IBM WebSphere eXtreme Scale バージョン 7.1.1

バージョン 7 リリース 1

管理ガイド

2011 年 11 月 21 日

本書は、WebSphere eXtreme Scale バージョン 7 リリース 1 モディフィケーション 1、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： WebSphere® eXtreme Scale Version7.1.1
Version 7 Release 1
Administration Guide
November 21, 2011

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 2011.12

目次

図

表

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 7.1.1 インフォメーション・センターからも入手することができます。

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/v7r1m1` 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。

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。

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ー・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

単純データ・グリッドを構成するには、入門用サンプル内に用意されている `objectgrid.xml` ファイルと `deployment.xml` ファイルを使用します。

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、コンテナー・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・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、開始時にコンテナー・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、一緒に使用される ObjectGrid XML と互換でなければなりません。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 エレメント内に定義された backingMap エレメントは、ObjectGrid XML 内にある backingMap と整合していなければなりません。すべての backingMap は、1 つの mapSet 内のみで参照する必要があります。

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ーが必要です。同期レプリカの数が 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ーを追加すると、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。複数のコンテナー・サーバー・インスタンスを開始する場合、それらのコンテナー・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の数を定義する方法

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。

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();
   ```

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。

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。コンテナー・サーバーの場所を追跡したり、データの配置を制御して、コンテナー・サーバーにホスティングしたりします。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ー・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、複数のコンテナー・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
   ```

   wxs_install_root の部分は、eXtreme Scale インストール・ルート・ディレクトリーへのパス、または抽出した eXtreme Scale 試用版 *wxs_install_root* のルート・ファイル・パスに置き換えてください。
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`

4. 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、コンテナー・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`

   コンテナー・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ー・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。

   コンテナー・サーバーは **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 c0 -catalogServiceEndPoints localhost:2809 -objectgridFile gettingstarted¥xml¥objectgrid.xml -deploymentPolicyFile gettingstarted¥xml¥deployment.xml`
   - Windows `startOgServer.bat c0 -catalogServiceEndPoints localhost:2809 -objectgridFile gettingstarted¥xml¥objectgrid.xml -deploymentPolicyFile gettingstarted¥xml¥deployment.xml`

5. 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`: このスクリプトは、単純な 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。

   a. データ・グリッドへのデータの追加:

      - 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 ユーティリティーおよび 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、69 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) カタログ・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. 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。
6. 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 ユーティリティーによるモニター

1. 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・コンテナー・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ー・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。

  コンテナー・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、ネットワーキング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。

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、コンテナー・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ー・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ー・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。詳しくは、を参照してください。

キャッシング・トポロジーに関する考慮事項

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。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ーの数**: 環境内には物理マシンとプロセッサー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、 59 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。

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・データはコンテナー・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ーが向上します。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。

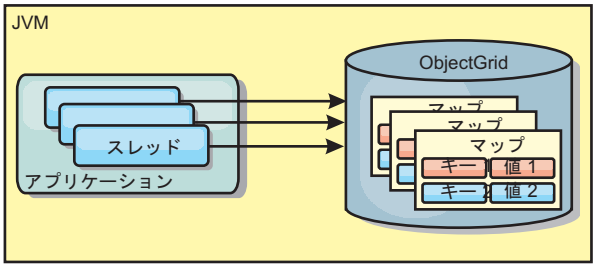


図 1. ローカルのメモリー内のキャッシュ・シナリオ

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。

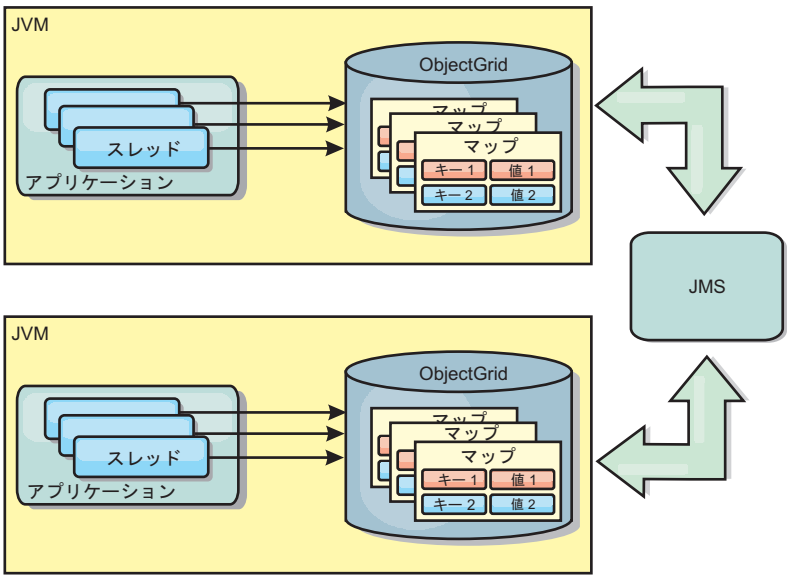


図 2. JMS によって変更が伝搬されるピア複製キャッシュ

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。

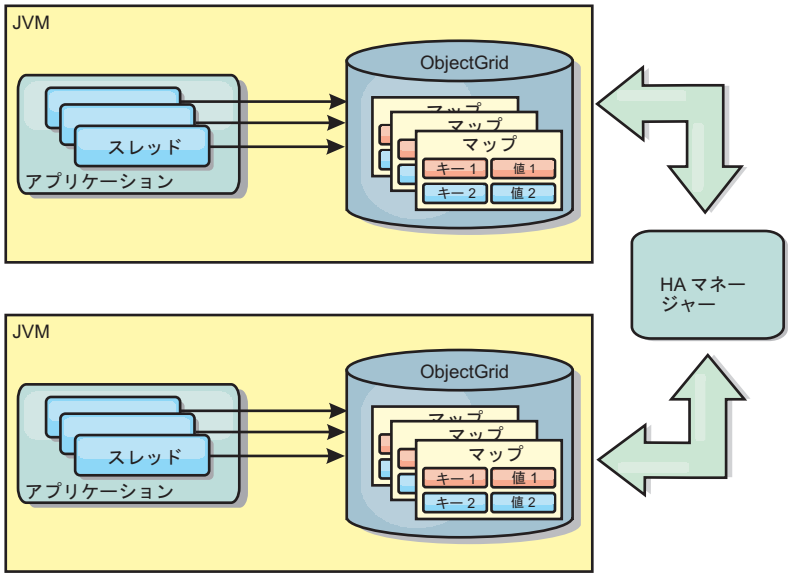


図 3. HA マネージャーによって変更が伝搬されるピア複製キャッシュ

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。

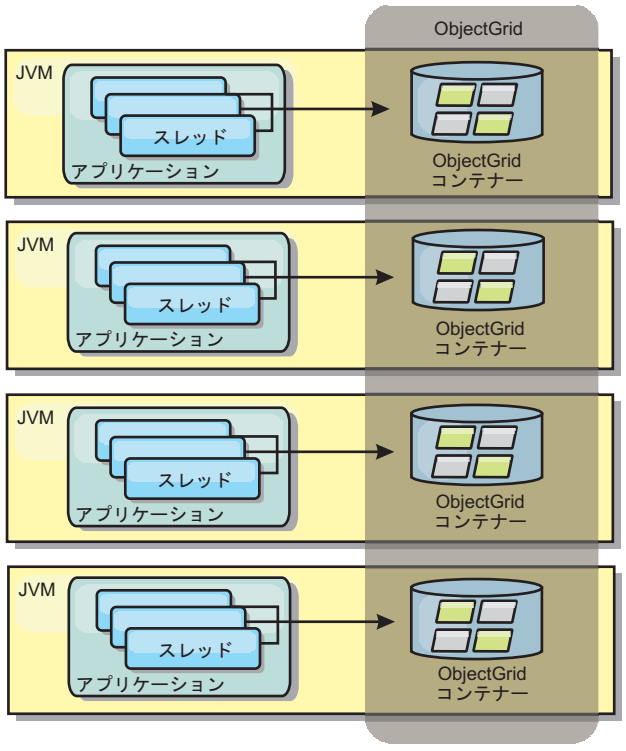


図 4. 組み込みキャッシュ

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の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・プロシージャー・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・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。

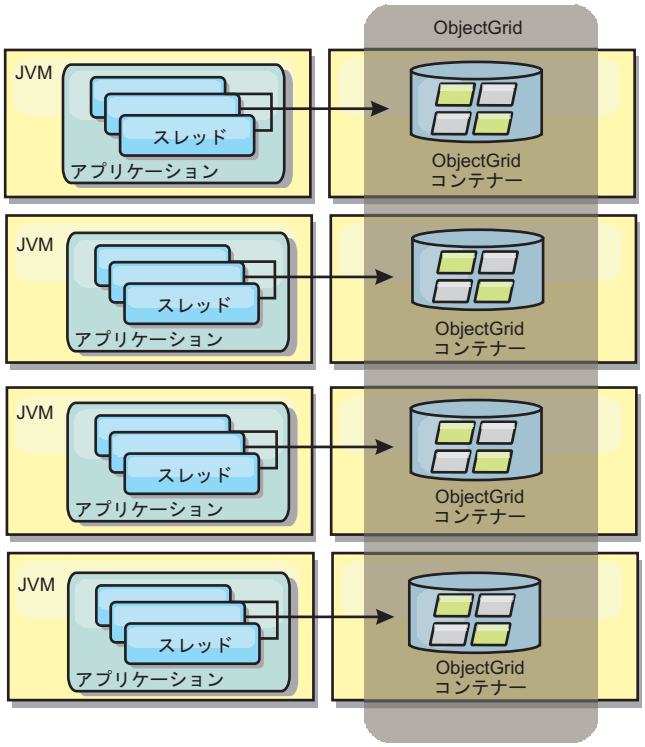


図 5. 分散キャッシュ

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。

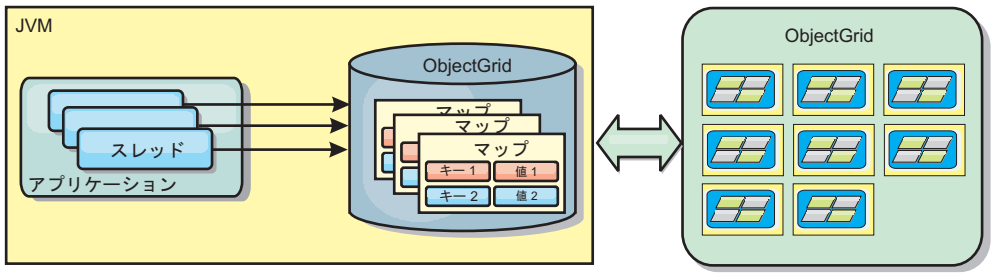


図 6. ニア・キャッシュ

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ーをなくすことです。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。

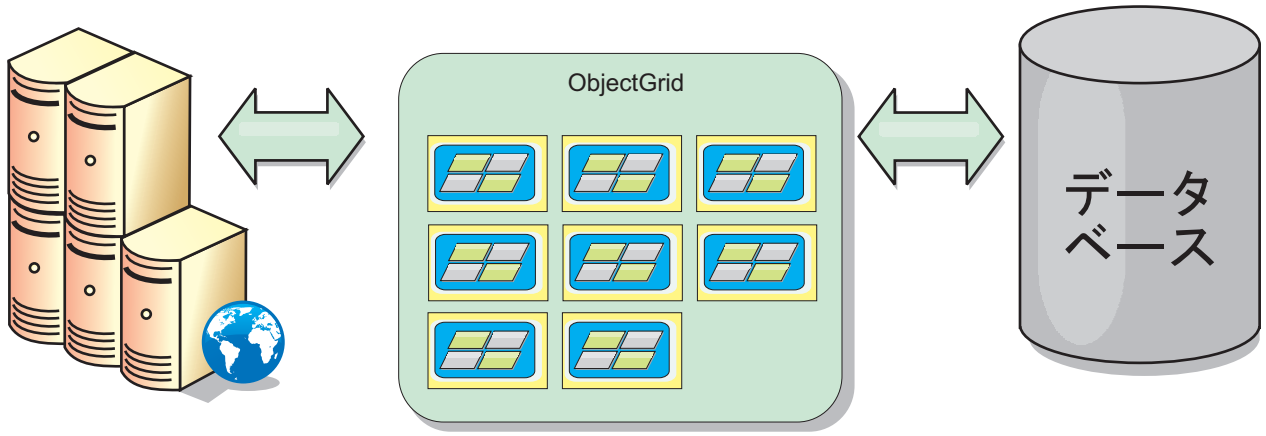


図 7. データベース・バッファーとしての *ObjectGrid*

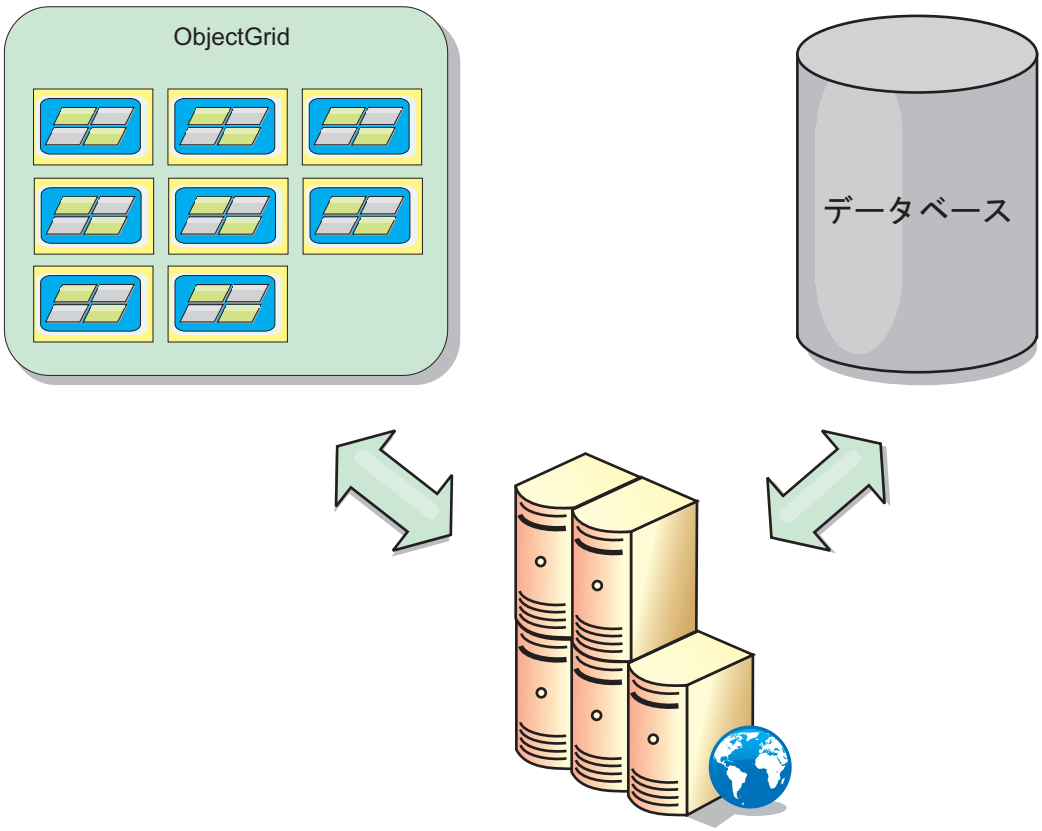


図 8. サイド・キャッシュとしての ObjectGrid

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する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

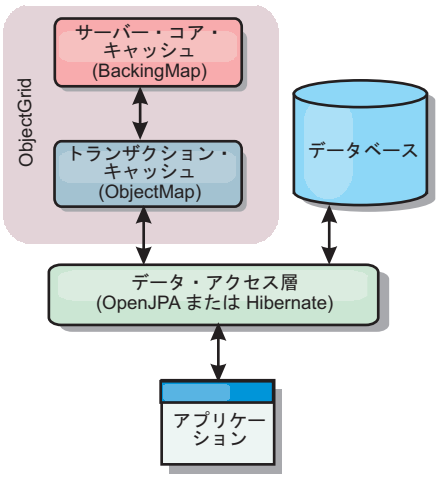


図 9. サイド・キャッシュ

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

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。

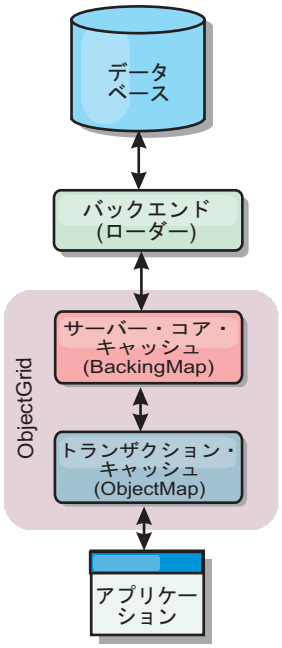


図 10. インライン・キャッシュ

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ース・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。

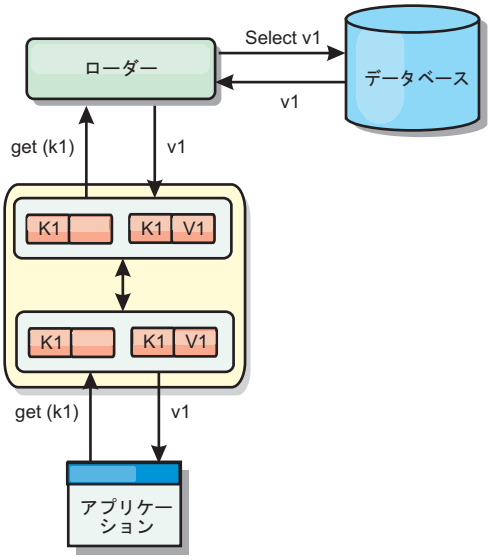


図 11. リードスルー・キャッシング

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。

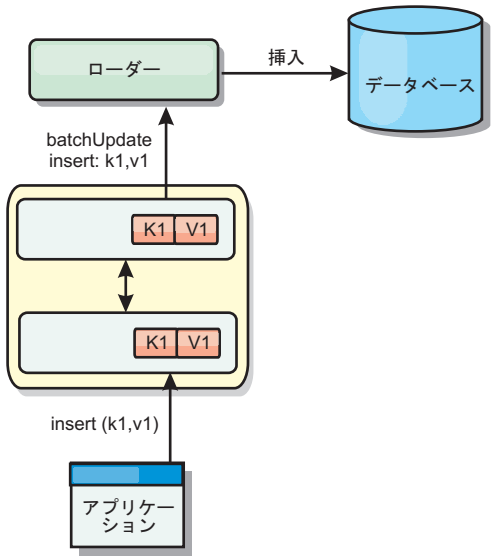


図 12. ライトスルー・キャッシング

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 内でバッファー化されてから、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。

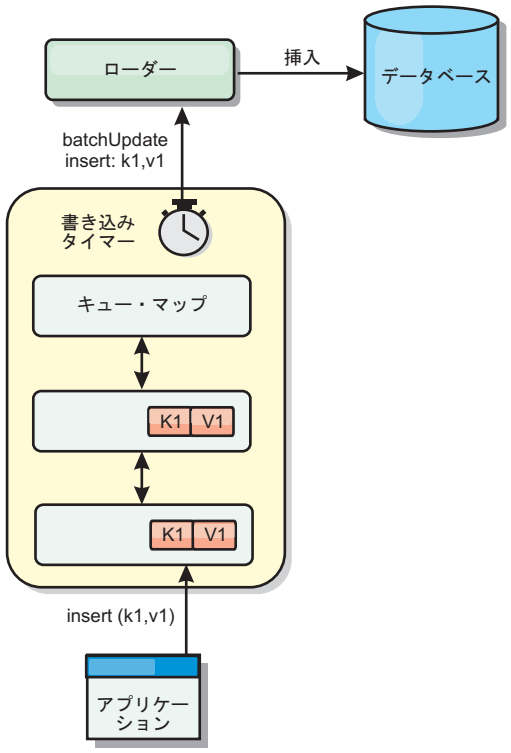


図 13. 後書きキャッシング

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。

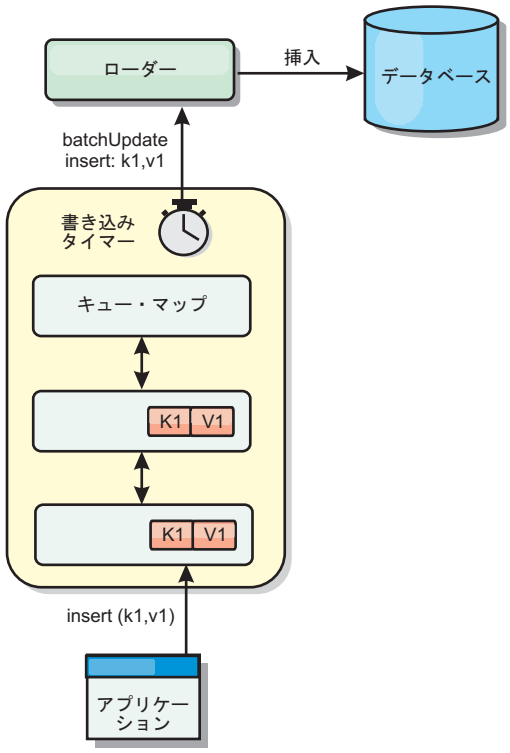


図 14. 後書きキャッシング

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 内でキューに入れられます。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 381 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られます。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。

アプリケーション設計に関する考慮事項

後書きサポートを使用可能にすることは簡単ですが、後書きサポートを扱うアプリケーションを設計する際には、注意すべき考慮事項があります。後書きサポートがない場合、ObjectGrid トランザクションにバックエンド・トランザクションが包含されます。 ObjectGrid トランザクションはバックエンド・トランザクションの開始前に開始し、バックエンド・トランザクションの終了後に終了します。

後書きサポートが有効な場合、ObjectGrid トランザクションは、バックエンド・トランザクションが開始する前に終了します。ObjectGrid トランザクションとバックエンド・トランザクションは切り離されます。

参照保全性の制約

後書きサポートで構成されているそれぞれのバックアップ・マップは、データをバックエンドにプッシュするための独自の後書きスレッドを持ちます。したがって、1 つの ObjectGrid トランザクションにさまざまなマップを更新するデータが含まれていても、バックエンドでは、それぞれ異なるバックエンド・トランザクションでデータの更新が行われます。例えば、トランザクション T1 はマップ Map1 のキー key1 とマップ Map2 のキー key2 を更新するとします。マップ Map1 に対する key1 更新は、1 つのバックエンド・トランザクションでバックエンドに対して更新され、マップ Map2 に対する key2 更新は、異なる後書きスレッドにより別のバックエンド・トランザクションでバックエンドに対して更新されます。Map1 に保管されたデータと Map2 に保管されたデータがバックエンドでの外部キー制約などの関係を持つ場合、更新が失敗する可能性があります。

バックエンド・データベースの参照保全性制約を設計するときは、順不同の更新に必ず対応できるようにしてください。

キュー・マップのロックの振る舞い

トランザクションの動作で他に大きく異なる点は、ロックの振る舞いです。ObjectGrid は、PESSIMISTIC、OPTIMISITIC、および NONE の 3 つの異なるロック・ストラテジーをサポートします。後書きキュー・マップは、 *バックアップ・マップに構成されているロック・ストラテジーに関係なく、ペシミスティック・ロック・ストラテジーを使用します。キュー・マップのロックを取得する操作には 2 つの異なるタイプがあります。

- ObjectGrid トランザクションのコミット時、またはフラッシュ (マップ・フラッシュまたはセッション・フラッシュ) の発生時、トランザクションはキュー・マップ内のキーを読み取り、キーに S ロックをかけます。

- ObjectGrid トランザクションのコミット時、トランザクションは、キーの S ロックを X ロックにアップグレードしようとします。

キュー・マップのこの余分な動作のため、ロックの動作に少々違いがあります。

- ユーザー・マップがペシミスティック・ロック・ストラテジーで構成されている場合、ロックの動作にほとんど違いはありません。フラッシュまたはコミットが呼び出されるたび、キュー・マップ内の同じキーに S ロックがかけられます。コミット時間中、ユーザー・マップ内のキーに X ロックが取得されるだけでなく、キュー・マップ内のキーに対しても X ロックが取得されます。
- ユーザー・マップが OPTIMISTIC または NONE ロック・ストラテジーで構成されている場合、ユーザー・トランザクションは PESSIMISTIC ロック・ストラテジーのパターンに従います。フラッシュまたはコミットが呼び出されるたびに、キュー・マップ内の同じキーに対して S ロックが取得されます。コミット時間の間、同じトランザクションを使用するキュー・マップ内のキーに対して X ロックが設定されます。

ローダー・トランザクションの再試行

ObjectGrid は、2 フェーズ・トランザクションまたは XA トランザクションをサポートしません。後書きスレッドは、キュー・マップからレコードを除去して、バックエンドに対してそのレコードを更新します。トランザクションの最中にサーバーに障害が起こると、一部のバックエンドの更新が失われる可能性があります。

後書きローダーは、失敗したトランザクションの書き込みを自動的に再試行し、データ損失を防ぐために未確定 LogSequence をバックエンドに送信します。このアクションを行うには、ローダーがべき等である必要があります。この意味は、Loader.batchUpdate(TxId, LogSequence) が同じ値で 2 回呼び出されたとき、それは適用された回数があたかも 1 回だったかのように、同じ結果を返すということです。ローダー実装は、この機能を使用可能にするため、RetryableLoader インターフェースを実装しなければなりません。詳しくは、API 資料を参照してください。

ローダーの障害

Loader 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。これは、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。

したがって、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、重複キー障害などのある種のデータ障害のため、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、更新のバッチ処理を解除し、データ障害を分離するため、1 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試行します。

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。

パフォーマンスの考慮事項

後書きキャッシング・サポートの場合、ローダー更新をトランザクションから除去することで、応答時間が増加します。また、データベース更新が結合されるため、データベース・スループットも増加します。データをキュー・マップからプルし、ローダーにプッシュされる後書きスレッドの導入によって生じるオーバーヘッドを理解しておく必要があります。

予想される使用パターンおよび環境に基づいて、最大更新数または最大更新時間を調整する必要があります。最大更新カウントまたは最大更新時間の値が小さすぎると、後書きスレッドのオーバーヘッドが、その利点を帳消しにするおそれがあります。これら 2 つのパラメーターに大きな値を設定する場合も、データのキューイングに必要なメモリー使用が増え、データベース・レコードが不整合になる時間が増加するおそれがあります。

最善のパフォーマンスを得るために、後書き関係のパラメーターは、以下の要因を考慮に入れて調整してください。

- 読み取りトランザクションと書き込みトランザクションの比率
- 同一レコード更新の頻度
- データベース更新の待ち時間

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。

概説

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。

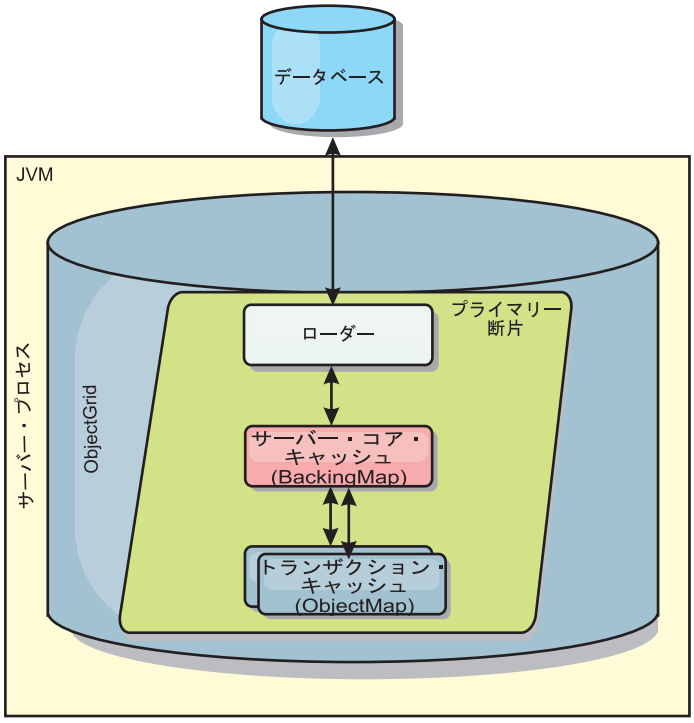


図 15. ローダー

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できます。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、45 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ず、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、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。

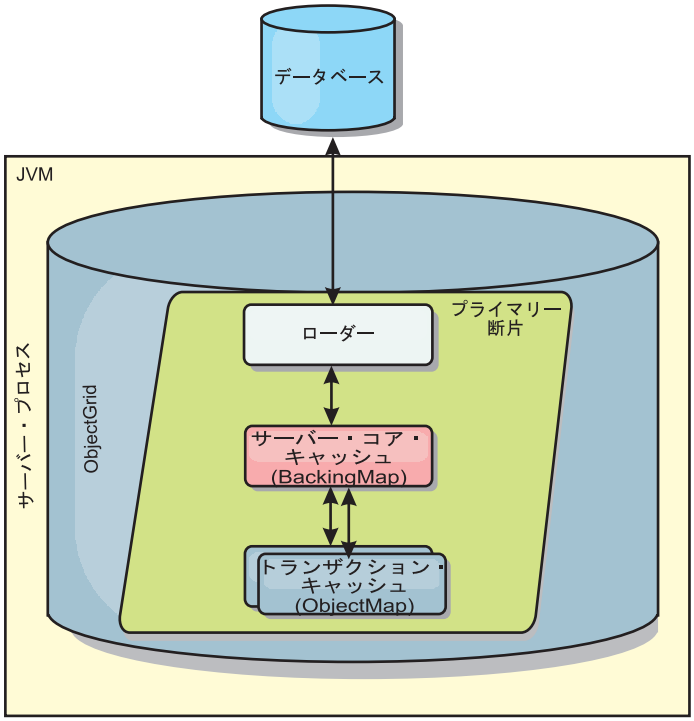


図 16. Loader プラグイン

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、356 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。

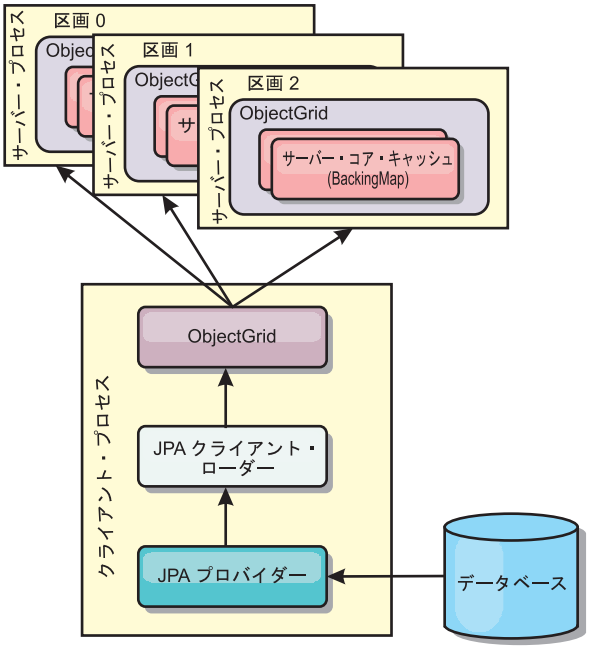


図 17. クライアント・ローダー

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。

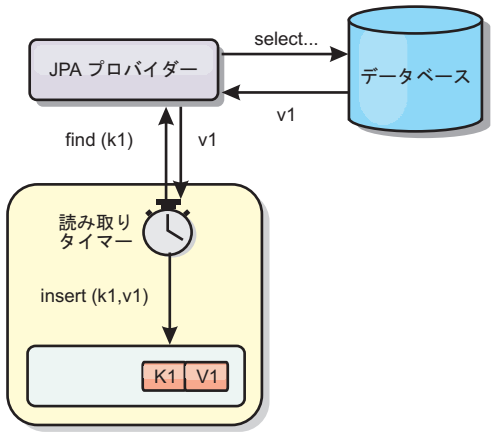


図 18. 定期的リフレッシュ

除去

スパース・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

スパース・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

Scale キャッシュ・データを削除するには、イベント・ベースの無効化メカニズムまたはプログラマチックな無効化メカニズムが使用できます。

イベント・ベースの無効化

スパース・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ー・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、索引プロキシー・オブジェクトが戻されます。索引プロキシー・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。索引プロキシー・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ー・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、索引プロキシー・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 の索引プロキシー・オブジェクトを取得し、その索引プロキシー・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・エント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えます。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

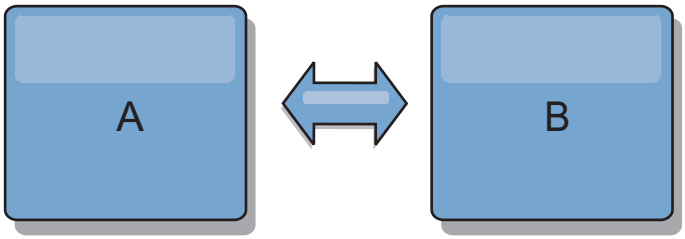
マルチマスター非同期レプリカ生成機能を使用すると、2 つ以上のデータ・グリッドを、互いの正確なミラーにすることができます。 各データ・グリッドは独立したカタログ・サービス・ドメイン内でホストされ、独自のカタログ・サービス、コンテナー・サーバー、および固有の名前を所有しています。 マルチマスター非同期レプリカ生成機能により、リンクを使用してカタログ・サービス・ドメインのコレクションを接続できます。すると、カタログ・サービス・ドメインは、リンクを介したレプリカ生成を使用して同期されます。カタログ・サービス・ドメイン間のリンクの定義を使用して、ほとんどどのトポロジーでも構成できます。

マルチマスター・レプリカ生成のためのトポロジー

マルチマスター・レプリカ生成を導入する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

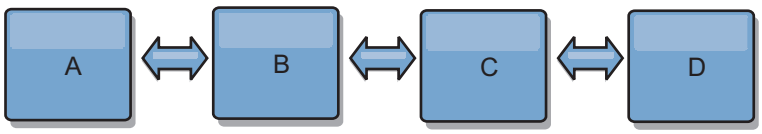
レプリカ生成データ・グリッドのインフラストラクチャーは、カタログ・サービス・ドメイン間を双方向のリンクで接続したカタログ・サービス・ドメインのグラフです。 リンクを使用して、2 つのカタログ・サービス・ドメインはデータ変更内容をやりとりできます。例えば、最も単純なトポロジーは、カタログ・サービス・ドメイン間に単一のリンクを持つ 1 対のカタログ・サービス・ドメインです。 カタログ・サービス・ドメインは、左から A、B、C というようにアルファベット順で指定されています。リンクは、遠距離にわたる広域ネットワーク (WAN) を経由する場合もあります。 リンクが遮断されたとしても、いずれかのカタログ・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、eXtreme Scale 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 つのドメインで行われたすべての変更を確実に他の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。 ドメイン B はドメイン A から変更をプルします。ドメイン C は、ドメイン A と C を接続するドメイン B を介してドメイン A から変更を受信します。同様に、ドメイン D はドメイン C を介して他のドメインから変更を受信します。この機能によって、変更のソースから変更を配布する負荷が分散されます。



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。

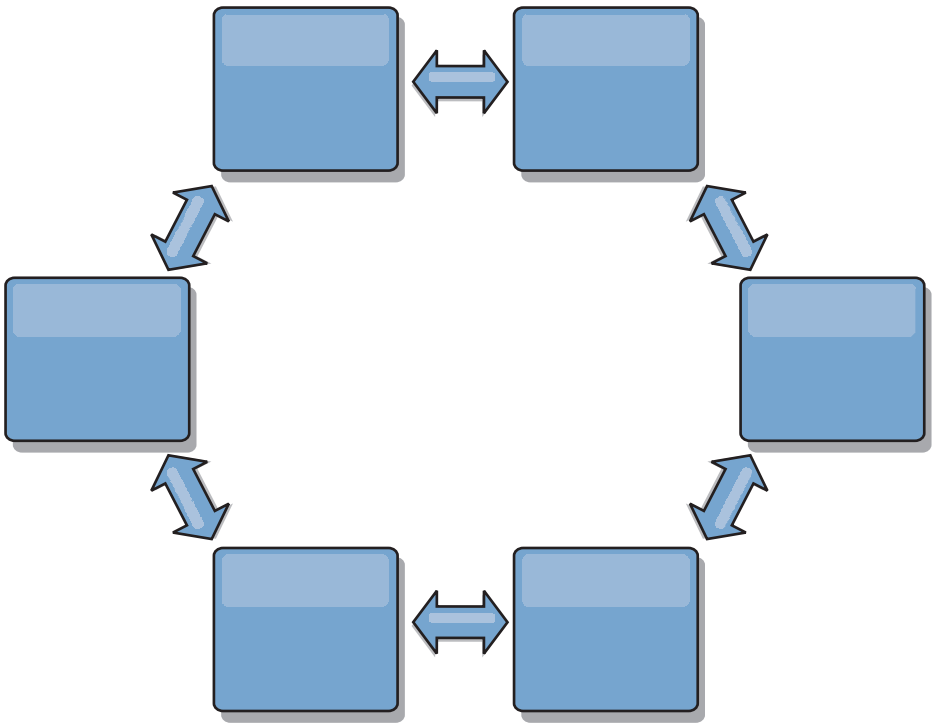
1. ドメイン D は、ドメイン C が再開されるまで孤立します。
2. ドメイン C は、ドメイン A のコピーであるドメイン B と自分自身を同期させます。

3. ドメイン D は、ドメイン C を使用して、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、ドメイン D が孤立していた間 (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、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

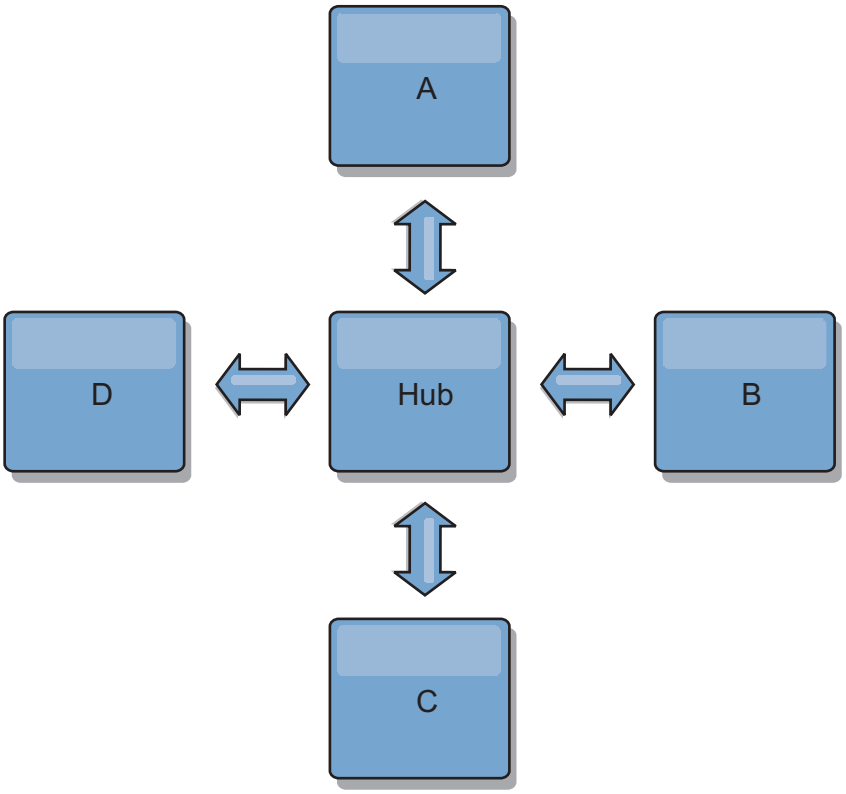
リング・トポロジーは、より回復力のあるトポロジーの例です。 カタログ・サービス・ドメインまたは単一リンクに障害が起こった場合でも、残ったカタログ・サービス・ドメインがまだ変更を取得できます。そのカタログ・サービス・ドメインは、障害から離れて、リングの周りを回ります。リング・トポロジーの大きさには関係なく、各カタログ・サービス・ドメインは他のカタログ・サービス・ドメインとのリンクを最大 2 つ持ちます。変更を伝搬するための待ち時間は長くなる場合があります。特定のカタログ・サービス・ドメインでの変更は、すべてのカタログ・サービス・ドメインにその変更が反映されるまで、複数のリンクを経由して伝搬する必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

ハブ・アンド・スポーク・トポロジーでは、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・トポロジーでは待ち時間が短縮されます。ハブ・ドメインは、リンク経由ですべてのスポーク・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 eXtreme Scale 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ー・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。

