IBM WebSphere eXtreme Scale
Version 7.1.1

Administration Guide

December 2013

This edition applies to version 7, release 1, modification 1, of WebSphere eXtreme Scale and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures vii

Tables ix

About the *Administration Guide* xi

Chapter 1. Getting started 1

Tutorial: Getting started with WebSphere eXtreme Scale 1
- Getting started tutorial lesson 1.1: Defining data grids with configuration files 1
- Getting started tutorial module 2: Create a client application 3
- Module 3: Running the sample application in the data grid 5
- Getting started tutorial lesson 4: Monitor your environment 7

Chapter 2. Planning 13

Planning overview 13
Planning the topology 14
- Local in-memory cache 14
- Peer-replicated local cache 15
- Embedded cache 17
- Distributed cache 19
- Database integration: Write-behind, in-line, and side caching 21
- Planning multiple data center topologies . . . 35

Interoperability with other products 49
Planning for configuration 51
- Planning for network ports 51
- Planning to use IBM eXtremeMemory 55
- Security overview 56

Planning for installation 59
- Hardware and software requirements. 59
- Java SE considerations. 60
- Java EE considerations. 62
- Directory conventions 63

Planning environment capacity 65
- Sizing memory and partition count calculation 65
- Sizing CPU per partition for transactions . . . 67
- Sizing CPUs for parallel transactions 67
- Dynamic cache capacity planning 68

Chapter 3. Tutorials 73

Tutorial: Querying a local in-memory data grid . . 73
- ObjectQuery tutorial - step 1. 73
- ObjectQuery tutorial - step 2. 74
- ObjectQuery tutorial - step 3. 75
- ObjectQuery tutorial - step 4. 77

Tutorial: Storing order information in entities . . . 81
- Entity manager tutorial: Creating an entity class 81
- Entity manager tutorial: Forming entity relationships 82
- Entity manager tutorial: Order Entity Schema . . 84
- Entity manager tutorial: Updating entries . . . 88
- Entity manager tutorial: Updating and removing entries with an index 88
- Entity manager tutorial: Updating and removing entries by using a query 89

Tutorial: Configuring Java SE security 90
- Java SE security tutorial - Step 1 90
- Java SE security tutorial - Step 2 92
- Java SE security tutorial - Step 3 93
- Java SE security tutorial - Step 4 95
- Java SE security tutorial - Step 5 98
- Java SE security tutorial - Step 6 102

Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server . . . 106
- Introduction: Integrate WebSphere eXtreme Scale security with WebSphere Application Server using the WebSphere Application Server Authentication plug-ins 106
- Module 1: Prepare WebSphere Application Server 107
- Module 2: Configure WebSphere eXtreme Scale to use WebSphere Application Server Authentication plug-ins 112
- Module 3: Configure transport security. . . . 119
- Module 4: Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server. 122
- Module 5: Use the **xscmd** tool to monitor data grids and maps. 127

Tutorial: Integrate WebSphere eXtreme Scale security in a mixed environment with an external authenticator 128
- Introduction: Security in a mixed environment 129
- Module 1: Prepare the mixed WebSphere Application Server and stand-alone environment 130
- Module 2: Configure WebSphere eXtreme Scale authentication in a mixed environment 135
- Module 3: Configure transport security. . . . 145
- Module 4: Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server. 148
- Module 5: Use the **xscmd** utility to monitor data grids and maps. 151

Tutorial: Running eXtreme Scale bundles in the OSGi framework 153
- Introduction: Starting and configuring the eXtreme Scale server and container to run plug-ins in the OSGi framework 154
- Module 1: Preparing to install and configure eXtreme Scale server bundles 155
- Module 2: Installing and starting eXtreme Scale bundles in the OSGi framework 159
- Module 3: Running the eXtreme Scale sample client 164

Module 4: Querying and upgrading the sample bundle 166

Chapter 4. Installing 171

Installation overview 171
Planning for installation 172
Installation topologies 172
Hardware and software requirements 177
Java SE considerations 179
Java EE considerations 180
Directory conventions 181
Installing WebSphere eXtreme Scale with the installation wizard. 183
Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in a stand-alone environment 184
Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server. 186
Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with silent mode 217
Response file for silent installation 219
Installing the REST data service 220
Installing the Eclipse Equinox OSGi framework with Eclipse Gemini for clients and servers . . . 223
Installing eXtreme Scale bundles 225
Verifying the installation 227
Taking the first steps after installation 228
Troubleshooting installation 228
Uninstalling WebSphere eXtreme Scale 230
Customizing WebSphere eXtreme Scale for z/OS 231
Installing the WebSphere Customization Toolbox 231
Generating customization definitions 232
Uploading and running customized jobs . . . 233

Chapter 5. Upgrading and migrating WebSphere eXtreme Scale. 235

Updating eXtreme Scale servers 235
Migrating to WebSphere eXtreme Scale Version 7.1.1 238
Installing fix packs using IBM Installation Manager 239
Installing fix packs using the GUI 239
Installing fix packs using the command line . . 241
Installing fix packs using a response file . . . 243
Uninstalling fix packs using IBM Installation Manager 245
Uninstalling fix packs using the GUI 245
Uninstalling fix packs using the command line 245
Uninstalling fix packs using response files. . . 247
Using the Update Installer to install maintenance packages 248
`xsadmin` tool to `xscmd` tool migration. 249
Deprecated properties and APIs 253
Removed properties and APIs 255

Chapter 6. Configuring 257

Configuration methods 257
Operational checklist 259
Configuring data grids 261
Configuring local deployments 261
Configuring evictors with XML files. 263
Configuring a locking strategy in the ObjectGrid descriptor XML file 264
Configuring the lock timeout value in the ObjectGrid descriptor XML file 266
Configuring peer-to-peer replication with JMS 267
Configuring dynamic maps. 275
Configuring deployment policies 277
Configuring distributed deployments 278
Controlling shard placement with zones . . . 280
Configuring catalog and container servers 293
Configuring catalog servers and catalog service domains 294
Configuring container servers 324
Configuring multiple data center topologies . . . 329
Configuring ports 334
Configuring ports in stand-alone mode. . . . 334
Configuring ports in a WebSphere Application Server environment 337
Servers with multiple network cards 338
Configuring transports 339
Configuring IBM eXtremeIO (XIO) 340
Configuring Object Request Brokers. 341
Configuring IBM eXtremeMemory 346
Configuring Java clients 347
Java client overrides 347
Configuring Java clients with an XML configuration 348
Configuring the REST gateway with an XML configuration 350
Configuring Java clients programmatically . . 352
Configuring the near cache 353
Configuring an evictor for the near cache . . . 354
Configuring Java Message Service (JMS)-based client synchronization 355
Configuring request and retry timeout values 357
Configuring eXtreme Scale connection factories . . 362
Configuring Eclipse environments to use eXtreme Scale connection factories 363
Configuring cache integration 364
Configuring HTTP session managers 364
Configuring the dynamic cache provider for WebSphere eXtreme Scale 396
JPA level 2 (L2) cache plug-in 401
Configuring database integration 419
Configuring JPA loaders 419
Configuring REST data services 423
Enabling the REST data service 424
Configuring application servers for the REST data service 432
Configuring Web browsers to access REST data service ATOM feeds 446
Using a Java client with REST data services . . 448
Visual Studio 2008 WCF client with REST data service. 450
Configuring servers for OSGi 451
Configuring eXtreme Scale plug-ins with OSGi Blueprint 452
Configuring servers with OSGi Blueprint . . . 454
Configuring servers with OSGI config admin 456

Chapter 7. Administering 459
Starting and stopping stand-alone servers 459
  Starting stand-alone servers 460
  Stopping stand-alone servers 472
Stopping servers gracefully with the `xscmd` utility 474
Starting and stopping servers in a WebSphere Application Server environment 475
Using the embedded server API to start and stop servers 476
  Embedded server API 479
Administering with the `xscmd` utility 482
Controlling placement 486
Managing ObjectGrid availability. 488
Managing data center failures 491
  Managing data center failures when quorum is enabled 491
Starting eXtreme Scale servers using the Eclipse Equinox OSGi framework 493
Installing and starting OSGi-enabled plug-ins . . 496
Administering OSGi-enabled services using the `xscmd` utility 497
  Updating OSGi services for eXtreme Scale plug-ins with `xscmd` 500
Administering with Managed Beans (MBeans) . . 503
  Accessing Managed Beans (MBeans) using the wsadmin tool 504
  Accessing Managed Beans (MBeans) programmatically 504

Chapter 8. Monitoring 511
Statistics overview. 511
Monitoring with the web console. 514
  Starting and logging on to the web console . . 514
  Connecting the web console to catalog servers 516
  Viewing statistics with the web console. . . . 518
  Monitoring with custom reports 525
Monitoring with CSV files 525
  CSV file statistics definitions 526
Enabling statistics 529
  Statistics modules 530
  Monitoring with the statistics API 532
Monitoring with the `xscmd` utility. 535
Monitoring with WebSphere Application Server PMI 537
  Enabling PMI 537
  Retrieving PMI statistics. 539
  PMI modules 541
  Accessing Managed Beans (MBeans) using the wsadmin tool 548
Monitoring server statistics with managed beans (MBeans). 549
Monitoring with vendor tools 550
  Monitoring with the IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale . 551
  Monitoring eXtreme Scale applications with CA Wily Introscope. 557
  Monitoring eXtreme Scale with Hyperic HQ . . 560
Monitoring eXtreme Scale information in DB2 . . 562

Chapter 9. Tuning performance. . . . 565
Tuning operating systems and network settings 565
ORB properties. 566
Tuning IBM eXtremeIO (XIO) 570
Tuning Java virtual machines 571
Tuning the heartbeat interval setting for failover detection 574
Tuning garbage collection with WebSphere Real Time 577
  WebSphere Real Time in a stand-alone environment. 578
  WebSphere Real Time in WebSphere Application Server 580
Tuning the dynamic cache provider 581

Chapter 10. Security 583
Data grid authentication. 583
Data grid security 584
Authenticating and authorizing clients 586
  Authenticating application clients 586
  Authorizing application clients 589
Configuring secure transport types 592
  Transport layer security and secure sockets layer 593
  Configuring Secure Sockets Layer (SSL) parameters for clients or servers 594
Java Management Extensions (JMX) security . . . 595
Security integration with external providers . . . 597
Securing the REST data service 598
Security integration with WebSphere Application Server 601
  Configuring client security on a catalog service domain 604
Enabling data grid authorization 605
Starting and stopping secure servers. 606
  Starting secure servers in a stand-alone environment. 606
  Starting secure servers in WebSphere Application Server 607
  Stopping secure servers 607
Configuring security profiles for the `xscmd` utility 607
Securing J2C client connections 609

Chapter 11. Troubleshooting 611
Troubleshooting and support for WebSphere eXtreme Scale 611
  Techniques for troubleshooting problems . . . 611
  Searching knowledge bases. 613
  Getting fixes. 614
  Contacting IBM Support. 615
  Exchanging information with IBM 616
  Subscribing to Support updates 617
Enabling logging 618
Collecting trace. 619
  Server trace options 621
Analyzing log and trace data 623
  Log analysis overview 623
  Running log analysis 624
  Creating custom scanners for log analysis . . . 625
  Troubleshooting log analysis 626
Troubleshooting installation 627

Troubleshooting cache integration 628
Troubleshooting the JPA cache plug-in 629
Troubleshooting IBM eXtremeMemory 630
Troubleshooting administration 631
Troubleshooting multiple data center configurations 632
Troubleshooting loaders 633
Troubleshooting XML configuration 635
Troubleshooting security. 638
IBM Support Assistant for WebSphere eXtreme Scale 638

Notices 641

Trademarks 643

Index 645

Figures

1. Local in-memory cache scenario. 15
2. Peer-replicated cache with changes that are propagated with JMS 16
3. Peer-replicated cache with changes that are propagated with the high availability manager. 16
4. Embedded cache 18
5. Distributed cache 19
6. Near cache 20
7. ObjectGrid as a database buffer 21
8. ObjectGrid as a side cache 22
9. Side cache 23
10. In-line cache 24
11. Read-through caching 25
12. Write-through caching 25
13. Write-behind caching 26
14. Write-behind caching 27
15. Loader 28
16. Loader plug-in 30
17. Client loader 31
18. Periodic refresh 32
19. Order Schema. 78
20. Order Entity Schema 85
21. Tutorial topology 109
22. Tutorial topology 132
23. Authentication flow 136
24. Development node. 172
25. Stand-alone topology with two data centers 173
26. WebSphere Application Server topology example 174
27. Mixed topology example. 175
28. WebSphere eXtreme Scale REST Data Service Files 222
29. Enable TimeToLive evictor with XML 263
30. Plugging in an evictor using XML 264
31. Primaries and replicas in zones 285
32. Example: Link between catalog service domains 331
33. Example: Hub and spoke topology 332
34. Example using the command line 336
35. Choosing an ORB 344
36. `objectGrid.xml` file 383
37. `objectGridDeployment.xml` file 384
38. `objectGridStandAlone.xml` file 385
39. `objectGridDeploymentStandAlone.xml` file 387
40. JPA intra-domain topology 402
41. JPA embedded topology 403
42. JPA embedded, partitioned topology 404
43. JPA remote topology 406
44. Getting started sample topology 424
45. Microsoft SQL Server Northwind sample schema diagram 425
46. Customer and Order entity schema diagram 426
47. Category and Product entity schema diagram 427
48. Customer and Order entity schema diagram 428
49. Eclipse Equinox process for installing and starting OSGi bundles with eXtreme Scale plug-ins 452
50. Availability states of an ObjectGrid instance 489
51. Eclipse Equinox process for including all configuration and metadata in an OSGi bundle 494
52. Eclipse Equinox process for specify configuration and metadata outside of an OSGi bundle 495
53. CollectPlacementPlan.java 506
54. CollectContainerStatus.java 508
55. CollectPlacementPlan.java 509
56. Statistics overview 511
57. MBean overview 513
58. ObjectGridModule module structure 541
59. ObjectGridModule module structure example 542
60. mapModule structure 543
61. mapModule module structure example 543
62. hashIndexModule module structure 545
63. hashIndexModule module structure example 545
64. agentManagerModule structure 546
65. agentManagerModule structure example 547
66. queryModule structure 548
67. QueryStats.jpg queryModule structure example 548
68. Authentication flow for servers within the same security domain. 602

Tables

1. Arbitration approaches 45
2. Features that require Java SE 5, Java SE 6, and Java SE 7 61
3. Features that require Java SE 5, Java SE 6, and Java SE 7 179
4. Runtime files for WebSphere eXtreme Scale 213
5. Runtime files for WebSphere eXtreme Scale Client 214
6. Runtime files for WebSphere eXtreme Scale full installation 215
7. Runtime files for WebSphere eXtreme Scale Client 216
8. Arguments for the **xsadmin** utility and **xscmd** equivalent commands. 250
9. Deprecated properties and APIs 253
10. Deprecated properties and APIs 254
11. Deprecated properties and APIs 254
12. Operational checklist 259
13. createXSDomain command arguments 302
14. defineDomainServers step arguments 302
15. configureClientSecurity step arguments 303
16. modifyXSDomain command arguments 307
17. modifyEndpoints step arguments 307
18. addEndpoints step arguments 309
19. removeEndpoints step arguments 310
20. configureClientSecurity step arguments 311
21. Catalog server endpoint status 317
22. Valid heartbeat values 322
23. Data grid configurations for ORB and eXtremeIO transport types 361
24. Custom properties for configuring connection factories 362
25. Custom properties for SIP session management with ObjectGrid 376
26. Add Archive to Repository 438
27. Install New Applications. 438
28. Add Archive to Repository 439
29. Install New Applications. 439
30. Archive to repository 441
31. Installation values 441
32. Valid heartbeat values 468
33. Valid heartbeat values 575
34. Credential authentication under client and server settings 588
35. Transport protocol to use under client transport and server transport settings . . . 593
36. Entity access rights 601
37. Trace options 621

About the *Administration Guide*

The WebSphere® eXtreme Scale documentation set includes three volumes that provide the information necessary to use, program for, and administer the WebSphere eXtreme Scale product.

WebSphere eXtreme Scale library

The WebSphere eXtreme Scale library contains the following books:

- The *Product Overview* contains a high-level view of WebSphere eXtreme Scale concepts, including use case scenarios, and tutorials.
- The *Installation Guide* describes how to install common topologies of WebSphere eXtreme Scale.
- The *Administration Guide* contains the information necessary for system administrators, including how to plan application deployments, plan for capacity, install and configure the product, start and stop servers, monitor the environment, and secure the environment.
- The *Programming Guide* contains information for application developers on how to develop applications for WebSphere eXtreme Scale using the included API information.

To download the books, go to the WebSphere eXtreme Scale library page.

You can also access the same information in this library in the 7.1.1 WebSphere eXtreme Scale Version 7.1.1 information center.

Using the books offline

All of the books in the WebSphere eXtreme Scale library contain links to the information center, with the following root URL: 7.1.1 `http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1m1` . These links take you directly to related information. However, if you are working offline and encounter one of these links, you can search for the title of the link in the other books in the library. The API documentation, glossary, and messages reference are not available in PDF books.

Who should use this book

This book is intended primarily for system administrators, security administrators, and system operators.

Getting updates to this book

You can get updates to this book by downloading the most recent version from the WebSphere eXtreme Scale library page.

How to send your comments

Contact the documentation team. Did you find what you needed? Was it accurate and complete? Send your comments about this documentation by e-mail to wasdoc@us.ibm.com.

Chapter 1. Getting started

After you install the product, you can use the getting started sample to test the installation and use the product for the first time.

Tutorial: Getting started with WebSphere eXtreme Scale

After you install WebSphere eXtreme Scale in a stand-alone environment, you can use the getting started sample application to verify your installation. The getting started sample application is an introduction to in-memory data grids. The getting started sample application is only included in full (client and server) installations of WebSphere eXtreme Scale.You can use the getting started sample application to verify the connection between your client installation and the appliance. The getting started sample application is an introduction to enterprise data grids.

Learning objectives

- Learn about the ObjectGrid descriptor XML file and deployment policy descriptor XML files that you use to configure your environment.
- Start catalog and container servers with the configuration files.
- Learn about developing a client application
- Run the client application to insert data into the data grid.
- Monitor your data grids with the web console.

Time required

60 minutes

Related tasks:

7.1.1 "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in a stand-alone environment" on page 184
You can install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in an environment that does not contain WebSphere Application Server or WebSphere Application Server Network Deployment. This type of installation is called a stand-alone installation.

Getting started tutorial lesson 1.1: Defining data grids with configuration files

The `objectgrid.xml` and `deployment.xml` files are required to start container servers.

The sample uses the `objectgrid.xml` and `deployment.xml` files that are in the *wxs_install_root*/`ObjectGrid/gettingstarted/xml` directory. These files are passed to the start commands to start container servers and a catalog server. The `objectgrid.xml` file is the ObjectGrid descriptor XML file. The `deployment.xml` file is the ObjectGrid deployment policy descriptor XML file. These files together define a distributed topology.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

ObjectGrid descriptor XML file

An ObjectGrid descriptor XML file is used to define the structure of the ObjectGrid that is used by the application. It includes a list of backing map configurations. These backing maps store the cache data. The following example is a sample `objectgrid.xml` file. The first few lines of the file include the required header for each ObjectGrid XML file. This example file defines the `Grid` ObjectGrid with `Map1` and `Map2` backing maps.

```
<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>
```

Deployment policy descriptor XML file

The deployment policy descriptor XML file is intended to be paired with the corresponding ObjectGrid XML, the `objectgrid.xml` file. In the following example, the first few lines of the `deployment.xml` file include the required header for each deployment policy XML file. The file defines the **`objectgridDeployment`** element for the `Grid` ObjectGrid that is defined in the `objectgrid.xml` file. The `Map1` and `Map2` BackingMaps that are defined within the `Grid` ObjectGrid are included in the `mapSet` mapSet.

```
<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>
```

The **`numberOfPartitions`** attribute of the **`mapSet`** element specifies the number of partitions for the map set. This attribute is optional; the default value is 1. The attribute value must be appropriate for the anticipated capacity of the data grid.

The **`minSyncReplicas`** attribute of the **`mapSet`** element specifies the minimum number of synchronous replicas for each partition in the map set. This attribute is optional; the default is `0`. Primary and replica shards are not placed until the catalog service domain can support the minimum number of synchronous replicas. To support the **`minSyncReplicas`** value, you need one more container server than the value of the **minSyncReplicas** attribute. If the number of synchronous replicas falls below the value of the **minSyncReplicas** attribute, write transactions are no longer allowed for that partition.

The **maxSyncReplicas** attribute of the **mapSet** element is to specify the maximum number of synchronous replicas for each partition in the map set. This attribute is optional; the default is `0`. No other synchronous replicas are placed for a partition after a catalog service domain reaches this number of synchronous replicas for that specific partition. Adding container servers that can support this ObjectGrid can result in an increased number of synchronous replicas if your **maxSyncReplicas** value is not already met. The sample set the **maxSyncReplicas** to `1`, which means the catalog service domain places one synchronous replica at most. If you start more than one container server, only one synchronous replica is placed in one of the container server instances.

Lesson checkpoint

In this lesson, you learned:
- How to use the ObjectGrid descriptor XML file to define maps that store the cache data.
- How to use the deployment descriptor XML file to define the number of partitions and replicas for the data grid.

Getting started tutorial module 2: Create a client application

Write client applications to insert, update, delete, and retrieve data from your data grid. You can use the sample application to learn about how to create an application for your environment.

Learning objectives

After completing the lessons in this module you will know how to do the following:
- Develop a Java client application

Getting started tutorial lesson 2.1: Creating a client application

To insert, delete, update, and retrieve data from your data grid, you must write a client application. The getting started sample includes a client application that you can use to learn about creating your own client application.

The `Client.java` file in the *wxs_install_root*`/ObjectGrid/gettingstarted/client/src/` directory is the client program that demonstrates how to connect to a catalog server, obtain the ObjectGrid instance, and use the ObjectMap API. The ObjectMap API stores data as key-value pairs and is ideal for caching objects that have no relationships involved. The following steps discuss the contents of the `Client.java` file.

If you need to cache objects that have relationships, use the EntityManager API.

1. Connect to the catalog service by obtaining a ClientClusterContext instance.

   To connect to the catalog server, use the connect method of ObjectGridManager API. The connect method that is used requires only the catalog server endpoint in the format of *`hostname`*`:`*`port`*. You can indicate multiple catalog server endpoints by separating the list of *`hostname`*`:`*`port`* values with commas. The following code snippet demonstrates how to connect to a catalog server and obtain a ClientClusterContext instance:

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

If the connections to the catalog servers succeed, the connect method returns a ClientClusterContext instance. The ClientClusterContext instance is required to obtain the ObjectGrid from the ObjectGridManager API.

2. Obtain an ObjectGrid instance.

   To obtain ObjectGrid instance, use the getObjectGrid method of the ObjectGridManager API. The getObjectGrid method requires both the ClientClusterContext instance and the name of the data grid instance. The ClientClusterContext instance is obtained during the connection to catalog server. The name of ObjectGrid instance is `Grid` that is specified in the `objectgrid.xml` file. The following code snippet demonstrates how to obtain the data grid by calling the getObjectGrid method of the ObjectGridManager API.

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

3. Get a Session instance.

   You can get a Session from the obtained ObjectGrid instance. A Session instance is required to get the ObjectMap instance, and perform transaction demarcation. The following code snippet demonstrates how to get a Session instance by calling the getSession method of the ObjectGrid API.

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

4. Get an ObjectMap instance.

   After getting a Session, you can get an ObjectMap instance from a Session instance by calling getMap method of the Session API. You must pass the name of map as parameter to getMap method to get the ObjectMap instance. The following code snippet demonstrates how to obtain ObjectMap by calling the getMap method of the Session API.

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

5. Use the ObjectMap methods.

   After an ObjectMap instance is obtained, you can use the ObjectMap API. Remember that the ObjectMap interface is a transactional map and requires transaction demarcation by using the begin and commit methods of the Session API. If there is no explicit transaction demarcation in the application, the ObjectMap operations run with auto-commit transactions.

   - The following code snippet demonstrates how to use the ObjectMap API with an auto-commit transaction.

     ```
     map1.insert(k, v);
     ```

   - The following code snippet demonstrates how to use the ObjectMap API with explicit transaction demarcation.

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

6. 7.1.1+ Optional: Close the Session. After all of the Session and ObjectMap operations are complete, close the session with the Session.close() method. Running this method returns the resources that were being used by the session.

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

   As a result, subsequent getSession() method calls return faster, and fewer Session objects are in the heap.

Related concepts:

Caching objects with no relationships involved (ObjectMap API)
ObjectMaps are like Java™ Maps that allow data to be stored as key-value pairs. ObjectMaps provide a simple and intuitive approach for the application to store data. An ObjectMap is ideal for caching objects that have no relationships involved. If object relationships are involved, then you should use the EntityManager API.

Related tasks:

"Tutorial: Storing order information in entities" on page 81
The tutorial for the entity manager shows you how to use WebSphere eXtreme Scale to store order information on a Web site. You can create a simple Java Platform, Standard Edition 5 application that uses an in-memory, local data grid. The entities use Java SE 5 annotations and generics.

Related information:

API documentation

Lesson checkpoint:

In this lesson, you learned how to create a simple client application for performing data grid operations.

Module 3: Running the sample application in the data grid

To run the sample application, you must first start catalog servers and container servers. Then, you can run your sample application.

Learning objectives

After completing the lessons in this module you will know how to do the following:

- Start catalog and container servers
- Run the getting started sample client application

Getting started tutorial lesson 3.1: Starting catalog and container servers

To run the sample client application, you must start a catalog server and a container server.

The `env.sh|bat` script is called by the other scripts to set needed environment variables. Normally you do not need to change this script.

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

To run the application, you must first start the catalog service process. The catalog service is the control center of the data grid. The catalog service tracks the locations of container servers, and controls the placement of data to host container servers. After the catalog service starts, you can start the container servers, which store the application data for the data grid. To store multiple copies of the data, you can start multiple container servers. When all the servers are started, you can run the client application to insert, update, remove, and get data from the data grid.

1. Open a terminal session or command-line window.
2. In a terminal session or command line window, navigate to the *wxs_install_root*`/ObjectGrid/gettingstarted` directory .

3. Run the following script to start a catalog service process on localhost:
   - UNIX Linux `./runcat.sh`
   - Windows `runcat.bat`

   The catalog service process runs in the current terminal window.

   You can also start the catalog service with the **startOgServer** command. Run the **startOgServer** from the *wxs_install_root*/`ObjectGrid/bin` directory:
   - UNIX Linux `./startOgServer.sh cs0 -catalogServiceEndPoints cs0:localhost:6600:6601 -listenerPort 2809`
   - Windows `startOgServer.bat cs0 -catalogServiceEndPoints cs0:localhost:6600:6601 -listenerPort 2809`
4. Open another terminal session or command-line window, and run the following command to start a container server instance:
   - UNIX Linux `./runcontainer.sh server0`
   - Windows `runcontainer.bat server0`

   The container server runs in the current terminal window. If you want to start more container server instances to support replication, you can repeat this step with a different server name.

   You can also start container servers with the **startOgServer** command. Run the **startOgServer** command from the *wxs_install_root*/`ObjectGrid/bin` directory:
   - 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`

Related tasks:

"Starting and stopping stand-alone servers" on page 459
You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

Related reference:

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Lesson checkpoint:

In this lesson, you learned:
- How to start catalog servers and container servers

Getting started tutorial lesson 3.2: Running the getting started sample client application

Use the following steps to run a client to interact with the data grid. The catalog server, container server, and client all run on a single server in this example.

Open a terminal session or command-line window to run client commands. The `runclient.sh|bat` script runs the simple create, retrieve, update, and delete (CRUD) client and starts the specified operation. The `runclient.sh|bat` script is run with the following parameters:

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

For *command*, use one of the following options:

- Specify as `i` to insert *value2* into data grid with key *value1*
- Specify as `u` to update object that is keyed by *value1* to *value2*
- Specify as `d` to delete object that is keyed by *value1*
- Specify as `g` to retrieve and display object that is keyed by *value1*

1. Add data to the data grid.

   Important: If your system is using double byte character sets (DBCS), you might see garbled or corrupted text when you insert data into the data grid with the **`runClient`** script. This text can display in the output or in the cache. To work around this issue, update the Java call in the **`runClient`** script to include the **`-Xargencoding`** argument, and then specify the DBCS as a Unicode character set. For example, use the command: `\u runClient i key\u2e81 Hello\2e84World`

   - UNIX Linux `./runclient.sh i key1 helloWorld`
   - Windows `runclient.bat i key1 helloWorld`
2. Search and display the value:
   - UNIX Linux `./runclient.sh g key1`
   - Windows `runclient.bat g key1`
3. Update the value:
   - UNIX Linux `./runclient.sh u key1 goodbyeWorld`
   - Windows `runclient.bat u key1 goodbyeWorld`
4. Delete the value:
   - UNIX Linux `./runclient.sh d key1`
   - Windows `runclient.bat d key1`

Lesson checkpoint:

Lessons learned

In this lesson, you learned:

- How to run the sample client application to insert, get, update, and delete data from the data grid.

Getting started tutorial lesson 4: Monitor your environment

You can use the **`xscmd`** utility and web console tools to monitor your data grid environment.

Related tasks:

"Viewing statistics with the web console" on page 518
You can monitor statistics and other performance information with the web console.

"Monitoring with the web console" on page 514
With the web console, you can chart current and historical statistics. This console provides some preconfigured charts for high-level overviews, and has a custom reports page that you can use to build charts from the available statistics. You can use the charting capabilities in the monitoring console of WebSphere eXtreme Scale to view the overall performance of the data grids in your environment.

"Starting and logging on to the web console" on page 514
Start the console server by running the **`startConsoleServer`** command and logging on to the server with the default user ID and password.

"Connecting the web console to catalog servers" on page 516
To start viewing statistics in the web console, you must first connect to catalog servers that you want to monitor. Additional steps are required if your catalog servers have security enabled.

"Monitoring with the **`xscmd`** utility" on page 535
The **`xscmd`** utility replaces the **`xsadmin`** sample utility as a fully supported monitoring and administration tool. With the **`xscmd`** utility, you can display textual information about your WebSphere eXtreme Scale topology.

"Administering with the **`xscmd`** utility" on page 482
With the **`xscmd`** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related reference:

"Web console statistics" on page 519
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **`stopOgServer`** script stops catalog and container servers.

Monitoring with the web console

With the web console, you can chart current and historical statistics. This console provides some preconfigured charts for high-level overviews, and has a custom reports page that you can use to build charts from the available statistics. You can use the charting capabilities in the monitoring console of WebSphere eXtreme Scale to view the overall performance of the data grids in your environment.

Install the web console as an optional feature when you run the installation wizard.

1. Start the console server. The **`startConsoleServer.bat|sh`** script for starting the console server is in the *wxs_install_root*/`ObjectGrid/bin` directory of your installation.
2. Log on to the console.
   a. From your web browser, go to `https://your.console.host:7443`, replacing `your.console.host` with the host name of the server onto which you installed the console.
   b. Log on to the console.
      - **User ID:** `admin`
      - **Password:** `admin`

      The console welcome page is displayed.
3. Edit the console configuration. Click **Settings** > **Configuration** to review the console configuration. The console configuration includes information such as:
   - Trace string for the WebSphere eXtreme Scale client, such as `*=all=disabled`

- The Administrator name and password
- The Administrator e-mail address

4. Establish and maintain connections to catalog servers that you want to monitor. Repeat the following steps to add each catalog server to the configuration.
   a. Click **Settings** > **eXtreme Scale Catalog Servers**.
   b. Add a new catalog server.
      1) Click the add icon (➕) to register an existing catalog server.
      2) Provide information, such as the host name and listener port. See "Planning for network ports" on page 51 for more information about port configuration and defaults.
      3) Click **OK**.
      4) Verify that the catalog server has been added to the navigation tree.
5. Group the catalog servers that you created into a catalog service domain. You must create a catalog service domain when security is enabled in your catalog servers because security settings are configured in the catalog service domain.
   a. Click **Settings** > **eXtreme Scale Domains** page.
   b. Add a new catalog service domain.
      1) Click the add icon (➕) to register a catalog service domain. Enter a name for the catalog service domain.
      2) After you create the catalog service domain, you can edit the properties. The catalog service domain properties follow:

         Name Indicates the host name of the domain, as assigned by the administrator.

         Catalog servers
         Lists one or more catalog servers that belong to the selected domain. You can add the catalog servers that you created in the previous step.

         Generator class
         Specifies the name of the class that implements the CredentialGenerator interface. This class is used to get credentials for clients. If you specify a value in this field, the value overrides the **`crendentialGeneratorClass`** property in the `client.properties` file.

         Generator properties
         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. If you specify a value in this field, the value overrides the **`credentialGeneratorProps`** property in the `client.properties` file.

         eXtreme Scale client properties path
         Specifies the path to the client properties file that you edited to include security properties in a previous step. For example, you might indicate the `c:\ObjectGridProperties\sampleclient.properties` file. If you want to stop the console

from trying to use secure connections, you can delete the value in this field. After you set the path, the console uses an unsecured connection.

3) Click **OK**.

4) Verify that the domain has been added to the navigation tree.

To view information about an existing catalog service domain, click the name of the catalog service domain in the navigation tree on the **Settings** > **eXtreme Scale Domains** page.

6. View the connection status. The **Current domain** field indicates the name of the catalog service domain that is currently being used to display information in the web console. The connection status displays next to the name of the catalog service domain.
7. View statistics for the data grids and servers, or create a custom report.

Monitoring with the `xscmd` utility

1. Optional: If client authentication is enabled: Open a command-line window. On the command line, set appropriate environment variables.
2. Go to the *wxs_home*/`bin` directory.

   `cd` *`wxs_home`*`/bin`
3. Run various commands to display information about your environment.
   - Show all the online container servers for the Grid data grid and the mapSet map set:

     ```
     xscmd -c showPlacement -g Grid -ms mapSet
     ```
   - Display the routing information for the data grid.

     ```
     xscmd -c routetable -g Grid
     ```
   - Display the number of map entries in the data grid.

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

Stopping the servers

After you are done using the client application and monitoring the getting started sample environment, you can stop the servers.

- If you used the script files to start the servers, use `<ctrl+c>` to stop the catalog service process and container servers in the respective windows.

  Note: You can only use `<ctrl+c>` to stop command scripts that start with "run". For example, **`runcat.bat`**.
- If you used the **`startOgServer`** command to start your servers, use the **`stopOgServer`** command to stop the servers.

  Stop the container server:

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

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

  Stop the catalog server:

  – UNIX Linux `stopOgServer.sh cs1 -catalogServiceEndPoints localhost:2809`

  – Windows `stopOgServer.bat cs1 -catalogServiceEndPoints localhost:2809`

Lesson checkpoint

In this lesson, you learned:

- How to start the web console and connect it to the catalog server.
- How to monitor data grid and server statistics.
- How to stop the servers.

Chapter 2. Planning

Before you install WebSphere eXtreme Scale and deploy your data grid applications, you must decide on your caching topology, complete capacity planning, review the hardware and software requirements, networking and tuning settings, and so on. You can also use the operational checklist to ensure that your environment is ready to have an application deployed.

For a discussion of the best practices that you can use when you are designing your WebSphere eXtreme Scale applications, read the following article on developerWorks®: Principles and best practices for building high performing and highly resilient WebSphere eXtreme Scale applications.

Planning overview

Before using WebSphere eXtreme Scale in a production environment, consider the following issues to optimize your deployment.

Caching topology considerations

Each type of cache topology has advantages and disadvantages. The caching topology you implement depends on the requirements of your environment and application. For more information about the different caching topologies, see "Planning the topology" on page 14.

Data capacity considerations

The following list includes items to consider:

- **Number of systems and processors**: How many physical machines and processors are needed in the environment?
- **Number of servers**: How many eXtreme Scale servers to host eXtreme Scale maps?
- **Number of partitions**: The amount of data stored in the maps is one factor in determining the number of partitions needed.
- **Number of replicas**: How many replicas are required for each primary in the domain?
- **Synchronous or asynchronous replication**: Is the data vital so that synchronous replication is required? Or is performance a higher priority, making asynchronous replication the correct choice?
- **Heap sizes**: How much data will be stored on each server?

For a detailed discussion of each of these considerations, see "Planning environment capacity" on page 65..

Installation considerations

You can install WebSphere eXtreme Scale in a stand-alone environment, or you can integrate the installation with WebSphere Application Server. To ensure that you are able to seamlessly upgrade your servers in the future, you must plan your environment accordingly. For the best performance, catalog servers should run on different machines than the container servers. If you must run your catalog servers and container servers on the same machine, then use separate installations of

WebSphere eXtreme Scale for the catalog and container servers. By using two installations, you can upgrade the installation that is running the catalog server first. See “Updating eXtreme Scale servers” on page 235.

Planning the topology

With WebSphere eXtreme Scale, your architecture can use local in-memory data caching or distributed client-server data caching. The architecture can have varied relationships with your databases. You can also configure the topology to span multiple data centers.

WebSphere eXtreme Scale requires minimal additional infrastructure to operate. The infrastructure consists of scripts to install, start, and stop a Java Platform, Enterprise Edition application on a server. Cached data is stored in the container servers, and clients remotely connect to the server.

In-memory environments

When you deploy in a local, in-memory environment, WebSphere eXtreme Scale runs within a single Java virtual machine and is not replicated. To configure a local environment you can use an ObjectGrid XML file or the ObjectGrid APIs.

Distributed environments

When you deploy in a distributed environment, WebSphere eXtreme Scale runs across a set of Java virtual machines, increasing the performance, availability and scalability. With this configuration, you can use data replication and partitioning. You can also add additional servers without restarting your existing eXtreme Scale servers. As with a local environment, an ObjectGrid XML file, or an equivalent programmatic configuration, is needed in a distributed environment. You must also provide a deployment policy XML file with configuration details

You can create either simple deployments or large, terabyte-sized deployments in which thousands of servers are needed.

Local in-memory cache

In the simplest case, WebSphere eXtreme Scale can be used as a local (non-distributed) in-memory data grid cache. The local case can especially benefit high-concurrency applications where multiple threads need to access and modify transient data. The data kept in a local data grid can be indexed and retrieved using queries. Queries help you to work with large in memory data sets. The support provided with the Java virtual machine (JVM), although it is ready to use, has a limited data structure.

The local in-memory cache topology for WebSphere eXtreme Scale is used to provide consistent, transactional access to temporary data within a single Java virtual machine.

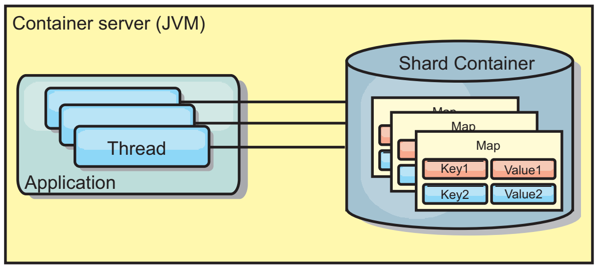


Figure 1. Local in-memory cache scenario

Advantages

- Simple setup: An ObjectGrid can be created programmatically or declaratively with the ObjectGrid deployment descriptor XML file or with other frameworks such as Spring.
- Fast: Each BackingMap can be independently tuned for optimal memory utilization and concurrency.
- Ideal for single-Java virtual machine topologies with small dataset or for caching frequently accessed data.
- Transactional. BackingMap updates can be grouped into a single unit of work and can be integrated as a last participant in 2-phase transactions such as Java Transaction Architecture (JTA) transactions.

Disadvantages

- Not fault tolerant.
- The data is not replicated. In-memory caches are best for read-only reference data.
- Not scalable. The amount of memory required by the database might overwhelm the Java virtual machine.
- Problems occur when adding Java virtual machines:
  - Data cannot easily be partitioned
  - Must manually replicate state between Java virtual machines or each cache instance could have different versions of the same data.
  - Invalidation is expensive.
  - Each cache must be warmed up independently. The warm-up is the period of loading a set of data so that the cache gets populated with valid data.

When to use

The local, in-memory cache deployment topology should only be used when the amount of data to be cached is small (can fit into a single Java virtual machine) and is relatively stable. Stale data must be tolerated with this approach. Using evictors to keep most frequently or recently used data in the cache can help keep the cache size low and increase relevance of the data.

Peer-replicated local cache

You must ensure the cache is synchronized if multiple processes with independent cache instances exist. To ensure that the cache instances are synchronized, enable a peer-replicated cache with Java Message Service (JMS).

WebSphere eXtreme Scale includes two plug-ins that automatically propagate transaction changes between peer ObjectGrid instances. The

JMSObjectGridEventListener plug-in automatically propagates eXtreme Scale changes using JMS.

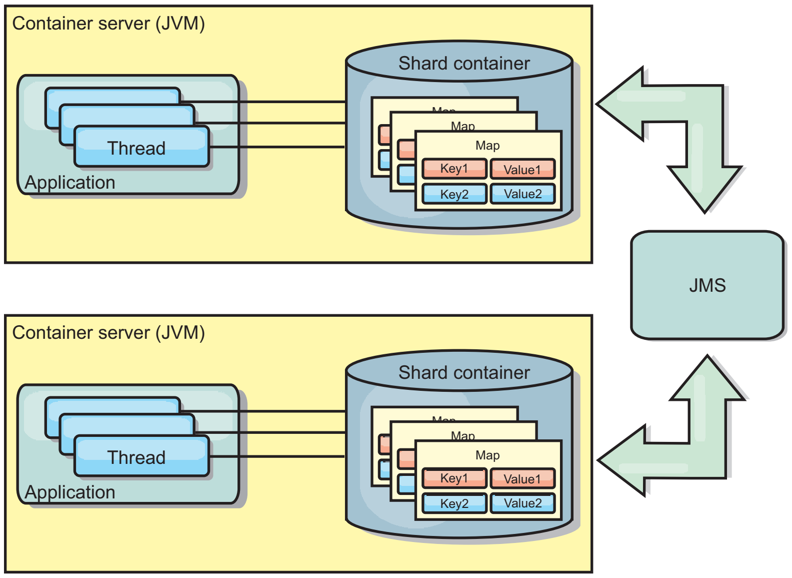


Figure 2. Peer-replicated cache with changes that are propagated with JMS

If you are running a WebSphere Application Server environment, the TranPropListener plug-in is also available. The TranPropListener plug-in uses the high availability (HA) manager to propagate the changes to each peer cache instance.

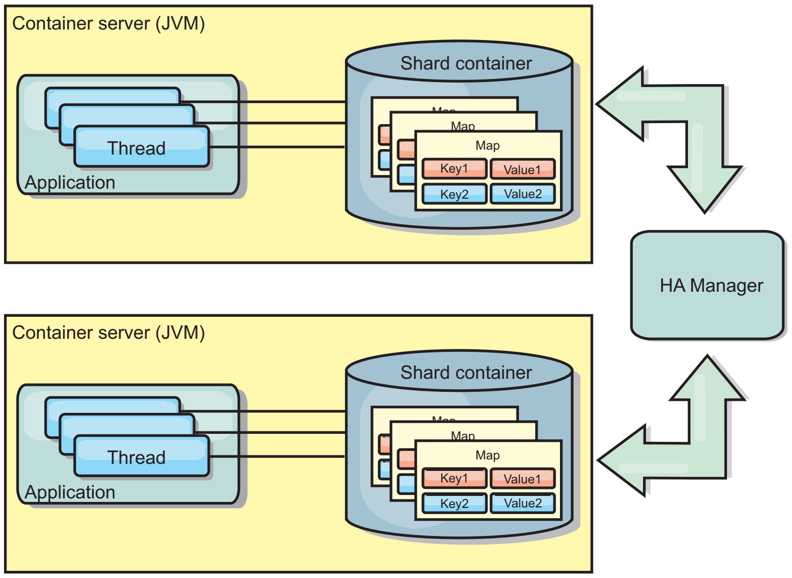


Figure 3. Peer-replicated cache with changes that are propagated with the high availability manager

Advantages

- The data is more valid because the data is updated more often.
- With the TranPropListener plug-in, like the local environment, the eXtreme Scale can be created programmatically or declaratively with the eXtreme Scale

deployment descriptor XML file or with other frameworks such as Spring. Integration with the high availability manager is done automatically.
- Each BackingMap can be independently tuned for optimal memory utilization and concurrency.
- BackingMap updates can be grouped into a single unit of work and can be integrated as a last participant in 2-phase transactions such as Java Transaction Architecture (JTA) transactions.
- Ideal for few-JVM topologies with a reasonably small dataset or for caching frequently accessed data.
- Changes to the eXtreme Scale are replicated to all peer eXtreme Scale instances. The changes are consistent as long as a durable subscription is used.

Disadvantages

- Configuration and maintenance for the JMSObjectGridEventListener can be complex. eXtreme Scale can be created programmatically or declaratively with the eXtreme Scale deployment descriptor XML file or with other frameworks such as Spring.
- Not scalable: The amount of memory required by the database may overwhelm the JVM.
- Functions improperly when adding Java virtual machines:
  - Data cannot easily be partitioned
  - Invalidation is expensive.
  - Each cache must be warmed-up independently

When to use

Use deployment topology only when the amount of data to be cached is small, can fit into a single JVM, and is relatively stable.

Embedded cache

WebSphere eXtreme Scale grids can run within existing processes as embedded eXtreme Scale servers or you can manage them as external processes.

Embedded grids are useful when you are running in an application server, such as WebSphere Application Server. You can start eXtreme Scale servers that are not embedded by using command line scripts and run in a Java process.

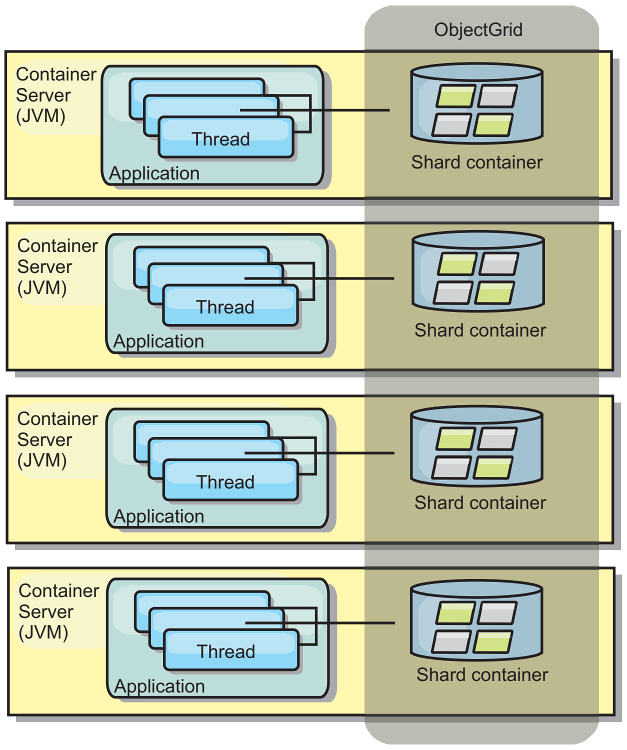


Figure 4. Embedded cache

Advantages

- Simplified administration since there are less processes to manage.
- Simplified application deployment since the grid is using the client application classloader.
- Supports partitioning and high availability.

Disadvantages

- Increased the memory footprint in client process since all of the data is collocated in the process.
- Increase CPU utilization for servicing client requests.
- More difficult to handle application upgrades since clients are using the same application Java archive files as the servers.
- Less flexible. Scaling of clients and grid servers cannot increase at the same rate. When servers are externally defined, you can have more flexibility in managing the number of processes.

When to use

Use embedded grids when there is plenty of memory free in the client process for grid data and potential failover data.

For more information, see the topic on enabling the client invalidation mechanism in the *Administration Guide*.

Distributed cache

WebSphere eXtreme Scale is most often used as a shared cache, to provide transactional access to data to multiple components where a traditional database would otherwise be used. The shared cache eliminates the need configure a database.

Coherency of the cache

The cache is coherent because all of the clients see the same data in the cache. Each piece of data is stored on exactly one server in the cache, preventing wasteful copies of records that could potentially contain different versions of the data. A coherent cache can also hold more data as more servers are added to the data grid, and scales linearly as the grid grows in size. Because clients access data from this data grid with remote procedural calls, it can also be known as a remote cache, or far cache. Through data partitioning, each process holds a unique subset of the total data set. Larger data grids can both hold more data and service more requests for that data. Coherency also eliminates the need to push invalidation data around the data grid because no stale data exists. The coherent cache only holds the latest copy of each piece of data.

If you are running a WebSphere Application Server environment, the TranPropListener plug-in is also available. The TranPropListener plug-in uses the high availability component (HA Manager) of WebSphere Application Server to propagate the changes to each peer ObjectGrid cache instance.

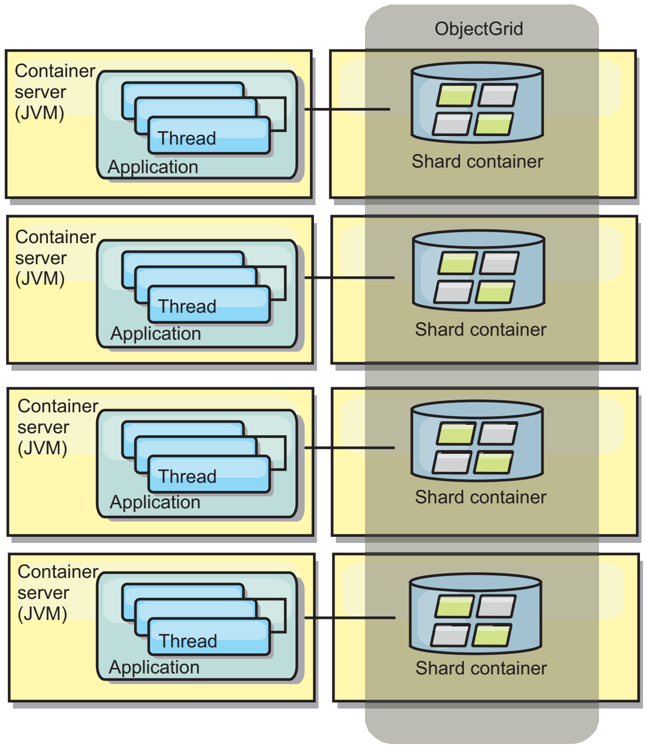


Figure 5. Distributed cache

Near cache

Clients can optionally have a local, in-line cache when eXtreme Scale is used in a distributed topology. This optional cache is called a near cache, an independent

ObjectGrid on each client, serving as a cache for the remote, server-side cache. The near cache is enabled by default when locking is configured as optimistic or none and cannot be used when configured as pessimistic.

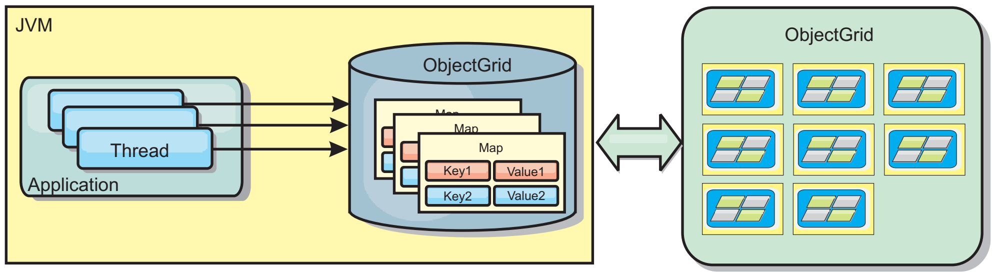


Figure 6. Near cache

A near cache is very fast because it provides in-memory access to a subset of the entire cached data set that is stored remotely in the eXtreme Scale servers. The near cache is not partitioned and contains data from any of the remote eXtreme Scale partitions.WebSphere eXtreme Scale can have up to three cache tiers as follows.

1. The transaction tier cache contains all changes for a single transaction. The transaction cache contains a working copy of the data until the transaction is committed. When a client transaction requests data from an ObjectMap, the transaction is checked first.
2. The near cache in the client tier contains a subset of the data from the server tier. When the transaction tier does not have the data, the data is fetched from the client tier, if available and inserted into the transaction cache
3. The data grid in the server tier contains the majority of the data and is shared among all clients. The server tier can be partitioned, which allows a large amount of data to be cached. When the client near cache does not have the data, it is fetched from the server tier and inserted into the client cache. The server tier can also have a Loader plug-in. When the data grid does not have the requested data, the Loader is invoked and the resulting data is inserted from the backend data store into the grid.

To disable the near cache, see "Configuring the near cache" on page 353.

Advantage

- Fast response time because all access to the data is local. Looking for the data in the near cache first saves a trip to the grid of servers, thus making even the remote data locally accessible.

Disadvantages

- Increases duration of stale data because the near cache at each tier may be out of synch with the current data in the data grid.
- Relies on an evictor to invalidate data to avoid running out of memory.

When to use

Use when response time is important and stale data can be tolerated.

