabled=true  
credentialAuthentication=Supported [-or-] Required  
credentialGeneratorProps=user:pass [-or-] {xor encoded user:pass}
```

**要確認:** credentialGeneratorProps プロパティーに指定するユーザーとパスワードは、認証レジストリー内の ID にマップできなければなりません。また、ObjectGrid に接続したり ObjectGrid を作成したりできる十分な ObjectGrid ポリシー権限を備えている必要があります。

サンプル ObjectGrid クライアント・ポリシー・ファイルは `restservice_home/security/security.ogclient.properties` にあります。クライアント・プロパティー・ファイルも参照してください。

3. eXtreme Scale REST データ・サービスでセキュリティを構成します。

eXtreme Scale セキュリティーと統合するには、eXtreme Scale REST データ・サービス構成プロパティー・ファイルには、以下の項目が含まれている必要があります。

```
ogClientPropertyFile=file_name
```

ogClientPropertyFile は、前のステップで言及した ObjectGrid クライアント・プロパティーが入っているプロパティー・ファイルの場所です。 REST サービスはこのファイルを使用して、セキュリティが使用可能になっている場合にグリッドに通信する eXtreme Scale クライアントを初期設定します。

```
loginType=basic [-or-] none
```

loginType プロパティーは、REST サービスでログイン・タイプを構成します。値 none を指定すると、credentialGeneratorProps で定義された「グローバ

ル」ユーザー ID とパスワードが、各トランザクションでグリッドに送信されます。値 `basic` を指定すると、REST サービスは HTTP BASIC チャレンジをクライアントに提示して、グリッドとの通信時に各トランザクションで送信する資格情報を要求します。

`ogClientPropertyFile` プロパティおよび `loginType` プロパティの詳細については、REST データ・サービスのプロパティ・ファイルを参照してください。

- アクセス規則を適用します。

また、アクセス規則として知られるサービス・スコープ構成規則によってアクセスを制御することもできます。2つのタイプのアクセス規則が存在します。1つはサービスによって許可される CRUD 操作を制御するサービス操作アクセス権限で、もう1つは特定のエンティティ・タイプに対して許可される CRUD 操作を制御するエンティティ・アクセス権限です。

eXtreme Scale REST データ・サービスでは、オプションとして、サービスおよびサービス内のエンティティに対するアクセスを制限するために構成可能なアクセス規則を使用できます。これらのアクセス規則は、REST サービスのアクセス権限プロパティ・ファイルで指定します。このファイルの名前は、REST データ・サービスのプロパティ・ファイル内で、`wxsRestAccessRightsFile` プロパティを使用して指定します。このプロパティについては詳しくは、REST データ・サービスのプロパティ・ファイルを参照してください。このファイルは通常、キーと値のペアが含まれた Java プロパティ・ファイルです。2つのタイプのアクセス規則が存在します。1つはサービスによって許可される CRUD 操作を制御するサービス操作アクセス権限で、もう1つは特定のエンティティ・タイプに対して許可される CRUD 操作を制御するエンティティ・アクセス権限です。

1. サービス操作権限を構成します。

サービス操作権限では、REST サービスで公開するすべての `ObjectGrid` または指定した個別 `ObjectGrid` のすべてのエンティティに適用するアクセス権限を指定します。

以下の構文を使用します。

```
serviceOperationRights=service_operation_right  
serviceOperationRights.grid_name -OR- *=service_operation_right
```

各部の意味は、次のとおりです。

- `serviceOperationRights` には、`NONE`、`READSINGLE`、`READMULTIPLE`、`ALLREAD`、`ALL` のいずれかを指定できます。
- `serviceOperationRights.grid_name -OR- *` は、アクセス権限がすべての `ObjectGrid` に適用されることを暗黙指定します。また、特定の `ObjectGrid` の名前を指定することもできます。

例:

```
serviceOperationsRights=ALL  
serviceOperationsRights.*=NONE  
serviceOperationsRights.EMPLOYEEGRID=READSINGLE
```

最初の例では、この REST サービスで公開されるすべての ObjectGrid ですべてのサービス操作を許可することを指定しています。2 番目の例では、最初の例と同様に、REST サービスによって公開されるすべての ObjectGrid に適用しています。ただし、アクセス権限を NONE として指定しており、ObjectGrid ではどのサービス操作も許可されません。最後の例では、特定のグリッドのサービス操作を制御する方法を示しています。この例では、EMPLOYEEGRID のすべてのエンティティに対して、結果が単一のレコードになる読み取りのみが許可されます。

REST サービスで想定されるデフォルトは serviceOperationsRights=ALL であり、このサービスで公開されるすべての ObjectGrid ですべての操作が許可されます。これは、デフォルトが NONE で、REST サービスでどの操作も許可されない Microsoft の実装とは異なります。

**重要:** サービス操作権限は、このファイルで指定された順序で評価されます。そのため、最後に指定された権限によって、それより前の権限がオーバーライドされます。

## 2. エンティティ・アクセス権限を構成します。

エンティティ・セット権限は、REST サービスで公開される特定の ObjectGrid エンティティに適用するアクセス権限を指定します。この権限により、サービス操作権限と比較して、個別 ObjectGrid エンティティに対するアクセス制御を厳格化および詳細化できます。

以下の構文を使用します。

```
entitySetRights.grid_name.entity_name=entity_set_right
```

各部の意味は、次のとおりです。

– *entity\_set\_right* には、以下の権限のいずれかを指定できます。

表 37. エンティティ・アクセス権限： サポートされる値。

アクセス権限	説明
NONE	データにアクセスするためのすべての権限を拒否します。
READSINGLE	単一のデータ項目の読み取りを許可します。
READMULTIPLE	データ・セットの読み取りを許可します。
ALLREAD	単一データ/複数のデータ・セットの読み取りを許可します。
WRITEAPPEND	データ・セットでの新規データ項目の作成を許可します。
WRITEREPLACE	データの置換を許可します。
WRITEDELETE	データ・セットからのデータ項目の削除を許可します。
WRITEMERGE	データのマージを許可します。
ALLWRITE	データの書き込み (つまり、作成、置換、マージ、削除) を許可します。
ALL	データの作成、読み取り、更新、および削除を許可します。

- *entity\_name* は、REST サービス内の特定の ObjectGrid の名前です。
- *grid\_name* は、指定した ObjectGrid 内の特定のエンティティの名前です。



注: それぞれの ObjectGrid およびそのエンティティに対してサービス操作権限とエンティティ・セット権限の両方を指定した場合は、以下の例に示すように、制限の厳しい方の権限が適用されます。なお、エンティティ・セット権限は、ファイル内で指定された順序で評価されます。最後に指定された権限によって、それより前の権限がオーバーライドされます。

**例 1:** serviceOperationsRights.NorthwindGrid=READSINGLE と entitySetRights.NorthwindGrid.Customer=ALL を指定した場合。Customer エンティティには、READSINGLE が適用されます。

**例 2:** serviceOperationsRights.NorthwindGrid=ALLREAD を指定し、entitySetRights.NorthwindGrid.Customer=ALLWRITE を指定すると、NorthwindGrid のすべてのエンティティに対して、読み取りのみが許可されます。ただし、Customer に対しては、(ALLWRITE が指定されているため) エンティティ・セット権限によってすべての読み取りが拒否されます。そのため、実質上、Customer エンティティのアクセス権限は NONE になります。

- トランSPORTを保護します。

トランSPORT・セキュリティは、Web クライアントと REST サービス間の接続に対しては、ホスティングしているコンテナ構成によって提供されます。REST サービスと eXtreme Scale グリッド間の接続に対しては、eXtreme Scale クライアント構成によってトランSPORT・セキュリティが提供されます。

1. クライアントおよび REST サービスからの接続を保護します。この接続のトランSPORT・セキュリティは、eXtreme Scale ではなくホスティング・コンテナ環境によって提供されます。
2. REST サービスおよび eXtreme Scale グリッドからの接続を保護します。この接続のトランSPORT・セキュリティは、eXtreme Scale で構成します。585 ページの『トランSPORT層セキュリティおよび Secure Sockets Layer』を参照してください。

---

## WebSphere Application Server とのセキュリティ統合

WebSphere eXtreme Scale が WebSphere Application Server 環境にデプロイされている場合、WebSphere Application Server からの認証フローおよびトランSPORT層セキュリティ構成を簡略化できます。

### 簡略化された認証フロー

eXtreme Scale クライアントおよびサーバーが WebSphere Application Server および同じセキュリティ・ドメインで稼働中の場合、WebSphere Application Server セキュリティ・インフラストラクチャーを使用して、クライアント認証資格情報を eXtreme Scale サーバーに伝搬することができます。例えば、サーブレットが eXtreme Scale クライアントとして動作して、同じセキュリティ・ドメインの eXtreme Scale サーバーに接続し、そのサーブレットが既に認証されている場合、認証トークンをクライアント (サーブレット) からサーバーに伝搬し、その後、WebSphere Application Server セキュリティ・インフラストラクチャーを使用して、認証トークンを元のクライアント資格情報に変換することができます。



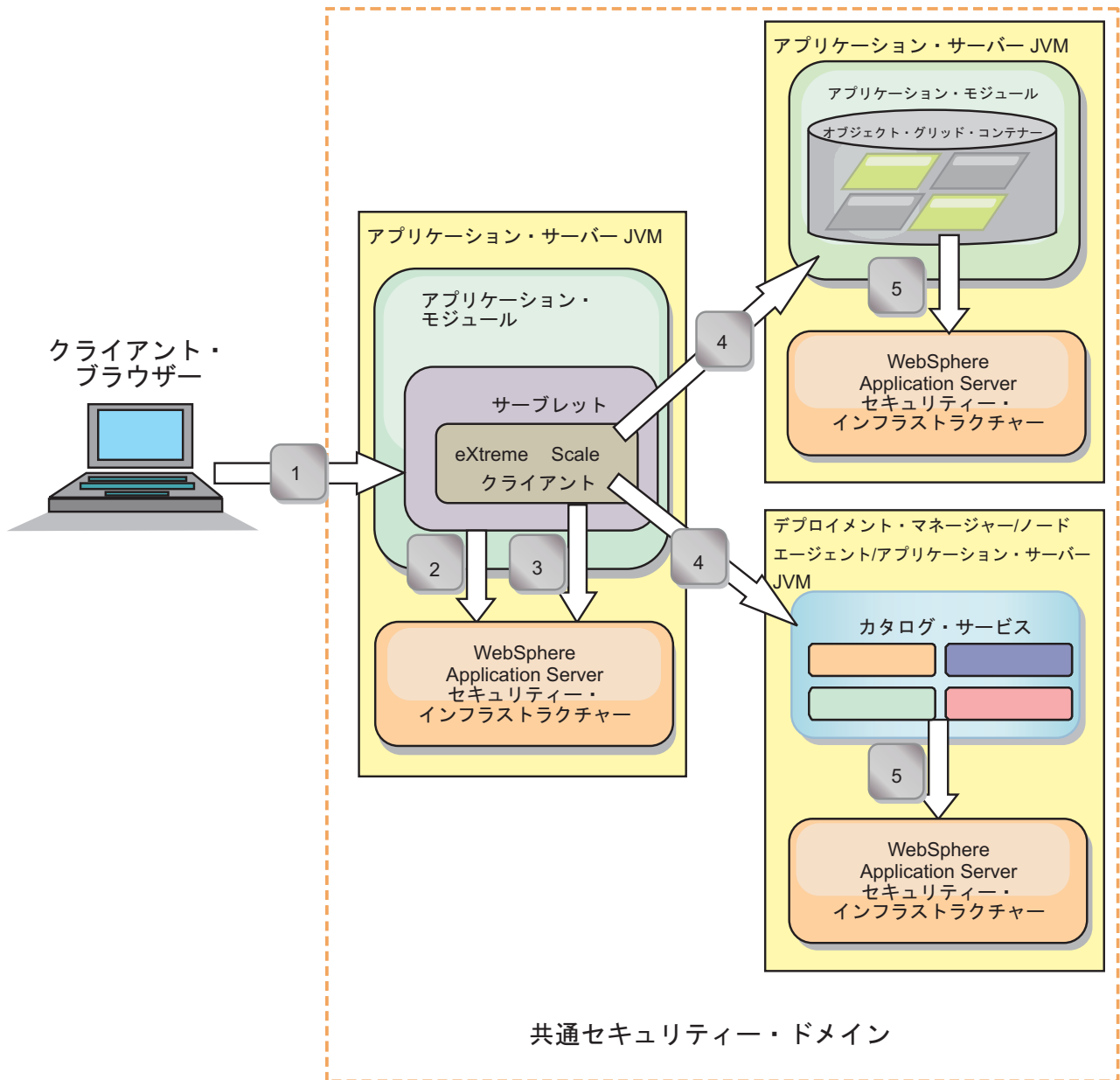


図 68. 同じセキュリティ・ドメイン内のサーバーの認証フロー

前の図で、アプリケーション・サーバーは同じセキュリティ・ドメイン内にあります。1 台のアプリケーション・サーバーが Web アプリケーションをホストしており、eXtreme Scale クライアントでもあります。別のアプリケーション・サーバーは、コンテナ・サーバーをホストしています。デプロイメント・マネージャーまたはノード・エージェントの Java 仮想マシン (JVM) は、カタログ・サービスをホストしています。図の矢印は、認証プロセス・フローを示しています。

1. エンタープライズ・アプリケーション・ユーザーは、Web ブラウザーを使用して、最初のアプリケーション・サーバーにユーザー名とパスワードを指定してログインします。
2. 最初のアプリケーション・サーバーは、クライアントのユーザー名とパスワードを WebSphere Application Server セキュリティ・インフラストラクチャーに送信して、ユーザー・レジストリーに対して認証を行います。例えば、このユーザー

ー・レジストリーは LDAP サーバーである場合があります。この結果として、セキュリティ情報がアプリケーション・サーバー・スレッドに保管されます。

3. JavaServer Pages (JSP) ファイルは、サーバー・スレッドからセキュリティ情報を取得するために eXtreme Scale クライアントとして機能します。JSP ファイルは、WebSphere Application Server セキュリティー・インフラストラクチャーを呼び出して、エンタープライズ・アプリケーション・ユーザーを表すセキュリティ・トークンを取得します。
4. eXtreme Scale クライアント、すなわち、JSP ファイルは、要求と一緒にセキュリティ・トークンを、他の JVM でホストされているコンテナ・サーバーおよびカタログ・サービスに送信します。カタログ・サーバーおよびコンテナ・サーバーは、WebSphere Application Server セキュリティー・トークンを eXtreme Scale クライアント資格情報として使用します。
5. カタログ・サーバーおよびコンテナ・サーバーは、セキュリティ・トークンをユーザーのセキュリティ・トークン情報に変換するために、セキュリティ・トークンを WebSphere Application Server セキュリティー・インフラストラクチャーに送信します。このユーザー・セキュリティ情報は、Subject オブジェクトによって示され、プリンシパル、公開資格情報、および秘密資格情報を含んでいます。この変換を行うことができるのは、eXtreme Scale クライアント、カタログ・サーバー、およびコンテナ・サーバーをホストしているアプリケーション・サーバーが同じ WebSphere Application Server Lightweight Third-Party Authentication (LTPA) トークンを共有しているためです。

## 認証統合

### WebSphere Application Server との分散セキュリティ統合:

分散モデルでは、以下のクラスを使用します。

- com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator
- com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenAuthenticator
- com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredential

これらのクラスの使用例については、90 ページの『チュートリアル: WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合』を参照してください。

サーバー・サイドで、WSTokenAuthenticator オーセンティケーターを使用して、WSTokenCredential オブジェクトを認証します。

### WebSphere Application Server とのローカル・セキュリティ統合:

ローカル ObjectGrid モデルでは、以下のクラスを使用します。

- com.ibm.websphere.objectgrid.security.plugins.builtins. WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins. WSSubjectValidationImpl

これらのクラスについて詳しくは、ローカル・セキュリティ・プログラミングを参照してください。WSSubjectSourceImpl クラスを SubjectSource プラグインとして構成し、WSSubjectValidationImpl クラスを SubjectValidation プラグインとして構成することができます。

## WebSphere Application Server でのトランスポート層セキュリティー・サポート

eXtreme Scale クライアント、コンテナ・サーバー、またはカタログ・サーバーが WebSphere Application Server プロセスで実行している場合、eXtreme Scale トランスポート・セキュリティーは WebSphere Application Server CSIV2 トランスポート設定によって管理されます。eXtreme Scale クライアントまたはコンテナ・サーバーについては、SSL 設定を構成するために eXtreme Scale クライアントまたはサーバーのプロパティーを使用するべきではありません。すべての SSL 設定は、WebSphere Application Server 構成で指定するようにしてください。

ただし、カタログ・サーバーは少し異なります。カタログ・サーバーは独自の専有トランスポート・パスを持っていますが、これは WebSphere Application Server CSIV2 トランスポート設定では管理できません。このため、SSL プロパティーは引き続き、カタログ・サーバーに対してサーバー・プロパティー・ファイルで構成する必要があります。詳しくは、90 ページの『チュートリアル: WebSphere eXtreme Scale セキュリティーの WebSphere Application Server との統合』を参照してください。

## カタログ・サービス・ドメインのクライアント・セキュリティーの構成

カタログ・サービス・ドメインのクライアント・セキュリティーを構成して、デフォルトのクライアント認証構成プロパティーを定義できます。クライアントをホスティングしている Java 仮想マシン (JVM) 内でクライアント・プロパティー・ファイルが見つからない場合、またはクライアントがセキュリティー・プロパティーをプログラムで指定しない場合、これらのプロパティーが使用されます。クライアント・プロパティー・ファイルが存在する場合、コンソールで指定したプロパティーがファイル内の値をオーバーライドします。これらのプロパティーは、`com.ibm.websphere.xs.sessionFilterProps` カスタム・プロパティーを使用して `splicer.properties` ファイルを指定するか、アプリケーション EAR ファイルを接合することでオーバーライドできます。

### 始める前に

- リモート・データ・グリッドでのクライアントの認証にどのような `CredentialGenerator` 実装を使用しているか把握しておく必要があります。WebSphere eXtreme Scale が提供する実装 (`UserPasswordCredentialGenerator` または `WSTokenCredentialGenerator`) のいずれかを使用できます。

`CredentialGenerator` インターフェースのカスタム実装を使用することもできます。カスタム実装はランタイム・クライアントおよびサーバーのクラスパス内に存在しなければなりません。WebSphere Application Server を使用して HTTP セッション・シナリオを構成する場合は、デプロイメント・マネージャーのクラスパス内とクライアントを実行しているアプリケーション・サーバーのクラスパス内に実装を配置する必要があります。

- カatalog・サービス・ドメインが定義されている必要があります。詳しくは、288 ページの『WebSphere Application Server でのカタログ・サービス・ドメインの作成』を参照してください。

## このタスクについて

サーバー・サイドの資格情報認証を有効にした場合は、次のいずれかのシナリオを構成して、カタログ・サービス・ドメインのクライアント・セキュリティーを構成する必要があります。

- サーバー・サイドのセキュリティー・ポリシーの **credentialAuthentication** プロパティーを「Required」に設定する。
- サーバー・サイドのセキュリティー・ポリシーの **credentialAuthentication** プロパティーを「Supported」に設定し、さらに ObjectGrid XML ファイル内に **authorizationMechanism** を指定する。

これらのシナリオでは、クライアントから資格情報が渡される必要があります。クライアントから渡される資格情報は、CredentialGenerator インターフェースを実装するクラスの `getCredential` メソッドから取得されます。HTTP セッション構成シナリオでは、ランタイムが、リモート・データ・グリッドに渡される資格情報を生成するときに使用する CredentialGenerator 実装を把握している必要があります。使用する CredentialGenerator 実装クラスを指定しないと、クライアントが認証されないため、リモート・データ・グリッドは、クライアントからの要求を拒否します。

## 手順

クライアント・セキュリティー・プロパティーを定義します。WebSphere Application Server 管理コンソールで、「システム管理」 > 「WebSphere eXtreme Scale」 > 「カタログ・サービス・ドメイン」 > 「*catalog\_service\_domain\_name*」 > 「クライアント・セキュリティー・プロパティー」をクリックします。そのページにあるクライアント・セキュリティー・プロパティーを指定し、変更を保存します。設定できるプロパティーのリストについては、305 ページの『クライアント・セキュリティー・プロパティー』を参照してください。

## タスクの結果

カタログ・サービス・ドメインで構成したクライアント・セキュリティー・プロパティーが、デフォルト値として使用されます。ユーザーが指定する値は、`client.properties` ファイルに定義されているプロパティーをオーバーライドします。

## 次のタスク

セッション管理に WebSphere eXtreme Scale を使用するようにアプリケーションを構成します。詳しくは、350 ページの『WebSphere Application Server の HTTP セッション管理のためのアプリケーションの自動接続』を参照してください。

---

## ローカル・セキュリティーの使用可能化

WebSphere eXtreme Scale によりいくつかのセキュリティー・エンドポイントが提供され、カスタム・メカニズムを統合できるようになります。ローカル・プログラミング・モデルにおける主なセキュリティー機能は許可で、認証サポートはありません。既存の WebSphere Application Server 認証とは別個に認証を行う必要があります。しかし、提供されるプラグインを使用して、Subject オブジェクトを取得および検証できます。

## このタスクについて

ローカル・セキュリティーは、ObjectGrid XML 記述子ファイルまたはプログラムで使用可能に設定できます。

### 手順

ローカル・セキュリティーを ObjectGrid XML 記述子ファイルで使用可能に設定します。

ObjectGridSample エンタープライズ・アプリケーションの例で使用される `secure-objectgrid-definition.xml` ファイルを以下の例に示します。セキュリティーを使用可能にするには、`securityEnabled` attribute 属性を `true` に設定します。