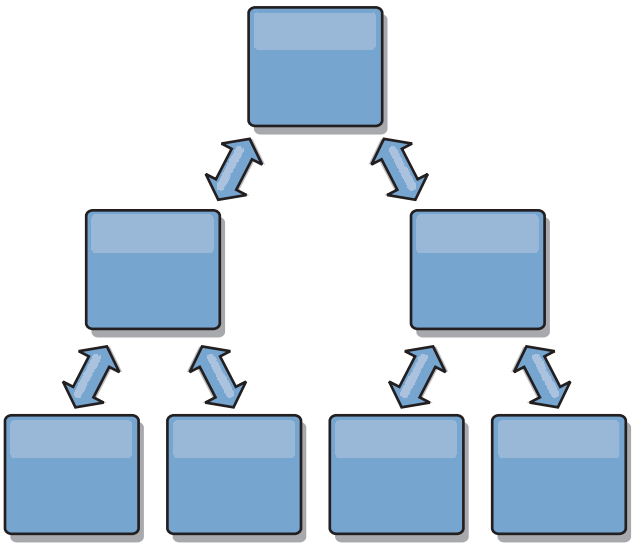
クライアントによって行われた変更は、クライアントに対してローカルで、非同期でハブに複製されます。 ハブはアービトレーション・ドメインとして機能し、すべての接続されたクライアントに変更を配布します。完全複製クライアントのトポロジーは、OpenJPA などのオブジェクト・リレーショナル・マッパーに信頼性の高い

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・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、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 eXtreme Scale 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ー・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・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・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が、単一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。

  – カタログ・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。

  – エンティティー・マネージャーを使用しない。エンティティー・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ー断片がデータ・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、この改訂情報は含められず、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。

スパース・キャッシュの問題

スパース・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、プライマリー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ー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ー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ー、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 つのカタログ・サービス・ドメインのライン	あり	1 つのカタログ・サービス・ドメインをアービターとして選択します。
3 つのカタログ・サービス・ドメインのライン	あり	真ん中のカタログ・サービス・ドメインがアービターでなければなりません。 真ん中のカタログ・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 つより多いカタログ・サービス・ドメインのライン	なし	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	あり	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・カタログ・サービス・ドメインでなければなりません。
N 個のカタログ・サービス・ドメインのリング	なし	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)	あり	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、変更が特定のカタログ・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タログ・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、すべてのカタログ・サービス・ドメインを他のすべてのカタログ・サービス・ドメインにリンクすることによって中間カタログ・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

タログ・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。

変更が他のカタログ・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。

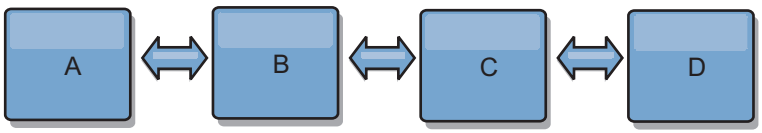
- ソース・カタログ・サービス・ドメイン上のプロセッサーとネットワーク帯域幅
- ソース・カタログ・サービス・ドメインとターゲット・カタログ・サービス・ドメインの間の中間カタログ・サービス・ドメイン数とリンク数
- ソース・カタログ・サービス・ドメイン、ターゲット・カタログ・サービス・ドメイン、および中間カタログ・サービス・ドメインで使用可能なプロセッサーとネットワーク・リソース

- **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。

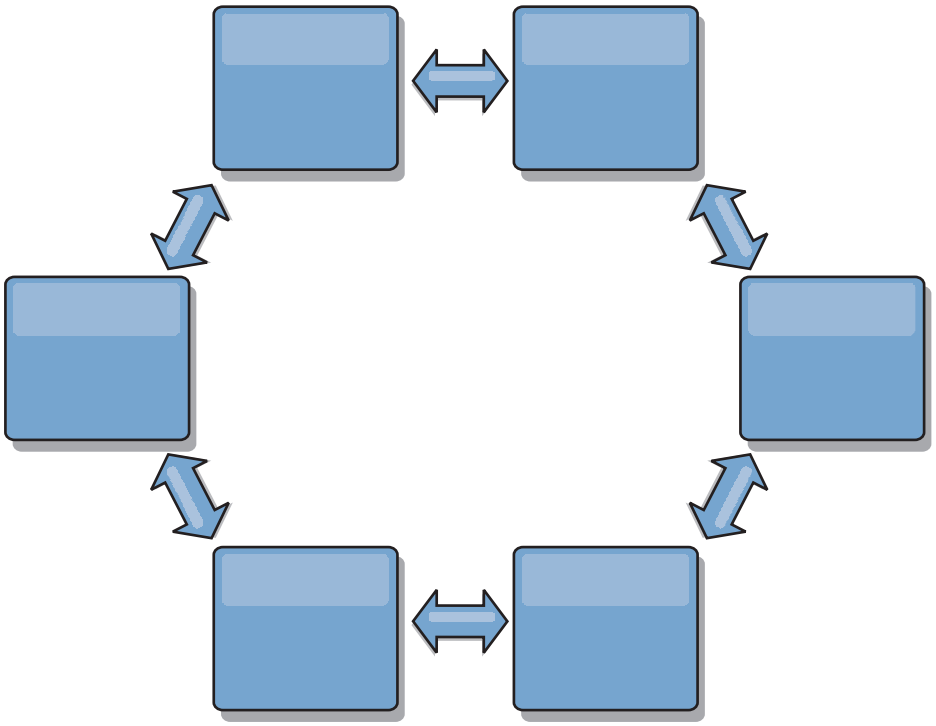
4 つのカタログ・サービス・ドメイン A、B、C、および D を持つライン・トポロジーを考えてみます。



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。

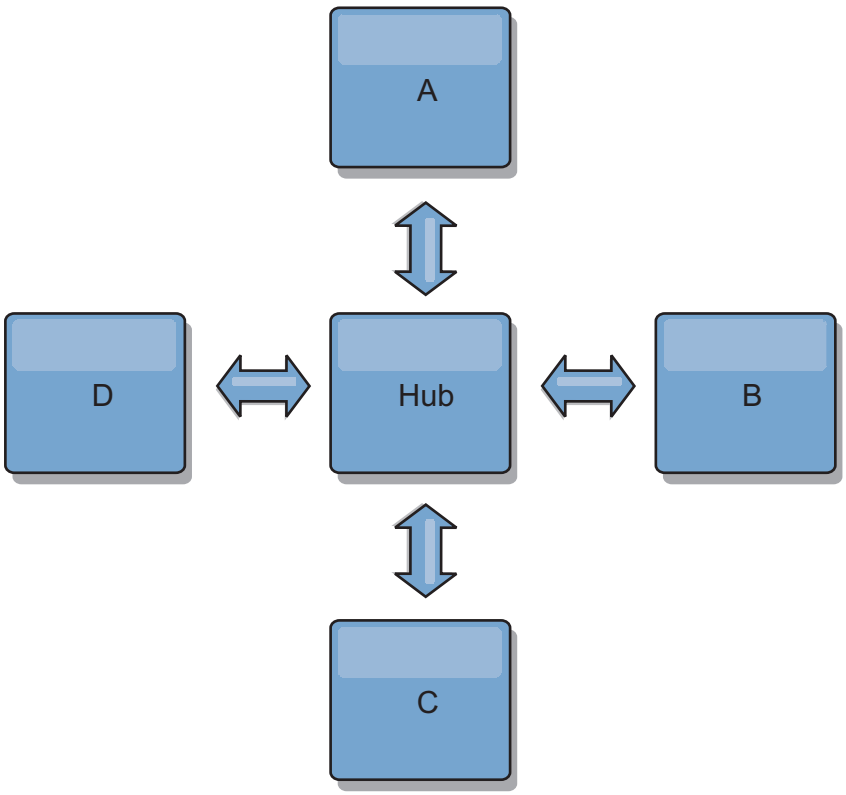
- ドメイン A が稼働中で B がダウン
- ドメイン A および B が稼働中で C がダウン
- A と B の間のリンクがダウン
- B と C の間のリンクがダウン
- C と D の間のリンクがダウン

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・プロパティーは、スケーラビリティーを提供するため、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、任意の一対の 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* 個のコンテナー 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、コンテナー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ーがあると、より多くのプロセッサーとネットワーク・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・バッファー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。各コンテナー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 WebSphere 製品とのインターオペラビリティー

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、コンテナー・サーバーおよびカタログ・サーバーをホストできます。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 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。

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。

ハードウェアおよびソフトウェアの要件

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

- **Web コンソールを使用しない場合**

  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。

- **Web コンソールを使用する場合**

  コンソールを使用する場合、それぞれのオペレーティング・システムについて以下の要件が適用されます。

  – Linux: 32 ビットまたは 64 ビット JVM
  – Linux PPC: 32 ビット JVM のみ
  – Windows: 32 ビット JVM のみ
  – AIX®: 32 ビット JVM のみ

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。

- 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

WebSphere Application Server 要件

- WebSphere Application Server バージョン 6.1.0.39 以降
- WebSphere Application Server バージョン 7.0.0.19 以降
- WebSphere Application Server バージョン 8.0.0.1 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 では、Java SE 5 以降が必要です。一般に、Java SE は、バージョンが新しい方が機能および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 以降と一緒に使用できます。使用しているバージョンは、現在、Java ランタイム環境 (JRE) ベンダーによってサポートされていなければなりません。

完全にサポートされている JRE は、*wxs_install_root*/java ディレクトリーにスタンドアロン 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インストールされており、クライアントとサーバーの両方で使用できるようになっています。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。

WebSphere eXtreme Scale は Java Development Kit (JDK) 5 を使用し、それ以降が使用可能になると、そちらの機能を使用します。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 または Java SE 6 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 以降でサポートされている	Java SE 6 以降でサポートされている
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: • `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。 • `ogagent.jar`: フィールド・アクセス・エンティティーのパフォーマンスを向上させます。	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:

  例: `/opt/IBM/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:

  例: `/opt/IBM/eXtremeScale/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`
- 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。

例: /opt/IBM/WebSphere/eXtremeScale/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。

例: /wxs-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: このトピックを適用するのは、COPY_TO_BYTES コピー・モードを使用していない場合です。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。

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ー・スペースを提供します。さらに、各 JVM は、データ・エント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。最も的確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。

```
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レプリカ断片) 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す。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。

プロセッサーのコストには、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、プロセッサー使用が飽和状態になるようにしてください。プロセッサー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。

プロセッサー使用量 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) 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ー・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なくなり、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の 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、プロセッサー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% のプロセッサー負荷でサーバー、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ー使用量を 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。動的キャッシュは、ビジネス・データや生成された HTML をキャッシュに入れるために、または、データ・レプリカ生成サービス (DRS) を使用してセル内のキャッシュ・データを同期化するために利用できます。

概説

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 = コンテナー・プロセス当たりの空きメモリー
- C = コンテナーの数
- 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。

256mb * 4 コンテナー/ (1 プライマリー + 1 レプリカ) = 512mb

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 コンテナー・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、レプリカの数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ー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の数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 = 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 = コンテナーの総数
- R = レプリカの総数

式 C

Ceiling(T/ (1+N)) = m

各部の意味は、次のとおりです。

- T = コンテナーの総数
- 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ー数

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、544 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、コンテナー・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・コンテナーは 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ーが 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・コンテナーの、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。コンテナー (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ー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。

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。

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 3. 運用チェックリスト

チェックリスト項目	詳細情報

AIX 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。
 	
TCP_KEEPINTVL
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。このプロパティーは、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。
`# no –o tcp_keepintvl`
現行設定を変更するには、次のコマンドを実行します。
`# no –o tcp_keepintvl=10`
TCP_KEEPINTVL 設定は、0.5 秒単位で設定します。
 	
TCP_KEEPINIT
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。このプロパティーは、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。
`# no –o tcp_keepinit`
現行設定を変更するには、次のコマンドを実行します。
`# no –o tcp_keepinit=40`
TCP_KEEPINIT 設定は、0.5 秒単位で設定します。
	• 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。

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す。 orb.properties ファイルは、java/jre/lib ディレクトリーにあります。	530 ページの『ORB プロパティー』

表 3. 運用チェックリスト (続き)

チェックリスト項目	詳細情報
startOgServer 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。 • **-jvmArgs** パラメーターを使用してヒープ設定を設定します。 • **-jvmArgs** 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ーを設定します。 • エージェント・モニターの構成用に **-jvmArgs** パラメーターを設定します。 **ポートの設定** 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、コンテナー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。 **リスナー・ポート** プロセス間の通信に使用するポートを指定する場合、**-listenerPort** 引数を使用できます。 **コア・グループ・ポート** 障害検出に使用するポートを指定する場合、**-haManagerPort** 引数を使用できます。この引数は、peerPort 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。 **JMX サービス・ポート** JMX サービスが使用するポートを指定する場合、**-JMXServicePort** 引数を使用できます。 **SSL ポート** `-Dcom.ibm.CSI.SSLPort=1234` を **-jvmArgs** 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。 **クライアント・ポート** カタログ・サービスのみで使用されます。この値は、**-catalogServiceEndPoints** 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。 `serverName:hostName:clientPort:peerPort`	433 ページの『**startOgServer** スクリプト』
次のセキュリティー設定が正しく構成されていることを検証します。 • トランスポート (SSL) • アプリケーション (認証と許可) セキュリティー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。	561 ページの『外部プロバイダーとのセキュリティー統合』

表 3. 運用チェックリスト (続き)

チェックリスト項目	詳細情報
ご使用の環境をモニターする方法を選択します。 • **xscmd** ツール: – カタログ・サーバーの JMX ポートが、**xscmd** ツールから可視である必要があります。コンテナー・サーバー・ポートも、コンテナーからの情報を収集する一部のコマンドにとってアクセス可能である必要があります。 • モニター・コンソール: 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。 • ベンダーのモニター・ツール: – Tivoli® Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• 498 ページの『**xscmd** ユーティリティーによるモニター』 • 558 ページの『Java Management Extensions (JMX) セキュリティー』 • 479 ページの『Web コンソールによるモニター』 • 514 ページの『IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale のモニター』 • 524 ページの『Hyperic HQ による eXtreme Scale のモニター』 • 521 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』

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ー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、コンテナー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、コンテナー・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。

コンテナー・サーバー

WebSphere eXtreme Scale コンテナー・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 コンテナー・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。コンテナー・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
peerPort と同義。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は、カタログ・サービスは使用可能なポートを自動的に生成します。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。 (WebSphere Application Server 環境のみで必須。)

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、コンテナー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、コンテナー・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。

7.1.1+ **xioChannel.xioContainerTCPSecure.Port**
サーバー上の eXtremeIO の SSL ポート番号を指定します。**transportType** プロパティーが `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ーが使用されます。

7.1.1+ **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** プロパティーが `TCP/IP` に設定されている場合のみこのプロパティーが使用されます。

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、コンテナー・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・プロパティー・ファイル内の **listenerPort** プロパティーを使用します。

haManagerPort
peerPort と同義。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は、カタログ・サービスは使用可能なポートを自動的に生成します。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。 (WebSphere Application Server 環境のみで必須。)

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場合、`-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>` 引数を使用して、Secure Socket Layer (SSL) ポートを構成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、コンテナー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、コンテナー・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。

277 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***」。

セキュリティーの概要

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およびストリーム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 へ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・セキュリティー

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ーを提供します。

グリッド・セキュリティー

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 およびコンテナー・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、使用するので、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詳しくは、 561 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。

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、エンティティー・マネージャー、照会、およびセキュリティー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、コンテナー・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

手順

1. 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 1』
   - 非セキュア・カタログ・サーバーの始動
   - 非セキュア・コンテナー・サーバーの始動
   - データにアクセスするクライアントの始動
   - **xscmd** ユーティリティーを使用した、マップ・サイズの表示
   - サーバーの停止
2. 79 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』
   - CredentialGenerator の使用
   - Authenticator の使用
   - セキュア・カタログ・サーバーの始動
   - セキュア・コンテナー・サーバーの始動
   - 保護 ObjectGrid にアクセスするクライアントの始動
   - **xscmd** ユーティリティーを使用した、マップ・サイズの表示
   - セキュア・サーバーの停止
3. 86 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』
   - JAAS 許可ポリシーの使用
4. 90 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』

- 鍵ストアおよびトラストストアの作成
- サーバーの SSL プロパティーの構成
- クライアントの SSL プロパティーの構成
- **xscmd** ユーティリティーを使用した、マップ・サイズの表示
- セキュア・サーバーの停止

Java SE セキュリティー・チュートリアル - ステップ 1

このトピックではシンプルで非セキュアなサンプル について説明します。また、利用可能な統合セキュリティーを強化するため、このチュートリアルのステップごとにセキュリティー機能を順次追加していきます。

始める前に

注: チュートリアルのこのステップで必要なファイルはすべて、次のセクションに示します。

手順

サンプルの実行

次のスクリプトを使用してカタログ・サービスを始動します。カタログ・サービスの開始について詳しくは、427 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。

1. bin ディレクトリーに移動します。`cd objectgridRoot/bin`
2. catalogServer という名前のカタログ・サーバーを始動します。
   - UNIX Linux `startOgServer.sh catalogServer`
   - Windows `startOgServer.bat catalogServer`
3. bin ディレクトリー `cd objectgridRoot/bin` に移動します。
4. 次のスクリプトを使用して c0 という名前のコンテナー・サーバーを起動します。
   - 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
     ```

例

コンテナー・サーバーの開始方法について詳しくは、430 ページの『コンテナー・サーバーの始動』を参照してください。

カタログ・サーバーとコンテナー・サーバーが始動されたならば、次のようにしてクライアントを起動します。

1. 再度、bin ディレクトリーに移動します。
2. `java -classpath ../lib/objectgrid.jar;../applib/secsample.jar com.ibm.websphere.objectgrid.security.sample.guide.SimpleApp`

secsample.jar ファイルには、SimpleApp クラスが含まれています。

このプログラムの出力は次のとおりです。

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

xscmd ユーティリティーを使用して、「accounting」グリッドのマップ・サイズを表示することもできます。

- ディレクトリー objectgridRoot/bin に移動します。
- **xscmd** ユーティリティーを使用して、マップ・サイズを表示します。
  - UNIX Linux `xscmd.sh -c showMapSizes -g accounting -ms mapSet1`
  - Windows `xscmd.bat -c showMapSizes -g accounting -ms mapSet1`

サーバーの停止

コンテナー・サーバー

以下のコマンドを使用してコンテナー・サーバー c0 を停止します。

UNIX Linux
```
stopOgServer.sh c0 -catalogServiceEndPoints
localhost:2809
```

Windows
```
stopOgServer.bat c0 -catalogServiceEndPoints localhost:2809
```

以下のメッセージが出力されます。

```
CWOBJ2512I: ObjectGrid server c0 stopped.
```

カタログ・サーバー

以下のコマンドを使用して、カタログ・サーバーを停止できます。

UNIX Linux
```
stopOgServer.sh catalogServer -catalogServiceEndPoints
localhost:2809
```

Windows
```
stopOgServer.bat catalogServer -catalogServiceEndPoints
localhost:2809
```

カタログ・サーバーをシャットダウンすると、次のメッセージが表示されます。

```
CWOBJ2512I: ObjectGrid server catalogServer stopped.
```

必要なファイル

下記のファイルは、SimpleApp の Java クラスです。

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");

        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;

    }

}
```

このクラスの getObjectGrid メソッドは、ObjectGrid を取得し、run メソッドは、カスタマー・マップからレコードを読み取り、値を更新します。

分散環境でこのサンプルを実行する場合、ObjectGrid 記述子 XML ファイル SimpleApp.xml およびデプロイメント XML ファイル SimpleDP.xml を作成します。以下の例で、これらのファイルを取り上げています。

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>
```

これは、「accounting」という 1 つの ObjectGrid インスタンスと「customer」という 1 つのマップ (mapSet「mapSet1」内にある) を含むシンプルな ObjectGrid 構成です。 SimpleDP.xml ファイルの特徴は、1 つの区画と 0 個の最小必要レプリカで構成される 1 つのマップ・セットです。

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 2

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 1』 を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、実例を示すサンプルのコードおよびスクリプトを含め、eXtreme Scale サーバー環境におけるクライアント認証の方法を明らかにします。

他のすべての認証メカニズムと同様に、最小の認証は以下のステップで構成されています。

1. 管理者は、認証を必須とするよう構成を変更します。
2. クライアントは、サーバーに資格情報を提供します。
3. サーバーは、その資格情報をレジストリーに対して認証します。

手順

1. **クライアント資格情報**

   クライアントの資格情報は、com.ibm.websphere.objectgrid.security.plugins.Credential インターフェースによって表されます。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。

このインターフェースでは、equals(Object) メソッドおよび hashCode() メソッドを明示的に定義します。Credential オブジェクトをサーバー・サイドの鍵として使用することによって認証済み Subject オブジェクトがキャッシュされるため、この 2 つのメソッドは重要です。

さらに、eXtreme Scale は資格情報を生成するプラグインを提供します。このプラグインは、com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator インターフェースによって示され、クライアント資格情報の生成に使用されます。これは、資格情報に期限がある場合に役立ちます。この場合は、getCredential() メソッドが呼び出されて資格情報が更新されます。詳しくは、「CredentialGenerator API 資料」を参照してください。

これら 2 つのインターフェースを eXtreme Scale クライアント・ランタイム対して実装することで、クライアント資格情報を取得することができます。

このサンプルは、eXtreme Scale が提供する以下の 2 つのサンプル・プラグインの実装を使用します。

```
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。

2. **サーバー・オーセンティケーター** 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、このクライアント 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。 eXtreme Scale サーバーは、要求を処理する前に 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、このクライアントを表す Subject オブジェクトが戻されます。

   そうすると、この Subject オブジェクトはキャッシュされますが、存続時間がセッション・タイムアウト値に達すると有効期限が切れます。ログイン・セッション・タイムアウト値は、クラスター XML ファイル内にある loginSessionExpirationTime プロパティーを使用して設定できます。例えば、`loginSessionExpirationTime="300"` と設定すると、Subject オブジェクトの有効期限は 300 秒で切れます。この Subject オブジェクトは、後で示すように、要求の認可に使用されます。

   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。詳しくは、「Authenticator API 資料」を参照してください。

   この例では、テストとサンプルを目的とする eXtreme Scale 組み込み実装である KeyStoreLoginAuthenticator を使用しています (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。

この KeyStoreLoginAuthenticator では KeyStoreLoginModule を使用し、JAAS ログイン・モジュール KeyStoreLogin を使用して鍵ストアでユーザーを認証します。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。以下の例では、JAAS 構成ファイル og_jaas.config に構成された keyStoreLogin 別名について示しています。

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

以下のコマンドでは、%OBJECTGRID_HOME%/security ディレクトリーに鍵ストア sampleKS.jks を作成し、パスワードとして sampleKS1 を使用します。また、アドミニストレーター・ユーザー、マネージャー・ユーザー、およびキャッシャー・ユーザーを表す 3 つのユーザー証明書が作成され、それぞれ独自のパスワードを使用します。

a. 次の eXtreme Scale ルート・ディレクトリーに移動します。

```
cd objectgridRoot
```

b. 「security」というディレクトリーを作成します。

```
mkdir security
```

c. 新規に作成した security ディレクトリーに移動します。

```
cd security
```

d. 鍵ツール (javaHOME/bin ディレクトリー内にある) を使用して、鍵ストア sampleKS.jks にユーザー「administator」をパスワード「administrator1」で作成します。

```
keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
-alias administrator -keypass administrator1
-dname CN=administrator,O=acme,OU=OGSample -validity 10000
```

e. 鍵ツール (javaHOME/bin ディレクトリー内にある) を使用して、鍵ストア sampleKS.jks にユーザー「manager」をパスワード「manager1」で作成します。

```
keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
-alias manager -keypass manager1
-dname CN=manager,O=acme,OU=OGSample -validity 10000
```

f. 鍵ツール (javaHOME/bin ディレクトリー内) を使用して、鍵ストア sampleKS.jks にユーザー「cashier」をパスワード「cashier1」で作成します。

```
keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
-alias cashier -keypass cashier1 -dname CN=cashier,O=acme,OU=OGSample
-validity 10000
```

クライアント・セキュリティー構成は、クライアント・プロパティー・ファイルに構成されます。以下のコマンドを使用し、%OBJECTGRID_HOME%/security ディレクトリーにコピーを作成します。

a. security ディレクトリーに移動します。

```
cd objectgridRoot/security
```

b. sampleClient.properties ファイルを client.properties ファイルにコピーします。

```
cp ../properties/sampleClient.properties client.properties
```

以下のプロパティーは、security ディレクトリーにある client.properties ファイルで強調表示されます。

a. **securityEnabled:** securityEnabled を true (デフォルト値) に設定すると、認証を含むクライアント・セキュリティーが使用可能になります。

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。

d. **singleSignOnEnabled:** false (デフォルト値) に設定します。シングル・サインオンは使用不可になります。

3. **サーバー・セキュリティー構成**

サーバー・セキュリティー構成は、セキュリティー記述子 XML ファイルおよびサーバー・セキュリティー・プロパティー・ファイルで指定されます。セキュリティー記述子 XML 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ー・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。

このサンプルで使用する security.xml ファイルを以下に示します。

```
<?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>
```

a. **securityEnabled:** true に設定し、認証を含むサーバー・セキュリティーを有効にします。

b. **loginSessionExpirationTime:** 値を 300 (デフォルト値) に設定します。

c. **authenticator:** 以下のように、オーセンティケーター・クラス KeyStoreLoginAuthenticator をクラスター XML ファイルに追加します。

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

d. **credentialAuthentication:** credentialAuthentication 属性を Required に設定し、サーバーが認証を必要とするようにします。

security.xml ファイルに関する詳しい説明は、セキュリティー記述子 XML ファイルを参照してください。

サーバーのプロパティー・ファイルを security ディレクトリーにコピーします。この時点で、このファイルを変更する必要はありません。

a. security ディレクトリーに移動します。

```
cd objectgridRoot/security
```

b. サンプル objectGrid の sampleServer.properties ファイルを properties ディレクトリーから新規の server.properties ファイルにコピーします。

```
cp ../properties/containerServer.properties server.properties
```

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。

4. **セキュア・クライアント** 以下の例に示すように、クライアント・アプリケーションを確実にサーバーに接続します。

```
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;

    }

}
```

非セキュアのアプリケーションとは、以下の 3 つの点で異なります。

a. 構成済みの client.properties ファイルを受け渡すことにより、ClientSecurityConfiguration オブジェクトを作成しています。

b. 渡されたユーザー ID とパスワードを使用することにより、UserPasswordCredentialGenerator を作成しています。

c. カタログ・サーバーに接続し、ClientSecurityConfiguration オブジェクトを受け渡すことにより ClientClusterContext から ObjectGrid を取得しています。

5. **アプリケーションの実行**

アプリケーションを実行するには、カタログ・サーバーを開始します。以下のように -clusterFile および -serverProps コマンド行オプションを発行して、セキュリティー・プロパティーを受け渡します。

a. bin ディレクトリーに移動します。

```
cd objectgridRoot/bin
```

b. カタログ・サーバーを起動します。

- 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"
```

次に、以下のスクリプトを使用し、セキュア・コンテナー・サーバーを起動します。

a. 再度、bin ディレクトリーに移動します。

```
cd objectgridRoot/bin
```

b. セキュア・コンテナー・サーバーを起動します。

- 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"
```

-serverProps を発行するとサーバー・プロパティー・ファイルが渡されます。

サーバーの始動後に、以下のコマンドを使用してクライアントを始動します。

a. `cd objectgridRoot/bin`

b.

```
java -classpath ../lib/objectgrid.jar;../applib/secsample.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。

secsample.jar ファイルには、SimpleApp クラスが含まれています。

SecureSimpleApp は、以下のリストに示されている 3 つのパラメーターを使用します。

a. ../security/client.properties ファイルは、クライアント・セキュリティー・プロパティー・ファイルです。

b. manager はユーザー ID です。

c. manager1 はパスワードです。

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.
```

xscmd ユーティリティーを使用して、「accounting」グリッドのマップ・サイズを表示することもできます。

- ディレクトリー objectgridRoot/bin に移動します。
- 次のように、オプションの -c showMapSizes コマンドを付けて **xscmd** コマンドを使用します。
  - UNIX Linux
    ```
    xscmd.sh -c showMapSizes -g accounting -m mapSet1 -username manager -password manager1
    ```
  - Windows
    ```
    xscmd.bat -c showMapSizes -g accounting -m mapSet1 -username manager -password manager1
    ```

これで、**stopOgServer** コマンドを使用して、コンテナー・サーバーまたはカタログ・サービス・プロセスを停止できます。ただし、セキュリティー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・プロパティー・ファイルは、以下の 2 つのプロパティーを定義して、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ー 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、チュートリアルの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。

```
CWOBJ2512I: 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・プロパティー・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、 547 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 3

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ー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 79 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ーがあります。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ーは、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` ディレクトリーに移動します。

      ```
      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。以下の ファイルは、セキュリティー・ディレクトリーに定義されます。

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    principal javax.security.auth.x500.X500Principal "CN=cashier,O=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,O=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。
- 最初の 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,O=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,O=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 プロパティーを使用して 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/ogAuth.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、 427 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の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・コンテナー・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"
```

前のコンテナー・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ー・サーバー・プロセスに設定します。

このチュートリアルの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/secsample.jar
com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
../security/client.properties manager manager1
```

ユーザー「manager」にはアカウンティング 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/secsample.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:O=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)
        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、549 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

チュートリアルの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 4』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 4

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ー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 86 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、コンテナー・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・セキュリティー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・セキュリティー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、コンテナー・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=Your Organizational Unit, O=Your
     Organization, 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。

   - *公開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。

   - *クライアントの公開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 プロパティー・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・セキュリティーを使用可能にするように ObjectGrid プロパティー・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 ディレクトリーにコピーします。

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

```
transportType=SSL-Required

alias=ogsample
contextProvider=IBMJSSE2
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、およびコンテナー・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ーは SSL 構成を設定するために使用されます。詳しくは、555 ページの『トランスポート層セキュリティーおよび 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、別のプロパティー 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 86 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、 79 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 コンテナー・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"
```

前のコンテナー・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ー・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/secsample.jar
com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
../security/client.properties manager manager1
```

ユーザー「manager」にはアカウンティング 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 mapSet1  -jp 11001 -ssl
-ts ..¥security¥trust.jks -tsp ogpass -tst jks
-user manager -pwd manager1
```

_ Windows

```
xscmd.bat -c showMapsizes -g accounting -m mapSet1  -jp 11001 -ssl
-ts ..¥security¥trust.jks -tsp ogpass -tst jks
-user manager -pwd manager1
```

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 - mapSet1 **********
*** 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・セキュリティーの構成
- 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** ユーティリティーを使用して、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、コンテナー・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・セキュリティーの構成
- 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。

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ー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、566 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ーを使用するための、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ーを使用して、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 コンテナー・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。

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、277 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ー・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ー・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。

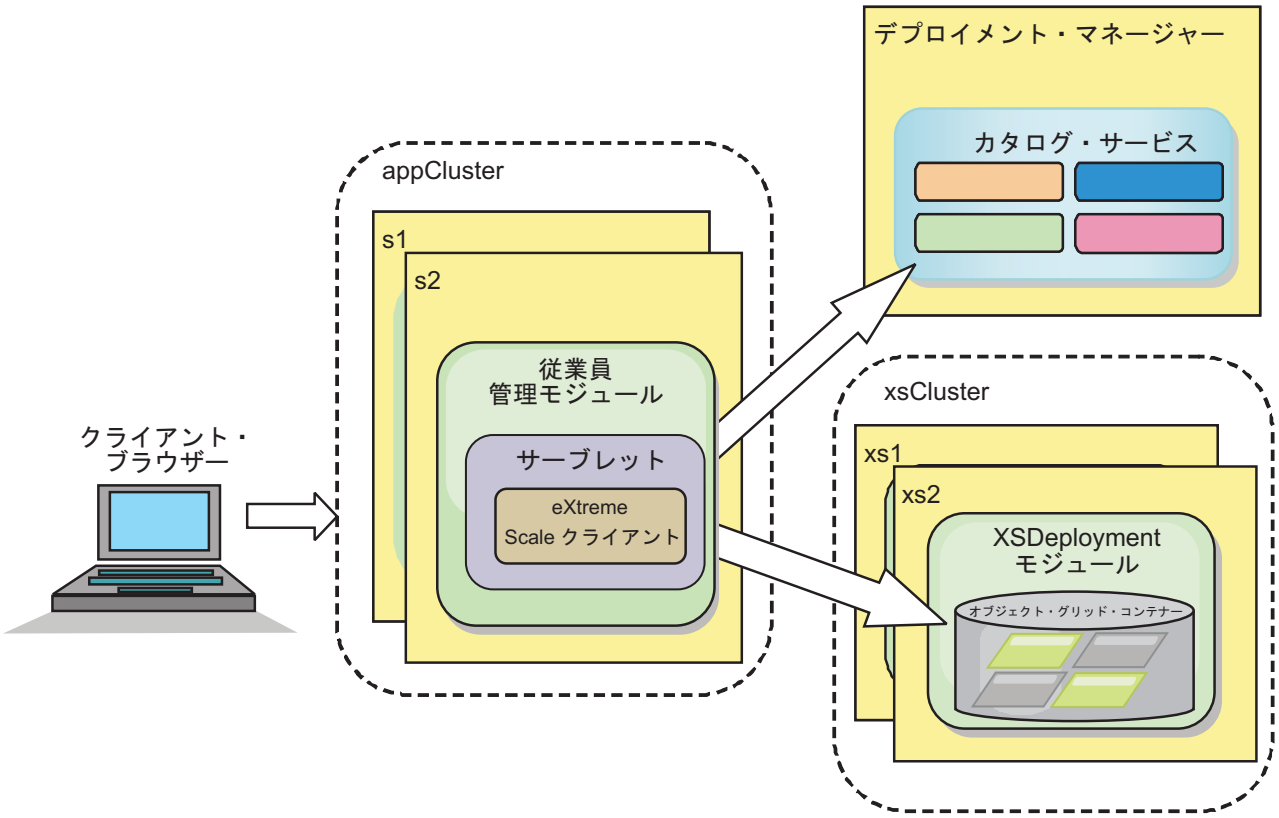


図 19. チュートリアルのトポロジー

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、従業員プロファイ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、コンテナー・サーバーに保管された従業員プロファイルを表示、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、その結果コンテナー・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ー構成が含まれます。
   - 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ー・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/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。

コンテナー・サーバーの自動始動について詳しくは、 296 ページの『コンテナー・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ー・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、「**セキュリティー**」 > 「**グローバル・セキュリティー**」をクリックします。
   b. 「**ユーザー・アカウント・リポジトリー**」に「**統合リポジトリー**」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、「**統合リポジトリー**」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。変更内容を保存します。

f. 「**グローバル・セキュリティー**」ページで、「**統合リポジトリー**」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. 「**管理セキュリティーを使用可能にする**」、「**アプリケーション・セキュリティーを使用可能にする**」、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ーを使用して、WebSphere Application Server 管理セキュリティー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、セキュリティー設定を使用可能にします。

セキュリティーを使用可能にし、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のサンプル・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。

レッスンのチェックポイント:

WebSphere Application Server セルのグローバル・セキュリティーを使用可能にし、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ーを作成しました。

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ー・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。
- コンテナー・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、 566 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ー・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ー・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、 566 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す。

transportType

transportType プロパティーは、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。

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:

デプロイメント・マネージャー・サーバーにサーバー・プロパティー・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、コンテナー・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・プロパティー・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・セキュリティーを構成しました。デプロイメント・マネージャーは、WebSphere Application Server 構成の中のカタログ・サーバー・プロセスをホストします。

レッスン 2.3: コンテナー・サーバー・セキュリティーの構成

コンテナー・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・セキュリティー XML ファイルに構成されているすべてのセキュリティー構成 (オーセンティケーター構成、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。コンテナー・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、コンテナー・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、このコンテナー・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す。

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、コンテナー・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。このセキュリティーを構成することで、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 クラスターを指定して、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、このクラスター内の s1 サーバーと s2 サーバーで実行されます。

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 クラスターを指定して、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。

このクラスター内の xs1 サーバーと xs2 サーバーがコンテナー・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・プロパティー・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・プロパティー・ファイルがロードされました。
/wxs_samples/security/catServer2.props
```

4. xsCluster クラスターを再始動します。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ー・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。

```
CWOBJ0913I: 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。
/wxs_samples/security/server2.props
```

5. appClusters クラスターを再始動します。 クラスター 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。

```
CWOBJ0924I: 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。
```

authenticationRetryCount、transportType、および clientCertificateAuthentication プロパティーに関する警告メッセージは無視してかまいません。プロパティー・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・プロパティー・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 プロパティーを適切に構成したか確認してください。プロパティー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・セキュリティーの構成

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・セキュリティーを構成します。

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、コンテナー・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ー・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・セキュリティーは 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・プロパティーとサーバー・プロパティーは、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ー・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・プロパティー・ファイルの中に、SSL プロパティー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 プロパティーのカタログ・サーバー・プロパティー・ファイルへの追加。
- ORB プロパティー・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、およびコンテナー・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 リポジトリー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。

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 プロパティーを編集します。 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・プロパティー・ファイルの中の次の SSL プロパティー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・プロパティーは `catServer2.props` ファイルにあります。keyStore プロパティーと trustStore プロパティーを更新して、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=IBMJSSE2
#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 ストリングですので、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 プロパティーの値を更新します。 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. カタログ・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 プロパティーを構成しました。

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、 107 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モジュールと同様に、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モジュールでは、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。

チュートリアルのこの部分では、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ときに、eXtreme Scale 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 のセキュリティー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ーが設定されている、*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ー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ー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、 107 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の認証モジュールで、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。統合リポジトリーがユーザー・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 プロパティーを使用した 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 プロパティーを設定します。このチュートリアルで使用するサンプル・トポロジーとは異なるトポロジーを使用する場合は、すべてのコンテナー・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. コンテナー・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>/EmployeeManagermentWeb/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>/EmployeeManagermentWeb/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。
8. 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、コンテナー・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` ディレクトリー内に置きます。

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、**「セキュリティー」** > **「グローバル・セキュリティー」** > **「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: プロパティー 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 プロパティーを使用した 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、コンテナー・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. コンテナー・サーバーを再始動します。 このチュートリアルでは、コンテナー・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>/EmployeeManagermentWeb/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、コンテナー・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、コンテナーおよびカタログ・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ーは、WebSphere Application Server MBean セキュリティー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ー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、コンテナー・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、95 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・セキュリティーを構成して、WebSphere Application Server CSIv2 構成と WebSphere eXtreme Scale プロパティー・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。コンテナー・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: コンテナー・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詳しくは、95 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 プロパティー・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。
- カタログ・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、 566 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ー・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。
- コンテナー・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ー・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。

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 コンテナー・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ー・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ー・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になります。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、277 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。