Related tasks:

"Configuring the near cache" on page 353
Clients can optionally have a local, in-line cache when eXtreme Scale is used in a distributed topology. This optional cache is called a near cache, an independent data grid on each client, serving as a cache for the remote, server-side cache. The near cache is enabled by default when locking is disabled, or is configured as optimistic, and cannot be used when configured as pessimistic.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Database integration: Write-behind, in-line, and side caching

WebSphere eXtreme Scale is used to front a traditional database and eliminate read activity that is normally pushed to the database. A coherent cache can be used with an application directly or indirectly using an object relational mapper. The coherent cache can then offload the database or backend from reads. In a slightly more complex scenario, such as transactional access to a data set where only some of the data requires traditional persistence guarantees, filtering can be used to offload even write transactions.

You can configure WebSphere eXtreme Scale to function as a highly flexible in-memory database processing space. However, WebSphere eXtreme Scale is not an object relational mapper (ORM). It does not know where the data in the data grid came from. An application or an ORM can place data in an eXtreme Scale server. It is the responsibility of the source of the data to make sure that it stays consistent with the database where data originated. This means eXtreme Scale cannot invalidate data that is pulled from a database automatically. The application or mapper must provide this function and manage the data stored in eXtreme Scale.

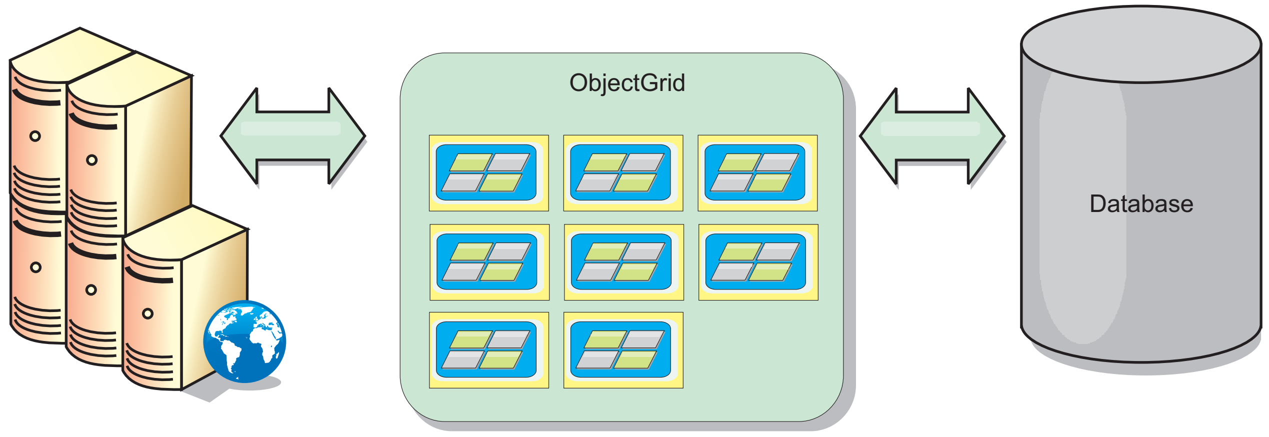


Figure 7. ObjectGrid as a database buffer

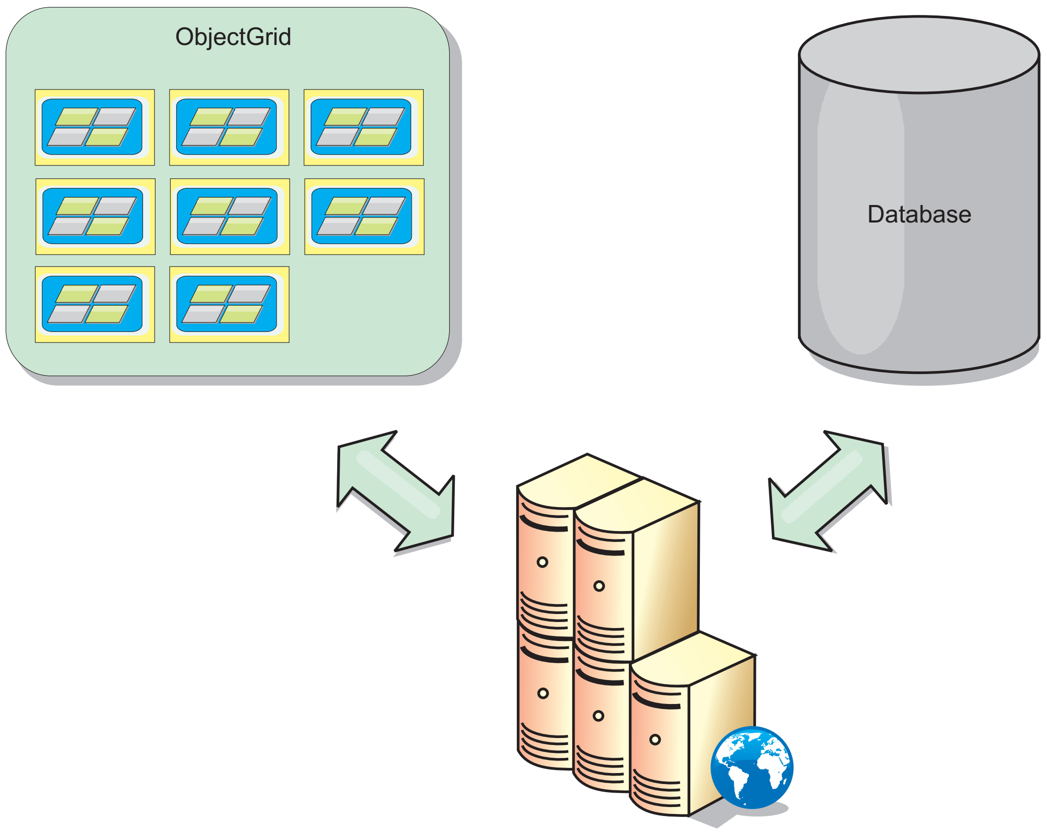


Figure 8. ObjectGrid as a side cache

Sparse and complete cache

WebSphere eXtreme Scale can be used as a sparse cache or a complete cache. A sparse cache only keeps a subset of the total data, while a complete cache keeps all of the data. and can be populated lazily, as the data is needed. Sparse caches are normally accessed using keys (instead of indexes or queries) because the data is only partially available.

Sparse cache

When a key is not present in a sparse cache, or the data is not available and a cache miss occurs, the next tier is invoked. The data is fetched, from a database for example, and is inserted into the data grid cache tier. If you are using a query or index, only the currently loaded values are accessed and the requests are not forwarded to the other tiers.

Complete cache

A complete cache contains all of the required data and can be accessed using non-key attributes with indexes or queries. A complete cache is preloaded with data from the database before the application tries to access the data. A complete cache can function as a database replacement after data is loaded. Because all of the data is available, queries and indexes can be used to find and aggregate data.

Side cache

When WebSphere eXtreme Scale is used as a side cache, the back end is used with the data grid.

Side cache

You can configure the product as a side cache for the data access layer of an application. In this scenario, WebSphere eXtreme Scale is used to temporarily store objects that would normally be retrieved from a back-end database. Applications check to see if the data grid contains the data. If the data is in the data grid, the data is returned to the caller. If the data does not exist, the data is retrieved from the back-end database. The data is then inserted into the data grid so that the next request can use the cached copy. The following diagram illustrates how WebSphere eXtreme Scale can be used as a side-cache with an arbitrary data access layer such as OpenJPA or Hibernate.

Side cache plug-ins for Hibernate and OpenJPA

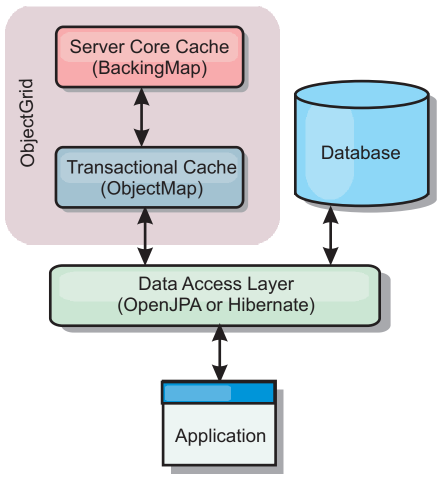

ObjectGrid
Server Core Cache (BackingMap)
Transactional Cache (ObjectMap)
Database
Data Access Layer (OpenJPA or Hibernate)
Application

Figure 9. Side cache

Cache plug-ins for both OpenJPA and Hibernate are included in WebSphere eXtreme Scale, so you can use the product as an automatic side-cache. Using WebSphere eXtreme Scale as a cache provider increases performance when reading and querying data and reduces load to the database. There are advantages that WebSphere eXtreme Scale has over built-in cache implementations because the cache is automatically replicated between all processes. When one client caches a value, all other clients can use the cached value.

In-line cache

You can configure in-line caching for a database back end or as a side cache for a database. In-line caching uses eXtreme Scale as the primary means for interacting with the data. When eXtreme Scale is used as an in-line cache, the application interacts with the back end using a Loader plug-in.

In-line cache

When used as an in-line cache, WebSphere eXtreme Scale interacts with the back end using a Loader plug-in. This scenario can simplify data access because applications can access the eXtreme Scale APIs directly. Several different caching scenarios are supported in eXtreme Scale to make sure the data in the cache and the data in the back end are synchronized. The following diagram illustrates how an in-line cache interacts with the application and back end.

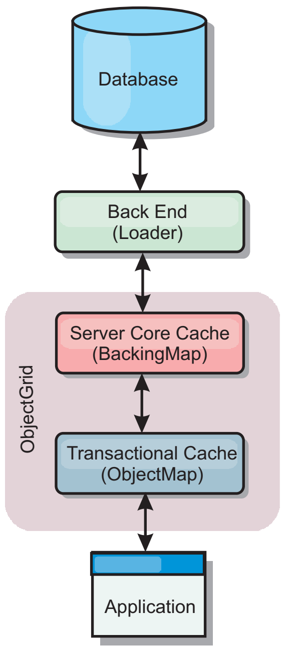


Figure 10. In-line cache

The in-line caching option simplifies data access because it allows applications to access the eXtreme Scale APIs directly. WebSphere eXtreme Scale supports several in-line caching scenarios, as follows.

- Read-through
- Write-through
- Write-behind

Read-through caching scenario

A read-through cache is a sparse cache that lazily loads data entries by key as they are requested. This is done without requiring the caller to know how the entries are populated. If the data cannot be found in the eXtreme Scale cache, eXtreme Scale will retrieve the missing data from the Loader plug-in, which loads the data from the back-end database and inserts the data into the cache. Subsequent requests for the same data key will be found in the cache until it is removed, invalidated or evicted.

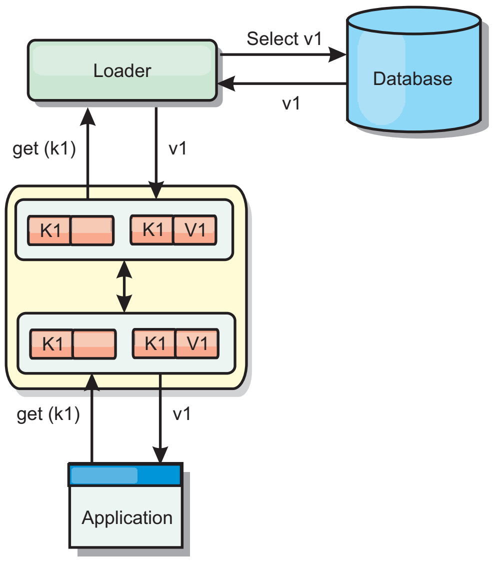


Figure 11. Read-through caching

Write-through caching scenario

In a write-through cache, every write to the cache synchronously writes to the database using the Loader. This method provides consistency with the back end, but decreases write performance since the database operation is synchronous. Since the cache and database are both updated, subsequent reads for the same data will be found in the cache, avoiding the database call. A write-through cache is often used in conjunction with a read-through cache.

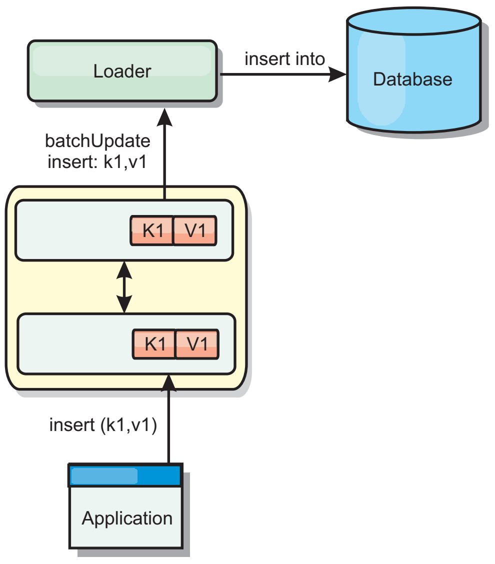


Figure 12. Write-through caching

Write-behind caching scenario

Database synchronization can be improved by writing changes asynchronously. This is known as a write-behind or write-back cache. Changes that would normally be written synchronously to the loader are instead buffered in eXtreme Scale and written to the database using a background thread. Write performance is significantly improved because the database operation is removed from the client transaction and the database writes can be compressed.

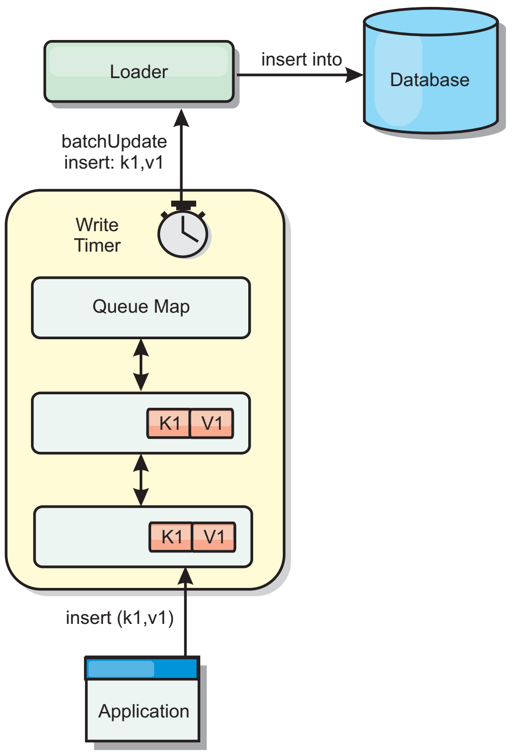


Figure 13. Write-behind caching

Write-behind caching

You can use write-behind caching to reduce the overhead that occurs when updating a database you are using as a back end.

Write-behind caching overview

Write-behind caching asynchronously queues updates to the Loader plug-in. You can improve performance by disconnecting updates, inserts, and removes for a map, the overhead of updating the back-end database. The asynchronous update is performed after a time-based delay (for example, five minutes) or an entry-based delay (1000 entries).

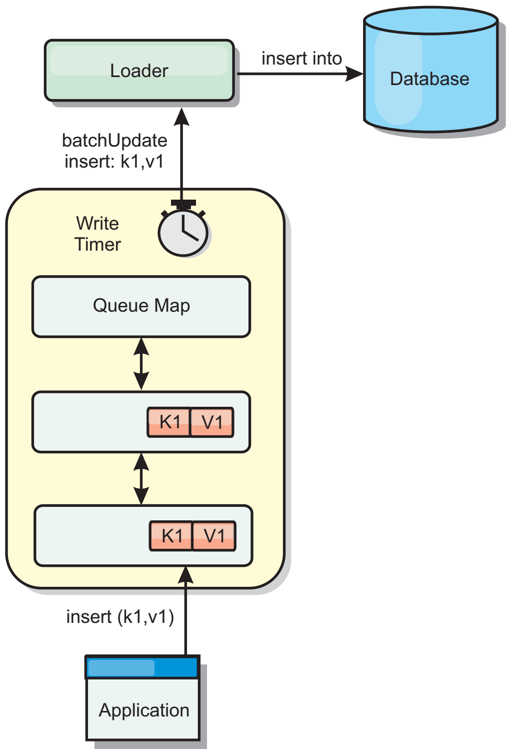


Figure 14. Write-behind caching

The write-behind configuration on a BackingMap creates a thread between the loader and the map. The loader then delegates data requests through the thread according to the configuration settings in the BackingMap.setWriteBehind method. When an eXtreme Scale transaction inserts, updates, or removes an entry from a map, a LogElement object is created for each of these records. These elements are sent to the write-behind loader and queued in a special ObjectMap called a queue map. Each backing map with the write-behind setting enabled has its own queue maps. A write-behind thread periodically removes the queued data from the queue maps and pushes them to the real back-end loader.

The write-behind loader only sends insert, update, and delete types of LogElement objects to the real loader. All other types of LogElement objects, for example, EVICT type, are ignored.

Write-behind support *is* an extension of the Loader plug-in, which you use to integrate eXtreme Scale with the database. For example, consult the “Configuring JPA loaders” on page 419 information about configuring a JPA loader.

Benefits

Enabling write-behind support has the following benefits:

- **Back end failure isolation:** Write-behind caching provides an isolation layer from back end failures. When the back-end database fails, updates are queued in the queue map. The applications can continue driving transactions to eXtreme Scale. When the back end recovers, the data in the queue map is pushed to the back-end.
- **Reduced back end load:** The write-behind loader merges the updates on a key basis so only one merged update per key exists in the queue map. This merge decreases the number of updates to the back-end database.

- **Improved transaction performance:** Individual eXtreme Scale transaction times are reduced because the transaction does not need to wait for the data to be synchronized with the back-end.

Loaders

With a Loader plug-in, a data grid map can behave as a memory cache for data that is typically kept in a persistent store on either the same system or another system. Typically, a database or file system is used as the persistent store. A remote Java virtual machine (JVM) can also be used as the source of data, allowing hub-based caches to be built using eXtreme Scale. A loader has the logic for reading and writing data to and from a persistent store.

Overview

Loaders are backing map plug-ins that are invoked when changes are made to the backing map or when the backing map is unable to satisfy a data request (a cache miss). The Loader is invoked when the cache is unable to satisfy a request for a key, providing read-through capability and lazy-population of the cache. A loader also allows updates to the database when cache values change. All changes within a transaction are grouped together to allow the number of database interactions to be minimized. A TransactionCallback plug-in is used in conjunction with the loader to trigger the demarcation of the backend transaction. Using this plug-in is important when multiple maps are included in a single transaction or when transaction data is flushed to the cache without committing.

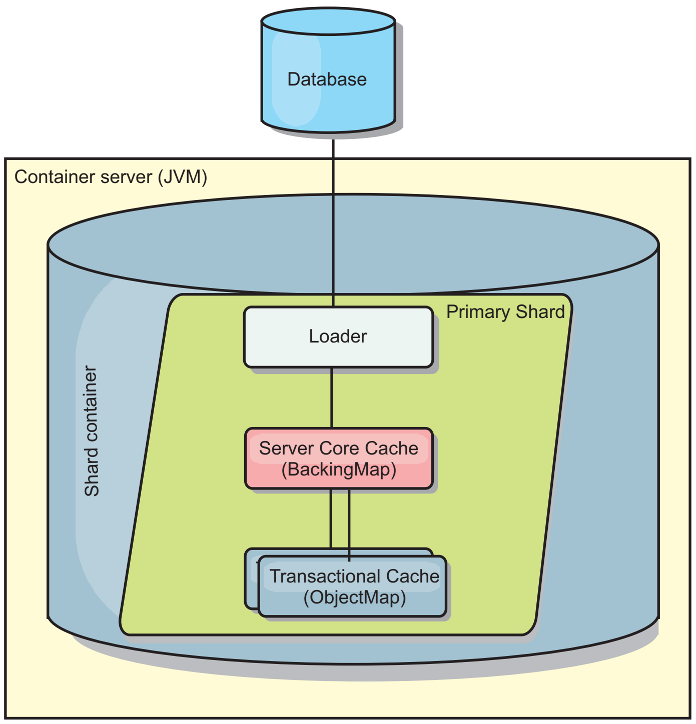


Figure 15. Loader

The loader can also use overqualified updates to avoid keeping database locks. By storing a version attribute in the cache value, the loader can see the before and after image of the value as it is updated in the cache. This value can then be used when updating the database or back end to verify that the data has not been updated. A Loader can also be configured to preload the data grid when it is started. When partitioned, a Loader instance is associated with each partition. If the "Company" Map has ten partitions, there are ten Loader instances, one per primary partition. When the primary shard for the Map is activated, the preloadMap method for the loader is invoked synchronously or asynchronously which allows loading the map partition with data from the back-end to occur automatically. When invoked synchronously, all client transactions are blocked, preventing inconsistent access to the data grid. Alternatively, a client preloader can be used to load the entire data grid.

Two built-in loaders can greatly simplify integration with relational database back ends. The JPA loaders utilize the Object-Relational Mapping (ORM) capabilities of both the OpenJPA and Hibernate implementations of the Java Persistence API (JPA) specification. See JPA Loaders for more information.

If you are using loaders in a multiple data center configuration, you must consider how revision data and cache consistency is maintained between the data grids. For more information, see “Loader considerations in a multi-master topology” on page 42.

Loader configuration

To add a Loader into the BackingMap configuration, you can use programmatic configuration or XML configuration. A loader has the following relationship with a backing map.
- A backing map can have only one loader.
- A client backing map (near cache) cannot have a loader.
- A loader definition can be applied to multiple backing maps, but each backing map has its own loader instance.

Related tasks:

“Monitoring eXtreme Scale information in DB2” on page 562
When the JPALoader or JPAEntityLoader is used with DB2® as the back-end database, eXtreme Scale-specific information can be passed to DB2. You can view this information by a performance monitor tool such as DB2 Performance Expert to monitor the eXtreme Scale applications that are accessing the database.

Data pre-loading and warm-up

In many scenarios that incorporate the use of a loader, you can prepare your data grid by pre-loading it with data.

When used as a complete cache, the data grid must hold all of the data and must be loaded before any clients can connect to it. When you are using a sparse cache, you can warm up the cache with data so that clients can have immediate access to data when they connect.

Two approaches exist for pre-loading data into the data grid: Using a Loader plug-in or using a client loader, as described in the following sections.

Loader plug-in

The loader plug-in is associated with each map and is responsible for synchronizing a single primary partition shard with the database. The preloadMap method of the loader plug-in is invoked automatically when a shard is activated. For example, if you have 100 partitions, 100 loader instances exist, each loading the data for its partition. When run synchronously, all clients are blocked until the preload has completed.

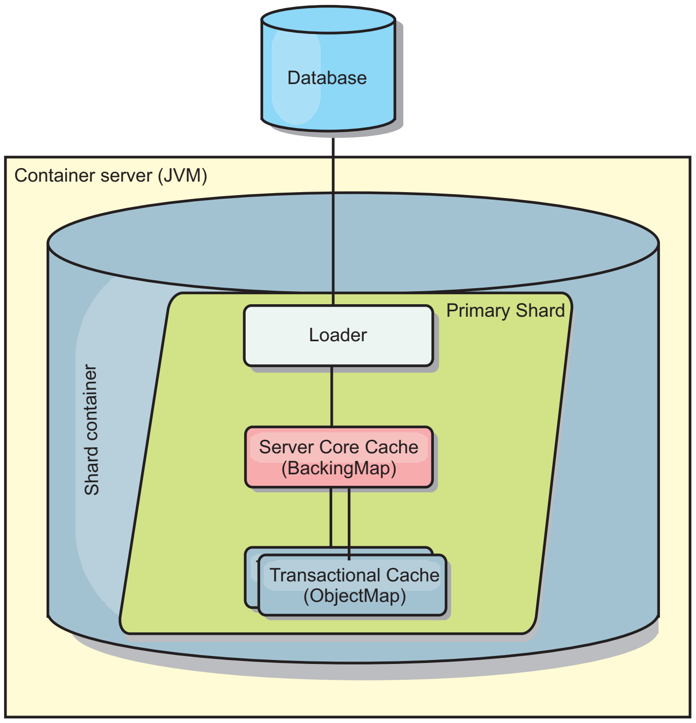


Figure 16. Loader plug-in

Client loader

A client loader is a pattern for using one or more clients to load the grid with data. Using multiple clients to load grid data can be effective when the partition scheme is not stored in the database. You can invoke client loaders manually or automatically when the data grid starts. Client loaders can optionally use the StateManager to set the state of the data grid to pre-load mode, so that clients are not able to access the grid while it is pre-loading the data. WebSphere eXtreme Scale includes a Java Persistence API (JPA)-based loader that you can use to automatically load the data grid with either the OpenJPA or Hibernate JPA providers. For more information about cache providers, see "JPA level 2 (L2) cache plug-in" on page 401.

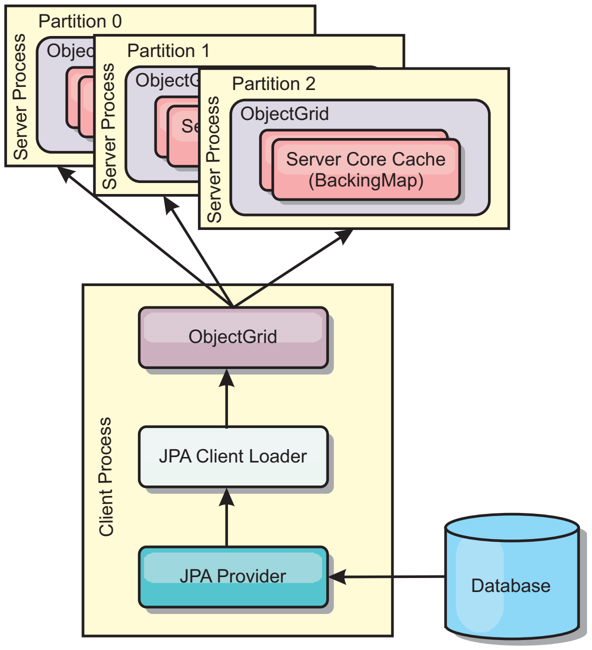


Figure 17. Client loader

Database synchronization techniques

When WebSphere eXtreme Scale is used as a cache, applications must be written to tolerate stale data if the database can be updated independently from an eXtreme Scale transaction. To serve as a synchronized in-memory database processing space, eXtreme Scale provides several ways of keeping the cache updated.

Database synchronization techniques

Periodic refresh

The cache can be automatically invalidated or updated periodically using the Java Persistence API (JPA) time-based database updater.The updater periodically queries the database using a JPA provider for any updates or inserts that have occurred since the previous update. Any changes identified are automatically invalidated or updated when used with a sparse cache. If used with a complete cache, the entries can be discovered and inserted into the cache. Entries are never removed from the cache.

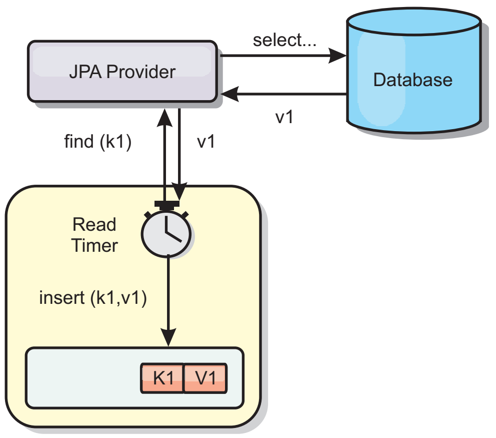


Figure 18. Periodic refresh

Eviction

Sparse caches can utilize eviction policies to automatically remove data from the cache without affecting the database. There are three built-in policies included in eXtreme Scale: time-to-live, least-recently-used, and least-frequently-used. All three policies can optionally evict data more aggressively as memory becomes constrained by enabling the memory-based eviction option.

Event-based invalidation

Sparse and complete caches can be invalidated or updated using an event generator such as Java Message Service (JMS). Invalidation using JMS can be manually tied to any process that updates the back-end using a database trigger. A JMS ObjectGridEventListener plug-in is provided in eXtreme Scale that can notify clients when the server cache has any changes. This can decrease the amount of time the client can see stale data.

Programmatic invalidation

The eXtreme Scale APIs allow manual interaction of the near and server cache using the Session.beginNoWriteThrough(), ObjectMap.invalidate() and EntityManager.invalidate() API methods. If a client or server process no longer needs a portion of the data, the invalidate methods can be used to remove data from the near or server cache. The beginNoWriteThrough method applies any ObjectMap or EntityManager operation to the local cache without calling the loader. If invoked from a client, the operation applies only to the near cache (the remote loader is not invoked). If invoked on the server, the operation applies only to the server core cache without invoking the loader.

Data invalidation

To remove stale cache data, you can use invalidation mechanisms.

Event-based invalidation

Sparse and complete caches can be invalidated or updated using an event generator such as Java Message Service (JMS). Invalidation using JMS can be manually tied to any process that updates the back-end using a database trigger. A JMS ObjectGridEventListener plug-in is provided in eXtreme Scale that can notify clients when the server cache changes. This type of notification decreases the amount of time the client can see stale data.

Event-based invalidation normally consists of the following three components.

- **Event queue:** An event queue stores the data change events. It could be a JMS queue, a database, an in-memory FIFO queue, or any kind of manifest as long as it can manage the data change events.
- **Event publisher:** An event publisher publishes the data change events to the event queue. An event publisher is usually an application you create or an eXtreme Scale plug-in implementation. The event publisher knows when the data is changed or it changes the data itself. When a transaction commits, events are generated for the changed data and the event publisher publishes these events to the event queue.
- **Event consumer:** An event consumer consumes data change events. The event consumer is usually an application to ensure the target grid data is updated with the latest change from other grids. This event consumer interacts with the event queue to get the latest data change and applies the data changes in the target grid. The event consumers can use eXtreme Scale APIs to invalidate stale data or update the grid with the latest data.

For example, JMSObjectGridEventListener has an option for a client-server model, in which the event queue is a designated JMS destination. All server processes are event publishers. When a transaction commits, the server gets the data changes and publishes them to the designated JMS destination. All the client processes are event consumers. They receive the data changes from the designated JMS destination and apply the changes to the client's near cache.

See “Configuring Java Message Service (JMS)-based client synchronization” on page 355 for more information.

Programmatic invalidation

The WebSphere eXtreme Scale APIs allow manual interaction of the near and server cache using the Session.beginNoWriteThrough(), ObjectMap.invalidate() and EntityManager.invalidate() API methods. If a client or server process no longer needs a portion of the data, the invalidate methods can be used to remove data from the near or server cache. The beginNoWriteThrough method applies any ObjectMap or EntityManager operation to the local cache without calling the loader. If invoked from a client, the operation applies only to the near cache (the remote loader is not invoked). If invoked on the server, the operation applies only to the server core cache without invoking the loader.

You can use programmatic invalidation with other techniques to determine when to invalidate the data. For example, this invalidation method uses event-based invalidation mechanisms to receive the data change events, and then uses APIs to invalidate the stale data.

Related tasks:

“Configuring the dynamic cache provider for WebSphere eXtreme Scale” on page 396
Installing and configuring the dynamic cache provider for eXtreme Scale depends on what your requirements are and the environment you have set up.

Indexing

Use the MapIndexPlugin plug-in to build an index or several indexes on a BackingMap to support non-key data access.

Index types and configuration

The indexing feature is represented by the MapIndexPlugin plug-in or Index for short. The Index is a BackingMap plug-in. A BackingMap can have multiple Index plug-ins configured, as long as each one follows the Index configuration rules.

You can use the indexing feature to build one or more indexes on a BackingMap. An index is built from an attribute or a list of attributes of an object in the BackingMap. This feature provides a way for applications to find certain objects more quickly. With the indexing feature, applications can find objects with a specific value or within a range of values of indexed attributes.

Two types of indexing are possible: static and dynamic. With static indexing, you must configure the index plug-in on the BackingMap before initializing the ObjectGrid instance. You can do this configuration with XML or programmatic configuration of the BackingMap. Static indexing starts building an index during ObjectGrid initialization. The index is always synchronized with the BackingMap and ready for use. After the static indexing process starts, the maintenance of the index is part of the eXtreme Scale transaction management process. When transactions commit changes, these changes also update the static index, and index changes are rolled back if the transaction is rolled back.

With dynamic indexing, you can create an index on a BackingMap before or after the initialization of the containing ObjectGrid instance. Applications have life cycle control over the dynamic indexing process so that you can remove a dynamic index when it is no longer needed. When an application creates a dynamic index, the index might not be ready for immediate use because of the time it takes to complete the index building process. Because the amount of time depends upon the amount of data indexed, the DynamicIndexCallback interface is provided for applications that want to receive notifications when certain indexing events occur. These events include ready, error, and destroy. Applications can implement this callback interface and register with the dynamic indexing process.

If a BackingMap has an index plug-in configured, you can obtain the application index proxy object from the corresponding ObjectMap. Calling the getIndex method on the ObjectMap and passing in the name of the index plug-in returns the index proxy object. You must cast the index proxy object to an appropriate application index interface, such as MapIndex, MapRangeIndex, or a customized index interface. After obtaining the index proxy object, you can use methods defined in the application index interface to find cached objects.

The steps to use indexing are summarized in the following list:
- Add either static or dynamic index plug-ins into the BackingMap.
- Obtain an application index proxy object by issuing the getIndex method of the ObjectMap.
- Cast the index proxy object to an appropriate application index interface, such as MapIndex, MapRangeIndex, or a customized index interface.
- Use methods that are defined in application index interface to find cached objects.

The HashIndex class is the built-in index plug-in implementation that can support both of the built-in application index interfaces: MapIndex and MapRangeIndex. You also can create your own indexes. You can add HashIndex as either a static or dynamic index into the BackingMap, obtain either MapIndex or MapRangeIndex index proxy object, and use the index proxy object to find cached objects.

Default index

If you want to iterate through the keys in a local map, you can use the default index. This index does not require any configuration, but it must be used against the shard, using an agent or an ObjectGrid instance retrieved from the ShardEvents.shardActivated(ObjectGrid shard) method.

Data quality consideration

The results of index query methods only represent a snapshot of data at a point of time. No locks against data entries are obtained after the results return to the application. Application has to be aware that data updates may occur on a returned data set. For example, the application obtains the key of a cached object by running the findAll method of MapIndex. This returned key object is associated with a data entry in the cache. The application should be able to run the get method on ObjectMap to find an object by providing the key object. If another transaction removes the data object from the cache just before the get method is called, the returned result will be null.

Indexing performance considerations

One of the main objectives of the indexing feature is to improve overall BackingMap performance. If indexing is not used properly, the performance of the application might be compromised. Consider the following factors before using this feature.

- **The number of concurrent write transactions:** Index processing can occur every time a transaction writes data into a BackingMap. Performance degrades if many transactions are writing data into the map concurrently when an application attempts index query operations.
- **The size of the result set that is returned by a query operation:** As the size of the resultset increases, the query performance declines. Performance tends to degrade when the size of the result set is 15% or more of the BackingMap.
- **The number of indexes built over the same BackingMap:** Each index consumes system resources. As the number of the indexes built over the BackingMap increases, performance decreases.

The indexing function can improve BackingMap performance drastically. Ideal cases are when the BackingMap has mostly read operations, the query result set is of a small percentage of the BackingMap entries, and only few indexes are built over the BackingMap.

Planning multiple data center topologies

Using multi-master asynchronous replication, two or more data grids can become exact copies of each other. Each data grid is hosted in an independent catalog service domain, with its own catalog service, container servers, and a unique name. With multi-master asynchronous replication, you can use links to connect a collection of catalog service domains. The catalog service domains are then synchronized using replication over the links. You can construct almost any topology through the definition of links between the catalog service domains.

Related tasks:

"Configuring multiple data center topologies" on page 329
With the multi-master asynchronous replication, you link a set of catalog service domains. The connected catalog service domains are then synchronized using replication over the links. You can define the links using properties files, at run time with Java Management Extensions (JMX) programs, or with command-line utilities. The set of current links for a domain is stored in the catalog service. You can add and remove links without restarting the catalog service domain that hosts the data grid.

Developing custom arbiters for multi-master replication
Change collisions might occur if the same records can be changed simultaneously in two places. In a multi-master replication topology, catalog service domains detect collisions automatically. When a catalog service domain detects a collision, it invokes an arbiter. Typically, collisions are resolved with the default collision arbiter. However, an application can provide a custom collision arbiter.

"Troubleshooting multiple data center configurations" on page 632
Use this information to troubleshoot multiple data center configurations, including linking between catalog service domains.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related information:

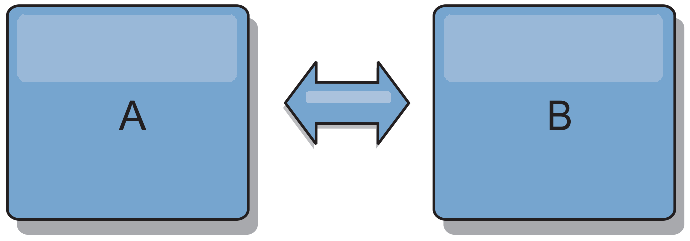
Improve response time and data availability with WebSphere eXtreme Scale multi-master capability

Topologies for multi-master replication

You have several different options when choosing the topology for your deployment that incorporates multi-master replication. Multi-master replication topologies can be implemented in the DataPower® XC10 Appliance by creating multiple collectives and linking them.

Links connecting catalog service domains

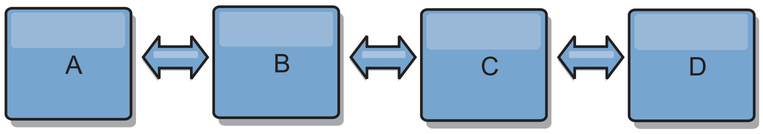
A replication data grid infrastructure is a connected graph of catalog service domains with bidirectional links among them. With a link, two catalog service domains can communicate data changes. For example, the simplest topology is a pair of catalog service domains with a single link between them. The catalog service domains are named alphabetically: A, B, C, and so on, from the left. A link can cross a wide area network (WAN), spanning large distances. Even if the link is interrupted, you can still change data in either catalog service domain. The topology reconciles changes when the link reconnects the catalog service domains. Links automatically try to reconnect if the network connection is interrupted.



After you set up the links, the product first tries to make every catalog service domain identical. Then, eXtreme Scale tries to maintain the identical conditions as changes occur in any catalog service domain. The goal is for each catalog service domain to be an exact mirror of every other catalog service domain connected by the links. The replication links between the catalog service domains help ensure that any changes made in one catalog service domain are copied to the other catalog service domains.

Line topologies

Although it is such a simple deployment, a line topology demonstrates some qualities of the links. First, it is not necessary for a catalog service domain to be connected directly to every other catalog service domain to receive changes. The catalog service domain B pulls changes from catalog service domain A. The catalog service domain C receives changes from catalog service domain A through catalog service domain B, which connects catalog service domains A and C. Similarly, catalog service domain D receives changes from the other catalog service domains through catalog service domain C. This ability spreads the load of distributing changes away from the source of the changes.



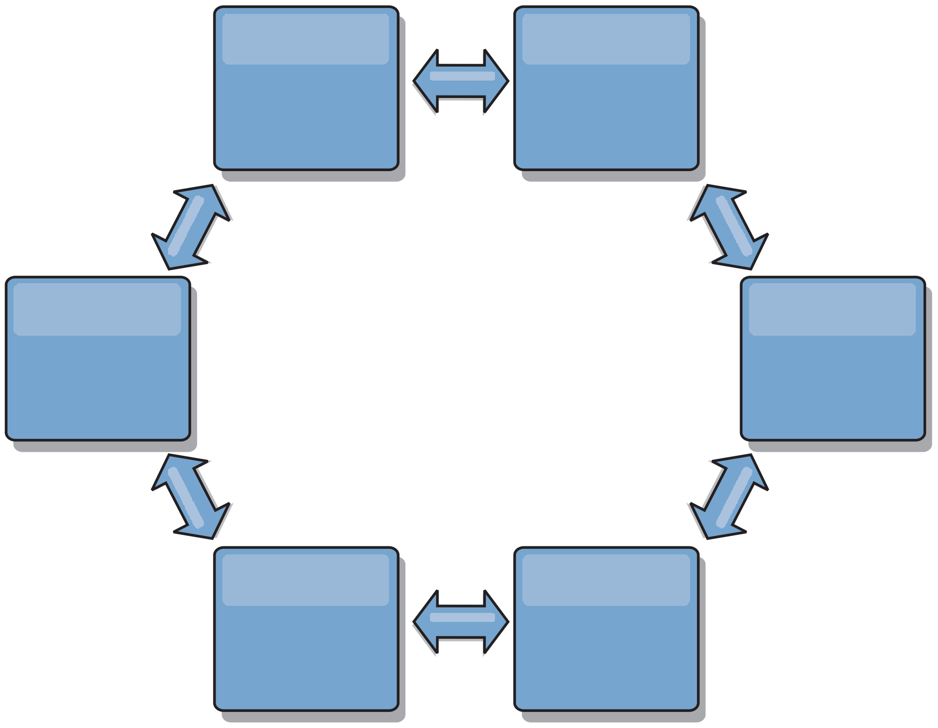
Notice that if catalog service domain C fails, the following actions would occur:

1. catalog service domain D would be orphaned until catalog service domain C was restarted
2. catalog service domain C would synchronize itself with catalog service domain B, which is a copy of catalog service domain A
3. catalog service domain D would use catalog service domain C to synchronize itself with changes on catalog service domain A and B. These changes initially occurred while catalog service domain D was orphaned (while catalog service domain C was down).

Ultimately, catalog service domains A, B, C, and D would all become identical to one other again.

Ring topologies

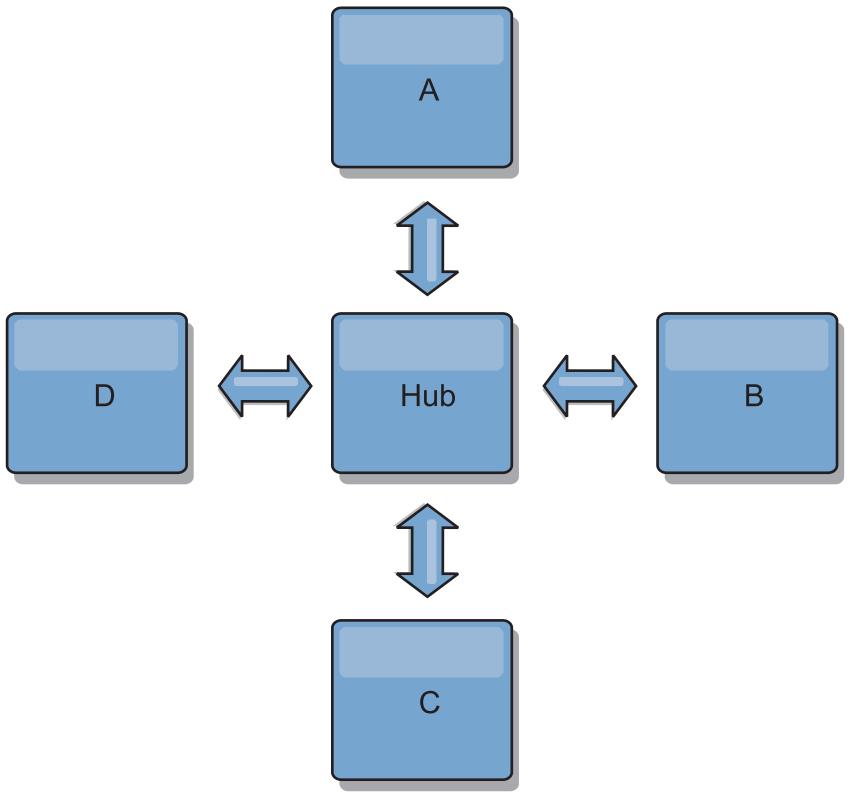
Ring topologies are an example of a more resilient topology. When a catalog service domain or a single link fails, the surviving catalog service domains can still obtain changes. The catalog service domains travel around the ring, away from the failure. Each catalog service domain has at most two links to other catalog service domains, no matter how large the ring topology. The latency to propagate changes can be large. Changes from a particular catalog service domain might need to travel through several links before all the catalog service domains have the changes. A line topology has the same characteristic.



You can also deploy a more sophisticated ring topology, with a root catalog service domain at the center of the ring. The root catalog service domain functions as the central point of reconciliation. The other catalog service domains act as remote points of reconciliation for changes occurring in the root catalog service domain. The root catalog service domain can arbitrate changes among the catalog service domains. If a ring topology contains more than one ring around a root catalog service domain, the catalog service domain can only arbitrate changes among the innermost ring. However, the results of the arbitration spread throughout the catalog service domains in the other rings.

Hub-and-spoke topologies

With a hub-and-spoke topology, changes travel through a hub catalog service domain. Because the hub is the only intermediate catalog service domain that is specified, hub-and-spoke topologies have lower latency. The hub catalog service domain is connected to every spoke catalog service domain through a link. The hub distributes changes among the catalog service domains. The hub acts as a point of reconciliation for collisions. In an environment with a high update rate, the hub might require run on more hardware than the spokes to remain synchronized. WebSphere eXtreme Scale is designed to scale linearly, meaning you can make the hub larger, as needed, without difficulty. However, if the hub fails, then changes are not distributed until the hub restarts. Any changes on the spoke catalog service domains will be distributed after the hub is reconnected.



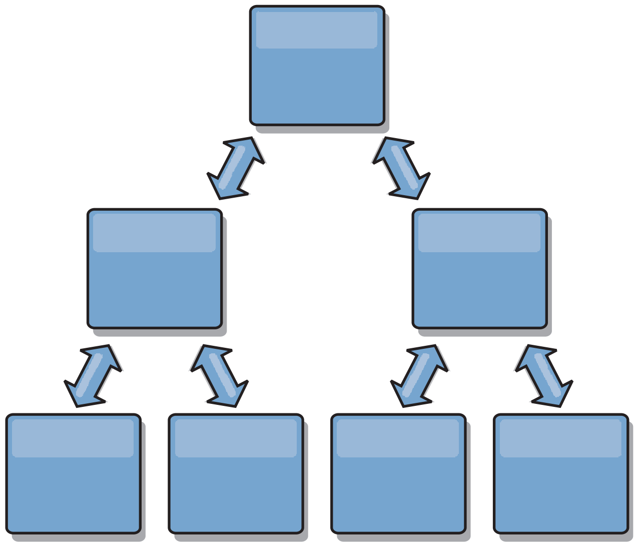
You can also use a strategy with fully replicated clients, a topology variation which uses a pair of servers that are running as a hub. Every client creates a self-contained single container data grid with a catalog in the client JVM. A client uses its data grid to connect to the hub catalog. This connection causes the client to synchronize with the hub as soon as the client obtains a connection to the hub.

Any changes made by the client are local to the client, and are replicated asynchronously to the hub. The hub acts as an arbitration catalog service domain, distributing changes to all connected clients. The fully replicated clients topology provides a reliable L2 cache for an object relational mapper, such as OpenJPA. Changes are distributed quickly among client JVMs through the hub. If the cache size can be contained within the available heap space, the topology is a reliable architecture for this style of L2.

Use multiple partitions to scale the hub catalog service domain on multiple JVMs, if necessary. Because all of the data still must fit in a single client JVM, multiple partitions increase the capacity of the hub to distribute and arbitrate changes. However, having multiple partitions does not change the capacity of a single catalog service domain.

Tree topologies

You can also use an acyclic directed tree. An acyclic tree has no cycles or loops, and a directed setup limits links to existing only between parents and children. This configuration is useful for topologies that have many catalog service domains. In these topologies, it is not practical to have a central hub that is connected to every possible spoke. This type of topology can also be useful when you must add child catalog service domains without updating the root catalog service domain.



A tree topology can still have a central point of reconciliation in the root catalog service domain. The second level can still function as a remote point of reconciliation for changes occurring in the catalog service domain beneath them. The root catalog service domain can arbitrate changes between the catalog service domains on the second level only. You can also use N-ary trees, each of which have N children at each level. Each catalog service domain connects out to *n* links.

Fully replicated clients

This topology variation involves a pair of servers that are running as a hub. Every client creates a self-contained single container data grid with a catalog in the client JVM. A client uses its data grid to connect to the hub catalog, causing the client to synchronize with the hub as soon as the client obtains a connection to the hub.

Any changes made by the client are local to the client, and are replicated asynchronously to the hub. The hub acts as an arbitration catalog service domain, distributing changes to all connected clients. The fully replicated clients topology provides a good L2 cache for an object relational mapper, such as OpenJPA. Changes are distributed quickly among client JVMs through the hub. As long as the cache size can be contained within the available heap space of the clients, this topology is a good architecture for this style of L2.

Use multiple partitions to scale the hub catalog service domain on multiple JVMs, if necessary. Because all of the data still must fit in a single client JVM, using multiple partitions increases the capacity of the hub to distribute and arbitrate changes, but it does not change the capacity of a single catalog service domain.

Related tasks:

"Configuring multiple data center topologies" on page 329
With the multi-master asynchronous replication, you link a set of catalog service domains. The connected catalog service domains are then synchronized using replication over the links. You can define the links using properties files, at run time with Java Management Extensions (JMX) programs, or with command-line utilities. The set of current links for a domain is stored in the catalog service. You can add and remove links without restarting the catalog service domain that hosts the data grid.

Developing custom arbiters for multi-master replication
Change collisions might occur if the same records can be changed simultaneously in two places. In a multi-master replication topology, catalog service domains detect collisions automatically. When a catalog service domain detects a collision, it invokes an arbiter. Typically, collisions are resolved with the default collision arbiter. However, an application can provide a custom collision arbiter.

Configuration considerations for multi-master topologies

Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

- **Map set requirements**

  Map sets must have the following characteristics to replicate changes across catalog service domain links:

  - The ObjectGrid name and map set name within a catalog service domain must match the ObjectGrid name and map set name of other catalog service domains. For example, ObjectGrid "og1" and map set "ms1" must be configured in catalog service domain A and catalog service domain B to replicate the data in the map set between the catalog service domains.
  - Is a `FIXED_PARTITION` data grid. `PER_CONTAINER` data grids cannot be replicated.
  - Has the same number of partitions in each catalog service domain. The map set might or might not have the same number and types of replicas.
  - Has the same data types being replicated in each catalog service domain.
  - Contains the same maps and dynamic map templates in each catalog service domain.
  - Does not use entity manager. A map set containing an entity map is not replicated across catalog service domains.
  - Does not use write-behind caching support. A map set containing a map that is configured with write-behind support is not replicated across catalog service domains.

  Any map sets with the preceding characteristics begin to replicate after the catalog service domains in the topology have been started.

- **Class loaders with multiple catalog service domains**

  Catalog service domains must have access to all classes that are used as keys and values. Any dependencies must be reflected in all class paths for data grid container Java virtual machines (JVM) for all domains. If a CollisionArbiter plug-in retrieves the value for a cache entry, then the classes for the values must be present for the domain that is starting the arbiter.

Related tasks:

"Configuring multiple data center topologies" on page 329
With the multi-master asynchronous replication, you link a set of catalog service domains. The connected catalog service domains are then synchronized using replication over the links. You can define the links using properties files, at run time with Java Management Extensions (JMX) programs, or with command-line utilities. The set of current links for a domain is stored in the catalog service. You can add and remove links without restarting the catalog service domain that hosts the data grid.

Developing custom arbiters for multi-master replication
Change collisions might occur if the same records can be changed simultaneously in two places. In a multi-master replication topology, catalog service domains detect collisions automatically. When a catalog service domain detects a collision, it invokes an arbiter. Typically, collisions are resolved with the default collision

arbiter. However, an application can provide a custom collision arbiter.

7.1.1 “Using the Update Installer to install maintenance packages” on page 248
Use the IBM® Update Installer to update your WebSphere eXtreme Scale or WebSphere eXtreme Scale Client environment with various types of maintenance, such as interim fixes, fix packs, and refresh packs.

Retrieving eXtreme Scale environment information with the **xscmd** utility

“Updating eXtreme Scale servers” on page 235
You can upgrade WebSphere eXtreme Scale to a new version, either by applying maintenance or installing a new version, without interrupting service.

“Migrating to WebSphere eXtreme Scale Version 7.1.1” on page 238
With the WebSphere eXtreme Scale installer, you cannot upgrade or modify a previous installation. You must uninstall the previous version before you install the new version. You do not need to migrate your configuration files because they are backward compatible. However, if you changed any of the script files that are shipped with the product, you must reapply these changes to the updated script files.

“Starting and stopping stand-alone servers” on page 459
You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

“Starting and stopping servers in a WebSphere Application Server environment” on page 475
Catalog and container servers can automatically start in a WebSphere Application Server or WebSphere Application Server Network Deployment environment.

Loader considerations in a multi-master topology

When you are using loaders in a multi-master topology, you must consider the possible collision and revision information maintenance challenges. The data grid maintains revision information about the items in the data grid so that collisions can be detected when other primary shards in the configuration write entries to the data grid. When entries are added from a loader, this revision information is not included and the entry takes on a new revision. Because the revision of the entry seems to be a new insert, a false collision could occur if another primary shard also changes this state or pulls the same information in from a loader.