```
<objectGrids>
  <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
    authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
  </objectGrid>
</objectGrids>
```

### 次のタスク

セキュリティーが使用可能に設定されたコンテナ・サーバーとカタログ・サーバーを開始します。

---

## セキュア・サーバーの始動と停止

サーバーを始動および停止するときに、セキュリティー固有の構成を指定することで、セキュリティーが使用可能になります。

### スタンドアロン環境でのセキュア・サーバーの始動

セキュアなスタンドアロン・サーバーを始動するには、`startOgServer` コマンドにパラメーターを指定することで、適切な構成ファイルを渡します。

#### 始める前に

認証または許可について、外部のクライアント・セキュリティー・プロバイダーを使用する場合は、`CLIENT_AUTH_LIB` 環境変数を定義します。コマンド行ウィンドウまたは端末ウィンドウを開き、使用しているオペレーティング・システムに適したコマンドを実行します。

- **Windows** `set CLIENT_AUTH_LIB=<path_to_security_JAR_or_classes>`
- **UNIX** `set CLIENT_AUTH_LIB=<path_to_security_JAR_or_classes> export CLIENT_AUTH_LIB`

`startOgServer` コマンドおよび `stopOgServer` コマンドが実行されると、この変数がクラスパスに追加されます。

### 手順

- セキュア・コンテナ・サーバーを始動します。

セキュア・コンテナ・サーバーの始動には、次のセキュリティー構成ファイルが必要です。

- **サーバー・プロパティ・ファイル:** サーバー・プロパティ・ファイルは、サーバーに固有のセキュリティ・プロパティを構成します。詳しくは、サーバー・プロパティ・ファイルを参照してください。

**startOgServer** スクリプトの中に次の引数を指定して、この構成ファイルの場所を指定します。

**-serverProps**

サーバー固有のセキュリティ・プロパティが含まれているサーバー・プロパティ・ファイルへのパスを指定します。このプロパティに対して指定されるファイル名の形式は、プレーン・ファイル・パス形式です。例えば、`../security/server.properties` などです。

- セキュア・カタログ・サーバーを始動します。

セキュア・カタログ・サービスを開始するには、次の構成ファイルが必要です。

- **セキュリティ記述子 XML ファイル:** セキュリティ記述子 XML ファイルは、カタログ・サーバーおよびコンテナ・サーバーを含む、すべてのサーバーに共通するセキュリティ・プロパティを記述します。プロパティの例の 1 つは、ユーザー・レジストリーおよび認証メカニズムを表すオーセンティケーター構成です。
- **サーバー・プロパティ・ファイル:** サーバー・プロパティ・ファイルは、サーバーに固有のセキュリティ・プロパティを構成します。

**startOgServer** スクリプトの中に次の引数を指定して、構成ファイルの場所を指定します。

**-clusterSecurityFile** および **-clusterSecurityUrl**

これらの引数は、セキュリティ記述子 XML ファイルの場所を指定します。**-clusterSecurityFile** パラメーターを使用してローカル・ファイルを指定するか、**-clusterSecurityUrl** パラメーターを使用して `objectGridSecurity.xml` ファイルの URL を指定します。

**-serverProps**

サーバー固有のセキュリティ・プロパティが含まれているサーバー・プロパティ・ファイルへのパスを指定します。このプロパティに対して指定されるファイル名の形式は、プレーン・ファイル・パス形式です。例えば、`c:/tmp/og/catalogserver.props` などです。

## WebSphere Application Server でのセキュア・サーバーの始動

WebSphere Application Server の中でセキュア・サーバーを始動するには、汎用 Java 仮想マシン (JVM) 引数の中にセキュリティ構成ファイルを指定する必要があります。

### 手順

- WebSphere Application Server の中でセキュア・カタログ・サービスを開始します。

カタログ・サーバーには、以下の 2 つの異なるレベルのセキュリティ情報が含まれます。



- `-Dobjectgrid.cluster.security.xml.url`: カタログ・サーバーおよびコンテナ・サーバーも含めて、すべてのサーバーに共通するセキュリティー・プロパティーを記述する `objectGridSecurity.xml` ファイルの場所を指定します。定義されたセキュリティー・プロパティーの例はオーセンティケーター構成で、これは、ユーザー・レジストリーおよび認証メカニズムを表します。このプロパティーに指定するファイル名は、URL 形式 (例えば `file:///tmp/og/objectGridSecurity.xml`) でなければなりません。
  - `-Dobjectgrid.server.props`: サーバー固有のセキュリティー・プロパティーが含まれているサーバー・プロパティー・ファイルを指定します。このプロパティーに対して指定されるファイル名の形式は、プレーン・ファイル・パス形式です。例えば、`c:/tmp/og/catalogserver.props` などです。
1. WebSphere Application Server 管理コンソールで、「システム管理」をクリックします。例えばデプロイメント・マネージャーなど、カタログ・サーバーをデプロイするプロセスをクリックします。
  2. 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」をクリックします。
  3. 「汎用 JVM 引数」フィールドに、プロパティーを入力します。追加する値の例は次のとおりです。
 

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/objectGridSecurity.xml
-Dobjectgrid.server.props=/tmp/og/catalogserver.props
```
  4. 「OK」をクリックして、変更を保存します。
- WebSphere Application Server の中でセキュア・コンテナ・サーバーを始動します。

コンテナ・サーバーは、カタログ・サーバーに接続するときに、オーセンティケーター構成やログイン・セッション・タイムアウトの設定といった、`objectGridSecurity.xml` ファイル内のセキュリティー構成を継承します。  
`-Dobjectgrid.server.props` プロパティーの中に、特定のコンテナ・サーバーのサーバー固有のセキュリティー・プロパティーも定義する必要があります。

このプロパティーに対して指定されるファイル名の形式は、単なるプレーン・ファイル・パス形式です。例えば、`c:/tmp/og/server.props` などです。

上記と同じ手順に従って、セキュリティー・プロパティーを汎用 JVM 引数に追加してください。

1. サーバーの Java 仮想マシン・ページを開きます。WebSphere Application Server 管理コンソールで、「サーバー」 > 「アプリケーション・サーバー」 > `server_name` > 「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」 をクリックします。
2. 「汎用 JVM 引数」フィールドに、プロパティーを入力します。追加する値の例は次のとおりです。
 

```
-Dobjectgrid.server.props=/opt/wxs/security/server2.props
```
3. 「OK」をクリックして、変更を保存します。

## セキュア・サーバーの停止

セキュアなカタログ・サーバーまたはコンテナ・サーバーを停止するには、1 つのセキュリティー構成ファイルが必要です。

## 手順

セキュアなカタログ・サーバーまたはコンテナ・サーバーを停止します。  
セキュア・サーバーの停止には、次のセキュリティ構成ファイルが必要です。

- **クライアント・プロパティ・ファイル:** クライアント・プロパティ・ファイルを使用して、クライアント・セキュリティ・プロパティを構成できます。クライアント・セキュリティ・プロパティは、クライアントがセキュア・サーバーに接続するために必要です。詳しくは、クライアント・プロパティ・ファイルを参照してください。

**stopOgServer** スクリプトの中に次の引数を指定して、構成ファイルの場所を指定します。

### **-clientSecurityFile**

クライアントのセキュリティ・プロパティを定義するクライアント・プロパティ・ファイルへのパスを指定します。このプロパティに指定するファイル名は、プレーン・ファイル・パス形式です。例えば、  
../security/objectGridClient.properties などです。

### 例

```
stopOgServer.bat|sh cs1 -catalogServiceEndpoints  
cs1:MyServer1.company.com:6601:6602,  
cs2:MyServer2.company.com:6601:6602,  
cs3:MyServer3.company.com:6601:6602  
-clientSecurityFile ../security/objectGridClient.properties
```

---

## xscmd ユーティリティのためのセキュリティ・プロファイルの構成

セキュリティ・プロファイルを作成すると、保存されたセキュリティ・パラメーターを使用して、セキュアな環境で **xscmd** ユーティリティを使用できます。

### 始める前に

**xscmd** ユーティリティのセットアップについては、473 ページの『**xscmd** ユーティリティによる管理』を参照してください。

### このタスクについて

**xscmd** コマンドの残り部分で **-ssp profile\_name** または **--saveSecProfile profile\_name** パラメーターを使用して、セキュリティ・プロファイルを保存できます。プロファイルには、ユーザー名とパスワード、資格情報生成プログラム、鍵ストア、トラストストア、およびトランスポート・タイプについての設定を含めることができます。

**xscmd** ユーティリティ内の **ProfileManagement** コマンド・グループには、セキュリティ・プロファイルの管理のためのコマンドが含まれます。

### 手順

- セキュリティ・プロファイルを保存します。

セキュリティ・プロファイルを保存するには、コマンドの残り部分で **-ssp profile\_name** または **--saveSecProfile profile\_name** パラメーターを使用します。このパラメーターをコマンドに追加すると、次のパラメーターが保存されます。

```

-al,--alias <alias>
-arc,--authRetryCount <integer>
-ca,--credAuth <support>
-cgc,--credGenClass <className>
-cgp,--credGenProps <property>
-cxpv,--contextProvider <provider>
-ks,--keyStore <filePath>
-ksp,--keyStorePassword <password>
-kst,--keyStoreType <type>
-prot,--protocol <protocol>
-pwd,--password <password>
-ts,--trustStore <filePath>
-tsp,--trustStorePassword <password>
-tst,--trustStoreType <type>
-tt,--transportType <type>
-user,--username <username>

```

セキュリティー・プロファイルは、

`user_home%.xscmd%profiles%security%<profile_name>.properties` ディレクトリに保存されます。

**重要:** `profile_name` パラメーターでは、ファイル名拡張子 `.properties` を含めないようにしてください。この拡張子は自動的にファイル名に付加されます。

- 保存したセキュリティー・プロファイルを使用します。

保存したセキュリティー・プロファイルを使用するには、実行するコマンドに、`-sp profile_name` または `--securityProfile profile_name` パラメーターを追加します。コマンド例: `xscmd -c listHosts -cep myhost.mycompany.com -sp myprofile`

- **ProfileManagement** コマンド・グループの中のコマンドをリストします。

次のコマンドを実行します。 `xscmd -lc ProfileManagement`

- 既存のセキュリティー・プロファイルをリストします。

次のコマンドを実行します。 `xscmd -c listProfiles -v`

- セキュリティー・プロファイルの中に保存されている設定を表示します。

次のコマンドを実行します。 `xscmd -c showProfile -pn profile_name`

- 既存のセキュリティー・プロファイルを削除します。

次のコマンドを実行します。 `xscmd -c RemoveProfile -pn profile_name`

---

## J2C クライアント接続の保護

Java 2 Connector (J2C) アーキテクチャーを使用して、WebSphere eXtreme Scale クライアントとアプリケーションの間の接続を保護します。

### このタスクについて

アプリケーションが接続ファクトリーを参照し、接続ファクトリーがリモート・データ・グリッドへの接続を確立します。各接続ファクトリーは単一の eXtreme Scale クライアント接続をホストし、この接続がすべてのアプリケーション・コンポーネントに対して再利用されます。

**重要:** eXtreme Scale クライアント接続にはニア・キャッシュが含まれることがあるため、アプリケーションが接続を共有しないことが重要です。アプリケーション間でオブジェクトを共有する問題を回避するためには、単一のアプリケーション・インスタンスに対して接続ファクトリーが存在する必要があります。

資格情報生成プログラムは、API を使用するか、クライアント・プロパティ・ファイルの中で設定できます。クライアント・プロパティ・ファイルの中で、`securityEnabled` プロパティと `credentialGenerator` プロパティが使用されます。以下のコード例は、印刷の都合上、複数行で表示されています。

```
securityEnabled=true
credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins.
  UserPasswordCredentialGenerator
credentialGeneratorProps=operator XXXXXX
```

クライアント・プロパティ・ファイルの中の資格情報生成プログラムと資格情報は、eXtreme Scale 接続操作と、デフォルトの J2C 資格情報に使用されます。したがって、API で指定された資格情報は、J2C の接続時に、J2C 接続のために使用されます。ただし、J2C 接続時に資格情報が指定されなければ、クライアント・プロパティ・ファイルの中の資格情報生成プログラムが使用されます。