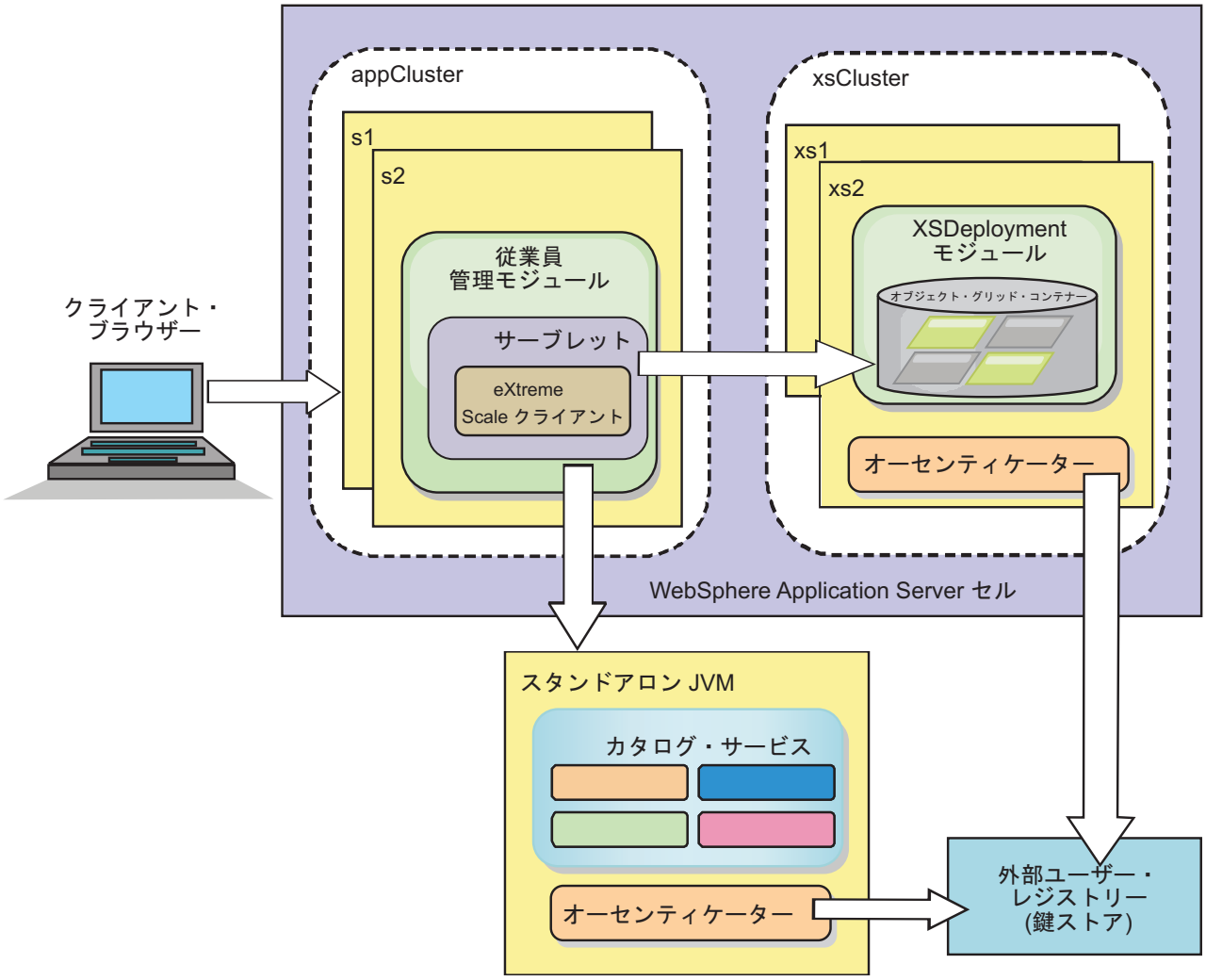


図 20. チュートリアルのトポロジー

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、従業員プロファイ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、コンテナー・サーバーに保管された従業員プロファイルを表示、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ー・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、その結果コンテナー・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ー・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ー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ー・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/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ー・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、178 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、「**セキュリティー**」 > 「**グローバル・セキュリティー**」をクリックします。
2. 「**ユーザー・アカウント・リポジトリー**」に「**統合リポジトリー**」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、「**統合リポジトリー**」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。変更内容を保存します。
6. 「**グローバル・セキュリティー**」ページで、「**統合リポジトリー**」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. 「**管理セキュリティーを使用可能にする**」、「**アプリケーション・セキュリティーを使用可能にする**」、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ーを使用して、WebSphere Application Server 管理セキュリティー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のサンプル・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ー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. カタログ・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 という名前でカタログ・サービス・ドメインを作成し、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します。
3. リモート・サーバーをカタログ・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。カタログ・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

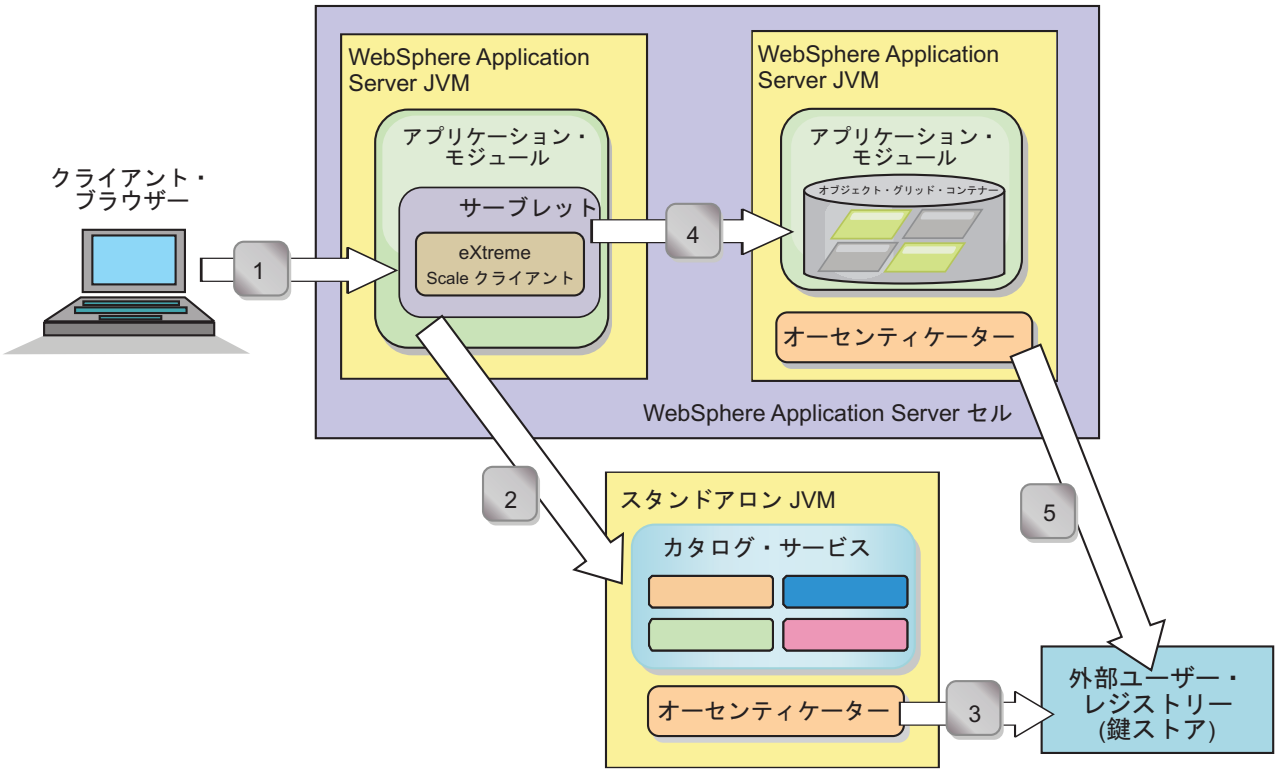


図 21. 認証フロー

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、コンテナー・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、コンテナー・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ー・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ー・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、コンテナー・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 コンテナー・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・プロパティーはプロパティー・ファイルを使用して構成します。クライアント・プロパティー・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・プロパティー・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ー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_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ー・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ー・プロパティーが含まれます。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ー・プロパティーが含まれます。

カタログ・サーバーとコンテナー・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。

Java SE 環境でセキュリティー 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ーは次のとおりです。

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.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。**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・プロパティーを指定できます。このチュートリアルの場合、catServer3.props ファイルは C ディレクトリーにあります。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。クライアント・プロパティー・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す。

transportType

transportType プロパティーは、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・プロパティーを渡す **-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: コンテナー・サーバー・セキュリティーの構成

コンテナー・サーバーがカタログ・サービスに接続すると、コンテナー・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。コンテナー・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、コンテナー・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 プロパティーは true に設定され、このコンテナー・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。クライアント・プロパティー・ファイル内では credentialAuthentication プロパティーが 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す。

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、コンテナー・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**」 > 「**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:

コンテナー・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・モデルのエントリーに追加する必要があります。

1. WebSphere Application Server 管理コンソールで、「**セキュリティー**」 > 「**グローバル・セキュリティー**」 > 「**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。このセキュリティーを構成することで、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 クラスターを指定して、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、このクラスター内の s1 サーバーと s2 サーバーで実行されます。
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 クラスターを指定して、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。

このクラスター内の xs1 サーバーと xs2 サーバーがコンテナー・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詳しくは、 133 ページの『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』を参照してください。
4. xsCluster クラスターを再始動します。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ー・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。

   CWOBJ0913I: 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 クラスターを再始動します。 クラスター 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。

   CWOBJ0924I: 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・プロパティー・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 プロパティーを適切に構成したか確認してください。プロパティー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。

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・セキュリティーの構成

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・セキュリティーを構成します。

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、コンテナー・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ー・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・セキュリティーは 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・プロパティーとサーバー・プロパティーは、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ー・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・プロパティー・ファイルの中で SSL プロパティー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 プロパティーのカタログ・サーバー・プロパティー・ファイルへの追加。
- ORB プロパティー・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、およびコンテナー・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 リポジトリー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の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=IBMJSSE2
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 ストリングですので、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。

これらのエントリーは、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 プロパティーの構成が終わったならば、transportType プロパティー値を TCP/IP から SSL-Required に変更してください。

client3.props ファイル内の SSL プロパティー:

client3.props ファイルの中でも SSL プロパティー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ーは、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・プロパティー・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ーを、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=IBMJSSE2
#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 プロパティーを構成しました。

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、135 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モジュールと同様に、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モジュールでは、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。

チュートリアルのこの部分では、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ときに、eXtreme Scale 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 のセキュリティー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ーが設定されています)、既存の objectGrid.xml ファイルを編集して、セキュリティー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ー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、135 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の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 プロパティーを使用した 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 プロパティーを設定します。このチュートリアルで使用するサンプル・トポロジーとは異なるトポロジーを使用する場合は、すべてのコンテナー・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. コンテナー・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>/EmployeeManagermentWeb/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 プロパティーの値を manager manager1 から operator operator1 に変更します。ファイルを編集すると、サーブレットが 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、コンテナー・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、コンテナー・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。コンテナー・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。

コンテナー・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ーは 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 myCell02¥myNode04¥xs2_C-1 myhost.mycompany.com myCell02¥myNode04¥xs2
   1 myCell02¥myNode04¥xs2_C-1 myhost.mycompany.com myCell02¥myNode04¥xs2
   2 myCell02¥myNode04¥xs2_C-1 myhost.mycompany.com myCell02¥myNode04¥xs2
   3 myCell02¥myNode04¥xs2_C-1 myhost.mycompany.com myCell02¥myNode04¥xs2
   4 myCell02¥myNode04¥xs2_C-1 myhost.mycompany.com myCell02¥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ー・サーバーからマップ・サイズを取得し、コンテナー・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 プロパティーを 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ー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。

Google Protocol Buffers シリアライザー・サンプルは *wxs_home*/samples/SerializerProto ディレクトリーにあります。

Binary JSON (BSON) シリアライザー・サンプルは *wxs_home*/samples/SerializerBSON ディレクトリーにあります。

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 コンテナーに 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ー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 コンテナーを開始し、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 コンテナーに 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 コンテナー・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 コンテナー・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 コンテナー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ると、以下の概念を理解し、次の目的を達成する方法がわかります。

- 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) 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 ディレクトリーにあります。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 コンテナー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 コンテナー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 3 つの構成ファイルが組み込まれています。これらのファイルを使用して、WebSphere eXtreme Scale グリッドおよびサーバーを開始し、構成します。

OSGi 構成ファイル:

このレッスンでは、次の構成ファイルについて検討します。

- `collocated.server.properties`
- `protoBufObjectGrid.xml`
- `protoBufDeployment.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・プロパティー・ファイルが、サーバー名、ポート番号、その他のサーバー・プロパティーを指定します。

ほとんどの場合は、サーバー・プロパティー・ファイルを設定するための構成を作成します。まれに、すべてのプロパティーがデフォルト値に設定された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・プロパティー・ファイルの詳細については、サーバー・プロパティー・ファイルのトピックを参照してください。

OSGi サンプルには、サンプルのサーバー・プロパティー・ファイル *wxs_sample_osgi_root*`/server/properties/collocated.server.properties` が組み込まれています。このサンプル・プロパティー・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ー・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・プロパティー・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 myShardListener を使用するように構成されています。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 を使用するように構成されています。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。この 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

このレッスンのモジュールを使用して、eXtreme Scale サーバー・バンドルを OSGi コンテナーにインストールし、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・プロパティーおよびコンテナー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 コンテナーを構成します。
- eXtreme Scale サンプル・バンドルの開始。

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・プロパティー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
- ProtoBufSamplePlugins-2.0.0.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 の開始およびインストールを行います。

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。

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
```

3. 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ー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、コンテナーを作成します。

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・プロパティーをカスタマイズし、WebSphere eXtreme Scale サーバーに追加します。

1. wxs_sample_osgi_root/server/properties/collocated.server.properties ファイルを編集します。
   a. workingDirectory プロパティーを equinox_root に変更します。
   b. traceFile プロパティーを 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。

   ```
   osgi> cm create com.ibm.websphere.xs.server

   osgi> cm put com.ibm.websphere.xs.server
   objectgrid.server.props
   wxs_sample_osgi_root/server/properties/collocated.serve
   r.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 objectgrid.server.props
```

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/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 コンテナーの構成

このレッスンを実行して、コンテナーを構成します。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。

コンテナーを作成するには、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、コンテナー・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・プロパティーをカスタマイズし、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ーを作成します。

   ```
   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/server/META-INF/protoBufObjectgrid.xml

   osgi> cm put com.ibm.websphere.xs.container-1291179621421-0
   deploymentPolicyFile wxs_sample_osgi_root/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 コンテナー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、作成したコンテナーをそれらの ObjectGrid XML ファイルにバインドする必要がありました。この構成により、eXtreme Scale コンテナーが、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。

次のステップを実行して、Google Protocol Buffers バンドル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、バンドルをインストールします。

```
osgi> install file:///wxs_sample_osgi_root/common/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 ディレクトリーにあります。

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/common/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. サンプル・プラグイン・バンドルを開始します。 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
   ```

2. Google Protocol Buffers バンドルを開始します。 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
   ```

3. サーバー・バンドルを開始します。 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 O MyShardListener@1253853884(version=1.0.0) order
com.ibm.websphere.samples.xs.serializer.proto.DataObjects1$Order$Builder
@1aba1aba(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 コンテナーの中で、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。eXtreme Scale ユーザー・ライブラリーを定義して、複数あるビルド・エラーを修正する必要があります。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. 「**新規**」をクリックします。
8. 「**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。
9. 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。
   a. *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。
   b. 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。

c. 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。

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、eXtreme Scale ユーザー・ライブラリーを定義し、サンプル・プロジェクトのサポート用 API 資料を組み込みました。これで、サンプル・クライアント・アプリケーションを開始する準備ができました。

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>
```

次のステップを完了して、クライアント・アプリケーションを開始します。

1. 次のコーディング例を使用して、使用している環境を反映するように、Client クラスの属性を変更します。

```
private String catHost = "localhost";
private int catListenerPort = 2809;
private String clientOGXML = "wxs_sample_osgi_root/client/META-INF/
clientProtoBufObjectgrid.xml";
private String gridName = "Grid";
private String mapName = "Map";
```

2. 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$Or
der$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。

サンプル・アプリケーションを実行する便利な方法として Eclipse プロジェクトが用意されています。

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

CWXSI0040I: 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

CWXSI0040I: 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.

     CWXSI0040I: 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
   myShardListener      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。
  1. 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。
  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, 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。

1. 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。

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 -g Grid -ms MapSet -sr
"myShardListener;2,myProtoBufSerializer;2"
```

以下の出力が表示されます。

```
CWXSI0040I: 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。

```
./xsadmin.sh -c osgiCheck -g Grid -ms MapSet -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. 次のコマンドを入力して、名前がそれぞれ myShardListener と myProtoBufSerializer のサービスのサービス・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

CWXSI0040I: 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 O 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** コマンドを使用して、Grid という名前の ObjectGrid および MapSet という名前のマップ・セットによって使用されるすべてにサービスに対して、使用する現行サービス・ランキングを照会する場合、次のようにします。
   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

      CWXSI0040I: 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、スケーラビリティー、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、 54 ページの『ハードウェアおよびソフトウェアの要件』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、 54 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、 229 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

フルインストールまたはクライアント・インストールのいずれかを使用して、WebSphere eXtreme Scale をスタンドアロン環境または WebSphere Application Server 環境にインストールできます。

インストール・タイプ

サポート・サイトからダウンロード可能なフルインストーラーおよび別個のクライアント・インストーラーは、さまざまなインストール・オプションを提供します。フルインストーラーを使用すると、カタログ・サーバーとコンテナー・サーバーの両方を実行できます。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。カタログ・サーバーまたはコンテナー・サーバーを実行しているノード上では、サーバー・インストール、またはサーバーおよびクライアントのインストールを使用します。

- **フルインストール:**
  - 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。クライアントのみをインストールしたい場合は、WebSphere eXtreme Scale クライアントのインストールを使用します。

- **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。クライアントのみをインストールするために、サポート・サイトのダウンロード・セクションから、適切なプラットフォーム用のクライアントのみのインストーラーをダウンロードできます。

環境オプション

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 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ー・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ー・サーバーのプロセスを手動で構成し、開始します。

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。次に、作成できるトポロジーの例をいくつか示します。

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。

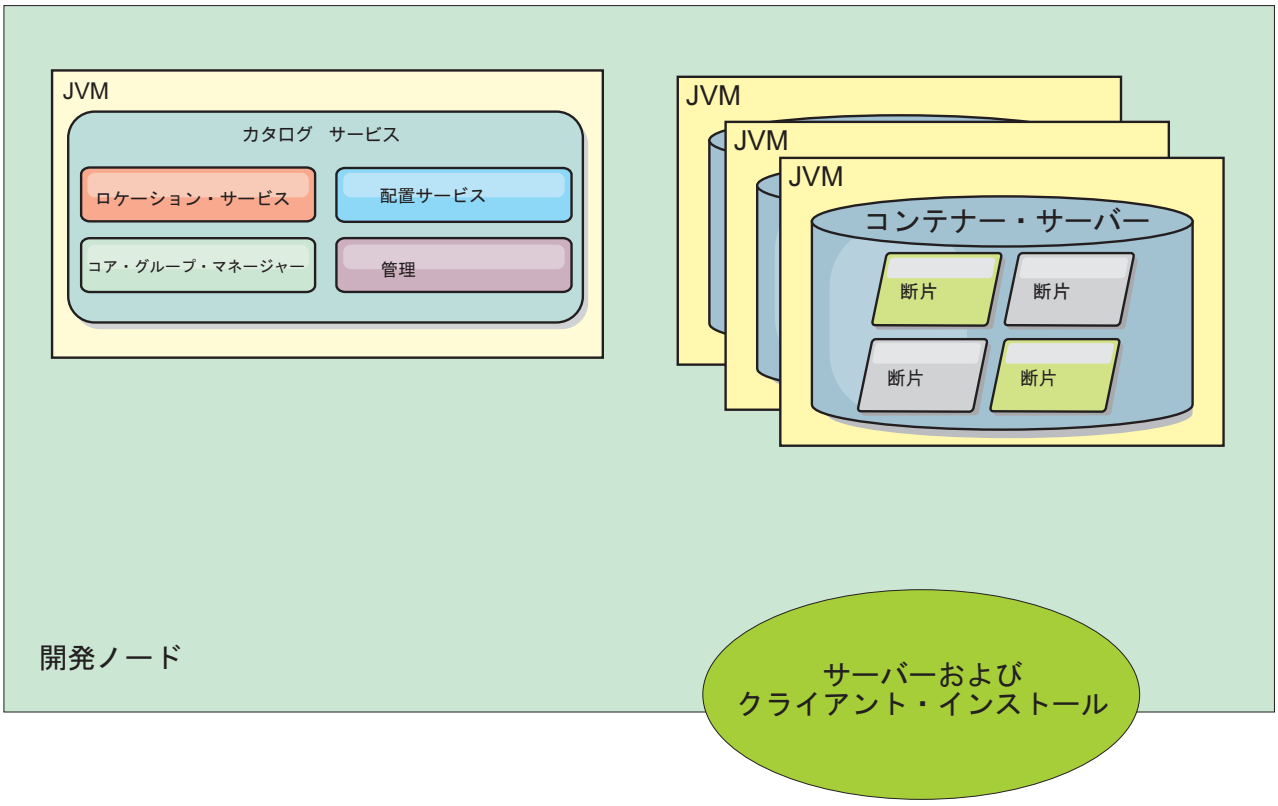


図 22. 開発ノード

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ー・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ー・サーバー内の断片間のレプリカ生成が可能になります。

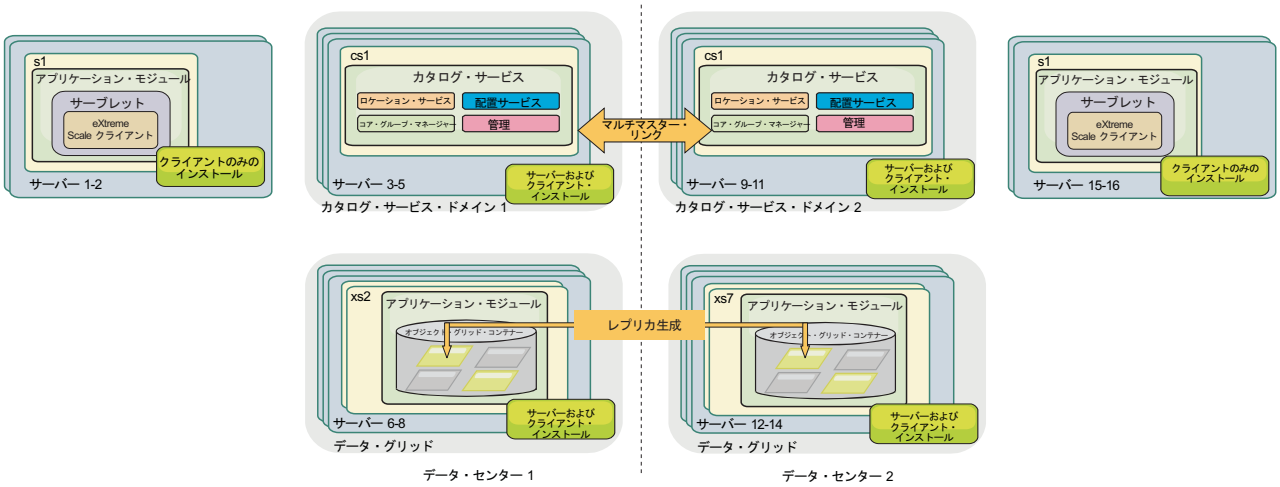


図 23. 2 つのデータ・センターがあるスタンドアロン・トポロジー

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、およびコンテナー・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。

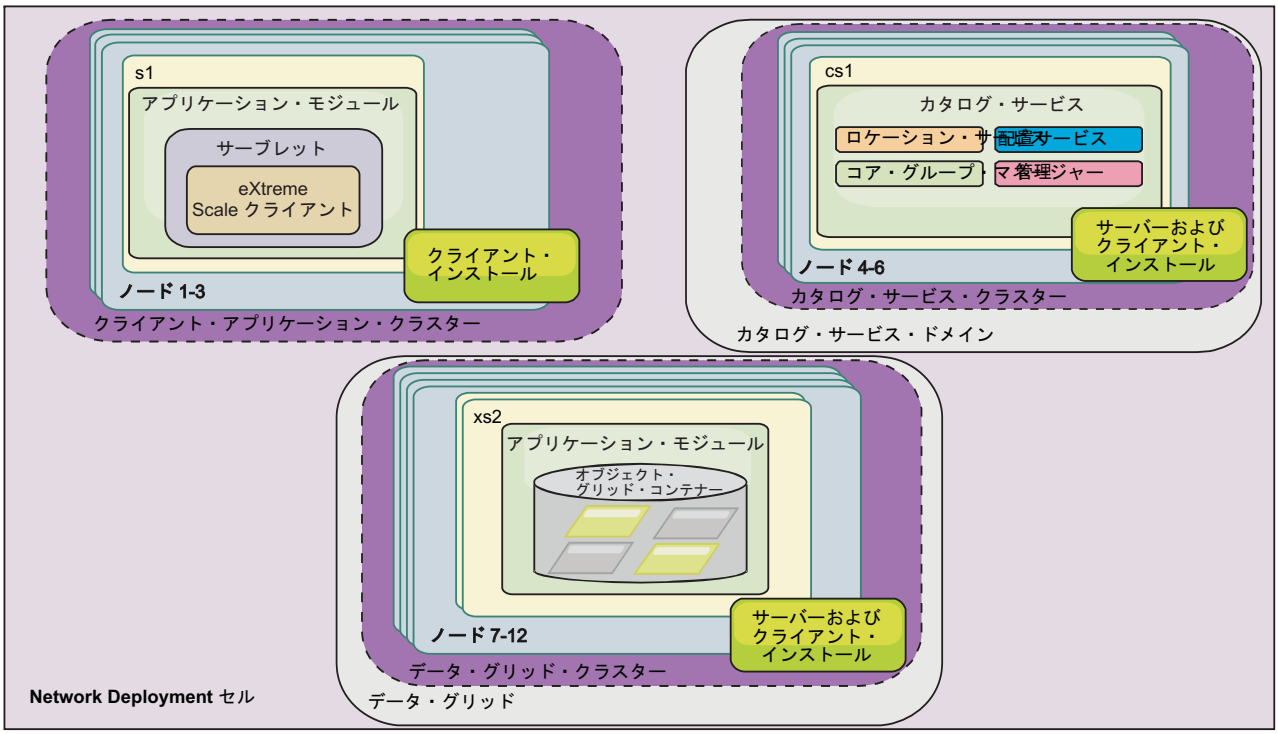


図 24. WebSphere Application Server トポロジー例

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ー・サーバーはスタンドアロン・モードで実行されます。

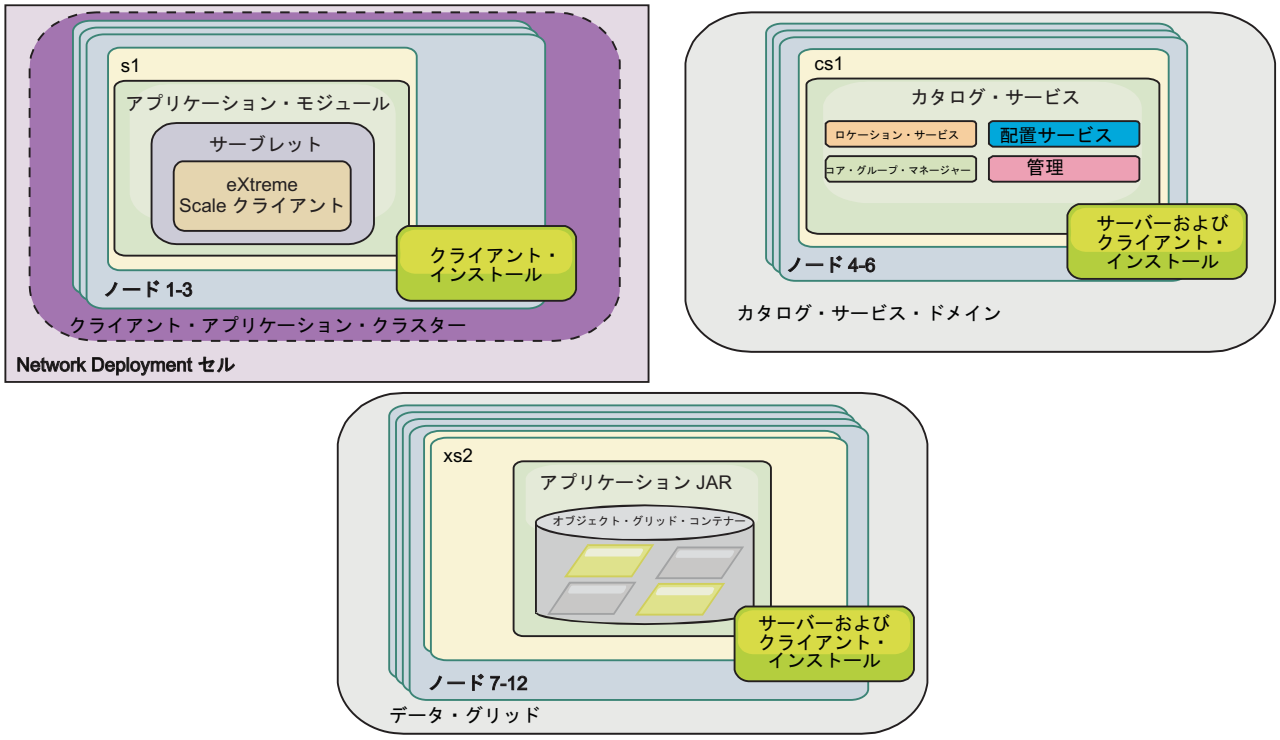


図 25. 混合トポロジー例

ハードウェアおよびソフトウェアの要件

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

- **Web コンソールを使用しない場合**

  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。

- **Web コンソールを使用する場合**

  コンソールを使用する場合、それぞれのオペレーティング・システムについて以下の要件が適用されます。

  – Linux: 32 ビットまたは 64 ビット JVM
  – Linux PPC: 32 ビット JVM のみ
  – Windows: 32 ビット JVM のみ
  – AIX: 32 ビット JVM のみ

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。

- 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

WebSphere Application Server 要件

- WebSphere Application Server バージョン 6.1.0.39 以降
- WebSphere Application Server バージョン 7.0.0.19 以降
- WebSphere Application Server バージョン 8.0.0.1 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 では、Java SE 5 以降が必要です。一般に、Java SE は、バージョンが新しい方が機能および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 以降と一緒に使用できます。使用しているバージョンは、現在、Java ランタイム環境 (JRE) ベンダーによってサポートされていなければなりません。

完全にサポートされている JRE は、*wxs_install_root*/java ディレクトリーにスタンドアロン 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インストールされており、クライアントとサーバーの両方で使用できるようになっています。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。

WebSphere eXtreme Scale は Java Development Kit (JDK) 5 を使用し、それ以降が使用可能になると、そちらの機能を使用します。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 または Java SE 6 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 以降でサポートされている	Java SE 6 以降でサポートされている
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: • 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。 • ogagent.jar: フィールド・アクセス・エンティティーのパフォーマンスを向上させます。	X	X

表 4. Java SE 5 または Java SE 6 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 以降でサポートされている	Java SE 6 以降でサポートされている
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:

  例: `/opt/IBM/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:

  例: `/opt/IBM/eXtremeScale/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

- 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。

例: /opt/IBM/WebSphere/eXtremeScale/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。

例: /wxs-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*

インストール・ウィザードによる WebSphere eXtreme Scale のインストール

インストール・ウィザードを使用して、スタンドアロン構成または WebSphere Application Server 構成の WebSphere eXtreme Scale をインストールできます。

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 または WebSphere eXtreme Scale クライアントを、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

- WebSphere Application ServerまたはWebSphere Application Server Network Deployment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。
- インストールするバージョン (バージョン 6.1 またはバージョン 7.0) に基づいて、WebSphere Application Server または WebSphere Application Server Network Deployment の最新のフィックスパックを適用して、製品レベルを更新してください。詳しくは、WebSphere Application Server の最新フィックスパックを参照してください。
- 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。

- クライアントのみをインストールする場合、DVD を使用してクライアントをインストールするか、サポート・サイトの「ダウンロード」セクションから、特定のプラットフォームの WebSphere eXtreme Scale クライアントをダウンロードします。

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・グリッドをホストし、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. ウィザードを使用して、インストールを完了します。
   - 以下のスクリプトを実行して、WebSphere eXtreme Scale フルインストール用のウィザードを開始します。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。
     - Linux UNIX *`dvd_root`*`/install`
     - Windows *`dvd_root`*`¥install.bat`
   - 以下のスクリプトを実行して、WebSphere eXtreme Scale クライアントのインストール用のウィザードを開始します。インストール・ファイルは、サポート・サイトの「ダウンロード」セクションからダウンロードする zip ファイルに入っています。
     - Linux UNIX `root/WXS_Client/install`
     - Windows `root¥WXS_Client¥install.bat`

   重要: uniform naming conventions (UNC) を使用してインストール・コマンドのファイル・パスを識別する場合、コマンドを実行した後、インストールする予定だった項目のいくつかがインストールされていないことがあります。問題を避けるために、ファイル・パスをネットワーク・ドライブにマップします。**`install`** コマンドをマップされたドライブに対して実行します。マップされたネットワーク・ドライブを使用すると、確実にすべての項目がインストールされます。
2. ウィザードのプロンプトに従います。

   オプション・フィーチャー・パネルに、インストールを選択できるフィーチャーがリストされます。ただし、製品のインストール後に、その製品環境にフィーチャーを順次追加することはできません。初期の製品インストール時にフィーチャーのインストールを選択しなかった場合、そのフィーチャーを追加するには、製品をアンインストールしてから再インストールする必要があります。

プロファイル拡張パネルには、eXtreme Scale のフィーチャーで拡張するために選択できる既存プロファイルがリストされます。既に使用中の既存プロファイルを選択すると、警告パネルが表示されます。インストールを続行するには、そのプロファイルに構成されているサーバーを停止するか、「**戻る**」をクリックして選択からそのプロファイルを除去します。

タスクの結果

Windows WebSphere eXtreme Scale クライアントを Windows にインストールした場合には、インストールの結果で以下のテキストが表示されることがあります。