Replication changes invoke the get method on the loader with a list of the keys that are not already in the data grid but are going to be changed during the replication transaction. When the replication occurs, these entries are collision entries. When the collisions are arbitrated and the revision is applied then a batch update is called on the loader to apply the changes to the database. All of the maps that were changed in the revision window are updated in the same transaction.

Preload conundrum

Consider a two data center topology with data center A and data center B. Both data centers have independent databases, but only data center A is has a data grid that is running. When you establish a link between the data centers for a multi-master configuration, the data grids in data center A begin pushing data to the new data grids in data center B, causing a collision with every entry. Another major issue that occurs is with any data that is in the database in data center B but not in the database in data center A. These rows are not populated and arbitrated, resulting in inconsistencies that are not resolved.

Solution to the preload conundrum

Because data that resides only in the database cannot have revisions, you must always fully preload the data grid from the local database before establishing the multi-master link. Then, both data grids can revision and arbitrate the data, eventually reaching a consistent state.

Sparse cache conundrum

With a sparse cache, the application first attempts to find data in the data grid. If the data is not in the data grid, the data is searched for in the database using the loader. Entries are evicted from the data grid periodically to maintain a small cache size.

This cache type can be problematic in a multi-master configuration scenario because the entries within the data grid have revisioning metadata that help detect when collisions occur and which side has made changes. When links between the data centers are not working, one data center can update an entry and then eventually update the database and invalidate the entry in the data grid. When the link recovers, the data centers attempt to synchronize revisions with each other. However, because the database was updated and the data grid entry was invalidated, the change is lost from the perspective of the data center that went down. As a result, the two sides of the data grid are out of synch and are not consistent.

Solution to the sparse cache conundrum

Hub and spoke topology:

You can run the loader only in the hub of a hub and spoke topology, maintaining consistency of the data while scaling out the data grid. However, if you are considering this deployment, note that the loaders can allow the data grid to be partially loaded, meaning that an evictor has been configured. If the spokes of your configuration are sparse caches but have no loader, then any cache misses have no way to retrieve data from the database. Because of this restriction, you should use a fully populated cache topology with a hub and spoke configuration.

Invalidations and eviction

Invalidation creates inconsistency between the data grid and the database. Data can be removed from the data grid either programmatically or with eviction. When you develop your application, you must be aware that revision handling does not replicate changes that are invalidated, resulting in inconsistencies between primary shards.

Invalidation events are not cache state changes and do not result in replication. Any configured evictors run independently from other evictors in the configuration. For example, you might have one evictor configured for a memory threshold in one catalog service domain, but a different type of less aggressive evictor in your other linked catalog service domain. When data grid entries are removed due to the memory threshold policy, the entries in the other catalog service domain are not affected.

Database updates and data grid invalidation

Problems occur when you update the database directly in the background while calling the invalidation on the data grid for the updated entries in a multi-master configuration. This problem occurs because the data grid cannot replicate the change to the other primary shards until some type of cache access moves the entry into the data grid.

Multiple writers to a single logical database

When you are using a single database with multiple primary shards that are connected through a loader, transactional conflicts result. Your loader implementation must specially handle these types of scenarios.

Mirroring data using multi-master replication

You can configure independent databases that are connected to independent catalog service domains. In this configuration, the loader can push changes from one data center to the other data center.

Related tasks:

"Configuring multiple data center topologies" on page 329
With the multi-master asynchronous replication, you link a set of catalog service domains. The connected catalog service domains are then synchronized using replication over the links. You can define the links using properties files, at run time with Java Management Extensions (JMX) programs, or with command-line utilities. The set of current links for a domain is stored in the catalog service. You can add and remove links without restarting the catalog service domain that hosts the data grid.

Developing custom arbiters for multi-master replication
Change collisions might occur if the same records can be changed simultaneously in two places. In a multi-master replication topology, catalog service domains detect collisions automatically. When a catalog service domain detects a collision, it invokes an arbiter. Typically, collisions are resolved with the default collision arbiter. However, an application can provide a custom collision arbiter.

Design considerations for multi-master replication

When implementing multi-master replication, you must consider aspects in your design such as: arbitration, linking, and performance.

Arbitration considerations in topology design

Change collisions might occur if the same records can be changed simultaneously in two places. Set up each catalog service domain to have about the same amount of processor, memory, network resources. You might observe that catalog service domains performing change collision handling (arbitration) use more resources than other catalog service domains. Collisions are detected automatically. They are handled with one of two mechanisms:

- **Default collision arbiter**: The default protocol is to use the changes from the lexically lowest named catalog service domain. For example, if catalog service domain A and B generate a conflict for a record, then the change from catalog service domain B is ignored. Catalog service domain A keeps its version and the record in catalog service domain B is changed to match the record from catalog service domain A. This behavior applies as well for applications where users or sessions are normally bound or have affinity with one of the data grids.

- **Custom collision arbiter**: Applications can provide a custom arbiter. When a catalog service domain detects a collision, it starts the arbiter. For information about developing a useful custom arbiter, see Developing custom arbiters for multi-master replication.

For topologies in which collisions are possible, consider implementing a hub-and-spoke topology or a tree topology. These two topologies are conducive to avoiding constant collisions, which can happen in the following scenarios:

1. Multiple catalog service domains experience a collision
2. Each catalog service domain handles the collision locally, producing revisions
3. The revisions collide, resulting in revisions of revisions

To avoid collisions, choose a specific catalog service domain, called an *arbitration catalog service domain* as the collision arbiter for a subset of catalog service domains. For example, a hub-and-spoke topology might use the hub as the collision handler. The spoke collision handler ignores any collisions that are detected by the spoke catalog service domains. The hub catalog service domain creates revisions, preventing unexpected collision revisions. The catalog service domain that is assigned to handle collisions must link to all of the domains for which it is responsible for handling collisions. In a tree topology, any internal parent domains handle collisions for their immediate children. In contrast, if you use a ring topology, you cannot designate one catalog service domain in the ring as the arbiter.

The following table summarizes the arbitration approaches that are most compatible with various topologies.

Table 1. Arbitration approaches. This table states whether application arbitration is compatible with various technologies.

Topology	**Application Arbitration?**	**Notes**
A line of two catalog service domains	Yes	Choose one catalog service domain as the arbiter.
A line of three catalog service domains	Yes	The middle catalog service domain must be the arbiter. Think of the middle catalog service domain as the hub in a simple hub-and-spoke topology.
A line of more than three catalog service domains	No	Application arbitration is not supported.
A hub with N spokes	Yes	Hub with links to all spokes must be the arbitration catalog service domain.
A ring of N catalog service domains	No	Application arbitration is not supported.
An acyclic, directed tree (n-ary tree)	Yes	All root nodes must rate their direct descendants only.

Linking considerations in topology design

Ideally, a topology includes the minimum number of links while optimizing trade-offs among change latency, fault tolerance, and performance characteristics.

- **Change latency**

Change latency is determined by the number of intermediate catalog service domains a change must go through before arriving at a specific catalog service domain.

A topology has the best change latency when it eliminates intermediate catalog service domains by linking every catalog service domain to every other catalog service domain. However, a catalog service domain must perform replication work in proportion to its number of links. For large topologies, the sheer number of links to be defined can cause an administrative burden.

The speed at which a change is copied to other catalog service domains depends on additional factors, such as:

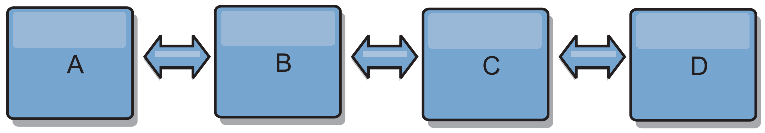
– Processor and network bandwidth on the source catalog service domain
– The number of intermediate catalog service domains and links between the source and target catalog service domain
– The processor and network resources available to the source, target, and intermediate catalog service domains

- **Fault tolerance**

  Fault tolerance is determined by how many paths exist between two catalog service domains for change replication.

  If you have only one link between a given pair of catalog service domains, a link failure disallows propagation of changes. Similarly, changes are not propagated between catalog service domains if any of the intermediate domains experiences link failure. Your topology could have a single link from one catalog service domain to another such that the link passes through intermediate domains. If so, then changes are not propagated if any of the intermediate catalog service domains is down.

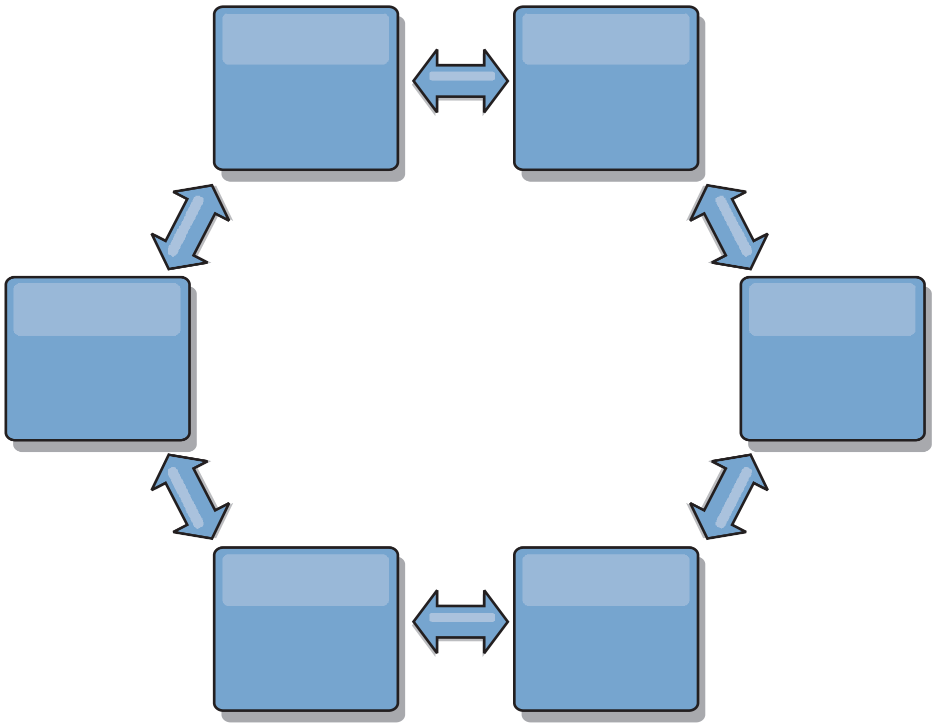
  Consider the line topology with four catalog service domains A, B, C, and D:



If any of these conditions hold, Domain D does not see any changes from A:

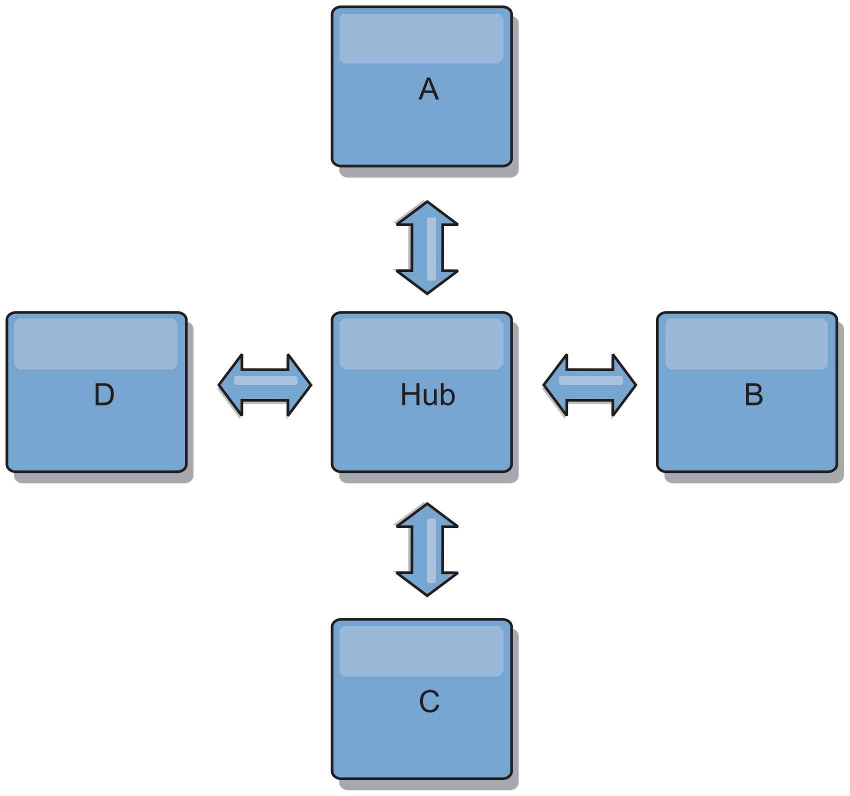
– Domain A is up and B is down
– Domains A and B are up and C is down
– The link between A and B is down
– The link between B and C is down
– The link between C and D is down

In contrast, with a ring topology, each catalog service domain can receive changes from either direction.



For example, if a given catalog service in your ring topology is down, then the two adjacent domains can still pull changes directly from each other.

All changes are propagated through the hub. Thus, as opposed to the line and ring topologies, the hub-and-spoke design is susceptible to break drown if the hub fails.



A single catalog service domain is resilient to a certain amount of service loss. However, larger failures such as wide network outages or loss of links between physical data centers can disrupt any of your catalog service domains.

- **Linking and performance**

  The number of links defined on a catalog service domain affects performance. More links use more resources and replication performance can drop as a result. The ability to retrieve changes for a domain A through other domains effectively offloads domain A from replicating its transactions everywhere. The change distribution load on a domain is limited by the number of links it uses, not how many domains are in the topology. This load property provides scalability, so the domains in the topology can share the burden of change distribution.

  A catalog service domain can retrieve changes indirectly through other catalog service domains. Consider a line topology with five catalog service domains.

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

  – A pulls changes from B, C, D, and E through B

  – B pulls changes from A and C directly, and changes from D and E through C

  – C pulls changes from B and D directly, and changes from A through B and E through D

  – D pulls changes from C and E directly, and changes from A and B through C

  – E pulls changes from D directly, and changes from A, B, and C through D

  The distribution load on catalog service domains A and E is lowest, because they each have a link only to a single catalog service domain. Domains B, C, and D each have a link to two domains. Thus, the distribution load on domains B, C, and D is double the load on domains A and E. The workload depends on the number of links in each domain, not on the overall number of domains in the topology. Thus, the described distribution of loads would remain constant, even if the line contained 1000 domains.

Multi-master replication performance considerations

Take the following limitations into account when using multi-master replication topologies:

- **Change distribution tuning**, as discussed in the previous section.
- **Replication link performance** WebSphere eXtreme Scale creates a single TCP/IP socket between any pair of JVMs. All traffic between the JVMs occurs through the single socket, including traffic from multi-master replication. Catalog service domains are hosted on at least *n* container JVMs, providing at least *n* TCP links to peer catalog service domains. Thus, the catalog service domains with larger numbers of containers have higher replication performance levels. More containers require more processor and network resources.
- **TCP sliding window tuning and RFC 1323** RFC 1323 support on both ends of a link yields more data for a round trip. This support results in higher throughput, expanding the capacity of the window by a factor of about 16,000.

  Recall that TCP sockets use a sliding window mechanism to control the flow of bulk data. This mechanism typically limits the socket to 64 KB for a round-trip interval. If the round-trip interval is 100 ms, then the bandwidth is limited to 640 KB/second without additional tuning. Fully using the bandwidth available on a link might require tuning that is specific to an operating system. Most operating systems include tuning parameters, including RFC 1323 options, to enhance throughput over high-latency links.

  Several factors can affect replication performance:

  – The speed at which eXtreme Scale retrieves changes.

  - The speed at which eXtreme Scale can service retrieve replication requests.
  - The sliding window capacity.
  - With network buffer tuning on both sides of a link, eXtreme Scale retrieves changes over the socket efficiently.
- **Object Serialization** All data must be serializable. If a catalog service domain is not using `COPY_TO_BYTES`, then the catalog service domain must use Java serialization or ObjectTransformers to optimize serialization performance.
- **Compression** WebSphere eXtreme Scale compresses all data sent between catalog service domains by default. Disabling compression is not currently available.
- **Memory tuning** The memory usage for a multi-master replication topology is largely independent of the number of catalog service domains in the topology.

  Multi-master replication adds a fixed amount of processing per Map entry to handle versioning. Each container also tracks a fixed amount of data for each catalog service domain in the topology. A topology with two catalog service domains uses approximately the same memory as a topology with 50 catalog service domains. WebSphere eXtreme Scale does not use replay logs or similar queues in its implementation. Thus, there is no recovery structure ready in the case that a replication link is unavailable for a substantial period and later restarts.

Related tasks:

"Configuring multiple data center topologies" on page 329
With the multi-master asynchronous replication, you link a set of catalog service domains. The connected catalog service domains are then synchronized using replication over the links. You can define the links using properties files, at run time with Java Management Extensions (JMX) programs, or with command-line utilities. The set of current links for a domain is stored in the catalog service. You can add and remove links without restarting the catalog service domain that hosts the data grid.

Developing custom arbiters for multi-master replication
Change collisions might occur if the same records can be changed simultaneously in two places. In a multi-master replication topology, catalog service domains detect collisions automatically. When a catalog service domain detects a collision, it invokes an arbiter. Typically, collisions are resolved with the default collision arbiter. However, an application can provide a custom collision arbiter.

Interoperability with other products

You can integrate WebSphere eXtreme Scale with other products, such as WebSphere Application Server and WebSphere Application Server Community Edition.

WebSphere Application Server

You can integrate WebSphere Application Server into various aspects of your WebSphere eXtreme Scale configuration. You can deploy data grid applications and use WebSphere Application Server to host container and catalog servers. Or, you can use a mixed environment that has WebSphere eXtreme Scale Client installed in the WebSphere Application Server environment with stand-alone catalog and container servers. You can also use WebSphere Application Server security in your WebSphere eXtreme Scale environment.

WebSphere Business Process Management and Connectivity products

WebSphere Business Process Management and Connectivity products, including WebSphere Integration Developer, WebSphere Enterprise Service Bus, and WebSphere Process Server, integrate with back end systems, such as CICS®, web services, databases, or JMS topics and queues. You can add WebSphere eXtreme Scale to the configuration to cache the output of these back end systems, increasing the overall performance of your configuration.

WebSphere Commerce

WebSphere Commerce can leverage WebSphere eXtreme Scale caching as a replacement to dynamic cache. By eliminating duplicate dynamic cache entries and the frequent invalidation processing necessary to keep the cache synchronized during high stress situations, you can improve performance, scaling, and high availability.

WebSphere Portal

You can persist HTTP sessions from WebSphere Portal into a data grid in WebSphere eXtreme Scale.

WebSphere Application Server Community Edition

WebSphere Application Server Community Edition can share session state, but not in an efficient, scalable manner. WebSphere eXtreme Scale provides a high performance, distributed persistence layer that can be used to replicate state, but does not readily integrate with any application server outside of WebSphere Application Server. You can integrate these two products to provide a scalable session management solution.

WebSphere Real Time

With support for WebSphere Real Time, the industry-leading real-time Java offering, WebSphere eXtreme Scale enables Extreme Transaction Processing (XTP) applications to have more consistent and predictable response times.

Monitoring

WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli® Monitoring and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

Related tasks:

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Using WebSphere eXtreme Scale to enhance WebSphere Portal and IBM Web Content Manager performance

Planning for configuration

Before configuring the hardware or software, understand the following considerations.

Planning for network ports

WebSphere eXtreme Scale servers require several ports to operate.

Important: Avoid hard coding port numbers from the ephemeral range of your operating system. If you set a port that belongs in the ephemeral range, port conflicts can occur.

Catalog service domain

A catalog service domain requires the following ports to be defined:

peerPort
Specifies the port for the high availability (HA) manager to communicate between peer catalog servers over a TCP stack. In WebSphere Application Server, this setting is inherited by the high availability manager port configuration.

clientPort
Specifies the port that peer catalog servers use to access each other's service data. While the value defined for peerPort is used for heartbeat communication between peers that are in the same domain, the clientPort is the port over which actual data gets exchanged. In WebSphere Application Server, this port is set through the catalog service domain configuration.

listenerPort (catalog server)

Specifies the port number to which the Object Request Broker transport protocol binds for communication.

Default: `2809`

Note: When a data grid server is run inside and the ORB transport protocol is being used, another port ORB_LISTENER_ADDRESS must also be opened. The BOOTSTRAP_ADDRESS port forwards requests to this port.

JMXConnectorPort

Defines the Secure Sockets Layer (SSL) port to which the Java Management Extensions (JMX) service binds. Only required if an SSL transport protocol is needed for JMX data.

`SSLPort` (optional)

For secure transport of grid data, the SSL port is used only when the ORB transport protocol is used. If an SSL port is not configured an ephemeral port is chosen at startup, and this can vary each time the catalog server is restarted. When security is enabled, you must use the following argument on the **`startOgServer`** script to configure the Secure Socket Layer (SSL) port: `-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>`.

Container servers

The WebSphere eXtreme Scale container servers also require several ports to operate. By default, an eXtreme Scale container server generates its HA manager port and listener port automatically. For an environment that has a firewall, it is advantageous for you to plan and control ports. For container servers to start with specific ports, you can use the following options in the **`startOgServer`** command.

haManagerPort

Specifies the port that is used by the high availability (HA) manager for heartbeat communication between peer container servers. The **`haManagerPort`** port is only used for peer-to-peer communication between container servers that are in same domain. If the haManagerPort property is not defined, then an ephemeral port is used. In WebSphere Application Server, this setting is inherited by the high availability manager port configuration.

Default: `A dynamic port is chosen`.

listenerPort (container server)

Specifies the port number to which the ORB transport protocol binds for communication.

Default: An ephemeral port is chosen.

Note: When a data grid server is run inside WebSphere Application Server and the ORB transport protocol is being used, another port ORB_LISTENER_ADDRESS must also be opened. The BOOTSTRAP_ADDRESS port forwards requests to this port.

JMXConnectorPort

Defines the Secure Sockets Layer (SSL) port to which the Java Management Extensions (JMX) service binds. Only required if an SSL transport protocol is needed for JMX data.

JMXServicePort

Required only for WebSphere eXtreme Scale in a stand-alone environment. Specifies the port number on which the MBean server listens for communication with Java Management Extensions (JMX).

Default: `1099`

7.1.1+ **xioChannel.xioContainerTCPSecure.Port**

Deprecated: This property is deprecated. The value that is specified by the listenerPort property is used instead.
Specifies the SSL port number of eXtremeIO on the server. This property is used only when the **`transportType`** property is set to `SSL-Supported` or `SSL-Required`.

7.1.1+ **xioChannel.xioContainerTCPNonSecure.Port**

Deprecated: This property is deprecated. The value that is specified by the listenerPort property is used instead.
Specifies the non-secure listener port number of eXtremeIO on the server. If you do not set the value, an ephemeral port is used. This property is used only when the **`transportType`** property is set to `TCP/IP`.

`SSLPort` (optional)

For secure transport of grid data, the SSL port is used only when the ORB transport protocol is used. If an SSL port is not configured an ephemeral port is chosen at startup, and this can vary each time the container server is restarted. When security is enabled, you must use the following argument on the **`startOgServer`** script to configure the Secure Socket Layer (SSL) port: `-jvmArgs -Dcom.ibm.CSI.SSLPort=<sslPort>`.

Proper planning of port control is essential when hundreds of Java virtual machines are started in a server. If a port conflict exists, container servers do not start.

Clients

WebSphere eXtreme Scale clients can receive callbacks from servers when you are using the DataGrid API or other multi-partition operations. Use the **`listenerPort`** property in the client properties file to specify the port on which the client listens for callbacks from the server.

listenerPort (client)

Specifies the port number to which the ORB transport protocol binds for communication. This setting configures the client to communicate with the catalog and container service. If a listener is not configured with the ORB transport protocol, an ephemeral port is chosen at startup. This port can vary each time the client application is started.

Default: An ephemeral port is chosen.

Note: When a data grid client is run inside WebSphere Application Server and the ORB transport protocol is being used, another port ORB_LISTENER_ADDRESS must also be opened. The BOOTSTRAP_ADDRESS port forwards requests to this port.

`SSLPort` (optional)

For secure transport of grid data, the SSL port is used only when the ORB transport protocol is used. When the ORB or XIO transport protocol is used, SSL is an optional configuration. When SSL is enabled with the XIO protocol, it does not use a separate SSL port and sends SSL traffic over the listener port. When SSL is enabled with the ORB transport protocol, both sides can initiate traffic. If an SSL port is not configured an ephemeral port is chosen at startup, and this can vary each time the client is restarted. When security is enabled, you must use the following system property when starting the client process: `-Dcom.ibm.CSI.SSLPort=<sslPort>`.

Ports in WebSphere Application Server

- The **`listenerPort`** value is inherited from the **`BOOTSTRAP_ADDRESS`** value for each WebSphere Application Server application server.
- The **`listenerPort`** value is inherited. The value is different depending on the type of transport you are using:
  - If you are using the ORB transport, the **`BOOTSTRAP_ADDRESS`** and the **`ORB_LISTENER_ADDRESS`** values for each WebSphere Application Server application server are used.
- The **`haManagerPort`** and **`peerPort`** values are inherited from the **`DCS_UNICAST_ADDRESS`** value for each WebSphere Application Server application server.
- The **`JMXServicePort`** and **`JMXConnectorPort`** values are inherited from the **`BOOTSTRAP_ADDRESS`** value for each WebSphere Application Server application server.
- The **`SSLPort`** value is inherited from the **`CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS`** value for each WebSphere Application Server application server.

You can define a catalog service domain in the administrative console. For more information, see "Creating catalog service domains in WebSphere Application Server" on page 299.

You can view the ports for a particular server by clicking one of the following paths in the administrative console:

- WebSphere Application Server Network Deployment Version 6.1: **Servers** > **Application Servers** > ***server_name*** > **Ports** > ***end_point_name***.
- WebSphere Application Server Network Deployment Version 7.0 and later: **Servers** > **Server Types** > **WebSphere Application Servers** > *server_name* > **Ports** > ***port_name***.

Related tasks:

"Configuring ports" on page 334
You must open ports to communicate among servers and with remote servers.

"Configuring ports in stand-alone mode" on page 334
You can configure the necessary ports for servers and clients in an eXtreme scale deployment by using command-line parameters, property files or programmatically. Most examples included in the following sections describe command-line parameters to the **`startOgServer`** script. Equivalent configuration options can also be set in properties files, using the embedded server API or the client API.

"Configuring ports in a WebSphere Application Server environment" on page 337
WebSphere eXtreme Scale catalog services, container servers and clients, when running in WebSphere Application Server processes, utilize ports and services already defined for the process.

7.1.1+ "Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related reference:

xscmd utility reference
You can use the following list of commands as a reference when you are using the **xscmd** utility.

Related information:

Port number settings in WebSphere Application Server versions

Planning to use IBM eXtremeMemory

By configuring eXtremeMemory, you can store objects in native memory instead of on the Java heap. When you configure eXtremeMemory, you can either allow the default amount of memory to be used, or you can calculate the amount of memory that you want to dedicate to eXtremeMemory.

Before you begin

- To learn more about eXtremeMemory, see IBM eXtremeMemory.
- Linux eXtremeMemory is supported on x86 64-bit Linux systems that are using a 64-bit SDK only.
- You must be using map sets that have all the maps configured with `COPY_TO_BYTES` or `COPY_TO_BYTES_RAW` copy modes. If any maps within the map set are not using one of these copy modes, objects are stored on the Java heap 7.1.1 and the Object Request Broker (ORB) is used.
- Linux You must have the shared resource, `libstdc++.so.5`, installed. Use the package installer of your 64-bit Linux distribution to install the required resource file. For more information, see "Troubleshooting IBM eXtremeMemory" on page 630.
- You cannot use eXtremeMemory in the following configuration scenarios:
  - 7.1.1 When you are using container servers that are running in a WebSphere Application Server environment.
  - 7.1.1 When you are using the ReplicationMapListener interface to create an implementation of an event listener for client-side maps that are in replication mode.
  - When you are using custom evictors. The only evictors that are supported with eXtremeMemory are the built-in TTL, LRU, LFU and dynamic cache evictors.
  - 7.1.1 When you are using composite indexes.
  - 7.1.1 When you are using dynamic indexes.
  - When you are using built-in write-behind loaders.
- You must have an existing data grid from which you can determine the total map sizes.
- You must ensure that all data grids have the same eXtremeMemory setting. All container servers must be started with the setting `enableXM=true` or `enableXM=false`. You cannot mix these settings.

About this task

By default, eXtremeMemory uses 25% of the total memory on the system. You can change this default with the **maxXMSize** property, which specifies the number of megabytes to dedicate for use by eXtremeMemory.

Procedure

Optional: Plan the appropriate **maxXMSize** property value to use. This value sets the maximum amount of memory, in megabytes, that is used by the server for eXtremeMemory.

1. In your existing configuration, determine the size per entry. Run the **xscmd -c showMapSizes** command to determine this size.
2. Calculate the **maxXMSize** value. To get maximum total size of entries (*maximum_total_size*), multiply the *size_per_entry* * *maximum_number_of_entries*. Use no more than 60% of **maxXMSize** to account for metadata processing. Multiply *maximum_total_size** `1.65` to get the **maxXMSize** value.
3. For data grids that have a high number of maps or shards, consider the memory usage of empty maps. This usage is about 1 MB per map. You can display the number of maps with the**xscmd -c showMapSizes** command. To compute the maxXMSize, multiply the *number_of_maps* * `1MB`, where the *number_of_maps* is the number of maps that is shown for a single container in the output of the **xscmd -c showMapSizes** command. Add the resulting value to the **maxXMSize** value that you calculated in the previous step.

What to do next

If the **maxXMSize** value is nearing the total physical memory, consider reducing the number of maps or partitions. You can also consider adding more servers to the deployment to reduce the demand on a single server.

Related concepts:

IBM eXtremeMemory
IBM eXtremeMemory enables objects to be stored in native memory instead of the Java heap. By moving objects off the Java heap, you can avoid garbage collection pauses, leading to more constant performance and predicable response times.

Security overview

WebSphere eXtreme Scale can secure data access, including allowing for integration with external security providers.

Note: In an existing non-cached data store such as a database, you likely have built-in security features that you might not need to actively configure or enable. However, after you have cached your data with eXtreme Scale, you must consider the important resulting situation that your backend security features are no longer in effect. You can configureeXtreme Scale security on necessary levels so that your new cached architecture for your data is also secured.
A brief summary of eXtreme Scale security features follows. For more detailed information about configuring security see the *Administration Guide* and the *Programming Guide*.

Distributed security basics

Distributed eXtreme Scale security is based on three key concepts:

Trustable authentication
: The ability to determine the identity of the requester. WebSphere eXtreme Scale supports both client-to-server and server-to-server authentication.

Authorization
: The ability to give permissions to grant access rights to the requester. WebSphere eXtreme Scale supports different authorizations for various operations.

Secure transport
: The safe transmission of data over a network. WebSphere eXtreme Scale supports the Transport Layer Security/Secure Sockets Layer (TLS/SSL) protocols.

Authentication

WebSphere eXtreme Scale supports a distributed client server framework. A client server security infrastructure is in place to secure access to eXtreme Scale servers. For example, when authentication is required by the eXtreme Scale server, an eXtreme Scale client must provide credentials to authenticate to the server. These credentials can be a user name and password pair, a client certificate, a Kerberos ticket, or data that is presented in a format that is agreed upon by client and server.

Authorization

WebSphere eXtreme Scale authorizations are based on subjects and permissions. You can use the Java Authentication and Authorization Services (JAAS) to authorize the access, or you can plug in a custom approach, such as Tivoli Access Manager (TAM), to handle the authorizations. The following authorizations can be given to a client or group:

Map authorization
: Perform insert, read, update, evict, or delete operations on Maps.

ObjectGrid authorization
: Perform object or entity queries and stream queries on ObjectGrid objects.

DataGrid agent authorization
: Allow DataGrid agents to be deployed to an ObjectGrid.

Server side map authorization
: Replicate a server map to client side or create a dynamic index to the server map.

Administration authorization
: Perform administration tasks.

Transport security

To secure the client server communication, WebSphere eXtreme Scale supports TLS/SSL. These protocols provide transport layer security with authenticity, integrity, and confidentiality for a secure connection between an eXtreme Scale client and server.

Grid security

In a secure environment, a server must be able to check the authenticity of another server. WebSphere eXtreme Scale uses a shared secret key string mechanism for this purpose. This secret key mechanism is similar to a shared password. All the eXtreme Scale servers agree on a shared secret string. When a server joins the data grid, the server is challenged to present the secret string. If the secret string of the joining server matches the one in the master server, then the joining server can join the grid. Otherwise, the join request is rejected.

Sending a clear text secret is not secure. The eXtreme Scale security infrastructure provides a SecureTokenManager plug-in to allow the server to secure this secret before sending it. You can choose how you implement the secure operation. WebSphere eXtreme Scale provides an implementation, in which the secure operation is implemented to encrypt and sign the secret.

Java Management Extensions (JMX) security in a dynamic deployment topology

JMX MBean security is supported in all versions of eXtreme Scale. Clients of catalog server MBeans and container server MBeans can be authenticated, and access to MBean operations can be enforced.

Local eXtreme Scale security

Local eXtreme Scale security is different from the distributed eXtreme Scale model because the application directly instantiates and uses an ObjectGrid instance. Your application and eXtreme Scale instances are in the same Java virtual machine (JVM). Because no client-server concept exists in this model, authentication is not supported. Your applications must manage their own authentication, and then pass the authenticated Subject object to the eXtreme Scale. However, the authorization mechanism that is used for the local eXtreme Scale programming model is the same as what is used for the client-server model.

Configuration and programming

For more information about configuring and programming for security, see “Security integration with external providers” on page 597 and Security API.

Related tasks:

“Tutorial: Configuring Java SE security” on page 90
With the following tutorial, you can create a distributed eXtreme Scale environment in a Java Platform, Standard Edition environment.

7.1.1+ “Installing WebSphere eXtreme Scale with the installation wizard” on page 183
You can use the installation wizard to install WebSphere eXtreme Scale for stand-alone or WebSphere Application Server configurations.

Related information:

“Introduction: Integrate WebSphere eXtreme Scale security with WebSphere Application Server using the WebSphere Application Server Authentication plug-ins” on page 106
In this tutorial, you integrate WebSphere eXtreme Scale security with WebSphere Application Server. First, you configure authentication with a simple web application that uses authenticated user credentials from the current thread to connect to the ObjectGrid. Then, you investigate the encryption of data that is transferred between the client and server with transport layer security. To give users varying levels of permissions, you can configure Java Authentication and Authorization Service (JAAS). After completing the configuration, you can use the **xscmd** utility to monitor your data grids and maps.

WebSphere Application Server: Securing applications and their environment

Planning for installation

Before you install the product, you must consider software and hardware requirements and Java environment settings.

Hardware and software requirements

Browse an overview of hardware and operating system requirements. Although you are not required to use a specific level of hardware or operating system for WebSphere eXtreme Scale, formally supported hardware and software options are available on the Systems Requirements page of the product support site. If a conflict exists between the information center and the System Requirements page, the information at the website takes precedence. Prerequisite information in the information center is provided as a convenience only.

See the System Requirements page for the official set of hardware and software requirements.

You can install and deploy the product in Java EE and Java SE environments. You can also bundle the client component with Java EE applications directly without integrating with WebSphere Application Server.

Hardware requirements

WebSphere eXtreme Scale does not require a specific level of hardware. The hardware requirements are dependent on the supported hardware for the Java Platform, Standard Edition installation that you use to run WebSphere eXtreme Scale. If you are using eXtreme Scale with WebSphere Application Server or another Java Platform, Enterprise Edition implementation, the hardware requirements of these platforms are sufficient for WebSphere eXtreme Scale.

Operating system requirements

7.1.1

- **Without the web console**

  eXtreme Scale does not require a specific operating system level. Each Java SE and Java EE implementation requires different operating system levels or fixes for problems that are discovered during the testing of the Java implementation. The levels required by these implementations are sufficient for eXtreme Scale.

- 7.1.1 **With the web console**

  The following requirements apply for each operating system if using the console:
  - Linux: 32 bit or 64 bit JVM
  - Linux PPC: 32 bit JVM only
  - Windows: 32 bit JVM only
  - AIX®: 32 bit JVM only

Web browser requirements

The web console supports the following Web browsers:

- Mozilla Firefox, version 3.5.x and later

- Mozilla Firefox, version 3.6.x and later
- Microsoft Internet Explorer, version 7 or 8

WebSphere Application Server requirements

7.1.1

- WebSphere Application Server Version 6.1.0.39 or later
- WebSphere Application Server Version 7.0.0.19 or later
- WebSphere Application Server Version 8.0.0.1 or later

See the Recommended fixes for WebSphere Application Server for more information.

Java requirements

7.1.1 Other Java EE implementations can use the eXtreme Scale run time as a local instance or as a client to eXtreme Scale servers. To implement Java SE, you must use Version 5 or later.

Related tasks:

"Migrating to WebSphere eXtreme Scale Version 7.1.1" on page 238
With the WebSphere eXtreme Scale installer, you cannot upgrade or modify a previous installation. You must uninstall the previous version before you install the new version. You do not need to migrate your configuration files because they are backward compatible. However, if you changed any of the script files that are shipped with the product, you must reapply these changes to the updated script files.

"Stopping stand-alone servers" on page 472
You can use the `stopOgServer` script to stop eXtreme Scale server processes.

7.1.1 "Installing WebSphere eXtreme Scale with the installation wizard" on page 183
You can use the installation wizard to install WebSphere eXtreme Scale for stand-alone or WebSphere Application Server configurations.

Java SE considerations

WebSphere eXtreme Scale requires Java SE 5, Java SE 6, or Java SE 7. In general, newer versions of Java SE have better functionality and performance.

Supported versions

You can use WebSphere eXtreme Scale with Java SE 5, Java SE 6 7.1.1.1+, and Java SE 7. The version that you use must be currently supported by the Java Runtime Environment (JRE) vendor. If you want to use Secure Sockets Layer (SSL), you must use an IBM Runtime Environment.

IBM Runtime Environment, Java Technology Edition Version 5, Version 6 7.1.1.1+, and Version 7 are supported for general use with the product. Version 6 Service Release 9 Fix Pack 2 is a fully supported JRE that is installed as a part of the stand-alone WebSphere eXtreme Scale and WebSphere eXtreme Scale Client installations in the *wxs_install_root*/`java` directory and is available to be used by both clients and servers. If you are installing WebSphere eXtreme Scale within WebSphere Application Server, you can use the JRE that is included in the

WebSphere Application Server installation. For the web console, you must use IBM Runtime Environment, Java Technology Edition Version 6 Service Release 7 and later service releases only.

WebSphere eXtreme Scale takes advantage of Java Development Kit (JDK) Version 5 , Version 6 7.1.1.1+ , and Version 7 functionality as it becomes available. Generally, newer versions of the Java Development Kit (JDK) and Java SE have better performance and functionality.

For more information, see Supported software.

WebSphere eXtreme Scale features that are dependent on Java SE

Table 2. Features that require Java SE 5, Java SE 6, and Java SE 7. WebSphere eXtreme Scale uses functionality that is introduced in Java SE 5 or Java SE 6 to provide the following product features.

Feature	Supported in Java SE 5 and later service releases	Supported in Java SE Version 6 7.1.1.1+ , Version 7 and later service releases
EntityManager API annotations (Optional: You can also use XML files)	X	X
Java Persistence API (JPA): JPA loader, JPA client loader, and JPA time-based updater	X	X
Memory-based eviction (uses MemoryPoolMXBean)	X	X
Instrumentation agents: • `wxssizeagent.jar`: Increases the accuracy of the used bytes map metrics. • `ogagent.jar`: Increases the performance of field-access entities.	X	X
Web console for monitoring		X

Upgrading the JDK in WebSphere eXtreme Scale

Common questions about the upgrade process for releases of WebSphere eXtreme Scale in both stand-alone and WebSphere Application Server environments follow:

- How do I upgrade the JDK that is included with WebSphere eXtreme Scale for WebSphere Application Server?

  You need to use the JDK upgrade process that is made available by WebSphere Application Server. For more information, see http://www-304.ibm.com/support/docview.wss?uid=swg21427178.
- Which version of the JDK should I use when using WebSphere eXtreme Scale in a WebSphere Application Server environment?

  You can use any level of JDK that is supported by WebSphere Application Server, for the supported version of WebSphere Application Server.

Related reference:

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

Tuning the IBM virtual machine for Java

Java EE considerations

As you prepare to integrate WebSphere eXtreme Scale in a Java Platform, Enterprise Edition environment, consider certain items, such as versions, configuration options, requirements and limitations, and application deployment and management.

Running eXtreme Scale applications in a Java EE environment

A Java EE application can connect to a remote eXtreme Scale application. Additionally, the WebSphere Application Server environment supports starting an eXtreme Scale server as an application starts in the application server.

If you use an XML file to create an ObjectGrid instance, and the XML file is in the module of the enterprise archive (EAR) file, access the file by using the getClass().getClassLoader().getResource("META-INF/objGrid.xml") method to obtain a URL object to use to create an ObjectGrid instance. Substitute the name of the XML file that you are using in the method call.

You can use startup beans for an application to bootstrap an ObjectGrid instance when the application starts, and to destroy the instance when the application stops. A startup bean is a stateless session bean with a com.ibm.websphere.startupservice.AppStartUpHome remote location and a com.ibm.websphere.startupservice.AppStartUp remote interface. The remote interface has two methods: the start method and the stop method. Use the start method to bootstrap the instance, and use the stop method to destroy the instance. The application uses the ObjectGridManager.getObjectGrid method to maintain a reference to the instance. See the information about accessing an ObjectGrid with the ObjectGridManager in the *Programming Guide* for more information.

Using class loaders

When application modules that use different class loaders share a single ObjectGrid instance in a Java EE application, verify the objects that are stored in eXtreme Scale and the plug-ins for the product are in a common loader in the application.

Managing the life cycle of ObjectGrid instances in a servlet

To manage the life cycle of an ObjectGrid instance in a servlet, you can use the init method to create the instance and the destroy method to remove the instance. If the instance is cached, it is retrieved and manipulated in the servlet code. See the information about accessing an ObjectGrid with the ObjectGridManager interface in the *Programming Guide* for more information.

Related reference:

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port

numbers, and so on.

Related information:

Tuning the IBM virtual machine for Java

Directory conventions

The following directory conventions are used throughout the documentation to reference special directories such as *wxs_install_root* and *wxs_home*. You access these directories during several different scenarios, including during installation and use of command-line tools.

wxs_install_root
The *wxs_install_root* directory is the root directory where WebSphere eXtreme Scale product files are installed. The *wxs_install_root* directory can be the directory in which the trial archive is extracted or the directory in which the WebSphere eXtreme Scale product is installed.

- Example when extracting the trial:

  Example: `/opt/IBM/WebSphere/eXtremeScale`
- Example when WebSphere eXtreme Scale is installed to a stand-alone directory:

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

  Windows **Example:** `C:\Program Files\IBM\WebSphere\eXtremeScale`
- Example when WebSphere eXtreme Scale is integrated with WebSphere Application Server:

  Example: `/opt/IBM/WebSphere/AppServer`

wxs_home
The *wxs_home* directory is the root directory of the WebSphere eXtreme Scale product libraries, samples, and components. This directory is the same as the *wxs_install_root* directory when the trial is extracted. For stand-alone installations, the *wxs_home* directory is the `ObjectGrid` subdirectory within the *wxs_install_root* directory. For installations that are integrated with WebSphere Application Server, this directory is the `optionalLibraries/ObjectGrid` directory within the *wxs_install_root* directory.

- Example when extracting the trial:

  Example: `/opt/IBM/WebSphere/eXtremeScale`
- Example when WebSphere eXtreme Scale is installed to a stand-alone directory:

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

  Windows **Example:** *wxs_install_root*`\ObjectGrid`
- Example when WebSphere eXtreme Scale is integrated with WebSphere Application Server:

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

was_root
The *was_root* directory is the root directory of a WebSphere Application Server installation:

Example: `/opt/IBM/WebSphere/AppServer`

restservice_home
The *restservice_home* directory is the directory in which the WebSphere

eXtreme Scale REST data service libraries and samples are located. This directory is named `restservice` and is a subdirectory under the *`wxs_home`* directory.

- Example for stand-alone deployments:

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

  Example: *wxs_home*`\restservice`
- Example for WebSphere Application Server integrated deployments:

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

`tomcat_root`
The *`tomcat_root`* is the root directory of the Apache Tomcat installation.

Example: `/opt/tomcat5.5`

`wasce_root`
The *`wasce_root`* is the root directory of the WebSphere Application Server Community Edition installation.

Example: `/opt/IBM/WebSphere/AppServerCE`

`java_home`
The *`java_home`* is the root directory of a Java Runtime Environment (JRE) installation.

UNIX **Example:** `/opt/IBM/WebSphere/eXtremeScale/java`

Windows **Example:** *wxs_install_root*`\java`

`samples_home`
The *`samples_home`* is the directory in which you extract the sample files that are used for tutorials.

UNIX **Example:** *wxs_home*`/samples`

Windows **Example:** *wxs_home*`\samples`

`dvd_root`
The *`dvd_root`* directory is the root directory of the DVD that contains the product.

Example: `dvd_root/docs/`

`equinox_root`
The *`equinox_root`* directory is the root directory of the Eclipse Equinox OSGi framework installation.

Example:`/opt/equinox`

`user_home`
The *`user_home`* directory is the location where user files are stored, such as security profiles.

Windows `c:\Documents and Settings\`*`user_name`*

UNIX `/home/`*`user_name`*

Planning environment capacity

If you have an initial data set size and a projected data set size, you can plan the capacity that you need to run WebSphere eXtreme Scale. By using these planning exercises, you can deploy WebSphere eXtreme Scale efficiently for future changes and maximize the elasticity of the data grid, which you would not have with a different scenario such as an in-memory database or other type of database.

Sizing memory and partition count calculation

You can calculate the amount of memory and partitions needed for your specific configuration.

Attention: This topic applies when you are **not** using the COPY_TO_BYTES copy mode. If you are using the COPY_TO_BYTES mode, then the memory size is much less and the calculation procedure is different. For more information about this mode, see Tuning the copy mode.

WebSphere eXtreme Scale stores data within the address space of Java virtual machines (JVM). Each JVM provides processor space for servicing create, retrieve, update, and delete calls for the data that is stored in the JVM. In addition, each JVM provides memory space for data entries and replicas. Java objects vary in size, therefore you must make a measurement to make an estimate of how much memory you need.

To size the memory that you need, load your application data into a single JVM. When the heap usage reaches 60%, note the number of objects that are used. This number is the maximum recommended object count for each of your Java virtual machines. To get the most accurate sizing, use realistic data and include any defined indexes in your sizing because indexes also consume memory. The best way to size memory use is to run garbage collection **`verbosegc`** output because this output gives you the numbers after garbage collection. You can query the heap usage at any given point through MBeans or programmatically, but those queries give you only a current snapshot of the heap. This snapshot might include uncollected garbage, so using that method is not an accurate indication of the consumed memory.

Scaling up the configuration

Number of shards per partition (numShardsPerPartition value)

To calculate the number of shards per partition, or the numShardsPerPartition value, add 1 for the primary shard plus the total number of replica shards you want. For more information about partitioning, see Partitioning.

```
numShardsPerPartition = 1 + total_number_of_replicas
```

Number of Java virtual machines (minNumJVMs value)

To scale up your configuration, first decide on the maximum number of objects that need to be stored in total. To determine the number of Java virtual machines you need, use the following formula:

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

Round this value up to the nearest integer value.

Number of shards (numShards value)

At the final growth size, use 10 shards for each JVM. As described before, each JVM has one primary shard and (N-1) shards for the replicas, or in this case, nine replicas. Because you already have a number of Java virtual machines to store the data, you can multiply the number of Java virtual machines by 10 to determine the number of shards:

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

Number of partitions If a partition has one primary shard and one replica shard, then the partition has two shards (primary and replica). The number of partitions is the shard count divided by 2, rounded up to the nearest prime number. If the partition has a primary and two replicas, then the number of partitions is the shard count divided by 3, rounded up to the nearest prime number.

```
numPartitions = numShards / numShardsPerPartition
```

Example of scaling

In this example, the number of entries begins at 250 million. Each year, the number of entries grows about 14%. After seven years, the total number of entries is 500 million, so you must plan your capacity accordingly. For high availability, a single replica is needed. With a replica, the number of entries doubles, or 1,000,000,000 entries. As a test, 2 million entries can be stored in each JVM. Using the calculations in this scenario the following configuration is needed:

- 500 Java virtual machines to store the final number of entries.
- 5000 shards, calculated by multiplying 500 Java virtual machines by 10.
- 2500 partitions, or 2503 as the next highest prime number, calculated by taking the 5000 shards, divided by two for primary and replica shards.

Starting configuration

Based on the previous calculations, start with 250 Java virtual machines and grow toward 500 Java virtual machines over five years. With this configuration, you can manage incremental growth until you arrive at the final number of entries.

In this configuration, about 200,000 entries are stored per partition (500 million entries divided by 2503 partitions). Set the **`numberOfBuckets`** parameter on the map that holds the entries to the closest higher prime number, in this example 70887, which keeps the ratio around three.

When the maximum number of Java virtual machines is reached

When you reach your maximum number of 500 Java virtual machines, you can still grow your data grid. As the number of Java virtual machines grows beyond 500, the shard count begins to drop below 10 for each JVM, which is below the recommended number. The shards start to become larger, which can cause problems. Repeat the sizing process considering future growth again, and reset the partition count. This practice requires a full data grid restart, or an outage of your data grid.

Number of servers

Attention: Do not use paging on a server under any circumstances.

A single JVM uses more memory than the heap size. For example, 1 GB of heap for a JVM actually uses 1.4 GB of real memory. Determine the available free RAM on the server. Divide the amount of RAM by the memory per JVM to get the maximum number of Java virtual machines on the server.

Sizing CPU per partition for transactions

Although a major functionality of eXtreme Scale is its ability for elastic scaling, it is also important to consider sizing and to adjust the ideal number of CPUs to scale up.

Processor costs include:
- Cost of servicing create, retrieve, update, and delete operations from clients
- Cost of replication from other Java virtual machines
- Cost of invalidation
- Cost of eviction policy
- Cost of garbage collection
- Cost of application logic
- Cost of serialization

Java virtual machines per server

Use two servers and start the maximum JVM count per server. Use the calculated partition counts from the previous section. Then, preload the Java virtual machines with enough data to fit on these two computers only. Use a separate server as a client. Run a realistic transaction simulation against this data grid of two servers.

To calculate the baseline, try to saturate the processor usage. If you cannot saturate the processor, then it is likely that the network is saturated. If the network is saturated, add more network cards and round robin the Java virtual machines over the multiple network cards.

Run the computers at 60% processor usage, and measure the create, retrieve, update, and delete transaction rate. This measurement provides the throughput on two servers. This number doubles with four servers, doubles again at 8 servers, and so on. This scaling assumes that the network capacity and client capacity is also able to scale.

As a result, eXtreme Scale response time should be stable as the number of servers is scaled up. The transaction throughput should scale linearly as computers are added to the data grid.

Sizing CPUs for parallel transactions

Single-partition transactions have throughput scaling linearly as the data grid grows. Parallel transactions are different from single-partition transactions because they touch a set of the servers (this can be all of the servers).

If a transaction touches all of the servers, then the throughput is limited to the throughput of the client that initiates the transaction or the slowest server touched. Larger data grids spread the data out more and provide more processor space, memory, network, and so on. However, the client must wait for the slowest server to respond, and the client must consume the results of the transaction.

When a transaction touches a subset of the servers, M out of N servers get a request. The throughput is then N divided by M times faster than the throughput of the slowest server. For example, if you have 20 servers and a transaction that touches 5 servers, then the throughput is 4 times the throughput of the slowest server in the data grid.

When a parallel transaction completes, the results are sent to the client thread that started the transaction. This client must then aggregate the results single threaded. This aggregation time increases as the number of servers touched for the transaction grows. However, this time depends on the application because it is possible that each server returns a smaller result as the data grid grows.

Typically, parallel transactions touch all of the servers in the data grid because partitions are uniformly distributed over the grid. In this case, throughput is limited to the first case.

Summary

With this sizing, you have three metrics, as follows.
- Number of partitions.
- Number of servers that are needed for the memory that is required.
- Number of servers that are needed for the required throughput.

If you need 10 servers for memory requirements, but you are getting only 50% of the needed throughput because of the processor saturation, then you need twice as many servers.

For the highest stability, you should run your servers at 60% processor loading and JVM heaps at 60% heap loading. Spikes can then drive the processor usage to 80–90%, but do not regularly run your servers higher than these levels.

Dynamic cache capacity planning

The Dynamic Cache API is available to Java EE applications that are deployed in WebSphere Application Server. You can use the dynamic cache to cache business data, generated HTML, or to synchronize the cached data in the cell by using the data replication service (DRS).