## 手順

1. J2C 接続が eXtreme Scale クライアントを表す場所のセキュア・アクセスをセットアップします。 `ClientPropertiesResource` 接続ファクトリー・プロパティまたは `ClientPropertiesURL` 接続ファクトリー・プロパティを使用して、クライアント認証を構成します。
2. eXtreme Scale の適切な資格情報生成プログラム・オブジェクトを参照する接続ファクトリーを使用するように、クライアント・セキュリティ・プロパティを構成します。これらのプロパティは、eXtreme Scale サーバー・セキュリティとも互換性があります。例えば、eXtreme Scale が WebSphere Application Server にインストールされている場合、WebSphere 資格情報には `WSTokenCredentialGenerator` 資格情報生成プログラムを使用します。あるいは、スタンドアロン環境で eXtreme Scale を実行するときは、`UserPasswordCredentialGenerator` 資格情報生成プログラムを使用します。次の例では、資格情報は、クライアント・プロパティ内の構成を使用するのではなく、API 呼び出しを使用して、プログラマチックに渡されます。

```
XSConnectionSpec spec = new XSConnectionSpec();
spec.setCredentialGenerator(new UserPasswordCredentialGenerator("operator", "xxxxxx"));
Connection conn = connectionFactory.getConnection(spec);
```

3. (オプション) 必要であれば、ニア・キャッシュを使用不可にします。

単一の接続ファクトリーからのすべての J2C 接続は、単一のニア・キャッシュを共有します。グリッド・エントリー許可とマップ許可はサーバーで検証されますが、ニア・キャッシュでは検証されません。アプリケーションが J2C 接続を作成するために複数の資格情報を使用し、これらの資格情報に対して、構成が特定のグリッド・エントリー許可およびマップ許可を使用する場合、ニア・キャッシュを使用不可にします。接続ファクトリー・プロパティ `ObjectGridResource` または `ObjectGridURL` を使用して、ニア・キャッシュを使用不可にします。`numberOfBuckets` プロパティを 0 に設定します。

4. (オプション) 必要であれば、セキュリティ・ポリシー設定を設定します。

J2EE アプリケーションに組み込みの eXtreme Scale リソース・アダプター・アーカイブ (RAR) ファイル構成が含まれる場合、アプリケーションのセキュリティー・ポリシー・ファイルの中に、追加のセキュリティー・ポリシー設定を設定することが必要な場合もあります。例えば、次のポリシーが必要です。

```
permission com.ibm.websphere.security.WebSphereRuntimePermission "accessRuntimeClasses";
permission java.lang.RuntimePermission "accessDeclaredMembers";
permission javax.management.MBeanTrustPermission "register";
permission java.lang.RuntimePermission "getClassLoader";
```

さらに、接続ファクトリーが使用するすべてのプロパティー・ファイルまたはリソース・ファイルには、ファイルまたは他の許可 (permission java.io.FilePermission "filePath"; など) が必要です。WebSphere Application Server の場合、ポリシー・ファイルは META-INF/was.policy で、これは J2EE EAR ファイルの中にあります。

## タスクの結果

カタログ・サービス・ドメインで構成したクライアント・セキュリティー・プロパティーが、デフォルト値として使用されます。ユーザーが指定する値は、client.properties ファイルに定義されているプロパティーをオーバーライドします。

## 次のタスク

eXtreme Scale データ・アクセス API を使用して、トランザクションに使用するクライアント・コンポーネントを開発します。





---

## 第 11 章 トラブルシューティング



このセクションで説明するログとトレース、メッセージ、およびリリース情報の他に、モニター・ツールを使用して環境内のデータのロケーション、データ・グリッド内のサーバーのアベイラビリティなどの問題を把握することができます。

WebSphere Application Server 環境で実行している場合、Performance Monitoring Infrastructure (PMI) を使用できます。スタンドアロン環境で実行している場合は、ベンダーのモニター・ツール (CA Wily Introscope あるいは Hyperic HQ など) を使用できます。また、`xscmd` ユーティリティを使用し、これをカスタマイズすれば、ご使用の環境に関するテキスト情報を表示させることができます。

---

### WebSphere eXtreme Scale のトラブルシューティングおよびサポート

ご使用の IBM 製品の問題を切り分け、解決するために、トラブルシューティング情報とサポート情報を利用できます。この情報には、ご使用の IBM 製品 (WebSphere eXtreme Scale を含む) と共に提供される問題判別リソースを使用するための説明が示されています。

#### 問題のトラブルシューティングのための手法

トラブルシューティングとは、体系的な方法で問題を解決することです。トラブルシューティングの目的は、想定どおりに機能しない理由を明らかにして、問題の解決方法を説明することです。ある種の一般的な手法が、トラブルシューティングのタスクに役立つこともあります。

トラブルシューティング・プロセスの最初のステップは、問題の内容を完全に記述することです。問題の記述によって、問題の原因を突き止めるためにどこから着手すべきかを、お客様自身と IBM 技術サポート担当者がわかるようになります。このステップでは、以下の基本的な項目について明確にする必要があります。

- 問題の症状はどのようなものか。
- 問題の発生場所はどこか。
- 問題はいつ発生したか。
- 問題が発生する条件は何か。
- 問題は再現可能か。

通常は、これらの質問に回答することで問題が適切に記述され、問題解決につながります。

#### 問題の症状はどのようなものか。

問題の記述を始めるときの最も明確な質問は「問題は何か」ということです。この質問は単純なように思われますが、問題をより具体的に説明する、さまざまな観点からの質問に細分化することができます。細分化した質問には以下のようなものがあります。

- だれが、または何が問題を報告するか。
- エラー・コードおよびメッセージの内容は何か。

- システムにどのような障害が発生したか。ループ、ハング、異常終了、パフォーマンス低下、不適切な結果などがあります。

### 問題の発生場所はどこか。

問題の発生源を特定することは、必ずしも簡単ではありませんが、これは問題を解決する上で最も重要な段階の 1 つです。報告元のコンポーネントと障害が発生しているコンポーネントの間に、いくつものテクノロジー層が存在している場合があります。ネットワーク、データ・グリッド、およびサーバーは、問題を調査するときには考慮すべきコンポーネントの一部にすぎません。

次のような質問は、問題の層の切り分けのため、問題の発生箇所に焦点が集まるようにします。

- 問題は 1 つのプラットフォームまたはオペレーティング・システムに固有か、それとも複数のプラットフォームまたはオペレーティング・システムに共通か。
- 現在の環境と構成はサポートされているか。
- その問題はどのユーザーに対しても起こるのか。
- (マルチサイトのインストール済み環境の場合。) その問題はすべてのサイトで起こるのか。

ある層で問題が報告されても、必ずしもその層が問題の発生源とは限りません。問題の発生源の特定作業には、問題が存在する環境を理解することが含まれます。しばらく時間をかけて、問題のある環境のすべての内容 (オペレーティング・システムとバージョン、対応するすべてのソフトウェアとバージョン、ハードウェア情報など) を記述してください。サポートされる構成の環境で実行していることを確認してください。多くの場合、問題をトレースすると、ソフトウェア・レベルに互換性がないことがわかります (一緒に実行することを意図していないソフトウェア・レベルであったり、完全には一緒にテストされていないソフトウェア・レベルであったりします)。

### 問題はいつ発生したか。

障害発生に至るイベントについて、特に発生が 1 回限りのケースについて、詳しい時系列対照表を作成してください。最も簡単に時系列対照表を作成する方法は、逆方向に作業することです。エラーが報告された時点 (できればミリ秒単位に至るまで精密に) から開始して、使用可能なログと情報を通じて逆方向に作業します。一般に、診断ログ内で最初に見つかる疑わしいイベントまで調べる必要があります。

イベントの詳細な時系列対照表を作成するには、以下の質問に答えます。

- 問題が昼間または夜間の特定の時刻のみに発生するか。
- どれくらいの頻度で問題が発生するか。
- どのような一連のイベントを経て、問題が報告された時点に至ったのか。
- 環境の変更 (ソフトウェアまたはハードウェアのアップグレードやインストールなど) の後に問題が発生したか。

このようなタイプの質問に回答することで、問題調査のための基準の枠組みが得られます。

## 問題が発生する条件は何か。

問題の発生時に稼働していたシステムとアプリケーションを知ることは、トラブルシューティングの重要な部分です。ご使用の環境に関する次のような質問は、問題の根本原因の特定に役立ちます。

- 問題は同じタスクの実行時に常に発生するか。
- この問題が発生するときに、必ず発生する一連のイベントがあるか。
- 他のいずれかのアプリケーションでも同時に障害が発生したか。

このようなタイプの質問に回答することによって、問題が発生した環境が明らかになり、依存関係の相関付けができます。同じ時間の頃に複数の問題が発生したからといって、それらの問題が必ずしも関連しているとは限らないことを覚えておいてください。

## 問題は再現可能か。

トラブルシューティングの観点からすると、理想的な問題とは、再現できる問題であるということです。通常、問題を再現できる場合は、調査に役立てるために自由に使用できるツールまたは手順の数が多くなります。このため、再現できる問題は、多くの場合デバッグや解決が簡単です。

しかし、再現できる問題には欠点があります。すなわち、その問題がビジネスに大きな影響を与える場合、問題が再発生することは好ましくありません。可能であれば、テスト環境または開発環境で問題を再現してください。一般に、このようにすると、調査時の柔軟性と管理能力が向上します。

- 問題をテスト・システムで再現することができるか。
- 複数のユーザーまたはアプリケーションで同じタイプの問題が発生しているか。
- 単一のコマンド、一連のコマンド、特定のアプリケーションのいずれを実行することによって問題を再現できるか。

## 知識ベースの検索

以下は英語のみの対応となります。多くの場合、IBM 知識ベースを検索することによって、問題の解決策を見つけることができます。使用可能なリソース、サポート・ツール、および検索方式を使用して、最適の結果を得ることができます。

### このタスクについて

WebSphere eXtreme Scale のインフォメーション・センターを検索すれば、役立つ情報を見つけることができます。ただし、疑問の回答を得たり問題を解決するために、インフォメーション・センター以外で情報を検索する必要があることもあります。

### 手順

必要な情報を知識ベースで検索するには、以下のいずれか 1 つまたは複数の方法を使用します。

- IBM Support Assistant (ISA) を使用してコンテンツを検索します。

ISA は、IBM ソフトウェア製品に関する質問への回答や問題の解決に役立つ、無料のソフトウェア保守ワークベンチです。ISA のダウンロードとインストールの手順については、ISA Web サイトを参照してください。

- IBM Support Portal を使用して必要なコンテンツを検索します。

IBM Support Portal は、IBM のすべてのシステム、ソフトウェア、およびサービスの、すべての技術サポート・ツールと情報が 1 つにまとめられた、中心となる場所です。この IBM Support Portal という 1 つの場所から、IBM エレクトロニック支援ポートフォリオにアクセスできます。各ページを調整して、問題の予防と迅速な問題解決に必要な情報やリソースに焦点を当てることができます。IBM Support Portal をよく理解するためには、このツールに関するデモ・ビデオ ([https://www.ibm.com/blogs/SPNA/entry/the\\_ibm\\_support\\_portal\\_videos](https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos)) をご覧ください。このビデオは、IBM Support Portal について紹介し、トラブルシューティングやその他のリソースについて説明し、ポートレットの移動、追加、削除によってページを調整する方法のデモを示します。

- WebSphere eXtreme Scale に関するコンテンツを検索するには、以下の追加の技術資料のいずれかを使用します。
  - WebSphere eXtreme Scale リリース情報
  - WebSphere eXtreme Scale サポート Web サイト
  - WebSphere eXtreme Scale フォーラム (forum)
- IBM マストヘッド検索を使用してコンテンツを検索します。IBM マストヘッド検索を使用するには、[ibm.com](http://ibm.com)<sup>®</sup> の任意のページの最上部にある「検索」フィールドに検索文字列を入力します。
- 外部の検索エンジン (Google、Yahoo、Bing など) を使用して、コンテンツを検索します。外部検索エンジンを使用すると、[ibm.com](http://ibm.com) ドメイン以外の情報が結果に含まれる可能性が高くなります。ただし、[ibm.com](http://ibm.com) 以外の場所にあるニュースグループ、フォーラム、およびブログで、IBM 製品に関する問題解決のための有用な情報が見つかることもあります。

**ヒント:** IBM 製品に関する情報を検索する場合は、「IBM」と製品の名称を検索に含めてください。

## フィックスの入手

以下は英語のみの対応となります。お客様の問題の解決に、プロダクトのフィックスが有効な場合があります。

### 手順

フィックスを見つけてインストールするには、以下のようにします。

1. フィックスを入手するために必要なツールを取得します。IBM Update Installer を使用して、WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのさまざまなタイプの保守パッケージをインストールして、適用します。Update Installer は定期的に保守されるため、そのツールの最新バージョンを使用する必要があります。
2. 必要なフィックスを特定します。WebSphere eXtreme Scale の推奨フィックスを参照して、最新のフィックスを選択してください。フィックスを選択すると、そのフィックスのダウンロード資料が開きます。

3. フィックスをダウンロードします。ダウンロード資料で、「Download package」セクションの最新フィックスのリンクをクリックします。
4. フィックスを適用します。ダウンロード資料の「Installation Instructions」セクションに記載されている説明に従ってください。
5. フィックスやその他の IBM サポート情報について週次の E メール通知を受信できるよう、登録してください。

## Fix Central からのフィックスの入手

以下は英語のみの対応となります。Fix Central を使用して、IBM サポートが推奨する、WebSphere eXtreme Scale を含むさまざまな製品のフィックスを見つけることができます。Fix Central では、ご使用のシステム用のフィックスを検索、選択、注文、およびダウンロードすることができ、その際に配信オプションを選択できます。以下は英語のみの対応となります。お客様の問題の解決に、WebSphere eXtreme Scale プロダクトのフィックスが有効な場合があります。

### 手順

フィックスを見つけてインストールするには、以下のようにします。

1. フィックスを入手するために必要なツールを取得します。製品アップデート・インストーラーがインストールされていない場合は、それを取得します。インストーラーは Fix Central からダウンロードできます。このサイトでは、アップデート・インストーラーのダウンロード、インストール、および構成の手順を提供しています。
2. 製品として選択し、解決したい問題に関連のある 1 つ以上のチェック・ボックスを選択します。
3. 必要なフィックスを特定して選択します。
4. フィックスをダウンロードします。
  - a. ダウンロード資料を開き、「Download Package」セクションのリンクをたどります。
  - b. ファイルのダウンロード時に、保守ファイルの名前が変更されていないことを確認してください。このような変更は、意図的に行われる場合もあれば、特定の Web ブラウザーやダウンロード・ユーティリティによって偶発的に行われる場合もあります。
5. フィックスを適用します。
  - a. ダウンロード資料の「Installation Instructions」セクションに記載されている説明に従ってください。
  - b. 詳しくは、製品資料の『Update Installer を使用したフィックスのインストール』のトピックを参照してください。
6. オプション: フィックスおよびその他の IBM サポートの更新については、登録して各週の E メール通知を受け取ってください。

## IBM サポートへの連絡

IBM サポートでは、製品の障害に関する支援を提供し、FAQ に回答し、ユーザーが製品に関する問題を解決する上で協力します。



## 始める前に

問題点に関する回答や解決方法について、リリース情報などのセルフ・ヘルプ・オプションを使用しても判明しない場合は、IBM サポートにご連絡ください。IBM サポートに連絡するには、お客様の会社または組織が有効な IBM 保守契約を締結しており、お客様が IBM に問題を送信する許可を受けている必要があります。使用可能なサポートのタイプについては、「*Software Support Handbook*」の『Support portfolio』のトピックを参照してください。

## 手順

問題について IBM サポートに連絡を取るには、次のようにしてください。

1. 問題を定義し、バックグラウンド情報を収集して、問題の重大度を判別します。詳しくは、「*Software Support Handbook*」の『Getting IBM support』のトピックを参照してください。
2. 診断情報を収集します。
3. IBM サポートに以下のいずれかの方法で問題を送信します。
  - IBM Support Assistant (ISA) を使用:
  - IBM サポート・ポータルによるオンライン: 「Service Request」ページの「Service Request」ポータルから、お客様のすべてのサービス要求をオープン、更新、および表示できます。
  - 電話: 自国での連絡先の電話番号を調べるには、Directory of worldwide contacts の Web ページを参照してください。

## タスクの結果

送信した問題がソフトウェア障害に関するものである場合、または文書の不備や不正確さに関するものである場合、IBM サポートはプログラム診断依頼書 (APAR) を作成します。APAR には問題が詳細に記載されます。IBM サポートでは、APAR が解決されてフィックスが配信されるまで、お客様が実施可能な回避策を可能な限り提供します。IBM では、解決された APAR を IBM サポート Web サイトに毎日公開しているため、同じ問題が発生した他のユーザーも同じ解決策を利用できます。