```
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.
```

`iscdeploy.sh` ファイルで障害が発生しても、エラーを無視して構いません。このエラーでは、問題は生じません。

次のタスク

- WebSphere Application Server バージョン 6.1 またはバージョン 7.0 を実行している場合、プロファイル管理ツール・プラグインまたは **manageprofiles** コマンドが使用できます。詳しくは、199 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。
- インストールを検査します。詳しくは、224 ページの『インストールの検査』を参照してください。
- WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント インストール済み環境の構成を開始します。詳しくは、226 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。

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
wxsdynacache.jar	クライアントおよびサーバー	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。
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 インスツルメンテーション・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。

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
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。
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。
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	ユーティリティー	optionalLibraries/ObjectGrid	xsadmin.jar ファイルには、eXtreme Scale 管理サンプル・ユーティリティーが含まれています。
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。
wxshyperic.jar	ユーティリティー	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	ユーティリティー	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・ユーティリティーが含まれています。
splicer.jar	ユーティリティー	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
wxsdynacache.jar	クライアントおよびサーバー	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。

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
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 インスツルメンテーション・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。
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。
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	ユーティリティー	optionalLibraries/ObjectGrid	xsadmin.jar ファイルには、eXtreme Scale 管理サンプル・ユーティリティーが含まれています。
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。
wxshyperic.jar	ユーティリティー	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	ユーティリティー	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・ユーティリティーが含まれています。
splicer.jar	ユーティリティー	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・ユーティリティーが含まれています。

Installation Factory プラグインを使用したカスタマイズ・パッケージの作成およびインストール

カスタマイズ・インストール・パッケージ (CIP) または統合インストール・パッケージ (IIP) を作成するには、 IBM® Installation Factory plug-in for WebSphere eXtreme Scale を使用します。 CIP には単一の製品インストール・パッケージとオプションの各種資産が含まれています。 IIP は、1 つ以上のインストール・パッケージを組み合わせて、自分でデザインした 1 つのインストール・ワークフローにします。

始める前に

eXtreme Scale のカスタマイズ・パッケージを作成してインストールする前に、次の製品をまずダウンロードしてください。

- IBM Installation Factory for WebSphere Application Server
- IBM Installation Factory plug-in for WebSphere eXtreme Scale

このタスクについて

Installation Factory を使用して、単一の製品コンポーネントを保守パッケージ、カスタマイズ・スクリプト、その他のファイルと組み合わせることによって、CIP が作成できます。 IIP を作成した場合、個別のコンポーネントまたはインストール・パッケージを単一のインストール・パッケージに集約します。

ビルド定義ファイル:

ビルド定義ファイルは、カスタマイズ・インストール・パッケージ (CIP) または統合インストール・パッケージ (IIP) をビルドしてインストールする方法を指定する XML 文書です。IBM Installation Factory for WebSphere eXtreme Scale は、ビルド定義ファイルのパッケージ詳細を読み取り、CIP または IIP を生成します。

CIP または IIP を作成する前に、各カスタマイズ・パッケージのビルド定義ファイルを作成する必要があります。ビルド定義ファイルは、インストールする製品コンポーネントまたはインストール・パッケージ、CIP または IIP のロケーション、組み込む保守パッケージ、インストール・スクリプト、および組み込むように選択したその他のファイルを説明します。IIP のビルド定義ファイル内で Installation Factory が各インストール・パッケージをインストールする順序を指定することもできます。

ビルド定義ウィザードによって、ビルド定義ファイルの作成プロセスを実行することができます。また、ウィザードを使用して、既存のビルド定義ファイルを変更することもできます。ビルド定義ウィザードの各パネルには、パッケージ ID、ビルド定義のインストール・ロケーション、カスタマイズ・パッケージのインストール・ロケーションなど、カスタマイズ・パッケージに関する情報を求めるプロンプトが出ます。この情報はすべて新規のビルド定義ファイルに保存されるか、変更されて既存のビルド定義ファイルに保存されます。詳しくは、CIP ビルド定義ウィザード・パネルおよび IIP ビルド定義ウィザード・パネルを参照してください。

ビルド定義ファイルのみを作成するには、コマンド行インターフェース・ツールを使用して、GUI の外部でカスタマイズ・パッケージを生成することができます。詳しくは、191 ページの『CIP または IIP のサイレント・インストール』を参照してください。

ビルド定義ファイルの作成と CIP の生成:

IBM Installation Factory plug-in for WebSphere eXtreme Scale は、ビルド定義ファイルに指定した詳細に従って、カスタマイズ・インストール・パッケージ (CIP) を生成します。ビルド定義では、インストールする製品パッケージ、CIP のロケーション、インストールに組み込む保守パッケージ、インストール・スクリプト・ファイル、および CIP に組み込むその他のファイルを指定します。

このタスクについて

ビルド定義ウィザードを使用して、ビルド定義ファイルを作成し、CIP を生成します。

手順

1. *IF_HOME*/bin ディレクトリーから次のスクリプトを実行して、 Installation Factory を開始します。
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`

   「**ビルド定義の新規作成**」アイコンをクリックします。
2. ビルド定義ファイルに組み込む製品を選択して「**終了**」をクリックし、ビルド定義ウィザードを開始します。
3. ウィザードのプロンプトに従います。

   「インストール・スクリプトとアンインストール・スクリプト」パネルで、「**スクリプトの追加...**」をクリックして、テーブルにカスタマイズ・インストール・スクリプトを取り込みます。スクリプト・ファイルのロケーションを入力し、エラー・メッセージが表示された場合に続行するチェック・ボックスをクリアします。デフォルトでは、操作は停止されます。「**OK**」をクリックしてパネルに戻ります。

タスクの結果

これで、ビルド定義ファイルが作成されてカスタマイズされ、接続モードでの作業を選択している場合には、CIP が生成されます。

ビルド定義ファイルから CIP を生成するオプションが、ビルド定義ウィザードにない場合は、 *IF_HOME*/bin ディレクトリーから `ifcli.sh|bat` スクリプトを実行して生成することができます。

次のタスク

CIP をインストールします。詳しくは、185 ページの『CIP のインストール』を参照してください。

CIP のインストール:

カスタマイズ・インストール・パッケージ (CIP) をインストールすることで、製品インストール処理を簡素化できます。 CIP は、1 つ以上の保守パッケージ、構成スクリプト、およびその他のファイルを含むことができる単一の製品インストール・イメージです。

始める前に

CIP をインストールする前に、ビルド定義ファイルを作成して、CIP に組み込むオプションを指定する必要があります。詳しくは、184 ページの『ビルド定義ファイルの作成と CIP の生成』を参照してください。

このタスクについて

CIP は、単一の製品コンポーネントを保守パッケージ、カスタマイズ・スクリプト、その他のファイルと組み合わせてインストールします。

手順

1. インストールの準備を行うワークステーション上で実行されているすべてのプロセスを停止します。 デプロイメント・マネージャーを停止するには、次のスクリプトを実行します。
   - Linux UNIX *profile_root*/bin/stopManager.sh
   - Windows *profile_root*¥bin¥stopManager.bat

   ノードを停止するには、次のスクリプトを実行します。
   - Linux UNIX *profile_root*/bin/stopNode.sh
   - Windows *profile_root*¥bin¥stopNode.bat
2. 次のスクリプトを実行して、インストールを開始します。
   - Linux UNIX *CIP_home*/bin/install
   - Windows *CIP_home*¥bin¥install.bat
3. ウィザードのプロンプトに従って、インストールを完了します。

   オプション・フィーチャー・パネルに、インストールを選択できるフィーチャーがリストされます。ただし、製品のインストール後に、その製品環境にフィーチャーを順次追加することはできません。初期の製品インストール時にフィーチャーのインストールを選択しなかった場合、そのフィーチャーを追加するには、製品をアンインストールしてから再インストールする必要があります。

   プロファイル拡張パネルには、eXtreme Scale のフィーチャーで拡張するために選択できる既存プロファイルがリストされます。既に使用中の既存プロファイルを選択すると、警告パネルが表示されます。インストールを続行するには、そのプロファイルに構成されているサーバーを停止するか、「**戻る**」をクリックして選択からそのプロファイルを除去します。

タスクの結果

CIP が正常にインストールされました。

次のタスク

WebSphere Application Server バージョン 6.1 またはバージョン 7.0 を実行している場合、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。詳しくは、 199 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。

インストール処理中に eXtreme Scale のプロファイルを拡張した場合には、ご使用の WebSphere Application Server 環境で、アプリケーションのデプロイ、カタログ・サービスの開始、およびコンテナーの開始を実行できます。詳しくは、 276 ページの『WebSphere eXtreme Scale と WebSphere Application Server の構成』を参照してください。

既存の製品インストール済み環境に保守を適用するための CIP のインストール:

カスタマイズ・インストール・パッケージ (CIP) をインストールすることによって、既存の製品インストール済み環境に保守パッケージを適用することができます。 CIP で既存のインストール済み環境に保守を適用する処理を一般的にスリップ・インストール と呼びます。

始める前に

ビルド定義ファイルを作成して、CIP に組み込むオプションを指定します。詳しくは、 184 ページの『ビルド定義ファイルの作成と CIP の生成』を参照してください。

このタスクについて

リフレッシュ・パックまたはフィックスパック、あるいはその両方を含む CIP で保守を適用する場合、以前にインストールされたすべてのプログラム診断依頼書 (APAR) はウィザードによってアンインストールされます。CIP が製品と同じレベルの場合、以前にインストールされた APAR は、CIP 内にパッケージ化されている場合に限り、その状態のままです。既存のインストール済み環境に保守を正常に適用するには、インストールされているフィーチャーを CIP に含める必要があります。

手順

1. インストールの準備を行うワークステーション上で実行されているすべてのプロセスを停止します。 デプロイメント・マネージャーを停止するには、次のスクリプトを実行します。
   - Linux UNIX *profile_root*/bin/stopManager.sh
   - Windows *profile_root*¥bin¥stopManager.bat

   ノードを停止するには、次のスクリプトを実行します。
   - Linux UNIX *profile_root*¥bin¥stopNode.sh

- Windows *profile_root*¥bin¥stopNode.bat

2. 次のスクリプトを実行して、インストールを開始します。
   - Linux UNIX *CIP_home*/bin/install
   - Windows *CIP_home*¥bin¥install.bat
3. ウィザードのプロンプトに従って、インストールを完了します。

   インストール・プレビュー要約には、結果として得られた製品バージョン、適用可能なフィーチャーおよび暫定修正がリストされます。次に、ウィザードが保守を正常に適用し、製品のフィーチャーを更新します。

タスクの結果

製品バイナリー・ファイルが、*was_root*/properties/version/nif/backup ディレクトリーにコピーされます。 IBM Update Installer を使用して更新をアンインストールし、ワークステーションを復元することができます。詳しくは、『既存の製品インストール済み環境からの CIP 更新のアンインストール』を参照してください。

既存の製品インストール済み環境からの CIP 更新のアンインストール:

製品全体を除去せずに、既存の製品インストール済み環境から CIP の更新を除去することができます。 CIP の更新をアンインストールするには、IBM Update Installer バージョン 7.0.0.4 を使用します。このタスクは、スリップ・アンインストール とも呼ばれます。

始める前に

製品の、少なくとも 1 つ以上の既存コピーがシステムにインストールされている必要があります。

手順

1. Update Installer のバージョン 7.0.0.4 を次の FTP サイトからダウンロードします。

   ftp://ftp.software.ibm.com/software/websphere/cw/process_server/FEP/UPDI/7004
2. Update Installer をインストールします。 詳しくは、WebSphere Application Server インフォメーション・センターの Update Installer for WebSphere Software のインストールを参照してください。
3. CIP のインストール後に環境に追加したフィックスパック、リフレッシュ・パック、または暫定修正があれば、それをアンインストールします。
4. スリップ・インストールに組み込んだ暫定修正があれば、それをアンインストールします。 これは、単一のフィックスパックまたはリフレッシュ・パックをアンインストールするプロセスと同じです。ただし、CIP に組み込まれていた保守は現在、単一の操作で組み込まれるようになりました。
5. Update Installer を使用して CIP をアンインストールします。 保守レベルは更新前の状態に戻り、CIP は、ファイル名に接頭部として追加される CIP ID によって示されます。 次の例では、保守パッケージの選択パネルで、他の通常の保守パッケージとは異なって CIP がどのように表示されるかを示しています。

CIP

`com.ibm.ws.cip.7000.wxs.primary.ext.pak`

タスクの結果

既存の製品インストール済み環境から CIP 更新を正常に除去しました。

ビルド定義ファイルの作成と IIP の生成:

IBM Installation Factory plug-in for WebSphere eXtreme Scale は、ビルド定義ファイルによって指定されるプロパティーに基づいて、IIP を生成します。ビルド定義ファイルには、IIP に含めるインストール・パッケージ、Installation Factory が各パッケージをインストールする順序、IIP の場所などの情報が含まれています。

このタスクについて

ビルド定義ウィザードを使用して、ビルド定義ファイルを作成し、IIP を生成します。

手順

1. *IF_HOME*/bin ディレクトリーから次のスクリプトを実行して、 Installation Factory を開始します。
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. 「**統合インストール・パッケージの新規作成**」アイコンをクリックして、ビルド定義ウィザードを開始します。
3. ウィザードのプロンプトに従います。
   a. 「IIP の構成 (Construct the IIP)」パネルで、サポートされるインストール・パッケージをリストから選択し、「**インストーラーの追加**」をクリックして、インストール・パッケージを IIP に追加します。 パッケージ名、パッケージ ID、パッケージ・プロパティーを示したパネルが表示されます。選択したパッケージに関する固有情報を表示するには、「**インストール・パッケージの情報を表示**」をクリックします。「**変更**」をクリックして、各オペレーティング・システムのインストール・パッケージのディレクトリー・パスを入力します。 WebSphere Extended Deployment のインストール・パッケージを現在追加している場合には、サポートされるすべてのオペレーティング・システムに同じパッケージを使用するオプションのチェック・ボックスを選択します。「**OK**」をクリックして、「IIP の構成 (Construct the IIP)」パネルに戻ります。デフォルトで、呼び出しが作成されます。
      - インストール・パッケージのディレクトリー・パスを変更するには、 IIP リストに使用されているインストール・パッケージからパッケージを選択し、「**変更**」をクリックします。
      - 呼び出しを変更するには、呼び出しを選択して、「**変更**」をクリックします。各オペレーティング・システムにおける呼び出しのデフォルトのインストール場所を指定します。デフォルトのインストール・モードにサイレント・インストールを選択する場合には、その場所を応答ファイルに指定します。

- 「**呼び出しの追加**」をクリックして、インストール・パッケージに呼び出しコントリビューションを追加します。呼び出しのプロパティーを指定できるパネルが表示されます。
- インストール・パッケージまたは呼び出しを除去するには、「**除去**」をクリックします。

4. 選択の要約を確認し、「**ビルド定義ファイルを保存し、統合インストール・パッケージを生成する**」オプションを選択してから、「**終了**」をクリックします。

   あるいは、IIP を生成せずにビルド定義ファイルを保存することもできます。このオプションを使用した場合には、*IF_home*/bin/ ディレクトリーから `ifcli.bat | ifcli.sh` スクリプトを実行することによって、ウィザードの外で IIP を実際に生成します。

タスクの結果

これで、IIP のビルド定義ファイルが作成され、カスタマイズされます。

次のタスク

IIP をインストールします。

IIP のインストール:

統合インストール・パッケージ (IIP) をインストールするには、IBM Installation Factory plug-in for WebSphere eXtreme Scale を使用します。 IIP は、1 つ以上のインストール・パッケージを組み合わせて、自分でデザインした 1 つのワークフローにします。

始める前に

CIP をインストールする前に、ビルド定義ファイルを作成して、CIP に組み込むオプションを指定する必要があります。詳しくは、188 ページの『ビルド定義ファイルの作成と IIP の生成』を参照してください。

このタスクについて

IIP には、1 つ以上の一般出荷可能インストール・パッケージ、 1 つ以上の CIP、その他のオプションのファイルおよびディレクトリーを含めることができます。 IIP をインストールすることによって、複数のインストール・パッケージ、つまりコントリビューション を 1 つのパッケージに集約し、コントリビューションを特定の順序でインストールしてすべてのインストールを完了します。

手順

1. 以下のスクリプトを実行して、ウィザードを開始します。
   - Linux UNIX *IIP_home*/bin/install
   - Windows *IIP_home*¥bin¥install.bat
2. 「ようこそ」パネルの「**製品情報 (About)**」をクリックして、パッケージ ID、サポートされるオペレーティング・システム、組み込まれるインストール・パッケージなど、IIP の詳細を表示します。

オプション: 各パッケージのインストール・オプションを変更するには、「**変更**」をクリックします。

オプション: ウィザード・パネルには、「**ログの表示**」ボタンが 2 つ表示されます。各パッケージのログを表示する場合は、インストール・パッケージをリストした表の横に表示される「**ログの表示**」ボタンをクリックします。 IIP の全体のログ詳細を表示する場合は、状況情報の横に表示される「**ログの表示**」ボタンをクリックします。

3. 実行するインストール・パッケージを選択し、「**インストール**」をクリックします。 IIP に含まれるすべてのコントリビューションのリストが、呼び出し順に表示されます。インストール中に実行しないコントリビューション呼び出しを指定するには、「**インストール名**」フィールドの横にあるチェック・ボックスをクリアします。

タスクの結果

IIP が正常にインストールされました。

IIP の既存ビルド定義ファイルの変更:

IIP のプロパティーに編集や追加を行って、インストールをさらにカスタマイズすることができます。

このタスクについて

IIP のプロパティーを変更するには、既存のビルド定義ファイルを変更します。

手順

1. *IF_HOME*/bin ディレクトリーから次のスクリプトを実行して、 Installation Factory を開始します。
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. 「**ビルド定義を開く**」アイコンをクリックし、変更するビルド定義ファイルを選択します。
3. 変更する IIP の具体的なプロパティーを選択します。 以下のリストには、可能な変更が含まれています。
   - 現行モード選択を変更します。接続モードでは、現行ワークステーションから、使用するビルド定義を作成し、また、オプションで IIP を生成します。切断モードでは、別のワークステーションで使用するビルド定義ファイルを作成します。
   - IIP がサポートする既存のオペレーティング・システムを追加または除去します。
   - IIP の既存の ID およびバージョンを編集します。
   - ビルド定義ファイルのターゲット・ロケーションを編集します。
   - IIP のターゲット・ロケーションを編集します。

- IIP のインストール・ウィザードを表示するかどうかを変更します。ウィザードでは、IIP に関する情報と、IIP 実行時のインストール・オプションが示されます。
- IIP に含まれるインストール・パッケージを追加、除去、および編集します。

  重要: サポートされるオペレーティング・システムを追加し、IIP 内のインストール・パッケージのプロパティーを更新していないと、選択したコントリビューションに、IIP がサポートするすべてのオペレーティング・システムに識別されているインストール・パッケージが含まれないことを示す警告メッセージを受け取ります。続行する場合には「**はい**」、インストール・パッケージを編集する場合には「**いいえ**」をクリックします。

4. 選択の要約を確認し、「**ビルド定義ファイルを保存し、統合インストール・パッケージを生成する**」を選択し、「**終了**」をクリックします。

CIP または IIP のサイレント・インストール:

ニーズに具体的に対応して構成する完全修飾応答ファイル、またはコマンド行に受け渡すパラメーターのいずれかを使用して、製品のカスタマイズ・インストール・パッケージ (CIP) または統合インストール・パッケージ (IIP) をサイレントにインストールすることができます。

始める前に

CIP または IIP のビルド定義ファイルを作成します。詳しくは、184 ページの『ビルド定義ファイルの作成と CIP の生成』を参照してください。

このタスクについて

サイレント・インストールは、グラフィカル・ユーザー・インターフェース (GUI) バージョンが使用するのと同じインストール・プログラムを使用します。ただし、ウィザード・インターフェースを表示する代わりに、サイレント・インストールは、カスタマイズされたファイルから、あるいはコマンド行にパスされたパラメーターからすべての応答を読み取ります。 IIP をサイレントにインストールする場合、コントリビューションの呼び出しには、応答ファイルに指定したオプションのほかに、コマンド行に直接指定したオプションを組み合わせて使用できます。ただし、コマンド行にコントリビューション・オプションを渡すと、IIP インストーラーは、特定のコントリビューションの応答ファイルに指定されたオプションをすべて無視します。詳しくは、IIP のサイレント・インストールを参照してください。

注: 完全修飾応答ファイル名を指定してください。相対パスを指定すると、エラーが発生したことをまったく示さずにインストールが失敗します。

手順

1. オプション: 応答ファイルを使用して CIP または IIP をインストールする場合は、まずファイルをカスタマイズします。
   a. 応答ファイル `wxssetup.response.txt` を製品 DVD からディスク・ドライブにコピーします。
   b. 任意のテキスト・エディターで応答ファイルを開き、編集します。 このファイルには、構成プロセスを支援するコメントが含まれています。ファイルには、次のパラメーターを組み込む必要があります。

- ご使用条件
- 製品インストールのロケーション

ヒント: インストーラーは、インストールに選択したロケーションを使用して WebSphere Application Server インスタンスのインストール場所を判別します。複数の WebSphere Application Server インスタンスが含まれるノードにインストールする場合、そのロケーションを明確に定義してください。

c. 次のスクリプトを実行して、カスタマイズ応答ファイルを開始します。

- Linux UNIX `install -options /absolute_path/response_file.txt -silent`
- Windows `install.bat -options C:¥drive_path¥response_file.txt -silent`

2. オプション: 特定のパラメーターをコマンド行に渡すことによって CIP または IIP をインストールする場合は、次のスクリプトを実行してインストールを開始します。

- Linux UNIX `install -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=install_location`
- Windows `install.bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=install_location`

ここで、*install_location* は、既存の WebSphere Application Server インストールのロケーションです。

3. 結果ログを検討して、エラーやインストールの失敗を調べます。

タスクの結果

CIP または IIP がサイレントにインストールされました。

次のタスク

WebSphere Application Server バージョン 6.1 またはバージョン 7.0 を実行している場合、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。

インストール処理中に eXtreme Scale のプロファイルを拡張した場合には、ご使用の WebSphere Application Server 環境で、アプリケーションのデプロイ、カタログ・サービスの開始、およびコンテナーの開始を実行できます。詳しくは、276 ページの『WebSphere eXtreme Scale と WebSphere Application Server の構成』を参照してください。

***wxssetup.response.txt* ファイル:**

完全修飾応答ファイルを使用して、WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント をサイレント・インストールできます。

注意:
インストール場所のパスの最後に、/ や ¥ など、末尾のスラッシュを追加しないでください。これらのパスは、installLocation 属性で指定されます。インストール場所の最後にスラッシュを追加すると、インストールが失敗することがあります。例えば次のパスは、インストール失敗の原因となります。

```
-OPT installLocation="/usr/IBM/WebSphere/eXtremeScale/"
```

このパスは次のように指定する必要があります。

```
-OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
```

WebSphere eXtreme Scale フルインストール用の応答ファイル

```
################################################################################
#
# IBM WebSphere eXtreme Scale V7.1.1 InstallShield Options File
#
# ウィザード名: インストール
# ウィザード・ソース: setup.jar
#
# ウィザードを「-options」コマンド行オプションを指定して実行した場合に、この
# ファイルを使用して、以下で指定したオプションでインストールを構成できます。
# 値の変更方法については、各設定の説明を参照してください。
# すべての値を一対の二重引用符で囲ってください。
#
# オプション・ファイルの一般的用途は、サイレント・モードでのウィザード実行です。
# オプション・ファイルの作成者は、グラフィカル・モードまたはコンソール・モードで
# ウィザードを実行せずに、ウィザード設定を指定できます。サイレント・モードの
# 実行でこのオプション・ファイルを使用するには、ウィザード実行時に以下の
# コマンド行引数を使用します。
#
#    -options "D:¥installImage¥WXS¥wxssetup.response" -silent
#
# なお、完全修飾応答ファイル名を使用する必要があります。
#
################################################################################

################################################################################
#
# ご使用条件の受け入れ
#
# 有効値:
#  true - ご使用条件を受け入れます。製品をインストールします。
#  false - ご使用条件に同意しません。インストールされません。
#
# インストールが行われない場合は、ユーザーの一時ディレクトリーにある
# 一時ログ・ファイルにそのことが記録されます。
#
# この応答ファイルの silentInstallLicenseAcceptance プロパティーを true に
# 変更することで、このプログラムに付属の IBM プログラムのご使用条件を確認して
# 同意したことになります。このご使用条件は、
# CD_ROOT¥XD¥wxs.primary.pak¥repository¥legal.xs¥license.xs にあります。
# このご使用条件に同意しない場合には、値を変更しないでください。また、
# プログラムをダウンロード、インストール、コピー、アクセス、
# 使用しないでください。入手元へプログラムおよびライセンス証書を速やかに
# 返却して、支払額を払い戻してください。
#
-OPT silentInstallLicenseAcceptance="false"


################################################################################
# ノンブロッキング前提条件検査
#
# ノンブロッキング前提条件検査を使用不可にする場合は、以下の行のコメントを
# 外してください。これにより、前提条件検査が失敗した場合でも、インストールを
```

```
# 進め、警告を記録するように、インストーラーに指示します。
#
#-OPT disableNonBlockingPrereqChecking="true"


################################################################################
#
# インストール場所
#
# 製品のインストール場所。製品をインストールする有効なディレクトリーを指定
# します。ディレクトリーにスペースが含まれている場合には、以下の Windows の
# 例のように、二重引用符でディレクトリーを囲みます。なお、インストール場所に
# スペースを含められるのは、Windows オペレーティング・システムのみです。
# Windows の場合、最大パス長は 60 文字です。
#
# 以下に、root ユーザーでインストールする場合の各オペレーティング・システムの
# デフォルト・インストール場所のリストを示します。デフォルトでは、この応答
# ファイルでは、Windows のインストール場所が使用されています。別の
# オペレーティング・システムのデフォルト・インストール場所を使用する場合は、
# 以下で、該当デフォルト・インストール場所項目のコメントを外し (「#」を削除)、
# Windows オペレーティング・システムの項目をコメント化 (「#」を追加) します。
#
# インストール場所は、WebSphere eXtreme Scale をスタンドアロン・デプロイメント
# としてインストールするのか、既存の WebSphere Application Server インストール
# 済み環境に統合するのかを決定するために使用されます。
#
# 指定場所が既存の WebSphere Application Server または WebSphere Network
# Deployment インストール済み環境の場合、eXtreme Scale は既存の WebSphere
# Application Server に統合されます。指定場所が新規または空のディレクトリー
# の場合は、WebSphere eXtreme Scale は、スタンドアロン・デプロイメントとして
# インストールされます。
#
# 注: 指定インストール場所に WebSphere eXtreme Scale、WebSphere eXtended
# Deployment DataGrid、または ObjectGrid の前のインストール済み環境が
# 存在する場合には、インストールは失敗します。
#
# AIX のデフォルト・インストール場所:
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# HP-UX、Solaris、または Linux のデフォルト・インストール場所:
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Windows のデフォルト・インストール場所:
#
-OPT installLocation="C:¥Program Files¥IBM¥WebSphere¥eXtremeScale"

#
# Unix で非 root ユーザーとして、または Windows で非管理者としてインストール
# する場合は、以下のデフォルト・インストール場所をお勧めします。選択する
# インストール場所に対する書き込み許可が備わった状態にしてください。
#
# AIX のデフォルト・インストール場所:
#
#  -OPT installLocation="<ユーザーの home>/IBM/WebSphere/eXtremeScale"
#
# HP-UX、Solaris、または Linux のデフォルト・インストール場所:
#
#  -OPT installLocation="<ユーザーの home>/IBM/WebSphere/eXtremeScale"
#
# Windows のデフォルト・インストール場所:
#
# -OPT installLocation="C:¥IBM¥WebSphere¥eXtremeScale"
```

```
################################################################################
#  オプション・フィーチャーのインストール
#
# 目的の各フィーチャーを「true」に設定することで、インストールするオプション・
# フィーチャーを指定します。インストールしないオプション・フィーチャーは、
# 「false」に設定します。
#
# オプション selectServer、selectClient、selectPF、および selectXSStreamQuery
# は、上述の installLocation オプションに WebSphere Application Server の
# インストール済み環境が含まれている場合にのみ有効です。これらのオプションは、
# WebSphere eXtreme Scale スタンドアロン・インストール済み環境では無視されます。
#
# WebSphere eXtreme Scale スタンドアロン・インストール済み環境では、eXtreme
# Scale サーバーおよびクライアントは自動的にインストールされます。eXtreme
# Scale スタンドアロン・インストール済み環境のフィーチャー・オプションは、
# selectXSConsoleOther および selectXSStreamQueryOther です。

#
# このオプションを選択すると、WebSphere eXtreme Scale コンソールの実行に
# WebSphere eXtreme Scale servers and the eXtreme Scale dynamic cache service
# provider. このオプションを選択した場合は、WebSphere eXtreme
# Scale クライアントも選択する (コメントを外して値を「true」に設定する) 必要が
# あります。
# そうしないと、サイレント・インストールが失敗します。
#
-OPT selectServer="true"

#
# このオプションを選択すると、WebSphere eXtreme Scale コンソールの実行に
# WebSphere eXtreme Scale client applications. 上の
# Server オプションを選択している場合は、このオプションも選択 (コメントを外して
# 値を「true」に設定) する必要があります。そうしないとサイレント・インストール
# が失敗します。
#
-OPT selectClient="true"

#
# このオプションを選択すると、WebSphere eXtreme Scale コンソールの実行に
# 必要なコンポーネントがインストールされます。このオプションを選択する場合は、
# コンソール・オプションは WebSphere eXtreme Scale スタンドアロン・デプロイメント
# でのみ有効であるため、上で指定するインストールの場所は新規ディレクトリー
# または空のディレクトリーにする必要があります。
# このオプションをインストールするには、以下のオプション行のコメントを外して、
# 値を「true」に設定します。
#-OPT selectXSConsoleOther="false"

#
# 以下のオプションを選択すると、非推奨機能がインストールされます。
#
# このオプションで、WebSphere Partition Facility がインストールされます。
# この機能は非推奨です。このオプションをインストールするには、以下の
# オプション行のコメントを外して、値を「true」に設定します。
#
#-OPT selectPF="false"

#
# このオプションにより、WebSphere eXtreme Scale StreamQuery for WAS が
# インストールされます。この機能は非推奨です。このオプションをインストール
# するには、以下のオプション行のコメントを外して、値を「true」に設定します。
# このオプションを選択した場合には、WebSphere eXtreme Scale クライアントも
# 選択 (コメントを外して、値を「true」に設定) する必要があります。
# そうしないと、サイレント・インストールが失敗します。
#
#-OPT selectXSStreamQuery="false"

#
# このオプションにより、WebSphere eXtreme Scale StreamQuery for J2SE が
```

```
# インストールされます。この機能は非推奨です。このオプションをインストール
# するには、以下のオプション行のコメントを外して、値を「true」に設定します。
# このオプションを選択した場合には、WebSphere eXtreme Scale クライアントも
# 選択 (コメントを外して、値を「true」に設定) する必要があります。
# そうしないと、サイレント・インストールが失敗します。
#
#-OPT selectXSStreamQueryOther="false"


################################################################################
# 拡張用プロファイル・リスト
#
# 拡張する既存プロファイルを指定するか、行をコメント化してインストールで
# 検出されたすべての既存プロファイルを拡張します。
#
# 複数のプロファイルを指定するには、コンマで各プロファイル名を区切ります。
# 例えば、「AppSrv01,Dmgr01,Custom01」。リスト内でスペースは使用できません。
#
-OPT profileAugmentList=""


################################################################################
# トレース制御
#
# このオプションで、トレース出力フォーマットを制御できます。
# -OPT traceFormat=ALL
#
# フォーマットとして、「text」および「XML」を選択できます。デフォルトでは、
# 両方のフォーマットが 2 つの異なるトレース・ファイルに生成されます。
#
# 片方のフォーマットのみが必要な場合は、以下のように、traceFormat オプションで
# フォーマットを指定します。
#
# 有効値:
#
#  text -  簡単に読めるように、トレース・ファイルの行をプレーン・テキスト・
#          フォーマットで生成します。
#  XML  -  トレース・ファイルの行は、標準 Java ロギング XML フォーマットで生成
#          されます。テキスト・エディターまたは XML エディター、あるいは以下の
#          URL にある Apache のチェーンソー・ツールを使用して表示できます。
#          (http://logging.apache.org/log4j/docs/chainsaw.html)
#
# 取り込まれるトレース情報の量は、以下のオプションで制御できます。
# -OPT traceLevel=INFO
#
# 有効値:
#
# トレース    数値
# レベル      レベル    説明
# -------  ---------  -------------------------------------------------------
# OFF         0       トレース・ファイルは生成されません。
# SEVERE      1       重大エラーのみがトレース・ファイルに出力されます。
# WARNING     2       致命的ではない例外および警告に関するメッセージが、
#                     トレース・ファイルに追加されます。
# INFO        3       通知メッセージがトレース・ファイルに追加されます。
#                     (これは、デフォルト・トレース・レベルです)
# CONFIG      4       構成関連メッセージがトレース・ファイルに追加されます。
# FINE        5       public メソッドのメソッド呼び出しをトレースします。
# FINER       6       getter と setter を除く非 public メソッドの
#                     メソッドをトレースします。
# FINEST      7       すべてのメソッド呼び出しをトレースします。トレースの
#                     出入り口に、パラメーターおよび戻り値が含まれます。
```

WebSphere eXtreme Scale クライアント インストール用の応答ファイル

```
#############################################################################
#
# IBM WebSphere eXtreme Scale Client V7.1.1 InstallShield Options File
#
# ウィザード名: インストール
# ウィザード・ソース: setup.jar
#
# ウィザードを「-options」コマンド行オプションを指定して実行した場合に、この
# ファイルを使用して、以下で指定したオプションでインストールを構成できます。
# 値の変更方法については、各設定の説明を参照してください。
# すべての値を一対の二重引用符で囲ってください。
#
# オプション・ファイルの一般的用途は、サイレント・モードでのウィザード実行です。
# オプション・ファイルの作成者は、グラフィカル・モードまたはコンソール・モードで
# ウィザードを実行せずに、ウィザード設定を指定できます。サイレント・モードの
# 実行でこのオプション・ファイルを使用するには、ウィザード実行時に以下の
# コマンド行引数を使用します。
#
#    -options "D:¥installImage¥WXS_Client¥wxssetup.response" -silent
#
# なお、完全修飾応答ファイル名を使用する必要があります。
#
#############################################################################

#############################################################################
#
# ご使用条件の受け入れ
#
# 有効値:
#  true - ご使用条件を受け入れます。製品をインストールします。
#  false - ご使用条件に同意しません。インストールされません。
#
# インストールが行われない場合は、ユーザーの一時ディレクトリーにある
# 一時ログ・ファイルにそのことが記録されます。
#
# この応答ファイルの silentInstallLicenseAcceptance プロパティーを true に
# 変更することで、このプログラムに付属の IBM プログラムのご使用条件を確認して
# 同意したことになります。このご使用条件は、
# CD_ROOT¥WXS_Cleint¥wxs.client.primary.pak¥repository¥legal.xs.client¥license.xs
# にあります。
# このご使用条件に同意しない場合には、値を変更しないでください。また、
# プログラムをダウンロード、インストール、コピー、アクセス、
# 使用しないでください。入手元へプログラムおよびライセンス証書を速やかに
# 返却して、支払額を払い戻してください。
#
-OPT silentInstallLicenseAcceptance="false"


#############################################################################
# ノンブロッキング前提条件検査
#
# ノンブロッキング前提条件検査を使用不可にする場合は、以下の行のコメントを
# 外してください。これにより、前提条件検査が失敗した場合でも、インストールを
# 進め、警告を記録するように、インストーラーに指示します。
#
#-OPT disableNonBlockingPrereqChecking="true"


#############################################################################
#
# インストール場所
#
# 製品のインストール場所。製品をインストールする有効なディレクトリーを指定
# します。ディレクトリーにスペースが含まれている場合には、以下の Windows の
# 例のように、二重引用符でディレクトリーを囲みます。なお、インストール場所に
```

```
# スペースを含められるのは、Windows オペレーティング・システムのみです。
# Windows の場合、最大パス長は 60 文字です。
#
# 以下に、root ユーザーでインストールする場合の各オペレーティング・システムの
# デフォルト・インストール場所のリストを示します。デフォルトでは、この応答
# ファイルでは、Windows のインストール場所が使用されています。別の
# オペレーティング・システムのデフォルト・インストール場所を使用する場合は、
# 以下で、該当デフォルト・インストール場所項目のコメントを外し (「#」を削除)、
# Windows オペレーティング・システムの項目をコメント化 (「#」を追加) します。
#
# インストール場所は、WebSphere eXtreme Scale をスタンドアロン・デプロイメント
# としてインストールするのか、既存の WebSphere Application Server インストール
# 済み環境に統合するのかを決定するために使用されます。
#
# 指定場所が既存の WebSphere Application Server または WebSphere Network
# Deployment インストール済み環境の場合、eXtreme Scale は既存の WebSphere
# Application Server に統合されます。指定場所が新規または空のディレクトリー
# の場合は、WebSphere eXtreme Scale は、スタンドアロン・デプロイメントとして
# インストールされます。
#
# 注: 指定インストール場所に WebSphere eXtreme Scale、WebSphere eXtended
# Deployment DataGrid、または ObjectGrid の前のインストール済み環境が
# 存在する場合には、インストールは失敗します。
#
# AIX のデフォルト・インストール場所:
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# HP-UX、Solaris、または Linux のデフォルト・インストール場所:
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Windows のデフォルト・インストール場所:
#
-OPT installLocation="C:¥Program Files¥IBM¥WebSphere¥eXtremeScale"

#
# Unix で非 root ユーザーとして、または Windows で非管理者としてインストール
# する場合は、以下のデフォルト・インストール場所をお勧めします。選択する
# インストール場所に対する書き込み許可が備わった状態にしてください。
#
# AIX のデフォルト・インストール場所:
#
#  -OPT installLocation="<ユーザーの home>/IBM/WebSphere/eXtremeScale"
#
# HP-UX、Solaris、または Linux のデフォルト・インストール場所:
#
#  -OPT installLocation="<ユーザーの home>/IBM/WebSphere/eXtremeScale"
#
# Windows のデフォルト・インストール場所:
#
# -OPT installLocation="C:¥IBM¥WebSphere¥eXtremeScale"


################################################################################
# 拡張用プロファイル・リスト
#
# 拡張する既存プロファイルを指定するか、行をコメント化してインストールで
# 検出されたすべての既存プロファイルを拡張します。
#
# 複数のプロファイルを指定するには、コンマで各プロファイル名を区切ります。
# 例えば、「AppSrv01,Dmgr01,Custom01」。リスト内でスペースは使用できません。
#
-OPT profileAugmentList=""
```

```
################################################################################
# トレース制御
#
# このオプションで、トレース出力フォーマットを制御できます。
# -OPT traceFormat=ALL
#
# フォーマットとして、「text」および「XML」を選択できます。デフォルトでは、
# 両方のフォーマットが 2 つの異なるトレース・ファイルに生成されます。
#
# 片方のフォーマットのみが必要な場合は、以下のように、traceFormat オプションで
# フォーマットを指定します。
#
# 有効値:
#
#  text -  簡単に読めるように、トレース・ファイルの行をプレーン・テキスト・
#          フォーマットで生成します。
#  XML  -  トレース・ファイルの行は、標準 Java ロギング XML フォーマットで生成
#          されます。テキスト・エディターまたは XML エディター、あるいは以下の
#          URL にある Apache のチェーンソー・ツールを使用して表示できます。
#          (http://logging.apache.org/log4j/docs/chainsaw.html)
#
# 取り込まれるトレース情報の量は、以下のオプションで制御できます。
# -OPT traceLevel=INFO
#
# 有効値:
#
# トレース    数値
# レベル      レベル    説明
# -------  ---------  ------------------------------------------------------
# OFF         0       トレース・ファイルは生成されません。
# SEVERE      1       重大エラーのみがトレース・ファイルに出力されます。
# WARNING     2       致命的ではない例外および警告に関するメッセージが、
#                     トレース・ファイルに追加されます。
# INFO        3       通知メッセージがトレース・ファイルに追加されます。
#                     (これは、デフォルト・トレース・レベルです)
# CONFIG      4       構成関連メッセージがトレース・ファイルに追加されます。
# FINE        5       public メソッドのメソッド呼び出しをトレースします。
# FINER       6       getter と setter を除く非 public メソッドの
#                     メソッドをトレースします。
# FINEST      7       すべてのメソッド呼び出しをトレースします。トレースの
#                     出入り口に、パラメーターおよび戻り値が含まれます。
```

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、178 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 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ー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、コンテナーの開始、および TCP ポートの構成を実行します。詳細については、 276 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.shlbat 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、コンテナーの開始、および TCP ポートの構成を実行します。詳しくは、 276 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、 178 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 ディレクトリーにあります。

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に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。

スタンドアロン WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストール

スタンドアロン WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントを、 WebSphere Application Server も WebSphere Application Server Network Deployment も含まれていない環境にインストールできます。

始める前に

- ターゲット・インストール・ディレクトリーが空であるか、存在していないことを確認します。

  重要: バージョン 7.1.1 のインストール先に指定したディレクトリーに、以前のバージョンの WebSphere eXtreme Scale または ObjectGrid コンポーネントが存在していると、この製品はインストールされません。例えば、既存の *wxs_install_root*/ObjectGrid フォルダーが存在する場合があります。他のインストール・ディレクトリーを選択するか、あるいは、インストールを取り消すことができます。次に、以前のインストールをアンインストールしてから、再度ウィザードを実行してください。
- IBM Runtime Environment は、 *wxs_install_root*/java フォルダーにスタンドアロン・インストールの一部としてインストールされます。
- クライアントのみをインストールする場合: サポート・サイトから、該当するプラットフォームの WebSphere eXtreme Scale クライアントをダウンロードします。

このタスクについて

製品をスタンドアロンとしてインストールする場合、WebSphere eXtreme Scale クライアントとサーバーを一緒にインストールします。スタンドアロン・モードの WebSphere eXtreme Scale クライアントのインストールでは、データ・グリッド内のデータにアクセスするクライアントをインストールします。 したがって、サーバーとクライアントのプロセスは、必要なすべてのリソースにローカル・アクセスします。また、スクリプトと Java アーカイブ (JAR) ファイルを使用して、WebSphere eXtreme Scale を既存の Java Platform, Standard Edition (J2SE) アプリケーションに組み込むこともできます。

重要: 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。非 root プロファイルを使用するには、ObjectGrid ディレクトリーの所有者を非 root プロファイルに変更する必要があります。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。

手順

1. ウィザードを使用して、DVD から、サーバーとクライアントの両方をインストールします。
   - 以下のスクリプトを実行して、WebSphere eXtreme Scale フルインストール用のウィザードを開始します。
     - Linux UNIX *dvd_root*/install
     - Windows *dvd_root*¥install.bat

- 以下のスクリプトを実行して、WebSphere eXtreme Scale クライアントのインストール用のウィザードを開始します。インストール・ファイルは、サポート・サイトからダウンロードする zip ファイルに入っています。
  - Linux UNIX `root/WXS_Client/install`
  - Windows `root¥WXS_Client¥install.bat`

  重要: uniform naming conventions (UNC) を使用してインストール・コマンドのファイル・パスを識別する場合、コマンドを実行した後、インストールする予定だった項目のいくつかがインストールされていないことがあります。問題を避けるために、ファイル・パスをネットワーク・ドライブにマップします。**install** コマンドをマップされたドライブに対して実行します。マップされたネットワーク・ドライブを使用すると、確実にすべての項目がインストールされます。
2. ウィザードのプロンプトに従って、「**終了**」をクリックします。

   制約事項: オプション・フィーチャー・パネルに、インストールを選択できるフィーチャーがリストされます。ただし、製品のインストール後に、その製品環境にフィーチャーを順次追加することはできません。初期の製品インストール時にフィーチャーのインストールを選択しなかった場合、そのフィーチャーを追加するには、製品をアンインストールしてから再インストールする必要があります。

タスクの結果

Windows WebSphere eXtreme Scale クライアントを Windows にインストールした場合には、インストールの結果で以下のテキストが表示されることがあります。

```
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.
```

`iscdeploy.sh` ファイルで障害が発生しても、エラーを無視して構いません。このエラーでは、問題は生じません。

次のタスク

- インストールを検査します。詳しくは、224 ページの『インストールの検査』を参照してください。
- WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント インストール済み環境の構成を開始します。詳しくは、226 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。

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	ユーティリティー	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 インスツルメンテーション・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。
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

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	ユーティリティー	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 インスツルメンテーション・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 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

WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント のサイレント・モードでのインストール

ニーズに合わせて専用に構成する完全修飾応答ファイルを使用するか、パラメーターをコマンド行に渡して、 WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントをサイレント・インストールします。

始める前に

- 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。
- ターゲット・インストール・ディレクトリーが空であるか、存在していないことを確認します。

  重要: バージョン 7.1.1 のインストール先に指定したディレクトリーに、以前のバージョンの WebSphere eXtreme Scale または ObjectGrid コンポーネントが存在していると、この製品はインストールされません。例えば、既存の *wxs_install_root*/ObjectGrid フォルダーが存在する場合があります。他のインストール・ディレクトリーを選択するか、あるいは、インストールを取り消すことができます。次に、以前のインストールをアンインストールしてから、再度ウィザードを実行してください。

このタスクについて

サイレント・インストールは、グラフィカル・ユーザー・インターフェース (GUI) バージョンが使用するのと同じインストール・プログラムを使用します。ただし、ウィザード・インターフェースを表示する代わりに、サイレント・インストールは、カスタマイズされたファイルから、あるいはコマンド行にパスされたパラメーターからすべての応答を読み取ります。各オプションの説明が含まれている、 192 ページの『wxssetup.response.txt ファイル』の例を参照してください。

手順

1. オプション: 応答ファイルを使用した WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントのインストールを選択する場合には、まず wxssetup.response.txt ファイルをカスタマイズします。

   要確認: 完全修飾応答ファイル名を指定してください。相対パスを指定すると、エラーが発生したことをまったく示さずにインストールが失敗します。

   a. カスタマイズする応答ファイルのコピーを作成します。

      WebSphere eXtreme Scale フルインストールの場合は、応答ファイルを製品 DVD からディスク・ドライブにコピーします。

      WebSphere eXtreme Scale クライアント の場合は、WebSphere eXtreme Scale クライアント zip ファイルをハード・ディスクに解凍して、応答ファイルを見つけます。

   b. 任意のテキスト・エディターで応答ファイルを開き、編集します。 前の応答ファイルの例には、各パラメーターの指定方法の詳細が示されています。以下のパラメーターを指定する必要があります。

      - ご使用条件

- インストール・ディレクトリー

ヒント: WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアントを WebSphere Application Server 環境にインストールする場合、インストーラーはインストール・ディレクトリーを使用して、既存の WebSphere Application Server インスタンスがインストールされている場所を判別します。複数の WebSphere Application Server インスタンスが含まれるノードにインストールする場合、そのロケーションを明確に定義してください。

c. 次のスクリプトを実行して、インストールを開始します。

WebSphere eXtreme Scale フルインストールの場合:

```
./install.sh|bat -options C:/drive_path/response_file.txt -silent
```

WebSphere eXtreme Scale クライアントのインストールの場合:

```
./WXS_Client/install.sh|bat -options C:/drive_path/response_file.txt -silent
```

GUI インストールを実行する場合にも応答ファイルを使用できます。 GUI インストールで応答ファイルを使用して、サイレント・インストールでは分からなかった問題をデバッグできます。 GUI インストールまたはサイレント・インストールで wxssetup.response ファイルを指定する際には、完全修飾パスを使用する必要があります。以下のスクリプトを実行し、応答ファイルを使用して GUI インストールを実行します。

- Linux UNIX `<install_home>/install.sh -options <full_install_path_required>/wxssetup.response`
- Windows `<install_home>¥install.exe -options c:¥<full_install_path_required>¥wxssetup.response`

2. オプション: 特定のパラメーターをコマンド行に渡すことによって eXtreme Scale をインストールする場合は、次のスクリプトを実行してインストールを開始します。

WebSphere eXtreme Scale フルインストールの場合:

```
./install.sh|bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=install_location
```

WebSphere eXtreme Scale クライアントのインストールの場合:

```
./WXS_Client/install.sh|bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=install_location
```

サイレント・インストール用の応答ファイル

製品のインストールをカスタマイズし、構成するためのパラメーターをコマンド行に指定します。

注: 完全修飾応答ファイル名を指定してください。相対パスを指定すると、エラーが発生したことをまったく示さずにインストールが失敗します。

パラメーター

コマンド行で、または製品のオプション・ファイルのインストール中に、次のパラメーターを渡すことができます。

-silent
グラフィカル・ユーザー・インターフェース (GUI) を抑止します。**-options** パ

ラメーターを指定して、インストーラーに、カスタマイズしたオプション・ファイルに従ってインストールを実行するように指示します。 **-options** パラメーターを指定しないと、代わりにデフォルト値が使用されます。

使用例

```
./install.sh|bat -silent -options options_file.txt
```

-options *path_name/file_name*
サイレント・インストールを実行するためにインストーラーが使用するオプション・ファイルを指定します。コマンド行のプロパティーが優先されます。

使用例

```
./install.sh|bat -options c:/path_name/options_file.txt
```

-log #!file_name @*event_type*
次のイベント・タイプを記録するインストール・ログ・ファイルを生成します。

- err
- wrn
- msg1
- msg2
- dbg
- ALL

使用例

```
./install.sh|bat -log # !c:/temp/logfiles.txt @ALL
```

-is:log *path_name/file_name*
GUI の始動中に、インストーラーの Java 仮想マシン (JVM) 検索を含むログ・ファイルを作成します。ログ・ファイルは、指定しないと作成されません。

使用例

```
./install.sh|bat -is:log c:/logs/javalog.txt
```

-is:javaconsole
インストール・プロセス中にコンソール・ウィンドウを表示します。

使用例

```
./install.sh|bat -is:javaconsole
```

-is:silent
インストーラーの開始時に表示される Java 初期化ウィンドウを抑止します。

使用例

```
./install.sh|bat -is:silent
```

-is:tempdir *path_name*
インストーラーがインストール時に使用する一時ディレクトリーを指定します。

使用例

```
./install.sh|bat -is:tempdir c:/temp
```

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 サー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 サーブレット・コンテナーのバージョン 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、384 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・グリッドで使用する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 コンテナー・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。

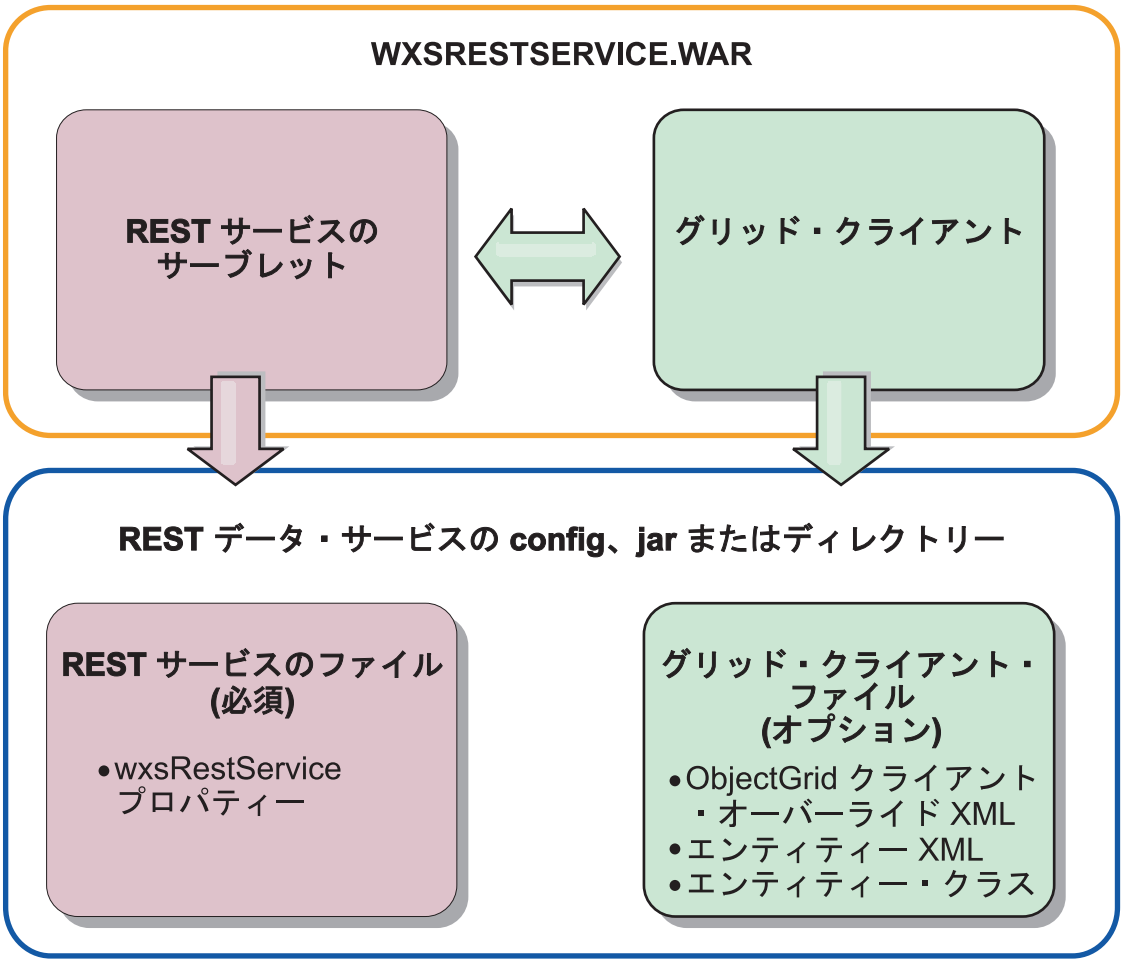


図 26. WebSphere eXtreme Scale REST データ・サービスのファイル

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/*エンティティー記述子 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ー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ー記述子 XML ファイルが必要です。エンティティー記述子 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。
- **エンティティー・クラス**。アノテーションが付けられたエンティティー・クラスまたはエンティティー記述子 XML ファイルを使用して、エンティティー・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ー・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・エンティティー 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ー・クラスが必要になります。

  エンティティーがサーバー上で 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
  ```

  プロパティー・ファイルには、以下が含まれます。

  ```
  catalogServiceEndPoints=localhost:2809
  objectGridNames=NorthwindGrid
  ```

  単一エンティティー、オーバーライド XML ファイル、およびエンティティー・クラスの例:

  ```
  restserviceconfig.jar:
  wxsRestService.properties
  ```

  プロパティー・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
  ```

  エンティティー・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"/>
  ```

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 コンテナーをサポートします。次の手順を使用して、Eclipse Gemini コンテナー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
   ```

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中のものと異なります。

    ```
    java -jar plugins\org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
    ```

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。

始める前に

このタスクは、次の製品のインストールが完了している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、eXtreme Scale バージョン 7.1.1 のサーバー・バンドルは com.ibm.websphere.xs.server_7.1.1 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、eXtreme Scale バージョン 7.1.1 のクライアント・バンドルは com.ibm.websphere.xs.client_7.1.1 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。

手順

OSGi コンソールを使用して、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、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。

   ```
   25      ACTIVE      com.ibm.websphere.xs.server_7.1.1
   ```

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、以下のようにします。

6. 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
7. 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ブランク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。
8. 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
   ```
9. 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。

   ```
   25      ACTIVE      com.ibm.websphere.xs.server_7.1.1
   ```

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。

インストールの検査

インストール・ウィザードが完了したら、インストール済み環境のいくつかの特徴を確認することでインストールを検査できます。

手順

- **スタンドアロン・インストールまたは WebSphere Application Server と統合されたインストールの場合:**

  次のいずれかの方式を使用して、インストールが正常に完了したか検査します。

  – WebSphere eXtreme Scale のバージョン情報コマンドを実行します。

    ```
    was_root/lib/> java -jar wsobjectgrid.jar version
    ```

    製品名、バージョン番号、およびビルド番号が結果に表示されます。

  – 適切なバージョン番号のプロパティー・ファイルを確認します。

- シグニチャー・ファイル: シグニチャー・ファイルは *was_root*/properties/version ディレクトリーにあります。フィックスパックをインストールした場合、fxtg ファイルも追加で含まれています。以下に、シグニチャー・ファイル名の例をいくつか示します。

  ```
  WebSphere_eXtreme_Scale.7.1.1..swtag
  WebSphere_eXtreme_Scale.7.1.0.2.fxtag
  WebSphere_eXtreme_Scale.7.1.0.3.fxtag
  ```

- WebSphere eXtreme Scale 製品ファイル:

  製品ファイルは *was_root*/properties/version ディレクトリーにあります。 WXS.product ファイルを探してください。このファイルの内容の例を以下に示します。

  ```
  <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE product SYSTEM "product.dtd">
    <product name="IBM WebSphere eXtreme Scale">
         <id>WXS</id>
         <version>7.1.1.0</version>
         <build-info
     date="8/5/11"
     level="a1132.68720"/>
    </product>
  ```

– ランタイム・ファイルがインストールされていることを確認します。インストールのタイプに応じたランタイム・ファイルのリストは、次のトピックに記載されています。

  - 211 ページの『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 ページの『WebSphere Application Server と統合された WebSphere eXtreme Scale 用のランタイム・ファイル』

- **WebSphere Application Server と統合されたインストールの場合、インストールが正常に完了したか確認できる次の追加の方法があります。**

  – WebSphere Application Server のバージョン情報コマンドを実行します。

  ```
  was_root/bin/> versionInfo.sh|.bat
  ```

  出力には、インストール済み製品のリストが表示され、インストール・ディレクトリー、インストールされた製品、バージョン、ビルド・レベル、ビルド日付などの情報が含まれます。

  ヒント: さらに詳しい情報を表示するには、**-maintenancePackages** パラメーターを追加します。

  ```
  was_root/bin/> versionInfo.sh|.bat -maintenancePackages
  ```

  – WebSphere Application Server 管理コンソールの「ようこそ」パネルを確認します。 http://localhost:9060/ibm/console にアクセスします。コンソールにログインします。WebSphere eXtreme Scale のバージョンが「ようこそ」パネルに表示されます。

  – ファースト・ステップ・コンソールを使用して、WebSphere Application Server インストールを WebSphere eXtreme Scale で拡張します。

  ```
  was_root/firststeps/WXS> firststeps.sh|.bat
  ```

  詳しくは、 199 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。

次のタスク

推測どおり、インストールが完了していなかった場合、インストールの問題をトラブルシューティングする必要があります。詳しくは、『インストールのトラブルシューティング』を参照してください。

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: 229 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。プロパティーや 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: 241 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ー・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: 427 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: 477 ページの『第 8 章 モニター』

インストールのトラブルシューティング

インストール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。

WebSphere eXtreme Scale のアンインストール

ご使用の環境から WebSphere eXtreme Scale を削除するには、ウィザードを使用するか、または、製品をサイレント・アンインストールすることができます。

始める前に

重要: アンインストーラーは、すべてのバイナリー・ファイルと、フィックスパックやインテリム・フィックスなどのすべての保守を同時に削除します。

手順

1. eXtreme Scale を実行中のすべてのプロセスを停止します。

   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アン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、アンインストールが不確定状態になります。

   - スタンドアロン eXtreme Scale をインストールしている場合は、スタンドアロン・サーバーの停止を参照して、プロセスを停止します。
   - eXtreme Scale を WebSphere Application Server の既存のインストール環境にインストールしている場合は、 WebSphere Application Server プロセスの停止の詳細について、 コマンド行ユーティリティー を参照してください。
   - Web コンソールを実行している場合は、`stopConsoleServer` コマンドを使用して Web コンソール・サーバーを停止します。`stopConsoleServer` スクリプ

トは、*wxs_install_root*/ObjectGrid/bin ディレクトリーにあります。アンインストールの実行前にこのサーバーを停止しなければ、プロセスは、アンインストール処理の中で自動的に停止します。

2. 製品をアンインストールします。 GUI アンインストールまたはサイレント・アンインストールを実行できます。

   注: サイレント・アンインストール/インストールまたは GUI アンインストール/インストールで応答ファイル wxssetup.response を指定する際には、常に完全修飾パスを指定する必要があります。 GUI アンインストールでは応答ファイルはオプションです。

   - **GUI を使用してアンインストールを実行するには、以下のようにします。**
     - Linux UNIX <*install_home*>/uninstall_wxs/uninstall
     - Windows <*install_home*>¥uninstall_wxs¥uninstall.exe

     wxssetup.response ファイルを使用して GUI アンインストールを実行する場合には、以下のコマンドのいずれかを使用します。

     - Linux UNIX

       ```
       <install_home>/uninstall_wxs/uninstall -options
       <full_install_path_required>/wxssetup.response
       ```

     - Windows

       ```
       <install_home>¥uninstall_wxs¥uninstall.exe -options
       <full_install_path_required>¥wxssetup.response
       ```

   - **応答ファイル wxssetup.response スクリプトを使用してサイレント・アンインストールを実行する場合には、以下のようにします。**
     - Linux UNIX

       ```
       <install_home>/uninstall_wxs/uninstall -options
       <full_install_path_required>/wxssetup.response -silent
       ```

     - Windows

       ```
       <install_home>¥uninstall_wxs¥uninstall.exe -options
       <full_install_path_required>¥wxssetup.response -silent
       ```

タスクの結果

これで、ご使用の環境から eXtreme Scale の削除ができました。

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 7.1.1 にマイグレーションできます。また、バージョン 7.1.1 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 233 ページの『Update Installer を使用して保守パッケージをインストールする』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 232 ページの『WebSphere eXtreme Scale バージョン 7.1.1 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・ファイルを入手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

サービスを中断せずにアップグレードするには、最初にカタログ・サーバーをアップグレードします。次に、コンテナー・サーバーおよびクライアントをアップグレードします。

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。 どのコンテナー・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
```

7.1.1+

```
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。 **stopOgserver** 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。**stopOgServer** 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ことが必要です。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。

stopOgServer 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。

```
stopOgServer <server_name>[,<server_name>]
xsadmin –teardown <server_name>[,<server_name>]
```

7.1.1+

```
xscmd –c teardown -sl <server_name>[,<server_name>]
```

上記の例では、**stopOgServer** 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、 442 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。 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 7.0.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: 232 ページの『WebSphere eXtreme Scale バージョン 7.1.1 へのマイグレーション』
- バージョン 7.1.0.x インストール済み環境からアップグレードする場合: 233 ページの『Update Installer を使用して保守パッケージをインストールする』

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、427 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、443 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ー・サーバーに対して、次のステップを繰り返して、コンテナー・サーバーをアップグレードします。 コンテナー・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ー・サーバーを停止します。 **stopOgserver** コマンドまたは **teardown** コマンドを使用して、コンテナー・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
```

7.1.1+

```
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. コンテナー・サーバーに更新をインストールします。 コンテナー・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 7.0.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: 『WebSphere eXtreme Scale バージョン 7.1.1 へのマイグレーション』
   - バージョン 7.1.0.x インストール済み環境からアップグレードする場合: 233 ページの『Update Installer を使用して保守パッケージをインストールする』

c. コンテナー・サーバーを再始動します。

d. 構成の残りのコンテナー・サーバーをアップグレードします。

3. 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。 7.1.1+

```
xsadmin –ch host –p 1099 –establishLink dname fdHostA:2809,fdHostB:2809

xscmd –c establishLink -cep host:2809 -fd dname -fe fdHostA:2809,fdHostB:2809
```

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 7.1.1 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、 54 ページの『ハードウェアおよびソフトウェアの要件』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 7.1.1 にアップグレードできます。

手順

1. 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 eXtreme Scale 環境で実行しているすべてのプロセスを停止するには、 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ー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。
4. eXtreme Scale バージョン 7.1.1 をインストールします。 詳しくは、 178 ページの『インストール・ウィザードによる WebSphere eXtreme Scale のインストール』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。 詳細については、 427 ページの『第 7 章 管理』を参照してください。

Update Installer を使用して保守パッケージをインストールする

IBM Update Installer を使用して、WebSphere eXtreme Scale または WebSphere eXtreme Scale クライアント 環境を暫定修正、フィックスパック、リフレッシュ・パックなど、さまざまなタイプの保守で更新します。

このタスクについて

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。

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、 439 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. Update Installer の最新バージョンをダウンロードします。 詳しくは、推奨フィックスを参照してください。
3. Update Installer をインストールします。 詳しくは、WebSphere Application Server インフォメーション・センターの Update Installer for WebSphere Software のインストールを参照してください。

4. インストールしようとする保守パッケージを *updi_root*/maintenance ディレクトリーにダウンロードします。 詳しくは、 サポート・サイトを参照してください。
5. Update Installer を使用して、暫定修正、修正パッケージ、またはリフレッシュ・パックをインストールします。 保守パッケージのインストールは、グラフィカル・ユーザー・インターフェース (GUI) を実行するか、Update Installer をサイレント・モードで実行することで行うことができます。

   updi_root ディレクトリーから次のコマンドを実行して、GUI を開始します。

   - Linux UNIX `update.sh`
   - Windows `update.bat`

   updi_root ディレクトリーから次のコマンドを実行して、Update Installer をサイレント・モードで実行します。

   - Linux UNIX `./update.sh -silent -options responsefile/file_name`
   - Windows `update.bat -silent -options responsefile¥file_name`

   インストール・プロセスが失敗した場合、 *updi_root*/logs/update/tmp ディレクトリーにある一時ログ・ファイルを参照してください。 Update Installer は、インストール・ログ・ファイルが入れられる *install_root*/logs/update/*maintenance_package*.install ディレクトリーを作成します。

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
-bp	• **-cep** *hostname:listener_port* • **--catalogEndpoint** *hostname:listener_port*	n/a
-ch	• **-cep** *hostname:listener_port* • **--catalogEndpoint** *hostname:listener_port*	n/a
-clear	**-c clearGrid**	**-g, -ms, -v, -m, (-cep)**
-containers	• **-c listCoreGroups** • **-c listCoreGroupMembers -cg** *core_group*	**-e, -I, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms**

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**
-continuous	• **-cnt** • **--continuous**	n/a
-coregroups	• **-c listCoreGroups** • **-c listCoreGroupMembers -cg** *core_group*	n/a
-dismissLink <*catalog_service_domain*>	**-c dismissLink**	• **-fd** <*foreignCatalogServiceDomain*> • **--foreignCatalogServiceDomain** <*foreignCatalogServiceDomain*>
-dmgr	n/a - この引数は、**xscmd** で自動的に判別される	n/a
-empties	新規コマンドに固有の引数	n/a
-establishLink <*foreign_domain_name*> <*host1:port1,host2:port2...*>	**-c establishLink**	• **-fd** <*foreignCatalogServiceDomain*> **-fe** <*host1:port1,host2:port2...*> • **--foreignCatalogServiceDomain** <*foreignCatalogServiceDomain*> **-foreignEndPoints** <*host1:port1,host2:port2...*>
-fc	• **-ct** • **--container**	n/a
-fh	• **-hf** • **--hostFilter**	n/a
-fm	• **-m** • **--map**	n/a
-fnp	• **-snp** • **--serversWithNoPrimaries**	n/a
-fp	• **-p** • **--partitionId**	n/a
-fs	• **-s** • **--server**	n/a
-fst	• **-st** <*shard_type*> • **--shardType** <*shard_type*> 断片値: P=primary A=asyncReplica S=syncReplica	n/a
-fz	• **-z** • **--zone**	n/a
-force	新規コマンドに固有の引数	

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
-g	• **-g** • **--objectGrid**	n/a
-getstatsspec	**-c getStatsSpec**	n/a
-getTraceSpec	**-c getTraceSpec**	n/a
-h	特定のコマンド名を付けても付けなくても help を実行できる • **-h** • **--help** • **-h** <*command_name*> • **--help** <*command_name*>	n/a
-hosts	**-c listHosts**	**-g, -ms, -st, -c, -s, -hf, -z**
-jmxUrl	• **-cep** *hostname*:*listener_port* • **--catalogEndpoint** *hostname*:*listener_port*	n/a
-l	**-c listObjectGridNames**	n/a
-m	• **-ms** • **--mapSet**	n/a
-mapsizes	**-c showMapSizes**	**-g, -ms, -cnt, -i, [-ct, -z, -s, -hf, sht [P,A,S], -p]**
-mbeanservers	**-c listAllJMXAddresses**	n/a
-overridequorum	**-c overrideQuorum**	n/a
-password	• **-pwd** • **--password**	n/a
-p	• **-cep** *hostname*:*listener_port* • **--catalogEndpoint** *hostname*:*listener_port*	n/a
-placementStatus	**-c placementServiceStatus**	**-g, -ms**
-primaries	**-c showPlacement -sf P**	**-e, -I, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms**
-profile	現行セキュリティー設定をセキュリティー・プロファイルとして保存する場合: • **-ssp** *profile_name* • **--saveSecProfile** *profile_name* 指定されたセキュリティー・プロファイルを使用する場合: • **-sp** *profile_name* • **--securityProfile** *profile_name*	
-quorumstatus	**-c showQuorumStatus**	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
-releaseShard <*container_server_name*> <*objectgrid_name*> <*map_set_name*> <*partition_name*>	**-c releaseShard**	**-c, -g, -ms, -p**
-reserved	• **-sf** *[R,U]* • **--shardFilter** *[R,U]* R=reserved, U=unassigned	n/a
-reserveShard <*container_server_name*> <*objectgrid_name*> <*map_set_name*> <*partition_name*>	**-c reserveShard**	**-c, -g, -ms, -p**
-resumeBalancing <*objectgrid_name*> <*map_set_name*>	**-c resumeBalancing**	**-g, -ms**
-revisions	**-c revisions**	**-s,-p,-g,-m**
-routetable	**-c routetable**	**-z, -hf,-p,-g,-ms**
-settracespec <*trace_string*>	**-c setTraceSpec**	**-spec** <*trace_string*>
-swapShardWithPrimary <*container_server_name*> <*objectgrid_name*> <*map_set_name*> <*partition_name*>	**-c swapShardWithPrimary**	**-c -g, -ms, -p**
-setstatsspec <*stats_spec*>	**-c setStatsSpec**	**-spec** <*stats_spec*>
-suspendBalancing <*objectgrid_name*> <*map_set_name*>	**-c suspendBalancing**	**-g, -ms**
-ssl	• **-ssl** • **--enableSSL**	n/a
-teardown	**-c teardown**	**-f, , -st, -snp, -c, -s, -p, -hf, -z, -g, -ms, -m**
-triggerPlacement	**-c triggerPlacement**	**-g, -ms**
-trustPass	• **-tsp** • **--trustStorePassword**	n/a
-trustPath	• **-ts** • **--trustStore**	n/a
-trustType	• **-tst** • **--trustStoreType**	n/a
-unassigned	**-c showPlacement -sf U**	**-e, -I, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms**
-username	• **-user** • **--username**	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
-v	• -v • --verbose	n/a
-xml	-c showPlacement	n/a

推奨されないプロパティーおよび API

次にリストされたプロパティーおよび API は、バージョン 7.1.1 リリースで使用を推奨されないものです。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。

7.1.1+ バージョン 7.1.1 での推奨されない項目

表 10. 推奨されないプロパティーおよび API

非推奨機能	推奨されるマイグレーション・アクション
com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener クラス 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。	7.1.1+ 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。
com.ibm.websphere.objectgrid.plugins.OptimisticCallback クラス 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。	7.1.1+ 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。
com.ibm.websphere.objectgrid.plugins.NoVersioningOptimisticCallback プラグイン 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が、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。	7.1.1+ 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 以下に設定してペシミスティック・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。
com.ibm.websphere.objectgrid.plugins.ObjectTransformer クラス 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。	7.1.1+ 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。
com.ibm.websphere.objectgrid.BackingMap.setMapEventListeners メソッド 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。	7.1.1+ 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。
com.ibm.websphere.objectgrid.ObjectGrid.setEventListeners メソッド 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。	7.1.1+ 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。

7.1.1+

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 11. 推奨されないプロパティーおよび 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
xsadmin xsadmin ユーティリティーは、デプロイメントのカスタム・ユーティリティー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。	7.1.1+ `xscmd` ユーティリティーを使用して、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 12. 推奨されないプロパティーおよび API

非推奨機能	推奨されるマイグレーション・アクション
catalog.services.cluster セルおよびサーバー・プロパティー: このカスタム・プロパティーは、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・プロパティーは、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・プロパティーを使用する場合と同じ構成が作成されます。詳しくは、を参照してください。
CoreGroupServicesMBean 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。
ServerMBean.updateTraceSpec() 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。
CoreGroupServicesMBean 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。
ServiceUnavailableException 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。
区画化機能 (WPF): 区画化機能は、プログラミング API のセットで、これにより Java EE アプリケーションは非対称クラスタリングをサポートできるようになります。	WPF の機能は、また、WebSphere eXtreme Scale でも実現することができます。
StreamQuery: 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

表 12. 推奨されないプロパティーおよび 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
静的グリッド構成: クラスター・デプロイメント XML ファイルを使用する静的なクラスター・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。
非推奨システム・プロパティー: サーバーおよびクライアントのプロパティー・ファイルを指定するシステム・プロパティーは推奨されていません。	これらの引数は、まだ使用できますが、ご使用のシステム・プロパティーを新しい値に変更してください。 **-Dcom.ibm.websphere.objectgrid.CatalogServerProperties** このプロパティーは、WebSphere eXtreme Scale バージョン 7.0 で使用すべきではありません。**-Dobjectgrid.server.props** プロパティーを使用してください。 **-Dcom.ibm.websphere.objectgrid.ClientProperties** このプロパティーは、WebSphere eXtreme Scale バージョン 7.0 で使用すべきではありません。**-Dobjectgrid.client.props** プロパティーを使用してください。 **-Dobjectgrid.security.server.prop** このプロパティーは、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。**-Dobjectgrid.server.prop** プロパティーを使用してください。 **-serverSecurityFile** 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。**-serverProps** 引数を使用してください。

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。

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ー・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・プロパティー・ファイル
サーバー・プロパティー・ファイルを使用して、トレース、ロギング、セキュリティー、ポートなど、カタログ・サーバーとコンテナー・サーバーの設定を定義します。サーバー・プロパティー・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・プロパティーを使用して定義できます。

クライアント・プロパティー・ファイル
クライアント・プロパティー・ファイルを使用して、ポートやセキュリティー設定など、クライアントのプロパティーを設定します。使用するクライアント・プロパティー・ファイルを指定するには、システム・プロパティーを使用するか、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ー・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。

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>
```

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、251 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、319 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。246 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) コンテナー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 commited.
            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、レプリカ生成関連のプロパティー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) の両方のコンテナーで 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 プロパティーはオプションです。

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ーがあります。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ーはオプションです。

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 メッセージが受信されたときに、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 と同じプロパティーを持つ拡張 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、コンテナー・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 コンテナー・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・コンテナー・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ー・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、始動時にコンテナー・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ー・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レプリカは 1 つだけです。同期レプリカの数は、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 エレメント内の区画には、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 つのコンテナー・サーバーが使用可能になるまで配置を据え置きます。コンテナー・サーバーが指定数に達すると、numInitialContainers 属性は無視されます。

7.1.1+ **placementDeferralInterval** プロパティーと **xscmd -c suspendBalancing** コマンドを使用して、コンテナー・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ーが改善されます。

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、コンテナー・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。コンテナー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ー・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・コンテナー・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、コンテナー・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。コンテナー・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ー・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ー・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ー・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、複数のコンテナー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。コンテナーは、単一の ObjectGrid の複数の断片をホストできます。

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、断片コンテナーである 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ー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ー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・グループを活用して、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

コンテナーのゾーン名は、447 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、包含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 つのマップを持つ「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になります。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="BldA" />
      <zone name="BldB" />
    </zoneRule>
    <zoneRule name="aysnc" exclusivePlacement="true">
      <zone name="BldA" />
      <zone name="BldB" />
    </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 primaySync 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になります。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、257 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、コンテナーごとの区画の配置が必要です。この配置ストラテジーはセッション・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ストラテジーは、`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。

コンテナーごとの配置により、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。

コンテナーごとの配置により、新しいプライマリー断片は、開始された各コンテナーに配置されます。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ーで、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。各コンテナーを適切なゾーン名で開始します。コンテナーを **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 のコンテナー・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
```

ご使用のコンテナーが 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、 257 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ー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ー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。

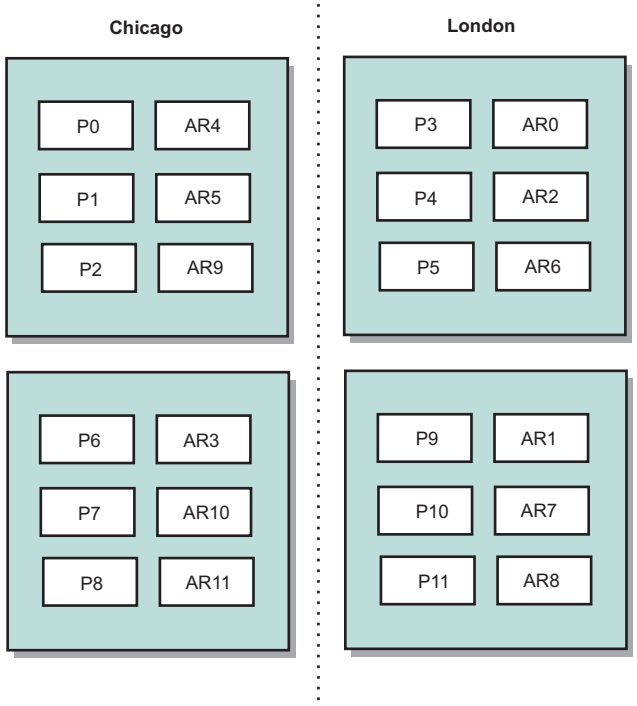


図 29. ゾーン内のプライマリーとレプリカ

クライアントの優先ゾーンを設定します。クライアント・プロパティー・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** プロパティーを組み込みます。このプロパティー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ー・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
```

このプロパティーにより、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

コンテナー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。

コンテナー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** プロパティー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、コンテナーの損失があると、レプリカ断片がプライマリー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 コンテナーが損失したとします。このコンテナーには区画 0、1、および 2 のプライマリー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ー断片は 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ーに経路指定します。

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ー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。

コンテナー・サーバーのゾーンの定義

ゾーンは、コンテナー・サーバーの集合です。コンテナー・サーバーは、1 つのゾーンにのみ所属できます。コンテナー・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

コンテナー・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、コンテナー・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。コンテナー・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ー・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・コンテナー・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ー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、433 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ー・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、444 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ー・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ー・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。コンテナー・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します。

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、 270 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ー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、プライマリー断片を 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ー断片は、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>
```

```
    <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 ゾーンでコンテナー・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 つ目のゾーン 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

失われない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、262 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、コンテナー、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、コンテナー・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ー・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ー・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ー・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、コンテナー・サーバーの検出とモニターをします。複数のカタログ・サーバーがまとまってカタログ・サービスを構成します。コンテナー・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。

このタスクについて

カタログ・サーバーとコンテナー・サーバーは、WebSphere Application Server プロセス内で開始するか、スタンドアロン Java SE プロセスとして開始するか、または Java SE アプリケーションにサーバーを組み込むことで開始できます。カタログ・サーバーとコンテナー・サーバーの構成方法は、使用するトポロジーによって異なります。

カタログ・サーバー

- **スタンドアロン・カタログ・サーバー**:

  スタンドアロン・カタログ・サーバーは、サーバー・プロパティー・ファイルを使用して構成します。カタログ・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 内で開始するカタログ・サーバー**:

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・プロパティー・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。

コンテナー・サーバー

- **スタンドアロン・コンテナー・サーバー**:

  スタンドアロン・コンテナー・サーバーは、サーバー・プロパティー・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。コンテナー・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ー・サーバー**:

  WebSphere Application Server 内のコンテナー・サーバーは、サーバー・プロパティー・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。コンテナー・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。コンテナー・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。

次のトピックを使用して、カタログ・サーバーとコンテナー・サーバーを構成してください。

ベスト・プラクティス: カタログ・サービス・ドメインを使用したカタログ・サービスのクラスタリング

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。

カタログ・サーバーの数

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。

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。

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。

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、**startOgServer** 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 13. 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ーは、WebSphere 管理コンソールで、コア・グループに対してカスタム・プロパティー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・プロパティー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、以下のプロパティー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・プロパティー (秒単位) または IBM_CS_FD_PERIOD_MILLIS カスタム・プロパティー (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・プロパティーを使用して指定します。

IBM_CS_FD_PERIOD_SEC プロパティーのデフォルト値は 20 で、IBM_CS_FD_CONSECUTIVE_MISSED プロパティーのデフォルト値は 10 です。IBM_CS_FD_PERIOD_MILLIS プロパティーを指定すると、設定されている IBM_CS_FD_PERIOD_SEC カスタム・プロパティーがオーバーライドされます。これらのプロパティーの値は、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 でのコア・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。

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ー・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、コンテナー・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、コンテナー・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 内で実行できます。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、 199 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・プロパティー・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・プロパティー・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、 277 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。

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、 443 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ー・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、 178 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ができないため、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ができません。ただし、WebSphere eXtreme Scale コンテナー・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ができます。

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 コンテナー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・プロパティー・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、 427 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 コンテナー・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。

- 『createXSDomain』
- 282 ページの『deleteXSDomain』
- 282 ページの『getDefaultXSDomain』
- 283 ページの『listXSDomains』
- 283 ページの『modifyXSDomain』
- 289 ページの『testXSDomainConnection』
- 289 ページの『testXSServerConnection』

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 14. createXSDomain コマンド引数

引数	説明
-name (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。
-default	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 `true` です。(ブール値: `true` または `false` に設定)
-properties	カタログ・サービス・ドメインのカスタム・プロパティーを指定します。

表 15. defineDomainServers ステップ引数

引数	説明
name_of_endpoint	カタログ・サービス・エンドポイントの名前を指定します。 • **既存のアプリケーション・サーバーの場合**: エンドポイントの名前は、*cell_name¥node_name¥server_name* の形式でなければなりません。 • **リモート・サーバーの場合**: 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。
custom_properties	カタログ・サービス・ドメイン・エンドポイントのカスタム・プロパティーを指定します。カスタム・プロパティーがない場合、この引数には一組の二重引用符 (`""`) を使用します。

表 15. *defineDomainServers* ステップ引数 (続き)

引数	説明
endpoint_ports	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、*<client_port>,<listener_port>* の順序で指定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。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。 **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ー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**: コンテナー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るためオプションです。

表 16. *configureClientSecurity* ステップ引数

引数	説明
-securityEnabled	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。(ブール値: `true` または `false` に設定)

表 16. *configureClientSecurity* ステップ引数 (続き)

引数	説明
-credentialAuthentication (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。 **常になし** クライアント証明書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** (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。
-authenticationRetryCount (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。
-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 実装クラスを指示します。
-credentialGeneratorProps	CredentialGenerator 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、setProperties(String) 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

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、特定のプロパティー配列を使用して **-defineDomainServers** のステップ引数を定義する必要があります。このプロパティー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タログ・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タログ・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 の使用:

```
$AdminTask createXSDomain {-interactive}
```

- Jython ストリングの使用:

```
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 の使用:

```
$AdminTask deleteXSDomain {-name TestDomain }
```

- Jython ストリングの使用:

```
AdminTask.deleteXSDomain('[-name TestDomain ]')
```

対話モードの使用例

- Jacl の使用:

```
$AdminTask deleteXSDomain {-interactive}
```

- Jython ストリングの使用:

```
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 の使用:

  ```
  $AdminTask getDefaultXSDomain {-interactive}
  ```

- Jython ストリングの使用:

  ```
  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 の使用:

  ```
  $AdminTask listXSDomains
  ```

- Jython ストリングを使用:

  ```
  AdminTask.listXSDomains
  ```

対話モードの使用例

- Jacl の使用:

  ```
  $AdminTask listXSDomains {-interactive}
  ```

- Jython ストリングの使用:

  ```
  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、特定のプロパティー配列を使用して **-modifyEndpoints**、**-addEndpoints**、および **-removeEndpoints** ステップ引数を定義する必要があります。このプロパティー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 17. modifyXSDomain コマンド引数

引数	説明
-name (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 17. modifyXSDomain コマンド引数 (続き)

引数	説明
-default	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)
-properties	カタログ・サービス・ドメインのカスタム・プロパティーを指定します。

表 18. modifyEndpoints ステップ引数

引数	説明
name_of_endpoint	カタログ・サービス・エンドポイントの名前を指定します。 • **既存のアプリケーション・サーバーの場合**: エンドポイントの名前は、*cell_name¥node_name¥server_name* の形式でなければなりません。 • **リモート・サーバーの場合**: 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。

表 18. modifyEndpoints ステップ引数 (続き)

引数	説明
endpoint_ports	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、*<client_port>,<listener_port>* の順序で指定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。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。 **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ー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**: コンテナー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 BOOTSTRAP_ADDRESS 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。

表 19. addEndpoints ステップ引数

引数	説明
name_of_endpoint	カタログ・サービス・エンドポイントの名前を指定します。 • **既存のアプリケーション・サーバーの場合**: エンドポイントの名前は、*cell_name*¥*node_name*¥*server_name* の形式でなければなりません。 • **リモート・サーバーの場合**: 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。
custom_properties	カタログ・サービス・ドメイン・エンドポイントのカスタム・プロパティーを指定します。カスタム・プロパティーがない場合、この引数には一組の二重引用符 ("") を使用します。

表 19. addEndpoints ステップ引数 (続き)

引数	説明
endpoint_ports	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、*<client_port>,<listener_port>* の順序で指定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。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。 **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ー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**: コンテナー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 BOOTSTRAP_ADDRESS 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。

表 20. removeEndpoints ステップ引数

引数	説明
name_of_endpoint	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 21. *configureClientSecurity* ステップ引数

引数	説明
-securityEnabled	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。(ブール値: `true` または `false` に設定)
-credentialAuthentication (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。 **常になし** クライアント証明書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** (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。
-authenticationRetryCount (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。
-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 実装クラスを指示します。
-credentialGeneratorProps	CredentialGenerator 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、setProperties(String) 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 の使用:

```
$AdminTask modifyXSDomain {-name TestDomain -default true -modifyEndpoints
{{xhost1.ibm.com "" ,2809}} -addEndpoints {{xhost2.ibm.com "" ,2809}}}
-removeEndpoints {{xhost3.ibm.com}}}
```

- Jython ストリングの使用:

```
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 の使用:

  ```
  $AdminTask modifyXSDomain {-interactive}
  ```

- Jython ストリングの使用:

  ```
  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 の使用:

  ```
  $Admintask testXSDomainConnection
  ```

- Jython ストリングの使用:

  ```
  AdminTask.testXSDomainConnection
  ```

対話モードの使用例

- Jacl の使用:

  ```
  $AdminTask testXSDomainConnection {-interactive}
  ```

- Jython ストリングの使用:

  ```
  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・セキュリティー・プロパティー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・セキュリティー・プロパティー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・セキュリティー・プロパティーは使用されません。

戻り値:

バッチ・モードの使用例

- Jacl の使用:

```
$Admintask testXSServerConnection {-host xhost1.ibm.com -listenerPort 2809}
```

- Jython ストリングの使用:

```
AdminTask.testXSServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
```