Overview

All dynamic cache instances created with the WebSphere eXtreme Scale dynamic cache provider are highly available by default. The level and memory cost of high availability depends on the topology used.

When using the embedded topology, the cache size is limited to the amount of free memory in a single server process, and each server process stores a full copy of the cache. As long as a single server process continues to run, the cache survives. The cache data will only be lost if all servers that access the cache are shut down.

For caching using the embedded partitioned topology, the cache size is limited to an aggregate of the free space available in all server processes. By default, the eXtreme Scale dynamic cache provider uses 1 replica for every primary shard, so each piece of cached data is stored twice.

Use the following formula A to determine the capacity of an embedded partitioned cache.

Formula A

F * C / (1 + R) = M

Where:
- F = Free memory per container process
- C = number of containers
- R = number of replicas
- M = Total size of the cache

For a WebSphere Application Server Network Deployment data grid that has 256 MB of available space in each process, with 4 server processes total, a cache instance across all of those servers could store up to 512 megabytes of data. In this mode, the cache can survive one server crashing without losing data. Also, up to two servers could be shut down sequentially without losing any data. So, for the previous example, the formula is as follows:

256mb * 4 containers/ (1 primary + 1 replica) = 512mb.

Caches using the remote topology have similar sizing characteristics as caches using embedded partitioned, but they are limited by the amount of available space in all eXtreme Scale container processes.

In remote topologies, it is possible to increase the number of replicas to provide a higher level of availability at the cost of additional memory overhead. In most dynamic cache applications this should be unnecessary, but you can edit the `dynacache-remote-deployment.xml` file to increase the number of replicas.

Use the following formulas, B and C, to determine the effect of adding more replicas on the high availability of the cache.

Formula B

N = Minimum(T -1, R)

Where:
- N = the number of processes that can crash simultaneously
- T = the total number of containers
- R = the total number of replicas

Formula C

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

Where:
- T = Total number containers
- N = Total number of replicas
- m = minimum number of containers needed to support the cache data.

For performance tuning with the dynamic cache provider, see “Tuning the dynamic cache provider” on page 581.

Cache sizing

Before an application using the WebSphere eXtreme Scale dynamic cache provider can be deployed, the general principals described in the previous section should be combined with the environmental data for the production systems. The first figure to establish is the total number of container processes and the amount of available memory in each process to hold cache data. When using the embedded topology, the cache containers will be co-located inside of the WebSphere Application server processes, so there is one container for each server that is sharing the cache. Determining the memory overhead of the application without caching enabled and the WebSphere Application Server is the best way to figure out how much space is available in the process. This can be done by analyzing verbose garbage collection data. When using the remote topology, this information can be found by looking at the verbose garbage collection output of a newly started standalone container that has not yet been populated with cache data. The last thing to keep in mind when figuring out how much space per process is available for cache data, is to reserve some heap space for garbage collection. The overhead of the container, WebSphere Application Server or stand-alone, plus the size reserved for the cache should not be more than 70% of the total heap.

After this information is collected, the values can be plugged into formula A, described previously, to determine the maximum size for the partitioned cache. Once the maximum size is known, the next step is to determine the total number of cache entries that can be supported, which requires determining the average size per cache entry. The simple way of doing this is to add 10% to the size of the customer object. See theTuning guide for dynamic cache and data replication service for more in depth information on sizing cache entries when using dynamic cache.

When compression is enabled it affects the size of the customer object, not the overhead of the caching system. Use the following formula to determine the size of a cached object when using compression:

$S = O * C + O * 0.10$

Where:
- S = Average size of cached object
- O = Average size of un-compressed customer object
- C = Compression ratio expressed as a fraction.

So, a 2 to 1 compression ratio is 1/2 = 0.50. Smaller is better for this value. If the object being stored is a normal POJO mostly full of primitive types, then assume a compression ratio of 0.60 to 0.70. If the object cached is a Servlet, JSP, or WebServices object, the optimal method for determining the compression ratio is to compress a representative sample with a ZIP compression utility. If this is not possible, then a compression ratio of 0.2 to 0.35 is common for this type of data.

Next, use this information to determine the total number of cache entries that can be supported. Use the following D formula:

Formula D

$T = S / A$

Where:

- T= Total number of cache entries
- S = Total size available for cache data as computed using formula A
- A = Average size of each cache entry

Finally, you must set the cache size on the dynamic cache instance to enforce this limit. The WebSphere eXtreme Scale dynamic cache provider differs from the default dynamic cache provider in this regard. Use the following formula to determine the value to set for the cache size on the dynamic cache instance. Use the following E formula:

Formula E

Cs = Ts / Np

Where:

- Ts = Total target size for the cache
- Cs = Cache Size setting to set on the dynamic cache instance
- Np = number of partitions. The default is 47.

Set the size of the dynamic cache instance to a value calculated by formula E on each server that shares the cache instance.

Related tasks:

"Configuring the dynamic cache provider for WebSphere eXtreme Scale" on page 396
Installing and configuring the dynamic cache provider for eXtreme Scale depends on what your requirements are and the environment you have set up.

Chapter 3. Tutorials

You can use tutorials to help you understand product usage scenarios, including entity manager, queries, and security.

Tutorial: Querying a local in-memory data grid

You can develop a local in-memory ObjectGrid that can store order information for a website, and use the ObjectQuery API to query the data grid.

Before you begin

Be sure to have `objectgrid.jar` file in the classpath.

About this task

Each step in the tutorial builds on the previous step. Follow each of the steps to build a simple Java Platform, Standard Edition Version 5 or later application that uses an in-memory, local data grid.

ObjectQuery tutorial - step 1

With the following steps, you can continue to develop a local, in-memory ObjectGrid that stores order information for an online retail store using the ObjectMap APIs. You define a schema for the map and run a query against the map.

Procedure

1. Create an ObjectGrid with a map schema.

   Create an ObjectGrid with one map schema for the map, then insert an object into the cache and later retrieve it using a simple query.

   `OrderBean.java`

```
public class OrderBean implements Serializable {
    String orderNumber;
    java.util.Date date;
    String customerName;
    String itemName;
    int quantity;
    double price;
}
```

2. Define the primary key.

   The previous code shows an OrderBean object. This object implements the java.io.Serializable interface because all objects in the cache must (by default) be Serializable.

   The orderNumber attribute is the primary key of the object. The following example program can be run in stand-alone mode. You should follow this tutorial in an Eclipse Java project that has the `objectgrid.jar` file added to the class path.

   `Application.java`

```
package querytutorial.basic.step1;

import java.util.Iterator;

import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.config.QueryConfig;
import com.ibm.websphere.objectgrid.config.QueryMapping;
import com.ibm.websphere.objectgrid.query.ObjectQuery;

public class Application
{

    static public void main(String [] args) throws Exception
    {
        ObjectGrid og = ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
        og.defineMap("Order");

        // Define the schema
        QueryConfig queryCfg = new QueryConfig();
        queryCfg.addQueryMapping(new QueryMapping("Order", OrderBean.class.getName(),
     "orderNumber", QueryMapping.FIELD_ACCESS));
        og.setQueryConfig(queryCfg);

        Session s = og.getSession();
        ObjectMap orderMap = s.getMap("Order");

        s.begin();
        OrderBean o = new OrderBean();
        o.customerName = "John Smith";
        o.date = new java.util.Date(System.currentTimeMillis());
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;
        orderMap.put(o.orderNumber, o);
        s.commit();

        s.begin();
        ObjectQuery query = s.createObjectQuery("SELECT o FROM Order o WHERE o.itemName='Widget'");
        Iterator result = query.getResultIterator();
        o = (OrderBean) result.next();
        System.out.println("Found order for customer: " + o.customerName);
        s.commit();
 // Close the session (optional in Version 7.1.1 and later) for improved performance
 s.close();
}
}
```

This eXtreme Scale application first initializes a local ObjectGrid with an automatically generated name. Next, the application creates a BackingMap and a QueryConfig that defines what Java type is associated with the map, the name of the field that is the primary key for the map, and how to access the data in the object. You then obtain a Session to get the ObjectMap instance and insert an OrderBean object into the map in a transaction.

After the data is committed into the cache, you can use ObjectQuery to find the OrderBean using any of the persistent fields in the class. Persistent fields are those that do not have the transient modifier. Because you did not define any indexes on the BackingMap, ObjectQuery must scan each object in the map using Java reflection.

What to do next

“ObjectQuery tutorial - step 2” demonstrates how an index can be used to optimize the query.

ObjectQuery tutorial - step 2

With the following steps, you can continue to create an ObjectGrid with one map and an index, along with a schema for the map. Then you can insert an object into the cache and later retrieve it using a simple query.

Before you begin

Be sure that you have completed "ObjectQuery tutorial - step 1" on page 73 before proceeding with this step of the tutorial.

Procedure

Schema and index

Application.java

```
// Create an index
    HashIndex idx= new HashIndex();
    idx.setName("theItemName");
    idx.setAttributeName("itemName");
    idx.setRangeIndex(true);
    idx.setFieldAccessAttribute(true);
    orderBMap.addMapIndexPlugin(idx);
}
```

The index must be a com.ibm.websphere.objectgrid.plugins.index.HashIndex instance with the following settings:

- The Name is arbitrary, but must be unique for a given BackingMap.
- The AttributeName is the name of the field or bean property which the indexing engine uses to introspect the class. In this case, it is the name of the field for which you will create an index.
- RangeIndex must always be true.
- FieldAccessAttribute should match the value set in the QueryMapping object when the query schema was created. In this case, the Java object is accessed using the fields directly.

When a query runs that filters on the itemName field, the query engine automatically uses the defined index. Using the index allows the query to run much faster and a map scan is not needed. The next step demonstrates how an index can be used to optimize the query.
Next step

ObjectQuery tutorial - step 3

With the following step, you can create an ObjectGrid with two maps and a schema for the maps with a relationship, then insert objects into the cache and later retrieve them using a simple query.

Before you begin

Be sure you have completed "ObjectQuery tutorial - step 2" on page 74 prior to proceeding with this step.

About this task

In this example, there are two maps, each with a single Java type mapped to it. The Order map has OrderBean objects and the Customer map has CustomerBean objects in it.

Procedure

Define maps with a relationship.

OrderBean.java

```
public class OrderBean implements Serializable {
    String orderNumber;
    java.util.Date date;
    String customerId;
    String itemName;
    int quantity;
    double price;
}
```

The OrderBean no longer has the customerName in it. Instead, it has the customerId, which is the primary key for the CustomerBean object and the Customer map.

CustomerBean.java

```
public class CustomerBean implements Serializable{
    private static final long serialVersionUID = 1L;
    String id;
    String firstName;
    String surname;
    String address;
    String phoneNumber;
}
```

The relationship between the two types or Maps follows:

Application.java

```
public class Application
{

    static public void main(String [] args)
        throws Exception
    {
        ObjectGrid og = ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
        og.defineMap("Order");
        og.defineMap("Customer");

        // Define the schema
        QueryConfig queryCfg = new QueryConfig();
        queryCfg.addQueryMapping(new QueryMapping(
            "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
        queryCfg.addQueryMapping(new QueryMapping(
            "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
        queryCfg.addQueryRelationship(new QueryRelationship(
             OrderBean.class.getName(), CustomerBean.class.getName(), "customerId", null));
        og.setQueryConfig(queryCfg);

        Session s = og.getSession();
        ObjectMap orderMap = s.getMap("Order");
        ObjectMap custMap = s.getMap("Customer");

        s.begin();
        CustomerBean cust = new CustomerBean();
        cust.address = "Main Street";
        cust.firstName = "John";
        cust.surname = "Smith";
        cust.id = "C001";
        cust.phoneNumber = "5555551212";
        custMap.insert(cust.id, cust);

        OrderBean o = new OrderBean();
        o.customerId = cust.id;
        o.date = new java.util.Date();
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;
        orderMap.insert(o.orderNumber, o);
        s.commit();

        s.begin();
        ObjectQuery query = s.createObjectQuery(
            "SELECT c FROM Order o JOIN o.customerId as c WHERE o.itemName='Widget'");
        Iterator result = query.getResultIterator();
        cust = (CustomerBean) result.next();
```

```
       System.out.println("Found order for customer: " + cust.firstName + " " + cust.surname);
 s.commit();
    // Close the session (optional in Version 7.1.1 and later) for improved performance
 s.close();
    }
}
```

The equivalent XML in the ObjectGrid deployment descriptor follows:

```
<?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="Order"/>
      <backingMap name="Customer"/>

      <querySchema>
         <mapSchemas>
           <mapSchema
             mapName="Order"
             valueClass="com.mycompany.OrderBean"
             primaryKeyField="orderNumber"
             accessType="FIELD"/>
           <mapSchema
             mapName="Customer"
             valueClass="com.mycompany.CustomerBean"
             primaryKeyField="id"
             accessType="FIELD"/>
         </mapSchemas>
         <relationships>
           <relationship
             source="com.mycompany.OrderBean"
             target="com.mycompany.CustomerBean"
             relationField="customerId"/>
         </relationships>
      </querySchema>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

What to do next

"ObjectQuery tutorial - step 4," expands the current step by including field and property access objects and additional relationships.

ObjectQuery tutorial - step 4

The following step shows how to create an ObjectGrid with four maps and a schema for the maps. Some of the maps maintain a one-to-one (unidirectional) and one-to-many (bidirectional) relationship. After creating the maps, you can then run the sample `Application.java` program to insert objects into the cache and run queries to retrieve these objects.

Before you begin

Be sure to have completed "ObjectQuery tutorial - step 3" on page 75 prior to continuing with the current step.

About this task

You are required to create four JAVA classes. These are the maps for the ObjectGrid:

- OrderBean.java
- OrderLineBean.java
- CustomerBean.java
- ItemBean.java

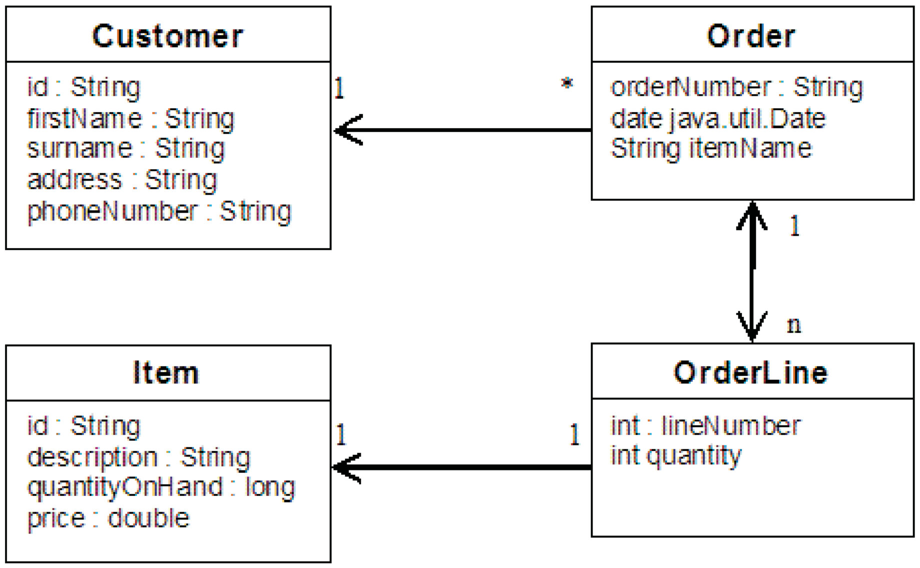


Figure 19. Order Schema. An Order schema has a one-to-one relationship with Customer and a one-to-many relationship with OrderLine. The OrderLine map has a one-to-one relationship with Item and includes the quantity ordered.

After creating these JAVA classes with these relationships, you can then run the sample `Application.java` program. This program lets you insert objects into the cache and retrieve these using several queries.

Procedure

1. Create the following JAVA classes:

   `OrderBean.java`

   ```
   public class OrderBean implements Serializable {
      String orderNumber;
      java.util.Date date;
      String customerId;
      String itemName;
      List<Integer> orderLines;
   }
   ```

   `OrderLineBean.java`

   ```
   public class OrderLineBean implements Serializable {
     int lineNumber;
     int quantity;
     String orderNumber;
     String itemId;
   }
   ```

   `CustomerBean.java`

   ```
   public class CustomerBean implements Serializable{
     String id;
     String firstName;
   ```

```
  String surname;
  String address;
  String phoneNumber;
}
```

ItemBean.java

```
public class ItemBean implements Serializable {
  String id;
  String description;
  long quantityOnHand;
  double price;
}
```

2. After creating the classes, you can run the sample Application.java:

Application.java

```
 public class Application static public void main(String [] args)throws Exception
       // Configure programatically
    objectGrid og = ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
       og.defineMap("Order");
       og.defineMap("Customer");
       og.defineMap("OrderLine");
       og.defineMap("Item");

       // Define the schema
       QueryConfig queryCfg = new QueryConfig();
       queryCfg.addQueryMapping(new QueryMapping("Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
       queryCfg.addQueryMapping(new QueryMapping("Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
       queryCfg.addQueryMapping(new QueryMapping("OrderLine", OrderLineBean.class.getName(), "lineNumber", QueryMapping.FIELD_ACCESS));
       queryCfg.addQueryMapping(new QueryMapping("Item", ItemBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
       queryCfg.addQueryRelationship(new QueryRelationship(OrderBean.class.getName(), CustomerBean.class.getName(), "customerId", null));
       queryCfg.addQueryRelationship(new QueryRelationship(OrderBean.class.getName(), OrderLineBean.class.getName(),
   "orderLines", "lineNumber"));
       queryCfg.addQueryRelationship(new QueryRelationship(OrderLineBean.class.getName(), ItemBean.class.getName(), "itemId", null));
       og.setQueryConfig(queryCfg);

       // Get session and maps;
       Session s = og.getSession();
       ObjectMap orderMap = s.getMap("Order");
       ObjectMap custMap = s.getMap("Customer");
       ObjectMap itemMap = s.getMap("Item");
       ObjectMap orderLineMap = s.getMap("OrderLine");

       // Add data
       s.begin();
       CustomerBean aCustomer = new CustomerBean();
       aCustomer.address = "Main Street";
       aCustomer.firstName = "John";
       aCustomer.surname = "Smith";
       aCustomer.id = "C001";
       aCustomer.phoneNumber = "5555551212";
       custMap.insert(aCustomer.id, aCustomer);

       // Insert an order with a reference to the customer, but without any OrderLines yet.
       // Because we are using CopyMode.COPY_ON_READ_AND_COMMIT, the
       // insert won't be copied into the backing map until commit time, so
       // the reference is still good.

   OrderBean anOrder = new OrderBean();
   anOrder.customerId = aCustomer.id;
   anOrder.date = new java.util.Date();
   anOrder.itemName = "Widget";
   anOrder.orderNumber = "1";
   anOrder.orderLines = new ArrayList();
   orderMap.insert(anOrder.orderNumber, anOrder);

       ItemBean anItem = new ItemBean();
       anItem.id = "AC0001";
       anItem.description = "Description of widget";
       anItem.quantityOnHand = 100;
       anItem.price = 1000.0;
       itemMap.insert(anItem.id, anItem);

   // Create the OrderLines and add the reference to the Order
       OrderLineBean anOrderLine = new OrderLineBean();
       anOrderLine.lineNumber = 99;
       anOrderLine.itemId = anItem.id;
       anOrderLine.orderNumber = anOrder.orderNumber;
       anOrderLine.quantity = 500;
       orderLineMap.insert(anOrderLine.lineNumber, anOrderLine);
       anOrder.orderLines.add(Integer.valueOf(anOrderLine.lineNumber));

   anOrderLine = new OrderLineBean();
       anOrderLine.lineNumber = 100;
```

```
        anOrderLine.itemId = anItem.id;
        anOrderLine.orderNumber = anOrder.orderNumber;
        anOrderLine.quantity = 501;
        orderLineMap.insert(anOrderLine.lineNumber, anOrderLine);
        anOrder.orderLines.add(Integer.valueOf(anOrderLine.lineNumber));
        s.commit();

    s.begin();
        // Find all customers who have ordered a specific item.
        ObjectQuery query = s.createObjectQuery("SELECT c FROM Order o JOIN o.customerId as c WHERE o.itemName='Widget'");
        Iterator result = query.getResultIterator();
        aCustomer = (CustomerBean) result.next();
        System.out.println("Found order for customer: " + aCustomer.firstName + " " + aCustomer.surname);
        s.commit();

    s.begin();
        // Find all OrderLines for customer C001.
        // The query joins are expressed on the foreign keys.
        query = s.createObjectQuery("SELECT ol FROM Order o JOIN o.customerId as c JOIN o.orderLines as ol WHERE c.id='C001'");
        result = query.getResultIterator();
        System.out.println("Found OrderLines:");
        while(result.hasNext()) {
            anOrderLine = (OrderLineBean) result.next();
            System.out.println(anOrderLine.lineNumber + ", qty=" + anOrderLine.quantity);
        }
 // Close the session (optional in Version 7.1.1 and later) for improved performance
 s.close();
   }
}
```

3. Using the XML configuration below (in the ObjectGrid deployment descriptor) is equivalent to the programmatic approach above.

```
<?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="Order"/>
  <backingMap name="Customer"/>
  <backingMap name="OrderLine"/>
  <backingMap name="Item"/>

<querySchema>
 <mapSchemas>
  <mapSchema
  mapName="Order"
  valueClass="com.mycompany.OrderBean"
  primaryKeyField="orderNumber"
  accessType="FIELD"/>
  <mapSchema
  mapName="Customer"
  valueClass="com.mycompany.CustomerBean"
  primaryKeyField="id"
  accessType="FIELD"/>
  <mapSchema
  mapName="OrderLine"
  valueClass="com.mycompany.OrderLineBean"
  primaryKeyField="
  lineNumber"
  accessType="FIELD"/>
  <mapSchema
  mapName="Item"
  valueClass="com.mycompany.ItemBean"
  primaryKeyField="id"
  accessType="FIELD"/>
  </mapSchemas>

<relationships>
 <relationship
  source="com.mycompany.OrderBean"
  target="com.mycompany.CustomerBean"
  relationField="customerId"/>
 <relationship
  source="com.mycompany.OrderBean"
  target="com.mycompany.OrderLineBean"
  relationField="orderLines"
  invRelationField="lineNumber"/>
 <relationship
  source="com.mycompany.OrderLineBean"
  target="com.mycompany.ItemBean"
  relationField="itemId"/>
    </relationships>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Tutorial: Storing order information in entities

The tutorial for the entity manager shows you how to use WebSphere eXtreme Scale to store order information on a Web site. You can create a simple Java Platform, Standard Edition 5 application that uses an in-memory, local data grid. The entities use Java SE 5 annotations and generics.

Before you begin

Ensure that you have met the following requirements before you begin the tutorial:

- You must have Java SE 5.
- You must have the `objectgrid.jar` file in your classpath.

Related concepts:

Caching objects with no relationships involved (ObjectMap API)
ObjectMaps are like Java Maps that allow data to be stored as key-value pairs. ObjectMaps provide a simple and intuitive approach for the application to store data. An ObjectMap is ideal for caching objects that have no relationships involved. If object relationships are involved, then you should use the EntityManager API.

Related information:

API documentation

"Getting started tutorial lesson 2.1: Creating a client application" on page 3
To insert, delete, update, and retrieve data from your data grid, you must write a client application. The getting started sample includes a client application that you can use to learn about creating your own client application.

Entity manager tutorial: Creating an entity class

Create a local ObjectGrid with one entity by creating an Entity class, registering the entity type, and storing an entity instance into the cache.

Procedure

1. Create the Order object. To identify the object as an ObjectGrid entity, add the @Entity annotation. When you add this annotation, all serializable attributes in the object are automatically persisted in eXtreme Scale, unless you use annotations on the attributes to override the attributes. The **`orderNumber`** attribute is annotated with `@Id` to indicate that this attribute is the primary key. An example of an Order object follows:

   `Order.java`

   ```
   @Entity
   public class Order
   {
       @Id String orderNumber;
       Date date;
       String customerName;
       String itemName;
       int quantity;
       double price;
   }
   ```

2. Run the eXtreme Scale Hello World application to demonstrate the entity operations. The following example program can be issued in stand-alone mode to demonstrate the entity operations. Use this program in an Eclipse Java project that has the `objectgrid.jar` file added to the class path. An example of a simple Hello world application that uses eXtreme Scale follows:

```
Application.java

package emtutorial.basic.step1;

import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.em.EntityManager;

public class Application
{

    static public void main(String [] args)
        throws Exception
    {
        ObjectGrid og =
    ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
        og.registerEntities(new Class[] {Order.class});

        Session s = og.getSession();
        EntityManager em = s.getEntityManager();

        em.getTransaction().begin();

        Order o = new Order();
        o.customerName = "John Smith";
        o.date = new java.util.Date(System.currentTimeMillis());
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;

        em.persist(o);
        em.getTransaction().commit();

        em.getTransaction().begin();
        o = (Order)em.find(Order.class, "1");
        System.out.println("Found order for customer: " + o.customerName);
        em.getTransaction().commit();
    }
}
```

This example application performs the following operations:

a. Initializes a local eXtreme Scale with an automatically generated name.
b. Registers the entity classes with the application by using the registerEntities API, although using the registerEntities API is not always necessary.
c. Retrieves a Session and a reference to the entity manager for the Session.
d. Associates each eXtreme Scale Session with a single EntityManager and EntityTransaction. The EntityManager is now used.
e. The registerEntities method creates a BackingMap object that is called Order, and associates the metadata for the Order object with the BackingMap object. This metadata includes the key and non-key attributes, along with the attribute types and names.
f. A transaction starts and creates an Order instance. The transaction is populated with some values. The transaction is then persisted by using the EntityManager.persist method, which identifies the entity as waiting to be included in the associated map.
g. The transaction is then committed, and the entity is included in the ObjectMap instance.
h. Another transaction is made, and the Order object is retrieved by using the key 1. The type cast on the EntityManager.find method is necessary. The Java SE 5 capability is not used to ensure that the `objectgrid.jar` file works on a Java SE Version 5 and later Java virtual machine.

Entity manager tutorial: Forming entity relationships

Create a simple relationship between entities by creating two entity classes with a relationship, registering the entities with the ObjectGrid, and storing the entity instances into the cache.

Procedure

1. Create the customer entity, which is used to store customer details independently from the Order object. An example of the customer entity follows:

   Customer.java
   ```
   @Entity
   public class Customer
   {
       @Id String id;
       String firstName;
       String surname;
       String address;
       String phoneNumber;
   }
   ```

   This class includes information about the customer such as name, address, and phone number.
2. Create the Order object, which is similar to the Order object in the “Entity manager tutorial: Creating an entity class” on page 81 topic. An example of the order object follows:

   Order.java
   ```

   @Entity
   public class Order
   {
       @Id String orderNumber;
       Date date;
       @ManyToOne(cascade=CascadeType.PERSIST) Customer customer;
       String itemName;
       int quantity;
       double price;
   }
   ```

   In this example, a reference to a Customer object replaces the customerName attribute. The reference has an annotation that indicates a many-to-one relationship. A many-to-one relationship indicates that each order has one customer, but multiple orders might reference the same customer. The cascade annotation modifier indicates that if the entity manager persists the Order object, it must also persist the Customer object. If you choose to not set the cascade persist option, which is the default option, you must manually persist the Customer object with the Order object.
3. Using the entities, define the maps for the ObjectGrid instance. Each map is defined for a specific entity, and one entity is named Order and the other is named Customer. The following example application illustrates how to store and retrieve a customer order:

   Application.java
   ```

   public class Application
   {
       static public void main(String [] args)
           throws Exception
       {
           ObjectGrid og =
        ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
           og.registerEntities(new Class[] {Order.class});

           Session s = og.getSession();
           EntityManager em = s.getEntityManager();

           em.getTransaction().begin();

           Customer cust = new Customer();
           cust.address = "Main Street";
           cust.firstName = "John";
   ```

```
        cust.surname = "Smith";
        cust.id = "C001";
        cust.phoneNumber = "5555551212";

        Order o = new Order();
        o.customer = cust;
        o.date = new java.util.Date();
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;

        em.persist(o);
        em.getTransaction().commit();

        em.getTransaction().begin();
        o = (Order)em.find(Order.class, "1");
        System.out.println("Found order for customer: "
     + o.customer.firstName + " " + o.customer.surname);
        em.getTransaction().commit();
 // Close the session (optional in Version 7.1.1 and later) for improved performance
 s.close();
    }
}
```

This application is similar to the example application that is in the previous step. In the preceding example, only a single class Order is registered. WebSphere eXtreme Scale detects and automatically includes the reference to the Customer entity, and a Customer instance for `John Smith` is created and referenced from the new Order object. As a result, the new customer is automatically persisted, because the relationship between two orders includes the cascade modifier, which requires that each object be persisted. When the Order object is found, the entity manager automatically finds the associated Customer object and inserts a reference to the object.

Entity manager tutorial: Order Entity Schema

Create four entity classes by using both single and bidirectional relationships, ordered lists, and foreign key relationships. The EntityManager APIs are used to persist and find the entities. Building on the Order and Customer entities that are in the previous parts of the tutorial, this tutorial step adds two more entities: the Item and OrderLine entities.

About this task

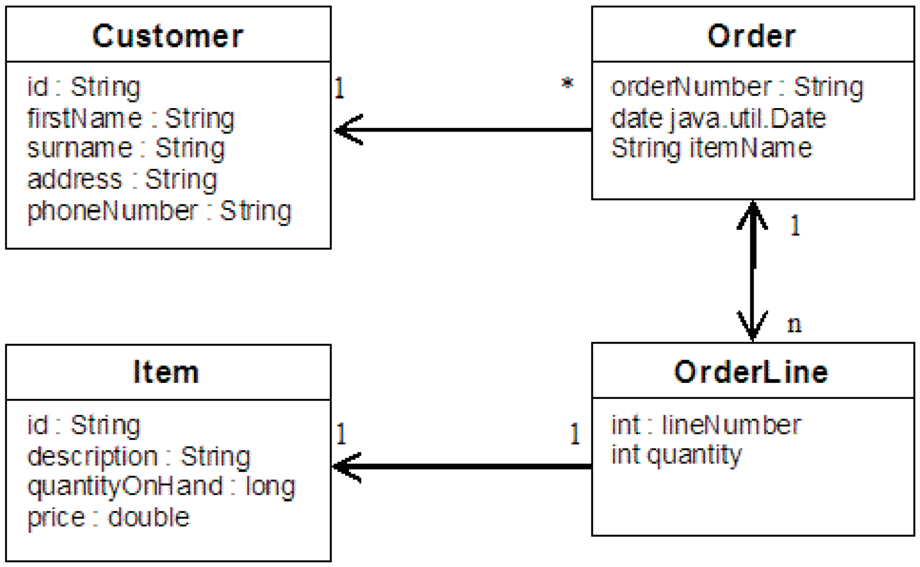


Figure 20. Order Entity Schema. An Order entity has a reference to one customer and zero or more OrderLines. Each OrderLine entity has a reference to a single item and includes the quantity ordered.

Procedure

1. Create the customer entity, which is similar to the previous examples.

   Customer.java
   ```
   @Entity
   public class Customer
   {
       @Id String id;
       String firstName;
       String surname;
       String address;
       String phoneNumber;
   }
   ```
2. Create the Item entity, which holds information about a product that is included in the store's inventory, such as the product description, quantity, and price.

   Item.java
   ```
   @Entity
   public class Item
   {
       @Id String id;
       String description;
       long quantityOnHand;
       double price;
   }
   ```
3. Create the OrderLine entity. Each Order has zero or more OrderLines, which identify the quantity of each item in the order. The key for the OrderLine is a compound key that consists of the Order that owns the OrderLine and an integer that assigns the order line a number. Add the cascade persist modifier to every relationship on your entities.

   OrderLine.java
   ```
   @Entity
   public class OrderLine
   {
   ```

```
    @Id @ManyToOne(cascade=CascadeType.PERSIST) Order order;
    @Id int lineNumber;
    @OneToOne(cascade=CascadeType.PERSIST) Item item;
    int quantity;
    double price;
}
```

4. Create the final Order Object, which has a reference to the Customer for the order and a collection of OrderLine objects.

   Order.java
   ```
   @Entity
   public class Order
   {
       @Id String orderNumber;
       java.util.Date date;
       @ManyToOne(cascade=CascadeType.PERSIST) Customer customer;
       @OneToMany(cascade=CascadeType.ALL, mappedBy="order")
      @OrderBy("lineNumber") List<OrderLine> lines;
   }
   ```

   The cascade ALL is used as the modifier for lines. This modifier signals the EntityManager to cascade both the PERSIST operation and the REMOVE operation. For example, if the Order entity is persisted or removed, then all OrderLine entities are also persisted or removed.

   If an OrderLine entity is removed from the lines list in the Order object, the reference is then broken. However, the OrderLine entity is not removed from the cache. You must use the EntityManager remove API to remove entities from the cache. The REMOVE operation is not used on the customer entity or the item entity from OrderLine. As a result, the customer entity remains even though the order or item is removed when the OrderLine is removed.

   The mappedBy modifier indicates an inverse relationship with the target entity. The modifier identifies which attribute in the target entity references the source entity, and the owning side of a one-to-one or many-to-many relationship. Typically, you can omit the modifier. However, an error is displayed to indicate that it must be specified if WebSphere eXtreme Scale cannot discover it automatically. An OrderLine entity that contains two of type Order attributes in a many-to-one relationship typically causes the error.

   The @OrderBy annotation specifies the order in which each OrderLine entity should be in the lines list. If the annotation is not specified, then the lines display in an arbitrary order. Although the lines are added to the Order entity by issuing ArrayList, which preserves the order, the EntityManager does not necessarily recognize the order. When you issue the find method to retrieve the Order object from the cache, the list object is not an ArrayList object.

5. Create the application. The following example illustrates the final Order object, which has a reference to the Customer for the order and a collection of OrderLine objects.
   a. Find the Items to order, which then become Managed entities.
   b. Create the OrderLine and attach it to each Item.
   c. Create the Order and associate it with each OrderLine and the customer.
   d. Persist the order, which automatically persists each OrderLine.
   e. Commit the transaction, which detaches each entity and synchronizes the state of the entities with the cache.
   f. Print the order information. The OrderLine entities are automatically sorted by the OrderLine ID.

   ```
   Application.java

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

```
{
    ...

    // Add some items to our inventory.
    em.getTransaction().begin();
    createItems(em);
    em.getTransaction().commit();

    // Create a new customer with the items in his cart.
    em.getTransaction().begin();
    Customer cust = createCustomer();
    em.persist(cust);

    // Create a new order and add an order line for each item.
    // Each line item is automatically persisted since the
// Cascade=ALL option is set.
    Order order = createOrderFromItems(em, cust, "ORDER_1",
 new String[]{"1", "2"}, new int[]{1,3});
    em.persist(order);
    em.getTransaction().commit();

    // Print the order summary
    em.getTransaction().begin();
    order = (Order)em.find(Order.class, "ORDER_1");
    System.out.println(printOrderSummary(order));
    em.getTransaction().commit();
}

public static Customer createCustomer() {
    Customer cust = new Customer();
    cust.address = "Main Street";
    cust.firstName = "John";
    cust.surname = "Smith";
    cust.id = "C001";
    cust.phoneNumber = "5555551212";
    return cust;
}

public static void createItems(EntityManager em) {
    Item item1 = new Item();
    item1.id = "1";
    item1.price = 9.99;
    item1.description = "Widget 1";
    item1.quantityOnHand = 4000;
    em.persist(item1);

    Item item2 = new Item();
    item2.id = "2";
    item2.price = 15.99;
    item2.description = "Widget 2";
    item2.quantityOnHand = 225;
    em.persist(item2);

}

public static Order createOrderFromItems(EntityManager em,
Customer cust, String orderId, String[] itemIds, int[] qty) {

    Item[] items = getItems(em, itemIds);

    Order order = new Order();
    order.customer = cust;
    order.date = new java.util.Date();
    order.orderNumber = orderId;
    order.lines = new ArrayList<OrderLine>(items.length);
for(int i=0;i<items.length;i++){
  OrderLine line = new OrderLine();
```

```
            line.lineNumber = i+1;
            line.item = items[i];
            line.price = line.item.price;
            line.quantity = qty[i];
            line.order = order;
            order.lines.add(line);
        }
        return order;
    }

    public static Item[] getItems(EntityManager em, String[] itemIds) {
        Item[] items = new Item[itemIds.length];
        for(int i=0;i<items.length;i++){
    items[i] = (Item) em.find(Item.class, itemIds[i]);
        }
        return items;
    }
```

The next step is to delete an entity. The EntityManager interface has a remove method that marks an object as deleted. The application should remove the entity from any relationship collections before calling the remove method. Edit the references and issue the remove method, or em.remove(object), as a final step.

Entity manager tutorial: Updating entries

If you want to change an entity, you can find the instance, update the instance and any referenced entities, and commit the transaction.

Procedure

Update entries. The following example demonstrates how to find the Order instance, change it and any referenced entities, and commit the transaction.

```
public static void updateCustomerOrder(EntityManager em) {
    em.getTransaction().begin();
    Order order = (Order) em.find(Order.class, "ORDER_1");
    processDiscount(order, 10);
    Customer cust = order.customer;
    cust.phoneNumber = "5075551234";
    em.getTransaction().commit();
}

public static void processDiscount(Order order, double discountPct) {
    for(OrderLine line : order.lines) {
        line.price = line.price * ((100-discountPct)/100);
    }
}
```

Flushing the transaction synchronizes all managed entities with the cache. When a transaction is committed, a flush automatically occurs. In this case, the Order becomes a managed entity. Any entities that are referenced from the Order, Customer, and OrderLine also become managed entities. When the transaction is flushed, each of the entities are checked to determine if they have been modified. Those that are modified are updated in the cache. After the transaction completes, by either being committed or rolled back, the entities become detached and any changes that are made in the entities are not reflected in the cache.

Entity manager tutorial: Updating and removing entries with an index

You can use an index to find, update, and remove entities.

Procedure

Update and remove entities by using an index. Use an index to find, update, and remove entities. In the following examples, the Order entity class is updated to use the @Index annotation. The @Index annotation signalsWebSphere eXtreme Scale to create a range index for an attribute. The name of the index is the same name as the name of the attribute and is always a MapRangeIndex index type.

Order.java
```
@Entity
public class Order
{
    @Id String orderNumber;
    @Index java.util.Date date;
    @OneToOne(cascade=CascadeType.PERSIST) Customer customer;
    @OneToMany(cascade=CascadeType.ALL, mappedBy="order")
    @OrderBy("lineNumber") List<OrderLine> lines;  }
```

The following example demonstrates how to cancel all orders that are submitted within the last minute. Find the order by using an index, add the items in the order back into the inventory, and remove the order and the associated line items from the system.

```
public static void cancelOrdersUsingIndex(Session s)
  throws ObjectGridException {
    // Cancel all orders that were submitted 1 minute ago
    java.util.Date cancelTime = new
   java.util.Date(System.currentTimeMillis() - 60000);
    EntityManager em = s.getEntityManager();
    em.getTransaction().begin();
    MapRangeIndex dateIndex = (MapRangeIndex)
   s.getMap("Order").getIndex("date");
    Iterator<Tuple> orderKeys = dateIndex.findGreaterEqual(cancelTime);
  while(orderKeys.hasNext()) {
   Tuple orderKey = orderKeys.next();
   // Find the Order so we can remove it.
   Order curOrder = (Order) em.find(Order.class, orderKey);
   // Verify that the order was not updated by someone else.
   if(curOrder != null && curOrder.date.getTime() >= cancelTime.getTime()) {
    for(OrderLine line : curOrder.lines) {
     // Add the item back to the inventory.
     line.item.quantityOnHand += line.quantity;
     line.quantity = 0;
    }
   em.remove(curOrder);
   }
  }
 em.getTransaction().commit();
}
```

Entity manager tutorial: Updating and removing entries by using a query

You can update and remove entities by using a query.

Procedure

Update and remove entities by using a query.

Order.java
```
@Entity
public class Order
{
    @Id String orderNumber;
    @Index java.util.Date date;
    @OneToOne(cascade=CascadeType.PERSIST) Customer customer;
    @OneToMany(cascade=CascadeType.ALL, mappedBy="order")
   @OrderBy("lineNumber") List<OrderLine> lines;
}
```

The order entity class is the same as it is in the previous example. The class still provides the @Index annotation, because the query string uses the date to find the entity. The query engine uses indices when they can be used.

```
public static void cancelOrdersUsingQuery(Session s) {
        // Cancel all orders that were submitted 1 minute ago
        java.util.Date cancelTime =
     new java.util.Date(System.currentTimeMillis() - 60000);
        EntityManager em = s.getEntityManager();
        em.getTransaction().begin();

        // Create a query that will find the order based on date.  Since
        // we have an index defined on the order date, the query
    // will automatically use it.
        Query query = em.createQuery("SELECT order FROM Order order
     WHERE order.date >= ?1");
        query.setParameter(1, cancelTime);
        Iterator<Order> orderIterator = query.getResultIterator();
    while(orderIterator.hasNext()) {
     Order order = orderIterator.next();
     // Verify that the order wasn't updated by someone else.
     // Since the query used an index, there was no lock on the row.
     if(order != null && order.date.getTime() >= cancelTime.getTime()) {
       for(OrderLine line : order.lines) {
        // Add the item back to the inventory.
        line.item.quantityOnHand += line.quantity;
        line.quantity = 0;
       }
       em.remove(order);
     }
   }
  em.getTransaction().commit();
}
```

Like the previous example, the cancelOrdersUsingQuery method intends to cancel all orders that were submitted in the past minute. To cancel the order, you find the order using a query, add the items in the order back into the inventory, and remove the order and associated line items from the system.

Tutorial: Configuring Java SE security

With the following tutorial, you can create a distributed eXtreme Scale environment in a Java Platform, Standard Edition environment.

Before you begin

Ensure that you are familiar with the basics of a distributed eXtreme Scale configuration.

About this task

Use this tutorial when you have installed eXtreme Scale in a stand-alone environment. Each step in the tutorial builds on the previous one. Follow each of the steps to secure a distributed eXtreme Scale and develop a simple Java SE application to access the secured eXtreme Scale.

Begin tutorial

Java SE security tutorial - Step 1

In order to work with the rest of the tutorial, you need to create and package a simple Java program and two XML files. These set of files defines a simple ObjectGrid configuration with one ObjectGrid instance named `accounting` and a `customer` map. The `SimpleDP.xml` file features a deployment policy of one map set configured with one partition and zero minimum required replicas.

Procedure

1. In a command line window, go to the *wxs_home* directory.
2. Create a directory called `applib`.
3. Ensure your development environment contains the `ogclient.jar` file in the classpath. For more information, see the *Programming Guide*.
4. Create and compile the following `SimpleApp.java` class:

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

   import com.ibm.websphere.objectgrid.ClientClusterContext;
   import com.ibm.websphere.objectgrid.ObjectGrid;
   import com.ibm.websphere.objectgrid.ObjectGridManager;
   import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
   import com.ibm.websphere.objectgrid.ObjectMap;
   import com.ibm.websphere.objectgrid.Session;

   public class SimpleApp {

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

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

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

           Session session = og.getSession();

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

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

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

           System.out.println("The customer name for ID 0001 is " + customer);
       }

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

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

           return og;

       }

   }
   ```
5. Compile the package with this file and name the JAR `sec_sample.jar`.
6. Go to the *wxs_home* directory, and create a directory called `xml`

7. In the*wxs_home*/xml directory, create the following configuration files:

   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>
```

   The following XML file configures the deployment environment.

   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>
```

Results

These files create a simple ObjectGrid configuration with one ObjectGrid an `accounting` instance and a `customer` map.

Java SE security tutorial - Step 2

Before you can verify that the `SimpleApp.java` sample runs, you need to start a catalog server and a container server. After starting these services successfully, you can then launch the client and run the sample. Additional security features are added incrementally in the steps of the tutorial to increase the amount of integrated security that is available.

Before you begin

To successfully complete this step of the tutorial, you should have access to the following files:

- Have access to the compiled `sec_sample.jar` package. This package contains the `SimpleApp.java` program.
- Have access to the necessary configuration files `SimpleApp.xml` and `SimpleDP.xml`.

You should have created these files in "Java SE security tutorial - Step 1" on page 90 of this tutorial.

You should also know how to:

- Start and stop a catalog servers and container servers. For more information, see "Starting and stopping stand-alone servers" on page 459.
- Run the **xscmd** utility in order verify the map size inserted into the data grid.

Procedure

1. In a command line window, go to the *wxs_home*/bin directory and start the catalog service.
   - UNIX Linux `./startOgServer.sh catalogServer`

- Windows `startOgServer.bat catalogServer`

2. Start a container service named `c0`:
   - UNIX Linux `./startOgServer.sh c0 -objectGridFile ../xml/SimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndPoints localhost:2809`
   - Windows `startOgServer.bat c0 -objectGridFile ..\xml\SimpleApp.xml -deploymentPolicyFile ..\xml\SimpleDP.xml  -catalogServiceEndPoints localhost:2809`
3. After the catalog server and container server have been started, run the `sec_sample.jar` sample as follows: UNIX java -classpath ../lib/objectgrid.jar:../applib/sec_sample.jar com.ibm.websphere.objectgrid.security.sample.guide.SimpleApp

   Windows java -classpath ..\lib\objectgrid.jar;..\applib\sec_sample.jar com.ibm.websphere.objectgrid.security.sample.guide.SimpleApp The output of the sample is: `The customer name for ID 0001 is fName lName` The getObjectGrid method in this class obtains an ObjectGrid, and the run method reads a record from the customer map and updates the value in the accounting grid.
4. Verify the size of the "customer" map inserted into the "accounting" grid, by issuing the **`xscmd`** command utility as follows:
   - UNIX Linux `./xscmd.sh -c showMapSizes -g accounting -ms mapSet1`
   - Windows `xscmd.bat -c showMapSizes -g accounting -ms mapSet1`
5. Stop a container server named `c0` with one of the following scripts:
   - UNIX Linux `./stopOgServer.sh c0 -catalogServiceEndPoints localhost:2809`
   - Windows `stopOgServer.bat c0 -catalogServiceEndPoints localhost:2809`

   If the server stopped successfully, then you will see the following message:

   ```
   CWOBJ2512I: ObjectGrid server c0 stopped.
   ```
6. Stop the catalog server with one of the following scripts:
   - UNIX Linux `./stopOgServer.sh catalogServer -catalogServiceEndPoints localhost:2809`
   - Windows `stopOgServer.bat catalogServer -catalogServiceEndPoints localhost:2809`

   If the server stopped successfully, then you will see the following message:

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

Java SE security tutorial - Step 3

The rest of the tutorial demonstrates how to enable client authentication before connecting to an eXtreme Scale server. To prepare for the next step of this tutorial, you need to package the `SecureSimpleApp.java` program into a JAR and create a set of configuration files, which include a `security.xml` file, and two JAAS configuration files. The `security.xml` file lets you write authentication into the environment, and the JAAS configuration files provide the authentication mechanism when connecting to the server.

About this task

Procedure

1. In a command line window, go to the *wxs_home*/`applib` directory you created in "Java SE security tutorial - Step 1" on page 90.
2. Create and compile the following `SecureSimpleApp.java` class:

   Attention: In the following example, some lines of code are continued on the next line for publication purposes.

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

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

   public class SecureSimpleApp extends SimpleApp {

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

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

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

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

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

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

           return og;

       }

   }
   ```
3. Ensure your development environment contains the `ogclient.jar` file in the classpath. For more information, see the *Programming Guide*.
4. Compile the package with these files and name the JAR `sec_sample.jar`.
5. Change to the *wxs_home* directory.
6. Create a directory called `security`.
7. Create a configuration file called `security.xml`. Server security properties are specified in this file. These properties are common for both catalog servers and container servers.

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

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

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

```
            </authenticator>
        </security>

    </securityConfig>
```

Java SE security tutorial - Step 4

Building on the previous step, the following topic shows how to implement client authentication in a distributed eXtreme Scale environment.

Before you begin

Be sure that you have completed "Java SE security tutorial - Step 3" on page 93. You need to have created and complied the `SecureSimpleApp.java` sample into a `sec_sample.jar` file, and created a configuration file called `security.xml`.

About this task

With client authentication enabled, a client is authenticated before connecting to the eXtreme Scale server. This section demonstrates how client authentication can be done in an eXtreme Scale server environment, using the sample `SecureSimpleApp.java`.

Client credential

The `SecureSimpleApp.java` sample uses the following two plug-in implementations to obtain client credentials:

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

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

For more information about these plug-ins, see Client authentication programming.

Server authenticator

The example uses an eXtreme Scale built-in implementation: `KeyStoreLoginAuthenticator`, which is for testing and sample purposes (a keystore is a simple user registry and should not be used for production). For more information, see the topic on authenticator plug-in under Client authentication programming.

Procedure

1. In a command line window, go to the *wxs_home* directory.
2. Change to the *wxs_home*/`security` directory you had created in "Java SE security tutorial - Step 3" on page 93.
3. Create a JAAS configuration file that enforces a method of authentication to the server, `og_jaas.config`. The `KeyStoreLoginAuthenticator` referenced in the `security.xml` file uses a keystore by using the JAAS login module "KeyStoreLogin". The keystore can be configured as an option to the KeyStoreLoginModule class.

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

   Important: Windows If you are using Windows, the directory path does not support backslashes. If you have used backslashes, you must escape any backslash (\ ) characters in the path. For example, if you want to use the path C:\opt\ibm, enter C:\\opt\\ibm in the properties file. Windows directories with spaces are not supported.

4. Change to the *java_home*/`bin` directory and run the keytool.
5. Change to the *wxs_home* /`security` directory, and create two users, "manager" and "cashier" with their own passwords.
   a. Use the keytool to create a user "manager" with password "manager1" in the keystore sampleKS.jks.
      - UNIX Linux
        ```
        keytool -genkey -v -keystore sampleKS.jks -storepass sampleKS1 \
        -alias manager -keypass manager1 \
        -dname CN=manager,O=acme,OU=OGSample -validity 10000
        ```
      - Windows
        ```
        keytool -genkey -v -keystore sampleKS.jks -storepass sampleKS1 ^
        -alias manager -keypass manager1 ^
        -dname CN=manager,O=acme,OU=OGSample -validity 10000
        ```
   b. Use the keytool to create a user "cashier" with password "cashier1" in the keystore sampleKS.jks.
      - UNIX Linux
        ```
        keytool -genkey -v -keystore sampleKS.jks -storepass sampleKS1 \
        -alias cashier -keypass cashier1 \
        -dname CN=cashier,O=acme,OU=OGSample -validity 10000
        ```
      - Windows
        ```
        keytool -genkey -v -keystore sampleKS.jks -storepass sampleKS1 ^
        -alias cashier -keypass cashier1 ^
        -dname CN=cashier,O=acme,OU=OGSample -validity 10000
        ```
6. Make a copy of the `sampleClient.properties` file located in `wxs_home/properties` directory to `wxs_home/security/client.properties`
   - UNIX Linux
     ```
     cp ../properties/sampleClient.properties client.properties
     ```
   - Windows
     ```
     copy ..\properties\sampleClient.properties client.properties
     ```
7. In the *wxs_home*/`security` directory, save it as `client.properties`

   Make the following changes to the `client.properties` file:
   a. **securityEnabled:** Set **`securityEnabled`** to true (default value) enables the client security, which includes authentication.
   b. **credentialAuthentication:** Set **`credentialAuthentication`** to Supported (default value), which means the client supports credential authentication.
   c. **transportType:** Set **`transportType`** to TCP/IP, which means no SSL will be used.
8. Copy the `sampleServer.properties` file into the *wxs_home*/`security` directory and save it as `server.properties`.
   - UNIX Linux
     ```
     cp ../properties/sampleServer.properties server.properties
     ```
   - Windows
     ```
     copy ..\properties\sampleServer.properties server.properties
     ```

   Make the following changes in the `server.properties` file:
   a. **securityEnabled:** Set the **`securityEnabled`** attribute to true.

b. **transportType:** Set **transportType** attribute to TCP/IP, which means no SSL is used.

c. **secureTokenManagerType:** Set **secureTokenManagerType** attribute to none to not configure the secure token manager.

9. Go to the *wxs_home*/bin directory and depending on your platform, issue one of the following commands to start a catalog server. You need to issue the**-clusterSecurityFile** and **-serverProps** command line options to pass in security properties:

   - 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"
     ```

10. Start a container server named c0 with one of the following scripts. The server property file is passed by issuing **-serverProps**.

    a.
    - UNIX Linux
      ```
      ./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"
      ```

11. After the catalog server and container server have been started, run the sec_sample.jar sample as follows:

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

    Linux Use a colon (:) for the classpath separator instead of a semicolon (;) as in the previous example.

    After you issue the class, the following output results:

    The customer name for ID 0001 is fName lName.

12. Verify the size of the "customer" map inserted into the "accounting" grid, by issuing the **xscmd** command utility as follows:

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

13. Optional: To stop the container or catalog servers, you can use the **stopOgServer** or **stopXsServer** command. However you need to provide a

security configuration file. The sample client property file defines the following two properties to generate a userID/password credential (manager/manager1).

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

Stop the container c0 with the following command.

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

If you do not provide the **`-clientSecurityFile`** option, you will see an exception with the following message.