## IBM との情報の交換

以下は英語のみの対応となります。問題の診断や特定を行うには、ご使用のシステムからのデータと情報を IBM サポートに提供していただく必要がある場合があります。また、IBM サポートから、問題判別に使用するツールまたはユーティリティーをご提供する場合があります。

### IBM サポートへの情報の送信

問題解決に必要な時間を減らすため、トレースおよび診断情報を IBM サポートに送信していただくことができます。

## 手順

IBM サポートに診断情報を提出するには、以下のようになります。

1. 問題管理レコード (PMR) を開きます。



2. 必要な診断データを収集します。診断データは、PMR の解決にかかる時間の節約に役立ちます。診断データは、手動で収集することも自動的に収集することもできます。
  - データを手動で収集する。
  - データを自動的に収集する。
3. .zip または .tar ファイル形式を使用してファイルを圧縮します。
4. ファイルを IBM に転送します。以下のいずれかの方法を使用して、IBM にファイルを転送することができます。
  - IBM Support Assistant
  - Service Request ツール
  - 標準的なデータのアップロード方法: FTP、HTTP
  - 機密保護機能のあるデータのアップロード方法: FTPS、SFTP、HTTPS
  - E メール

z/OS 製品を使用していて、PMR の送信に ServiceLink / IBMLink を使用している場合は、E メールまたは FTP を使用して IBM サポートに診断データを送信できます。

これらのデータ交換方法は、すべて IBM サポートの Web サイトで説明されています。

## IBM サポートからの情報の受信

IBM 技術サポートの担当者から、診断ツールやその他のファイルのダウンロードをお願いする場合があります。FTP を使用して、これらのファイルをダウンロードすることができます。

### 始める前に

IBM 技術サポートの担当者から、ファイルのダウンロードに使用する推奨サーバーと、アクセス先の正確なディレクトリーおよびファイル名が指定されていることを確認してください。

### 手順

IBM サポートからファイルをダウンロードするには、以下のようにします。

1. FTP を使用して、IBM 技術サポートの担当者が指定したサイトに接続し、anonymous としてログインします。パスワードとして E メール・アドレスを使用してください。
2. 以下のように、適切なディレクトリーに移動します。
  - a. /fromibm ディレクトリーに移動します。

```
cd fromibm
```
  - b. IBM 技術サポートの担当者が指定したディレクトリーに移動します。

```
cd nameofdirectory
```
3. セッションのバイナリー・モードを有効にします。

```
binary
```

4. **get** コマンドを使用して、IBM 技術サポートの担当者が指定したファイルをダウンロードします。

```
get filename.extension
```

5. FTP セッションを終了します。

```
quit
```

## サポート更新情報のサブスクライブ

ご使用の IBM 製品に関する重要な情報を常に入手するために、更新情報をサブスクライブすることができます。

### このタスクについて

製品の更新情報を受け取るようにサブスクライブすることによって、特定の IBM サポート・ツールおよびリソースに関する重要な技術情報と更新を受け取ることができます。次の 2 つの方法のうちいずれかを使用して、更新情報をサブスクライブできます。

#### ソーシャル・メディアのサブスクリプション

製品に関し、次の RSS フィードが使用可能です。

- WebSphere eXtreme Scale フォーラム の RSS フィード

RSS の一般情報 (入門情報、RSS を使用できる IBM Web ページのリストなど) については、IBM ソフトウェア・サポート RSS フィード・サイトをご覧ください。

#### マイ通知 (My Notifications)

「マイ通知 (My Notifications)」では、任意の IBM 製品のサポート更新情報をサブスクライブできます。「マイ通知 (My Notifications)」は、以前に使用されていた同様のツール「マイ・サポート」に置き換わるものです。「マイ通知 (My Notifications)」を使用すると、E メールによる告知を、毎日、または週 1 回受信するように指定できます。発表、ヒント、製品フラッシュ (アラートとも言う)、ダウンロード、ドライバーなど、受信する情報のタイプを指定できます。「マイ通知 (My Notifications)」では、通知を受信する製品、およびユーザーのニーズに最も適した配信方法をカスタマイズして分類することができます。

### 手順

サポート更新情報をサブスクライブするには、次のようにします。

1. WebSphere eXtreme Scale フォーラム の RSS フィードをサブスクライブします。
  - a. 「サブスクリプション」ページで、RSS フィード・アイコンをクリックします。
  - b. フィードのサブスクライブに使用するオプションを選択してください。
  - c. 「サブスクライブ」をクリックします。
2. 「マイ通知 (My Notifications)」をサブスクライブするには、IBM サポート・ポータルにアクセスし、「通知」ポートレットの「マイ通知 (My Notifications)」をクリックします。
3. IBM ID とパスワードを使用してサインインし、「送信」をクリックします。

4. 更新を受け取る内容と方法を指定します。
  - a. 「サブスクライブ」タブをクリックします。
  - b. 該当するソフトウェア・ブランドまたはハードウェアのタイプを選択します。
  - c. 1 つ以上の製品名を選択して、「続行」をクリックします。
  - d. 更新を E メールで受け取るか、オンラインで指定されたフォルダーに受け取るか、RSS または Atom フィードで受け取るか、更新の受信方法に関する設定を選択します。
  - e. 受信する資料更新のタイプ (製品ダウンロードに関する新規情報、ディスカッション・グループのコメントなど) を選択します。
  - f. 「実行依頼」をクリックします。

## タスクの結果

RSS フィードと「マイ通知 (My Notifications)」の設定を変更するまで、要求した更新情報に関する通知を受け取ることとなります。必要に応じて、この設定内容は変更することができます (ある製品の使用を中止し、別の製品の使用を開始した場合など)。

---

## ロギング可能化

ログを使用して、環境のモニターおよびトラブルシューティングを実行できます。

### このタスクについて

ログは、構成に応じて異なる場所にさまざまなフォーマットで保存されます。

### 手順

#### • スタンドアロン環境でのロギング可能化

スタンドアロン・カタログ・サーバーの場合、ログは **startOgServer** コマンドを実行した場所にあります。コンテナ・サーバーの場合、デフォルトの場所を使用するか、カスタムのログの場所を設定できます。

- **デフォルトのログの場所:** ログは、サーバー・コマンドが実行されたディレクトリにあります。wxs\_home/bin ディレクトリでサーバーを開始した場合、ログおよびトレース・ファイルは bin ディレクトリ内の logs/<server\_name> ディレクトリにあります。
- **カスタムのログの場所:** コンテナ・サーバー・ログに別の場所を指定するには、次の内容を持つプロパティ・ファイル (server.properties など) を作成します。

```
workingDirectory=<directory>
traceSpec=
systemStreamToFileEnabled=true
```

**workingDirectory** プロパティは、ログおよびオプションのトレース・ファイルのルート・ディレクトリです。WebSphere eXtreme Scale は、コンテナ・サーバーの名前を持つディレクトリを作成し、SystemOut.log ファイル、SystemErr.log ファイル、およびトレース・ファイルをそこに入れます。

コンテナ開始中にプロパティ・ファイルを使用するには、**-serverProps** オプションを使用して、サーバー・プロパティ・ファイルのロケーションを指定します。

- **WebSphere Application Server** でのロギング可能化

詳しくは、WebSphere Application Server: ロギングの使用可能化および使用不能化を参照してください。

- **FFDC** ファイルを取得します。

FFDC ファイルは、IBM サポートがデバッグの補助とするファイルです。これらのファイルは、問題が発生したときに IBM サポートによって要求される場合があります。これらのファイルは、ffdc という名前のディレクトリに存在し、以下のファイルに類似したファイルが含まれています。

```
server2_exception.log  
server2_20802080_07.03.05_10.52.18_0.txt
```

## 次のタスク

指定された場所にあるログ・ファイルを表示します。SystemOut.log ファイル内で探す共通のメッセージは、次の例のような開始確認メッセージです。

CW0BJ1001I: ObjectGrid サーバー catalogServer01 は要求を処理する準備ができています。

ログ・ファイル内の特定のメッセージについて詳しくは、メッセージを参照してください。

---

## トレースの収集

トレースを使用して、環境のモニターおよびトラブルシューティングを実行できます。IBM サポートに協力を依頼する場合、サーバーに関するトレースを提供する必要があります。

### このタスクについて

トレースの収集は、WebSphere eXtreme Scale のデプロイメントにおける問題のモニターと解決に役立ちます。トレースの収集方法は、構成によって決まります。収集できるさまざまなトレース仕様のリストについては、620 ページの『トレース・オプション』を参照してください。

### 手順

- **WebSphere Application Server** 環境内でトレースを収集します。

カタログおよびコンテナ・サーバーが WebSphere Application Server 環境内にある場合、詳しくは、WebSphere Application Server: トレースによる処理を参照してください。

- **スタンドアロン・カタログまたはコンテナ・サーバーの始動コマンド**を使用して、トレースを収集します。

**startOgServer** コマンドに **-traceSpec** パラメーターと **-traceFile** パラメーターを使用して、カタログ・サービスまたはコンテナ・サーバーでトレースを設定できます。以下に例を示します。

```
startOgServer.sh catalogServer -traceSpec ObjectGridPlacement=all=enabled -traceFile /home/user1/logs/trace.log
```

**-traceFile** パラメーターはオプションです。**-traceFile** で場所を指定しなければ、トレース・ファイルは、システム出力ログ・ファイルと同じ場所に作成されます。これらのパラメーターについて詳しくは、457 ページの『**startOgServer** スクリプト』を参照してください。

- **プロパティ・ファイルを使用して、スタンドアロン・カタログまたはコンテナ・サーバーでトレースを収集します。**

プロパティ・ファイルからトレースを収集するには、次の内容のファイル (例えば `server.properties` ファイル) を作成します。

```
workingDirectory=<directory>
traceSpec=<trace_specification>
systemStreamToFileEnabled=true
```

**workingDirectory** プロパティは、ログおよびオプションのトレース・ファイルのルート・ディレクトリです。**workingDirectory** 値が設定されていない場合は、デフォルトの作業ディレクトリは、サーバーの始動に使用された場所です (例えば `wxs_home/bin`)。サーバーの始動中にプロパティ・ファイルを使用するには、**startOgServer** コマンドに **-serverProps** パラメーターを使用し、サーバー・プロパティ・ファイルの場所を指定します。サーバー・プロパティ・ファイルおよびそのファイルの使用法について詳しくは、サーバー・プロパティ・ファイルを参照してください。

- **スタンドアロン・クライアントでトレースを収集します。**

クライアント・アプリケーションの始動スクリプトにシステム・プロパティを追加することにより、スタンドアロン・クライアントでトレースの収集を開始することができます。次の例では、トレース設定は、`com.ibm.samples.MyClientProgram` アプリケーションに対して指定されています。

```
java -DtraceSettingsFile=MyTraceSettings.properties
-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
-Djava.util.logging.configureByServer=true com.ibm.samples.MyClientProgram
```

詳しくは、WebSphere Application Server: クライアントおよびスタンドアロン・アプリケーションでのトレースの使用可能化を参照してください。

- **ObjectGridManager インターフェースを使用してトレースを収集します。**

**ObjectGridManager** インターフェースで実行時にトレースを設定することもできます。**ObjectGridManager** インターフェースでのトレース設定を使用すると、**eXtreme Scale** に接続してトランザクションをコミットしている間に **eXtreme Scale** クライアント上でトレースを取得することができます。**ObjectGridManager** インターフェースでトレースを設定するには、トレース仕様およびトレース・ログを指定します。

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

ObjectGridManager インターフェースについて詳しくは、*プログラミング・ガイド* 内の ObjectGridManager インターフェースを使用した ObjectGrid との相互作用についての情報を参照してください。

- **xscmd** ユーティリティを使用して、コンテナ・サーバーでトレースを収集します。

**xscmd** ユーティリティを使用してトレースを収集するには、**-c setTraceSpec** コマンドを使用します。**xscmd** ユーティリティを使用して、開始時ではなく実行時にスタンドアロン環境でトレースを収集します。すべてのサーバーおよびカタログ・サービスに対してトレースを収集できます。あるいは、ObjectGrid 名およびその他のプロパティに基づいてサーバーをフィルタリングすることもできます。例えば、カタログ・サービス・サーバーへのアクセスを使用して ObjectGridReplication トレースを収集するには、以下を実行します。

```
xscmd -c setTraceSpec "ObjectGridReplication=all=enabled"
```

トレース仕様を **\*=all=disabled** に設定することでトレースを使用不可にすることもできます。

## タスクの結果

トレース・ファイルは、指定された場所に作成されます。

## トレース・オプション

トレースを使用可能にすることで、ご使用の環境に関する情報を IBM サポートに提供することができます。

### トレースについて

WebSphere eXtreme Scale のトレースは、いくつかの異なるコンポーネントに分けられます。使用するトレース・レベルを指定できます。一般的なトレースのレベルには、all、debug、entryExit、および event があります。

トレース・ストリングの例は、以下のとおりです。

```
ObjectGridComponent=level=enabled
```

トレース・ストリングは連結することができます。\* (アスタリスク) 記号を使用してワイルドカード値を指定します (例: ObjectGrid\*=all=enabled)。トレースを IBM サポートに提供する必要がある場合は、特定のトレース・ストリングが要求されます。例えば、レプリカ生成に関する問題が発生した場合には、ObjectGridReplication=debug=enabled トレース・ストリングが要求される可能性があります。

### トレース仕様

#### ObjectGrid

汎用コア・キャッシュ・エンジン。

#### ObjectGridCatalogServer

汎用カタログ・サービス。

#### ObjectGridChannel

静的デプロイメント・トポロジー通信。



**ObjectGridClientInfo**

DB2 クライアント情報。

**ObjectGridClientInfoUser**

DB2 ユーザー情報。

**ObjectgridCORBA**

動的デプロイメント・トポロジー通信。

**ObjectGridDataGrid**

AgentManager API。

**ObjectGridDynaCache**

WebSphere eXtreme Scale 動的キャッシュ・プロバイダー

**ObjectGridEntityManager**

EntityManager API。Projector オプションとともに使用。

**ObjectGridEvictors**

ObjectGrid 組み込み Evictor。

**ObjectGridJPA**

Java Persistence API (JPA) ローダー

**ObjectGridJPACache**

JPA キャッシュ・プラグイン

**ObjectGridLocking**

ObjectGrid キャッシュ・エントリー・ロック・マネージャー。

**ObjectGridMBean**

管理 Bean

**ObjectGridMonitor**

ヒストリカル・モニター・インフラストラクチャー。

**ObjectGridNative**

WebSphere eXtreme Scale ネイティブ・コード・トレース。eXtremeMemory  
ネイティブ・コードを含む。

**ObjectGridOSGi**

WebSphere eXtreme Scale OSGi 統合コンポーネント。

**ObjectGridPlacement**

カタログ・サーバー断片配置サービス。

**ObjectGridQuery**

ObjectGrid 照会。

**ObjectGridReplication**

レプリケーション・サービス。

**ObjectGridRouting**

クライアント/サーバー・ルーティングの詳細。

**ObjectGridSecurity**

セキュリティー・トレース。

**ObjectGridSerializer**

DataSerializer プラグイン・インフラストラクチャー。

**ObjectGridStats**

ObjectGrid 統計。

**ObjectGridTransactionManager**

WebSphere eXtreme Scale トランザクション・マネージャー。

**ObjectGridWriteBehind**

ObjectGrid 後書き。

**ObjectGridXM**

一般 IBM eXtremeMemory トレース。

**ObjectGridXMEviction**

eXtremeMemory 除去トレース。

**ObjectGridXMTransport**

eXtremeMemory 一般トランスポート・トレース。

**ObjectGridXMTransportInbound**

eXtremeMemory インバウンド特定のトランスポート・トレース。

**ObjectGridXMTransportOutbound**

eXtremeMemory アウトバウンド特定のトランスポート・トレース。

**Projector**

EntityManager API 内のエンジン。

**QueryEngine**

オブジェクト照会 API および EntityManager 照会 API のための照会エンジン。

**QueryEnginePlan**

照会計画トレース。

**TCPChannel**

IBM eXtremeIO TCP/IP チャンネル。

**XsByteBuffer**

WebSphere eXtreme Scale バイト・バッファ・トレース。

---

## ログおよびトレース・データの分析

ログ分析ツールを使用して、ランタイムのパフォーマンスを分析したり、環境内で発生した問題を解決したりできます。

### このタスクについて

環境内の既存のログ・ファイルやトレース・ファイルからレポートを生成できます。これらのビジュアル・レポートには次のような用途があります。

- ランタイム環境の状況およびパフォーマンスの分析
  - デプロイメント環境の整合性
  - ログの頻度
  - 実行中トポロジーと構成されているトポロジーの比較
  - 予定外のトポロジー変更
  - クォーラム状況

- 区画のレプリカ生成の状況
- メモリー、スループット、プロセッサ使用量などの統計
- **環境内の問題のトラブルシューティング**
  - 特定時点でのトポロジー・ビュー
  - クライアント障害時のメモリー、スループット、プロセッサ使用量などの統計
  - 現在のフィックスバック・レベル、チューニング設定
  - クォーラム状況