対話モードの使用例

- Jacl の使用:

```
$AdminTask testXSServerConnection {-interactive}
```

- Jython ストリングの使用:

```
AdminTask.testXSServerConnection ('[-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ー・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。

Name:

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ー・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。

Name:

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 コンテナーは、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、すべてのコンテナー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 構成に事前に存在している必要があります。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ーを定義します。

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ー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: コンテナー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 22. 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・プロパティー:

カスタム CredentialGenerator 実装クラスのプロパティーを指定します。このプロパティーは、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ー値は、「**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す。

カタログ・サービス・ドメインのカスタム・プロパティー:

カスタム・プロパティーを定義すると、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・プロパティー**」の順にクリックします。新規カスタム・プロパティーを作成するには、「**新規**」をクリックします。

Name:

カタログ・サービス・ドメインのカスタム・プロパティー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・プロパティーの値を指定します。

WebSphere Application Server のコンテナー・サーバーの構成

WebSphere Application Server 内のコンテナー・サーバーは、サーバー・プロパティー・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、コンテナー・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、 277 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 内のコンテナー・サーバーを作成するには、コンテナー・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 コンテナー・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、 199 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・プロパティー・ファイルを構成し、サーバー・プロパティー・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・プロパティー・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、『コンテナー・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。

コンテナー・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 環境内のコンテナー・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ー・サーバー Java 仮想マシン (JVM) として登録されます。コンテナー・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` フォルダーに 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` フォルダーで `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)

CWOBJ3013E: 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、462 ページの『配置の制御』を参照してください。

次のタスク

コンテナー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ー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ー 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

エンティティー・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、エンティティー (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。
  - WebSphere Application Server 環境内で稼働するコンテナー・サーバーを使用する場合
  - 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
  - ReplicationMapListener インターフェースを使用して、レプリカ生成モードのクライアント・サイド・マップ用にイベント・リスナーの実装を作成する場合

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。

.

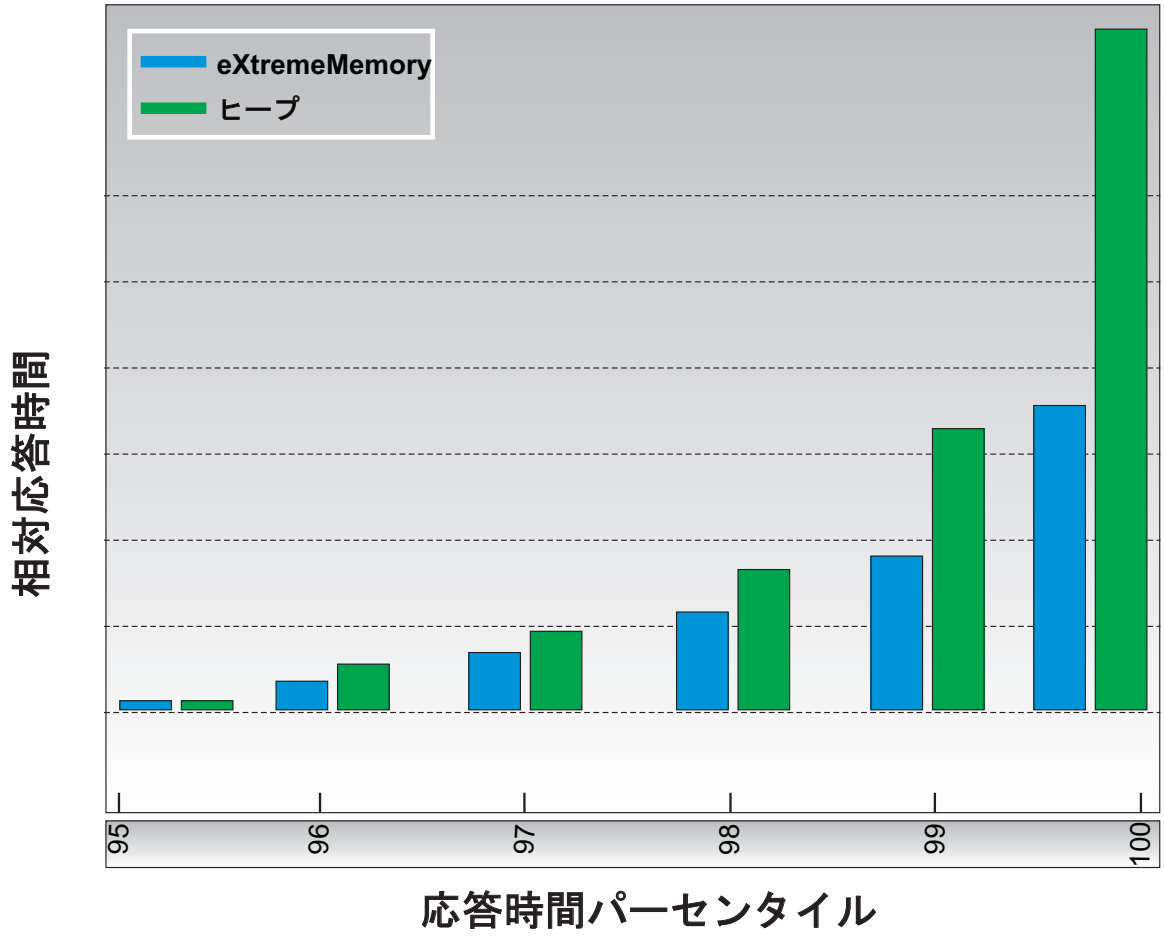


図 30. eXtremeMemory とヒープ・ストレージの応答時間の比較

eXtremeMemory を使用すると、コンテナー・サーバー間の通信に eXtremeIO が使用されます。オブジェクトは、コンテナー・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ー・サーバーで必要なサーバー・プロパティーを設定し、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** プロパティー値を決定します。
   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ー・サーバーのサーバー・プロパティー・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・プロパティーにより、新しいトランスポートが使用可能になります。

必須プロパティー

7.1.1+ **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・エントリーは、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ー・サーバーは、**enableXM** プロパティー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ー

7.1.1+ **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%

オプションのプロパティー

7.1.1+ **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`

7.1.1+ **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`

7.1.1+ **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`

7.1.1+ **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`

7.1.1+ **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** プロパティーが `TCP/IP` に設定されている場合のみこのプロパティーが使用されます。

7.1.1+ **xioChannel.xioContainerTCPSecure.Port**
サーバー上の eXtremeIO の SSL ポート番号を指定します。
transportType プロパティーが 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ーが使用されます。

3. コンテナー・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、 427 ページの『スタンドアロン・サーバーの始動と停止』および 443 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、プロパティー・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ーを使用できます。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、 39 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・プロパティー・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、 45 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・プロパティー・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・プロパティー・ファイルを参照してください。

  重要: プロパティー名では大/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ーに指示します。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。

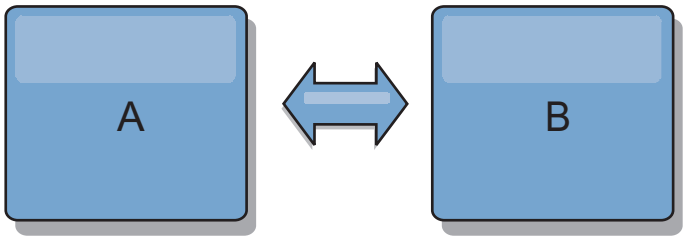


図 31. カタログ・サービス・ドメイン間のリンク

ドメイン A 内のカタログ・サーバーのサーバー・プロパティー・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・プロパティー・ファイルを以下のようにします。 2 つのプロパティー・ファイルが似ていることに着目してください。

```
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・ネーム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・プロパティー・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

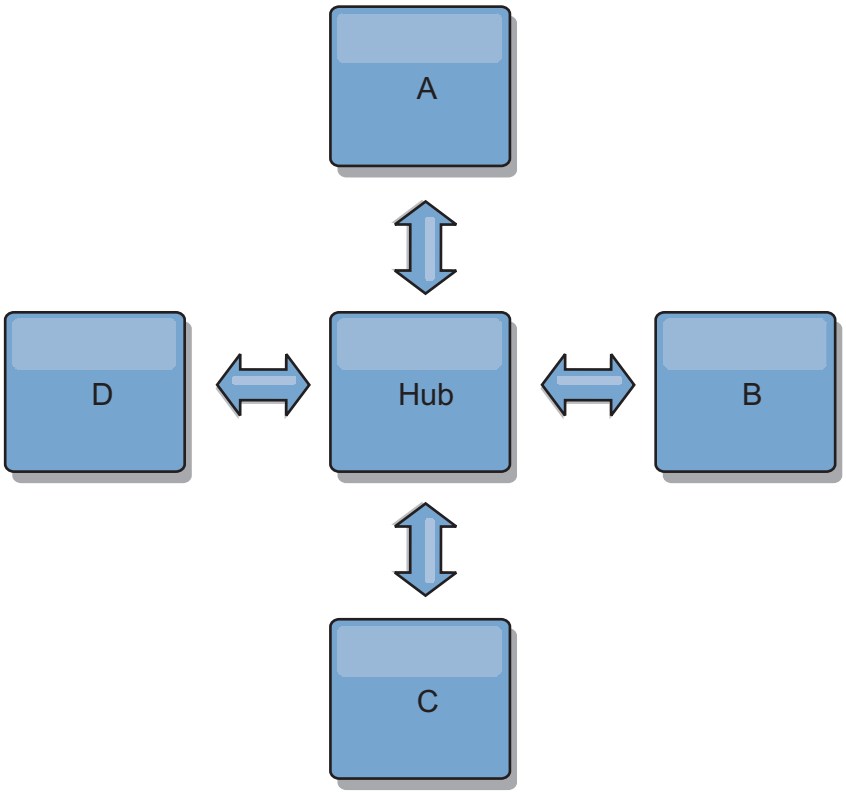


図 32. ハブおよびスポーク・トポロジー

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・プロパティー・ファイルになります。

```
domainName=Hub
```

スポーク A のサーバー・プロパティーは次のとおりです。

```
domainName=A
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

スポーク B のサーバー・プロパティーは次のとおりです。

```
domainName=B
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

スポーク C のサーバー・プロパティーは次のとおりです。

```
domainName=C
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

スポーク D のサーバー・プロパティーは次のとおりです。

```
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ー・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ー・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。

   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>
   コンテナーでは、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
```

カタログ・サービス・エンドポイントは、catalogServiceEndPoints サーバー・プロパティーを使用して設定することもできます。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、またカタログ・サービスとコンテナー・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、コンテナー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、コンテナー・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

リスナー・ポートとリスナー・ホストは、listenerHost および listenerPort サーバー・プロパティーを使用して設定することもできます。

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。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。

デフォルト: 1099

JMX サービス・ポートは、JMXServicePort サーバー・プロパティーを使用して設定することもできます。 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、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 33. コマンド行の使用例： hostA で最初のカタログ・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 番目のカタログ・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・カタログ・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・サーバーとカタログ・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、カタログ・サーバー、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。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>
-listenerPort <port>
```

リスナー・ポートとリスナー・ホストは、listenerHost および listenerPort サーバー・プロパティーを使用して設定することもできます。

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>
peerPort と同義。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は、カタログ・サービスは使用可能なポートを自動的に生成します。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。 (WebSphere Application Server 環境のみで必須。)

HA マネージャー・ポートは、HAManagerPort サーバー・プロパティーを使用して設定することもできます。

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、次のコマンド行オプションを使用します。

```
-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>
```

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。

   ```
   -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、コンテナー・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とき、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ます。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、69 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. コンテナー・サーバー・エンドポイント

   WebSphere eXtreme Scale コンテナー・サーバーは、Java EE モジュール内でホスティングされます。コンテナー・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。コンテナー・サービスが使用するポートについて詳しくは、69 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ーを開始する方法について詳しくは、 296 ページの『コンテナー・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(catalogServerAddresses, 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す。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す。

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ー・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・プロパティー
- startOgServer.sh|bat 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・プロパティー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。

7.1.1+ 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ー・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

7.1.1+ 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、コンテナー・サーバー上でバイト列にシリアライズされます。詳しくは、299 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ーを受け渡し、データ・グリッドのトランス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、 530 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_root*properties/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

7.1.1+ **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、 582 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、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る場合があります。

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** プロパティーを追加します。 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。プロパティーと設定の説明は、 530 ページの『ORB プロパティー』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

7.1.1+ **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、 582 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。

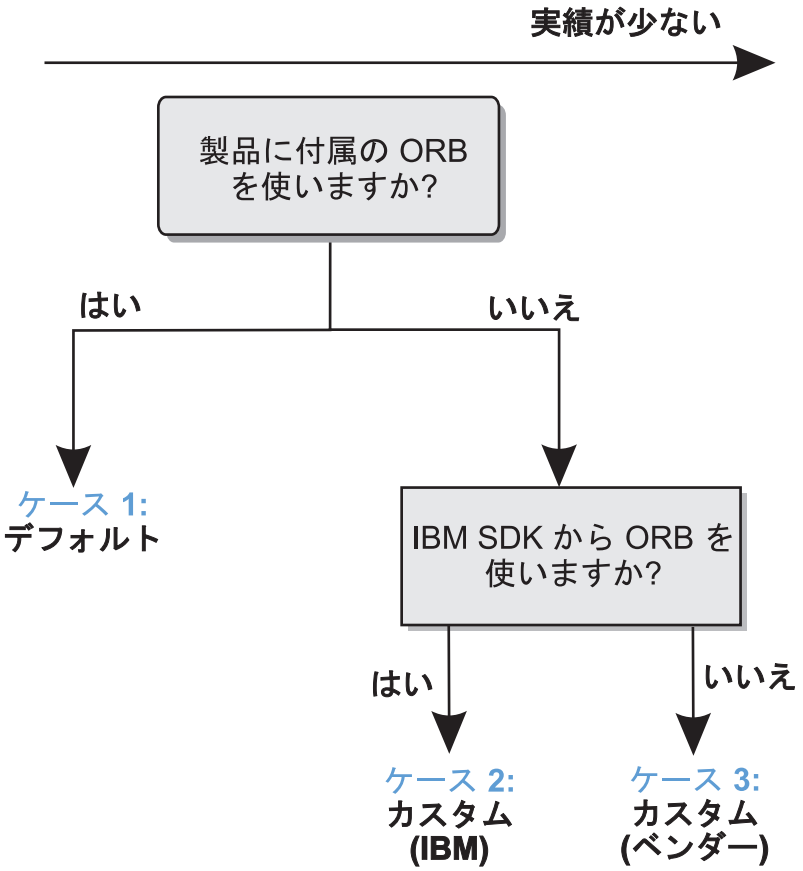


図 34. ORB の選択

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、Sun Microsystems 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。変更内容を保存します。

     - 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ーを追加します。

```
-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、コンテナー・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. コンテナー・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ー・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"
                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>
```

コンテナー・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。eXtreme Scale 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、 251 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;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>
```

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・プロパティー・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・プロパティー設定に優先します。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 アベイラビリティー例外であれば、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・プロパティー・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・プロパティー・ファイル内の requestRetryTimeout プロパティーを追加または変更します。クライアント・プロパティーは、デフォルトでは `objectGridClient.properties` ファイルです。requestRetryTimeout プロパティー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、このプロパティーを除去するか、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・プロパティー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・プロパティー・ファイルを作成します。 以下の例では、クライアント・プロパティー・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・プロパティーを設定します。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・プロパティーがインスタンスに設定されています。クライアント・プロパティー・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;
  ClientClusterContext 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・プロパティー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・プロパティー・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。

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 コンテナーと連

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ー、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、 178 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・セキュリティーを有効にする必要があります。サーバー・プロパティー・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。

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` ディレクトリー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 コンテナー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ー・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ー・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ー・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ー・サーバーと通信します。

コンテナー・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 コンテナーを開始します。

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、328 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・プロパティー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・プロパティー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・プロパティーを構成します。

     a. WebSphere Application Server 管理コンソールで、カスタム・プロパティー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・プロパティー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・プロパティー**」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・プロパティーを設定するには、「**アプリケーション・サーバー**」 > 「***<server_name>***」 > 「**管理**」 > 「**カスタム・プロパティー**」をクリックします。プロパティー名は

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・プロパティーを追加します。このカスタム・プロパティーの値には、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・プロパティー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・プロパティーの名前を <*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・プロパティー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ーブレット・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ーが入ったスプライサー・プロパティー・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:**

a. `cd wxs_home/optionalLibraries/ObjectGrid/session/bin`

b. `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:**

a. `cd was_root/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear was_root/session/samples/splicer.properties`

接合されるサーブレット・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ー・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、346 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。

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、各サー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ーを変更できます。構成プロパティーは `splicer.properties` ファイル内のプロパティー名と同一です。 `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、346 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 コンテナー・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 コンテナー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 コンテナーとリモート・コンテナー・サーバー間のネットワークの問題
- リモート・コンテナー・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 コンテナー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 コンテナー・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 コンテナー・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ー・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、 324 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、 277 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。
- カタログ・サービス・ドメインを構成する際、コンテナー・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、 569 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・プロパティー・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** プロパティーを true に設定します。
- 7.1.1+ 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 コンテナー・カスタム・プロパティーが true に設定されていること。このプロパティーによって、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す。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・プロパティーを false に設定します。このカスタム・プロパティーについて詳しくは、 588 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 microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.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 microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.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 microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.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・プロパティー**」 > 「**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: デプロイメント・マネージャー・プロファイル・パス上の splicer.properties ファイル
  – 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** プロパティーを true に設定します。

splicer.properties ファイルのプロパティーに関して詳しくは、 349 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 コンテナー設定」 > 「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・プロパティー」 > 「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」* > 「コンテナー設定」 > 「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・プロパティーを設定します。このカスタム・プロパティー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 コンテナー**」 > 「**カスタム・プロパティー**」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。

以下のプロパティーを使用して、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・プロパティー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 コンテナー**」 > 「**カスタム・プロパティー**」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、機能のプロパティーに設定されているものと同じプロパティーを所有する必要があります。

表 23. ObjectGrid を使用した SIP セッション管理のためのカスタム・プロパティー

property	値	デフォルト
com.ibm.sip.ha.replicator.type	`OBJECTGRID`: SIP セッション・ストアとして ObjectGrid を使用	
min.synchronous.replicas	同期レプリカの最小数	0
max.synchronous.replicas	同期レプリカの最大数	0
max.asynchronous.replicas	非同期レプリカの最大数	1
auto.replace.lost.shards	詳しくは、 254 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	• `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 • `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	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・グリッドをホストするコンテナー・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。
  - コンテナー・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。
  - コンテナー・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、 178 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 層でのプロセッサー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と、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・データをコンテナー・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・プロパティーを使用して、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詳しくは、324 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. コンテナー・サーバーを WebSphere Application Server の外部に置くリモート・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 コンテナー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、コンテナーを開始します。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
   ```

   コンテナー・サーバーの開始方法について詳しくは、430 ページの『コンテナー・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、コンテナー・サーバーの構成および開始方法について詳しくは、295 ページの『WebSphere Application Server のコンテナー・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 コンテナー・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 コンテナー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 コンテナーとリモート・コンテナー・サーバー間のネットワークの問題

- リモート・コンテナー・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 コンテナー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 コンテナー・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 コンテナー・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 コンテナーのデフォルト・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ー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 コンテナー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 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ーブレット・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ーが含まれたスプライサー・プロパティー・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`

接合されるサーブレット・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ー・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、346 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。

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、各サー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ーを変更できます。構成プロパティーは `splicer.properties` ファイル内のプロパティー名と同一です。 `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、346 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. 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ーのタイプによって異なります。

   - リモート・コンテナー・サーバー: `ogclient.jar` と `sessionobjectgrid.jar`

- 組み込みコンテナー・サーバー: `objectgrid.jar` と `sessionobjectgrid.jar`

3. オプション: リモート・コンテナー・サーバーを使用する場合、コンテナー・サーバーを開始します。 詳しくは、 430 ページの『コンテナー・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 コンテナー・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 コンテナー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 コンテナーとリモート・コンテナー・サーバー間のネットワークの問題
- リモート・コンテナー・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 コンテナー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 コンテナー・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 コンテナー・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ー・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、コンテナー・サーバーは Web コンテナー層で始動します。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 35. 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** プロパティー。
- 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。

7.1.1+ **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、コンテナー・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** プロパティーが 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 コンテナー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 36. 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** プロパティー。
- 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 属性を除き、すべての mapSet プロパティーは変更可能です。

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 エレメントの placementStrategy 属性

リモート XML パッケージ

コンテナーがスタンドアロン・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 37. 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** プロパティー。
- 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。

7.1.1+ **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、コンテナー・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** プロパティーが 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 コンテナーが無効にし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 38. 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** プロパティー。
- 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 属性を除き、すべての mapSet プロパティーは変更可能です。

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 エレメントの placementStrategy 属性

サーブレット・コンテキスト初期化パラメーター

以下に示すサーブレット・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で必要なスプライサー・プロパティー・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 コンテナーが開始されます。

`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 コンテナー・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、このプロパティーは 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 コンテナー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 コンテナー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ので、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 コンテナーの最大スレッド・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・データをエント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い場合や属性のサイズが大きい場合は、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・プロパティー・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ー。このプロパティーが、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** プロパティー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** プロパティー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** プロパティー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である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 コンテナー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。このプロパティーの値は、Web コンテナー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 コンテナー・カスタム・プロパティー 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

ション・パーシスタンスを構成する前に、Web コンテナー・カスタム・プロパティーに HttpSessionIdReuse=false カスタム・プロパティーを設定します。

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 コンテナー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・プロパティー

このファイルに説明されている追加プロパティーを使用する場合は、有効にするプロパティー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ー・ファイル。
#
# このプロパティー・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、このプロパティー・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・
# プロパティーを使用して、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。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 WebShere 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、178 ページの

178 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・セキュリティーを有効にする必要があります。サーバー・プロパティー・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。

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ーが使用されます。これらの構成プロパティーは、Java 仮想マシン (JVM) のカスタム・プロパティーとして設定する必要がある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ーに該当します。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・プロパティーではなくシステム・プロパティー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・プロパティーを使用して、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・プロパティーは、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・プロパティーを作成するには、「**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・プロパティー**」 > 「**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・プロパティー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・プロパティーは、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・プロパティーをベース・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ーは、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ーになります。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ようにするには、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・プロパティーを作成し、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す。

   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・プロパティー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・プロパティーの値を `true` に設定します。このカスタム・プロパティーは、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。

3. 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。
4. 動的キャッシュ・サービスのトポロジーを構成します。

   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・トポロジーです。このカスタム・プロパティーは、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・プロパティーの名前を com.ibm.websphere.xs.dynacache.topology と入力します。

   このプロパティーに許可される 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

   組み込みトポロジーまたは組み込み区画化トポロジーを使用する場合は、com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent カスタム・プロパティーを 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・プロパティーは、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。
5. オプション: 組み込み区画化トポロジーを使用する場合、動的キャッシュ・サービスの初期コンテナー数を構成します。

   初期コンテナーの数を構成することで、組み込み区画化トポロジー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・プロパティーとして設定します。

   プロパティーの名前を com.ibm.websphere.xs.dynacache.num_initial_containers と入力します。

   この構成プロパティーの推奨値は、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。

   組み込みトポロジーまたは組み込み区画化トポロジーの場合は、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・プロパティーを設定して、初期コンテナーをすぐに使用できるようにします。

   ```
   com.ibm.ws.cache.CacheConfig.createCacheAtServerStartup=true
   ```
6. 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、 277 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・プロバイダーで使用し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。詳細は、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 コンテナーの CLASSPATH に配置する必要があります。詳しくは、 430 ページの『コンテナー・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 コンテナー・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 コンテナー・サーバーに保管されます。コンテナー・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ーを作成します。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 コンテナー・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・コンテナー・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ー・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・コンテナーの XML 構成ファイルは、*wxs_install_root*/customLibraries/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・コンテナー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、実行中のコンテナー・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: 一連のコンテナー・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・コンテナーのセットを使用しなければなりません。コンテナーの数とそれらのコンテナー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、63 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・コンテナー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 (コンテナー) 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ーおよびプロパティーは、`persistence.xml` ファイルで構成できます。これらのプロパティーの構成について詳しくは、364 ページの『JPA キャッシュ構成プロパティー』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 ローダーを使用します。詳しくは、JPA ローダーを参照してください。

JPA L2 キャッシュ・トポロジーに関する考慮事項

次の要因が、構成するトポロジー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、359 ページの『組み込みトポロジー』または 358 ページの『イントラドメイン・トポロジー』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、360 ページの『組み込みの区画化トポロジー』、または 362 ページの『リモート・トポロジー』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。トポロジーごとに、読み取り操作および書き込み操作の処理は異なります。
   - 359 ページの『組み込みトポロジー』: ローカル読み取り、リモート書き込み
   - 358 ページの『イントラドメイン・トポロジー』: ローカル読み取り、ローカル書き込み
   - 360 ページの『組み込みの区画化トポロジー』: 区画化: リモート読み取り、リモート書き込み
   - 362 ページの『リモート・トポロジー』: 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、組み込みトポロジーおよびイントラドメイン・トポロジーを使用します。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・トポロジーを使用します。
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、 359 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・エンティティーにリレーションシップがなければ、ターゲット・エンティティーが取り出されます。ターゲット・エンティティーに 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・エンティティー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できる必要があります。

7.1.1+ 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ー・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。

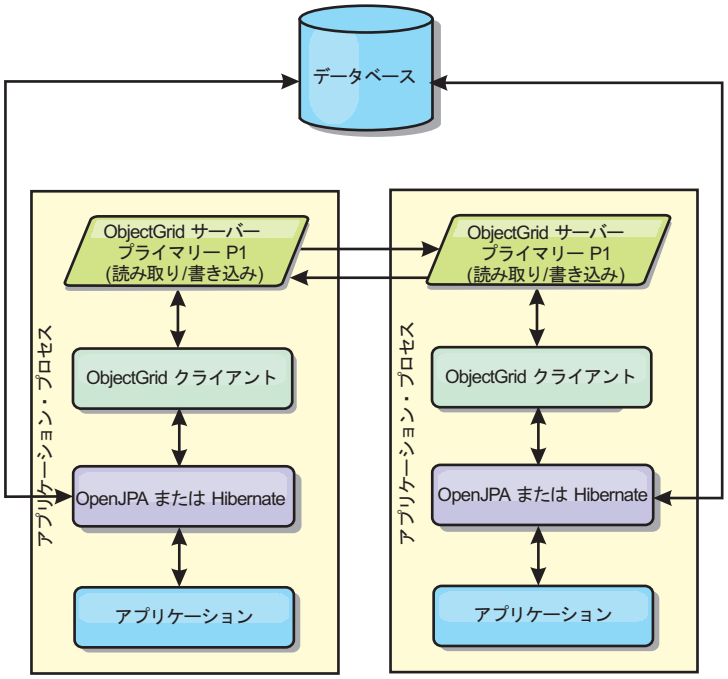


図 39. JPA イントラドメイン・トポロジー

イントラドメイン・トポロジーでの関連 JPA キャッシュ構成プロパティー:

```
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、コンテナー・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。
- すべてのコンテナー・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 コンテナー・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ー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。

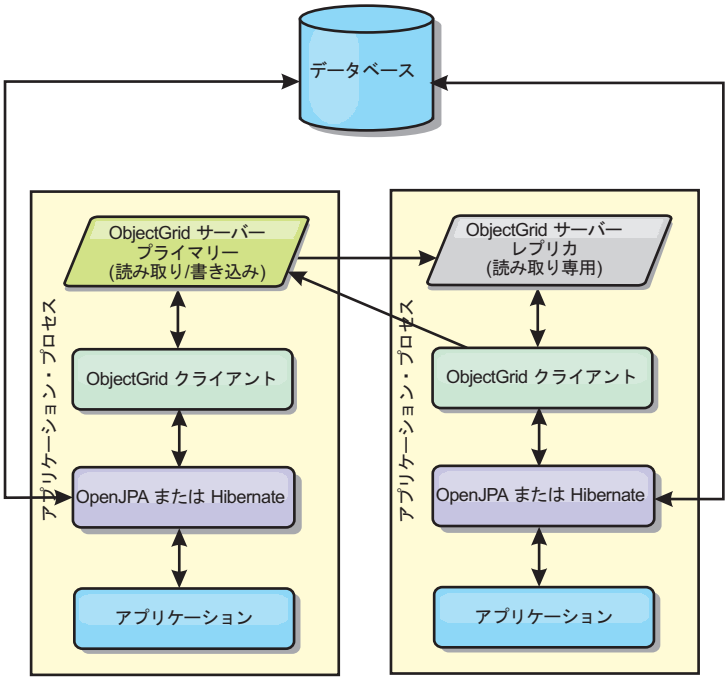


図 40. JPA 組み込みトポロジー

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ー:

```
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、エンティティー・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・キャッシュからエンティティー・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。

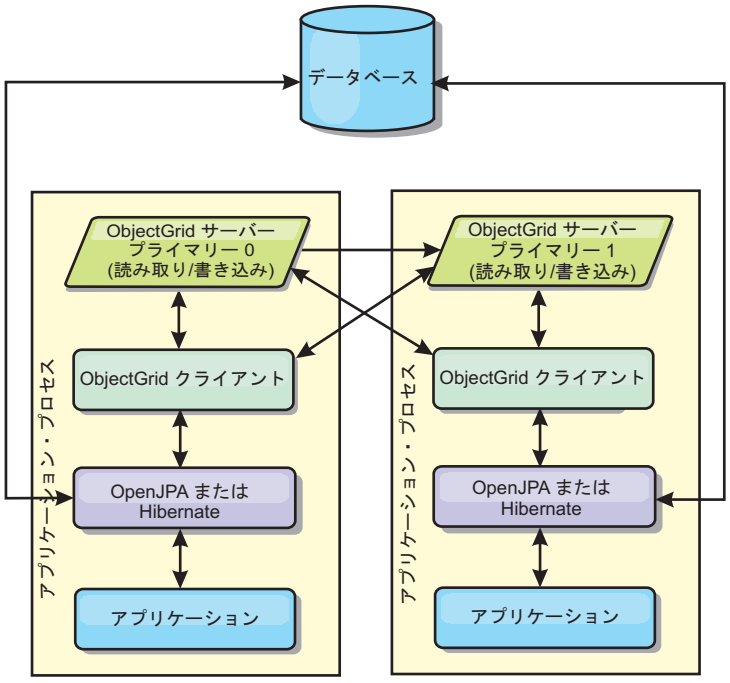


図 41. JPA 組み込み区画化トポロジー

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ー:

```
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、エンティティー・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・キャッシュからエンティティー・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せます。 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。

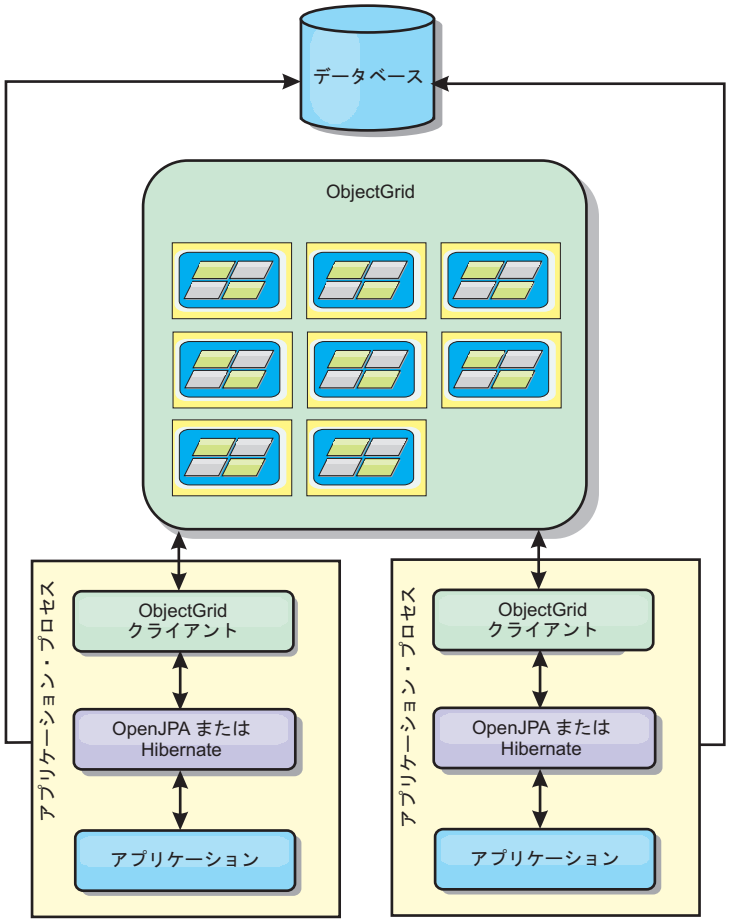


図 42. JPA リモート・トポロジー

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ー:

```
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、プロパティー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 とのすべての対話が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ー

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ー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 ローダーを使用します。詳しくは、381 ページの『JPA ローダーの構成』を参照してください。

プロパティーの場所

これらのプロパティーは、persistence.xml ファイルの中で構成できます。これらのプロパティー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 でプロパティーを設定できます。

```
<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ー

構成で何の値も指定されない場合、次のデフォルト・プロパティー値が使用されます。

- **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)

プロパティー

次のプロパティーを使用して、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 プロパティーを設定し、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・プロパティー・ファイルに関して詳しくは、サーバー・プロパティー・ファイルを参照してください。

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。このプロパティーは、 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。

7.1.1+ **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ー・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ー・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ー・サーバーに、1 つのプライマリー断片を配置します。

7.1.1+ **PlacementScopeTopology**

カタログ・サービス・ドメイン内のコンテナー・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。ハブに障害が起こると、別のコンテナー・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ーは、リングに追加される最初のコンテナーと最後のコンテナー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ーラブルですが、2 つのリンクに障害が起こるだけで一時的に切断されてしまいます。コンテナー・サーバーに障害が起こると、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。このプロパティーは、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。このプロパティーは、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に使用するスレッド・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るべきプロパティーの詳細については、356 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 ローダーを使用してください。詳しくは、381 ページの『JPA ローダーの構成』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ーを設定して、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ーを設定できます。

   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ーは廃棄されます。

   ```
   <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ー・リストに ObjectGridName プロパティー、ObjectGridType プロパティー、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ーを指定すれば、キャッシュ構成をカスタマイズすることができます。以下に例を示します。

   ```
   <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ーのリストについては、364 ページの『JPA キャッシュ構成プロパティー』を参照してください。

2. persistence.xml ファイル内では、openjpa.RemoteCommitProvider プロパティーを 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、371 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、コンテナー・サーバーを開始し、コンテナー・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 コンテナー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ー・サーバー・プロセスの両方があります。コンテナー・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、 427 ページの『スタンドアロン・サーバーの始動』と 430 ページの『コンテナー・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` というプロパティー・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** プロパティー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.AnnuityPersistebleObject</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.Payor" />
      <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.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.AnnuityPersistebleObject"
                  readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
      <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="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.AnnuityPersistebleObject">
      <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. 各エンティティーは、完全修飾エンティティー・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、エンティティーは第 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のある Entity クラスで、L2 キャッシュから除外するエンティティー・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. エンティティー・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 を共有します。
3. QueryCache のサポートには ObjectGridQueryCache マップが必要です。
4. 各エンティティー・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.AnnuityPersistebleObject" />
      <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

プロパティー・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、356 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 ローダーを使用してください。詳しくは、381 ページの『JPA ローダーの構成』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 ディレクトリー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・ディレクトリー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ーを設定して、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ー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** プロパティーの値は 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** プロパティーの値を 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 プロパティーを使用して、eXtreme Scale キャッシュ構成プロパティーを指定します。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 プロパティー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ーはオプションです。

     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 プロパティー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すると、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す。

     objectgrid.configuration プロパティーに設定できる値の詳細については、364 ページの『JPA キャッシュ構成プロパティー』を参照してください。
   - **objectgrid.hibernate.regionNames**: objectgrid.hibernate.regionNames プロパティー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ー・クラスで、そのエンティティー・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ー・アノテーションが設定されているとします。これにより、このエンティティー・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、動的決定の可能性があるすべての 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を処理します。

   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、 378 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、コンテナー・サーバーを開始し、コンテナー・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 コンテナーおよびそのピアと通信します。

各 JPA エンティティーには、エンティティー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ー・サーバー・プロセスの両方があります。コンテナー・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、427 ページの『スタンドアロン・サーバーの始動』と 430 ページの『コンテナー・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 というプロパティー・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** プロパティー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.AnnuityPersistebleObject</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 ディレクトリーに置くと、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。