```
>> 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
```

You can also shut down the catalog server using the following command. However, if you want to continue trying the next step tutorial, you can let the catalog server stay running.

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

If you do shutdown the catalog server, you will see the following output.

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

Now, you have successfully made your system partially secure by enabling authentication. You configured the server to plug in the user registry, configured the client to provide client credentials, and changed the client property file and cluster XML file to enable authentication.

If you provide an invalidate password, you see an exception stating that the user name or password is not correct.

For more details about client authentication, see “Authenticating application clients” on page 586.

Next step of tutorial

Java SE security tutorial - Step 5

After authenticating a client, as in the previous step, you can give security privileges through eXtreme Scale authorization mechanisms.

Before you begin

Be sure to have completed “Java SE security tutorial - Step 4” on page 95 prior to proceeding with this task.

About this task

The previous step of this tutorial demonstrated how to enable authentication in an eXtreme Scale grid. As a result, no unauthenticated client can connect to your server and submit requests to your system. However, every authenticated client has the same permission or privileges to the server, such as reading, writing, or deleting data that is stored in the ObjectGrid maps. Clients can also issue any type of query. This section demonstrates how to use eXtreme Scale authorization to give various authenticated users varying privileges.

Similar to many other systems, eXtreme Scale adopts a permission-based authorization mechanism. WebSphere eXtreme Scale has different permission categories that are represented by different permission classes. This topic features MapPermission. For complete category of permissions, see Client authorization programming.

In WebSphere eXtreme Scale, the com.ibm.websphere.objectgrid.security.MapPermission class represents permissions to the eXtreme Scale resources, specifically the methods of ObjectMap or JavaMap interfaces. WebSphere eXtreme Scale defines the following permission strings to access the methods of ObjectMap and JavaMap:

- read: Grants permission to read the data from the map.
- write: Grants permission to update the data in the map.
- insert: Grants permission to insert the data into the map.
- remove: Grants permission to remove the data from the map.
- invalidate: Grants permission to invalidate the data from the map.
- all: Grants all permissions to read, write, insert, remote, and invalidate.

The authorization occurs when a client calls a method of ObjectMap or JavaMap. The eXtreme Scale runtime environment checks different map permissions for different methods. If the required permissions are not granted to the client, an AccessControlException results.

This tutorial demonstrates how to use Java Authentication and Authorization Service (JAAS) authorization to grant authorization map accesses for different users.

Procedure

1. **Enable eXtreme Scale authorization**. To enable authorization on the ObjectGrid, you need to set the securityEnabled attribute to `true` for that particular ObjectGrid in the XML file. Enabling security on the ObjectGrid means that you are enabling authorization. Use the following commands to create a new ObjectGrid XML file with security enabled.
   a. Navigate to the `xml` directory.
      ```
      cd objectgridRoot/xml
      ```
   b. Copy the `SimpleApp.xml` file to the `SecureSimpleApp.xml` file.
      - UNIX Linux
        ```
        cp SimpleApp.xml SecureSimpleApp.xml
        ```
      - Windows
        ```
        copy SimpleApp.xml SecureSimpleApp.xml
        ```
   c. Open the `SecureSimpleApp.xml` file and add `securityEnabled="true"` on the ObjectGrid level as the following XML shows:

```
<?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. **Define the authorization policy.** In the previous client authentication topic, you created the users, cashier and manager, in the keystore. In this example, the user "cashier" only has read permissions to all the maps, and the user "manager" has all permissions. JAAS authorization is used in this example. You must create a JAAS authorization policy file to grant permissions to principals. Create the following `og_auth.policy` file in the `objectgridRoot/security` directory:

   og_auth.policy
   ```
   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";
   };
   ```

   Note:
   - The `codebase "http://www.ibm.com/com/ibm/ws/objectgridRoot/security/PrivilegedAction"` is a specially-reserved URL for ObjectGrid. All ObjectGrid permissions granted to principals should use this special code base.
   - The first grant statement grants "read" map permission to principal `"CN=cashier,O=acme,OU=OGSample"`, so the cashier has only map read permission to all the maps in the ObjectGrid accounting.
   - The second grant statement grants "all" map permission to principal `"CN=manager,O=acme,OU=OGSample"`, so the manager has all permissions to maps in the ObjectGrid accounting.

   Now you can launch a server with an authorization policy. The JAAS authorization policy file can be set using the standard -D property: `-Djava.security.policy=../security/og_auth.policy`

3. **Run the application.**

   After you create the above files, you can run the application.

   Use the following commands to start the catalog server. For more information about starting the catalog service, see "Starting a stand-alone catalog service" on page 461.

   a. Navigate to the bin directory: `cd objectgridRoot/bin`

   b. Start the catalog server.

      - 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"
        ```

The security.xml and server.properties files were created in the previous step of this tutorial.

c. You can then start a secure container server using the following script. Run the following script from the bin directory:

- 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.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.policy="..\security\og_auth.policy"
  ```

Notice the following differences from the previous container server start command:

- The SecureSimpleApp.xml file is used instead of SimpleApp.xml file, which is the result of your running the sample sec_sample.jar file to set client authentication.
- Another -Djava.security.policy argument was added to set the JAAS authorization policy file to the container server process.

Use the same command as in the previous step of the tutorial:

a. Navigate to the bin directory.

- UNIX Linux

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

- Windows

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

b. Because user "manager" has all permissions to maps in the accounting ObjectGrid, the application runs properly.

   Now, instead of using user "manager", use user "cashier" to launch the client application.

c. Navigate to the bin directory.

- UNIX Linux

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

- Windows

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

The following exception results:

Attention: In the following example, some lines of code are continued on the next line for publication purposes.

```
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)
        at java.security.AccessController.doPrivileged(AccessController.java:242)
        at com.ibm.ws.objectgrid.security.MapAuthorizer.check(MapAuthorizer.java:66)
        at com.ibm.ws.objectgrid.security.SecuredObjectMapImpl.checkMapAuthorization(SecuredObjectMapImpl.java:429)
        at com.ibm.ws.objectgrid.security.SecuredObjectMapImpl.update(SecuredObjectMapImpl.java:490)
        at com.ibm.ws.objectgrid.SessionImpl.processLogSequence(SessionImpl.java:1913)
        at com.ibm.ws.objectgrid.SessionImpl.processLogSequence(SessionImpl.java:1805)
        at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processLogSequence(ServerCoreEventProcessor.java:1011)
        ... 14 more
```

This exception occurs because the user "cashier" does not have write permission, so it cannot update the map customer.

Now your system supports authorization. You can define authorization policies to grant different permissions to different users. For more information about authorization, see "Authorizing application clients" on page 589.

What to do next

Complete the next step of the tutorial. See "Java SE security tutorial - Step 6."

Java SE security tutorial - Step 6

The following step explains how you can enable a security layer for communication between your environment's endpoints.

Before you begin

Be sure you have completed "Java SE security tutorial - Step 5" on page 98 prior to proceeding with this task.

About this task

The eXtreme Scale topology supports both Transport Layer Security/Secure Sockets Layer (TLS/SSL) for secure communication between ObjectGrid endpoints (client, container servers, and catalog servers). This step of the tutorial builds upon the previous steps to enable transport security.

Procedure

1. **Create TLS/SSL keys and keystores**.

   In order to enable transport security, you must create a keystore and trust store. This exercise only creates one key and trust-store pair. These stores are used for ObjectGrid clients, container servers, and catalog servers, and are created with the JDK keytool.

   - *Create a private key in the keystore*

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

     Using this command, a keystore key.jks is created with a key "ogsample" stored in it. This keystore key.jks will be used as the SSL keystore.

   - *Export the public certificate*

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

     Using this command, the public certificate of key "ogsample" is extracted and stored in the file temp.key.

   - *Import the client's public certificate to the trust store*

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

     Using this command, the public certificate was added to keystore trust.jks. This trust.jks is used as the SSL trust store.

2. **Configure ObjectGrid property files**.

   In this step, you must configure the ObjectGrid property files to enable transport security.

   First, copy the key.jks and trust.jks files into the objectgridRoot/security directory.

   Set the following properties in the `client.properties` and `server.properties` file.

   ```
   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: The value of transportType is set to "SSL-Required", which means the transport requires SSL. So all the ObjectGrid endpoints (clients, catalog servers, and container servers) should have SSL configuration set and all transport communication will be encrypted.

   The other properties are used to set the SSL configurations. See "Transport layer security and secure sockets layer" on page 593 for a detailed explanation. Make sure you follow the instructions in this topic to update your `orb.properties` file.

   Make sure you follow this page to update your `orb.properties` file.

In the `server.properties` file, you must add an additional property clientAuthentication and set it to false. On the server side, you do not need to trust the client.

```
clientAuthentication=false
```

3. **Run the application**.

   The commands that you use in this step are the same as the commands in the "Java SE security tutorial - Step 3" on page 93 topic.

   a. Navigate to the `cd objectgridRoot/bin` directory, and use the following commands to start a catalog server:

      - 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"
        ```

      The `security.xml` and `server.properties` files were created in the "Java SE security tutorial - Step 2" on page 92 page.

      Use the **`-JMXServicePort`** option to explicitly specify the JMX port for the server. This option is required to use the **`xscmd`** utility.

   b. From the `objectgridRoot/bin` directory, start a secure ObjectGrid container server:

      - 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.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.policy="..\security\og_auth.policy"
        ```

   c. From the `objectgridRoot/bin` directory, run the following command to complete client authentication:

      - UNIX Linux

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

      - Windows

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

      Because user "manager" has permission to all the maps in the accounting ObjectGrid, the application runs successfully.

4. **Use the `xscmd` utility to show the map sizes of the "accounting" data grid**.

   a. From the `objectgridRoot/bin` directory, use the **`xscmd`** command to show the map sizes:

      - UNIX Linux

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

- Windows

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

  You see the following output.

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

5. **Troubleshoot running the application with an incorrect keystore**.

   If your truststore does not contain the public certificate of the private key in the keystore, an exception that the key cannot be trusted occurs.

   To show this exception, create another keystore `key2.jks`.

   ```
   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
   ```

   Then modify the `server.properties` file to make the keystore point to this new keystore `key2.jks`:

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

   a. From the `cd objectgridRoot/bin` directory, assume that you run the following commands, which use an incorrect keystore, to start the catalog server:

      - 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.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.policy="..\security\og_auth.policy"
        ```

      You receive the following exception:

```
CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN "CN=ogsample,
OU=Your Organizational Unit, O=Your Organization, L=Your City, ST=Your State,
C=Your Country" was sent from target host:port "9.23.39.177:36407". The signer may
need to be added to local trust store
"/opt/IBM/WebSphere/eXtremeScale/ObjectGrid/security/trust.jks"
located in SSL configuration alias "DefaultSystemProperties" loaded from SSL
configuration file "System Properties". The extended error message from the SSL
handshake exception is: "PKIX path building failed: java.security.cert.CertPathBuilderException:
unable to find valid certification path to requested target".

CWPKI0040I: An SSL handshake failure occurred from a secure client. The server's SSL signer
has to be added to the client's trust store. A retrieveSigners utility is provided to download
signers from the server but requires administrative permission. Check with your administrator
to have this utility run to setup the secure environment before running the client. Alternatively,
the com.ibm.ssl.enableSignerExchangePrompt can be enabled in ssl.client.props for "DefaultSSLSettings"
in order to allow acceptance of the signer during the connection attempt.
```

      To correct the exception, change the `server.properties` file back to use the `key.jks` file.

Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server

This tutorial demonstrates how to secure a WebSphere eXtreme Scale server deployment in a WebSphere Application Server environment.

Learning objectives

The learning objectives for this tutorial follow:

- Configure WebSphere eXtreme Scale to use WebSphere Application Server authentication plug-ins
- Configure WebSphere eXtreme Scale transport security to use WebSphere Application Server CSIv2 configuration
- Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server
- Use a custom login module for group-based JAAS authorization
- Use WebSphere eXtreme Scale **xscmd** utility in WebSphere Application Server environment

Time required

This tutorial takes approximately 4 hours from start to finish.

Introduction: Integrate WebSphere eXtreme Scale security with WebSphere Application Server using the WebSphere Application Server Authentication plug-ins

In this tutorial, you integrate WebSphere eXtreme Scale security with WebSphere Application Server. First, you configure authentication with a simple web application that uses authenticated user credentials from the current thread to connect to the ObjectGrid. Then, you investigate the encryption of data that is transferred between the client and server with transport layer security. To give users varying levels of permissions, you can configure Java Authentication and Authorization Service (JAAS). After completing the configuration, you can use the **xscmd** utility to monitor your data grids and maps.

This tutorial assumes that all of your WebSphere eXtreme Scale clients, container servers, and catalog servers are deployed in the WebSphere Application Server environment.

Learning objectives

The learning objectives for this tutorial follow:

- Configure WebSphere eXtreme Scale to use WebSphere Application Server authentication plug-ins
- Configure WebSphere eXtreme Scale transport security to use WebSphere Application Server CSIv2 configuration
- Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server
- Use a custom login module for group-based JAAS authorization
- Use WebSphere eXtreme Scale **xscmd** utility in WebSphere Application Server environment

Time required

This tutorial takes approximately 4 hours from start to finish.

Skill level

Intermediate.

Audience

Developers and administrators that are interested in the security integration between WebSphere eXtreme Scale and WebSphere Application Server.

System requirements and topology

- WebSphere Application Server Version 6.1 or Version 7.0.0.11 or later
- Update the Java runtime to apply the following fix: IZ79819: IBMJDK FAILS TO READ PRINCIPAL STATEMENT WITH WHITESPACE FROM SECURITY FILE

This tutorial uses four WebSphere Application Server application servers and one deployment manager to demonstrate the sample.

Prerequisites

A basic understanding of the following items is helpful before you start this tutorial:

- WebSphere eXtreme Scale programming model
- Basic WebSphere eXtreme Scale security concepts
- Basic WebSphere Application Server security concepts

For a background information about WebSphere eXtreme Scale and WebSphere Application Server security integration, see “Security integration with WebSphere Application Server” on page 601.

Related concepts:

“Security overview” on page 56
WebSphere eXtreme Scale can secure data access, including allowing for integration with external security providers.

Related tasks:

7.1.1+ “Installing WebSphere eXtreme Scale with the installation wizard” on page 183
You can use the installation wizard to install WebSphere eXtreme Scale for stand-alone or WebSphere Application Server configurations.

Related information:

WebSphere Application Server: Securing applications and their environment

Module 1: Prepare WebSphere Application Server

Before you start the tutorial to integrate with WebSphere eXtreme Scale, you must create a basic security configuration in WebSphere Application Server.

Learning objectives

With the lessons in this module, you learn how to:

- Configure WebSphere Application Server security to use an internal file-based federated repository as a user account registry.
- Create user groups and users.
- Create clusters for the application and WebSphere eXtreme Scale servers.

Time required

This module takes approximately 60 minutes.

Lesson 1.1: Understand the topology and get the tutorial files

To prepare your environment for the tutorial, you must configure WebSphere Application Server security. You configure administration and application security using internal file-based federated repositories as a user account registry.

This lesson guides you through the sample topology and applications that are used to in the tutorial. To begin running the tutorial, you must download the applications and place the configuration files in the correct locations for your environment. You can download the sample application from the WebSphere eXtreme Scale wiki.

WebSphere Application Server sample topology: This tutorial guides you through creating four WebSphere Application Server application servers to demonstrate using the sample applications with security enabled. These application servers are grouped into two clusters, each with two servers:

- **appCluster cluster**: Hosts the EmployeeManagement sample enterprise application. This cluster has two application servers: s1 and s2.
- **xsCluster cluster**: Hosts the eXtreme Scale container servers. This cluster has two application servers: xs1 and xs2.

In this deployment topology, the s1 and s2 application servers are the client servers that access data that is being stored in the data grid. The xs1 and xs2 servers are the container servers that host the data grid.

The catalog server is deployed in the deployment manager process by default. This tutorial uses the default behavior. Hosting the catalog server in the deployment manager is not a recommended practice in a production environment. In a production environment, you should create a catalog service domain to define where catalog servers start. See “Creating catalog service domains in WebSphere Application Server” on page 299 for more information.

Alternative configuration: You can host all of the application servers in a single cluster, such as in the appCluster cluster. With this configuration, all of the servers in the cluster are both clients and container servers. This tutorial uses two clusters to distinguish between the application servers that are hosting the clients and container servers.

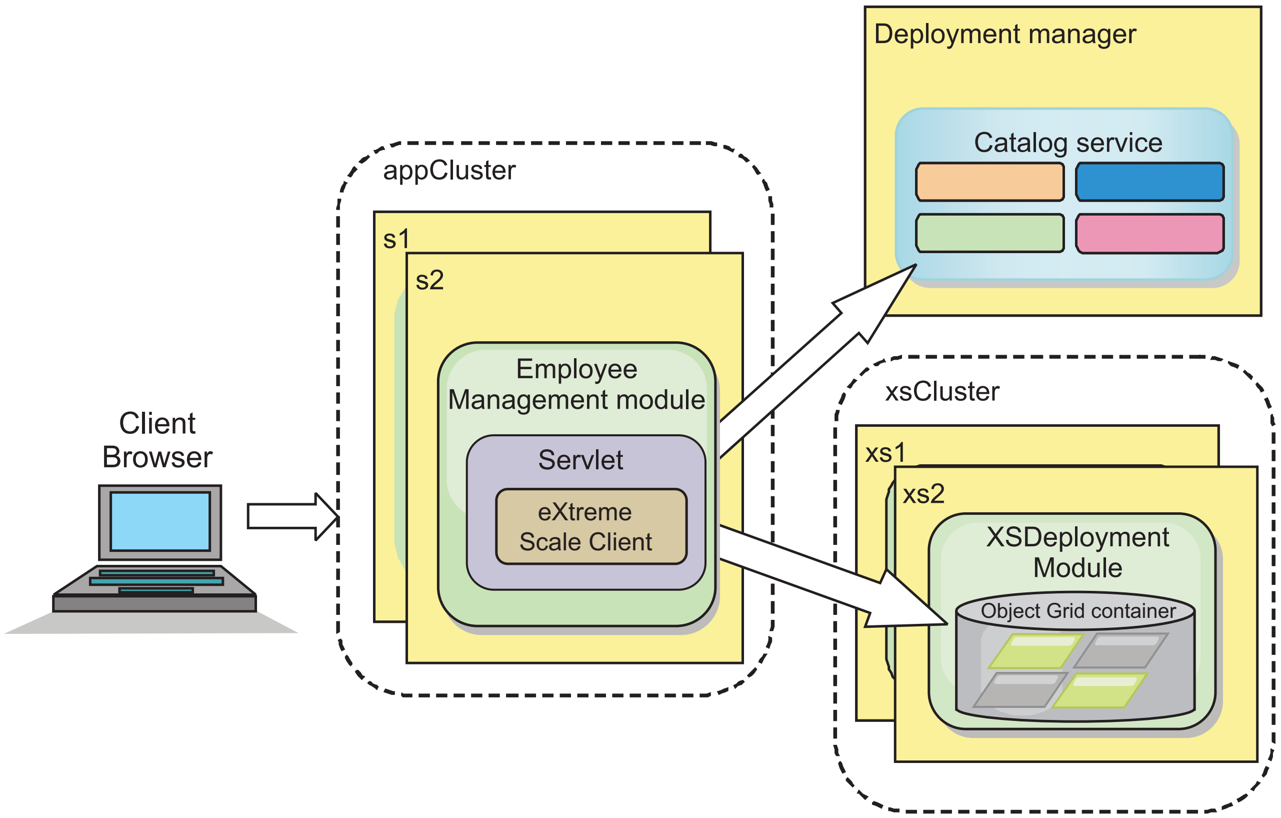


Figure 21. Tutorial topology

Applications: In this tutorial, you are using two applications and one shared library file:

- **`EmployeeManagement.ear`**: The `EmployeeManagement.ear` application is a simplified Java 2 Platform, Enterprise Edition (J2EE) enterprise application. It contains a web module to manage the employee profiles. The web module contains the `management.jsp` file to display, insert, update, and delete employee profiles that are stored in the container servers.
- **`XSDeployment.ear`**: This application contains an enterprise application module with no application artifacts. The cache objects are packaged in the `EmployeeData.jar` file. The `EmployeeData.jar` file is deployed as a shared library for the `XSDeployment.ear` file, so that the `XSDeployment.ear` file can access the classes. The purpose of this application is to package the eXtreme Scale configuration files. When this enterprise application is started, the eXtreme Scale configuration files are automatically detected by the eXtreme Scale run time, so the container servers are created. These configuration files include the `objectGrid.xml` and `objectGridDeployment.xml` files.
- **`EmployeeData.jar`**: This jar file contains one class: the com.ibm.websphere.sample.xs.data.EmployeeData class. This class represents employee data that is stored in the grid. This Java archive (JAR) file is deployed with the `EmployeeManagement.ear` and `XSDeployment.ear` files as a shared library.

Get the tutorial files:

1. Download the `WASSecurity.zip` and `security.zip` files. You can download the sample application from the WebSphere eXtreme Scale wiki.
2. Extract the `WASSecurity.zip` file to a directory for viewing the binary and source artifacts, for example the `/wxs_samples/` directory. This directory is

referred to as *samples_home* for the remainder of the tutorial. For a description of the contents of the WASSecurity.zip file and how to load the source into your Eclipse workspace, see the README.txt file in the package.

3. Extract the security.zip file to the *samples_home* directory. The security.zip file contains the following security configuration files that are used in this tutorial:
   - catServer2.props
   - server2.props
   - client2.props
   - securityWAS2.xml
   - xsAuth2.props

About the configuration files:

The objectGrid.xml and objectGridDeployment.xml files create the data grids and maps that store the application data.

These configuration files must be named objectGrid.xml and objectGridDeployment.xml. When the application server starts, eXtreme Scale detects these files in the META-INF directory of the EJB and web modules. If these files are found, it assumed that the Java virtual machine (JVM) acts as a container server for the defined data grids in the configuration files.

objectGrid.xml file

The objectGrid.xml file defined one ObjectGrid named Grid. The Grid data grid has one map, the Map1 map, that stores the employee profile for the application.

```
<?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 file

The objectGridDeployment.xml file specifies how to deploy the Grid data grid. When the grid is deployed, it has five partitions and one synchronous replica.

```
<?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>
```

Lesson checkpoint:

In this lesson, you learned about the topology for the tutorial and added the configuration files and sample applications to your environment.

If you want to learn more about automatically starting container servers, see "Configuring WebSphere Application Server applications to automatically start container servers" on page 327.

Lesson 1.2: Configure the WebSphere Application Server environment

To prepare your environment for the tutorial, you must configure WebSphere Application Server security. Enable administration and application security using internal file-based federated repositories as a user account registry. Then, you can create server clusters to host the client application and container servers.

The following steps were written using WebSphere Application Server Version 7.0. However, you can also apply the concepts apply to earlier versions of WebSphere Application Server.

Configure WebSphere Application Server security:

1. Configure WebSphere Application Server security.
   a. In the WebSphere Application Server administrative console, click **Security** > **Global Security**.
   b. Select **Federated repositories** as the **Available realm definition**. Click **Set as current**.
   c. Click **Configure..** to go to the Federated repositories panel.
   d. Enter the **Primary administrative user name**, for example, `admin`. Click **Apply.**
   e. When prompted, enter the administrative user password and click **OK**. Save your changes.
   f. On the **Global Security** page, verify that **Federated repositories** setting is set to the current user account registry.
   g. Select the following items: **Enable administrative security**, **Enable application security**, and **Use Java 2 security to restrict application access to local resources**. Click **Apply** and save your changes.
   h. Restart the deployment manager and any running application servers.

   The WebSphere Application Server administrative security is enabled using the internal file-based federated repositories as the user account registry.
2. Create two user groups: adminGroup and operatorGroup.
   a. Click **Users and groups** > **Manage groups** > **Create...**
   b. Type `adminGroup` as the group name. Enter `Administration group` as the description. Click **Create**.
   c. Click **Create alike**. Type `operatorGroup` as the group name. Enter `Operator group` as the description. Click **Create**.
   d. Click **Close**.
3. Create users `admin1` and `operator1`.
   a. Click **Users and groups** > **Manage users** > **Create...**
   b. Create a user called `admin1` with the first name `Joe` and last name `Doe` with the password `admin1`. Click **Create**.

c. Create a second user. Click **Create alike** to create a a user called `operator1` with the first name `Jane` and last name `Doe` with the password `operator1`. Click **Create**. Click **Close**.

4. Add users to the user groups. Add the `admin1` user to the `adminGroup` and the `operator1` user to the `operatorGroup`.
   a. Click **Users and groups** > **Manage users**.
   b. Search for users to add to groups. Click **Search..** and set the search for value to an asterisk (*) to display all the users.
   c. From the search result, click the `admin1` user, and click the **Groups** tab. Click **Add** to add the group.
   d. Search the groups to find the available groups. Click the `adminGroup` and click **Add**.
   e. Repeat these steps to add the `operator1` user to the `operatorGroup` user group.
5. Save your changes, log out of the administrative console, and restart the deployment manager and node agent to enable the security settings.

You enabled security and created users and user groups have administrative and operator access to your WebSphere Application Server configuration.

Create server clusters:

Create two server clusters in your WebSphere Application Server configuration: The `appCluster` cluster to host the sample application for the tutorial and the `xsCluster` cluster to host the data grid.

1. In the WebSphere Application Server administrative console, open the clusters panel. Click **Servers** > **Clusters** > **WebSphere application server clusters** > **New**.
2. Type `appCluster` as the cluster name, leave the **Prefer local** option selected, and click **Next**.
3. Create servers in the cluster. Create a server named `s1`, keeping the default options. Add an additional cluster member named `s2`.
4. Complete the remaining steps in the wizard to create the cluster. Save the changes.
5. Repeat these steps to create the `xsCluster` cluster. This cluster has two servers, named `xs1` and `xs2`.

Lesson checkpoint:

You enabled global security for the WebSphere Application Server cell, created users and user groups, and created clusters to host the application and data grid.

Module 2: Configure WebSphere eXtreme Scale to use WebSphere Application Server Authentication plug-ins

After you have created the WebSphere Application Server configuration, you can integrate WebSphere eXtreme Scale authentication with WebSphere Application Server.

When a WebSphere eXtreme Scale client connects to a container server that requires authentication, the client must provide a credential generator represented by the com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator interface. A credential generator is a factory to create a client credential. A client credential can be: a user name and password pair, a Kerberos ticket, a client certificate, or client identification data in any format that the client and server agree upon. See the Credential API documentation for more details. In this sample, the WebSphere eXtreme Scale client is the EmployeeManagment web application that is deployed in the appCluster cluster. The client credential is a WebSphere security token that represents the web user identity.

Learning objectives

With the lessons in this module, you learn how to:
- Configure client server security.
- Configure catalog server security.
- Configure container server security.
- Install and run the sample application.

Time required

This module takes approximately 60 minutes.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Related information:

"Lesson 2.1: Configure client server security"
The client properties file indicates the CredentialGenerator implementation class to use.

Credential API documentation

"Lesson 2.2: Configure catalog server security" on page 114
A catalog server contains two different levels of security information: The security properties that are common to all the WebSphere eXtreme Scale servers, including the catalog service and container servers, and the security properties that are specific to the catalog server.

Lesson 2.1: Configure client server security

The client properties file indicates the CredentialGenerator implementation class to use.

Configure the client properties file with the **`-Dobjectgrid.client.props`** JVM property. The file name specified for this property is an absolute file path, such as *samples_home*`/security/client2.props`. See Client properties file for more information about the client properties file.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Related information:

"Module 2: Configure WebSphere eXtreme Scale to use WebSphere Application Server Authentication plug-ins" on page 112
After you have created the WebSphere Application Server configuration, you can integrate WebSphere eXtreme Scale authentication with WebSphere Application Server.

Credential API documentation

Client properties file contents:
This example uses WebSphere Application Server security tokens as the client credential. The `client2.props` file is in the *samples_home*/`security` directory. The `client2.props` file includes the following settings:

securityEnabled
When set to `true`, indicates that the client must send available security information to the server.

credentialAuthentication
When set to `Supported`, indicates that the client supports credential authentication.

credentialGeneratorClass
Indicates the com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator class so the client retrieves the security tokens from the thread. See "Security integration with WebSphere Application Server" on page 601 for more information about how security tokens are retrieved.

Setting the client properties file using Java virtual machine (JVM) properties:
In the administrative console, complete the following steps to both the `s1` and `s2` servers in the `appCluster` cluster. If you are using a different topology, complete the following steps to all of the application servers to which the `EmployeeManagement` application is deployed.

1. **Servers** > **WebSphere application servers** > ***server_name*** > **Java and Process Management** > **Process definition** > **Java Virtual Machine**.
2. Create the following generic JVM property to set the location of the client properties file:

   ```
   -Dobjectgrid.client.props=samples_home/security/client2.props
   ```
3. Click **OK** and save your changes.

Lesson checkpoint:

You edited the client properties file and configured the servers in the `appCluster` cluster to use the client properties file. This properties file indicates the CredentialGenerator implementation class to use.

Lesson 2.2: Configure catalog server security

A catalog server contains two different levels of security information: The security properties that are common to all the WebSphere eXtreme Scale servers, including the catalog service and container servers, and the security properties that are specific to the catalog server.

The security properties that are common to the catalog servers and container servers are configured in the security XML descriptor file. An example of common properties is the authenticator configuration, which represents the user registry and authentication mechanism. See Security descriptor XML file for more information about the security properties.

To configure the security XML descriptor file, create a -Dobjectgrid.cluster.security.xml.url property in the Java virtual machine (JVM) argument. The file name specified for this property is in an URL format, such as `file:///`*`samples_home`*`/security/securityWAS2.xml`.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Related information:

"Module 2: Configure WebSphere eXtreme Scale to use WebSphere Application Server Authentication plug-ins" on page 112
After you have created the WebSphere Application Server configuration, you can integrate WebSphere eXtreme Scale authentication with WebSphere Application Server.

`securityWAS2.xml` file:

In this tutorial, the `securityWAS2.xml` file is in the *samples_home*`/security` directory. The content of the `securityWAS2.xml` file with the comments removed follows:

```
<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>
```

The following properties are defined in the `securityWAS2.xml` file:

securityEnabled
: The securityEnabled property is set to `true`, which indicates to the catalog server that the WebSphere eXtreme Scale global security is enabled.

authenticator
: The authenticator is configured as the com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenAuthenticator class. With this built-in implementation of the Authenticator plug-in, the WebSphere eXtreme Scale server can convert the security tokens to a Subject object. See "Security integration with WebSphere Application Server" on page 601 for more information about how the security tokens are converted.

`catServer2.props` file:

The server property file stores the server-specific properties, which include the server-specific security properties. See Server properties file for more information. You can configure the server property file with the `-Dobjectgrid.server.props` property in the JVM argument. Specify the file name value for this property is an absolute path, such as *`samples_home`*`/security/catServer2.props`. For this tutorial, a `catServer2.props` file is included in the *samples_home*`/security` directory. The content of the `catServer2.props` file with comments removed follows:

securityEnabled
: The securityEnabled property is set to `true` to indicate that this catalog server is a secure server.

credentialAuthentication
: The credentialAuthentication property is set to `Required`, so any client that is connecting to the server is required to provide a credential.

secureTokenManagerType
: The secureTokenManagerType is set to `none` to indicate that the authentication secret is not encrypted when joining the existing servers.

authenticationSecret
: The authenticationSecret property is set to `ObjectGridDefaultSecret`. This secret string is used to join the eXtreme Scale server cluster. When a server joins the data grid, it is challenged to present the secret string. If the secret string of the joining server matches the string in the catalog server, the joining server is accepted. If the string does not match, the join request is rejected.

transportType
: The transportType property is set to `TCP/IP` initially. Later in the tutorial, transport security is enabled.

Setting the server properties file with JVM properties:

Set the server properties file on the deployment manager server. If you are using a different topology than the topology for this tutorial, set the server properties file on all of the application servers that you are using to host catalog servers.

1. Open the Java virtual machine configuration for the server. In the administrative console, click **System administration** > **Deployment manager** > **Java and Process Management** > **Process definition** > **Java Virtual Machine**.
2. Add the following generic JVM arguments:
```
-Dobjectgrid.cluster.security.xml.url=file:///samples_home/security/securityWAS2.xml
-Dobjectgrid.server.props=samples_home/security/catServer2.props
```
3. Click **OK** and save your changes.

Lesson checkpoint:

You configured catalog server security by associating the `securityWAS2.xml` and `catServer2.props` files with the deployment manager, which hosts the catalog server process in the WebSphere Application Server configuration.

Lesson 2.3: Configure container server security

When a container server connects to the catalog service, the container server gets all the security configurations that are configured in the Object Grid Security XML file, such as authenticator configuration, the login session timeout value, and other configuration information. A container server also has its own server-specific security properties in the server property file.

Configure the server property file with the `-Dobjectgrid.server.props` Java virtual machine (JVM) property. The file name for this property is an absolute file path, such as *samples_home*`/security/server2.props`.

In this tutorial, the container servers are hosted in the `xs1` and `xs2` servers in the `xsCluster` cluster.

`server2.props` file:

The `server2.props` file is in the *samples_home*`/security` directory under the `WASSecurity` directory. The properties that are defined in the `server2.props` file follow:

securityEnabled
: The securityEnabled property is set to `true` to indicate that this container server is a secure server.

credentialAuthentication
: The credentialAuthentication property is set to `Required`, so any client that is connecting to the server is required to provide a credential.

secureTokenManagerType
: The secureTokenManagerType is set to `none` to indicate that the authentication secret is not encrypted when joining the existing servers.

authenticationSecret
: The authenticationSecret property is set to `ObjectGridDefaultSecret`. This secret string is used to join the eXtreme Scale server cluster. When a server joins the data grid, it is challenged to present the secret string. If the secret string of the joining server matches the string in the catalog server, the joining server is accepted. If the string does not match, the join request is rejected.

Setting the server properties file with JVM properties:

Set the server properties file on the xs1 and xs2 servers. If you are not using the topology for this tutorial, set the server properties file on all of the application servers that you are using to host container servers.

1. Open the Java virtual machine page for the server. **Servers** > **Application servers** > *server_name* > **Java and Process Management** > **Process definition** > **Java Virtual Machine**
2. Add the generic JVM arguments:
   `-Dobjectgrid.server.props=`*`samples_home`*`/security/server2.props`
3. Click **OK** and save your changes.

Lesson checkpoint:

Now the WebSphere eXtreme Scale server authentication is secured. By configuring this security, all the applications that try to connect to the WebSphere eXtreme Scale servers are required to provide a credential. In this tutorial, the WSTokenAuthenticator is the authenticator. As a result, the client is required to provide a WebSphere Application Server security token.

Lesson 2.4: Install and run the sample

After authentication is configured, you can install and run the sample application.

Creating a shared library for the `EmployeeData.jar` file:

1. In the WebSphere Application Server administrative console, open the **Shared Libraries** page. Click **Environment** > **Shared libraries**.
2. Choose the **cell** scope.
3. Create the shared library. Click **New**. Enter `EmployeeManagementLIB` as the **Name**. Enter the path to the `EmployeeData.jar` in the classpath, for example, *samples_home*`/WASSecurity/EmployeeData.jar`.

4. Click **Apply**.

Installing the sample:

1. Install the `EmployeeManagement.ear` file.
   a. To begin the installation, click **Applications** > **New application** > **New Enterprise Application**. Choose the detailed path for installing the application.
   b. On the **Map modules to servers** step, specify the `appCluster` cluster to install the EmployeeManagementWeb module.
   c. On the **Map shared libraries** step, select the `EmployeeManagementWeb` module.
   d. Click **Reference shared libraries**. Select the `EmployeeManagementLIB` library.
   e. Map the `webUser` role to **All Authenticated in Application's Realm**.
   f. Click **OK**.

   The clients run in the `s1` and `s2` servers in this cluster.
2. Install the sample `XSDeployment.ear` file.
   a. To begin the installation, click **Applications** > **New application** > **New Enterprise Application**. Choose the detailed path for installing the application.
   b. On the **Map modules to servers** step, specify the `xsCluster` cluster to install the `XSDeploymentWeb` web module.
   c. On the **Map shared libraries** step, select the `XSDeploymentWeb` module.
   d. Click **Reference shared libraries**. Select the `EmployeeManagementLIB` library.
   e. Click **OK**.

   The `xs1` and `xs2` servers in this cluster host the container servers.
3. Restart the deployment manager. When the deployment manager starts, the catalog server also starts. If you look at the `SystemOut.log` file of the deployment manager, you can see the following message that indicates that the eXtreme Scale server properties file is loaded.
   ```
   CWOBJ0913I: Server property files have been loaded:
   /wxs_samples/security/catServer2.props.
   ```
4. Restart the xsCluster cluster. When the xsCluster starts, the `XSDeployment` application starts, and a container server is started on the xs1 and xs2 servers respectively. If you look at the `SystemOut.log` file of the xs1 and xs2 servers, the following message that indicates the server properties file is loaded is displayed:
   ```
   CWOBJ0913I: Server property files have been loaded:
   /wxs_samples/security/server2.props.
   ```
5. Restart the appClusters cluster. When the cluster appCluster starts, the EmployeeManagement application also starts. If you look at the `SystemOut.log` file of the s1 and s2 servers, you can see the following message that indicates that the client properties file is loaded.
   ```
   CWOBJ0924I: The client property file {0} has been loaded.
   ```
   You can ignore the warning messages regarding the authenticationRetryCount, transportType, and clientCertificateAuthentication properties. The default values are be used because the values were not specified in the properties file.If you are using WebSphere eXtreme Scale Version 7.0, the English-only CWOBJ9000I message displays to indicate that the client property file has been loaded. If you do not see the expected message, verify that you configured the

-Dobjectgrid.server.props or -Dobjectgrid.client.props property in the JVM argument. If you do have the properties configured, make sure the dash (-) is a UTF character.

Running the sample application:

1. Run the `management.jsp` file. In a web browser, access `http://<your_servername>:<port>/EmployeeManagementWeb/management.jsp`. For example, you might use the following URL: `http://localhost:9080/EmployeeManagementWeb/management.jsp`.
2. Provide authentication to the application. Enter the credentials of the user that you mapped to the webUser role. By default, this user role is mapped to all authenticated users. Type `admin1` as your user ID and `admin1` as your password. A page to display, add, update, and delete employees displays.
3. Display employees. Click **Display an Employee**. Enter `emp1@acme.com` as the email address, and click **Submit**. A message displays that the employee cannot be found.
4. Add an employee. click **Add an Employee**. Enter `emp1@acme.com` as the email address, enter `Joe` as the first name, and `Doe` as the last name. Click **Submit**. A message displays that an employee with the `emp1@acme.com` address has been added.
5. Display the new employee. Click **Display an Employee**. Enter `emp1@acme.com` as the email address with empty fields for the first and last names, and click **Submit**. A message displays that the employee has been found, and the correct names are displayed in the first name and last name fields.
6. Delete the employee. Click **Delete an employee**. Enter `emp1@acme.com` and click **Submit**. A message is displayed that the employee has been deleted.

Lesson checkpoint:

You installed and ran the sample application. Because this tutorial uses WebSphere Application Server integration, you cannot see the scenario when a client fails to authenticate to the eXtreme Scale server. If the user authenticates to the WebSphere Application Server successfully, eXtreme Scale is also successfully authenticated.

Module 3: Configure transport security

Configure transport security to secure data transfer between the clients and servers in the configuration.

In the previous module in the tutorial, you enabled WebSphere eXtreme Scale authentication. With authentication, any application that tries to connect to the WebSphere eXtreme Scale server is required to provide a credential. Therefore, no unauthenticated client can connect to the WebSphere eXtreme Scale server. The clients must be an authenticated application that is running in a WebSphere Application Server cell.

With the configuration up to this module, the data transfer between the clients in the appCluster cluster and servers in the xsCluster cluster is not encrypted. This configuration might be acceptable if your WebSphere Application Server clusters are installed on servers behind a firewall. However, in some scenarios, non-encrypted traffic is not accepted for some reasons even though the topology is protected by firewall. For example, a government policy might enforce encrypted traffic. WebSphere eXtreme Scale supports Transport Layer Security/Secure Sockets Layer (TLS/SSL) for secure communication between ObjectGrid endpoints, which include client servers, container servers, and catalog servers.

In this sample deployment, the eXtreme Scale clients and container servers are all running in the WebSphere Application Server environment. Client or server properties are not necessary to configure the SSL settings because the eXtreme Scale transport security is managed by the Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) transport settings. WebSphere eXtreme Scale servers use the same Object Request Broker (ORB) instance as the application servers in which they run. Specify all the SSL settings for client and container servers in the WebSphere Application Server configuration using these CSIv2 transport settings. The catalog server has its own proprietary transport paths that do not use Internet Inter-ORB Protocol (IIOP) or Remote Method Invocation (RMI). Because of these proprietary transport paths, the catalog server cannot be managed by the WebSphere Application Server CSIV2 transport settings. Therefore, you must configure the SSL properties in the server properties file for the catalog server.

Learning objectives

After completing the lessons in this module, you know how to:
- Configure CSIv2 inbound and outbound transport.
- Add SSL properties to the catalog server properties file.
- Check the ORB properties file.
- Run the sample.

Time required

This module takes approximately 60 minutes.

Prerequisites

This step of the tutorial builds upon the previous modules. Complete the previous modules in this tutorial before you configure transport security.

Lesson 3.1: Configure CSIv2 inbound and outbound transport

To configure Transport Layer Security/Secure Sockets Layer (TLS/SSL) for the server transport, set the Common Secure Interoperability Protocol Version 2 (CSIv2) inbound transport and CSIv2 outbound transport to `SSL-Required` for all the WebSphere Application Server servers that host clients, catalog servers, and container servers.

In the tutorial example topology, you must set these properties for the, s1, s2, xs1, and xs2 application servers. The following steps configure the inbound and outbound transports for all the servers in the configuration.

Set the inbound and outbound transports in the administrative console. Make sure that administrative security is enabled.
- **WebSphere Application Server Version 6.1**: Click **Security** > **Secure Administration** > **Application..** > **RMI/IIOP Security** and change the transport type to **SSL-Required**.
- **WebSphere Application Server Version 7.0**: Click **Security** > **Global Security** > **RMI/IIOP Security** > **CSIv2 inbound communications**. Change the transport type under the CSIv2 Transport Layer to **SSL-Required**. Repeat this step to configure CSIv2 outbound communications.

You can use centrally managed endpoint security settings, or you can configure SSL repositories. See Common Secure Interoperability Version 2 transport inbound settings for more information.

Lesson 3.2: Add SSL properties to the catalog server properties file

The catalog server has its own proprietary transport paths that cannot be managed by the WebSphere Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) transport settings. Therefore, you must configure the Secure Sockets Layer (SSL) properties in the server properties file for the catalog server.

To configure catalog server security, additional steps are necessary because the catalog server has its own proprietary transport paths. These transport paths cannot be managed by the Application Server CSIV2 transport settings.

1. Edit the SSL properties in the `catServer2.props` file. To configure catalog server security, uncomment the following SSL properties in the catalog server properties file. For this tutorial, the catalog server properties are in the `catServer2.props` file. Update the keyStore and trustStore properties to refer to the proper location in your environment.

   ```
   #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
   ```

   The `catServer2.props` file is using the default WebSphere Application Server node level keystore and truststore. If you are deploying a more complex deployment environment, you must choose the correct keystore and truststore. In some cases, you must create a keystore and truststore and import the keys from keystores from the other servers. Notice that the `WebAS` string is the default password of the WebSphere Application Server keystore and truststore. See Default self-signed certificate configuration for more details.
2. In the `catServer2.props` file, update the value of the transportType property. For previous steps of the tutorial, the value was set to `TCP/IP`. Change the value to `SSL-Required`.
3. Restart the deployment manager to activate the changes to the catalog server security settings.

Lesson checkpoint:

You configured the SSL properties for the catalog server.

Lesson 3.3: Run the sample

Restart all the servers and run the sample application again. You should be able to run through the steps without any problems.

See "Lesson 2.4: Install and run the sample" on page 117 for more information about running and installing the sample application.

Lesson checkpoint:

You ran the sample application with transport security enabled.

Module 4: Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server

Now that you have configured authentication for clients, you can further configure authentication to give different users varying permissions. For example, an operator user might only be able to view data, while an administrator user can perform all operations.

After authenticating a client, as in the previous module in this tutorial, you can give security privileges through eXtreme Scale authorization mechanisms. The previous module of this tutorial demonstrated how to enable authentication for a data grid using integration with WebSphere Application Server. As a result, no unauthenticated client can connect to the eXtreme Scale servers or submit requests to your system. However, every authenticated client has the same permission or privileges to the server, such as reading, writing, or deleting data that is stored in the ObjectGrid maps. Clients can also issue any type of query.

This part of the tutorial demonstrates how to use eXtreme Scale authorization to give authenticated users varying privileges. WebSphere eXtreme Scale uses a permission-based authorization mechanism. You can assign different permission categories that are represented by different permission classes. This module features the MapPermission class. For a list of all possible permissions, see Client authorization programming.

In WebSphere eXtreme Scale, the `com.ibm.websphere.objectgrid.security.MapPermission` class represents permissions to the eXtreme Scale resources, specifically the methods of the ObjectMap or JavaMap interfaces. WebSphere eXtreme Scale defines the following permission strings to access the methods of ObjectMap and JavaMap:

- **read**: Grants permission to read the data from the map.
- **write**: Grants permission to update the data in the map.
- **insert**: Grants permission to insert the data into the map.
- **remove**: Grants permission to remove the data from the map.
- **invalidate**: Grants permission to invalidate the data from the map.
- **all**: Grants all permissions to read, write, insert, remote, and invalidate.

The authorization occurs when an eXtreme Scale client uses a data access API, such as the ObjectMap ,JavaMap, or EntityManager APIs. The run time checks corresponding map permissions when the method is called. If the required permissions are not granted to the client, an AccessControlException exception results. This tutorial demonstrates how to use Java Authentication and Authorization Service (JAAS) authorization to grant authorization map access for different users.

Learning objectives

After completing the lessons in this module, you know how to:

- Enable authorization for WebSphere eXtreme Scale.
- Enable user-based authorization.
- Configure group-based authorization.

Time required

This module takes approximately 60 minutes.

Prerequisites

You must complete the prior modules in this tutorial before configuring authentication.

Related concepts:

Client authorization programming
WebSphere eXtreme Scale supports Java Authentication and Authorization Service (JAAS) authorization that is ready to use and also supports custom authorization using the ObjectGridAuthorization interface.

Lesson 4.1: Enable WebSphere eXtreme Scale authorization

To enable authorization in WebSphere eXtreme Scale, you must enable security on a specific ObjectGrid.

To enable authorization on the ObjectGrid, you must set the **`securityEnabled`** attribute to true for that particular ObjectGrid in the XML file. For this tutorial, you can either use the `XSDeployment_sec.ear` file in the *samples_home*/`WASSecurity` directory, which has already has security set in the `objectGrid.xml` file, or you can edit the existing `objectGrid.xml` file to enable security. This lesson demonstrates how to edit the file to enable security.

1. Extract the files in the `XSDeployment.ear` file, and then unzip the `XSDeploymentWeb.war` file.
2. Open the `objectGrid.xml` file and set the securityEnabled attribute to `true` on the ObjectGrid level. See an example of this attribute in the following 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="Grid" securityEnabled="true">
               <backingMap name="Map1" />
           </objectGrid>
       </objectGrids>

   </objectGridConfig>
   ```

   If you have multiple ObjectGrids defined, then you must set this attribute on each data grid.
3. Repackage the `XSDeploymentWeb.war` and `XSDeployment.ear` files to include your changes. Name the file `XSDeployment_sec.ear` so you do not overwrite the original package.
4. Uninstall the existing `XSDeployment` application and install the `XSDeployment_sec.ear` file. See "Lesson 2.4: Install and run the sample" on page 117 for more information about deploying applications.

Lesson checkpoint:

You enabled security on the ObjectGrid, which also enables authorization on the data grid.

Lesson 4.2: Enable user-based authorization

In the authentication module of this tutorial, you created two users: `operator1` and `admin1`. You can assign varying permissions to these users with Java Authentication and Authorization Service (JAAS) authorization.

Defining the Java Authentication and Authorization Service (JAAS) authorization policy using user principals:
You can assign permissions to the users that you previously created. Assign the `operator1` user read permissions only to all maps. Assign the `admin1` user all permissions. Use the JAAS authorization policy file to grant permissions to principals.

Edit the JAAS authorization file. The `xsAuth2.policy` file is in the *samples_home*/`security` directory:

```
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";
};
```

In this file, the `http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction` codebase is a specially reserved URL for ObjectGrid. All ObjectGrid permissions that are granted to principals should use this special code base. The following permissions are assigned in this file:
- The first grant statement grants `read` map permission to the `operator1` principal. The `operator1` user has only map read permission to the Map1 map the Grid ObjectGrid instance.
- The second grant statement grants all map permission to the admin1 principal. The `admin1` user has all permissions to the Map1 map in the Grid ObjectGrid instance.
- The principal name is `defaultWIMFileBasedRealm/operator1`, but not `Operator1`. WebSphere Application Server automatically adds the realm name to the principal name when federated repositories are used as the user account registry. Adjust this value if needed.

Setting the JAAS authorization policy file using JVM properties:
Use the following steps to set JVM properties for the xs1 and xs2 servers, which are in the xsCluster cluster. If you are using a topology that is different from the sample topology that is used in this tutorial, set the file on all of your container servers.
1. In the administrative console, click **Servers** > **Application servers** > *server_name* > **Java and Process Management** > **Process definition** > **Java Virtual Machine.**
2. Add generic JVM arguments.

   Note: When containers are running in WebSphere Application Server, you cannot use the `-Djava.security.policy` argument because this file overrides the WebSphere Application Server administrative access authorization. Therefore, use `-Djava.security.auth.policy` to set the JAAS authorization policy.

   Enter the following generic JVM arguments or replace the `-Djava.security.auth.policy` entry with the following text:

   `-Djava.security.auth.policy=`*`samples_home`*`/security/xsAuth2.policy`
3. Click **OK** and save your changes.

Running the sample application to test authorization:
You can use the sample application to test the authorization settings. The administrator user continues to have all permissions in the Map1 map, including

displaying and adding employees. The operator user should only be able to view employees because that user was assigned read permission only.

1. Restart all of the application servers that are running container servers.
2. Open the `EmployeeManagementWeb` application. In a web browser, open `http://<host>:<port>/EmployeeManagermentWeb/management.jsp`.
3. Log in to the application as an administrator. Use the user name `admin1` and password `admin1`.
4. Attempt to display an employee. Click **Display an Employee** and search for the `authemp1@acme.com` email address. A message displays that the user cannot be found.
5. Add an employee. Click **Add an Employee**. Add the email `authemp1@acme.com`, the first name `Joe`, and the last name `Doe`. Click **Submit**. A message displays that the employee has been added.
6. Log in as the operator user. Open a second Web browser window and open `http://<host>:<port>/EmployeeManagermentWeb/management.jsp`. Use the user name `operator1` and password `operator1`.
7. Attempt to display an employee. Click **Display an Employee** and search for the `authemp1@acme.com` email address. The employee is displayed.
8. Add an employee. Click **Add an Employee**. Add the email `authemp2@acme.com`, the first name `Joe`, and the last name `Doe`. Click **Submit**. The following message displays:

   ```
   An exception occurs when Add the employee. See below for detailed exception messages.
   ```

   The following exception is in the exception chain:

   ```
   java.security.AccessControlException: Access denied
   (com.ibm.websphere.objectgrid.security.MapPermission Grid.Map1 insert)
   ```

   This message displays because the `operator1` user does not have permission to insert data into the Map1 map.

If you are running with a version of WebSphere Application Server that is earlier than Version 7.0.0.11, you might see a java.lang.StackOverflowError error on the container server. This error is caused by a problem with the IBM Developer Kit. The problem is fixed in the IBM Developer Kit that is shipped with WebSphere Application Server Version 7.0.0.11 and later.

Lesson checkpoint:

In this lesson, you configured authorization by assigning permissions to specific users.

Lesson 4.3: Configure group-based authorization

In the previous lesson, you assigned individual user-based authorization with user principals in the Java Authentication and Authorization Service. (JAAS) authorization policy. However, when you have hundreds or thousands of users, use group-based authorization, which authorizes access based on groups instead of individual users.

Unfortunately, the Subject object that is authenticated from the WebSphere Application Server only contains a user principal. This object does not contain a group principal. You can add a custom login module to populate the group principal into the Subject object.