## ログ分析の概要

環境内の問題のトラブルシューティングに役立つ **xsLogAnalyzer** ツールを使用できます。

### すべてのフェイルオーバー・メッセージ

フェイルオーバー・メッセージの総数を一定時間のチャートで表示します。また、影響を受けたサーバーを含む、フェイルオーバー・メッセージのリストも表示します。

### すべての eXtreme Scale 重大メッセージ

メッセージ ID を関連する説明およびユーザー処置と一緒に表示します。これにより、メッセージを検索する時間を節約できます。

### すべての例外

メッセージ、発生回数、例外の影響を受けたサーバーも含めて、上位 5 つの例外を表示します。

### トポロジーの要約

ログ・ファイルに基づいて、どのようにトポロジーが構成されているかをダイアグラムで表示します。この要約を使用して実際の構成と比較することができ、構成エラーを特定できる場合があります。

### トポロジーの整合性: オブジェクト・リクエスト・ブローカー (ORB) 比較表

環境での ORB 設定を表示します。環境全体で設定が整合しているか判別するのを助けるために、この表を使用できます。

### イベント・タイムライン・ビュー

データ・グリッドで発生したライフサイクル・イベント、例外、重大なメッセージ、初期障害データ・キャプチャー機能 (FFDC) イベントなどのさまざまなアクションのタイムライン図を表示します。

## ログ分析の実行

任意のコンピューターのログ・ファイルやトレース・ファイルのセットに対して **xsLogAnalyzer** ツールを実行できます。

### 始める前に

- ログおよびトレースを使用可能にします。詳しくは、617 ページの『ロギング可能化』と 618 ページの『トレースの収集』を参照してください。
  - ログ・ファイルを収集します。ログ・ファイルを構成した方法に応じて、ログ・ファイルはさまざまな場所にあります。デフォルトのログ設定を使用している場合は、次の場所からログ・ファイル入手できます。
    - スタンドアロン・インストールの場合: `wxs_install_root/bin/logs/<server_name>`
    - WebSphere Application Server と統合されたインストールの場合: `was_root/logs/<server_name>`
  - トレース・ファイルを収集します。トレース・ファイルを構成した方法に応じて、トレース・ファイルはさまざまな場所にあります。デフォルトのトレース設定を使用している場合は、次の場所からトレース・ファイル入手できます。
    - スタンドアロン・インストールの場合: 特定のトレース値を設定していない場合、トレース・ファイルはシステム出力のログ・ファイルと同じ場所に作成されます。
    - WebSphere Application Server と統合されたインストールの場合: `was_root/profiles/server_name/logs`
- ログ・アナライザー・ツールを使用する予定のコンピューターにログ・ファイルとトレース・ファイルをコピーします。
- 生成されるレポートのカスタム・スキャナーを作成する場合は、ツールを実行する前にスキャナー仕様プロパティ・ファイルと構成ファイルを作成してください。詳しくは、625 ページの『ログ分析用カスタム・スキャナーの作成』を参照してください。

### 手順

1. **xsLogAnalyzer** ツールを実行します。

スクリプトは次の場所にあります。

- スタンドアロン・インストールの場合: `wxs_install_root/ObjectGrid/bin`
- WebSphere Application Server と統合されたインストールの場合: `was_root/bin`

**ヒント:** ログ・ファイルが大きい場合、レポートを実行するときに

**-startTime**、**-endTime**、および **-maxRecords** パラメーターを使用して、スキャンするログ・エントリーの数を制限することを検討してください。レポートを実行するときにこれらのパラメーターを使用すると、レポートが見やすくなるうえ、レポートをより効率的に実行できます。同一セットのログ・ファイルを対象に複数のレポートを実行できます。

```
xsLogAnalyzer.sh|bat -logsRoot c:%myxlogs -outDir c:%myxlogs%out  
-startTime 11.09.27_15.10.56.089 -endTime 11.09.27_16.10.56.089 -maxRecords 100
```

#### **-logsRoot**

評価するログ・ディレクトリーへの絶対パスを指定します (必須)。

### **-outDir**

レポートの出力を書き込む既存のディレクトリーを指定します。値を指定しないと、レポートは **xsLogAnalyzer** ツールのルート・ロケーションに書き込まれます。

### **-startTime**

ログ内の評価する開始時刻を指定します。日付のフォーマットは、*year.month.day\_hour.minute.second.millisecond* です。

### **-endTime**

ログ内の評価する終了時刻を指定します。日付のフォーマットは、*year.month.day\_hour.minute.second.millisecond* です。

**-trace** トレース・ストリング (ObjectGrid\*=all=enabled など) を指定します。

### **-maxRecords**

レポート内に生成するレコードの最大数を指定します。デフォルトは 100 です。値を 50 と指定した場合、指定された期間の最初の 50 レコードが生成されます。

2. 生成されたファイルを開きます。出力ディレクトリーを定義しなかった場合、レポートは *report\_date\_time* というフォルダー内に生成されます。レポートのメインページを開くには、*index.html* ファイルを開きます。
3. レポートを使用して、ログ・データを分析します。次のヒントを使用して、レポート表示のパフォーマンスを最大にしてください。
  - ログ・データの照会のパフォーマンスを最大にするには、できるだけ具体的な情報を使用します。例えば、*server\_host\_name* の照会より *server* の照会のほうが実行時間が長くなり、返される結果も多くなります。
  - 一部のビューでは、一度に表示されるデータ・ポイントの数が制限されます。ビュー内の現在のデータを変更して (開始時刻や終了時刻を変更するなどして)、表示される時間セグメントを調整できます。

## 次のタスク

**xsLogAnalyzer** ツールや生成されるレポートのトラブルシューティングの詳細については、627 ページの『ログ分析のトラブルシューティング』を参照してください。

## ログ分析用カスタム・スキャナーの作成

ログ分析用のカスタム・スキャナーを作成できます。スキャナーを構成してから、**xsLogAnalyzer** ツールを実行すると、結果がレポート内に生成されます。カスタム・スキャナーは、指定された正規表現に基づいてイベント・レコードのログをスキャンします。

### 手順

1. カスタム・スキャナーで実行する正規表現を指定したスキャナー仕様プロパティ・ファイルを作成します。
  - a. プロパティ・ファイルを作成し、保存します。ファイルは *loganalyzer\_root/config/custom* ディレクトリー内に存在しなければなりません。ファイルには好きな名前を付けることができます。ファイルは新規ス

キャナーで使用されるので、スキャナーの名前をプロパティー・ファイルの名前的一部分にすると (例えば、`my_new_server_scanner_spec.properties`) 便利です。

- b. 次のプロパティーを `my_new_server_scanner_spec.properties` ファイルに組み込みます。

```
include.regular_expression = REGULAR_EXPRESSION_TO_SCAN
```

`REGULAR_EXPRESSION_TO_SCAN` 変数は、ログ・ファイルのフィルタリングに使用する正規表現です。

例: `xception` と `rror` 両方のストリングを含んでいる行 (ストリングの出現順序は問いません) のインスタンスをスキャンするには、

**include.regular\_expression** プロパティーを次の値に設定します。

```
include.regular_expression = (xception.+rror)|(rror.+xception)
```

ストリング `xception` の前か後にストリング `rror` が出現すると、この正規表現によってイベントが記録されます。

例:ログ内の各行をスキャンして、句 `xception` または句 `rror` のいずれかのストリングを含んでいる行のインスタンスを探すには、

**include.regular\_expression** プロパティーを次の値に設定します。

```
include.regular_expression = (xception)|(rror)
```

`rror` ストリングまたは `xception` ストリングのいずれかが存在する場合、この正規表現によってイベントが記録されます。

2. **xsLogAnalyzer** ツールがスキャナーを作成するために使用する構成ファイルを作成します。
  - a. 構成ファイルを作成し、保存します。ファイルは `loganalyzer_root/config/custom` ディレクトリー内に存在しなければなりません。ファイルの名前は、`scanner_nameScanner.config` のようにします。ここで、`scanner_name` は、新規スキャナーの固有の名前です。例えば、このファイルは `serverScanner.config` という名前にできます。
  - b. 次のプロパティーを `scanner_nameScanner.config` ファイルに組み込みます。

```
scannerSpecificationFiles = LOCATION_OF_SCANNER_SPECIFICATION_FILE
```

`LOCATION_OF_SCANNER_SPECIFICATION_FILE` 変数は、前のステップで作成した仕様ファイルの場所 (パス) です。例: `loganalyzer_root/config/custom/my_new_scanner_spec.properties`。セミコロンで区切ったリストを使用して、複数のスキャナー仕様ファイルを指定することもできます。

```
scannerSpecificationFiles = LOCATION_OF_SCANNER_SPECIFICATION_FILE1;LOCATION_OF_SCANNER_SPECIFICATION_FILE2
```

3. **xsLogAnalyzer** ツールを実行します。詳しくは、624 ページの『ログ分析の実行』を参照してください。

## タスクの結果

**xsLogAnalyzer** ツールを実行すると、構成したカスタム・スキャナー用の新しいタブがレポートに含まれています。各タブには、次のビューがあります。



## チャート

記録されたイベントを示すプロット・グラフ。イベントは検出された順序で表示されます。

## テーブル

記録されたイベントのテーブル表示。

## 要約レポート

# ログ分析のトラブルシューティング

**xsLogAnalyzer** ツールおよびこのツールで生成されるレポートに関する問題を診断し、修正する場合、次のトラブルシューティング情報を使用してください。

## 手順

- **問題:** **xsLogAnalyzer** ツールを使用してレポートを生成中、メモリー不足状態が発生する。発生する可能性があるエラーの例は、次のとおりです：  
`java.lang.OutOfMemoryError: GC overhead limit exceeded.`

**解決策:** **xsLogAnalyzer** ツールは Java 仮想マシン (JVM) 内で実行されます。

**xsLogAnalyzer** ツールの実行時に、ある設定を指定することで、ヒープ・サイズを大きくするよう JVM を構成してから、ツールを実行することができます。ヒープ・サイズを大きくすることで、より多くのイベント・レコードを JVM メモリー内に保管できるようになります。オペレーティング・システムに十分なメイン・メモリーがあれば、最初は 2048M を設定してはじめてください。

**xsLogAnalyzer** ツールを実行するのと同じコマンド行インスタンスで、次のように最大 JVM ヒープ・サイズを設定します。

```
java -XmxHEAP_SIZEm
```

*HEAP\_SIZE* 値は、任意の整数にでき、JVM ヒープに割り振られるメガバイト数を表します。例えば、`java -Xmx2048m` を実行できます。メモリー不足メッセージが続く場合、または 2048m 以上のメモリーを割り振るためのリソースがない場合は、ヒープ内に保持するイベントの数を制限してください。**-maxRecords** パラメーターを **xsLogAnalyzer** コマンドに渡すと、ヒープ内のイベントの数を制限できます。

- **問題:** **xsLogAnalyzer** ツールで生成されたレポートを開くと、ブラウザーがハングするか、ページが読み込まれない。

**原因:** 生成された HTML ファイルが大きすぎるため、ブラウザーが読み込むことができません。これらのファイルが大きすぎる理由は、分析対象のログ・ファイルの範囲が広すぎるためです。

**解決策:** **xsLogAnalyzer** ツールを実行するときに **-startTime**、**-endTime**、および **-maxRecords** パラメーターを使用して、スキャンするログ・エントリーの数を制限することを検討してください。レポートを実行するときにこれらのパラメーターを使用すると、レポートが見やすくなるうえ、レポートをより効率的に実行できます。同一セットのログ・ファイルを対象に複数のレポートを実行できます。

## インストールのトラブルシューティング

インストールおよび更新の問題をトラブルシューティングする場合、この情報を使用してください。

### 手順

- **問題:** インストール・コマンドをリモート・コンピューター (¥¥mymachine¥downloads¥ など) から実行すると、次のメッセージが表示される: 現在のディレクトリとして上記のパスで CMD.EXE を開始しました。UNC パスはサポートされません。 Windows ディレクトリを既定で使用します。 結果として、インストールが正常に完了しない。

**解決策:** リモート・コンピューターをネットワーク・ドライブにマップしてください。例えば、Windows の場合、「マイ コンピュータ」を右クリックし、「ネットワーク ドライブの割り当て」を選択して、リモート・コンピューターへの汎用名前付け規則 (UNC) パスを組み込むことができます。その後は、ネットワーク・ドライブから正常にインストール・スクリプトを実行できます。例:

```
y:¥¥mymachine¥downloads¥WXS¥install.bat.
```

- **問題:** インストールが正常に完了しない。

**解決策:** ログ・ファイルをチェックして、インストールがどこで失敗したか確認します。インストールが正常に完了しない場合は、ログは `wxs_install_root/logs/wxs` ディレクトリーにあります。

- **問題:** インストール時に重大な障害が発生する。

**解決策:** ログ・ファイルをチェックして、インストールがどこで失敗したか確認します。インストールが部分的に完了した時点でインストールが失敗した場合、ログは一般的に `user_root/wxs_install_logs/` ディレクトリーにあります。

- **Windows** **問題:** Windows 上で WebSphere eXtreme Scale クライアント をインストールする場合、インストールの結果に次のテキストが表示されることがある。

```
Success: The installation of the following product was successful:  
WebSphere eXtreme Scale Client. Some configuration steps have errors.  
For more information, refer to the following log file:  
<WebSphere Application Server install root>¥logs¥wxs_client¥install¥log.txt"  
Review the installation log (log.txt) and review the deployment manager  
augmentation log.
```

**解決策:** `iscdeploy.sh` ファイルで発生した障害の場合、エラーは無視してかまいません。このエラーでは、問題は生じません。

- **問題:**

フルインストールの環境で、WebSphere eXtreme Scale クライアント に対してアップデート・インストーラーを使用して保守のみを適用しようとする、以下のメッセージが表示されます。

```
Prerequisite checking has failed. Click Back to select a different package,  
or click Cancel to exit.
```

Failure messages are:

```
Required feature wxs.client.primary is not found.
```

WebSphere eXtreme Scale クライアント がインストールされていて、アップデート・インストーラーを使用してフル保守パッケージを適用しようとする、次のメッセージが表示されます。

Prerequisite checking has failed. Click Back to select a different package, or click Cancel to exit.

Failure messages are:

Required feature wxs.primary is not found.

**解決策:** インストールする保守パッケージはインストール済み環境のタイプと一致している必要があります。ご使用のインストール済み環境のタイプに適合する保守パッケージをダウンロードし、適用してください。

---

## キャッシュ統合のトラブルシューティング

HTTP セッションや動的キャッシュ構成など、キャッシュ統合構成の問題をトラブルシューティングする場合、この情報を使用してください。

### 手順

- **問題:** HTTP セッション ID が再使用されない。

**原因:** セッション ID は再使用できます。セッション・パーシスタンスのデータ・グリッドをバージョン 7.1.1 以上で作成すると、セッション ID の再使用は自動的に有効になります。しかし、それより前の構成で作成した場合、この設定が既に誤った値で設定されている可能性があります。

**解決策:** 次の設定をチェックして、HTTP セッション ID の再使用が有効であるか確認します。

- splicer.properties ファイルの reuseSessionId プロパティは、true に設定する必要があります。
- HttpSessionIdReuse カスタム・プロパティ値は、true に設定する必要があります。このカスタム・プロパティは、WebSphere Application Server 管理コンソールの次のいずれかのパスで設定されている可能性があります。
  - 「サーバー」 > 「*server\_name*」 > 「セッション管理」 > 「カスタム・プロパティ」
  - 「動的クラスター」 > 「*dynamic\_cluster\_name*」 > 「サーバー・テンプレート」 > 「セッション管理」 > 「カスタム・プロパティ」
  - 「サーバー」 > 「サーバー・タイプ」 > 「WebSphere Application Server」 > 「*server\_name*」を選択してから、「サーバー・インフラストラクチャー」セクションの下で次の順にクリックします。「Java およびプロセス管理」 > 「プロセス定義」 > 「Java 仮想マシン」 > 「カスタム・プロパティ」
  - 「サーバー」 > 「サーバー・タイプ」 > 「WebSphere Application Server」 > 「*server\_name*」 > 「Web コンテナ設定」 > 「Web コンテナ」

カスタム・プロパティ値を更新した場合、eXtreme Scale セッション管理を再構成し、splicer.properties ファイルが変更を認識できるようにしてください。

- **問題:** データ・グリッドを使用して HTTP セッションを保管する際、トランザクションの負荷が高いと SystemOut.log ファイルに CWOBJ0006W メッセージが出力される。

CWOBJ0006W: 例外が発生しました:  
com.ibm.websphere.objectgrid.ObjectGridRuntimeException:  
java.util.ConcurrentModificationException

このメッセージは、splicer.properties ファイル内の **replicationInterval** パラメーターの値がゼロより大きい値に設定されていて、かつ Web アプリケーションが HttpSession の属性として設定された List オブジェクトを変更した場合にのみ発生します。