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、356 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 エンティティー・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・プロパティー・マップを作成し、デフォルトのパーシスタンス・プロパティー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なエンティティー・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 エンティティー・マップに構成されているエンティティー・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 エンティティー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 プロパティー・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>
```

エンティティーには、JPA アノテーションおよび eXtreme Scale エンティティー・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>
    <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、コンテナー・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、エンティティー・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、エンティティー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ー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。

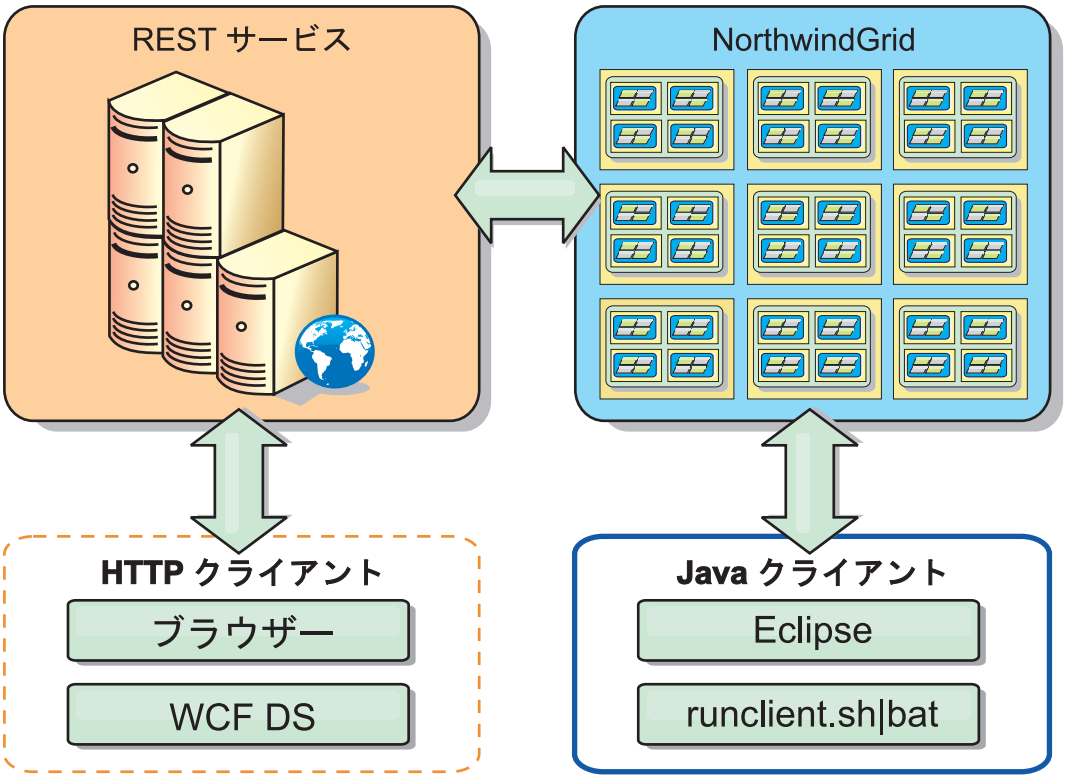


図 43. 開始用 (getting started) サンプル・トポロジー: REST データ・サービスを使用する HTTP クライアントと Java クライアントは同じデータ・グリッドにアクセスできます。

手順

1. eXtreme Scale データ・グリッドを構成および開始します。 386 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。 395 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。 415 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。 417 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 エンティティー・メタデータを表し、各エンティティーを 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 つのコンテナー・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、エンティティー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。

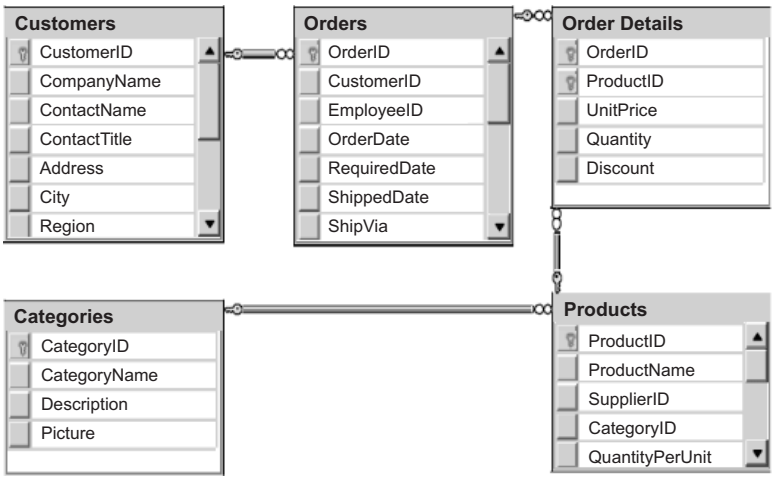


図 44. Microsoft SQL Server Northwind サンプルのスキーマ図

拡張が容易な種類のサンプルを作成するためには、各エンティティーまたは関連エンティティー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

ティー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、エンティティーを 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。

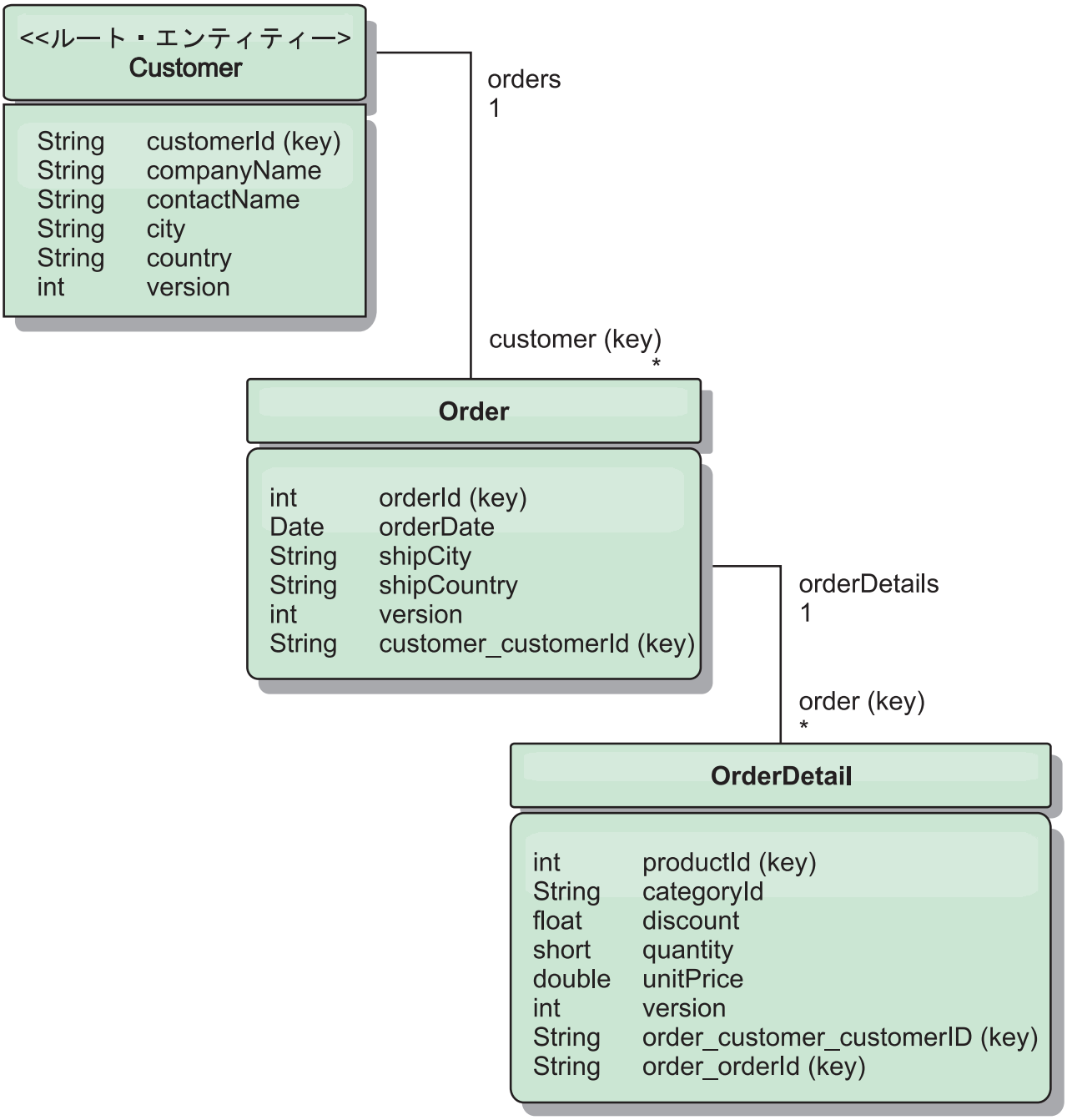


図 45. Customer および Order エンティティーのスキーマ図

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・エンティティーとして使用し、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。

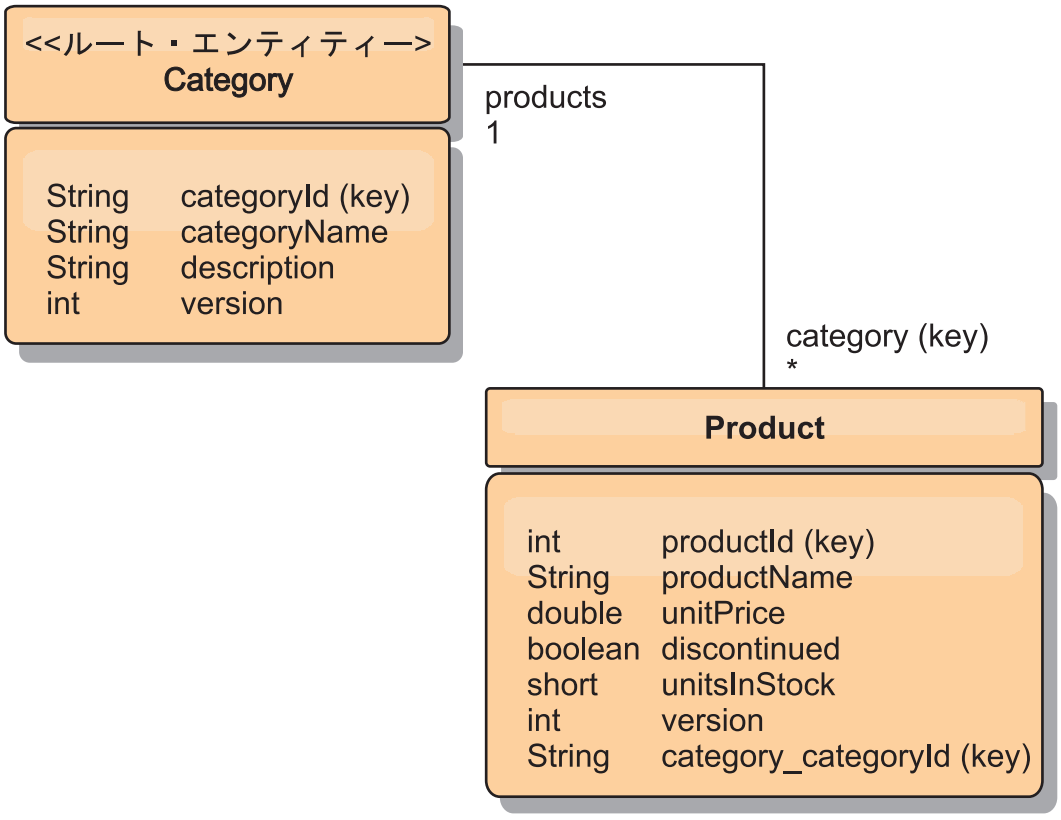


図 46. Category および Product エンティティーのスキーマ図

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、『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ー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、各エンティティーは区画に分割されたルート・エンティティー、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、子エンティティーをアドレス指定するためのルート・エンティティーからのアソシエーションの使用方法の例です。

```
/Customer('ACME')/order(100)
```

WCF Data Services では、子エンティティー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・エンティティーへのアソシエーションの作成により実現され、ルート・エンティティーへの 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。 エンティティーがキーの一部として組み込まれる場合、親エンティティーの属性はキー・プロパティーとして公開されます。

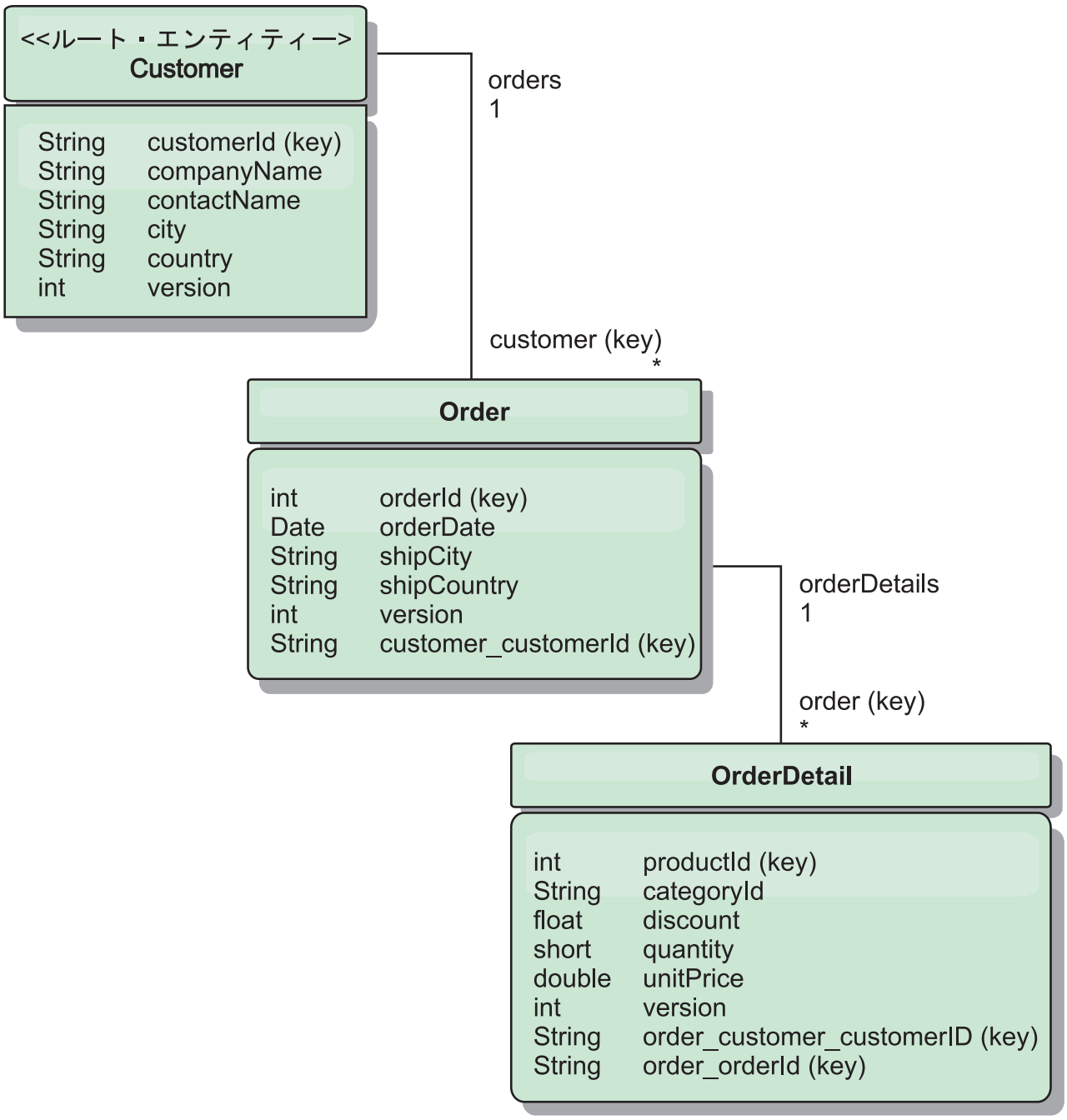


図 47. Customer および Order エンティティーのスキーマ図

Customer/Order エンティティーのスキーマ図で、どのように各エンティティーが 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 エンティティー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ーとして公開します。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 エンティティーの属性名 (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。

エンティティーには、非ルート・エンティティーのすべての上位エンティティーがルートへのキー・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・エンティティー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 エンティティーへのキー・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 エンティティーから Product エンティティーへのアソシエーションは分割され、Orders および Product インベントリーを個別に区画に分割することができます。 OrderDetail エンティティーは、強いリレーションシップの代わりにカテゴリーと製品 ID を保管します。 2 つのエンティティー・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・エンティティーが Category で、各 Product には Category エンティティーへのアソシエーションがあることを表しています。Category エンティティーは Product ID に組み込まれています。 REST データ・サービスは、キー・プロパティーを公開します。 category_categoryId で Product に直接アドレスを指定できます。

Category はルート・エンティティーなので、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 エンティティーに 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 つのコンテナー・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、コンテナー・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 番目のコンテナー・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端末ウィンドウに表示されると、コンテナー・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、開始したコンテナーの名前です。

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ーが開始されます。

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 つのコンテナー・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、コンテナー・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 番目のコンテナー・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

コンテナー・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、コンテナー・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ーの名前です。

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・ディレクトリー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 セキュリティー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、394 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。217 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。
   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。

```
CWOBJ4000I: 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。 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、386 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。`CWOBJ4000I: 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) または Sun の JRE か JDK のバージョン 5 以上をインストールして、 JAVA_HOME 環境変数を設定します。
- 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、 392 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、 217 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 リポジトリーに追加します。
   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「**リポジトリー**」リンクをクリックします。
   c. 「**リポジトリー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 24. リポジトリー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

   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ー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 25. 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。

```
CWOBJ4000I: 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 リポジトリーに追加します。

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「**リポジトリー**」リンクをクリックします。

3) 「**リポジトリー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 26. リポジトリー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 27. 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

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。

`CWOBJ4000I: 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」プロパティー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、386 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ーを設定します。

```
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 リポジトリーに追加します。
      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「**リポジトリー**」をクリックします。
      3) 「**リポジトリー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. リポジトリーへのアーカイブ

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 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 29. インストール値

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`

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。

CWOBJ4000I: 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 または Sun の JRE か JDK のバージョン 5 以上をインストールして、JAVA_HOME 環境変数を設定します。
- 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. Sun JRE または JDK を使用する場合は、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、 241 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、 384 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、 217 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ー許可を追加します。
   - 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。

        ```
        CWOBJ4000I: 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 のプロパティー・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 のプロパティー・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。

```
CWOBJ4000I: 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、386 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. Sun JRE または JDK を使用する場合は、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

から

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

     から

     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

      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ーを設定します。

```
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。

`CWOBJ4000I: 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 のプロパティー・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。 `CWOBJ4000I: 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

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、ブラウザー内で 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。古いブラウザーでは、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。

最新のブラウザー (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ーでネイティブ表示できます。以下のトピックでは、ブラウザー内で 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、ブラウザーに戻ります。
  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は表示されません。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 が含まれます。 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 の各エンティティー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 エンティティーをデータ・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 エンティティー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 エンティティーが含まれます (最大数は <maxItems>)。

   - `display customer companyCode`

     Customer エンティティー、および関連付けられた Order エンティティーと OrderDetail エンティティーを表示します。

   - `display category categoryId`

製品 Category エンティティーおよび関連付けられた Product エンティティーを表示します。

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、コンテナー・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・ディレクトリー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 コンテナー・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 コンテナー・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 の各エンティティー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 エンティティーをデータ・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 エンティティー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 エンティティーが含まれます (最大数は <maxItems>)。

- `display customer <companyCode>`

  Customer エンティティー、および関連付けられた Order エンティティーと OrderDetail エンティティーを表示します。

- `display category <categoryId>`

  製品 Category エンティティーおよび関連付けられた Product エンティティーを表示します。

- `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ーを削除します。

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ー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 コンテナー・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。

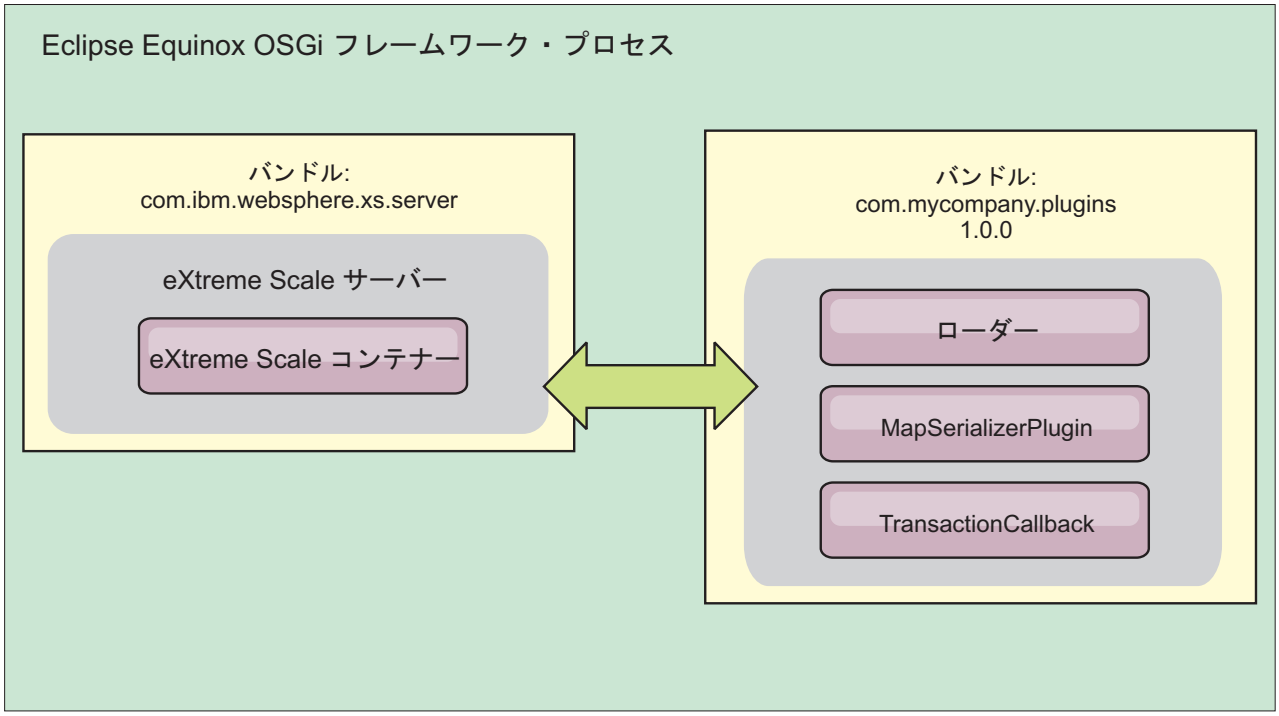


図 48. eXtreme Scale プラグインを含んでいる OSGi バンドルをインストールおよび開始する Eclipse Equinox プロセス

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 コンテナーによって 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 を使用するように構成しなければな

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ないため、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 つのプロパティーがあります。blueprintContainer プロパティーには blueprintContainer 参照を設定し、beanId プロパティーには 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 コンテナーを使用して 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 コンテナー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 コンテナー内に公開される 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 コンテナー・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 コンテナーを使用する 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ーを構成する方法を説明します。この手順の結果として、コンテナー・バンドルが作成されます。コンテナー・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ーを開始します。

コンテナー・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: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・プロパティーを使用して 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ーの詳細については、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 コンテナーの XML 定義を追加します。例えば、次のようにします。

```
<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ー・バンドル内に保管します。 Blueprint XML は OSGI-INF/blueprint ディレクトリー内に保管し、Blueprint コンテナー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 コンテナー・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。コンテナー・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、コンテナー・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ー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 コンテナー・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・プロパティー・ファイルを参照するように設定します。コンテナーを構成するには、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ーを開始します。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ー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、標準プロパティー・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 コンテナー・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・プロパティー・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 コンテナーの管理サービス・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 コンテナー・サーバーが構成されました。

次のタスク

コンテナー・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ー・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ー・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、571 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、コンテナー・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、276 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、セキュリティー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、433 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。

3. 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ー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 コンテナー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。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、302 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、571 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。
- **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);

  //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、444 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。

コンテナー・サーバーの始動

コンテナー・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

コンテナー・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、コンテナーがホストする eXtreme Scale サーバーを指定します。コンテナーに渡される XML の各 ObjectGrid をホストするようにコンテナー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、すべてコンテナー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ー・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: コンテナーでは、**-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。コンテナーを開始する場合は、**-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ーを開始してください。カタログ・サーバー ORB が、コンテナー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ーを開始します。**

  必須ではありませんが、コンテナーの開始時には、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ー 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には、StartOgServer スクリプトを変更する代わりに、-jvmArgs -cp C:¥ . . . ¥DirectoryPOJOs¥POJOs.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・プロパティー・ファイルを使用してコンテナーを開始します。**

  サーバー・プロパティー・ファイルを使用して、コンテナー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・プロパティー・ファイルを使用してコンテナー 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・プロパティー・ファイルです。 server.properties ファイルに関して詳しくは、サーバー・プロパティー・ファイルを参照してください。

- **コンテナー・サーバーをプログラマチックに始動します。**

  コンテナー・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、444 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ー・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には、 startOgServer スクリプトを変更する代わりに、`-jvmArgs -cp C:¥ . . . ¥DirectoryPOJOs¥POJOs.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>
コンテナーでは、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ー・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ー・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>
peerPort と同義。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は、カタログ・サービスは使用可能なポートを自動的に生成します。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。 (WebSphere Application Server 環境のみで必須。)

-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。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。

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、コンテナー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、コンテナー・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・サーバー上のクォーラムを使用可能にします。詳しくは、カタログ・サーバー・クォー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ーを始動し、その後必要に応じてパラメーター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、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>
サーバーが始動したとき使用可能になるトレースの有効範囲を指定するストリングを指定します。

例:

- ObjectGrid=all=enabled
- ObjectGrid*=all=enabled

-traceFile <trace file>
トレース情報を保存するファイルのパスを指定します。

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。

コンテナー・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]
```

コンテナー・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。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。**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、コンテナー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、コンテナー・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、コンテナーがホストする 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、コンテナーがホストする 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ーを始動し、その後必要に応じてパラメーター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>
トレース情報を保存するファイルのパスを指定します。

例: `../logs/c4Trace.log`

-traceSpec <trace specification>
サーバーが始動したとき使用可能になるトレースの有効範囲を指定するストリングを指定します。

例:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-zone <zone name>
サーバー内のすべてのコンテナー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、262 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

- **単一のコンテナー・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ー・サーバーを停止します。単一のコンテナー・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。複数のコンテナー・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、コンテナー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。コンテナーの開始に **-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。

- **複数のコンテナー・サーバーを停止します。**

  同時に複数のコンテナー・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。コンテナー・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ーを停止する場合は、**-teardown** パラメーターを使用します。詳しくは、 442 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 ディレクトリーにあります。詳しくは、 479 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。**

  コンテナー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 ディレクトリーに移動します。

     ```
     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、 580 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、 444 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ー・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ー・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。

コンテナー・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、コンテナー・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。コンテナー・サーバーを始動するときに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に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>
サーバーが始動したとき使用可能になるトレースの有効範囲を指定するストリングを指定します。

例:

- ObjectGrid=all=enabled
- ObjectGrid*=all=enabled

-traceFile <trace file>
トレース情報を保存するファイルのパスを指定します。

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ー・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ー・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ー・サーバーを構成します。

- 276 ページの『WebSphere Application Server でのカタログ・サービスの構成』
- 295 ページの『WebSphere Application Server のコンテナー・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ー・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**: コンテナー・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、296 ページの『コンテナー・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、276 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 でのコンテナー・サーバーの開始**

  コンテナー・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、コンテナー・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ー・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詳しくは、 442 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ー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・プロパティー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。コンテナー・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、コンテナー・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。この使用法では、本来は揮発性のデータ・ストアにスケーラビリティー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ーがあります。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ー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、複数のプロパティー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ーを直接設定できます。これらのプロパティーは、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ー・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関与するプロセス・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・プロパティーの構成が必要です。 Server クラスは、新規の Container インスタンスの作成に関与します。 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ー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ーを作成します。 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、使用可能にできます。

   ```
   DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(new
      URL("file://urltodeployment.xml"),
    new URL("file://urltoobjectgrid.xml"));
   Container container = server.createContainer(policy);
   ```

5. コンテナーを除去してクリーンアップします。

コンテナー・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。コンテナーで teardown メソッドを実行すると、コンテナー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ーを除去します。

コンテナーのクリーンアップ処理には、そのコンテナー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、多くのコンテナーと断片が含まれます。コンテナー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、このコンテナー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 でコンテナー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ー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ようにする方法を示しています。コンテナーの例と同様に、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ーを使用できますが、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ーは、サーバーの初期化に使用されます。

サーバー・プロパティーの設定

サーバー・プロパティーは、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ーが読み取られます。作成後にプロパティー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ー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 コンテナーの組み込み

JVM が複数の eXtreme Scale コンテナー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 コンテナー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、実動中にも実用的です。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、コンテナー・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ーを作成する前にプロパティー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ー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、444 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 ユーティリティーによる管理

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ー・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 ユーティリティーは、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・ユーティリティー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・ユーティリティー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** ユーティリティー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、234 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. 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、**startOgServer** 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。

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 コンテナー・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 コンテナーを開始する前に、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 コンテナー・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ー・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。

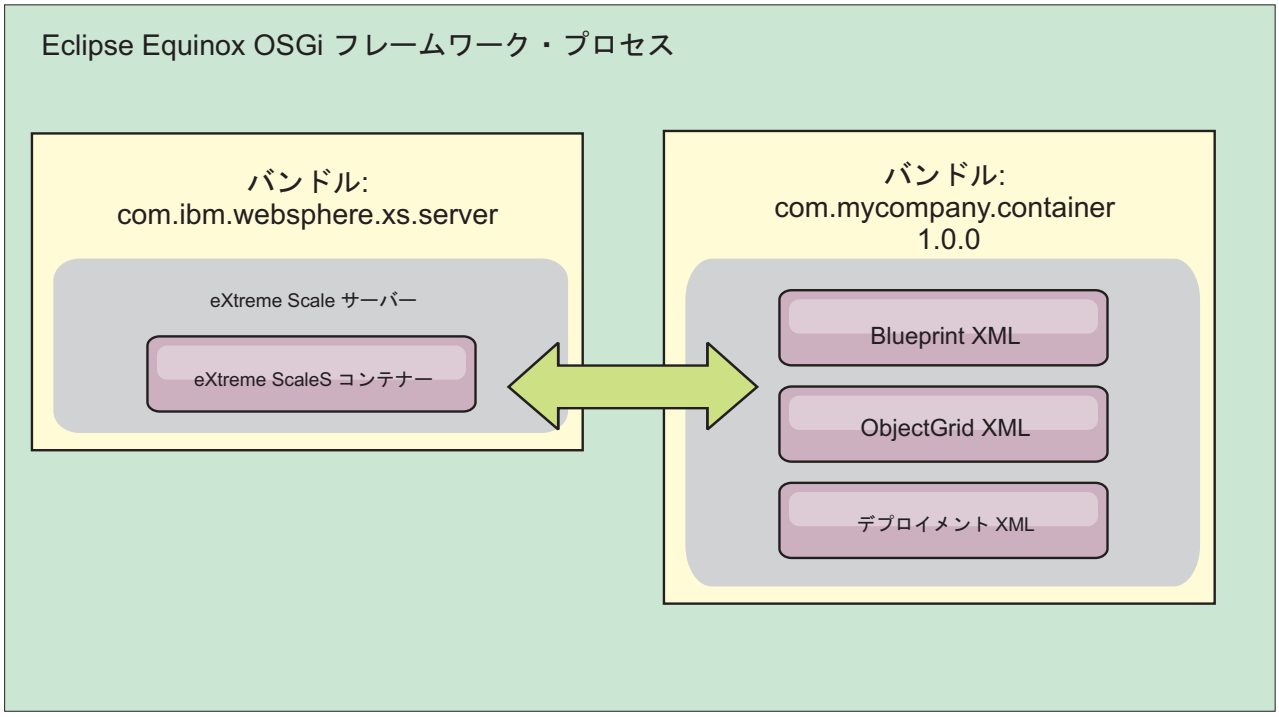


図 49. OSGi バンドルにすべての構成およびメタデータを含めるための Eclipse Equinox プロセス

- 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。

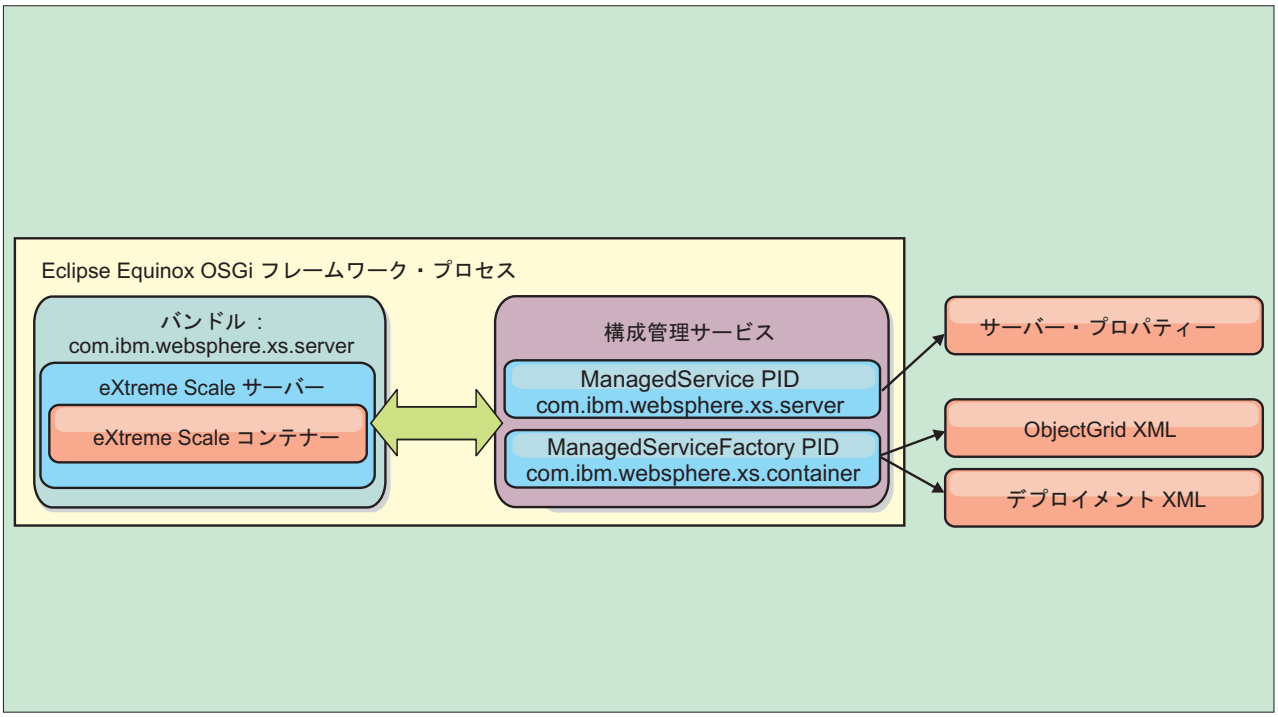


図 50. OSGi バンドルの外部で構成およびメタデータを指定するための Eclipse Equinox プロセス

- 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 つ以上のコンテナーが構成されます。

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・コンテナー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 コンテナー・サーバーを構成できます。サーバーおよびコンテナーの 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、コンテナー・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、ハンドルはコンテナーを停止します。

Blueprint XML で eXtreme Scale コンテナー・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 コンテナー・サーバーを始動します。
  1. コンテナー・バンドルを作成します。
  2. コンテナー・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. コンテナー・バンドルを開始します。
- OSGi Configuration Admin を使用して、eXtreme Scale コンテナー・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ーを構成します。
  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ーは自動的に始動します。
- ServerFactory API を使用して、eXtreme Scale コンテナー・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。 222 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わりの方式があります。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詳しくは、「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、 459 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>  install <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ブランク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 コンテナー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 ユーティリティーによる OSGi 対応サービスの管理

xscmd ユーティリティーを使用して、各コンテナー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ー・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** ユーティリティーを使用すると、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 コンテナー・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 <service name>;<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。1 つ以上のランキングが使用不可の場合は、ゼロ以外のエラー・レベルと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設定されます。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 <service name>;<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、464 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 コンテナーを更新する。
  - 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になります。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ー・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す。
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・コンテナー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、どのリビ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 コンテナー・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、各コンテナー・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ーに 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

CWXSI0040I: 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

CWXSI0040I: The command osgiCheck has completed successfully.
```

9. 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
```

10. 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
```

11. 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
```

12. eXtreme Scale コンテナー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。

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ー・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。すべてのコンテナー・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、後続のコンテナー・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。

- コンテナー・サーバーの開始時、断片のバ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。

  コンテナー・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・セットの断片のバランシング

を停止します。コンテナー・サーバーが開始されたら、**xscmd -c resumeBalancing** コマンドを使用して、コンテナー・サーバーでの断片の配置を開始できます。

- 7.1.1+ **placementDeferralInterval** プロパティーを構成する。

  placementDeferralInterval プロパティーは、コンテナー・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す。

  カタログ・サーバーのサーバー・プロパティー・ファイル内で **placementDeferralInterval** プロパティーを設定します。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。このプロパティーは、コンテナー・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。このプロパティーのデフォルト値は 15 秒です。デフォルト値では、コンテナー・サーバーが開始しても、プロパティー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。複数のコンテナー・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ー・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、最初のコンテナー・サーバーの 10 秒後に 2 番目のコンテナー・サーバーが開始すれば、2 番目のコンテナー・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 番目のコンテナー・サーバーの 20 秒後に 3 番目のコンテナー・サーバーが開始した場合、配置は既に最初の 2 つのコンテナー・サーバーで開始されています。

  コンテナー・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。

  - 複数のコンテナー・サーバーを同時に開始し、各コンテナー・サーバーの SystemOut.log ファイルで CWOBJ1001 メッセージを確認します。各コンテナー・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、コンテナー・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** プロパティーを調整して、この間隔により多くのコンテナー・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、最初のコンテナー・サーバーが最後のコンテナー・サーバーの 90 秒前に開始している場合、プロパティーを 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ー・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** プロパティーの値に関係なく、**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** プロパティーで指定された時間がまだ経過していないときや、バランシング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バランシングする。

  xscmd -c balanceShardTypes コマンドを使用して、構成内で実行中のコンテナー・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。各コンテナー・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・セットの断片のバランシングを停止または開始します。バランシング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。
  - コンテナー・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バランシング。

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。

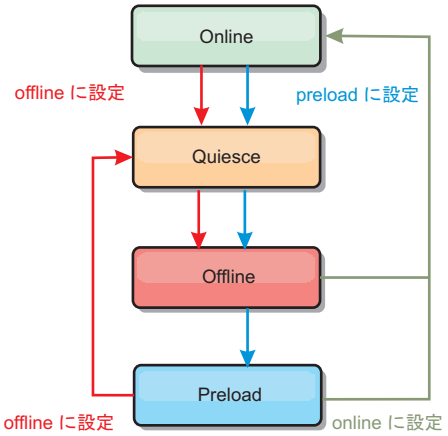


図 51. ObjectGrid インスタンスの可用性状態

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、遷移的状態です。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す。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 は遷移的状態であるため、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ので、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す。

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、コンテナー・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** ユーティリティーを使用して、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、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。
  - カタログ・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。

CWOBJ1254W: 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** ユーティリティー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** ユーティリティー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** ユーティリティー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、 442 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、すべてのコンテナー・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。一部のコンテナー・サーバーのキーが他のコンテナー・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
```

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ーを参照します。

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 ユーティリティーを使用して、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
- コンテナー 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 の各コンテナー・サーバーとコンテナー・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 52. 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 コンテナーの開始時、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。

– コンテナー・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 つのコンテナー・サーバーがアクティブなことがわかります。Container-0 コンテナー・サーバーは 4 つのプライマリー断片をホスティングしています。Container-1 コンテナー・サーバーは、Container-0 コンテナー・サーバー上の各プライマリー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 コンテナー・サーバーに残りの断片が残されています。さらに 2 つのコンテナー・サーバーが開始すると、Unassigned コンテナー・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>
```

- **コンテナー MBean サーバーへの接続**

コンテナー・サーバーは MBean をホスティングして、コンテナー・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ー・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 53. CollectContainerStatus.java

サンプル・プログラムが各コンテナーのコンテナー・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 54. 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 プロパティー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、558 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。

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。

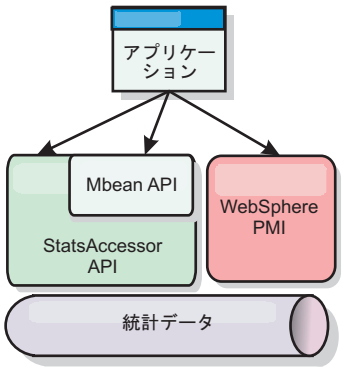


図 55. 統計の概説

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さえできます。詳しくは、500 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 トポロジー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・ユーティリティー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。

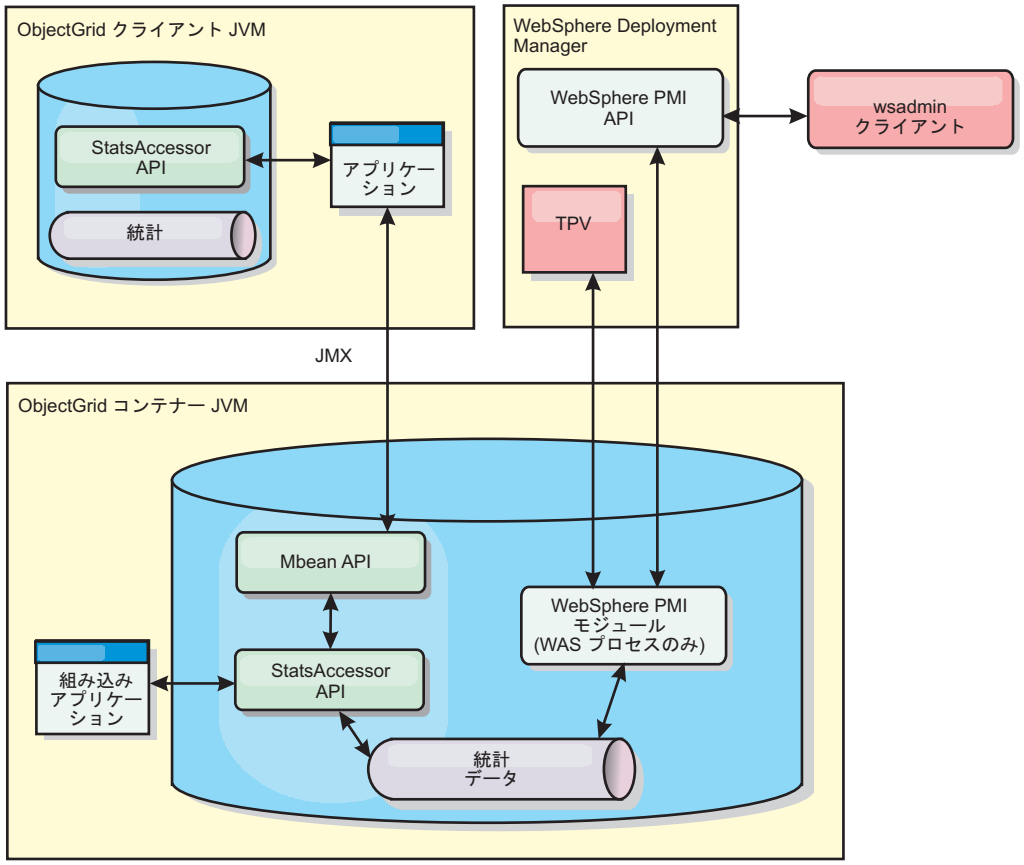


図 56. MBean の概説

詳しくは、サンプル: **xsadmin** ユーティリティー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
- 524 ページの『Hyperic HQ による eXtreme Scale のモニター』
- 521 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