For this tutorial, the custom login module is named com.ibm.websphere.samples.objectgrid.security.lm.WASAddGroupLoginModule. The module is in the `groupLM.jar` file. Place this JAR file in the `WAS-INSTALL/lib/ext` directory.

The WASAddGroupLoginModule retrieves the public group credential from the WebSphere Application Server subject and creates a Group principal, com.ibm.websphere.samples.objectgrid.security.WSGroupPrincipal, to represent the group. This group principal can then be used for group authorization. The groups are defined in the `xsAuthGroup2.policy` file:

```
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";
};
```

The principal name is the `WSGroupPrincipal`, which represents the group.

Adding the custom login module:
The custom login module must be added to each of the following system login module entries: If you are using Lightweight Third Party Authentication (LTPA), add the entry to the RMI_INBOUND login modules. LTPA is the default authentication mechanism for WebSphere Application Server Version 7.0. For a WebSphere Application Server Network Deployment configuration, you only need to configure the LTPA authentication mechanism configuration entries.

Use the following steps to configure the supplied com.ibm.websphere.samples.objectgrid.security.lm.WASAddGroupLoginModule login module:

1. In the administrative console, click **Security** > **Global Security** > **Java Authentication and Authorization Service** > **System logins** > ***login_module_name*** > **JAAS login modules** > **New**.
2. Enter the class name as `com.ibm.websphere.sample.xs.security.lm.WASAddGroupLoginModule`.
3. Optional: Add a property `debug` and set the value to `true`.
4. Click **Apply** to add the new module to the login module list.

Setting the JAAS Authorization Policy file using JVM Properties:

In the administrative console, perform the following steps to xs1 and xs2 servers in the xsCluster. If a different deployment topology is used, perform the following steps to the application servers that host the container servers.

1. In the administrative console, click **Servers** > **Application servers** > *server_name* > **Java and Process management** > **Process definition** > **Java virtual machine**
2. Add generic JVM arguments.

   Note: When containers are running in WebSphere Application Server, you cannot use the `-Djava.security.policy` argument because this file overrides the WebSphere Application Server administrative access authorization. Therefore, use `-Djava.security.auth.policy` to set the JAAS authorization policy.

Enter the following generic JVM arguments or replace the `-Djava.security.auth.policy` entry with the following text:

```
-Djava.security.auth.policy=samples_home/security/xsAuthGroup2.policy
```

3. Click **OK** and save your changes.

Testing group authorization with the sample application:

You can test that group authorization is configured by the login module with the sample application.

1. Restart the container servers. For this tutorial, the container servers are the xs1 and xs2 servers.
2. Log in to the sample application. In a web browser, open `http://<host>:<port>/EmployeeManagementWeb/management.jsp` and login with the user name `admin1` and password `admin1`.
3. Display an employee. Click **Display an Employee** and search for the `authemp2@acme.com` email address. A message displays that the user cannot be found.
4. Add an employee. Click **Add an Employee**. Add the email `authemp2@acme.com`, the first name `Joe`, and the last name `Doe`. Click **Submit**. A message displays that the employee has been added.
5. Log in as the operator user. Open a second web browser window and open the following URL: `http://<host>:<port>/EmployeeManagermentWeb/management.jsp`. Use the user name `operator1` and password `operator1`.
6. Attempt to display an employee. Click **Display an Employee** and search for the `authemp2@acme.com` email address. The employee is displayed.
7. Add an employee. Click **Add an Employee**. Add the email `authemp3@acme.com`, the first name `Joe`, and the last name `Doe`. Click **Submit**. The following message displays:

   ```
   An exception occurs when Add the employee. See below for detailed exception messages.
   ```

   The following exception is in the exception chain:

   ```
   java.security.AccessControlException: Access denied
   (com.ibm.websphere.objectgrid.security.MapPermission Grid.Map1 insert)
   ```

   This message displays because the operator user does not have permission to insert data into the Map1 map.

Lesson checkpoint:

You configured groups to simplify the assignment of permission to the users of your application.

Module 5: Use the `xscmd` tool to monitor data grids and maps

You can use the **`xscmd`** tool to show the primary data grids and map sizes of the `Grid` data grid. The **`xscmd`** tool uses the MBean to query all of the data grid artifacts, such as primary shards, replica shards, container servers, map sizes, and so on.

In this tutorial, the container and catalog servers are running in WebSphere Application Server application servers. The WebSphere eXtreme Scale run time registers the Managed Beans (MBean) with the MBean server that is created by the WebSphere Application Server run time. The security that is used by the **xscmd** tool is provided by the WebSphere Application Server MBean security. Therefore, WebSphere eXtreme Scale specific security configuration is not necessary.

1. Using a command-line tool, open the *DMGR_PROFILE*/bin directory.
2. Run the **xscmd** tool.

   Use the **-c showPlacement -sf P** command to list the placement of the primary shards. Linux UNIX

   ```
   xscmd.sh -g Grid -ms mapSet -c showPlacement -sf P
   ```

   Windows

   ```
   xscmd.bat -g Grid -ms mapSet -c showPlacement -sf P
   ```

   Before you can view the output, you are prompted to log in with your WebSphere Application Server ID and password.

Related tasks:

"Monitoring with the **xscmd** utility" on page 535
The **xscmd** utility replaces the **xsadmin** sample utility as a fully supported monitoring and administration tool. With the **xscmd** utility, you can display textual information about your WebSphere eXtreme Scale topology.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Lesson checkpoint

You used the **xscmd** tool in WebSphere Application Server.

Tutorial: Integrate WebSphere eXtreme Scale security in a mixed environment with an external authenticator

This tutorial demonstrates how to secure WebSphere eXtreme Scale servers that are partially deployed in a WebSphere Application Server environment.

In the deployment for this tutorial, the container servers are deployed in WebSphere Application Server. The catalog server is deployed as stand-alone server, and is started in a Java Standard Edition (Java SE) environment.

Because the catalog server is not deployed in WebSphere Application Server, you cannot use the WebSphere Application Server Authentication plug-ins. For more information about the process of configuring WebSphere Application Server Authentication plug-ins, see "Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server" on page 106. In this tutorial, a different authenticator is required for catalog server authentication. You configure a keystore authenticator to authenticate the clients.

Learning objectives

The learning objectives for this tutorial follow:

- Configure WebSphere eXtreme Scale to use the KeyStoreLoginAuthenticator plug-in

- Configure WebSphere eXtreme Scale transport security to use WebSphere Application Server CSIv2 configuration and the WebSphere eXtreme Scale properties file
- Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server
- Use the **xscmd** utility to monitor the data grids and maps that you created in the tutorial.

Time required

This tutorial takes approximately 4 hours from start to finish.

Introduction: Security in a mixed environment

In this tutorial, you integrate WebSphere eXtreme Scale security in a mixed environment. The container servers run within WebSphere Application Server, and the catalog service runs in stand-alone mode. Because the catalog server is in stand-alone mode, you must configure an external authenticator.

Important: If both your container servers and catalog server are running within WebSphere Application Server, you can use the WebSphere Application Server Authentication plug-ins or an external authenticator. For more information about using the WebSphere Application Server Authentication plug-ins, see "Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server" on page 106.

Learning objectives

The learning objectives for this tutorial follow:
- Configure WebSphere eXtreme Scale to use the KeyStoreLoginAuthenticator plug-in
- Configure WebSphere eXtreme Scale transport security to use WebSphere Application Server CSIv2 configuration and the WebSphere eXtreme Scale properties file
- Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server
- Use the **xscmd** utility to monitor the data grids and maps that you created in the tutorial.

Time required

This tutorial takes approximately 4 hours from start to finish.

Skill level

Intermediate.

Audience

Developers and administrators that are interested in the security integration between WebSphere eXtreme Scale and WebSphere Application Server and configuring external authenticators.

System requirements

- WebSphere Application Server Version 6.1 or Version 7.0.0.11 or later with the following fixes applied:interim fix PM20613 and interim fix PM15818.
- The catalog server must be running on a stand-alone installation, not an installation that is integrated with WebSphere Application Server.
- Update the Java runtime to apply the following fix: IZ79819: IBMJDK FAILS TO READ PRINCIPAL STATEMENT WITH WHITESPACE FROM SECURITY FILE
- The stand-alone node that runs the catalog service must use the IBM Software Development Kit Version 1.6 J9. This Software Development Kit is included in the WebSphere Application Server installation. The catalog server node must be a stand-alone installation because you cannot run the **startOgServer** command within an installation of WebSphere eXtreme Scale on WebSphere Application Server.

This tutorial uses four WebSphere Application Server application servers and one deployment manager to demonstrate the sample.

Prerequisites

A basic understanding of the following items is helpful before you start this tutorial:

- WebSphere eXtreme Scale programming model
- Basic WebSphere eXtreme Scale security concepts
- Basic WebSphere Application Server security concepts

For a background information about WebSphere eXtreme Scale and WebSphere Application Server security integration, see "Security integration with WebSphere Application Server" on page 601.

Module 1: Prepare the mixed WebSphere Application Server and stand-alone environment

Before you start the tutorial, you must create a basic topology that includes container servers that run within WebSphere Application Server. In this tutorial, the catalog servers run in stand-alone mode.

Learning objectives

With the lessons in this module, you learn how to:

- Understand the mixed topology and the files that are necessary for the tutorial
- Configure WebSphere Application Server to run the container servers

Time required

This module takes approximately 60 minutes.

Lesson 1.1: Understand the topology and get the tutorial files

To prepare your environment for the tutorial, you must configure the catalog and container servers for the topology.

This lesson guides you through the sample topology and applications that are used to in the tutorial. To begin running the tutorial, you must download the applications and place the configuration files in the correct locations for your environment. You can download the sample application from the IBM elastic caching community.

Topology: In this tutorial, you create the following clusters in the WebSphere Application Server cell:

- **appCluster cluster**: Hosts the EmployeeManagement sample enterprise application. This cluster has two application servers: s1 and s2.
- **xsCluster cluster**: Hosts the eXtreme Scale container servers. This cluster has two application servers: xs1 and xs2.

In this deployment topology, the s1 and s2 application servers are the client servers that access data that is being stored in the data grid. The xs1 and xs2 servers are the container servers that host the data grid.

Alternative configuration: You can host all of the application servers in a single cluster, such as in the appCluster cluster. With this configuration, all of the servers in the cluster are both clients and container servers. This tutorial uses two clusters to distinguish between the application servers that are hosting the clients and container servers.

In this tutorial, you configure a catalog service domain that consists of a remote server that is not in the WebSphere Application Server cell. This configuration is not the default, which results in the catalog servers running on the deployment manager and other processes in the WebSphere Application Server cell. See "Creating catalog service domains in WebSphere Application Server" on page 299 for more information about creating a catalog service domain that consists of remote servers.

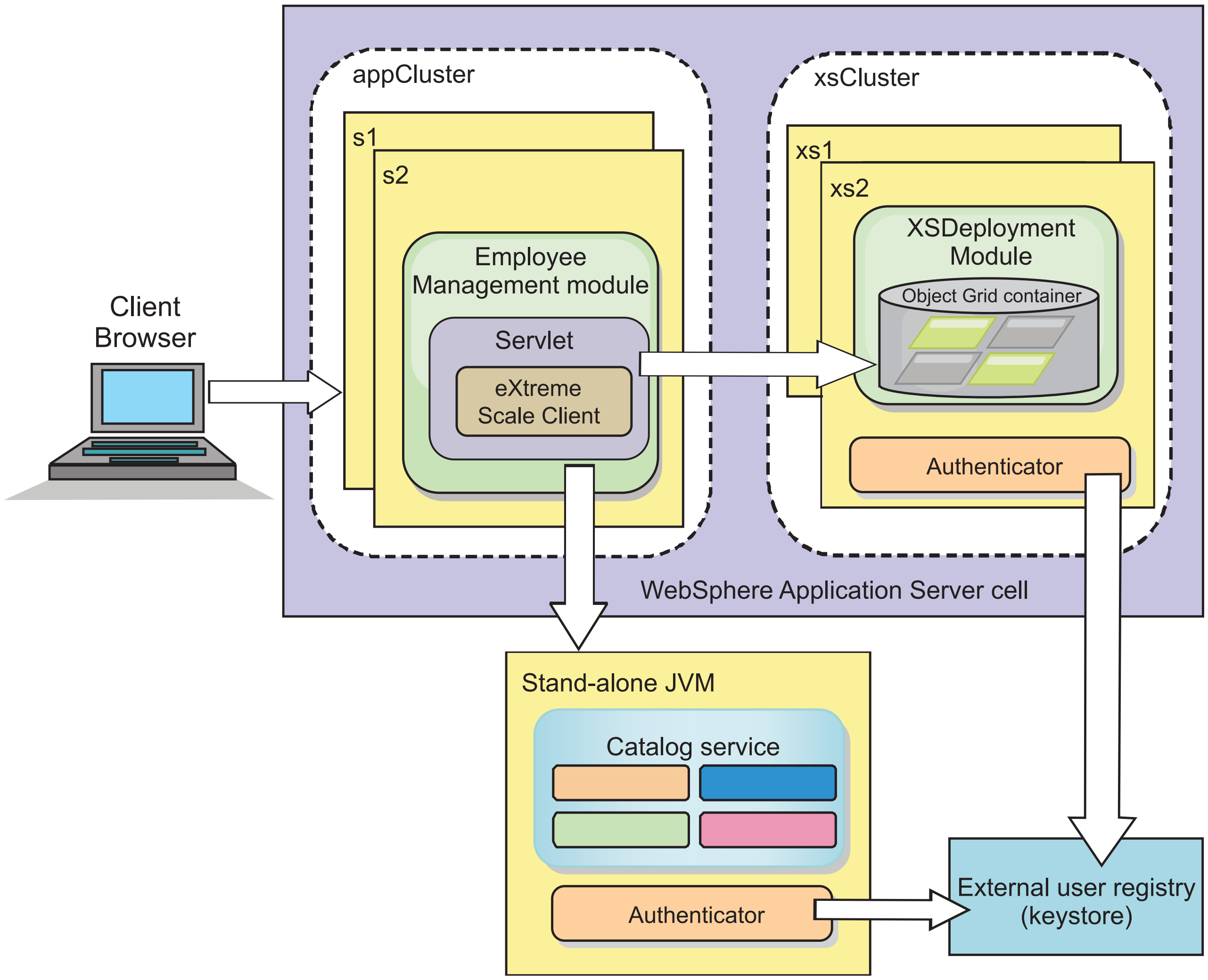


Figure 22. Tutorial topology

Applications: In this tutorial, you are using two applications and one shared library file:

- **`EmployeeManagement.ear`**: The `EmployeeManagement.ear` application is a simplified Java 2 Platform, Enterprise Edition (J2EE) enterprise application. It contains a web module to manage the employee profiles. The web module contains the `management.jsp` file to display, insert, update, and delete employee profiles that are stored in the container servers.
- **`XSDeployment.ear`**: This application contains an enterprise application module with no application artifacts. The cache objects are packaged in the `EmployeeData.jar` file. The `EmployeeData.jar` file is deployed as a shared library for the `XSDeployment.ear` file, so that the `XSDeployment.ear` file can access the classes. The purpose of this application is to package the eXtreme Scale configuration file and property file. When this enterprise application is started, the eXtreme Scale configuration files are automatically detected by the eXtreme Scale run time, so the container servers are created. These configuration files include the `objectGrid.xml` and `objectGridDeployment.xml` files.
- **`EmployeeData.jar`**: This jar file contains one class: the com.ibm.websphere.sample.xs.data.EmployeeData class. This class represents employee data that is stored in the grid. This Java archive (JAR) file is deployed with the `EmployeeManagement.ear` and `XSDeployment.ear` files as a shared library.

Get the tutorial files:

1. Download the `WASSecurity.zip` and `security_extauth.zip` files from the WebSphere eXtreme Scale wiki.
2. Extract the `WASSecurity.zip` file to a directory for viewing the binary and source artifacts, for example a `wxs_samples/` directory. This directory is referred to as *samples_home* for the remainder of the tutorial. Refer to the `README.txt` file in the package for a description of the contents and how to load the source into your Eclipse workspace. The following ObjectGrid configuration files are in the `META-INF` directory:
   - `objectGrid.xml`
   - `objectGridDeployment.xml`
3. Create a directory to store the property files that are used to secure this environment. For example, you might create the `/opt/wxs/security` directory.
4. Extract the `security_extauth.zip` file to *samples_home*. The `security_extauth.zip` file contains the following security configuration files that are used in this tutorial:. These configuration files follow:
   - `catServer3.props`
   - `server3.props`
   - `client3.props`
   - `security3.xml`
   - `xsAuth3.props`
   - `xsjaas3.config`
   - `sampleKS3.jks`

About the configuration files:

The `objectGrid.xml` and `objectGridDeployment.xml` files create the data grids and maps that store the application data.

These configuration files must be named `objectGrid.xml` and `objectGridDeployment.xml`. When the application server starts, eXtreme Scale detects these files in the `META-INF` directory of the EJB and web modules. If these files are found, it assumed that the Java virtual machine (JVM) acts as a container server for the defined data grids in the configuration files.

`objectGrid.xml` file

The `objectGrid.xml` file defined one ObjectGrid named `Grid`. The `Grid` data grid has one map, the `Map1` map, that stores the employee profile for the application.

```
<?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` file

The `objectGridDeployment.xml` file specifies how to deploy the `Grid` data grid. When the grid is deployed, it has five partitions and one synchronous replica.

```
<?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>
```

Lesson checkpoint:

In this lesson, you learned about the topology for the tutorial and added the configuration files and sample applications to your environment.

Lesson 1.2: Configure the WebSphere Application Server environment

To prepare your environment for the tutorial, you must configure WebSphere Application Server security. Enable administration and application security using internal file-based federated repositories as a user account registry. Then, you can create server clusters to host the client application and container servers. You also must create and start the catalog servers.

The following steps were written using WebSphere Application Server Version 7.0. However, you can also apply the concepts apply to earlier versions of WebSphere Application Server.

Configure WebSphere Application Server security:
Create and augment profiles for the deployment manager and nodes with

WebSphere eXtreme Scale. See 7.1.1 “Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server” on page 186 for more information.

Configure WebSphere Application Server security.

1. In the WebSphere Application Server administrative console, click **Security** > **Global Security**.
2. Select **Federated repositories** as the **Available realm definition**. Click **Set as current**.
3. Click **Configure..** to go to the Federated repositories panel.
4. Enter the **Primary administrative user name**, for example, `admin`. Click **Apply.**
5. When prompted, enter the administrative user password and click **OK**. Save your changes.
6. On the **Global Security** page, verify that **Federated repositories** setting is set to the current user account registry.
7. Select the following items: **Enable administrative security**, **Enable application security**, and **Use Java 2 security to restrict application access to local resources**. Click **Apply** and save your changes.
8. Restart the deployment manager and any running application servers.

The WebSphere Application Server administrative security is enabled using the internal file-based federated repositories as the user account registry.

Create server clusters:

Create two server clusters in your WebSphere Application Server configuration: The `appCluster` cluster to host the sample application for the tutorial and the `xsCluster` cluster to host the data grid.

1. In the WebSphere Application Server administrative console, open the clusters panel. Click **Servers** > **Clusters** > **WebSphere application server clusters** > **New**.
2. Type `appCluster` as the cluster name, leave the **Prefer local** option selected, and click **Next**.
3. Create servers in the cluster. Create a server named `s1`, keeping the default options. Add an additional cluster member named `s2`.
4. Complete the remaining steps in the wizard to create the cluster. Save the changes.
5. Repeat these steps to create the `xsCluster` cluster. This cluster has two servers, named `xs1` and `xs2`.

Create a catalog service domain:

After configuring the server cluster and security, you must define where catalog servers start.

Define a catalog service domain in WebSphere eXtreme Scale

1. In the WebSphere Application Server administrative console, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains**.
2. Create the catalog service domain. Click **New**. Create the catalog service domain with the name `catalogService1`, and enable the catalog service domain as the default.
3. Add remote servers to the catalog service domain. Select **Remote server**. Provide the host name where the catalog server is running. Use the listener port value of `16809` for this example.
4. Click **OK** and save your changes.

Lesson checkpoint:

You enabled security in WebSphere Application Server, and created the server topolgy for WebSphere eXtreme Scale.

Module 2: Configure WebSphere eXtreme Scale authentication in a mixed environment

By configuring authentication, you can reliably determine the identity of the requester. WebSphere eXtreme Scale supports both client-to-server and server-to-server authentication.

Authentication flow

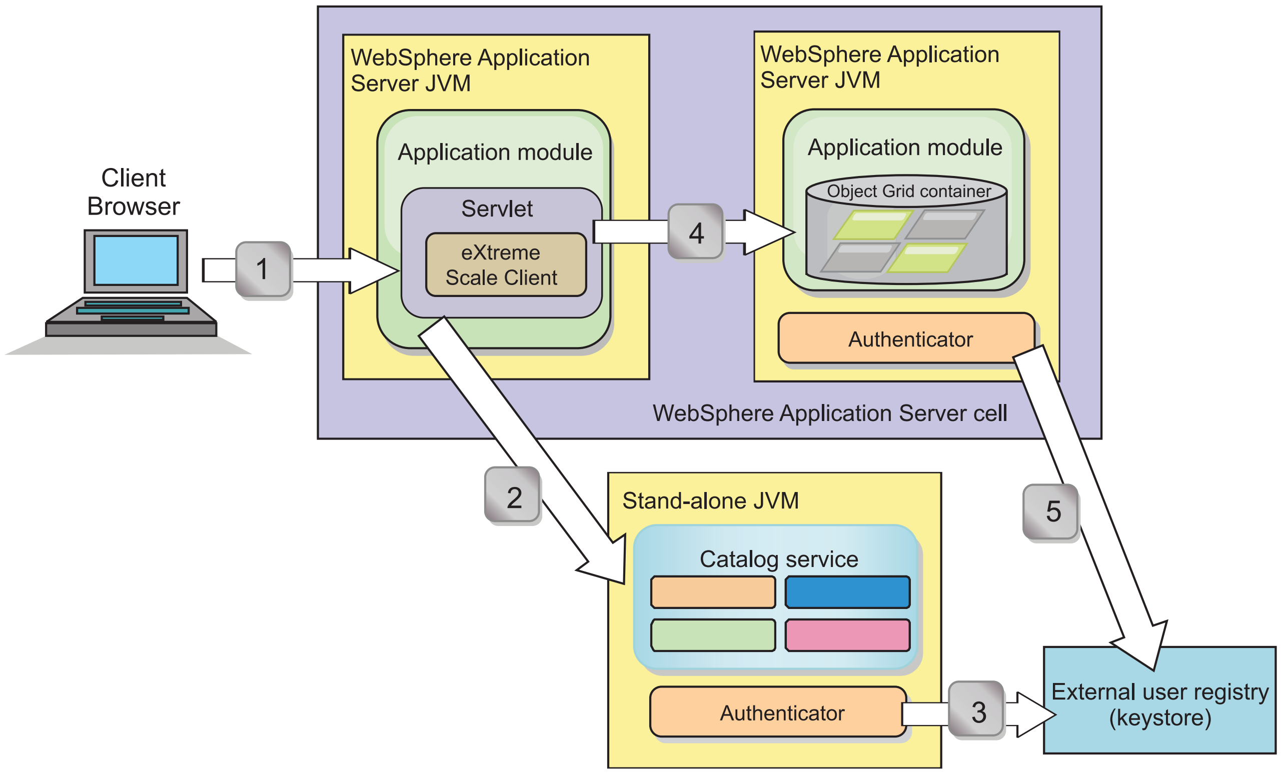


Figure 23. Authentication flow

The previous diagram shows two application servers. The first application server hosts the web application, which is also a WebSphere eXtreme Scale client. The second application server hosts a container server. The catalog server is running in a stand-alone Java virtual machine (JVM) instead of WebSphere Application Server.

The arrows marked with numbers in the diagram indicate the authentication flow:

1. An enterprise application user accesses the web browser, and logs in to the first application server with a user name and password. The first application server sends the client user name and password to the security infrastructure to authenticate to the user registry. This user registry is a keystore. As a result, the security information is stored on the WebSphere Application Server thread.
2. The JavaServer Pages (JSP) file acts as a WebSphere eXtreme Scale client to retrieve the security information from the client property file. The JSP application that is acting as the WebSphere eXtreme Scale client sends the WebSphere eXtreme Scale client security credential along with the request to the catalog server. Sending the security credential with the request is considered a *runAs* model. In a runAs model, the web browser client runs as a WebSphere eXtreme Scale client to access the data stored in the container server. The client uses a Java virtual machine (JVM)-wide client credential to connect to the WebSphere eXtreme Scale servers. Using the runAs model is like connecting to a database with a data source level user ID and password.
3. The catalog server receives the WebSphere eXtreme Scale client credential, which includes the WebSphere Application Server security tokens. Then, the catalog server calls the authenticator plug-in to authenticate the client credential. The authenticator connects to the external user registry and sends the client credential to the user registry for authentication.
4. The client sends the user ID and password to the container server that is hosted in the application server.

5. The container service, hosted in the application server, receives the WebSphere eXtreme Scale client credential, which is the user id and password pair. Then, the container server calls the authenticator plug-in to authenticate the client credential. The authenticator connects to the keystore user registry and sends the client credential to the user registry for authentication

Learning objectives

With the lessons in this module, you learn how to:

- Configure WebSphere eXtreme Scale client security.
- Configure WebSphere eXtreme Scale catalog server security.
- Configure WebSphere eXtreme Scale container server security.
- Install and run the sample application.

Time required

This module takes approximately 60 minutes.

Lesson 2.1: Configure WebSphere eXtreme Scale client security

You configure the client properties with a properties file. The client properties file indicates the CredentialGenerator implementation class to use.

Client properties file contents:

The tutorial uses WebSphere Application Server security tokens for the client credential. The *samples_home*/`security_extauth` directory contains the `client3.props` file.

The `client3.props` file includes the following settings:

securityEnabled
Enables WebSphere eXtreme Scale client security. The value is set to `true` to indicate that the client must send available security information to the server.

credentialAuthentication
Specifies the client credential authentication support. The value is set to `Supported` to indicate that the client supports credential authentication.

credentialGeneratorClass
Specifies the name of the class that implements the com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator interface. The value is set to the com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator class so that the client retrieves the security information from the UserPasswordCredentialGenerator class.

credentialGeneratorProps
Specifies the user name and password: `manager manager1`. The user name is `manager`, and the password is `manager1`. You can also use the **`FilePasswordEncoder.bat|sh`** command to encode this property using an exclusive or (xor) algorithm.

Setting the client properties file using Java virtual machine (JVM) properties:
In the administrative console, complete the following steps to both the `s1` and `s2` servers in the `appCluster` cluster. If you are using a different topology, complete

the following steps to all of the application servers to which the `EmployeeManagement` application is deployed.

1. Click **Servers** > **WebSphere application servers** > *server_name* > **Java and Process Management** > **Process definition** > **Java Virtual Machine**.
2. Create the following generic JVM property to set the location of the client properties file:

   `-Dobjectgrid.client.props=`*`samples_home`*`/security_extauth/client3.props`

   When you connect to a secure data grid, you must configure the client application to provide a valid client security configuration. You can configure the client security configuration through the client application, or you can defined the configuration in a client properties file that has the same value of the JVM property, **`objectgrid.client.props`**. When you use the **`objectgrid.client.props`** property, the ObjectGridManager obtains the client security configuration from the client properties file and uses this information to connect to the data grid.
3. Click **OK** and save your changes.

Lesson checkpoint:

You edited the client properties file and configured the servers in the `appCluster` cluster to use the client properties file. This properties file indicates the CredentialGenerator implementation class to use.

Lesson 2.2: Configure catalog server security

A catalog server contains two different levels of security information: The first level contains the security properties that are common to all the WebSphere eXtreme Scale servers, including the catalog service and container servers. The second level contains the security properties that are specific to the catalog server.

The security properties that are common to the catalog servers and container servers are configured in the security XML descriptor file. An example of common properties is the authenticator configuration, which represents the user registry and authentication mechanism. See Security descriptor XML file for more information about the security properties.

To configure the security XML descriptor file in a Java SE environment, use a **`-clusterSecurityFile`** option when you run the **`startOgServer`** command. Specify a value in a file format, such as *samples_home*`/security_extauth/security3.xml`.

`security3.xml` file:

In this tutorial, the `security3.xml` file is in the *samples_home*`/security_extauth` directory. The content of the `security3.xml` file with the comments removed follows:

```
<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>
```

The following properties are defined in the `security3.xml` file:

securityEnabled
The securityEnabled property is set to `true`, which indicates to the catalog server that the WebSphere eXtreme Scale global security is enabled.

authenticator
The authenticator is configured as the com.ibm.websphere.objectgrid.security.plugins.builtins. KeyStoreLoginAuthenticator class. With this built-in implementation of the Authenticator plug-in, the user ID and password is passed to verify that it is configured in the keystore file. The KeyStoreLoginAuthenticator class uses a `KeyStoreLogin` login module alias, so a Java Authentication and Authorization Service (JAAS) login configuration is required.

`catServer3.props` file:

The server property file stores the server-specific properties, which include the server-specific security properties. See Server properties file for more information. You can use **`-serverProps`** option to specify the catalog server property when you run the **`startOgServer`** command. For this tutorial, a `catServer3.props` file is in the c directory. The content of the `catServer3.props` file with the comments removed follows:

```
securityEnabled=true
credentialAuthentication=Required
transportType=TCP/IP
secureTokenManagerType=none
authenticationSecret=ObjectGridDefaultSecret
```

securityEnabled
The securityEnabled property is set to `true` to indicate that this catalog server is a secure server.

credentialAuthentication
The credentialAuthentication property is set to `Required`, so any client that is connecting to the server is required to provide a credential. Iin the client property file, the credentialAuthentication value is set to `Supported`, so the server receives the credentials that are sent by the client.

secureTokenManagerType
The secureTokenManagerType is set to `none` to indicate that the authentication secret is not encrypted when joining the existing servers.

authenticationSecret
The authenticationSecret property is set to `ObjectGridDefaultSecret`. This secret string is used to join the eXtreme Scale server cluster. When a server joins the data grid, it is challenged to present the secret string. If the secret string of the joining server matches the string in the catalog server, the joining server is accepted. If the string does not match, the join request is rejected.

transportType
The transportType property is set to `TCP/IP` initially. Later in the tutorial, transport security is enabled.

`xsjaas3.config` file:

Because the KeyStoreLoginAuthenticator implementation uses a login module, you must configure the login model with a JAAS authentication login configuration file. The contents of the `xsjaas3.config` file follows:

```
KeyStoreLogin{
com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule required
     keyStoreFile="samples_home/security_extauth/sampleKS3.jks" debug = true;
};
```

If you used a location for *samples_home* other than `/wxs_samples/`, you need to update the location of the keyStoreFile. This login configuration indicates that the com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule module is used as the login module. The keystore file is set to the `sampleKS3.jks` file.

Important: Windows If you are using Windows, the directory path does not support backslashes. If you have used backslashes, you must escape any backslash (\ ) characters in the path. For example, if you want to use the path C:\opt\ibm, enter C:\\opt\\ibm in the properties file. Windows directories with spaces are not supported.

The `sampleKS3.jks` sample keystore file stores two user IDs and the passwords: manager/manager1 and cashier/cashier1.

You can use the following **`keytool`** commands to create this keystore:
- `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`

Start the catalog server with security enabled:

To start the catalog server, issue the **`startOgServer`** command with the **`-clusterSecurityFile`** and **`-serverProps`** parameters to pass in the security properties.

Use a stand-alone installation of WebSphere eXtreme Scale to run the catalog server. When using the stand-alone installation image, you must use the IBM SDK. You can use the SDK that is included with WebSphere Application Server by setting the *JAVA_HOME* variable to point to the IBM SDK. For example, `set JAVA_HOME=`*`was_root`*`/IBM/WebSphere/AppServer/java/`

1. Go to the `bin` directory.

   `cd` *`wxs_home`*`/bin`

2. Run the **`startOgServer`** command.

   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"
   ```

After you run the **`startOgServer`** command, a secure server starts with listener port 16809, client port 16601, peer port 16602, and JMX port 16099. If a port conflict exists, change the port number to an unused port number.

Stop a catalog server that has security enabled:

You can use the **stopOgServer** command to stop the catalog server.

1. Go to the `bin` directory.

   ```
   cd wxs_home/bin
   ```

2. Run the **stopOgServer** command. 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
   ```

Lesson checkpoint:

You configured catalog server security by associating the `security3.xml`, `catServer3.props`, `xsjaas3.config` files with the catalog service.

Lesson 2.3: Configure container server security

When a container server connects to the catalog service, the container server gets all the security configurations that are configured in the Object Grid Security XML file. The ObjectGrid Security XML file defines authenticator configuration, the login session timeout value, and other configuration information. A container server also has its own server-specific security properties in the server property file.

Configure the server property file with the -Dobjectgrid.server.props Java virtual machine (JVM) property. The file name specified for this property is an absolute file path, such as *samples_home*/`security_extauth/server3.props`.

In this tutorial, the container servers are hosted in the `xs1` and `xs2` servers in the `xsCluster` cluster.

`server3.props` file:

The `server3.props` file is in the *samples_home*/`security_extauth/` directory. The content of the `server3.props` file follows:

```
securityEnabled=true
credentialAuthentication=Required
secureTokenManagerType=none
authenticationSecret=ObjectGridDefaultSecret
```

securityEnabled
: The securityEnabled property is set to `true` to indicate that this container server is a secure server.

credentialAuthentication
: The credentialAuthentication property is set to `Required`, so any client that is connecting to the server is required to provide a credential. In the client property file, the credentialAuthentication property is set to `Supported`, so the server receives the credential that is sent by the client.

secureTokenManagerType
: The secureTokenManagerType is set to `none` to indicate that the authentication secret is not encrypted when joining the existing servers.

authenticationSecret
: The authenticationSecret property is set to `ObjectGridDefaultSecret`. This secret string is used to join the eXtreme Scale server cluster. When a server

joins the data grid, it is challenged to present the secret string. If the secret string of the joining server matches the string in the catalog server, the joining server is accepted. If the string does not match, the join request is rejected.

Setting the server properties file with JVM properties:

Set the server properties file on the xs1 and xs2 servers. If you are not using the topology for this tutorial, set the server properties file on all of the application servers that you are using to host container servers.

1. Open the Java virtual machine page for the server. **Servers** > **WebSphere application servers** > *server_name* > **Java and Process Management** > **Process definition** > **Java Virtual Machine**.
2. Add the generic JVM argument:
   `-Dobjectgrid.server.props=`*`samples_home`*`/security_extauth/server3.props`
3. Click **OK** and save your changes.

Adding the custom login module:

The container server uses the same KeyStoreAuthenticator implementation as the catalog server. The KeyStoreAuthenticator implementation uses a **`KeyStoreLogin`** login module alias, so you must add a custom login module to the application login model entries.

1. In the WebSphere Application Server administrative console, click **Security** > **Global security** > **Java Authentication and Authorization Service**.
2. Click **Application logins**.
3. Click **New**, add an alias `KeyStoreLogin`. Click **Apply**.
4. Under **JAAS login modules**, click **New**.
5. Enter `com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule` as the module class name, and choose **SUFFICIENT** as the authentication strategy. Click **Apply**.
6. Add the `keyStoreFile` custom property with value *`samples_home`*`/security_extauth/sampleKS.jks`.

   Important: Windows If you are using Windows, the directory path does not support backslashes. If you have used backslashes, you must escape any backslash (\ ) characters in the path. For example, if you want to use the path C:\opt\ibm, enter C:\\opt\\ibm in the properties file. Windows directories with spaces are not supported.
7. Optional: Add the `debug` custom property with value `true`.
8. Save the configuration.

Lesson checkpoint:

Now the WebSphere eXtreme Scale server authentication is secured. By configuring this security, all the applications that try to connect to the WebSphere eXtreme Scale servers are required to provide a credential. In this tutorial, the KeyStoreLoginAuthenticator is the authenticator. As a result, the client is required to provide a user name and password.

Lesson 2.4: Install and run the sample

After authentication is configured, you can install and run the sample application.

Creating a shared library for the `EmployeeData.jar` file:

1. In the WebSphere Application Server administrative console, open the **Shared Libraries** page. Click **Environment** > **Shared libraries**.
2. Choose the **cell** scope.
3. Create the shared library. Click **New**. Enter `EmployeeManagementLIB` as the **Name**. Enter the path to the `EmployeeData.jar` in the classpath, for example, *samples_home*/`WASSecurity/EmployeeData.jar`.
4. Click **Apply**.

Installing the sample:

1. Install the `EmployeeManagement_extauth.ear` file under the *samples_home*/`security_extauth` directory.

   Important: The `EmployeeManagement_extauth.ear` file is different from the *samples_home*/`WASSecurity/EmployeeManagement.ear` file. The manner in which the ObjectGrid session is retrieved has been updated to use the credential that is cached in the client property file in the `EmployeeManagement_extauth.ear` application. See the comments in the com.ibm.websphere.sample.xs.DataAccessor class in the *samples_home*/`WASSecurity/EmployeeManagementWeb` project to see the code that was updated for this change.
   a. To begin the installation, click **Applications** > **New application** > **New Enterprise Application**. Choose the detailed path for installing the application.
   b. On the **Map modules to servers** step, specify the `appCluster` cluster to install the EmployeeManagementWeb module.
   c. On the **Map shared libraries** step, select the `EmployeeManagementWeb` module.
   d. Click **Reference shared libraries**. Select the `EmployeeManagementLIB` library.
   e. Map the `webUser` role to **All Authenticated in Application's Realm**.
   f. Click **OK**.

   The clients run in the `s1` and `s2` servers in this cluster.
2. Install the sample `XSDeployment.ear` file that is in the *samples_home*/`WASSecurity` directory.
   a. To begin the installation, click **Applications** > **New application** > **New Enterprise Application**. Choose the detailed path for installing the application.
   b. On the **Map modules to servers** step, specify the `xsCluster` cluster to install the `XSDeploymentWeb` web module.
   c. On the **Map shared libraries** step, select the `XSDeploymentWeb` module.
   d. Click **Reference shared libraries**. Select the `EmployeeManagementLIB` library.
   e. Click **OK**.

   The `xs1` and `xs2` servers in this cluster host the container servers.
3. Verify that the catalog server is started. For more information about starting a catalog server for this tutorial, see “Start the catalog server with security enabled” on page 140.
4. Restart the xsCluster cluster. When the xsCluster starts, the `XSDeployment` application starts, and a container server is started on the xs1 and xs2 servers

respectively. If you look at the `SystemOut.log` file of the xs1 and xs2 servers, the following message that indicates the server properties file is loaded is displayed:

```
CWOBJ0913I: Server property files have been loaded:
samples_home/security_extauth/server3.props.
```

5. Restart the appClusters cluster. When the cluster appCluster starts, the EmployeeManagement application also starts. If you look at the `SystemOut.log` file of the s1 and s2 servers, you can see the following message that indicates that the client properties file is loaded.

   ```
   CWOBJ0924I: The client property file {0} has been loaded.
   ```

   If you are using WebSphere eXtreme Scale Version 7.0, the English-only CWOBJ9000I message displays to indicate that the client property file has been loaded. If you do not see the expected message, verify that you configured the -Dobjectgrid.server.props or -Dobjectgrid.client.props property in the JVM argument. If you do have the properties configured, make sure the dash (-) is a UTF character.

Running the sample application:

1. Run the `management.jsp` file. In a web browser, access `http://<your_servername>:<port>/EmployeeManagementWeb/management.jsp`. For example, you might use the following URL: `http://localhost:9080/EmployeeManagementWeb/management.jsp`.
2. Provide authentication to the application. Enter the credentials of the user that you mapped to the webUser role. By default, this user role is mapped to all authenticated users. Type any valid user name and password, such as the administrative user name and password. A page to display, add, update, and delete employees displays.
3. Display employees. Click **Display an Employee**. Enter `emp1@acme.com` as the email address, and click **Submit**. A message displays that the employee cannot be found.
4. Add an employee. click **Add an Employee**. Enter `emp1@acme.com` as the email address, enter `Joe` as the given name, and `Doe` as the surname. Click **Submit**. A message displays that an employee with the `emp1@acme.com` address has been added.
5. Display the new employee. Click **Display an Employee**. Enter `emp1@acme.com` as the email address with empty fields for the first and surnames, and click **Submit**. A message displays that the employee has been found, and the correct names are displayed in the given name and surname fields.
6. Delete the employee. Click **Delete an employee**. Enter `emp1@acme.com` and click **Submit**. A message is displayed that the employee has been deleted.

Because the catalog server transport type is set to `TCP/IP`, verify that the server s1 and s2 outbound transport setting is not set to `SSL-Required`. Otherwise, an exception occurs. If you look at the system out file of the catalog server, `logs/cs1/SystemOut.log` file, the following debug output to indicates the key store authentication:

```
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:
...
```

Lesson checkpoint:

You installed and ran the sample application.

Module 3: Configure transport security

Configure transport security to secure data transfer between the clients and servers in the configuration.

In the previous module in the tutorial, you enabled WebSphere eXtreme Scale authentication. With authentication, any application that tries to connect to the WebSphere eXtreme Scale server is required to provide a credential. Therefore, no unauthenticated client can connect to the WebSphere eXtreme Scale server. The clients must be an authenticated application that is running in a WebSphere Application Server cell.

With the configuration up to this module, the data transfer between the clients in the appCluster cluster and servers in the xsCluster cluster is not encrypted. This configuration might be acceptable if your WebSphere Application Server clusters are installed on servers behind a firewall. However, in some scenarios, non-encrypted traffic is not accepted for some reasons even though the topology is protected by firewall. For example, a government policy might enforce encrypted traffic. WebSphere eXtreme Scale supports Transport Layer Security/Secure Sockets Layer (TLS/SSL) for secure communication between ObjectGrid endpoints, which include client servers, container servers, and catalog servers.

In this sample deployment, the eXtreme Scale clients and container servers are all running in the WebSphere Application Server environment. Client or server properties are not necessary to configure the SSL settings because the eXtreme Scale transport security is managed by the Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) transport settings. WebSphere eXtreme Scale servers use the same Object Request Broker (ORB) instance as the application servers in which they run. Specify all the SSL settings for client and container servers in the WebSphere Application Server configuration using these CSIv2 transport settings. You must configure the SSL properties in the server properties file for the catalog server.

Learning objectives

After completing the lessons in this module, you know how to:
- Configure CSIv2 inbound and outbound transport.
- Add SSL properties to the catalog server properties file.
- Check the ORB properties file.
- Run the sample.

Time required

This module takes approximately 60 minutes.

Prerequisites

This step of the tutorial builds upon the previous modules. Complete the previous modules in this tutorial before you configure transport security.

Lesson 3.1: Configure CSIv2 inbound and outbound transport

To configure Transport Layer Security/Secure Sockets Layer (TLS/SSL) for the server transport, set the Common Secure Interoperability Protocol Version 2 (CSIv2) inbound transport and CSIv2 outbound transport to `SSL-Required` for all the WebSphere Application Server servers that host clients, catalog servers, and container servers.

In the tutorial example topology, you must set these properties for the, s1, s2, xs1, and xs2 application servers. The following steps configure the inbound and outbound transports for all the servers in the configuration.

Set the inbound and outbound transports in the administrative console. Make sure that administrative security is enabled.

- **WebSphere Application Server Version 6.1**: Click **Security** > **Secure Administration** > **Application..** > **RMI/IIOP Security** and change the transport type to **SSL-Required**.
- **WebSphere Application Server Version 7.0**: Click **Security** > **Global Security** > **RMI/IIOP Security** > **CSIv2 inbound communications**. Change the transport type under the CSIv2 Transport Layer to **SSL-Required**. Repeat this step to configure CSIv2 outbound communications.

You can use centrally managed endpoint security settings, or you can configure SSL repositories. See Common Secure Interoperability Version 2 transport inbound settings for more information.

Lesson 3.2: Add SSL properties to the catalog server properties file

The catalog server is running outside of WebSphere Application Server, so you must configure the SSL properties in the server properties file.

The other reason to configure the SSL properties in the server properties file is because the catalog server has its own proprietary transport paths that cannot be managed by the WebSphere Application Server Common Secure Interoperability Protocol Version 2 (CSIV2) transport settings. Therefore, you must configure the Secure Sockets Layer (SSL) properties in the server properties file for the catalog server.

SSL properties in the `catServer3.props` file:

```
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
```

The `catServer3.props` file is using the default WebSphere Application Server node level keystore and truststore. If you are deploying a more complex deployment environment, you must choose the correct keystore and truststore. In some cases, you must create a keystore and truststore and import the keys from keystores from the other servers. Notice that the `WebAS` string is the default password of the WebSphere Application Server keystore and truststore. See Default self-signed certificate configuration for more details.

These entries are already included in the *samples_home*/`security_extauth/catServer3.props` file as comments. You can uncomment the entries and make the appropriate updates for your installation to the *was_root*, <*deployment_manager_name*>, <*cell_name*>, and <*node_name*> variables.

After configuring the SSL properties, change the transportType property value from `TCP/IP` to `SSL-Required`.

SSL properties in the `client3.props` file:

You must also configure the SSL properties in the `client3.props` file because this file is used when you stop the catalog server that is running outside of WebSphere Application Server.

These properties have no effect on the client servers that are running in WebSphere Application Server because they are using the WebSphere Application Server Common Security Interoperability Protocol Version 2 (CSIV2) transport settings. However, when you stop the catalog server you must provide a client properties file on the **`stopOgServer`** command. Set the following properties in the <SAMPLES_HOME>/security_extauth/client3.props file to match the values specified above in the catServer3.props file:

```
#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
```

As with the `catServer3.props` file, you can use the comments that are already provided in the *samples_home*/`security_extauth/client3.props` file with appropriate updates to *was_root*, <*deployment_manager_name*>, <*cell_name*>, and <*node_name*> variables to match your environment.

Lesson checkpoint:

You configured the SSL properties for the catalog server.

Lesson 3.3: Run the sample

Restart all the servers and run the sample application again. You should be able to run through the steps without any problems.

See "Lesson 2.4: Install and run the sample" on page 142 for more information about running and installing the sample application.

Module 4: Use Java Authentication and Authorization Service (JAAS) authorization in WebSphere Application Server

Now that you have configured authentication for clients, you can further configure authorization to give different users varying permissions. For example, an "operator" user might only be able to view data, while a "manager" user can perform all operations.

After authenticating a client, as in the previous module in this tutorial, you can give security privileges through eXtreme Scale authorization mechanisms. The previous module of this tutorial demonstrated how to enable authentication for a data grid using integration with WebSphere Application Server. As a result, no unauthenticated client can connect to the eXtreme Scale servers or submit requests to your system. However, every authenticated client has the same permission or privileges to the server, such as reading, writing, or deleting data that is stored in the ObjectGrid maps. Clients can also issue any type of query.

This part of the tutorial demonstrates how to use eXtreme Scale authorization to give authenticated users varying privileges. WebSphere eXtreme Scale uses a permission-based authorization mechanism. You can assign different permission categories that are represented by different permission classes. This module features the MapPermission class. For a list of all possible permissions, see Client authorization programming.

In WebSphere eXtreme Scale, the `com.ibm.websphere.objectgrid.security.MapPermission` class represents permissions to the eXtreme Scale resources, specifically the methods of the ObjectMap or JavaMap interfaces. WebSphere eXtreme Scale defines the following permission strings to access the methods of ObjectMap and JavaMap:

- **read**: Grants permission to read the data from the map.
- **write**: Grants permission to update the data in the map.
- **insert**: Grants permission to insert the data into the map.
- **remove**: Grants permission to remove the data from the map.
- **invalidate**: Grants permission to invalidate the data from the map.
- **all**: Grants all permissions to read, write, insert, remote, and invalidate.

The authorization occurs when an eXtreme Scale client uses a data access API, such as the ObjectMap ,JavaMap, or EntityManager APIs. The run time checks corresponding map permissions when the method is called. If the required permissions are not granted to the client, an AccessControlException exception results. This tutorial demonstrates how to use Java Authentication and Authorization Service (JAAS) authorization to grant authorization map access for different users.

Learning objectives

After completing the lessons in this module, you know how to:

- Enable authorization for WebSphere eXtreme Scale.
- Enable user-based authorization.

Time required

This module takes approximately 60 minutes.

Lesson 4.1: Enable WebSphere eXtreme Scale authorization

To enable authorization in WebSphere eXtreme Scale, you must enable security on a specific ObjectGrid.

To enable authorization on the ObjectGrid, you must set the **securityEnabled** attribute to true for that particular ObjectGrid in the XML file. For this tutorial, you can either use the XSDeployment_sec.ear file from the *samples_home*/WASSecurity directory, which has already has security set in the objectGrid.xml file, or you can edit the existing objectGrid.xml file to enable security. This lesson demonstrates how to edit the file to enable security.

1. Optional: Extract the files in the XSDeployment.ear file, and then unzip the XSDeploymentWeb.war file.
2. Optional: Open the objectGrid.xml file and set the **securityEnabled** attribute to true on the ObjectGrid level. See an example of this attribute in the following example:

   ```
   <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>
   ```

   If you have multiple ObjectGrids defined, then you must set this attribute on each grid.
3. Optional: Repackage the XSDeploymentWeb.war and XSDeployment.ear files to include your changes.
4. Required: Uninstall the XSDeployment.ear file and then install the updated XSDeployment.ear. You can either use the file you modified in the previous steps, or you can install the XSDeployment_sec.ear file that is provided in the *samples_home*/WASSecurity directory. See "Lesson 2.4: Install and run the sample" on page 142 for more information about installing the application.
5. Restart all of the application servers to enable WebSphere eXtreme Scale authorization.

Lesson checkpoint:

You enabled security on the ObjectGrid, which also enables authorization on the data grid.

Lesson 4.2: Enable user-based authorization

In the authentication module of this tutorial, you created two users: operator and manager. You can assign varying permissions to these users with Java Authentication and Authorization Service (JAAS) authorization.

Defining the Java Authentication and Authorization Service (JAAS) authorization policy using user principals:

You can assign permissions to the users that you previously created. Assign the operator user only read permissions to all maps. Assign the manager user all permissions. Use the JAAS authorization policy file to grant permissions to principals.

Edit the JAAS authorization file. The `xsAuth3.policy` file is in the *samples_home*/`security_extauth` directory.

```
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";
};
```

In this file, the `http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction` codebase is a specially reserved URL for ObjectGrid. All ObjectGrid permissions that are granted to principals should use this special code base. The following permissions are assigned in this file:

- The first grant statement grants `read` map permission to the `"CN=operator,O=acme,OU=OGSample"` principal. The `"CN=operator,O=acme,OU=OGSample"` user has only map read permission to the Map1 map the Grid ObjectGrid instance.
- The second grant statement grants all map permission to the `"CN=manager,O=acme,OU=OGSample"` principal. The `"CN=manager,O=acme,OU=OGSample"` user has all permissions to the Map1 map in the Grid ObjectGrid instance.

Setting the JAAS authorization policy file using JVM properties:
Use the following steps to set JVM properties for the xs1 and xs2 servers, which are in the xsCluster cluster. If you are using a topology that is different from the sample topology that is used in this tutorial, set the file on all of your container servers.

1. In the administrative console, click **Servers** > **Application servers** > *server_name* > **Java and process management** > **Process definition** > **Java virtual machine.**
2. Add the following generic JVM arguments:

   `-Djava.security.policy=`*samples_home*`/security_extauth/xsAuth3.policy`
3. Click **OK** and save your changes.

Running the sample application to test authorization:
You can use the sample application to test the authorization settings. The manager user continues to have all permissions in the Map1 map, including displaying and adding employees. The operator user should only be able to view employees because that user was assigned read permission only.

1. Restart all of the application servers that are running container servers. For this tutorial, restart the `xs1` and `xs2` servers.
2. Open the `EmployeeManagementWeb` application. In a web browser, open `http://<host>:<port>/EmployeeManagermentWeb/management.jsp`.
3. Log in to the application using any valid user name and password.
4. Attempt to display an employee. Click **Display an Employee** and search for the `authemp1@acme.com` email address. A message displays that the user cannot be found.
5. Add an employee. Click **Add an Employee**. Add the email `authemp1@acme.com`, the given name `Joe`, and the surname `Doe`. Click **Submit**. A message displays that the employee has been added.

6. Edit the *samples_home*/`security_extauth/client3.props` file. Change the value of credentialGeneratorProps property from `manager manager1` to `operator operator1`. After you edit the file, the servlet uses user name "operator" and password "operator1" to authenticate to the WebSphere eXtreme Scale servers.
7. Restart the appCluster cluster to pick up the changes in the *samples_home*/`security_extauth/client3.props` file.
8. Attempt to display an employee. Click **Display an Employee** and search for the `authemp1@acme.com` email address. The employee is displayed.
9. Add an employee. Click **Add an Employee**. Add the email `authemp2@acme.com`, the given name `Joe`, and the surname `Doe`. Click **Submit**. The following message displays:

   ```
   An exception occurs when Add the employee. See below for detailed exception messages.
   ```

   The detailed exception text follows:

   ```
   java.security.AccessControlException: Access denied
   (com.ibm.websphere.objectgrid.security.MapPermission Grid.Map1 insert)
   ```

   This message displays because the `operator` user does not have permission to insert data into the Map1 map.

If you are running with a version of WebSphere Application Server that is earlier than Version 7.0.0.11, you might see a java.lang.StackOverflowError error on the container server. This error is caused by a problem with the IBM Developer Kit. The problem is fixed in the IBM Developer Kit that is shipped with WebSphere Application Server Version 7.0.0.11 and later.

Lesson checkpoint:

In this lesson, you configured authorization by assigning permissions to specific users.

Module 5: Use the `xscmd` utility to monitor data grids and maps

You can use the **`xscmd`** utility to show the primary data grids and map sizes of the `Grid` data grid. The **`xscmd`** tool uses the MBean to query all of the data grid artifacts, such as primary shards, replica shards, container servers, map sizes, and other data.

In this tutorial, the catalog server is running as a stand-alone Java SE server. The container servers are running in WebSphere Application Server application servers.

For the catalog server, a MBean server is created in the stand-alone Java virtual machine (JVM). When you use the **`xscmd`** tool on the catalog server, WebSphere eXtreme Scale security is used.

For the container servers, the WebSphere eXtreme Scale run time registers the Managed Beans (MBean) with the MBean server that is created by the WebSphere Application Server run time. The security that is used by the **`xscmd`** tool is provided by the WebSphere Application Server MBean security.
1. Using a command-line tool, open the *`DMGR_PROFILE`*`/bin` directory.
2. Run the **`xscmd`** tool. Use the **`-c showPlacement -st P`** parameters as in the following examples:

   Linux UNIX

   ```
   xscmd.sh -c showPlacement -cep localhost:16099 -g Grid -ms mapSet -sf P
   -user manager -pwd manager1
   ```

Windows

```
xscmd.bat -c showPlacement -cep localhost:16099 -g Grid -m mapSet -sf P
-user manager -pwd manager1
```

The user name and password are passed to the catalog server for authentication.

3. View the command results.