**解決策:** 変更された List オブジェクトを含んでいる属性を複製し、複製した属性をセッション・オブジェクトに設定します。

---

## JPA キャッシュ・プラグインのトラブルシューティング

JPA キャッシュ・プラグイン構成の問題をトラブルシューティングする場合、この情報を使用してください。これらの問題は、Hibernate 構成と OpenJPA 構成のどちらでも発生する可能性があります。

### 手順

- **問題:** 次の例外が表示される: CacheException: ObjectGrid サーバーを取得できません。

EMBEDDED または EMBEDDED\_PARTITION **ObjectGridType** 属性値を指定している場合、eXtreme Scale キャッシュは、ランタイムからサーバー・インスタンスを取得しようとしています。Java Platform, Standard Edition 環境では、組み込みカタログ・サービスを持つ eXtreme Scale サーバーが始動されます。組み込みカタログ・サービスは、ポート 2809 を listen しようとしています。そのポートを別のプロセスが使用している場合、エラーが発生します。

**解決策:** 外部カタログ・サービス・エンドポイントが、例えば、objectGridServer.properties ファイルにより指定されている場合、ホスト名またはポートの指定に誤りがあると、このエラーが発生します。ポートの競合を修正してください。

- **問題:** 次の例外が表示される: CacheException: 構成済みの REMOTE ObjectGrid に対して REMOTE ObjectGrid を取得できません。objectGridName = [ObjectGridName]、PU 名 = [persistenceUnitName]

このエラーは、指定されたカタログ・サービス・エンドポイントからキャッシュが ObjectGrid インスタンスを取得できないために発生します。

**解決策:** この問題は、一般的にホスト名またはポートに誤りがあるために発生します。

- **問題:** 次の例外が表示される: CacheException: 2 つの PU [persistenceUnitName\_1, persistenceUnitName\_2] を EMBEDDED ObjectGridType の同一の ObjectGridName [ObjectGridName] では構成できません。

多数のパーシスタンス・ユニットが構成されている場合に、これらのユニットの eXtreme Scale キャッシュが同じ ObjectGrid 名および EMBEDDED ObjectGridType 属性値で構成されていると、この例外が発生します。これらのパーシスタンス・ユニット構成は、同じまたは異なる persistence.xml ファイルに入れることができます。

**解決策:** ObjectGridType 属性値が EMBEDDED の場合、各パーシスタンス・ユニットの ObjectGrid 名が固有であることを確認する必要があります。

- **問題:** 次の例外が表示される: CacheException: REMOTE ObjectGrid [ObjectGridName] に必要な BackingMaps [mapName\_1, mapName\_2,...] が含まれていません。

REMOTE ObjectGrid タイプの場合、取得されたクライアント・サイド ObjectGrid に、パーシスタンス・ユニットのキャッシュをサポートするエンティティ・バックアップ・マップが完全に揃っていないと、この例外が発生します。例えば、パーシスタンス・ユニット構成に 5 つのエンティティ・クラスがリストされているが、取得された ObjectGrid には 2 つの BackingMap しかない場合などです。取得された ObjectGrid に 10 の BackingMap があつたとしても、必要な 5 つのエンティティ BackingMap のいずれかがその 10 の BackingMap 内で見つからないと、やはりこの例外が発生します。

**解決策:** バックアップ・マップ構成が、パーシスタンス・ユニットのキャッシュをサポートすることを確認してください。

---

## IBM eXtremeMemory および IBM eXtremeIO のトラブルシューティング

以下の情報を使用して、eXtremeMemory および eXtremeIO のトラブルシューティングを行います。

### 手順

**問題:** 共有リソース libstdc++.so.5 がインストールされていない場合、コンテナ・サーバーを始動したときに、IBM eXtremeMemory ネイティブ・ライブラリーがロードされない。

**Linux** **症状:** Linux 64 ビット・オペレーティング・システムで、enableXM サーバー・プロパティを true に設定してコンテナ・サーバーを始動しようとする、libstdc++.so.5 共有リソースがインストールされていなければ、次の例のようなエラーを受け取ります。

```
00000000 Initialization W CW0BJ0006W: An exception occurred: java.lang.reflect.InvocationTargetException
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:56)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:39)
at java.lang.reflect.Constructor.newInstance(Constructor.java:527)
at com.ibm.websphere.objectgrid.server.ServerFactory.initialize(ServerFactory.java:350)
at com.ibm.websphere.objectgrid.server.ServerFactory$2.run(ServerFactory.java:303)
at java.security.AccessController.doPrivileged(AccessController.java:202)
at com.ibm.websphere.objectgrid.server.ServerFactory.getInstance(ServerFactory.java:301)
at com.ibm.ws.objectgrid.InitializationService.main(InitializationService.java:302)
```

```
Caused by: com.ibm.websphere.objectgrid.ObjectGridRuntimeException: java.lang.UnsatisfiedLinkError:
OffheapMapdbg (Not found in java.library.path)
at com.ibm.ws.objectgrid.ServerImpl.<init>(ServerImpl.java:1033)
... 9 more Caused by: java.lang.UnsatisfiedLinkError: OffheapMapdbg (Not found in java.library.path)
at java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1011)
at java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:975)
at java.lang.System.loadLibrary(System.java:469)
at com.ibm.ws.objectgrid.io.offheap.ObjectGridHashTableOH.initializeNative(ObjectGridHashTableOH.java:112)
at com.ibm.ws.objectgrid.io.offheap.ObjectGridHashTableOH.<clinit>(ObjectGridHashTableOH.java:87)
```

```
at java.lang.J9VMInternals.initializeImpl(Native Method)
at java.lang.J9VMInternals.initialize(J9VMInternals.java:200)
at com.ibm.ws.objectgrid.ServerImpl.<init>(ServerImpl.java:1028)
... 9 more
```

**原因:** 共有リソース `libstdc++.so.5` がインストールされていません。

**問題の診断:** リソース `libstdc++.so.5` がインストールされていることを確認するには、インストール済み環境の `ObjectGrid/native` ディレクトリーから次のコマンドを発行します。

```
ldd lib0ffheapMap.so
```

共有ライブラリーをインストールしていない場合、次のエラーを受け取ります。

```
ldd lib0ffheapMap.so
libstdc++.so.5 => not found
```

**問題の解決:** 64 ビットの Linux ディストリビューションのパッケージ・インストーラーを使用して、必要なリソース・ファイルをインストールします。パッケージは、`compat-libstdc++-33.x86_64` または `libstdc++5` としてリストされていることもあります。必要なリソースをインストールした後、インストール済み環境の `ObjectGrid` ディレクトリーから次のコマンドを発行して、`libstdc++5` パッケージがインストールされていることを確認します。

```
ldd lib0ffheapMap.so
```

---

## 管理のトラブルシューティング

サーバーの開始や停止、`xscmd` ユーティリティーの使用など、管理についてのトラブルシューティングを行う場合、この情報を使用してください。

### 手順

- **問題:** WebSphere Application Server インストール済み環境の `profile_root/bin` ディレクトリーに管理スクリプトがない。

**原因:** インストール済み環境を更新しても、新しいスクリプト・ファイルは自動的にプロファイルにインストールされません。

**解決策:** `profile_root/bin` ディレクトリーからスクリプトを実行する必要がある場合、プロファイルを拡張解除し、最新のリリースで拡張し直してください。詳しくは、コマンド・プロンプトを使用したプロファイルの拡張解除と 225 ページの『WebSphere eXtreme Scale のプロファイルの作成および拡張』を参照してください。

- **問題:** `xscmd` コマンドの実行時に次のメッセージが画面に表示される。

```
java.lang.IllegalStateException: Placement service MBean not available.
[]
   at
com.ibm.websphere.samples.objectgrid.admin.OGAdmin.main(OGAdmin.java:1449)
   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
   at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
   at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37)
   at java.lang.reflect.Method.invoke(Method.java:611)
   at com.ibm.ws.bootstrap.WSLauncher.main(WSLauncher.java:267)
Ending at: 2011-11-10 18:13:00.000000484
```

**原因:** カタログ・サーバーで接続の問題が発生しました。



**解決策:** カタログ・サーバーが実行中であり、ネットワーク経由で使用可能なことを確認します。このメッセージは、カタログ・サービス・ドメインが定義されている場合に、実行中のカタログ・サーバーが 2 つより少ないとき発生することもあります。そのような環境は、2 つのカタログ・サーバーが開始されるまで使用できません。

- **8.5+** **問題:** `xscmd` コマンドの実行時に次のメッセージが画面に表示される。

CWXSIO066E: 一致しない引数 .. が検出されました

**原因:** 入力されたコマンド・フォーマットは `xscmd` ユーティリティーによって認識されませんでした。

**解決策:** コマンドのフォーマットを確認してください。 `-c findbyKey` コマンドを使用して正規表現を実行するときに、この問題が発生する可能性があります。詳しくは、483 ページの『データの照会と無効化』を参照してください。

---

## 複数データ・センター構成のトラブルシューティング

カタログ・サービス・ドメイン間のリンクなど、複数データ・センター構成に関する問題をトラブルシューティングする場合、この情報を使用してください。

### 手順

**問題:** 1 つ以上のカタログ・サービス・ドメインのデータが欠落している。例えば、`xscmd -c establishLink` コマンドを実行したとします。リンクされた各カタログ・サービス・ドメインのデータを見ると、例えば `xscmd -c showMapSizes` コマンドの結果とデータが異なるような場合があります。

**解決策:** この問題は、`xscmd -c showLinkedPrimaries` コマンドを使用してトラブルシューティングできます。このコマンドは、リンクされている外部プライマリーを含め、各プライマリー断片を表示します。

前述のシナリオの場合、`xscmd -c showLinkedPrimaries` コマンドを実行することで、最初のカタログ・サービス・ドメインのプライマリー断片は 2 番目のカタログ・サービス・ドメインのプライマリー断片にリンクされていても、2 番目のカタログ・サービス・ドメインから最初のカタログ・サービス・ドメインへのリンクが存在しないことを発見できることがあります。そのような場合、2 番目のカタログ・サービス・ドメインから最初のカタログ・サービス・ドメインに対し、`xscmd -c establishLink` コマンドを再実行することもできます。

---

## ローダーのトラブルシューティング

データベース・ローダーの問題をトラブルシューティングする場合、この情報を使用してください。

### 手順

- **問題:** ローダーがデータベースと通信できません。 `LoaderNotAvailableException` 例外が発生します。

**説明:** ローダー・プラグインは、バックエンド・データベースと通信できない場合、失敗することがあります。この障害は、データベース・サーバーまたはネットワーク接続がダウンしている場合に発生することがあります。後書きローダー

は、更新をキューに入れ、データ変更を定期的にローダーにプッシュしようと試みます。ローダーは、`LoaderNotAvailableException` 例外をスローして、データベース接続の問題があることを `ObjectGrid` ランタイムに通知しなければなりません。

**解決策:** ローダー実装で、データ障害または物理的ローダー障害を識別できるようになっている必要があります。データ障害は `LoaderException` または `OptimisticCollisionException` としてスローまたは再スローされる必要がありますが、物理的なローダーの障害は `LoaderNotAvailableException` としてスローまたは再スローされる必要があります。 `ObjectGrid` は、これら 2 つの例外を異なる方法で処理します。

- `LoaderException` が後書きローダーによってキャッチされると、後書きローダーはその例外を、重複キー障害などの障害とみなします。後書きローダーは、更新のバッチ処理を解除し、データ障害を分離するために一度に 1 レコードずつ更新しようとします。1 レコードの更新時に再度 `LoaderException` がキャッチされると、失敗した更新レコードが作成され、失敗した更新マップのログに記録されます。
- `LoaderNotAvailableException` が後書きローダーによってキャッチされると、データベース・エンドに接続できない (例えば、データベース・バックエンドがダウンしている、データベース接続が使用可能でない、ネットワークがダウンしているなど) ため、後書きローダーはそれを障害とみなします。後書きローダーは 15 秒待ってから、データベースへのバッチ更新を再実行します。

一般的な間違いは、`LoaderNotAvailableException` がスローされるべきなのに、`LoaderException` がスローされることです。後書きローダーでキューに入れられたすべてのレコードは、失敗更新レコードとなります。このような場合、バックエンド障害分離の目的が果たせなくなります。

- **問題:** WebSphere Application Server 内で OpenJPA ローダーと DB2 を使用すると、カーソルのクローズ例外が発生する。

DB2 からの次の例外が `org.apache.openjpa.persistence.PersistenceException` ログ・ファイルに出力されます。

```
[jcc][t4][10120][10898][3.57.82] Invalid operation: result set is closed.
```

**解決策:** デフォルトで、アプリケーション・サーバーは `resultSetHoldability` カスタム・プロパティを値 2 (`CLOSE_CURSORS_AT_COMMIT`) で構成します。このプロパティにより、DB2 はトランザクション境界でその `resultSet`/カーソルを閉じます。この例外を除去するには、カスタム・プロパティの値を 1 (`HOLD_CURSORS_OVER_COMMIT`) に変更してください。 WebSphere Application Server セルの次のパスで、`resultSetHoldability` カスタム・プロパティを設定します。「リソース」 > 「JDBC プロバイダー」 > 「DB2 Universal JDBC Driver Provider」 > 「データ・ソース」 > 「`data_source_name`」 > 「カスタム・プロパティ」 > 「新規」。

- **問題:** DB2 が次の例外を表示する: デッドロックまたはタイムアウトのため、現在のトランザクションがロールバックされました。理由コード "2"..  
SQLCODE=-911, SQLSTATE=40001, DRIVER=3.50.152

この例外が発生する原因は、WebSphere Application Server 内で OpenJPA と DB2 を実行中に生じるロック競合の問題です。WebSphere Application Server のデフォルトの分離レベルは反復可能読み取り (RR) で、これは DB2 の長期ロックを獲得します。**解決策:**

分離レベルを読み取りコミット済みに設定して、ロック競合を減らしてください。WebSphere Application Server セルの次のパスで、データ・ソースの `webSphereDefaultIsolationLevel` カスタム・プロパティを設定し、分離レベルを 2 (TRANSACTION\_READ\_COMMITTED) に設定します。「リソース」 > 「JDBC プロバイダー」 > 「JDBC\_provider」 > 「データ・ソース」 > 「data\_source\_name」 > 「カスタム・プロパティ」 > 「新規」。  
`webSphereDefaultIsolationLevel` カスタム・プロパティとトランザクション分離レベルの詳細については、データ・アクセスの分離レベル設定の要件を参照してください。

- **問題:** JPALoader または JPAEntityLoader のプリロード機能を使用している際、コンテナ・サーバー内の区画に対して、次の CWOBJ1511I メッセージが表示されない。CWOBJ1511I: GRID\_NAME:MAPSET\_NAME:PARTITION\_ID (プライマリー) が作動可能になっています。

代わりに、`preloadPartition` プロパティで指定された区画をアクティブにするコンテナ・サーバーで `TargetNotAvailableException` 例外が発生します。

**解決策:** JPALoader または JPAEntityLoader を使用してデータをマップにプリロードする場合、`preloadMode` 属性を `true` に設定してください。JPALoader または JPAEntityLoader の `preloadPartition` プロパティを 0 から `total_number_of_partitions - 1` の範囲の値に設定すると、JPALoader または JPAEntityLoader は、バックエンド・データベースからデータをマップにプリロードしようとします。以下のコード・スニペットは、非同期プリロードが有効になるよう `preloadMode` 属性を設定する方法を表しています。

```
BackingMap bm = og.defineMap( "map1" );  
bm.setPreloadMode( true );
```

`preloadMode` 属性は、次の例に示すように、XML ファイルを使用して設定することもできます。

```
<backingMap name="map1" preloadMode="true" pluginCollectionRef="map1"  
lockStrategy="OPTIMISTIC" />
```

---

## XML 構成のトラブルシューティング

eXtreme Scale の構成時に、XML ファイルで予想外の動作が発生する場合があります。次のセクションでは、起こり得る問題と解決策について説明します。

### 手順

- **問題:** デプロイメント・ポリシーと ObjectGrid XML ファイルは一致している必要があります。

デプロイメント・ポリシーと ObjectGrid XML ファイルは一致している必要があります。一致する ObjectGrid 名およびマップ名がない場合には、エラーが発生します。

ObjectGrid XML ファイル内の backingMap リストがデプロイメント・ポリシー XML ファイル内のマップ参照リストと一致しない場合は、カタログ・サーバーでエラーが発生します。

例えば、以下の ObjectGrid XML ファイルおよびデプロイメント・ポリシー XML ファイルはコンテナ・プロセスを開始する場合に使用されます。デプロイメント・ポリシー・ファイルには、ObjectGrid XML ファイルでリストされているマップ参照よりも多くのマップ参照があります。

**ObjectGrid.xml - incorrect example**

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="accounting">
      <backingMap name="payroll" readOnly="false" />
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

**deploymentPolicy.xml - incorrect example**

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="accounting">
    <mapSet name="mapSet1" numberOfPartitions="4" minSyncReplicas="1"
maxSyncReplicas="2" maxAsyncReplicas="1">
      <map ref="payroll"/>
      <map ref="ledger"/>
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