- **システム要件**
  - AIX、Linux、または Windows システムを使用して、Web コンソールを実行します。
  - コンソール・サーバーをホストしているシステム上に、スタンドアロン WebSphere eXtreme Scale サーバーをインストールします。詳しくは、210 ページの『スタンドアロン 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。
  - コンソール・サーバー・システムは、カタログ・サービスに接続できる必要があります。また、逆に、カタログ・サービスは、Web コンソール・サーバーに接続できる必要があります。
- **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ーを使用します。
  - 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。次のプロパティー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、『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、479 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、427 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・プロパティー・ファイルを構成する必要があります。 サーバー・プロパティー・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・プロパティー・ファイルを編集して、SSL 構成のプロパティー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ーが必要です。

```
#--------------------------------------------------------------------------------
# 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}PDM2OjErLyg\=
#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、プロパティー・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、69 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) カタログ・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. カタログ・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。 カタログ・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、カタログ・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タログ・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. 新規カタログ・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、カタログ・サービス・ドメインを登録します。カタログ・サービス・ドメインの名前を入力します。
      2) カタログ・サービス・ドメインを作成した後、プロパティーを編集できます。カタログ・サービス・ドメインのプロパティーは次のとおりです。

         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。

         カタログ・サーバー
         選択したドメインに属する 1 つ以上のカタログ・サーバーをリストします。前のステップで作成したカタログ・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 **crendentialGeneratorClass** プロパティーが、指定した値でオーバーライドされます。

生成プログラム・プロパティー
CredentialGenerator 実装クラスのプロパティーを指定します。このプロパティーが、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 プロパティー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** プロパティーが、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・プロパティー・パス
前のステップで SSL プロパティーを含める編集をしたクライアント・プロパティー・ファイルへのパスを指定します。例えば、c:¥ObjectGridProperties¥sampleclient.properties ファイルを示します。コンソールが SSL 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、 479 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、481 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、490 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、使用メモリー、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。

- 485 ページの『データ・グリッド・ドメインの概要』
- 485 ページの『データ・グリッドの概説』
- 485 ページの『データ・グリッドの詳細』
- 486 ページの『サーバーの概要』
- 486 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』
  - 487 ページの『カスタム・レポート: コンテナー・サーバー統計』
  - 488 ページの『カスタム・レポート: データ・グリッド統計』
  - 489 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。トップ 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。トップ 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。トップ 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トップ 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: コンテナー・サーバー統計

カスタム・レポートを作成することで、コンテナー・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ーの数
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ーの数を表示します。最高の安定度を実現するために、60% のプロセッサー負荷でサーバー、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ー使用量を 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% のプロセッサー負荷でサーバー、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ー使用量を 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ーの数

このマップで使用可能なプロセッサーの数を表示します。最高の安定度を実現するために、60% のプロセッサー負荷でサーバー、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ー使用量を 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、およびコンテナー・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、479 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、481 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、個々のコンテナー・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・プロパティー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ーを指定してサーバー・プロパティー・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ーについて詳しくは、サーバー・プロパティー・ファイルを参照してください。
2. サーバーを再始動して、サーバー・プロパティー・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、492 ページの『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
JVM 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。

ServerName
JVM のサーバー名を指定します。

ホスト名
JVM のホスト名を指定します。

DomainName
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
JVM で使用可能なバイト数を指定します。

MaxMemory
JVM に割り振り可能な最大バイト数を指定します。

TotalMemory
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
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ーの数を表示します。最高の安定度を実現するために、60% のプロセッサー負荷でサーバー、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ー使用量を 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
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
マップの名前を指定します。

OgName
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。

PartitionId
区画 ID を指定します。

MapSetName
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
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 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
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
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
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
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
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
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
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
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
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
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
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

Count 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。

TimeStamp
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
データ・グリッドの名前を指定します。

PartitionId
区画 ID を指定します。

Hostname
ホスト名を指定します。

DomainName
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
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。

MinTime
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。

MeanTime
トランザクションにかかった平均時間を示します。

TotalTime
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。

AvgTransTime
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。

AvgThroughPut
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。

SumOfSquares
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
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。

LastCount
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
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
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 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す。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 ストリングの両方を使用して 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 ストリングを使用すれば、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

ーフェースであるため、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();

}
```

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。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。

概説

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。

xscmd ユーティリティーによるモニター

xscmd ユーティリティーは、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・ユーティリティーに取って代わります。**xscmd** ユーティリティーを使用すれば、WebSphere eXtreme Scale トポロジーのテキスト情報を表示できます。

始める前に

- **xscmd** ユーティリティーを使用して結果を表示するには、データ・グリッド・トポロジー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

ー・サーバーは、始動済みでなければなりません。詳しくは、 427 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** ユーティリティーの開始について詳しくは、 449 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。

このタスクについて

xscmd ユーティリティーを使用して、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・コンテナーを表示する方法と、*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** ユーティリティーを実行してください。

手順

xscmd ユーティリティーを使用して、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・コンテナー・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`

  すべてのコンテナー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ー・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 インターフェースで 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 ユーティリティーは、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含める場合、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す。

- 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、『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、 503 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、 470 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、サーバーを再始動します。
  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・サーバーを再始動すると、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、 501 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、505 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ー**」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 サーブレットに移動し、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・タイプを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。

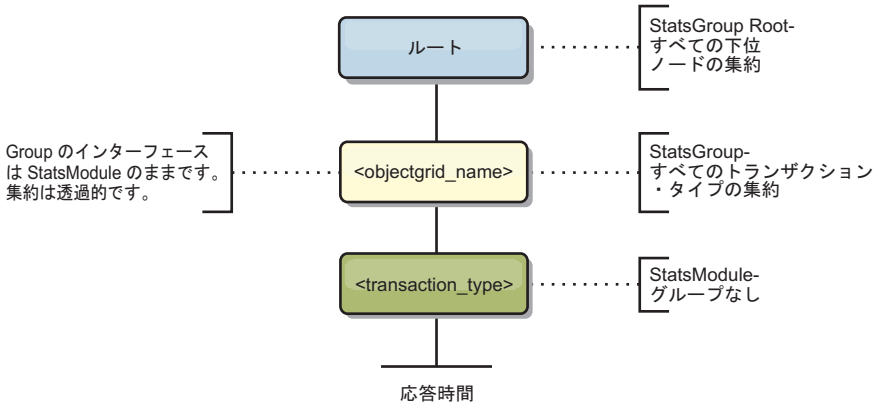


図 57. ObjectGridModule モジュールの構造

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。

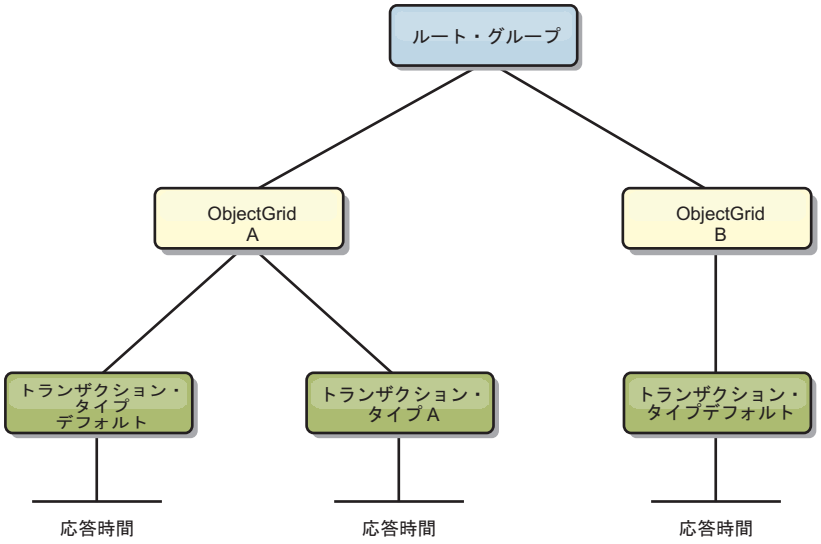


図 58. ObjectGridModule モジュール構造の例

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係するマップ・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。

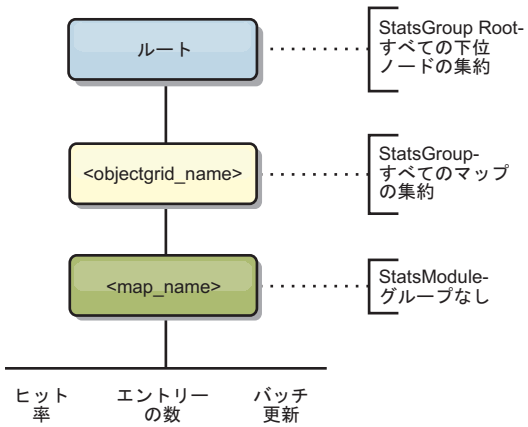
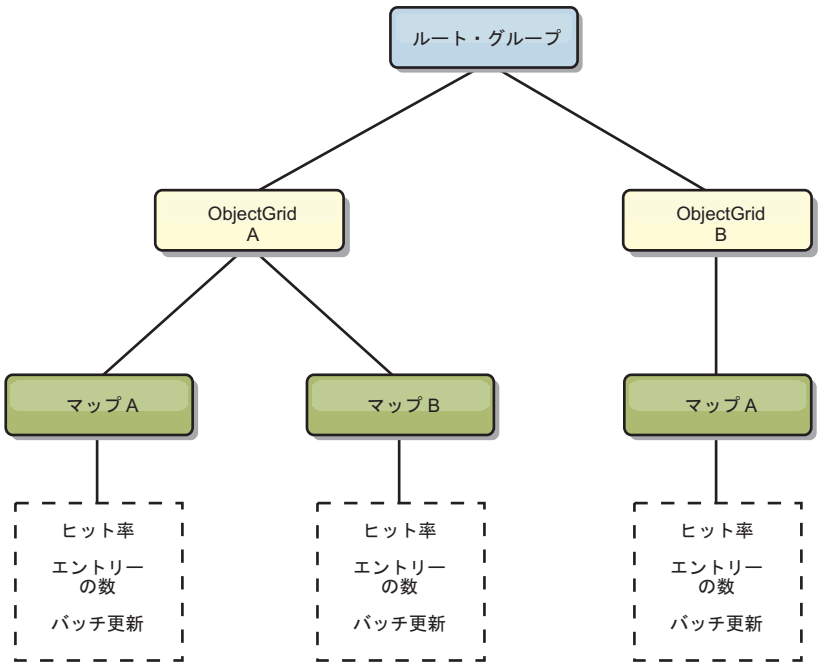


図 59. mapModule 構造

次の図は mapModule 構造の例です。

図 60. 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 HashIndexe 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。

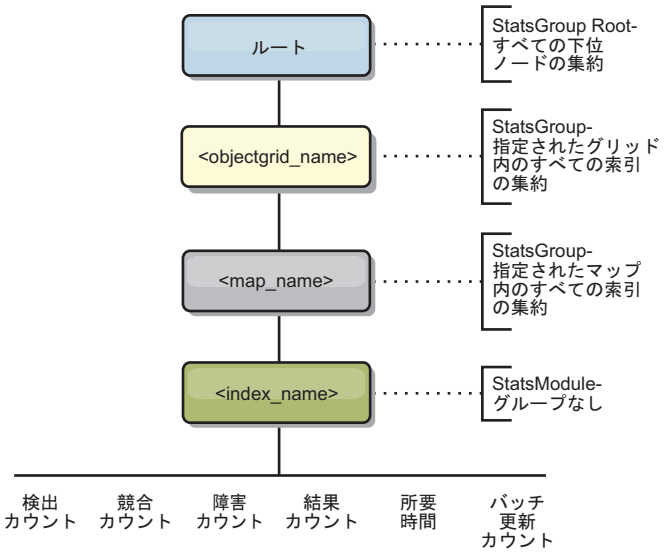


図 61. hashIndexModule モジュール構造

次の図は hashIndexModule 構造の例です。

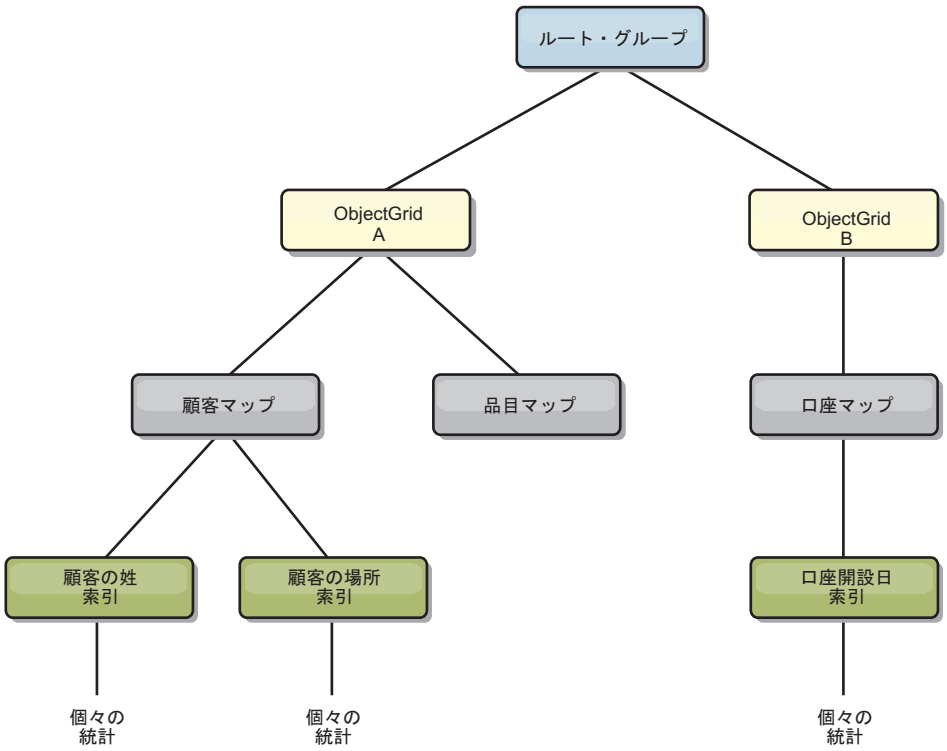


図 62. hashIndexModule モジュール構造の例

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。

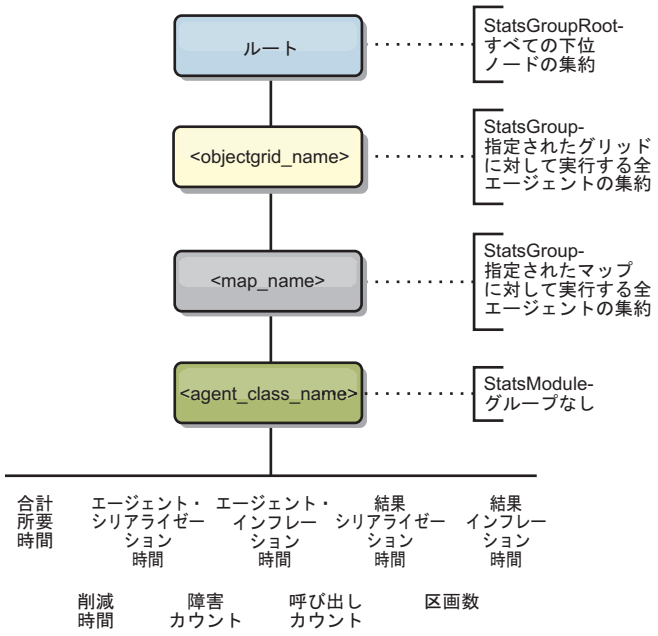


図 63. agentManagerModule 構造

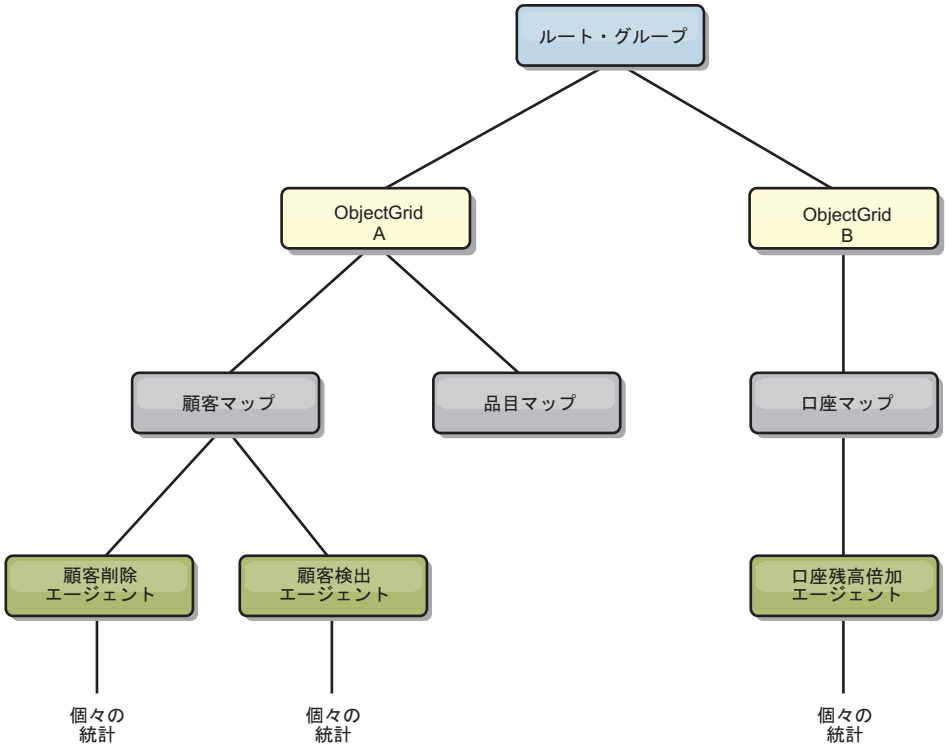


図 64. agentManagerModule 構造の例

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。

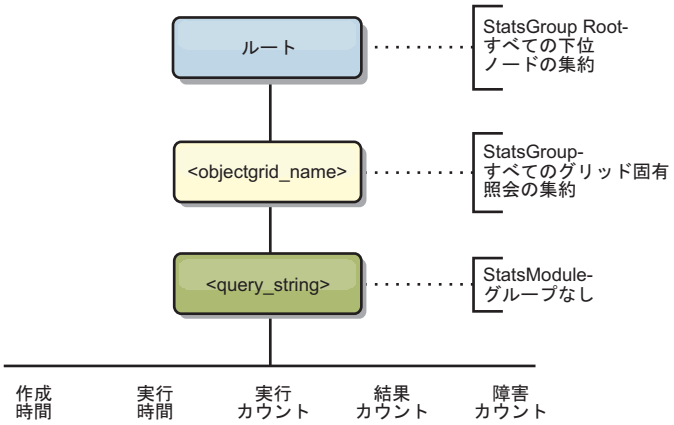


図 65. queryModule の構造

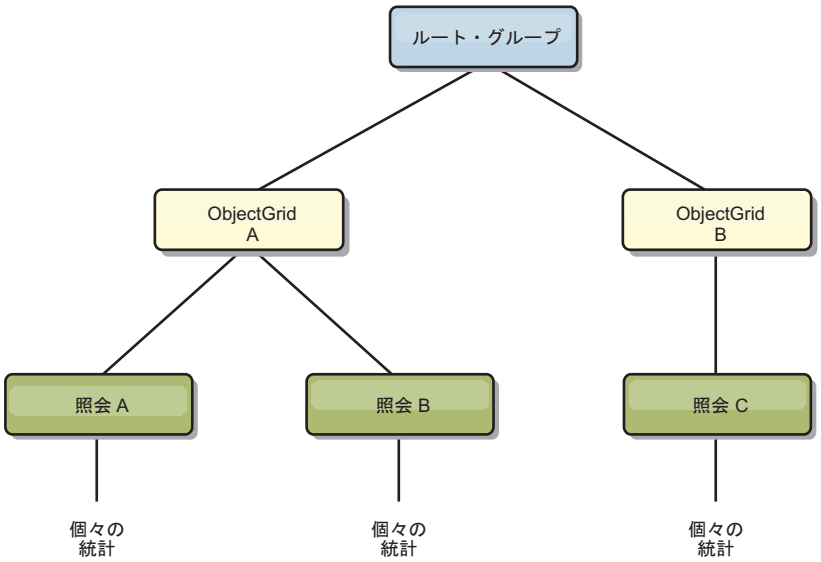


図 66. QueryStats.jpg queryModule 構造の例

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 ユーティリティーを使用して、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・プロパティー・ファイルを使用:**

  サーバー・プロパティー・ファイルの `statsSpec=<StatsSpec>` というキー値エントリー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・プロパティー・ファイルに関して詳しくは、サーバー・プロパティー・ファイルを参照してください。

- **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、 470 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、 471 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** ユーティリティー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 メソッドのインスツ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、 513 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** ユーティリティー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ーに解凍します。
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ーから、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ようにする必要があります。 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で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ー・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 コンテナー・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. コンテナー・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、コンテナー・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、コンテナー・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: コンテナー・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 コンテナーがあり、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ーからの出力は*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 # ./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):
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 コンテナー・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://localho
st: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。

5. すべての eXtreme Scale コンテナー・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 コンテナー・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。

a. "-JMXServicePort" "15000" (rhea00b02_c0 の場合)
b. "-JMXServicePort" "15001" (rhea00b02_c1 の場合)

a. 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。

   1) cd <ITM_install>/bin
   2) itmcmd agent –o inst1 start xt

b. 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。

   1) cd <ITM_install>/bin

2) `itmcmd agent –o inst1 stop xt`

6. すべての eXtreme Scale コンテナー・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・プロパティーを構成して、コンテナー・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・ユーティリティーで、 -setstatsspec all=enabled パラメーターを使用して、すべてのアクティブ・コンテナー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。

概説

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を 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、コンテナーを調整します。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、コンテナー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ーを除去するときに呼び出される 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 コンテナー・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` ディレクトリーにあります。

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、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、 433 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 `wxshyperic.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・ディレクトリーに置きます。 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` ディレクトリーに置きます。

      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` ディレクトリーに置きます。

      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` ディレクトリーに置きます。

      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。このプロパティーは、*agent_home*`/conf` ディレクトリーにあります。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・プロパティー・ファイルで、**statsSpec** プロパティーの情報を参照してください。これにより、コンテナー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、 513 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® 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、578 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 コンテナー・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。コンテナー・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。コンテナー・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・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:0000fffe
TcpTimedWaitDelay = dword:0000001e
```

Solaris

```
ndd -set /dev/tcp tcp_time_wait_interval 60000
fndd -set /dev/tcp tcp_keepalive_interval 15000
ndd -set /dev/tcp tcp_fin_wait_2_flush_interval 67500
ndd -set /dev/tcp tcp_conn_req_max_q 16384
ndd -set /dev/tcp tcp_conn_req_max_q0 16384
ndd -set /dev/tcp tcp_xmit_hiwat 400000
ndd -set /dev/tcp tcp_recv_hiwat 400000
ndd -set /dev/tcp tcp_cwnd_max 2097152
ndd -set /dev/tcp tcp_ip_abort_interval 20000
ndd -set /dev/tcp tcp_rexmit_interval_initial 4000
ndd -set /dev/tcp tcp_rexmit_interval_max 10000
ndd -set /dev/tcp tcp_rexmit_interval_min 3000
ndd -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 プロパティー

オブジェクト・リクエスト・ブローカー (ORB) プロパティーは、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ーは、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・プロパティー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 ディレクトリーにあります。WebSphere Application Server java/jre/lib ディレクトリーにある 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 プロパティーが更新されます。この動作を望まない場合は、カスタム・プロパティーまたは 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ー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・コンテナー・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ーについては、「**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、プロセス当たりに許容されるオープン・ファイルの数に制限があります。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
```

プロパティーの説明

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

プロパティー名: 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。このプロパティーは、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。このプロパティー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ーの値は慎重に考慮してください。

接続タイムアウト

プロパティー名: 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。このプロパティーは、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、このプロパティー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

プロパティー名: 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。このプロパティーは、要求タイムアウト・プロパティーと類似しています。

スレッド・プールの設定

このプロパティーは、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。このプロパティー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

プロパティー名: 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

プロパティー名:
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

プロパティー名: com.ibm.CORBA.ThreadPool.IsGrowable

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

プロパティー名: 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

プロパティー名: 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

プロパティー名: 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

プロパティー名: 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、セキュリティー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・プロシージャー・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 プロパティー値を false に設定してください。セキュリティー・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実行されるのを避けます。

7.1.1+ 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、 299 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、コンピューターで使用可能なヒープ・サイズおよびプロセッサー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ー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 **startOgServer** 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: Sun JVM を使用する場合は、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場合があります。

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 以降をサポートしています。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 5

またはそれ以降を使用して、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。

WebSphere eXtreme Scale が使用されるほとんどのシナリオでは、JVM の Java SE バージョン 6 のほうが、Edition 5 よりうまく機能します。最良のパフォーマンスのためには、一般に最新バージョンの Java Platform, Standard Edition を使用します。

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 フォルダーにあります。

```
# IBM JDK properties for ORB
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.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。

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、**startOgServer** 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 30. 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ーは、WebSphere 管理コンソールで、コア・グループに対してカスタム・プロパティー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・プロパティー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、以下のプロパティー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・プロパティー (秒単位) または IBM_CS_FD_PERIOD_MILLIS カスタム・プロパティー (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・プロパティーを使用して指定します。

    IBM_CS_FD_PERIOD_SEC プロパティーのデフォルト値は 20 で、IBM_CS_FD_CONSECUTIVE_MISSED プロパティーのデフォルト値は 10 です。IBM_CS_FD_PERIOD_MILLIS プロパティーを指定すると、設定されている IBM_CS_FD_PERIOD_SEC カスタム・プロパティーがオーバーライドされます。これらのプロパティーの値は、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 でのコア・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。

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ー使用量を正しく設定する。

  WebSphere Real Time にはプロセッサー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ーの `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ー容量を提供する値) 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ー負荷を 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ーを開始する際、**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ーを開始します。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・プロシージャー・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ー・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、 534 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、539 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ー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 31. 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
なし	常になし	使用不可
なし	サポートされる	使用不可
なし	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す。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詳しくは、*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と、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ライズ済みフォームでエンコードし、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ため、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トランスポートする必要があるため、トランスポート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ため、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。

Sun 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。

Sun 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、Sun JDK でトランスポート・セキュリティーは使用可能になりません。この処理を行うためには、WebSphere Application Server に同梱されている Sun JDK が必要です。 WebSphere Application Server に同梱された Sun 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、313 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・チュートリアルのステップ 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、コンテナー・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、566 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 32. 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 プロパティー	サーバーの transportType プロパティー	結果のプロトコル
TCP/IP	TCP/IP	TCP/IP
TCP/IP	SSL サポート	TCP/IP

表 32. 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 (続き):

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 プロパティー	サーバーの transportType プロパティー	結果のプロトコル
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・セキュリティー構成に **transportType** プロパティーを設定する方法については、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。
2. コンテナーおよびカタログ・サーバー・セキュリティー構成に **transportType** プロパティーを設定する方法については、サーバー・プロパティー・ファイルを参照してください。

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、このプロパティー・ファイルを JVM クライアント・プロパティー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。プロパティー・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。特定のプロパティー・ファイルを使用してコンテナー・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、セキュリティー・サーバー・プロパティー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 に関する詳細は、 470 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ー・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 がクライアントをどのように認証するのかについて詳しくは、 79 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』「*製品概要*」でセキュリティーに関するチュートリアル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ー・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 JMXCconnectorServer
    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。そのようなスレッドが、JMX 仕様操作を呼び出す場合、これらは、保持許可と呼ばれています。 JMX 操作が実行されると、セキュリティー・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がチェックされます。

MBean ポリシー定義は、Java ポリシー形式に従います。例えば、以下のポリシーでは、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ていますが、com.ibm.websphere.objectgrid ドメインに対しては制限しています。

```
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ー・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ー・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・プロパティーを使用して、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 クライアントに構成する必要のある必要があります。

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、次のプロパティーを使用します。

1. -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE

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ー・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 インターフェースで setJMXServicePort メソッドを呼び出すことができます。

コンテナー・サーバーで JMX セキュア・トランスポートを使用可能にするには、JMX サービス・ポートを設定する必要があります。Transport Layer Security/Secure Sockets Layer (TLS/SSL) を使用し、カタログ・サーバーからコンテナー・サーバー情報を表示したい場合、JMX サービス・ポートの設定は必須です。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。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。

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。

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ー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、 561 ページの『外部プロバイダーとのセキュリティー統合』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ーを構成します。

   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・セキュリティー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ーを構成します。

   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、セキュリティー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 プロパティーおよび loginType プロパティーの詳細については、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。

- 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。

  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ー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ー・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ー・ファイル内で、wxsRestAccessRightsFile プロパティーを使用して指定します。このプロパティーについて詳しくは、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。このファイルは通常、キーと値のペアが含まれた Java プロパティー・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。

  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ー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
     ```

     説明:

     - 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ーに対して、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. エンティティー・アクセス権限を構成します。

   エンティティー・セット権限は、REST サービスで公開される特定の ObjectGrid エンティティー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 エンティティー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`*

   説明:

   – *entity_set_right* には、以下の権限のいずれかを指定できます。

表 33. エンティティー・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ーの名前です。

   注: それぞれの ObjectGrid およびそのエンティティーに対してサービス操作権限とエンティティー・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、エンティティー・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 エンティティーには、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ーに対して、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) エンティティー・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 エンティティーのアクセス権限は NONE になります。

- トランスポートを保護します。

  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ー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スポート・セキュリティーが提供されます。

  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。 この接続のトランスポート・セキュリティーは、eXtreme Scale ではなくホスティング・コンテナー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。 この接続のトランスポート・セキュリティーは、eXtreme Scale で構成します。 555 ページの『トランスポート層セキュリティーおよび Secure Sockets Layer』を参照してください。

WebSphere Application Server とのセキュリティー統合

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スポート層セキュリティー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ー・ドメインで稼働中の場合、 WebSphere Application Server セキュリティー・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、同じセキュリティー・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 セキュリティー・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。

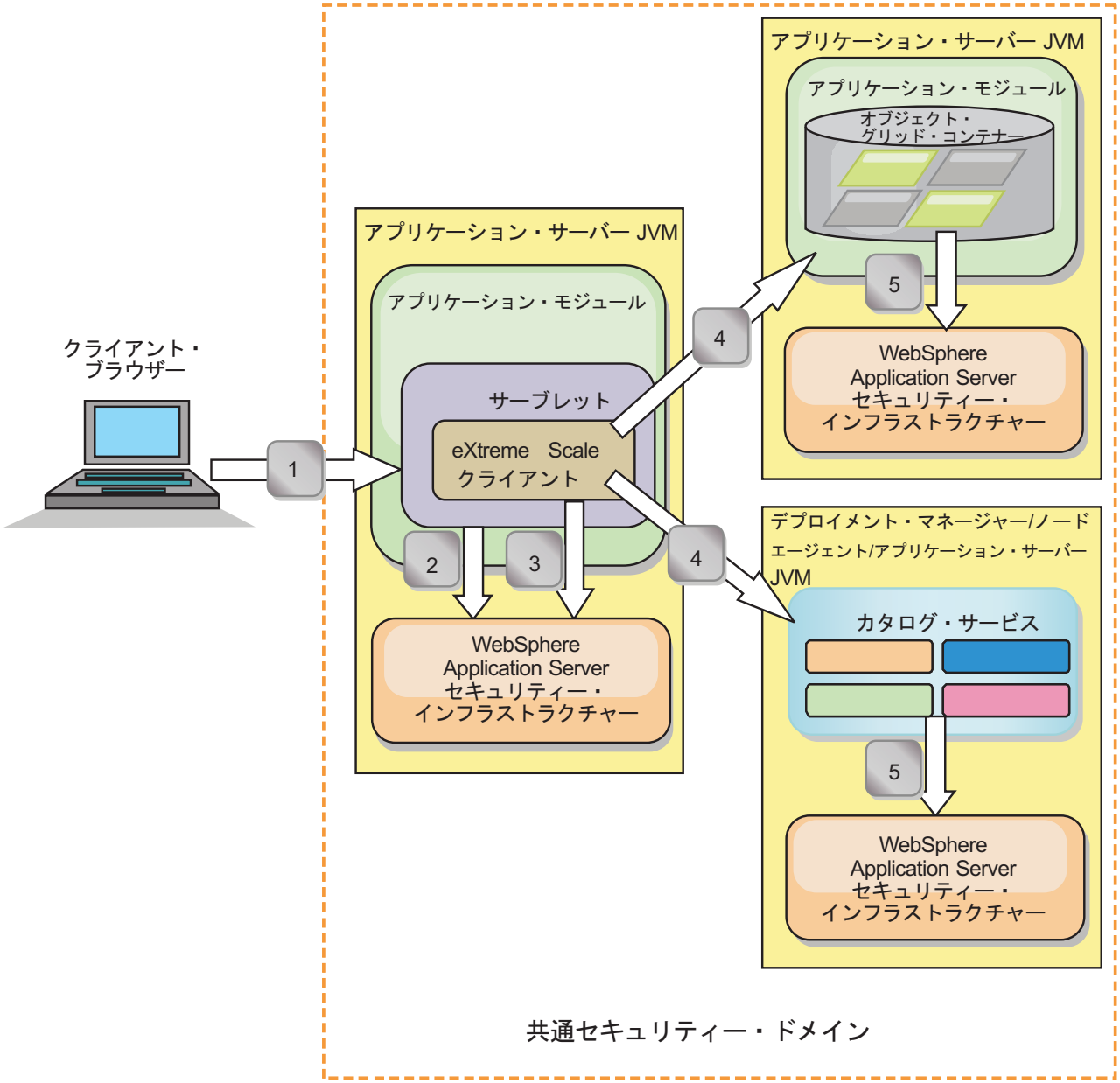


図 67. 同じセキュリティー・ドメイン内のサーバーの認証フロー

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ー・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、コンテナー・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 セキュリティー・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・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、セキュリティー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ー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ー・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ー・トークンを、他の JVM でホストされているコンテナー・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ー・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ー・サーバーは、セキュリティー・トークンをユーザーのセキュリティー・トークン情報に変換するために、セキュリティー・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・セキュリティー情報は、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、およびコンテナー・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ー統合:

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、95 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、WSTokenAuthentication 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・セキュリティー統合:

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・セキュリティー・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、コンテナー・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ー・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、95 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。
- カタログ・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、277 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、293 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、328 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">
    ...
</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ー・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・コンテナー・サーバーを始動します。

  セキュア・コンテナー・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・プロパティー・ファイル:** サーバー・プロパティー・ファイルは、サーバーに固有のセキュリティー・プロパティーを構成します。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。

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・ファイルへのパスを指定します。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、`../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。

  – **セキュリティー記述子 XML ファイル:** セキュリティー記述子 XML ファイルは、カタログ・サーバーおよびコンテナー・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ー・プロパティーを記述します。プロパティーの例の 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・プロパティー・ファイル:** サーバー・プロパティー・ファイルは、サーバーに固有のセキュリティー・プロパティーを構成します。

  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、セキュリティー記述子 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・ファイルへのパスを指定します。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。

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ー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ー情報が含まれます。

– -Dobjectgrid.cluster.security.xml.url: カタログ・サーバーおよびコンテナー・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/catalog.server.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・コンテナー・サーバーを始動します。

  コンテナー・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、特定のコンテナー・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ー・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ー・サーバーを停止します。
セキュア・サーバーの停止には、次のセキュリティー構成ファイルが必要です。

- **クライアント・プロパティー・ファイル:** クライアント・プロパティー・ファイルを使用して、クライアント・セキュリティー・プロパティーを構成できます。クライアント・セキュリティー・プロパティーは、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・プロパティー・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。このプロパティー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 ユーティリティーのためのセキュリティー・プロファイルの構成

セキュリティー・プロファイルを作成すると、保存されたセキュリティー・パラメーターを使用して、セキュアな環境で **xscmd** ユーティリティーを使用できます。

始める前に

xscmd ユーティリティーのセットアップについて詳しくは、449 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。

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、セキュリティー・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 ユーティリティー内の **ProfileManagement** コマンド・グループには、セキュリティー・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

- セキュリティー・プロファイルを保存します。

  セキュリティー・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 ディレクトリーに保存されます。

- 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*

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ー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** ユーティリティーを使用し、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。

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。コンテナー・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ーにあります。*wxs_home*/bin ディレクトリー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 ディレクトリー内の logs/<*server_name*> ディレクトリーにあります。

  – **カスタムのログの場所:** コンテナー・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ー・ファイル (server.properties など) を作成します。

  ```
  workingDirectory=<directory>
  traceSpec=
  systemStreamToFileEnabled=true
  ```

  workingDirectory プロパティーは、ログおよびオプションのトレース・ファイルのルート・ディレクトリーです。WebSphere eXtreme Scale は、コンテナー・サーバーの名前を持つディレクトリーを作成し、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。コンテナー開始中にプロパティー・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・プロパティー・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ーに存在し、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。

CWOBJ1001I: 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、580 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ー・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ー・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ー・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、433 ページの『**startOgServer** スクリプト』を参照してください。

- **プロパティー・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ー・サーバーでトレースを収集します。**

  プロパティー・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 プロパティーは、ログおよびオプションのトレース・ファイルのルート・ディレクトリーです。**workingDirectory** 値が設定されていなければ、デフォルトの作業ディレクトリーは、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ー・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・プロパティー・ファイルの場所を指定します。サーバー・プロパティー・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・プロパティー・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・プロパティー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 ユーティリティーを使用して、コンテナー・サーバーでトレースを収集します。**

  xscmd ユーティリティー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** ユーティリティーを使用して、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ー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 トレースを収集にするには、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。

7.1.1+ **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。

7.1.1+ **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。

7.1.1+ **ObjectGridSerializer**
DataSerializer プラグイン・インフラストラクチャー。

ObjectGridStats
ObjectGrid 統計。

ObjectGridStreamQuery
ストリーム照会 API。

7.1.1+ **ObjectGridTransactionManager**
WebSphere eXtreme Scale トランザクション・マネージャー。

ObjectGridWriteBehind
ObjectGrid 後書き。

7.1.1+ **ObjectGridXM**
一般 IBM eXtremeMemory トレース。

7.1.1+ **ObjectGridXMEviction**
eXtremeMemory 除去トレース。

7.1.1+ **ObjectGridXMTransport**
eXtremeMemory 一般トランスポート・トレース。

7.1.1+ **ObjectGridXMTransportInbound**
eXtremeMemory インバウンド特定のトランスポート・トレース。

7.1.1+ **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。

7.1.1+ **TCPChannel**
IBM eXtremeIO TCP/IP チャネル。

7.1.1+ **XsByteBuffer**
WebSphere eXtreme Scale バイト・バッファー・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
  - ロギングの頻度
  - 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、プロセッサー使用量などの統計
- **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、プロセッサー使用量などの統計
- 現在のフィックスパック・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、577 ページの『ロギング可能化』と 578 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・ファイルを入手できます。
  - 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・ファイルを入手できます。
  - 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ー・ファイルと構成ファイルを作成してください。詳しくは、585 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:\myxslogs -outDir c:\myxslogs\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、587 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ー・ファイルを作成します。
   a. プロパティー・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・ファイルの名前の一部にすると (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)
```

この正規表現によって、ストリング xception の前か後にストリング rror が出現すると、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)
```

この正規表現によって、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、イベントが記録されます。

2. **xsLogAnalyer** 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_name*Scanner.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_name*Scanner.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、 583 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 -Xmx`*`HEAP_SIZE`*`m`

  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い理由は、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。

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

- 7.1.1+ **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` プロパティーは、`true` に設定する必要があります。
  - `HttpSessionIdReuse` カスタム・プロパティー値は、`true` に設定する必要があります。このカスタム・プロパティーは、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・プロパティー**」
- 「**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・プロパティー**」
- 「**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・プロパティー**」
- 「**サーバー**」 > 「**サーバー・タイプ**」 > 「**WebSphere Application Server**」 > 「*server_name*」 > 「**Web コンテナー設定**」 > 「**Web コンテナー**」

カスタム・プロパティー値を更新した場合、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ー・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 つのエンティティー・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 があったとしても、必要な 5 つのエンティティー 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。

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は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 199 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。

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:** 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・プロパティーを値 2 (CLOSE_CURSORS_AT_COMMIT) で構成します。このプロパティーにより、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・プロパティーの値を 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・プロパティーを設定します: 「**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・プロパティー**」 > 「**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・プロパティーを設定し、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・プロパティー**」 > 「**新規**」。 webSphereDefaultIsolationLevel カスタム・プロパティー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、コンテナー・サーバー内の区画に対して、次の CWOBJ1511 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 プロパティーで指定された区画をアクティブにするコンテナー・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 プロパティーを `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ー・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。

```
CWOBJ3179E: 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。

```
CWOBJ3178E: 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 エレメントの authorizationMechanism 属性
  - backingMap エレメントの copyMode 属性
  - backingMap エレメントの lockStrategy 属性
  - backingMap エレメントの ttlEvictorType 属性
  - property エレメントの type 属性
  - objectGrid エレメントの initialState
  - backingMap エレメントの 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。

セキュリティーのトラブルシューティング

この情報を使用して、セキュリティー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、コンテナー・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: 556 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ー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、コンテナー・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ーで、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)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。

〒242-8502
神奈川県大和市下鶴間1623番14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 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、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

- AIX
- CICS®
- cloudscape
- DB2
- Domino®
- IBM
- Lotus®
- RACF®
- Redbooks®
- Tivoli
- WebSphere
- z/OS®

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]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

A

C

D

E

H

I

J

M

O

P

Q

R

S

T

W

X

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21