**メッセージ:** デプロイメント・ポリシーが ObjectGrid XML ファイルと非互換である場合、SystemOut.log ファイル内にエラー・メッセージが発生します。上記の例では、以下のようなメッセージが発生します。

```
CWOB3179E: ObjectGrid アカウンティング・デプロイメント記述子ファイルの mapSet mapSet1
内にあるマップ元帳参照が、ObjectGrid XML の有効なバックアップ・マップを参照していません。
```

デプロイメント・ポリシーで、ObjectGrid XML ファイル内にリストされた backingMap へのマップ参照が欠落している場合、SystemOut.log ファイル内にエラー・メッセージが発生します。以下に例を示します。

```
CWOB3178E: ObjectGrid XML 内で参照された ObjectGrid アカウンティングのマップ元帳が
デプロイメント記述子ファイル内で見つかりませんでした。
```

**解決策:** 正しいリストが入ったファイルを判別し、それに合わせて該当するコードを変更します。

- **問題:** XML ファイルの間で ObjectGrid 名が間違っており、これがエラーも引き起こします。

ObjectGrid の名前は ObjectGrid XML ファイルおよびデプロイメント・ポリシー XML ファイルの両方で参照されます。

**メッセージ:** IncompatibleDeploymentPolicyException の例外が原因となって、ObjectGridException が発生します。以下に例を示します。

原因: com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: objectGridName が "accountin" の objectgridDeployment には、ObjectGrid XML 内に対応する objectGrid がありません。

ObjectGrid XML ファイルは ObjectGrid 名のマスター・リストです。デプロイメント・ポリシーに ObjectGrid XML ファイルに含まれていない ObjectGrid 名がある場合、エラーが発生します。

**解決策:** ObjectGrid 名のスペルなど、詳細を確認します。ObjectGrid XML ファイルまたはデプロイメント・ポリシー XML ファイルに対し、余分な名前を除去するか、または欠落している ObjectGrid 名を追加します。このメッセージ例では、objectGridName が「accounting」のところが、ミススペルのため「accountin」になっています。

- **問題:** XML ファイルの一部の属性は一定の値のみを割り当てることができます。これらの属性には、スキーマによって列挙された許容値が含まれています。以下のリストには、属性の一部が挙げられています。
  - objectGrid エLEMENTの authorizationMechanism 属性
  - backingMap エLEMENTの copyMode 属性
  - backingMap エLEMENTの lockStrategy 属性
  - backingMap エLEMENTの ttlEvictorType 属性
  - property エLEMENTの type 属性
  - objectGrid エLEMENTの initialState
  - backingMap エLEMENTの evictionTriggers

これら属性の 1 つに無効値が割り当てられていると、XML 妥当性検査は失敗します。以下の XML ファイルの例では、正しくない値 INVALID\_COPY\_MODE が使用されています。

```
INVALID_COPY_MODE example
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="accounting">
      <backingMap name="payroll" copyMode="INVALID_COPY_MODE"/>
    </objectGrid/>
  </objectGrids>
</objectGridConfig>
```

次のメッセージがログに表示されます。

```
CWOBJ2403E: The XML file is invalid. A problem has been detected
with < null > at line 5. The error message is cvc-enumeration-valid:
Value 'INVALID_COPY_MODE' is not facet-valid with respect to enumeration
'[COPY_ON_READ_AND_COMMIT, COPY_ON_READ, COPY_ON_WRITE, NO_COPY, COPY_TO_BYTES]'.
It must be a value from the enumeration.
```

- **問題:** XML ファイル内の属性またはタグが欠落しているか間違っているため、エラーが発生します。例えば、次の例の ObjectGrid XML ファイルでは、閉じの </objectGrid > タグが欠落しています。

**missing attributes - example XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="accounting">
      <backingMap name="payroll" />
    </objectGrids>
  </objectGridConfig>
```

**メッセージ:**

```
CWOBj2403E: The XML file is invalid. A problem has been detected with
< null > at line 7. The error message is The end-tag for element type "objectGrid"
must end with a '>' delimiter.
```

無効な XML ファイルに関する `ObjectGridException` が、XML ファイルの名前で発生します。

**解決策:** XML ファイル内に、必要なタグと属性が正しい形式で指定されていることを確認します。

- **問題:** XML ファイルが、正しくない構文または欠落している構文でフォーマット済みである場合、`CWOBj2403E` がログに表示されます。例えば、XML 属性の 1 つで引用符が欠落している場合、次のメッセージが表示されます。

```
CWOBj2403E: The XML file is invalid. A problem has been detected with
< null > at line 7. The error message is Open quote is expected for attribute
"maxSyncReplicas" associated with an element type "mapSet".
```

また、無効な XML ファイルに関する `ObjectGridException` も発生します。

**解決策:** 示された XML 構文エラーには、さまざまな解決策を使用できます。XML スクリプトの作成について関係する資料を調べてください。

- **問題:** 存在しないプラグイン・コレクションを参照すると、XML ファイルが無効になります。例えば、XML を使用して `BackingMap` プラグインを定義する場合、`backingMap` エレメントの `pluginCollectionRef` 属性は `backingMapPluginCollection` を参照する必要があります。`pluginCollectionRef` 属性は `backingMapPluginCollection` エレメントと一致している必要があります。

**メッセージ:**

`pluginCollectionRef` 属性が `backingMapPluginConfiguration` エレメントのどの ID 属性とも一致しない場合は、以下のメッセージまたは同様のメッセージがログに表示されます。

```
[7/14/05 14:02:01:971 CDT] 686c060e XmlErrorHandl E CWOBj9002E:
This is an English only Error message: Invalid XML file. Line: 14; URI:
null; Message: Key 'pluginCollectionRef' with
value 'bookPlugins' not found for identity constraint of
element 'objectGridConfig'.
```

以下の XML ファイルを使用すると、エラーが生じます。`BackingMap` の名前 `book` の `pluginCollectionRef` 属性は `bookPlugins` に設定され、1 つの `backingMapPluginCollection` が `collection1` の ID を持つことに注意してください。

**referencing a non-existent attribute XML - example**

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="bookstore">
      <backingMap name="book" pluginCollectionRef="bookPlugin" />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="collection1">
      <bean id="Evictor"
        className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

**解決策:**



問題を修正するには、それぞれの `pluginCollectionRef` の値が `backingMapPluginCollection` エレメントのいずれか 1 つの ID に一致するようにしてください。このエラーが発生しないように、`pluginCollectionRef` の名前を `collection1` に変更するだけです。あるいは、既存の `backingMapPluginCollection` の ID を `pluginCollectionRef` に一致するように変更するか、または `pluginCollectionRef` に一致する ID を持つ追加の `backingMapPluginCollection` を追加してエラーを訂正します。

- **問題:** IBM Software Development Kit (SDK) バージョン 5 は、スキーマに基づく XML 妥当性検査に使用する Java API for XML Processing (JAXP) 機能の実装を含みます。この実装を含まない SDK を使用する場合、妥当性検査の試みが失敗する場合があります。

必要な実装を持たない SDK で XML の妥当性検査をしようとすると、ログに以下のエラーが表示されます。

```
XmlConfigBuild XML validation is enabled
SystemErr R com.ibm.websphere.objectgrid
SystemErr R at com.ibm.ws.objectgrid.ObjectGridManagerImpl.getObjectGridConfigurations
(ObjectGridManagerImpl.java:182)
SystemErr R at com.ibm.ws.objectgrid.ObjectGridManagerImpl.createObjectGrid(ObjectGridManagerImpl.java:309)
SystemErr R at com.ibm.ws.objectgrid.test.config.DocTest.main(DocTest.java:128)
SystemErr R Caused by: java.lang.IllegalArgumentException: No attributes are implemented
SystemErr R at org.apache.crimson.jaxp.DocumentBuilderFactoryImpl.setAttribute(DocumentBuilderFactoryImpl.java:93)
SystemErr R at com.ibm.ws.objectgrid.config.XmlConfigBuilder.<init>(XmlConfigBuilder.java:133)
SystemErr R at com.ibm.websphere.objectgrid.ProcessConfigXML$2.runProcessConfigXML.java:99)...
```

使用した SDK は、スキーマに対して XML ファイルの妥当性検査をするのに必要な JAXP 機能の実装を含んでいません。

**解決策:** この実装を含まない SDK を使用して XML の妥当性検査を行う場合は、Apache Xerces をダウンロードして、その Java アーカイブ (JAR) ファイルをクラスパスに組み込みます。この問題を回避するために、Xerces をダウンロードして JAR ファイルをクラスパスに組み込むと、XML ファイルを正常に妥当性検査できます。

---

## セキュリティのトラブルシューティング

この情報を使用して、セキュリティ構成に関する問題のトラブルシューティングを行います。

### 手順

- **問題:** 接続のクライアント・エンドは、`transportType` 設定値が `SSL-Required` に設定された Secure Sockets Layer (SSL) を必要とします。しかし、接続のサーバー・エンドは、SSL をサポートしておらず、`transportType` 設定値が `TCP/IP` に設定されています。この結果として、次の例外が発生し、それにより別の例外が連鎖して発生したことが、ログ・ファイルに記録されます。

```
java.net.ConnectException: connect: Address is invalid on local machine, or
port is not valid on remote machine
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:389)
at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:250)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:237)
at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:385)
at java.net.Socket.connect(Socket.java:540)
at
com.ibm.rmi.transport.TCPTransportConnection.createSocket(TCPTransportConnection.java:155)
at
com.ibm.rmi.transport.TCPTransportConnection.createSocket(TCPTransportConnection.java:167)
```

この例外にあるアドレスは、カタログ・サーバー、コンテナ・サーバー、またはクライアントのアドレスである可能性があります。

**解決策:** 586 ページの『セキュア・トランスポート・タイプの構成』にあるクライアントとサーバー間の有効なセキュリティ構成の表を参照してください。

- エージェントが使用されるときに、クライアントは、サーバーにエージェント呼び出しを送信します。そして、サーバーは、応答をクライアントに送り返してエージェント呼び出しを確認します。エージェントが処理を終了すると、サーバーは接続を開始してエージェントの結果を送信します。これは、接続の観点から、コンテナ・サーバーをクライアントにします。したがって、TLS/SSL が構成されている場合、クライアントの公開証明書がサーバーのトラストストアにインポートされることを確認してください。

---

## IBM Support Assistant for WebSphere eXtreme Scale

データの収集、症状の分析、製品情報の入手に IBM Support Assistant を使用することができます。

### IBM Support Assistant Lite

IBM Support Assistant Lite for WebSphere eXtreme Scale は、問題判別シナリオのための自動データ収集および症状分析支援を提供します。

IBM Support Assistant Lite を使用することで、適切な信頼性、可用性、保守性のトレース・レベルを設定して (トレース・レベルはツールにより自動的に設定されます) 問題を再現するのにかかる時間を短縮し、問題判別を合理化できます。さらに支援が必要であれば、IBM Support Assistant Lite は適切なログ情報を IBM サポートに送信するために必要な労力も削減します。

IBM Support Assistant Lite は、WebSphere eXtreme Scale バージョン 7.1.0 の各インストール済み環境に組み込まれています。

### IBM Support Assistant

IBM® Support Assistant (ISA) を使用すると、製品、教育、およびサポートのリソースに素早くアクセスすることができます。これにより、IBM ソフトウェア製品に関し、IBM サポートに問い合わせをする必要なく、自力で質問に回答し、問題を解決することが容易になります。さまざまな製品固有のプラグインにより、インストール済みの特定の製品に合わせて IBM Support Assistant をカスタマイズすることができます。また、IBM Support Assistant は、IBM サポートが特定の問題の原因を判別するのに役立つシステム・データ、ログ・ファイルなどの情報を収集することもできます。

IBM Support Assistant はご使用のワークステーションにインストールするユーティリティで、WebSphere eXtreme Scale サーバー・システム自体に直接インストールするものではありません。Support Assistant のメモリーおよびリソース要件は、WebSphere eXtreme Scale サーバー・システムのパフォーマンスに悪影響を与える可能性があります。組み込まれたポータブル診断コンポーネントは、サーバーの通常の運用に対する影響を最小限に抑えるように設計されています。

IBM Support Assistant を使用すると、次のような点で役立ちます。

- 複数の IBM 製品にわたり、IBM およびそれ以外の知識と情報源の中で検索を行うことで、質問に回答し、問題を解決する。
- 製品固有の Web リソース (製品とサポートのホーム・ページ、カスタマー・ニュースグループおよびフォーラム、スキルとトレーニングのリソース、トラブルシューティングに関する情報、よくあるご質問など) から追加情報を見つける。
- Support Assistant で使用可能なターゲット診断ツールを使用して、製品固有の問題を診断するお客様の能力を高める。
- (汎用の、もしくは製品や症状に固有のデータを収集して) 診断データの収集を単純化し、お客様と IBM が問題を解決する助けとなる。
- カスタマイズされたオンライン・インターフェースを介して、IBM サポートに対する問題発生事象の報告を支援する。(前述の診断データやその他の情報を新規または既存の発生事象に添付する機能を含む。)

そして最後に、組み込まれたアップデーター機能を使用して、追加のソフトウェア製品や機能が使用可能になったときにそれらに対するサポートを取得することができます。IBM Support Assistant を WebSphere eXtreme Scale と併用するようにセットアップするには、まず IBM Support Assistant をインストールします。このときインストールには、IBM サポートの「概要」Web ページ ([http://www-947.ibm.com/support/entry/portal/Overview/Software/Other\\_Software/IBM\\_Support\\_Assistant](http://www-947.ibm.com/support/entry/portal/Overview/Software/Other_Software/IBM_Support_Assistant))からダウンロードしたイメージで提供されるファイルを使用します。次に、IBM Support Assistant を使用して、製品のアップデートを探し、あればインストールします。また、ご使用の環境にある他の IBM ソフトウェア用のプラグインが使用可能であれば、インストールすることもできます。IBM Support Assistant のさらに詳しい情報と最新バージョンが、IBM Support Assistant の Web ページで入手できます。

(<http://www.ibm.com/software/support/isa/>)



---

## 特記事項

本書に記載の製品、プログラム、またはサービスが日本においては提供されていない場合があります。日本で利用可能な製品、プログラム、またはサービスについては、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。IBM 製品、プログラムまたはサービスに代えて、IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM によって明示的に指定されたものを除き、他社の製品と組み合わせた場合の動作の評価と検証はお客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510  
東京都中央区日本橋箱崎町19番21号  
日本アイ・ビー・エム株式会社  
法務・知的財産  
知的財産権ライセンス渉外

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation  
Mail Station P300  
522 South Road  
Poughkeepsie, NY 12601-5400  
USA  
Attention: Information Requests

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。





---

## 商標

IBM、IBM ロゴおよび [ibm.com](http://ibm.com) は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、[www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml) の「Copyright and trademark information」をご覧ください。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会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。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。



# 索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

## [ア行]

アーキテクチャー (architecture)  
トポロジー 12  
後書き  
データベース統合 26  
アンインストール  
更新 215, 220  
製品ファイル  
コマンド行 221  
サイレント 222  
GUI 220  
イベント・ベースの妥当性検査 34  
イベント・リスナー (event listener) 265  
インストール  
製品ファイル  
コマンド行 190  
サイレント 194, 197  
GUI 185  
製品ファイルの取得 182  
トラブルシューティング 628  
IBM Installation Manager  
コマンド行 188  
サイレント 192  
GUI 183  
Installation Manager ファイルの取得  
182  
WebSphere Application Server 206  
WebSphere Application Server Network  
Deployment 206  
インストール後のタスク 235  
インライン・キャッシュ 22  
運用チェックリスト 254  
応答時間  
チューニング、ガーベッジ・コレクションの  
Real Time 570  
Real Time  
スタンドアロン環境 570  
オブジェクト・リクエスト・ブローカー  
(Object Request Broker (ORB))  
カスタム構成 333  
オブジェクト・リクエスト・ブローカー  
(ORB)  
構成 331

オブジェクト・リクエスト・ブローカー  
(ORB) (続き)  
スタンドアロン eXtreme Scale 332  
orb.properties ファイル 560  
properties 560  
WebSphere Application Server 331  
オペレーティング・システム  
tuning 559  
オンライン 477

## [カ行]

開始  
カタログ・サーバー 457  
カタログ・サービス 457  
コンテナ・サーバー 457  
プログラマチック 468  
REST データ・サービス用のサーバー  
421  
servers 451  
カスタム・プロパティ  
ORB プロパティ 560  
カタログ・サーバー  
構成 285  
カタログ・サービス  
開始、WebSphere Application Server を  
実行していない環境での 451  
カタログ・サービス・ドメイン 467  
高可用性 (high availability) 286  
構成 285  
ベスト・プラクティス 286  
cluster 286  
WebSphere Application Server 288  
WebSphere Application Server での開始  
467  
カタログ・サービス・ドメイン 286  
管理用タスク 290  
WebSphere Application Server 288  
可用性  
状態の管理 477  
可用性 区画 (AP) 38  
完全キャッシュ 21  
管理  
トラブルシューティング 632  
overview 451  
WebSphere Application Server 288  
キャッシュ  
分散 17  
ローカル 13  
embedded 16

キャッシュ統合  
構成 346  
トラブルシューティング 629  
キャパシティー・プランニング 65  
クォーラム  
オーバーライド 480  
構成 307  
区画単位 68  
組み込みキャッシュ 16  
クライアント  
無効化 (invalidation) 338  
overview 335  
XML 構成 336  
クライアント許可  
作成者限定アクセス 579  
custom 579  
JAAS 579  
クライアント接続  
administering  
JCA の使用 503  
クライアント/サーバー・セキュリティー  
トランスポート層セキュリティー  
(TLS) 585  
Secure Sockets Layer (SSL) 585  
TCP/IP 585  
グリッド許可 583  
計画 11, 559  
アプリケーション・デプロイメント  
11  
運用チェックリスト 254  
オペレーティング・システム 559  
ネットワーク設定 559  
計算  
区画数 65  
メモリー・サイズ設定 65  
更新 239  
製品ファイル 208  
コマンド行 210  
サイレント 213  
GUI 208  
WebSphere Application Server と  
WebSphere eXtreme Scale 243  
構成 53  
データ・センター・トポロジー 320  
メソッド 253  
overview 253  
構成ファイル  
デプロイメント・ポリシー・ゾーンの  
例 281  
Hibernate 400  
orb.properties ファイル 560

コヒーレント・キャッシュ 20  
コンテナ・サーバー  
開始 454  
構成 312  
overview 285  
配置 475  
WebSphere Application Server  
構成 313  
自動的に始動 314

## [サ行]

サーバーの停止 463  
サーバー・プロパティ  
enableXm 317  
maxXmSize 317  
xIOContainerTCPNonSecurePort 317  
サイド・キャッシュ  
データベース統合 22  
索引  
データ品質 35  
performance 35  
サポート 640  
時間ベース・データ・アップデーター  
408  
照会 483  
初期構成 235  
推奨されない API 248, 251  
スタンドアロン (stand-alone)  
オブジェクト・リクエスト・ブローカ  
ー (Object Request Broker  
(ORB)) 333  
REST 421  
スタンドアロン・サーバー  
開始 451  
スパス・キャッシュ 21  
製品概要  
製品統合  
WebSphere Application Server での  
91  
セキュア・サーバー  
開始 601  
停止 601, 604  
REST データ・サービス 592  
WebSphere Application Server 602  
セキュリティ・プロファイル 604  
セッション・マネージャー  
パーシスタンス、データ・グリッドへ  
の 351  
WebSphere Application Server 346,  
360  
セッション・マネージャーのインターオペ  
ラビリティ  
WebSphere 製品との 52  
接続ファクトリー  
構成 342

接続ファクトリー (続き)  
リソース参照の作成 345  
ゾーン  
コンテナ・サーバー 280  
全体に渡るストライピング 271  
ゾーンの例 271  
断片配置 271  
データ・センター 271  
デプロイメント・ポリシー記述子  
XML ファイル 281  
モニター (monitor) 284  
ルーティング 276  
WAN 上 271

## [タ行]

タイムアウト要求再試行 340  
他のサーバーとの統合 52  
チュートリアル 73  
エンドポイント間のセキュア通信 86  
カタログ・サーバー・セキュリティ  
構成 101  
カタログ・サーバー・セキュリティ  
の構成 99, 126  
許可 82  
許可の構成  
グループの 112  
許可を使用可能にする 109, 138  
ユーザーの 110, 139  
クライアント許可 73  
クライアント認証 77, 78  
クライアント・アプリケーションの始  
動  
OSGi フレームワーク内 157  
クライアント・オーセンティケーター  
73  
クライアント・サーバー・セキュリテ  
ィー  
構成 98  
クライアント・セキュリティの構成  
125  
構成ファイル 147  
混合環境の計画 117  
コンテナ・サーバー・セキュリテ  
ィーの構成 129  
サービス・ランキングの検出 160  
サービス・ランキングの更新 161  
サービス・ランキングの照会 158  
サンプル OSGi バンドル 145  
サンプルのインストール 102, 131  
サンプルの実行 102, 108, 131, 137  
サンプル・クライアントの実行  
OSGi 内 155  
製品セキュリティの統合  
WebSphere Application Server での  
90

チュートリアル (続き)  
セキュリティの統合  
混合環境での 115  
チュートリアル・ファイルのアクセス  
92, 117  
データ・グリッドとマップのモニター  
xscmd による 114, 141  
トポロジーの概要 92, 117  
トランスポートの構成  
アウトバウンド 106, 135  
インバウンド 106, 135  
トランスポート・セキュリティの構  
成 105, 134  
認証の構成  
混合環境での 123  
バンドルのインストール 150  
バンドルの開始 143  
バンドルの更新 158  
バンドルの照会 158  
非セキュアなサンプル 73, 75  
非セキュアの例 73  
Eclipse のセットアップ  
OSGi 用 156  
eXtreme Scale コンテナの構成 152  
eXtreme Scale サーバーの構成 152  
eXtreme Scale バンドルのインストー  
ル 150  
eXtreme Scale バンドルをインストー  
ルする準備 145  
Google Protocol Buffers のインストー  
ル 153  
JAAS 許可の使用 137  
JAAS 許可を使用 108  
OSGi  
クライアントの実行 155  
クライアントの始動 157  
クライアントを実行する Eclipse の  
セットアップ 156  
構成ファイル 147  
コンテナの構成 152  
サーバーの構成 152  
サービス・ランキングの検出 160  
サービス・ランキングの更新 161  
サービス・ランキングの照会 158  
サンプル・バンドル 145  
バンドルのアップグレード 158  
バンドルのインストール 150  
バンドルの開始 143, 150, 154  
バンドルの照会 158  
バンドルをインストールする準備  
145  
プロトコル・バッファのインスト  
ール 153  
overview 144  
OSGi バンドルの開始 154

チュートリアル (続き)  
overview  
サーバーとコンテナの開始 144  
SSL プロパティの追加 107, 135  
WebSphere Application Server 92  
WebSphere Application Server の構成  
96, 121  
WebSphere Application Server 用の構成  
98  
データベース  
後書きキャッシュ (write-behind  
cache) 26  
サイド・キャッシュ 22  
スパース・キャッシュおよび完全キャ  
ッシュ 21  
データの準備 30  
データのプリロード 30  
データベースの同期手法 32  
同期 32  
ライトスルー・キャッシュ  
(write-through cache) 23  
リードスルー・キャッシュ  
(read-through cache) 23  
データベース統合  
構成 405  
データ・グリッド  
構成 256  
データ・グリッド・セキュリティ  
トークン・マネージャー 584  
JSSE 584  
データ・センター  
構成 320  
障害の管理 480  
トポロジー構成 320  
停止  
プログラマチック 468  
ディレクトリー規則 63, 175  
デプロイメント・ポリシー  
構成 268  
統計  
使用可能化 522  
統計 API 525  
overview 505  
動的キャッシュ (dynamic cache)  
tuning 574  
動的キャッシュ・プロバイダー  
キャパシティー・プランニング 69  
構成 373  
トポロジー  
プラン 12  
installation 168  
トラブルシューティング  
インストール 628  
管理 632  
キャッシュ統合 629

トラブルシューティング (続き)  
製品ファイル  
インストール 236  
問題の特定、手法 609  
HTTP セッション 629  
トラブルシューティングおよびサポート  
概説 609  
既知の問題の検索 611  
トラブルシューティングの手法 609  
フィックスの入手 612  
Fix Central 613  
IBM サポート 614  
IBM サポートのサブスクリプション 616  
トランスポート 317  
構成 330  
eXtremeIO 317  
ORB 331  
トレース・データ 622

## [ナ行]

入門  
overview 1  
認証  
セキュリティの統合  
混合環境での 116  
ネットワーク 559  
ネットワーク・カード  
構成 330  
ネットワーク・ポート  
計画 53

## [ハ行]

配置 475  
パスワード  
Web コンソール 508  
パフォーマンス・チューニング 559  
パフォーマンス・モニター・インフラストラ  
クチャー  
使用可能化 530  
統計の取得 532  
モジュール 534  
ピアツーピア・レプリカ生成 260  
ファースト・ステップ・コンソール 226  
フィックス  
取得 612  
フェイルオーバー (failover)  
構成 310, 567  
複数データ・センター構成 633  
プラン  
installation 59, 167  
プロファイル  
拡張 225  
コマンドによる拡張 228

プロファイル (続き)  
コマンドによる作成 228  
作成 225  
非 root ユーザー 234  
UI による拡張 227  
UI による作成 226  
プロファイル管理ツール・プラグイン  
プロファイル拡張 227  
プロファイル作成 226  
overview 225  
分散キャッシュ 17  
分散デプロイメント  
構成 268  
並列トランザクション 68  
ベスト・プラクティス  
Real Time  
スタンドアロン環境 570  
変更の配布  
ピア JVM 261  
ポート  
構成 324  
スタンドアロン構成 324  
WebSphere Application Server 329

## [マ行]

マルチマスター・データ・グリッド・レプ  
リカ生成  
計画 38  
マルチマスター・レプリカ生成  
計画 38  
計画、ローダーの 44  
構成の計画 43  
設計の計画 46  
トポロジー 38  
無効化 (invalidation) 265, 483  
モニター  
エージェント 543  
統計 API 525  
統計モジュール 523  
ベンダー・ツールの概要 543  
CA Wily Introscope 550  
csv ファイル 518  
DB2 556  
Hyperic HQ 554  
overview 505  
Performance Monitoring Infrastructure  
(PMI) 529  
Tivoli Enterprise Monitoring Agent に  
よる 543

## [ヤ行]

要件  
ソフトウェア 59, 171

要件 (続き)

ハードウェア (hardware) 59, 171

## [ラ行]

ランタイム・ファイル

スタンドアロン (stand-alone) 173, 180

WebSphere Application Server 178

リスナー (listener)

Java Message Service (JMS) 265

利点

後書きキャッシング 26

レプリカ生成 (replication)

JMS イベント・リスナー 265

JMS による構成 260

ローカル・キャッシュ

ピア・レプリカ生成 14

ローカル・セキュリティ

使用可能化 601

ローカル・デプロイメント 256

ローダー

データベース 28

JPA 405

troubleshooting 633

ロールバック

製品ファイル

サイレント 218

GUI 216

ログ 617

ログ分析

概説 623

カスタム 625

実行 624

トラブルシューティング 627

ログ・エレメント 261

ログ・シーケンス 261

ログ・データ 622

ロック

オプティミスティック 258

なし 258

プログラマチックに構成 258

ベシミスティック 258

XML による構成 258

## A

AP 38

API

管理 468

組み込みサーバー (embedded server) 471

統計 525

AvailabilityState 477

MBean 525

API (続き)

StateManager 477

AvailabilityState API 477

## C

commands

manageprofiles 228

routetable 480

startOgServer 451

stopOgServer 451

teardown 466

CPU のサイズ設定

トランザクション 68

並列トランザクションの場合 68

CSV ファイル

統計定義 519

csv ファイル 518

## D

data

照会 483

無効化 483

DB2 556

## E

Eclipse Equinox

環境のセットアップ 198

enableXm プロパティ 317

Evictor

XML による構成 257

eXtreme IO 317

eXtreme Scale の概要 11

eXtreme メモリー 317

eXtremeIO

構成 317

eXtremeMemory

構成 317

## H

Hibernate

構成 396

XML による構成 400

HTTP セッション

splicer.properties ファイル 371

HTTP セッション・マネージャー

構成 346

構成用のパラメーター 368

WebSphere Application Server 346

WebSphere Virtual Enterprise 360

XML による構成 363

Hyperic HQ 554

## I

IBM Support Assistant 640

IBM Tivoli Monitoring 543

installation

計画 59, 167

トポロジー 163

overview 163

REST データ・サービス 200

types 163

Introscope 550

## J

Java EE

考慮事項 62, 174

Java Message Service (JMS)

イベント・リスナー (event

listener) 265

ピアツーピア・レプリカ生成 260

Java Persistence API (JPA)

キャッシュ・トポロジー

組み込み区画化 379, 386

embedded 379

remote 379

キャッシュ・プラグイン

概要 379

構成 386

構成

embedded 386

overview 405

remote 386

時間ベース・データ・アップデーター

構成 408

Java SE

考慮事項 61, 173

Java 仮想マシン 564

JCA

administering

クライアント接続 503

JDK

考慮事項 61, 173

JMS

ピアツーピア・レプリカ生成 260

JMX セキュリティのアクセス制御

セキュア・トランスポート 588

認証 588

JAAS サポート 588

JPA キャッシュ・プラグイン

troubleshooting 630

JVM 564

## M

Managed Bean 542

manageprofiles コマンド 225



maxXmSize プロパティ 317

## MBean

wsadmin 498, 541

## MBeans

アクセス、セキュリティーを使用可能にした 588

プログラマチック 498

を使用した管理 497

overview 542

migrating 239

migration 242

## O

offline 477

## OpenJPA

キャッシュ・プラグイン

構成 390

ObjectGrid XML ファイル

example 393

## ORB

構成 331

custom 333

WebSphere Application Server 331

## OSGi

サーバーの構成 447

サーバーの始動 486

サービスの管理 491

チュートリアル

クライアントの実行 155

クライアントの始動 157

クライアントを実行する Eclipse の

セットアップ 156

構成ファイル 147

コンテナの構成 152

サーバーの構成 152

サービス・ランキングの検出 160

サービス・ランキングの更新 161

サービス・ランキングの照会 158

サンプル・バンドル 145

バンドルのアップグレード 158

バンドルのインストール 150

バンドルの開始 150, 154

バンドルの実行 143

バンドルの照会 158

バンドルをインストールする準備  
145

プロトコル・バッファのインスト  
ール 153

overview 144

バンドルのインストール 203

プラグインのインストール 489

Eclipse Equinox 環境 198

## OSGi コンテナ

Apache Aries Blueprint 構成 445

OSGi プラグイン

構成 444

を使用した管理 494

## P

Performance Monitoring Infrastructure  
(PMI)

モニター 529

## PMI

モニター 529

preload 477

properties

オブジェクト・リクエスト・ブローカ  
ー (ORB) 560

## Q

quiesce 477

## R

Real Time

スタンドアロン環境 570

チューニング、ガーベッジ・コレクシ  
ョンの 570

WebSphere Application Server 572

REST データ・サービス

アプリケーション・サーバー  
構成 421

構成

overview 409

取得および更新、データの

overview 414

使用可能化

overview 411

スタンドアロン・データ・グリッド

開始 417

データ・グリッド (data grid)

開始 419

データ・モデル

overview 411

保護 (securing) 592

Apache Tomcat

開始 435

デプロイメント 432

ATOM フィード

構成 438

installation 200

Java クライアント

構成 440

Visual Studio 2008 WCF クライアント

構成 442

WebSphere Application Server

開始 424

REST データ・サービス (続き)

WebSphere Application Server (続き)

デプロイメント 421

WebSphere Application Server

Community Edition

開始 430

デプロイメント 426

routetable コマンド 480

## S

Secure Sockets Layer (SSL)

カタログ・サーバー 509

security

概要 591

許可 57

クライアント・セキュリティー 599

構成 599

シングル・サインオン (SSO) 577

セキュア・トランスポート 57

統合 591

統合、WebSphere Application Server と  
の 596

トランスポート・タイプ 586

認証 57, 577

プラグイン 601

ローカル 601

J2C クライアント接続 605

overview 577

troubleshooting 639

SIP

セッション管理 357

session 357

Spring

キャッシュ抽象化 402

キャッシュ・プロバイダー 402

SSL パラメーター 587

startOgServer 451, 454

options 457

stopOgServer 451, 465

## T

teardown コマンド 466, 480

trace

構成のオプション 620

troubleshooting 618

troubleshooting 609

trace 618

XML 構成 635

tuning

オペレーティング・システム 559

ガーベッジ・コレクション (garbage  
collection)

Real Time 570

tuning (続き)

ネットワーク設定 559

ネットワーク・ポート 53

Java 仮想マシン 564

## W

wasprofile コマンド 225

Web コンソール

開始 508

カスタム・レポート 518

カタログ・サーバー接続 509

統計 511

統計の説明 512

overview 507

WebSphere Application Server

構成、WebSphere eXtreme Scale との  
288

WebSphere eXtreme Scale

構成、WebSphere Application Server と  
の 288

WebSphere Portal

構成 358

Wily Introscope 550

wsadmin

MBean 498, 541

wsadmin ツール

カタログ・サービス・ドメイン 290

MBeans 498, 541

## X

xIOContainerTCPNonSecurePort プロパティ

ー 317

XML 構成

troubleshooting 635

xsadmin

xscmd へのマイグレーション 245

xscmd

セキュリティ・プロファイル 604

マイグレーション 245

xscmd ユーティリティー

管理 473

モニターに使用 527

xsloganalyzer 624, 625





Printed in Japan