```
*** 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. Run the **xscmd** tool. Use the **-c showMapSizes** parameter as in the following examples:

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
```

The user name and password are passed to the catalog server for authentication. After you run the command, you are prompted for the WebSphere Application Server user ID and password to authenticate to WebSphere Application Server. You must provide this login information because the **-c showMapSizes** option gets the map size from each container server, which requires the WebSphere Application Server security.

5. Optional: You can change the `PROFILE/properties/sas.client.props` file to run the command without the user ID and password being required. Change the com.ibm.CORBA.loginSource property from `prompt` to `properties` and then provide the user ID and password. An example of the properties in the `PROFILE/properties/sas.client.props` file follows:

```
com.ibm.CORBA.loginSource=properties
# RMI/IIOP user identity
com.ibm.CORBA.loginUserid=Admin
com.ibm.CORBA.loginPassword=xxxxxx
```

6. Optional: If you are using the **xscmd** command on a WebSphere eXtreme Scale stand-alone installation, then you must add the following options:
   - If you are using WebSphere eXtreme Scale security:

     ```
     -user
     -pwd
     ```

   - If you are using WebSphere eXtreme Scale security with custom credential generation:

     ```
     -user
     -pwd
     -cgc
     -cgp
     ```

   - If SSL is enabled:

     ```
     -tt
     -cxpv
     -prot
     -ks
     -ksp
     ```

```
-kst
-ts
-tsp
-tst
```

If WebSphere eXtreme Scale security and SSL are both enabled, then both set of parameters are required.

Related tasks:

"Monitoring with the **xscmd** utility" on page 535
The **xscmd** utility replaces the **xsadmin** sample utility as a fully supported monitoring and administration tool. With the **xscmd** utility, you can display textual information about your WebSphere eXtreme Scale topology.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Lesson checkpoint

You used the **xscmd** tool to monitor data grids and maps in your configuration.

Tutorial: Running eXtreme Scale bundles in the OSGi framework

The OSGi sample builds on the Google Protocol Buffers serializer samples. When you complete this set of lessons, you will have run the serializer sample plug-ins in the OSGi framework.

Learning objectives

This sample demonstrates the OSGi bundles. The serializer plug-in is incidental and is not required. The OSGi sample is available on the WebSphere eXtreme Scale samples gallery. You must download the sample, and extract it into the `*wxs_home*/samples` directory. The root directory for the OSGi sample is `wxs_home/samples/OSGiProto`.

The command examples in this tutorial assume that you are running on the UNIX operating system. You must adjust the command example to run on a Windows operating system.

After completing the lessons in this tutorial, you will understand the OSGi sample concepts and know how to complete the following objectives:

- Install the WebSphere eXtreme Scale server bundle into the OSGi container to start the eXtreme Scale server.
- Set up your eXtreme Scale development environment to run the sample client.
- Use the xscmd command to query the service ranking of the sample bundle, upgrade it to a new service ranking, and verify the new service ranking.

Time required

This module takes approximately 60 minutes to complete.

Prerequisites

In addition to downloading and extracting the serializer samples, this tutorial also has the following prerequisites:

- Install and extract the eXtreme Scale product

- Set up the Eclipse Equinox Environment

Introduction: Starting and configuring the eXtreme Scale server and container to run plug-ins in the OSGi framework

In this tutorial you start an eXtreme Scale server in the OSGi framework, start an eXtreme Scale container, and wire the sample plug-ins with eXtreme Scale runtime environment.

Learning objectives

After completing the lessons in this tutorial you will understand the OSGi sample concepts and know how to complete the following objectives:
- Install the WebSphere eXtreme Scale server bundle into the OSGi container to start the eXtreme Scale server.
- Set up your eXtreme Scale development environment to run the sample client.
- Use the xscmd command to query the service ranking of the sample bundle, upgrade it to a new service ranking, and verify the new service ranking.

Time required

This tutorial takes approximately 60 minutes to finish. If you explore other concepts related to this tutorial, it might take longer to complete.

Skill level

Intermediate.

Audience

Developers and administrators who want to build, install, and run eXtreme Scale bundles into the OSGi framework.

System requirements

- Luminis OSGi Configuration Admin command line client, version 0.2.5
- Apache Felix File Install, version 3.0.2
- When using Eclipse Gemini as the Blueprint container provider, the following are required:
  - Eclipse Gemini Blueprint, version 1.0.0
  - Spring Framework, version 3.0.5
  - SpringSource AOP Alliance API, version 1.0.0
  - SpringSource Apache Commons Logging, version 1.1.1
- When using Apache Aries as the Blueprint Container provider, you must have the following requirements:
  - Apache Aries, latest snapshot
  - ASM library
  - PAX logging

Prerequisites

To complete this tutorial, you must download the sample, and extracted it into the `wxs_home/samples` directory. The root directory for the OSGi sample is `wxs_home/samples/OSGiProto`.

Expected results

When you complete this tutorial, you will have installed the sample bundles and run an eXtreme Scale client to insert data into the grid. You can also expect to query and update those sample bundles using the dynamic capabilities that the OSGi container provides.

Related concepts:

OSGi framework overview
OSGi defines a dynamic module system for Java. The OSGi service platform has a layered architecture, and is designed to run on various standard Java profiles. You can start WebSphere eXtreme Scale servers and clients in an OSGi container.

Related tasks:

"Installing the Eclipse Equinox OSGi framework with Eclipse Gemini for clients and servers" on page 223
If you want to deploy WebSphere eXtreme Scale in the OSGi framework, then you must set up the Eclipse Equinox Environment.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Module 1: Preparing to install and configure eXtreme Scale server bundles

Complete this module to explore OSGi sample bundles and examine configuration files that you use to configure the eXtreme Scale server.

Learning objectives

After completing the lessons in this module, you will understand the concepts and know how to complete the following objectives:

- Locate and explore the bundles that are included in the OSGi sample.
- Examine configuration files that are used to configure the eXtreme Scale grid and server.

Lesson 1.1: Understand the OSGi sample bundles

Complete this lesson to locate and explore the bundles that are provided in the OSGi sample.

OSGi sample bundles:

Other than the bundles that are configured in the `config.ini` file, which is shown in the topic about setting up the Eclipse Equinox environment, the following additional bundles are used in the OSGi sample:

`objectgrid.jar`
: The WebSphere eXtreme Scale server runtime bundle. This bundle is located in the *wxs_home*`/lib directory`.

`com.google.protobuf_2.4.0a.jar`
: The Google Protocol Buffers, version 2.4.0a bundle. This bundle is located in the *wxs_sample_osgi_root*`/lib directory`.

`ProtoBufSamplePlugins-1.0.0.jar`
: Version 1.0.0 of the user plug-in bundle with sample

ObjectGridEventListener and MapSerializerPlugin plug-in implementations. This bundle is located in the `wxs_sample_osgi_root/lib` directory. The services are configured with service ranking 1.

This version uses the standard Blueprint XML to configure the eXtreme Scale plug-in services. The service class is a user-implemented class for WebSphere eXtreme Scale interface, com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory. The user-implemented class creates a bean for each request and works similar to a prototype-scoped bean.

ProtoBufSamplePlugins-2.0.0.jar

Version 2.0.0 of the user plug-in bundle with sample ObjectGridEventListener and MapSerializerPlugin plug-in implementations. This bundle is located in the `wxs_sample_osgi_root/lib` directory. The services are configured with service ranking 2.

This version uses the standard Blueprint XML to configure the eXtreme Scale plug-in services. The service class is using a WebSphere eXtreme Scale, built-in class, com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactoryImpl, which uses the BlueprintContainer service. Using the standard Blueprint XML configuration, the beans can be configured either as a prototype scope or singleton scope. The bean is not configured as a shard scope.

ProtoBufSamplePlugins-Gemini-3.0.0.jar

Version 3.0.0 of the user plug-in bundle with sample ObjectGridEventListener and MapSerializerPlugin plug-in implementations. This bundle is located in the `wxs_sample_osgi_root/lib` directory. The services are configured with service ranking 3.

This version uses the Eclipse Gemini-specific Blueprint XML to configure the eXtreme Scale plug-in services. The service class is using a WebSphere eXtreme Scale built-in class, com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactoryImpl, which uses the BlueprintContainer service. The way to configure a shard scope bean is using a Gemini-specific approach. This version configures the myShardListener bean as a shard scope bean by providing `{http://www.ibm.com/schema/objectgrid}shard` as the scope value, and configuring a dummy attribute so that the custom scope is recognized by Gemini. This is due to the following Eclipse issue: https://bugs.eclipse.org/bugs/show_bug.cgi?id=348776

ProtoBufSamplePlugins-Aries-4.0.0.jar

Version 4.0.0 of the user plug-in bundle with sample ObjectGridEventListener and MapSerializerPlugin plug-in implementations. This bundle is located in the `wxs_sample_osgi_root/lib` directory. The services are configured with service ranking 4.

This version uses standard Blueprint XML to configure the eXtreme Scale plug-in services. The service class is using a WebSphere eXtreme Scale, built-in class, com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactoryImpl, which uses the BlueprintContainer service. Using the standard Blueprint XML configuration, the beans can be configured using a custom scope. This version configures the myShardListenerbean as a shard scoped bean by providing `{http://www.ibm.com/schema/objectgrid}shard` as the scope value.

`ProtoBufSamplePlugins-Activator-5.0.0.jar`

Version 5.0.0 of the user plug-in bundle with sample ObjectGridEventListener and MapSerializerPlugin plug-in implementations. This bundle is located in the *`wxs_sample_osgi_root`*`/lib` directory. The services are configured with service ranking 5.

This version does not use Blueprint container at all. In this version, the services are registered using OSGi service registration. The service class is a user-implemented class for the WebSphere eXtreme Scale interface, com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory. The user-implemented class creates a bean for each request. It works similar to a prototype-scoped bean.

Lesson checkpoint:

By exploring the bundles that are provided with the OSGi sample, you can better understand how to develop your own implementations that will run in the OSGi container.

You learned:
- About bundles that included with the OSGi sample
- The location of those bundles
- The service ranking that each bundle has been configured with

Lesson 1.2: Understand the OSGi configuration files

The OSGi sample includes configuration files that you use to start and configure the WebSphere eXtreme Scale grid and server.

OSGi configuration files:
In this lesson, you will explore the following configuration files that are included with the OSGI sample:
- `collocated.server.properties`
- `protoBufObjectGrid.xml`
- `protoBufDeployment.xml`
- `blueprint.xml`

collocated.server.properties

A server configuration is required to start a server. When the eXtreme Scale server bundle is started, it does not start a server. It waits for the configuration PID, `com.ibm.websphere.xs.server`, to be created with a server property file. This server property file specifies the server name, port number, and other server properties.

In most cases, you create a configuration to set the server property file. In rare cases, you might want only to start a server, with every property set to a default value. In that case, you can create a configuration called `com.ibm.websphere.xs.server` with value set to `default`.

For more details about the server property file, see the Server properties file topic.

The OSGi sample server properties file starts a single catalog. This sample property file starts a single catalog service and a container server in the OSGi framework process. eXtreme Scale clients connect to port 2809 and JMX clients connect to port 1099. The content of the sample server property file is:

```
serverName=collocatedServer
isCatalog=true
catalogClusterEndPoints=collocatedServer:localhost:6601:6602
traceSpec=ObjectGridOSGi=all=enabled
traceFile=logs/trace.log
listenerPort=2809
JMXServicePort=1099
```

protoBufObjectGrid.xml

The sample `protoBufObjectGrid.xml` ObjectGrid descriptor XML file contains the following content, with comments removed.

```
<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>
```

There are two plug-ins configured in this ObjectGrid descriptor XML file:

ObjectGridEventListener
The shard-level plug-in. For each ObjectGrid instance, there is an instance of ObjectGridEventListener. It is configured to use the OSGi service myShardListener. That means when the grid is created, the ObjectGridEventListener plug-in uses the OSGi service myShardListener with the highest service ranking available.

MapSerializerPlugin
The map-level plug-in. For the backing map named`Map`, there is a MapSerializerPlugin plug-in configured. It is configured to use the OSGI service myProtoBufSerializer. That means when the map is created, the MapSerializerPlugin plug-in uses the service, myProtoBufSerializer, with the highest ranked service ranking available.

protoBufDeployment.xml

The deployment descriptor XML file describes the deployment policy for the grid named `Grid`, which uses five partitions. See the following code example of the XML file:

```
<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

As an alternative to using the `collocated.server.properties` file in conjunction with configuration PID, `com.ibm.websphere.xs.server`, you can include the ObjectGrid XML and deployment XML files in an OSGi bundle, along with a Blueprint XML file as shown in the following example:

```
<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>
```

Lesson checkpoint:

In this lesson, you learned about the configuration files that are used in the OSGi sample. Now, when you start and configure the eXtreme Scale grid and server, you will understand which files are being used in these processes and how these files interact with your plug-ins in the OSGi framework.

Module 2: Installing and starting eXtreme Scale bundles in the OSGi framework

Use the lessons in this module to install the eXtreme Scale server bundle into the OSGi container, and start the WebSphere eXtreme Scale server.

Starting the server in the OSGi framework does not mean that your OSGi bundles are ready to run. You must configure the server properties and containers so that the OSGi bundles that you install are recognized and can run correctly.

Learning objectives

After completing the lessons in this module, you will understand the concepts and know how to complete the following tasks:
- Install eXtreme Scale bundles using the Equinox OSGi console.
- Configure the eXtreme Scale server.
- Configure the eXtreme Scale container.
- Install and start eXtreme Scale sample bundles.

Prerequisites

To complete this module, the following tasks are required before you begin:
- Install and extract the eXtreme Scale product

- Set up the Eclipse Equinox Environment

You must also prepare to access the following files to complete the lessons in this module:

- `objectgrid.jar` bundle. You install this eXtreme Scale bundle.
- `collocated.server.properties` file. You add the server properties to this configuration file.

You can expect to install and start the following bundles:

- `protobuf-java-2.4.0a-bundle.jar` bundle
- `ProtoBufSamplePlugins-1.0.0.jar` bundle

Lesson 2.1: Start the console and install the eXtreme Scale server bundle

In this lesson, you use the Equinox OSGi console to install the WebSphere eXtreme Scale server bundle.

1. Use the following command to start the Equinox OSGi console:

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

2. After the OSGi console is started, issue the ss command in the console, and the following bundles are started:

   Attention: If you completed the task, Installing eXtreme Scale bundles, then the bundle has already been activated. If the bundle is started, then stop the bundle before you complete this step.

   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. Install the `objectgrid.jar` bundle. To start a server in the Java virtual machine (JVM), you need to install an eXtreme Scale server bundle. This eXtreme Scale server bundle can start a server and create containers. Use the following command to install the `objectgrid.jar` file:

   ```
   osgi> install file:///wxs_home/lib/objectgrid.jar
   ```

   See the following example:

   ```
   osgi> install file:///opt/wxs/ObjectGrid/lib/objectgrid.jar
   ```

   Equinox displays its bundle ID; for example:

   ```
   Bundle id is 19
   ```

   Remember: Your bundle ID might be different. The file path must be an absolute URL to the bundle path. Relative paths are not supported.

Lesson checkpoint:

In this lesson, you used the Equinox OSGi console to install the `objectgrid.jar` bundle, which you will use to start a server and create a container later in this tutorial.

Lesson 2.2: Customize and configure the eXtreme Scale server

Use this lesson to customize and add the server properties to the WebSphere eXtreme Scale server.

1. Edit the `wxs_sample_osgi_root/projects/server/properties/collocated.server.properties` file.
   a. Change the traceFile property to `equinox_root/logs/trace.log`.
2. Save the file.
3. Enter the following lines of code in the OSGI console to create the server configuration from the file. The following example is displayed on multiple lines for publication purposes.

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

4. To view the configuration, run the following command:

```
osgi> cm get com.ibm.websphere.xs.server
Configuration for service (pid) "com.ibm.websphere.xs.server"
(bundle location = null)
key                     value
----                    ----
objectgrid.server.props  wxs_sample_osgi_root/projects/server/properties/collocated.server.properties
service.pid             com.ibm.websphere.xs.server
```

Lesson checkpoint:

In this lesson, you edited the `wxs_sample_osgi_root/projects/server/properties/collocated.server.properties` file to specify server settings, such as the working directory and the location for the trace log files.

Lesson 2.3: Configure the eXtreme Scale container

Complete this lesson to configure a container, which includes the WebSphere eXtreme Scale ObjectGrid descriptor XML file and ObjectGrid deployment XML file. These files include the configuration for the grid and its topology.

To create a container, first create a configuration service using the managed service factory process identification number (PID), com.ibm.websphere.xs.container. The service configuration is a managed service factory, so you can create multiple service PIDs from the factory PID. Then, to start the container service, set the objectgridFile and deploymentPolicyFile PIDs to each service PID.

Complete the following steps to customize and add the server properties to the OSGi framework:

1. In the OSGI console, enter the following command to create the container from the file:
   ```
   osgi> cm createf com.ibm.websphere.xs.container
   PID: com.ibm.websphere.xs.container-1291179621421-0
   ```
2. Enter the following commands to bind the newly created PID to the ObjectGrid XML files.

   Remember: The PID number will be different from what is included in this example.

```
osgi> cm put com.ibm.websphere.xs.container-1291179621421-0 objectgridFile wxs_sample_osgi_root/projects/server/META-INF/protoBufObjectgrid.xml

osgi> cm put com.ibm.websphere.xs.container-1291179621421-0 deploymentPolicyFile wxs_sample_osgi_root/projects/server/META-INF/protoBufDeployment.xml
```

3. Use the following command to display the configuration:

```
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
```

Lesson checkpoint:

In this lesson, you created a configuration service, which you used to create an eXtreme Scale container. Since the ObjectGrid XML files contain the configuration for the grid and its topology, you had to bind the container that you created to those ObjectGrid XML files. With this configuration, the eXtreme Scale container can recognize the OSGi bundles that you will run later in this tutorial.

Lesson 2.4: Install the Google Protocol Buffers and sample plug-in bundles

Complete this tutorial to install the `protobuf-java-2.4.0a-bundle.jar` bundle and the `ProtoBufSamplePlugins-1.0.0.jar` plug-in bundle using the Equinox OSGi console.

Install the Google Protocol Buffers plug-in:
Complete the following steps to install the Google Protocol Buffers plug-in.

In the OSGI console, enter the following command to install the plug-in:

```
osgi> install file:///wxs_sample_osgi_root/lib/com.google.protobuf_2.4.0a.jar
```

The following output is displayed:

```
Bundle ID is 21
```

Sample plug-in bundles overview:
The OSGi sample includes five sample bundles that include eXtreme Scale plug-ins, including a custom ObjectGridEventListener and MapSerializerPlugin

plug-in. The MapSerializerPlugin plug-in uses the Google Protocol Buffers sample and messages provided by the MapSerializerPlugin sample.

The following bundles are located in *wxs_sample_osgi_root*/lib directory: ProtoBufSamplePlugins-1.0.0.jar and the ProtoBufSamplePlugins-2.0.0.jar.

The blueprint.xml file has the following content with comments removed:

```
<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>
```

The Blueprint XML file exports two services, myShardListener and myProtoBufSerializer. These two services are referenced in the protoBufObjectgrid.xml file.

Install the sample plug-in bundle:

Complete the following steps to install the ProtoBufSamplePlugins-1.0.0.jar bundle.

Run the following command in the Equinox OSGi console to install the ProtoBufSamplePlugins-1.0.0.jar plugin bundle:

```
osgi> install file:///wxs_sample_osgi_root/lib/ProtoBufSamplePlugins-1.0.0.jar
```

The following output is displayed:

```
Bundle ID is 22
```

Lesson checkpoint:

In this lesson, you installed the protobuf-java-2.4.0a-bundle.jar bundle and the ProtoBufSamplePlugins-1.0.0.jar plug-in bundle.

Lesson 2.5: Start the OSGi bundles

The WebSphere eXtreme Scale server is packaged as an OSGi server bundle. Complete this lesson to install the eXtreme Scale server bundle as well as other OSGi bundles that you have installed.

1. Run the **ss** command to view the IDs for each bundle.

```
osgi> ss

Framework is launched.

id State Bundle
0 ACTIVE org.eclipse.osgi_3.6.1.R36x_v20100806
1 ACTIVE org.eclipse.osgi.services_3.2.100.v20100503
2 ACTIVE org.eclipse.osgi.util_3.2.100.v20100503
3 ACTIVE org.eclipse.equinox.cm_1.0.200.v20100520
4 ACTIVE com.springsource.org.apache.commons.logging_1.1.1
5 ACTIVE com.springsource.org.aopalliance_1.0.0
6 ACTIVE org.springframework.aop_3.0.5.RELEASE
7 ACTIVE org.springframework.asm_3.0.5.RELEASE
```

```
8 ACTIVE org.springframework.beans_3.0.5.RELEASE
9 ACTIVE org.springframework.context_3.0.5.RELEASE
10 ACTIVE org.springframework.core_3.0.5.RELEASE
11 ACTIVE org.springframework.expression_3.0.5.RELEASE
12 ACTIVE org.apache.felix.fileinstall_3.0.2
13 ACTIVE net.luminis.cmc_0.2.5
15 ACTIVE org.eclipse.gemini.blueprint.core_1.0.0.RELEASE
16 ACTIVE org.eclipse.gemini.blueprint.extender_1.0.0.RELEASE
17 ACTIVE org.eclipse.gemini.blueprint.io_1.0.0.RELEASE
19 RESOLVED com.ibm.websphere.xs.server_7.1.1
21 RESOLVED Google_ProtoBuf_2.4.0
22 RESOLVED ProtoBufPlugins_1.0.0
```

2. Start each bundle that you have installed. You must start the bundles in a specific order. See the order of the bundle IDs from the previous example.
   a. Start the sample plug-in bundle, `ProtoBufPlugins_1.0.0`. Run the following command in the Equinox OSGi console to start the bundle. In this example, the bundle ID of the sample plug-in is 22.
      ```
      osgi> start 22
      ```
   b. Start the Google Protocol Buffers bundle, `Google_ProtoBuf_2.4.0`. Run the following command in the Equinox OSGi console to start the bundle. In this example, the bundle ID of the Google Protocol Buffers plug-in is 21.
      ```
      osgi> start 21
      ```
   c. Start the server bundle, `com.ibm.websphere.xs.server_7.1.1`. Run the following command in the OSGi console to start the server. In this example, the bundle ID of the eXtreme Scale server bundle is 19.
      ```
      osgi> start 19
      ```

After you start the server, the MyShardListener event listener is started and ready to insert or update records. You can see the following output on the OSGi console to confirm that the plug-in bundle has started successfully:

```
SystemOut O MyShardListener@1253853884(version=1.0.0) order
com.ibm.websphere.samples.xs.serializer.proto.DataObjects1$Order$Builder
@1aba1aba(22) inserted
```

Lesson checkpoint:

In this lesson, you started two plug-in bundles and the server bundle in the eXtreme Scale container that you configured for the OSGi framework.

Module 3: Running the eXtreme Scale sample client

The WebSphere eXtreme Scale server is now running in an OSGi environment. Complete the steps in this module to run an WebSphere eXtreme Scale client that inserts data into the grid.

Learning objectives

After completing the lessons in this module you will know how to complete the following tasks:

- Run a client application that connects to the grid and inserts and retrieves some data from it.
- Start an order using a non-OSGi client application.

Prerequisites

Complete Module 2: Installing and starting eXtreme Scale bundles in the OSGi framework.

Lesson 3.1: Set up Eclipse to run the client and build the samples

Complete this lesson to import the Eclipse project that you will use to run the client and build the sample plug-ins.

The sample includes a Java SE client program that connects to the grid and inserts and retrieves data from it. It also includes projects that you can use to build and redeploy the OSGi bundles.

The provided project has been tested with Eclipse 3.x and later, and requires only the standard Java development project perspective. Complete the following steps to set up of your WebSphere eXtreme Scale development environment.

1. Open Eclipse to a new or existing workspace.
2. From the File menu, select **Import**.
3. Expand the General folder. Select **Existing Projects into Workspace**, and click **Next**.
4. In the **Select root directory** field, type or browse to the `*wxs_sample_osgi_root*` directory. Click **Finish**. Several new projects are displayed in your workspace. Build errors will be fixed by defining two user libraries. Complete the next steps to define the user libraries.
5. From the Window menu, select **Preferences**.
6. Expand the **Java** > **Build Path** branch, and select **User Libraries**.
7. Define the eXtreme Scale user library.
   a. Click **New**.
   b. Type `eXtremeScale` in the **User Library Name** field, and click **OK**.
   c. Select the new user library, and click **Add JARs**.
      1) Browse and select the `objectgrid.jar` file from the *wxs_install_root*/`lib` directory. Click **OK**.
      2) To include API documentation for the ObjectGrid APIs, select the API documentation location for the `objectgrid.jar` file that you added in the previous step. Click **Edit**.
      3) In the location path box for the API documentation, select the `Javadoc.zip` file that is included in the following directory: *wxs_install_root*`/docs/javadoc.zip`.
8. Define the Google Protocol Buffers user library.
   a. Click **New**.
   b. Type `com.google.protobuf` in the **User Library Name** field, and click **OK**.
   c. Select the new user library, and click **Add JARs**.
      1) Browse and select the `com.google.protobuf_2.4.0.a.jar` file from the *wxs_sample_osgi_root*`/lib` directory. Click **OK**.

Lesson checkpoint:

In this lesson, you imported the sample Eclipse project and defined the user libraries that fixed any build errors.

Lesson 3.2: Start a client and insert data into the grid

Complete this lesson to start a non-OSGi client and run a client application.

The Java client application is `com.ibm.websphere.samples.xs.proto.client.Client`. The Eclipse project, wxs.sample.osgi.protobuf.client, contains the Java client application. The main class file is `com.ibm.websphere.samples.xs.proto.client.Client`.

This client uses a client override, ObjectGrid descriptor XML file to override the OSGi configuration, so that the client can run in a non-OSGi environment. See the following content of the file with comments and headers removed. Some lines of code are displayed on multiple lines for formatting purposes.

```
<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>
```

Click **Run As** > **Java Application** to run the client application.

When you run the application, the following message is displayed. The message indicates that an order was inserted:

```
order
com.ibm.websphere.samples.xs.serializer.proto.DataObjects1$Order$Builder@5d165d16(5000000) inserted
```

Lesson checkpoint:

In this lesson, you started the `com.ibm.websphere.samples.xs.proto.client.Client` application, which produced an order.

Module 4: Querying and upgrading the sample bundle

Complete the lessons in this module to use the **`xscmd`** command to query the service ranking of the sample bundle, upgrade it to a new service ranking, and verify the new service ranking.

Learning objectives

After completing the lessons in this module you will know how to complete the tasks:

- Query the current service ranking for a service.
- Query the current ranking for all services.
- Query all available rankings for a service.
- Query all available service rankings.
- Use the xscmd tool to verify whether specific service rankings are available.

- Update service rankings for sample OSGi services.

Prerequisites

Complete Module 3: Running the eXtreme Scale sample client.

Lesson 4.1: Query service rankings

Complete this lesson to query current service rankings as well as those service rankings that are available for upgrade.

- Query the current service ranking for a service. Enter the following command to query the current service ranking being used for service, myShardListener, which is used by the ObjectGrid named `Grid` and map set named `MapSet`.
  1. Switch to the following directory:
     ```
     cd wxs_home/bin
     ```
  2. Enter the following command to query the current service ranking for the service, myShardListener.
     ```
     ./xscmd.sh -c osgiCurrent -g Grid -ms MapSet -sn myShardListener
     ```
     The following output is displayed:
     ```
     OSGi Service Name: myShardListener
     ObjectGrid Name MapSet Name Server Name        Current Ranking
     --------------- ----------- -----------        ---------------
     Grid            MapSet      collocatedServer   1

     CWXSI0040I: The command osgiCurrent has completed successfully.
     ```
- Query the current ranking for all services. Enter the following command to query the current service ranking for all services that are used by the ObjectGrid named `Grid` and map set named `MapSet`.
  1. Switch to the following directory:
     ```
     cd wxs_home/bin
     ```
  2. Enter the following command to query the current service ranking for all services.
     ```
     ./xscmd.sh -c osgiCurrent -g Grid -ms MapSet
     ```
     The following output is displayed:
     ```
     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.
     ```
- Query all available rankings for a service. Enter the following command to query all of the available service rankings for the service named myShardListener.
  1. Switch to the following directory:
     ```
     cd wxs_home/bin
     ```
  2. Enter the following command to query all available rankings for a service.
     ```
     ./xscmd.sh -c osgiAll -sn myShardListener
     ```
     The following output is displayed:
     ```
     Server: collocatedServer
        OSGi Service Name Available Rankings
        ----------------- ------------------
        myShardListener   1
     ```

```
   Summary - All servers have the same service rankings.

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

The output is grouped by the server. In this example, only the following server exists: collocatedServer.

- Query all available service rankings. Enter the following command to query all of the available service rankings for all services.
  1. Switch to the following directory:
     ```
     cd wxs_home/bin
     ```
  2. Enter the following command to query all available service rankings.
     ```
     ./xscmd.sh -c osgiAll
     ```
     The following output is displayed:
     ```
     Server: collocatedServer
        OSGi Service Name    Available Rankings
        -----------------    ------------------
        myProtoBufSerializer 1
        myShardListener      1

     Summary - All servers have the same service rankings.
     ```
- Install and start Version 2 of the plug-in bundle. In the server OSGi console, install a new bundle that contains a new version of the Order class and the MapSerializerPlugin plug-in. See Lesson 2.4: Install the Google Protocol Buffers and sample plug-in bundles for details about how to install the `ProtoBufSamplePlugins-2.0.0.jar` bundle.
  1. After the installation, start the new bundle. The services for your new bundle are available, but they are not used by the eXtreme Scale server yet. You must run a service update request to use a service with a specific version.
- Now when you query all the available service rankings again, the service ranking 2 is added in the output.
  1. Switch to the following directory:
     ```
     cd wxs_home/bin
     ```
  2. Enter the following command to query all available service rankings.
     ```
     ./xscmd.sh -c osgiAll
     ```
     The following output is displayed:
     ```
     Server: collocatedServer
        OSGi Service Name    Available Rankings
        -----------------    ------------------
        myProtoBufSerializer 1, 2
        myShardListener      1, 2

     Summary - All servers have the same service rankings.
     ```

Lesson checkpoint:

In this tutorial, you queried currently specified and all available service rankings. You also displayed the service ranking for a new bundle that you installed and started.

Lesson 4.2: Determine whether specific service rankings are available

Complete this lesson to determine whether specific service rankings are available for the service names that you specify.

1. Enter the following command to determine whether the service named myShardListener, with service ranking 2 and service named myProtoBufSerializer, with service ranking 2 are available. The service ranking list is passed in using `-sr` option.
   a. Switch to the following directory:
      ```
      cd wxs_home/bin
      ```
   b. Enter the following command to determine whether the services are available:
      ```
      ./xscmd.sh -c osgiCheck -sr "myShardListener;2,myProtoBufSerializer;2"
      ```
      The following output is displayed:
      ```
      CWXSI0040I: The command osgiCheck has completed successfully.
      ```
2. Enter the following command to determine whether the service named myShardListener, with service ranking 2 and the service named myProtoBufSerializer, with service ranking 3 are available.
   a. Switch to the following directory:
      ```
      cd wxs_home/bin
      ```
   b. Enter the following command to determine whether the services are available:
      ```
      ./xscmd.sh -c osgiCheck -sr "myShardListener;2,myProtoBufSerializer;3"
      ```
      The following output is displayed:
      ```
      Server           OSGi Service         Unavailable Rankings
      ------           ------------         --------------------
      collocatedServer myProtoBufSerializer          3
      ```

Lesson checkpoint:

In this lesson, you specified the services myShardListener and myProtoBufSerializer, along with specific service rankings to determine whether those rankings were available.

Lesson 4.3: Update the service rankings

Complete this lesson to update current service rankings that you queried.

1. Update the service rankings of the services, myShardListener and myProtoBufSerializer, to service ranking 2. The service ranking list is passed in using `-sr` option.
   a. Switch to the following directory:
      ```
      cd wxs_home/bin
      ```
   b. Enter the following command to update the service rankings:
      ```
      ./xscmd.sh -c osgiUpdate -g Grid -ms MapSet -sr "myShardListener;2,myProtoBufSerializer;2"
      ```
      The following output is displayed:
      ```
      Update succeeded for the following service rankings:
      Service              Ranking
      -------              -------
      myProtoBufSerializer 2
      myShardListener      2

      CWXSI0040I: The command osgiUpdate has completed successfully.
      ```
      The following output is displayed on the OSGi console:

```
SystemOut O MyShardListener@326505334(version=2.0.0) order
com.ibm.websphere.samples.xs.serializer.proto.DataObjects2$Order$Builder@
22342234(34) updated
```

Notice that the MyShardListener service is now version 2.0.0, which has service ranking 2.

2. Run the **xscmd** command to query the current service ranking for all services that are used by the ObjectGrid named `Grid` and the map set named `MapSet`.
   a. Switch to the following directory:
   ```
   cd wxs_home/bin
   ```
   b. Enter the following command to query the service rankings for all services that are used by `Grid` and `MapSet`:
   ```
   ./xscmd.sh -c osgiCurrent -g Grid -ms MapSet
   ```
   The following output is displayed:
   ```
   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.
   ```

Lesson checkpoint:

In this lesson, you updated the service rankings for services myShardListener and myProtoBufSerializer.

.

Chapter 4. Installing

WebSphere eXtreme Scale is an in-memory data grid that you can use to dynamically partition, replicate, and manage application data and business logic across multiple servers. After determining the purposes and requirements of your deployment, install eXtreme Scale on your system.

Before you begin

- Before you begin the installation, you should have an understanding of WebSphere eXtreme Scale caching architectures, cache and database integration, serialization, scalability and availability. See Product overview for more information.
- Plan your WebSphere eXtreme Scale deployment. For more information about the different caching topologies, sizing information, and more, see Chapter 2, "Planning," on page 13.
- Verify that your environment meets the prerequisites to install eXtreme Scale. See "Hardware and software requirements" on page 59 for more information.
- For more information on environments and other requirements, see "Planning for installation" on page 172.
- If you are installing an upgrade on a previous version of WebSphere eXtreme Scale, follow the steps described in "Updating eXtreme Scale servers" on page 235.

Installation overview

You can install WebSphere eXtreme Scale in a stand-alone or WebSphere Application Server environment.

WebSphere eXtreme Scale environment types

- **WebSphere Application Server environment:**

  By installing WebSphere eXtreme Scale on the nodes in your WebSphere Application Server environment, you can automatically start catalog servers and container servers in the same cell as your deployment manager and other application servers.
- **Stand-alone environment:**

  In a stand-alone installation, you install WebSphere eXtreme Scale in an environment that does not have WebSphere Application Server. With a stand-alone environment, you manually configure and start the catalog server and container server processes.

WebSphere eXtreme Scale installation types

If you have servers that are running client applications that access the data grid, you can use a client-only installation. Or you can choose a full (client and server) installation on nodes that run catalog servers or container servers.

- **Full (Client and Server) installation:**
  - When you are installing on WebSphere Application Server, you can choose to install the client only or both the client and server.
  - When you are installing in a stand-alone environment, you can choose to install the client-only or both the client and server.

- **Client installation:**

  You can use the client-only installation on nodes that are running the client applications.

Planning for installation

Before you install the product, you must consider your environment.

Installation topologies

With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Development node

The simplest installation scenario is creating a development node. In this scenario, you install the client and server installation of WebSphere eXtreme Scale one time on the node where you want to develop your application.

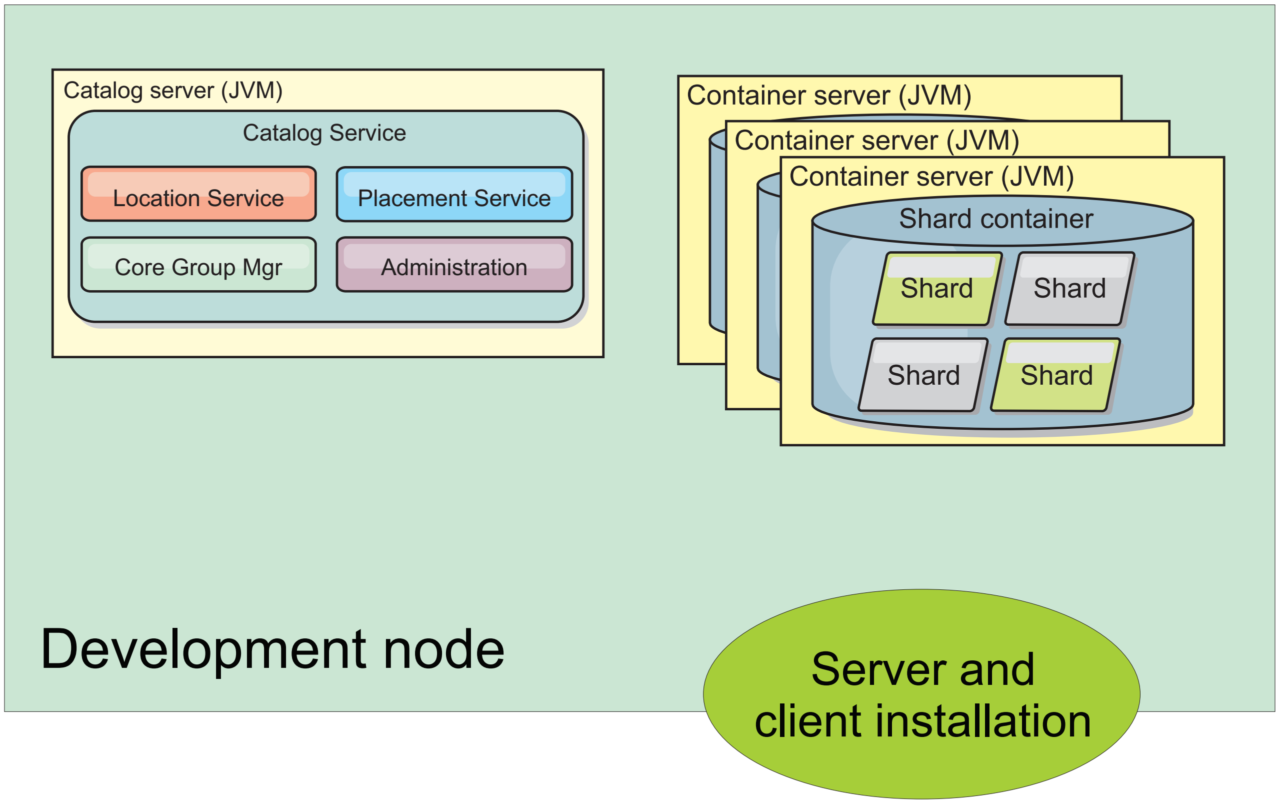


Figure 24. Development node

After you complete the installation on your development node, you can configure your development environment and begin writing your applications.

Stand-alone topology

A stand-alone topology consists of servers that are not running on WebSphere Application Server. You can create many different stand-alone topologies, but the following topology is included as an example. In this topology, two data centers are present. In each data center, WebSphere eXtreme Scale full installations (client and server) and client-only installations are installed on the physical servers. The client-only installations are on the nodes that are running the web applications that are using the data grid. These nodes do not run any catalog or container servers, so the server installation is not required. A multi-master link connects the two catalog service domains in the configuration. The multi-master link enables replication between the shards in the container servers in the different data centers.

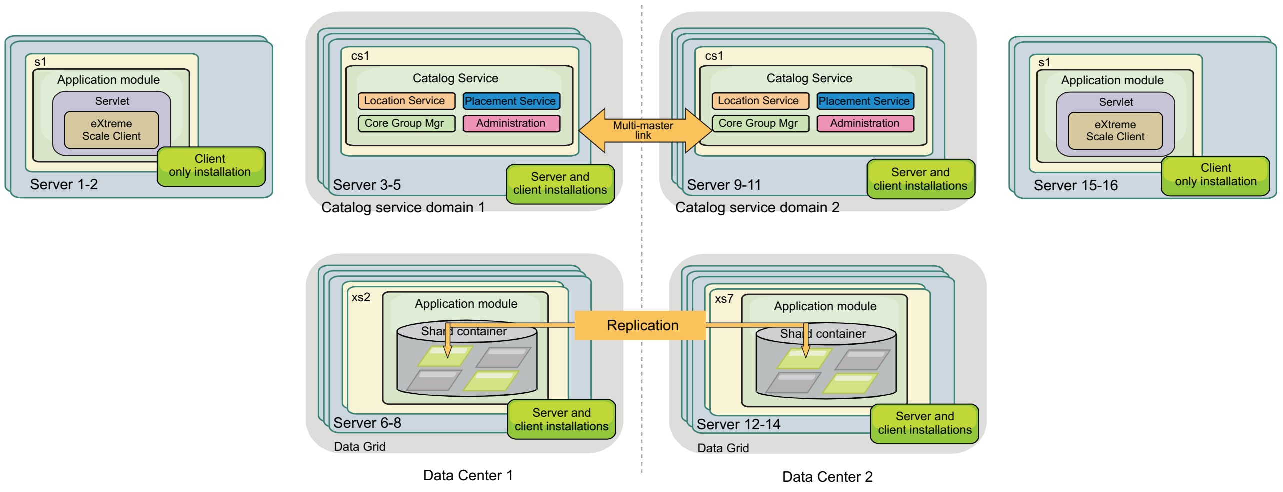


Figure 25. Stand-alone topology with two data centers

Advantages to using a stand-alone topology:

- Flexible integration options that can be embedded with vendor frameworks and libraries.
- Smaller footprint than a WebSphere Application Server topology.
- Fewer licensing requirements than a WebSphere Application Server topology.
- Expanded Java Runtime Environment (JRE) options.

WebSphere Application Server topology

You can also create an installation that runs entirely in a WebSphere Application Server cell. The clients, catalog servers, and container servers each have an associated cluster. The nodes that run the application have the client-only installation. The other nodes have the client and server installation.

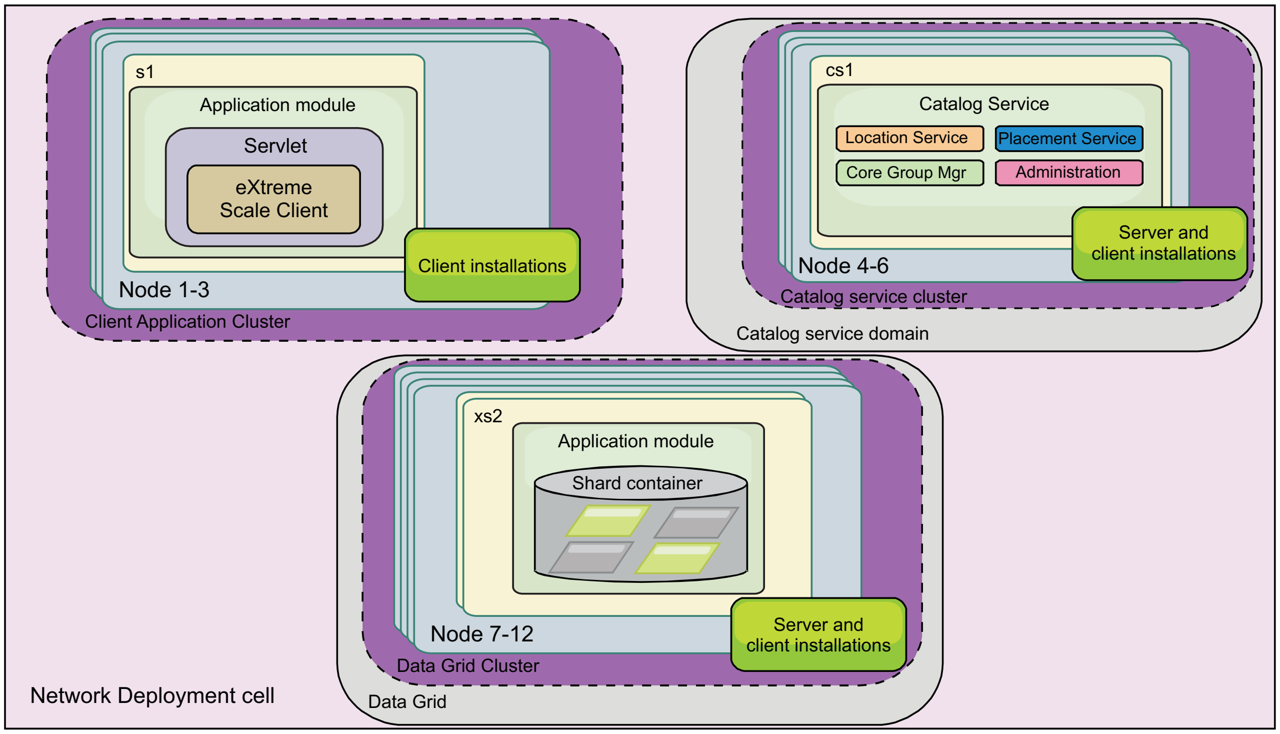


Figure 26. WebSphere Application Server topology example

Advantages of using a WebSphere Application Server topology.

- Centralized and consistent administration and configuration.
- Security integration.
- Java EE application integration.
- Performance monitoring infrastructure (PMI) integration.
- Integration with the following WebSphere Application Server components: OpenJPA L2 cache, dynamic cache, and HTTP session persistence.

Mixed topology

You can create a mixed topology that contains both WebSphere Application Server and stand-alone servers. In the following example, the client applications are running in the WebSphere Application Server cell, while the catalog servers and container servers are running in stand-alone mode.

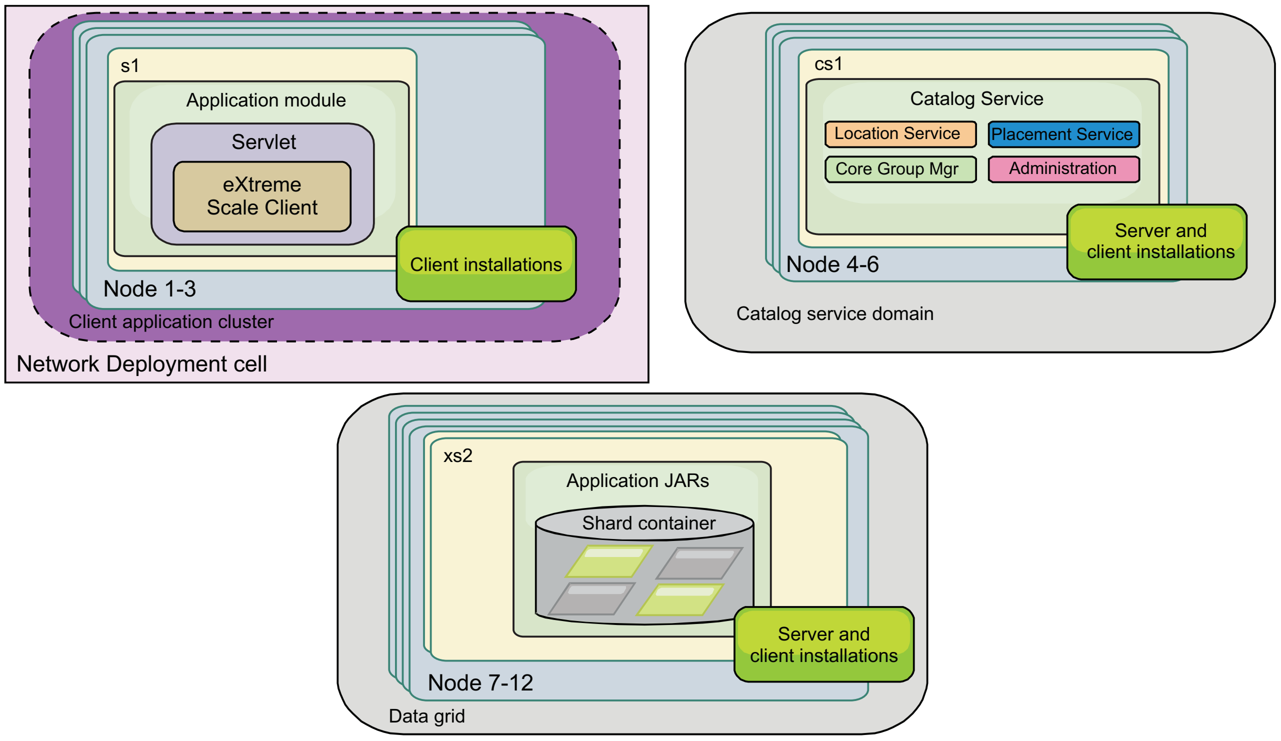


Figure 27. Mixed topology example

Related tasks:

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

"Configuring catalog servers and catalog service domains" on page 294
The catalog service hosts logic that is typically idle during steady states. As a result, the catalog service minimally influences scalability. The service is built to service hundreds of container servers that become available simultaneously. For high availability, configure the catalog service into a catalog service domain.

"Configuring the quorum mechanism" on page 319
The quorum mechanism is configured for each catalog service. You must enable the quorum mechanism on all of the catalog servers in the catalog service domain.

"Tuning the heartbeat interval setting for failover detection" on page 321
You can configure the amount of time between system checks for failed servers with the heartbeat interval setting. This setting applies to catalog servers only.

"Configuring the catalog service in WebSphere Application Server" on page 298
Catalog service processes can run in WebSphere Application Server. The server life cycle in WebSphere Application Server determines when the catalog service starts and stops.

"Creating catalog service domains in WebSphere Application Server" on page 299
Catalog service domains define a group of catalog servers that manage the placement of shards and monitor the health of container servers in your data grid.

"Configuring catalog and container servers" on page 293
WebSphere eXtreme Scale has two types of servers: catalog servers and container servers. Catalog servers control the placement of shards and discover and monitor the container servers. Multiple catalog servers can join a catalog service domain to provide high availability to the environment. A container server is a Java virtual machine (JVM) that stores the application data for the data grid.

"Starting and stopping stand-alone servers" on page 459
You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

"Using the embedded server API to start and stop servers" on page 476
With WebSphere eXtreme Scale, you can use a programmatic API for managing the life cycle of embedded servers and containers. You can programmatically configure the server with any of the options that you can also configure with the command line options or file-based server properties. You can configure the embedded server to be a container server, a catalog service, or both.

"Configuring WebSphere Application Server applications to automatically start container servers" on page 327
Container servers in a WebSphere Application Server environment start automatically when a module starts that has the eXtreme Scale XML files included.

"Configuring container servers in WebSphere Application Server" on page 325
Configure container servers in WebSphere Application Server by using a server properties file and deployment policy XML file that is embedded into a Java EE application module. Container servers stop and start when the application is stopped and started.

"Controlling placement" on page 486
You can use several different options to control when shards are placed on various container servers in the configuration. During startup, you might choose to delay the placement of shards. When you are running all of your container servers, you might need to suspend, resume, or change placement while you maintain servers.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**`startOgServer`** script" on page 466
The **`startOgServer`** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

"Catalog service domain administrative tasks" on page 301
You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Hardware and software requirements

Browse an overview of hardware and operating system requirements. Although you are not required to use a specific level of hardware or operating system for WebSphere eXtreme Scale, formally supported hardware and software options are available on the Systems Requirements page of the product support site. If a conflict exists between the information center and the System Requirements page, the information at the website takes precedence. Prerequisite information in the information center is provided as a convenience only.

See the System Requirements page for the official set of hardware and software requirements.

You can install and deploy the product in Java EE and Java SE environments. You can also bundle the client component with Java EE applications directly without integrating with WebSphere Application Server.

Hardware requirements

WebSphere eXtreme Scale does not require a specific level of hardware. The hardware requirements are dependent on the supported hardware for the Java Platform, Standard Edition installation that you use to run WebSphere eXtreme Scale. If you are using eXtreme Scale with WebSphere Application Server or another Java Platform, Enterprise Edition implementation, the hardware requirements of these platforms are sufficient for WebSphere eXtreme Scale.

Operating system requirements

7.1.1

- **Without the web console**

eXtreme Scale does not require a specific operating system level. Each Java SE and Java EE implementation requires different operating system levels or fixes for problems that are discovered during the testing of the Java implementation. The levels required by these implementations are sufficient for eXtreme Scale.

- 7.1.1 **With the web console**

  The following requirements apply for each operating system if using the console:
  - Linux: 32 bit or 64 bit JVM
  - Linux PPC: 32 bit JVM only
  - Windows: 32 bit JVM only
  - AIX: 32 bit JVM only

Web browser requirements

The web console supports the following Web browsers:
- Mozilla Firefox, version 3.5.x and later
- Mozilla Firefox, version 3.6.x and later
- Microsoft Internet Explorer, version 7 or 8

WebSphere Application Server requirements

7.1.1

- WebSphere Application Server Version 6.1.0.39 or later
- WebSphere Application Server Version 7.0.0.19 or later
- WebSphere Application Server Version 8.0.0.1 or later

See the Recommended fixes for WebSphere Application Server for more information.

Java requirements

7.1.1 Other Java EE implementations can use the eXtreme Scale run time as a local instance or as a client to eXtreme Scale servers. To implement Java SE, you must use Version 5 or later.

Related tasks:

"Migrating to WebSphere eXtreme Scale Version 7.1.1" on page 238
With the WebSphere eXtreme Scale installer, you cannot upgrade or modify a previous installation. You must uninstall the previous version before you install the new version. You do not need to migrate your configuration files because they are backward compatible. However, if you changed any of the script files that are shipped with the product, you must reapply these changes to the updated script files.

"Stopping stand-alone servers" on page 472
You can use the `stopOgServer` script to stop eXtreme Scale server processes.

7.1.1 "Installing WebSphere eXtreme Scale with the installation wizard" on page 183
You can use the installation wizard to install WebSphere eXtreme Scale for stand-alone or WebSphere Application Server configurations.

Java SE considerations

WebSphere eXtreme Scale requires Java SE 5, Java SE 6, or Java SE 7. In general, newer versions of Java SE have better functionality and performance.

Supported versions

You can use WebSphere eXtreme Scale with Java SE 5, Java SE 6 7.1.1.1+, and Java SE 7. The version that you use must be currently supported by the Java Runtime Environment (JRE) vendor. If you want to use Secure Sockets Layer (SSL), you must use an IBM Runtime Environment.

IBM Runtime Environment, Java Technology Edition Version 5, Version 6 7.1.1.1+, and Version 7 are supported for general use with the product. Version 6 Service Release 9 Fix Pack 2 is a fully supported JRE that is installed as a part of the stand-alone WebSphere eXtreme Scale and WebSphere eXtreme Scale Client installations in the *wxs_install_root*/`java` directory and is available to be used by both clients and servers. If you are installing WebSphere eXtreme Scale within WebSphere Application Server, you can use the JRE that is included in the WebSphere Application Server installation. For the web console, you must use IBM Runtime Environment, Java Technology Edition Version 6 Service Release 7 and later service releases only.

WebSphere eXtreme Scale takes advantage of Java Development Kit (JDK) Version 5 , Version 6 7.1.1.1+, and Version 7 functionality as it becomes available. Generally, newer versions of the Java Development Kit (JDK) and Java SE have better performance and functionality.

For more information, see Supported software.

WebSphere eXtreme Scale features that are dependent on Java SE

Table 3. Features that require Java SE 5, Java SE 6, and Java SE 7. WebSphere eXtreme Scale uses functionality that is introduced in Java SE 5 or Java SE 6 to provide the following product features.

Feature	**Supported in Java SE 5 and later service releases**	**Supported in Java SE Version 6 7.1.1.1+, Version 7 and later service releases**
EntityManager API annotations (Optional: You can also use XML files)	X	X
Java Persistence API (JPA): JPA loader, JPA client loader, and JPA time-based updater	X	X
Memory-based eviction (uses MemoryPoolMXBean)	X	X

Table 3. Features that require Java SE 5, Java SE 6, and Java SE 7 (continued). WebSphere eXtreme Scale uses functionality that is introduced in Java SE 5 or Java SE 6 to provide the following product features.

Feature	Supported in Java SE 5 and later service releases	Supported in Java SE Version 6 7.1.1.1+, Version 7 and later service releases
Instrumentation agents: • `wxssizeagent.jar`: Increases the accuracy of the used bytes map metrics. • `ogagent.jar`: Increases the performance of field-access entities.	X	X
Web console for monitoring		X

Upgrading the JDK in WebSphere eXtreme Scale

Common questions about the upgrade process for releases of WebSphere eXtreme Scale in both stand-alone and WebSphere Application Server environments follow:

- How do I upgrade the JDK that is included with WebSphere eXtreme Scale for WebSphere Application Server?

  You need to use the JDK upgrade process that is made available by WebSphere Application Server. For more information, see http://www-304.ibm.com/support/docview.wss?uid=swg21427178.
- Which version of the JDK should I use when using WebSphere eXtreme Scale in a WebSphere Application Server environment?

  You can use any level of JDK that is supported by WebSphere Application Server, for the supported version of WebSphere Application Server.

Related reference:

"**`startOgServer`** script" on page 466
The **`startOgServer`** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

Tuning the IBM virtual machine for Java

Java EE considerations

As you prepare to integrate WebSphere eXtreme Scale in a Java Platform, Enterprise Edition environment, consider certain items, such as versions, configuration options, requirements and limitations, and application deployment and management.

Running eXtreme Scale applications in a Java EE environment

A Java EE application can connect to a remote eXtreme Scale application. Additionally, the WebSphere Application Server environment supports starting an eXtreme Scale server as an application starts in the application server.

If you use an XML file to create an ObjectGrid instance, and the XML file is in the module of the enterprise archive (EAR) file, access the file by using the getClass().getClassLoader().getResource("META-INF/objGrid.xml") method to obtain a URL object to use to create an ObjectGrid instance. Substitute the name of the XML file that you are using in the method call.

You can use startup beans for an application to bootstrap an ObjectGrid instance when the application starts, and to destroy the instance when the application stops. A startup bean is a stateless session bean with a com.ibm.websphere.startupservice.AppStartUpHome remote location and a com.ibm.websphere.startupservice.AppStartUp remote interface. The remote interface has two methods: the start method and the stop method. Use the start method to bootstrap the instance, and use the stop method to destroy the instance. The application uses the ObjectGridManager.getObjectGrid method to maintain a reference to the instance. See the information about accessing an ObjectGrid with the ObjectGridManager in the *Programming Guide* for more information.

Using class loaders

When application modules that use different class loaders share a single ObjectGrid instance in a Java EE application, verify the objects that are stored in eXtreme Scale and the plug-ins for the product are in a common loader in the application.

Managing the life cycle of ObjectGrid instances in a servlet

To manage the life cycle of an ObjectGrid instance in a servlet, you can use the init method to create the instance and the destroy method to remove the instance. If the instance is cached, it is retrieved and manipulated in the servlet code. See the information about accessing an ObjectGrid with the ObjectGridManager interface in the *Programming Guide* for more information.

Related reference:

"**`startOgServer`** script" on page 466
The **`startOgServer`** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

Tuning the IBM virtual machine for Java

Directory conventions

The following directory conventions are used throughout the documentation to reference special directories such as *`wxs_install_root`* and *`wxs_home`*. You access these directories during several different scenarios, including during installation and use of command-line tools.

`wxs_install_root`
The *`wxs_install_root`* directory is the root directory where WebSphere eXtreme Scale product files are installed. The *`wxs_install_root`* directory can be the directory in which the trial archive is extracted or the directory in which the WebSphere eXtreme Scale product is installed.

- Example when extracting the trial:

  Example: `/opt/IBM/WebSphere/eXtremeScale`
- Example when WebSphere eXtreme Scale is installed to a stand-alone directory:

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

Windows **Example:** `C:\Program Files\IBM\WebSphere\eXtremeScale`

- Example when WebSphere eXtreme Scale is integrated with WebSphere Application Server:

  Example: `/opt/IBM/WebSphere/AppServer`

`wxs_home`

The *`wxs_home`* directory is the root directory of the WebSphere eXtreme Scale product libraries, samples, and components. This directory is the same as the *`wxs_install_root`* directory when the trial is extracted. For stand-alone installations, the *`wxs_home`* directory is the `ObjectGrid` subdirectory within the *`wxs_install_root`* directory. For installations that are integrated with WebSphere Application Server, this directory is the `optionalLibraries/ObjectGrid` directory within the *`wxs_install_root`* directory.

- Example when extracting the trial:

  Example: `/opt/IBM/WebSphere/eXtremeScale`
- Example when WebSphere eXtreme Scale is installed to a stand-alone directory:

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

  Windows **Example:** *wxs_install_root*`\ObjectGrid`
- Example when WebSphere eXtreme Scale is integrated with WebSphere Application Server:

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

`was_root`

The *`was_root`* directory is the root directory of a WebSphere Application Server installation:

Example: `/opt/IBM/WebSphere/AppServer`

`restservice_home`

The *`restservice_home`* directory is the directory in which the WebSphere eXtreme Scale REST data service libraries and samples are located. This directory is named `restservice` and is a subdirectory under the *`wxs_home`* directory.

- Example for stand-alone deployments:

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

  Example: *wxs_home*`\restservice`
- Example for WebSphere Application Server integrated deployments:

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

`tomcat_root`

The *`tomcat_root`* is the root directory of the Apache Tomcat installation.

Example: `/opt/tomcat5.5`

`wasce_root`

The *`wasce_root`* is the root directory of the WebSphere Application Server Community Edition installation.

Example: `/opt/IBM/WebSphere/AppServerCE`

`java_home`

The *`java_home`* is the root directory of a Java Runtime Environment (JRE) installation.

UNIX **Example:** `/opt/IBM/WebSphere/eXtremeScale/java`

Windows **Example:** *wxs_install_root*`\java`

`samples_home`
: The *`samples_home`* is the directory in which you extract the sample files that are used for tutorials.

  UNIX **Example:** *wxs_home*`/samples`

  Windows **Example:** *wxs_home*`\samples`

`dvd_root`
: The *`dvd_root`* directory is the root directory of the DVD that contains the product.

  Example: `dvd_root/docs/`

`equinox_root`
: The *`equinox_root`* directory is the root directory of the Eclipse Equinox OSGi framework installation.

  Example:`/opt/equinox`

`user_home`
: The *`user_home`* directory is the location where user files are stored, such as security profiles.

  Windows `c:\Documents and Settings\`*`user_name`*

  UNIX `/home/`*`user_name`*

Installing WebSphere eXtreme Scale with the installation wizard

7.1.1

You can use the installation wizard to install WebSphere eXtreme Scale for stand-alone or WebSphere Application Server configurations.

Related concepts:

"Security overview" on page 56
WebSphere eXtreme Scale can secure data access, including allowing for integration with external security providers.

"Hardware and software requirements" on page 59
Browse an overview of hardware and operating system requirements. Although you are not required to use a specific level of hardware or operating system for WebSphere eXtreme Scale, formally supported hardware and software options are available on the Systems Requirements page of the product support site. If a conflict exists between the information center and the System Requirements page, the information at the website takes precedence. Prerequisite information in the information center is provided as a convenience only.

Related reference:

"Response file for silent installation" on page 219
Specify parameters at the command line to customize and configure your product installation.

Related information:

"Introduction: Integrate WebSphere eXtreme Scale security with WebSphere Application Server using the WebSphere Application Server Authentication plug-ins" on page 106
In this tutorial, you integrate WebSphere eXtreme Scale security with WebSphere Application Server. First, you configure authentication with a simple web application that uses authenticated user credentials from the current thread to connect to the ObjectGrid. Then, you investigate the encryption of data that is transferred between the client and server with transport layer security. To give users varying levels of permissions, you can configure Java Authentication and Authorization Service (JAAS). After completing the configuration, you can use the **xscmd** utility to monitor your data grids and maps.

WebSphere Application Server: Securing applications and their environment

Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in a stand-alone environment

You can install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in an environment that does not contain WebSphere Application Server or WebSphere Application Server Network Deployment. This type of installation is called a stand-alone installation.

Before you begin

- Verify that the target installation directory is empty or does not exist.

  Important: If a previous version of WebSphere eXtreme Scale or the ObjectGrid component exists in the directory that you specify to install Version 7.1.1, the product is not installed. For example, you might have a previously existing *was_root*/`ObjectGrid` folder. You can either select a different installation directory or cancel the installation. Next, uninstall the previous installation and run the wizard again.
- An IBM Runtime Environment is installed as a part of the stand-alone installation in the *wxs_install_root*/`java` folder.
- If you are installing the client only: Download the WebSphere eXtreme Scale Client for the appropriate platform from the Support site.

About this task

When you install the product as stand-alone, you install the WebSphere eXtreme Scale client and server together. With the WebSphere eXtreme Scale Client installation in stand-alone mode, you are installing a client to access the data in your data grids. Server and client processes, therefore, access all required resources locally. You can also embed WebSphere eXtreme Scale into existingJava Platform, Standard Edition (J2SE) applications by using scripts and Java archive (JAR) files.

Attention: You can also use a non-root (non-administrator) profile for WebSphere eXtreme Scale in a stand-alone environment. To use a non-root profile, you must change the owner of the `ObjectGrid` directory to the non-root profile. Then you can log in with that non-root profile and operate eXtreme Scale as you normally would for a root (administrator) profile.

Procedure

1. Use the wizard to install both the server and the client from the DVD.
   - Run the following script to start the wizard for the WebSphere eXtreme Scale full installation:

  - Linux UNIX *dvd_root*/install
  - Windows *dvd_root*\install.bat
- Run the following script to start the wizard for the WebSphere eXtreme Scale Client installation. The installation files are in the zip file that you download from the Support site:
  - Linux UNIX root/WXS_Client/install
  - Windows root\WXS_Client\install.bat

  Attention: If you use uniform naming conventions (UNC) to identify file paths in your installation command, the items you anticipate installing may not all be installed after the command runs. To avoid trouble, map the file path to a network drive. Run the **install** command against the mapped drive. Using a mapped network drive ensures that all the items are installed.

2. Follow the prompts in the wizard, and click **Finish**.

   Restriction: The optional features panel lists the features from which you can select to install. However, features cannot be added incrementally to the product environment after the product is installed. If you choose not to install a feature with the initial product installation, you must uninstall and reinstall the product to add the feature.

Results

Windows If you are installing the WebSphere eXtreme Scale Client on Windows, you might see the following text in the results of the installation:

```
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.
```

If you see a failure with the `iscdeploy.sh` file, you can ignore the error. This error does not cause any problems.

What to do next

- 7.1.1 Verify the installation. For more information, see “Verifying the installation” on page 227.
- Start configuring your WebSphere eXtreme Scale or WebSphere eXtreme Scale Client installation. For more information, see “Taking the first steps after installation” on page 228.

Related reference:

“Response file for silent installation” on page 219
Specify parameters at the command line to customize and configure your product installation.

Related information:

"Tutorial: Getting started with WebSphere eXtreme Scale" on page 1
After you install WebSphere eXtreme Scale in a stand-alone environment, you can use the getting started sample application to verify your installation. The getting started sample application is an introduction to in-memory data grids. The getting started sample application is only included in full (client and server) installations of WebSphere eXtreme Scale.You can use the getting started sample application to verify the connection between your client installation and the appliance. The getting started sample application is an introduction to enterprise data grids.

Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server

You can install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in an environment in which WebSphere Application Server or WebSphere Application Server Network Deployment is installed. You can use the existing features of WebSphere Application Server or WebSphere Application Server Network Deployment to enhance your eXtreme Scale applications.

Before you begin

- Install WebSphere Application Server or WebSphere Application Server Network Deployment. See Installing your application serving environment for more information.
- Based on what version you install, Version 6.1 or Version 7.0, apply the latest fix pack for WebSphere Application Server or WebSphere Application Server Network Deployment to update your product level. See the Latest fix packs for WebSphere Application Server for more information.
- Verify that the target installation directory does not contain an existing installation of WebSphere eXtreme Scale or WebSphere eXtreme Scale Client.
- Stop all processes that are running in your WebSphere Application Server or WebSphere Application Server Network Deployment environment. See Command-line utilities for more information about the **`stopManager`**, **`stopNode`**, and **`stopServer`** commands.

  CAUTION:
  Ensure that any running processes are stopped. If the running processes are not stopped, the installation proceeds, creating unpredictable results and leaving the installation in an undetermined state on some platforms.
- If you are installing the client only, you can either use the DVD to install the client or download the WebSphere eXtreme Scale Client for the specific platform from the downloads section on the Support site.

Important: When you install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client, it should be in the same directory in which you installed WebSphere Application Server. For example, if you installed WebSphere Application Server in `C:\`*was_root*, then you should also choose `C:`*was_root* as the target directory for your WebSphere eXtreme Scale or WebSphere eXtreme Scale Client installation.

About this task

Integrate eXtreme Scale with WebSphere Application Server or WebSphere Application Server Network Deployment to apply the features of eXtreme Scale to your Java Platform, Enterprise Edition applications. Java EE applications host data grids and access the data grids using a client connection.

Procedure

1. Use the wizard to complete the installation.
   - Run the following script to start the wizard for the WebSphere eXtreme Scale full installation. You can choose to install the client only or both the server and client:
     - Linux UNIX *dvd_root*/install
     - Windows *dvd_root*\install.bat
   - Run the following script to start the wizard for theWebSphere eXtreme Scale Client installation. The installation files are in the zip file that you download from the downloads section on the Support site:
     - Linux UNIX root/WXS_Client/install
     - Windows root\WXS_Client\install.bat

   Attention: If you use uniform naming conventions (UNC) to identify file paths in your installation command, the items you anticipate installing may not all be installed after the command runs. To avoid trouble, map the file path to a network drive. Run the **install** command against the mapped drive. Using a mapped network drive ensures that all the items are installed.
2. Follow the prompts in the wizard.

   The optional features panel lists the features from which you can choose to install. However, features cannot be added incrementally to the product environment after the product is installed. If you choose not to install a feature with the initial product installation, you must uninstall and reinstall the product to add the feature.

   The Profile augmentation panel lists existing profiles that you can select to augment with the features of eXtreme Scale. If you select existing profiles that are already in use, however, a warning panel is displayed. To continue with the installation, either stop the servers that are configured in the profiles, or click **Back** to remove the profiles from your selection.

Results

Windows If you are installing the WebSphere eXtreme Scale Client on Windows, you might see the following text in the results of the installation:

```
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.
```

If you see a failure with the `iscdeploy.sh` file, you can ignore the error. This error does not cause any problems.

What to do next

- If you are running WebSphere Application Server Version 6.1 or Version 7.0, you can use the Profile Management Tool plug-in or the **manageprofiles** command. For more information, see “Creating and augmenting profiles for WebSphere eXtreme Scale” on page 203.
- 7.1.1 Verify the installation. For more information, see “Verifying the installation” on page 227.

- Start configuring your WebSphere eXtreme Scale or WebSphere eXtreme Scale Client installation. For more information, see “Taking the first steps after installation” on page 228.

Related reference:

“Response file for silent installation” on page 219
Specify parameters at the command line to customize and configure your product installation.

Using the Installation Factory plug-in to create and install customized packages

Use the IBM Installation Factory plug-in for WebSphere eXtreme Scale to create a customized installation package (CIP) or an integrated installation package (IIP). A CIP contains a single product installation package and various optional assets. An IIP combines one or more installation packages into a single installation workflow that you design.

Before you begin

Before you create and install customized packages for eXtreme Scale, you must first download the following products:

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

About this task

Using the Installation Factory, you can create a CIP by combining a single product component with maintenance packages, customization scripts, and other files. When you create an IIP, you aggregate individual components, or installation packages, into a single installation package.

Build definition file:

A build definition file is an XML document that specifies how to build and install a customized installation package (CIP) or an integrated installation package (IIP). The IBM Installation Factory for WebSphere eXtreme Scale reads the package details of the build definition file to generate a CIP or an IIP.

Before you can create a CIP or an IIP, you must create a build definition file for each customized package. The build definition file describes which product components, or installation packages, to install, the location of the CIP or IIP, the maintenance packages to include, the installation scripts, and other files that you choose to include. You can also specify in the build definition file for the IIP the order in which the Installation Factory installs each installation package.

The Build definition wizard steps you through the process of creating a build definition file. You can also use the wizard to modify an existing build definition file. Each panel in the Build definition wizard prompts you for information about a customized package, such as the package identification, the installation location for the build definition, and the installation location for the customized package. All of this information is saved in the new build definition file, or modified and saved in an existing build definition file. For more information, see the CIP Build definition wizard panels and the IIP Build definition wizard panels.

To create only the build definition file, you can use the command-line interface tool to generate the customized package outside of the GUI. See "Silently installing a CIP or an IIP" on page 195 for more information.

Creating a build definition file and generating a CIP:

The IBM Installation Factory plug-in for WebSphere eXtreme Scale generates a customized installation package (CIP) according to the details that you specify in the build definition file. The build definition specifies the product package to install, the location of the CIP, the maintenance packages to include in the installation, the install script files, and any additional files to include in the CIP.

About this task

You can use the Build definition wizard to create a build definition file and generate a CIP.

Procedure

1. Run the following script from the *IF_HOME*/`bin` directory to start the Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`

   Click the **New Build Definition** icon.
2. Select the product to include in the build definition file, and click **Finish** to start the Build definition wizard.
3. Follow the prompts in the wizard.

   On the Install and Uninstall Scripts panel, click **Add Scripts...** to populate the table with any customized installation scripts. Type the location of the script files, and clear the check box to continue if an error message is displayed. The operation is stopped by default. Click **OK** to return to the panel.

Results

You created and customized the build definition file, and you generated the CIP if you chose to work in the connected mode.

If the Build definition wizard does not provide you with the option to generate the CIP from the build definition file, you can still generate it by running the `ifcli.sh|bat` script from the *IF_HOME*/`bin` directory.

What to do next

Install the CIP. See "Installing a CIP" for more information.

Installing a CIP:

Simplify the product installation process by installing a customized installation package (CIP). A CIP is a single product installation image that can include one or more maintenance packages, configuration scripts, and other files.

Before you begin

Before you can install a CIP, you must create a build definition file to specify what options to include in the CIP. See "Creating a build definition file and generating a CIP" on page 189 for more information.

About this task

A CIP combines and installs a single product component with maintenance packages, customization scripts, and other files.

Procedure

1. Stop all processes that are running on the workstation you are preparing for installation. To stop the deployment manager, run the following script:
   - Linux UNIX `profile_root/bin/stopManager.sh`
   - Windows `profile_root\bin\stopManager.bat`

   To stop the nodes, run the following script:
   - Linux UNIX `profile_root/bin/stopNode.sh`
   - Windows `profile_root\bin\stopNode.bat`
2. Run the following script to start the installation:
   - Linux UNIX `CIP_home/bin/install`
   - Windows `CIP_home\bin\install.bat`
3. Follow the prompts in the wizard to complete the installation.

   The optional features panel lists the features from which you can choose to install. However, features cannot be added incrementally to the product environment after the product is installed. If you choose not to install a feature with the initial product installation, you must uninstall and reinstall the product to add the feature.

   The Profile augmentation panel lists existing profiles that you can select to augment with the features of eXtreme Scale. If you select existing profiles that are already in use, however, a warning panel is displayed. To continue with the installation, either stop the servers that are configured in the profiles, or click **Back** to remove the profiles from your selection.

Results

You successfully installed the CIP.

What to do next

If you are running WebSphere Application Server Version 6.1 or Version 7.0, you can use the Profile Management Tool plug-in or the **manageprofiles** command to create and augment profiles. See "Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203 for more information.

If you augmented profiles for eXtreme Scale during the installation process, you can deploy applications, start a catalog service, and start the containers in your WebSphere Application Server environment. See "Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297 for more information.

Installing a CIP to apply maintenance to an existing product installation:

You can apply maintenance packages to an existing product installation by installing a customized installation package (CIP). The process of applying maintenance to an existing installation with a CIP is commonly referred to as a *slip installation*.

Before you begin

Create a build definition file to specify what options to include in the CIP. See "Creating a build definition file and generating a CIP" on page 189 for more information.

About this task

When applying maintenance with a CIP that contains a refresh pack, a fix pack, or both, all previously installed authorized program analysis reports (APAR) are uninstalled by the wizard. If the CIP is at the same level as the product, previously installed APARs remain only if they are packaged in the CIP. To successfully apply maintenance to an existing installation, you must include the installed features in the CIP.

Procedure

1. Stop all processes that are running on the workstation you are preparing for installation. To stop the deployment manager, run the following script:
   - Linux UNIX `profile_root/bin/stopManager.sh`
   - Windows `profile_root\bin\stopManager.bat`

   To stop the nodes, run the following script:
   - Linux UNIX `profile_root\bin\stopNode.sh`
   - Windows `profile_root\bin\stopNode.bat`
2. Run the following script to start the installation:
   - Linux UNIX `CIP_home/bin/install`
   - Windows `CIP_home\bin\install.bat`
3. Follow the prompts in the wizard to complete the installation.

   The installation preview summary lists the resulting product version and any applicable features and interim fixes. Next, the wizard successfully applies the maintenance, and updates the features of the product.

Results

The product binary files are copied to the *was_root*`/properties/version/nif/backup` directory. You can use the IBM Update Installer to uninstall the update and restore your workstation. See "Uninstalling CIP updates from an existing product installation" for more information.

Uninstalling CIP updates from an existing product installation:

You can remove CIP updates from an existing product installation without removing the entire product. Use the IBM Update Installer Version 7.0.0.4 to uninstall any CIP updates. This task is also referred to as a *slip uninstallation*.

Before you begin

You must have at least one existing copy of the product installed on the system.

Procedure

1. Download Version 7.0.0.4 of the Update Installer from the following FTP site:

   ftp://ftp.software.ibm.com/software/websphere/cw/process_server/FEP/UPDI/7004
2. Install the Update Installer. See Installing the Update Installer for WebSphere Software in the WebSphere Application Server Information Center for more information.
3. Uninstall any fix pack, refresh pack, or interim fix that you added to your environment after you installed the CIP.
4. Uninstall any interim fixes that you included in the slip installation. This process is the same as uninstalling a single fix pack or refresh pack. However, the maintenance that was included in the CIP is now included in a single operation.
5. Uninstall the CIP by using the Update Installer. The maintenance levels return to the pre-update state, and the CIP is denoted by the CIP identifier that is added as a prefix to its file name. The following example shows how a CIP is displayed differently than other regular maintenance packages on the maintenance package selection panel:

   CIP

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

Results

You successfully removed the CIP updates from an existing product installation.

Creating a build definition file and generating an IIP:

The IBM Installation Factory plug-in for WebSphere eXtreme Scale generates an IIP based on the properties that the build definition file provides. The build definition file contains information such as which installation packages to include in the IIP, the order in which the Installation Factory installs each package, and the location of the IIP.

About this task

You can use the Build definition wizard to create a build definition file and generate an IIP.

Procedure

1. Run the following script from the *IF_HOME*/bin directory to start the Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Click the **Create New Integrated Installation Package** icon to start the Build definition wizard.
3. Follow the prompts in the wizard.
   a. On the Construct the IIP panel, select a supported installation package from the list, and click **Add Installer** to add the installation package to the IIP. A

panel that displays the package name, the package identifier, and the package properties is displayed. To view specific information about the selected package, click **View Installation Package Information**. Click **Modify** to enter the directory path to the installation package for each operating system. If you are currently adding an installation package for WebSphere Extended Deployment, select the checkbox, which provides you with the option to use the same package for all supported operating systems. Click **OK** and return to the Construct the IIP panel. An invocation is created by default.

- To modify the directory path to an installation package, select the package from the Installation packages used in the IIP list, and click **Modify**.
- To modify an invocation, select the invocation, and click **Modify**. Specify the default installation location for the invocation on each operating system. Specify the location to the response file if you select a silent installation as the default installation mode.
- Click **Add Invocation** to add an invocation contribution to the installation package. A panel from which you can specify properties for the invocation is displayed.
- Click **Remove** to remove installation packages or invocations.

4. Review the summary of your selections, select the **Save build definition file and generate integrated installation package** option, and click **Finish**.

   Alternatively, you can save the build definition file without generating the IIP. With this option, you actually generate the IIP outside of the wizard by running the `ifcli.bat | ifcli.sh` script from the *IF_home*/bin/ directory.

Results

You created and customized the build definition file for an IIP.

What to do next

Install the IIP.

Installing an IIP:

Use the IBM Installation Factory plug-in for WebSphere eXtreme Scale to install an integrated installation package (IIP). An IIP combines one or more installation packages into a single workflow that you design.

Before you begin

Before you can install a CIP, you must create a build definition file to specify what options to include in the CIP. See “Creating a build definition file and generating an IIP” on page 192 for more information.

About this task

An IIP can include one or more generally available installation packages, one or more CIPs, and other optional files and directories. By installing an IIP, you aggregate multiple installation packages, or *contributions*, into a single package, and you then install the contributions in a specific order to complete an end-to-end installation.

Procedure

1. Run the following script to start the wizard:
   - Linux UNIX *IIP_home*`/bin/install`
   - Windows *IIP_home*`\bin\install.bat`
2. Click **About** on the Welcome panel to view the details of the IIP, such as the package identifier, the supported operating systems, and the included installation packages.

   Optional: To modify the installation options for each package, click **Modify**.

   Optional: Two **View Log** buttons are displayed on the wizard panel. To view the log of each package, click the **View Log** button that is displayed next to the table that lists the installation packages. To view the overall log details of the IIP, click the **View Log** button that is displayed next to the status information.
3. Select the installation packages to run, and click **Install**. A list of all the contributions in the order of invocation that the IIP contains is displayed. To designate which contribution invocations should not be run during the installation, clear the checkbox located next to the **Installation name** field.

Results

You successfully installed an IIP.

Modifying an existing build definition file for an IIP:

You can edit or add to the properties of an IIP to further customize the installation.

About this task

To change the properties of an IIP, modify the existing build definition file.

Procedure

1. Run the following script from the *IF_HOME*`/bin` directory to start the Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Click the **Open Build Definition** icon, and select the build definition file that you want to modify.
3. Select the specific properties of the IIP that you want to modify. The following list contains the possible modifications that you can make:
   - Change your current mode selection. In connected mode, you create the build definition for use, and optionally generate the IIP, from your current workstation. In disconnected mode, you create the build definition file for use on another workstation.
   - Add or remove the existing operating systems that the IIP supports.
   - Edit the existing identifier and version for the IIP.
   - Edit the target location for the build definition file.
   - Edit the target location for the IIP.
   - Change whether to display an installation wizard for the IIP. The wizard provides information about the IIP and the installation options when the IIP runs.

- Add, remove, and edit the installation packages that are contained in the IIP.

  Important: If you added a supported operating system and you have not updated the properties of the installation package in the IIP, you receive a warning message stating that the selected contributions do not contain installation packages that are identified for all of the operating systems that the IIP supports. Click **Yes** to continue, or click **No** to edit the installation package.

4. Review the summary of your selections, select **Save build definition file and generate integrated installation package**, and click **Finish**.

Silently installing a CIP or an IIP:

You can silently install a customized installation package (CIP) or an integrated installation package (IIP) for the product by using either a fully-qualified response file, which you configure specifically to your needs, or parameters that you pass to the command line.

Before you begin

Create the build definition file for the CIP or IIP. See "Creating a build definition file and generating a CIP" on page 189 for more information.

About this task

A silent installation uses the same installation program that the graphical user interface (GUI) version uses. However, instead of displaying a wizard interface, the silent installation reads all of your responses from a file that you customize, or from parameters that you pass to the command line. If you are silently installing an IIP, you can invoke a contribution with a combination of options that you specify directly on the command line, as well as options that you specify in a response file. However, any contribution options that you pass to the command line causes the IIP installer to ignore all of the options that are specified in a specific contribution's response file. See the detailed Installing an IIP silently for more information.

Note: You must specify the fully-qualified response file name. Specifying the relative path causes the installation to fail with no indication that an error occurred.

Procedure

1. Optional: If you choose to install the CIP or IIP using a response file, first customize the file.
   a. Copy the response file, `wxssetup.response.txt`, from the product DVD to your disk drive.
   b. Open and edit the response file in the text editor of your choice. The file includes comments to assist the configuration process and must include these parameters:
      - The license agreement
      - The location of the product installation

      Tip: The installer uses the location that you select for your installation to determine where your WebSphere Application Server instance is installed. If you install on a node with multiple WebSphere Application Server instances, clearly define your location.

c. Run the following script to start your customized response file.

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

2. Optional: If you choose to install the CIP or IIP by passing certain parameters to the command line, run the following script to start the installation:

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

   where *install_location* is the location of your existing WebSphere Application Server installation.

3. Review the resulting logs for errors or an installation failure.

Results

You silently installed the CIP or IIP.

What to do next

If you are running WebSphere Application Server Version 6.1 or Version 7.0, you can use the Profile Management Tool plug-in or the **`manageprofiles`** command to create and augment profiles.

If you augmented profiles for eXtreme Scale during the installation process, you can deploy applications, start a catalog service, and start the containers in your WebSphere Application Server environment. See "Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297 for more information.

Related reference:
"Response file for silent installation" on page 219
Specify parameters at the command line to customize and configure your product installation.

`wxssetup.response.txt` *file:*

You can use a fully qualified response file to install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client silently.

CAUTION:
Do not add trailing slashes, such as / or \, to the end of the installation location paths. These paths are specified with the installLocation attribute. Adding a slash to the end of the installation location can cause the installation to fail. For example, the following path would cause the installation to fail:

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

The path should be specified as:

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

Response file for WebSphere eXtreme Scale full installation

```
################################################################################
#
# IBM WebSphere eXtreme Scale V7.1.1 InstallShield Options File
#
# Wizard name: Install
# Wizard source: setup.jar
#
# This file can be used to configure Install with the options specified below
# when the wizard is run with the "-options" command line option. Read each
# setting's documentation for information on how to change its value.
# Please enclose all values within a single pair of double quotes.
#
# A common use of an options file is to run the wizard in silent mode. This lets
# the options file author specify wizard settings without having to run the
# wizard in graphical or console mode. To use this options file for silent mode
# execution, use the following command line arguments when running the wizard:
#
#    -options "D:\installImage\WXS\wxssetup.response" -silent
#
# Note that the fully qualified response file name must be used.
#
################################################################################

################################################################################
#
# License Acceptance
#
# Valid Values:
#  true - Accepts the license. Will install the product.
#  false - Declines the license. Install will not occur.
#
# If no install occurs, this will be logged to a temporary log file in the
# user's temporary directory.
#
# By changing the silentInstallLicenseAcceptance property in this response file
# to "true", you agree that you have reviewed and agree to the terms of the
# IBM International Program License Agreement accompanying this program, which
# is located at CD_ROOT\XD\wxs.primary.pak\repository\legal.xs\license.xs.  If
# you do not agree to these terms, do not change the value or otherwise
# download, install, copy, access, or use the program and promptly return the
# program and proof of entitlement to the party from whom you acquired it to
# obtain a refund of the amount you paid.
#
-OPT silentInstallLicenseAcceptance="false"


################################################################################
# Non-blocking Prerequisite Checking
#
# If you want to disable non-blocking prerequisite checking, uncomment
# the following line. This will notify the installer to continue with
# the installation and log the warnings even though the prerequisite checking
# has failed.
```

```
#
#-OPT disableNonBlockingPrereqChecking="true"


################################################################################
#
# Install Location
#
# The install location of the product. Specify a valid directory into which the
# product should be installed. If the directory contains spaces, enclose it in
# double-quotes as shown in the Windows example below. Note that spaces in the
# install location is only supported on Windows operating systems. Maximum path
# length is 60 characters for Windows.
#
# Below is the list of default install locations for each supported operating
# system when you're installing as a root user.  By default, in this response
# file, the Windows install location is used.  If you want to use the default
# install location for another operating system, uncomment the appropriate
# default install location entry (by removing '#') and then comment out
# (by adding '#') the Windows operating system entry below.
#
# The install location is used to determine if WebSphere eXtreme Scale should
# be installed as a stand-alone deployment or if it should be integrated with
# an existing WebSphere Application Server installation.
#
# If the location specified is an existing WebSphere Application Server or
# WebSphere Network Deployment installation, then eXtreme Scale is integrated
# with the exising WebSphere Application Server.  If the location specified is
# a new or empty directory, then WebSphere eXtreme Scale is installed as a
# stand-alone deployment.
#
# Note: If the install location specified contains a previous installation of
# WebSphere eXtreme Scale, WebSphere eXtended Deployment DataGrid or
# ObjectGrid, the installation will fail.
#
# AIX Default Install Location:
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# HP-UX, Solaris or Linux Default Install Location:
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Windows Default Install Location:
#
-OPT installLocation="C:\Program Files\IBM\WebSphere\eXtremeScale"

#
# If you are installing as a non-root user on Unix or a non-administrator on
# Windows, the following default install locations are suggested. Be sure you
# have write permission for the install location chosen.
#
# AIX Default Install Location:
#
#  -OPT installLocation="<user's home>/IBM/WebSphere/eXtremeScale"
#
# HP-UX, Solaris or Linux Default Install Location:
#
#  -OPT installLocation="<user's home>/IBM/WebSphere/eXtremeScale"
#
# Windows Default Install Location:
#
# -OPT installLocation="C:\IBM\WebSphere\eXtremeScale"


################################################################################
```

```
#  Optional Features Installation
#
# Specify which of the optional features you wish to install by setting each
# desired feature to "true".  Set any optional features you do not want to
# install to "false".
#
# The options selectServer, selectClient, selectPF, and selectXSStreamQuery are
# only valid when the installLocation option above contains an installation of
# WebSphere Application Server.  The options are ignored on an WebSphere eXtreme
# Scale standalone installation.
#
# On the WebSphere eXtreme Scale standalone installation, the eXtreme Scale
# server and client are automatically installed.  The feature options for the
# eXtreme Scale standalone installation are selectXSConsoleOther and
# selectXSStreamQueryOther.

#
# This option, when selected, installs the components that are required to run
# WebSphere eXtreme Scale servers and the eXtreme Scale dynamic cache service
# provider. If this option is selected, then the WebSphere eXtreme Scale Client
# must also be selected by being uncommented and set to a value of "true".
# Otherwise, silent install will FAIL.
#
-OPT selectServer="true"

#
# This option, when selected, installs the components that are required to run
# WebSphere eXtreme Scale client applications. If the Server option is selected
# above, then this option must also be selected by being uncommented and set to
# a value of "true" or silent install will FAIL.
#
-OPT selectClient="true"

#
# This option, when selected, installs the components that are required to run
# the WebSphere eXtreme Scale Console. If this option is selected, the install
# location specified above must be a new or empty directory because the console
# option is only valid for WebSphere eXtreme Scale stand-alone deployment. To
# install this option, the following option line must be uncommented and and set
# to a value of "true".
#-OPT selectXSConsoleOther="false"

#
# The following options, if selected will install DEPRECATED functionality.
#
# This option selects WebSphere Partition Facility for installation.
# This functionality is DEPRECATED.  To install this option, the following
# option line must be uncommented and set to a value of "true".
#
#-OPT selectPF="false"

#
# This option selects WebSphere eXtreme Scale StreamQuery for WAS for
# installation.  This functionality is DEPRECATED.  To install this option,
# the following option line must be uncommented and set to a value of "true".
# If this option is selected, then the WebSphere eXtreme Scale Client
# must also be selected by being uncommented and set to a value of "true".
# Otherwise, silent install will FAIL.
#
#-OPT selectXSStreamQuery="false"

#
# This option selects WebSphere eXtreme Scale StreamQuery for J2SE for
# installation.  This functionality is DEPRECATED.  To install this option,
# the following option line must be uncommented and set to a value of "true".
# If this option is selected, then the WebSphere eXtreme Scale Client
# must also be selected by being uncommented and set to a value of "true".
# Otherwise, silent install will FAIL.
#
#-OPT selectXSStreamQueryOther="false"


################################################################################
#  Profile list for augmentation
#
# Specify which of the existing profiles you wish to augment or comment the
# line to augment every existing profiles detected by the intallation.
#
# To specify multiple profiles, use comma to separate different profile names.
# For example, "AppSrv01,Dmgr01,Custom01". The list must not contain any spaces.
#
-OPT profileAugmentList=""


################################################################################
# Tracing Control
#
# The trace output format can be controlled via the option
# -OPT traceFormat=ALL
#
# The choices for the format are 'text' and 'XML'. By default, both formats will
# be produced, in two different trace files.
#
# If only one format is required, use the traceFormat option to specify which
# one, as follows:
#
# Valid Values:
#
#  text -  Lines in the trace file will be in a plain text format for easy
#          readability.
#  XML  -  Lines in the trace file will be in the standard Java logging XML
#          format which can be viewed using any text or XML editor or using the
#          Chainsaw tool from Apache at the following URL:
#          (http://logging.apache.org/log4j/docs/chainsaw.html).
#
# The amount of trace info captured can be controlled using the option:
# -OPT traceLevel=INFO
#
# Valid Values:
#
# Trace    Numerical
# Level      Level    Description
# -------  ---------  ---------------------------------------------------------
# OFF        0        No trace file is produced
# SEVERE     1        Only severe errors are output to trace file
# WARNING    2        Messages regarding non-fatal exceptions and warnings are
#                     added to trace file
# INFO       3        Informational messages are added to the trace file
#                     (this is the default trace level)
# CONFIG     4        Configuration related messages are added to the trace file
# FINE       5        Tracing method calls for public methods
# FINER      6        Tracing method calls for non public methods except
#                     getters and setters
# FINEST     7        Trace all method calls, trace entry/exit will include
#                     parameters and return value
```

Response file for WebSphere eXtreme Scale Client installation

```
################################################################################
#
# IBM WebSphere eXtreme Scale Client V7.1.1 InstallShield Options File
#
# Wizard name: Install
# Wizard source: setup.jar
```

```
#
# This file can be used to configure Install with the options specified below
# when the wizard is run with the "-options" command line option. Read each
# setting's documentation for information on how to change its value.
# Please enclose all values within a single pair of double quotes.
#
# A common use of an options file is to run the wizard in silent mode. This lets
# the options file author specify wizard settings without having to run the
# wizard in graphical or console mode. To use this options file for silent mode
# execution, use the following command line arguments when running the wizard:
#
#    -options "D:\installImage\WXS_Client\wxssetup.response" -silent
#
# Note that the fully qualified response file name must be used.
#
################################################################################

################################################################################
#
# License Acceptance
#
# Valid Values:
#  true - Accepts the license. Will install the product.
#  false - Declines the license. Install will not occur.
#
# If no install occurs, this will be logged to a temporary log file in the
# user's temporary directory.
#
# By changing the silentInstallLicenseAcceptance property in this response file
# to "true", you agree that you have reviewed and agree to the terms of the
# IBM International Program License Agreement accompanying this program, which
# is located at
# CD_ROOT\WXS_Cleint\wxs.client.primary.pak\repository\legal.xs.client\license.xs.
# If you do not agree to these terms, do not change the value or otherwise
# download, install, copy, access, or use the program and promptly return the
# program and proof of entitlement to the party from whom you acquired it to
# obtain a refund of the amount you paid.
#
-OPT silentInstallLicenseAcceptance="false"


################################################################################
# Non-blocking Prerequisite Checking
#
# If you want to disable non-blocking prerequisite checking, uncomment
# the following line. This will notify the installer to continue with
# the installation and log the warnings even though the prerequisite checking
# has failed.
#
#-OPT disableNonBlockingPrereqChecking="true"


################################################################################
#
# Install Location
#
# The install location of the product. Specify a valid directory into which the
# product should be installed. If the directory contains spaces, enclose it in
# double-quotes as shown in the Windows example below. Note that spaces in the
# install location is only supported on Windows operating systems. Maximum path
# length is 60 characters for Windows.
#
# Below is the list of default install locations for each supported operating
# system when you're installing as a root user.  By default, in this response
# file, the Windows install location is used.  If you want to use the default
# install location for another operating system, uncomment the appropriate
# default install location entry (by removing '#') and then comment out
# (by adding '#') the Windows operating system entry below.
#
# The install location is used to determine if WebSphere eXtreme Scale should
# be installed as a stand-alone deployment or if it should be integrated with
# an existing WebSphere Application Server installation.
#
# If the location specified is an existing WebSphere Application Server or
# WebSphere Network Deployment installation, then eXtreme Scale is integrated
# with the exising WebSphere Application Server.  If the location specified is
# a new or empty directory, then WebSphere eXtreme Scale is installed as a
# stand-alone deployment.
#
# Note: If the install location specified contains a previous installation of
# WebSphere eXtreme Scale, WebSphere eXtended Deployment DataGrid or
# ObjectGrid, the installation will fail.
#
# AIX Default Install Location:
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# HP-UX, Solaris or Linux Default Install Location:
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Windows Default Install Location:
#
-OPT installLocation="C:\Program Files\IBM\WebSphere\eXtremeScale"

#
# If you are installing as a non-root user on Unix or a non-administrator on
# Windows, the following default install locations are suggested. Be sure you
# have write permission for the install location chosen.
#
# AIX Default Install Location:
#
#  -OPT installLocation="<user's home>/IBM/WebSphere/eXtremeScale"
#
# HP-UX, Solaris or Linux Default Install Location:
#
#  -OPT installLocation="<user's home>/IBM/WebSphere/eXtremeScale"
#
# Windows Default Install Location:
#
# -OPT installLocation="C:\IBM\WebSphere\eXtremeScale"


################################################################################
#  Profile list for augmentation
#
# Specify which of the existing profiles you wish to augment or comment the
# line to augment every existing profiles detected by the intallation.
#
# To specify multiple profiles, use comma to separate different profile names.
# For example, "AppSrv01,Dmgr01,Custom01". The list must not contain any spaces.
#
-OPT profileAugmentList=""


################################################################################
# Tracing Control
#
# The trace output format can be controlled via the option
# -OPT traceFormat=ALL
#
# The choices for the format are 'text' and 'XML'. By default, both formats will
# be produced, in two different trace files.
```

```
#
# If only one format is required, use the traceFormat option to specify which
# one, as follows:
#
# Valid Values:
#
#  text -  Lines in the trace file will be in a plain text format for easy
#          readability.
#  XML  -  Lines in the trace file will be in the standard Java logging XML
#          format which can be viewed using any text or XML editor or using the
#          Chainsaw tool from Apache at the following URL:
#          (http://logging.apache.org/log4j/docs/chainsaw.html).
#
# The amount of trace info captured can be controlled using the option:
# -OPT traceLevel=INFO
#
# Valid Values:
#
# Trace    Numerical
# Level      Level    Description
# -------  ---------  ---------------------------------------------------------
# OFF        0        No trace file is produced
# SEVERE     1        Only severe errors are output to trace file
# WARNING    2        Messages regarding non-fatal exceptions and warnings are
#                     added to trace file
# INFO       3        Informational messages are added to the trace file
#                     (this is the default trace level)
# CONFIG     4        Configuration related messages are added to the trace file
# FINE       5        Tracing method calls for public methods
# FINER      6        Tracing method calls for non public methods except
#                     getters and setters
# FINEST     7        Trace all method calls, trace entry/exit will include
#                     parameters and return value
```

Creating and augmenting profiles for WebSphere eXtreme Scale

After you install the product, create unique types of profiles and augment existing profiles for WebSphere eXtreme Scale.

Before you begin

Install WebSphere eXtreme Scale. See "Installation overview" on page 171 for more information.

About this task

Augmenting profiles for use with WebSphere eXtreme Scale is optional, but is required in the following usage scenarios:

- To automatically start a catalog service or container in a WebSphere Application Server process. Without augmenting the server profiles, servers can only be started programmatically using the ServerFactory API or as separate processes with the 7.1.1 **startOgServer** script.
- To use Performance Monitoring Infrastructure (PMI) to monitor WebSphere eXtreme Scale metrics.
- To display the version of WebSphere eXtreme Scale in the WebSphere Application Server administrative console.

If you are running WebSphere eXtreme Scale within WebSphere Application Server Version 6.1 , you can use the Profile Management Tool plug-in or the **manageprofiles** command to create and augment profiles.

What to do next

Depending on which task you choose to complete, launch the First steps console for assistance with configuring and testing your product environment. The First steps console is in the following directory:

- Windows *wxs_install_root*`\firststeps\wxs\firststeps.bat`
- UNIX Linux *wxs_install_root*`/firststeps/wxs/firststeps.sh`

You can also create or augment additional profiles by repeating any of the preceding tasks.

Related reference:

"Response file for silent installation" on page 219
Specify parameters at the command line to customize and configure your product installation.

"**`manageprofiles`** command" on page 206
You can use the **`manageprofiles`** utility to create profiles with the WebSphere eXtreme Scale template, and augment and unaugment existing application server profiles with the eXtreme Scale augment templates. To use the features of the product, your environment must contain at least one profile augmented for the product.

Using the graphical user interface to create profiles:

Use the graphical user interface (GUI), which is provided by the Profile Management Tool plug-in, to create profiles for WebSphere eXtreme Scale. A profile is a set of files that define the runtime environment.

Before you begin

You cannot use the GUI to augment profiles in the following scenario:

- **64-bit installations of WebSphere Application Server**:

  The profile management tool does not exist for 64-bit installations of WebSphere Application Server. Use the **`manageprofiles`** script from the command line for these installations.

About this task

To use the product features, the Profile Management Tool plug-in enables the GUI to assist you in setting up profiles, such as a WebSphere Application Server profile, a deployment manager profile, a cell profile, and a custom profile. You can augment profiles during or after the installation of WebSphere eXtreme Scale.

Procedure

Use the Profile Management Tool GUI to create profiles. Choose one of the following options to start the wizard:

- Select **Profile Management Tool** from the First steps console.
- Access the Profile Management Tool from the **Start** menu.
- Run the `./pmt.sh|bat` script from the *`install_root`*`/bin/ProfileManagement` directory.

What to do next

You can create additional profiles or augment existing profiles. To restart the Profile Management tool, run the **`./pmt.sh|bat`** command from the *`was_root`*`/bin/ProfileManagement` directory, or select **Profile Management Tool** in the First steps console.

Start a catalog service, start containers, and configure TCP ports in your WebSphere Application Server environment. See "Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297 for more information.

Related reference:

"**`manageprofiles`** command" on page 206
You can use the **`manageprofiles`** utility to create profiles with the WebSphere eXtreme Scale template, and augment and unaugment existing application server profiles with the eXtreme Scale augment templates. To use the features of the product, your environment must contain at least one profile augmented for the product.

Using the graphical user interface to augment profiles:

After you install the product, you can augment an existing profile to make it compatible with WebSphere eXtreme Scale.

About this task

When you augment an existing profile, you change the profile by applying a product-specific augmentation template. For example, WebSphere eXtreme Scale servers do not start automatically unless the server profile is augmented with the xs_augment template.

- Augment the profile with the xs_augment template if you installed the eXtreme Scale client or the client and server.
- Augment the profile with the pf_augment template only if you installed the partitioning facility.
- Apply both of the templates if your environment contains the eXtreme Scale client and the partitioning facility.

Procedure

Use the Profile Management Tool GUI to augment profiles for eXtreme Scale. Choose one of the following options to start the wizard:

- Select **Profile Management Tool** from the First steps console.
- Access the Profile Management Tool from the **Start** menu.
- Run the ./pmt.sh|bat script from the *`was_root`*`/bin/ProfileManagement` directory.

What to do next

You can augment additional profiles. To restart the Profile Management tool, run the **`./pmt.sh|bat`** command from the *`was_root`*`/bin/ProfileManagement` directory, or select **Profile Management Tool** in the First steps console.

Start a catalog service, start containers, and configure TCP ports in your WebSphere Application Server environment. See "Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297 for more information.

Related reference:

"**manageprofiles** command"
You can use the **manageprofiles** utility to create profiles with the WebSphere eXtreme Scale template, and augment and unaugment existing application server profiles with the eXtreme Scale augment templates. To use the features of the product, your environment must contain at least one profile augmented for the product.

manageprofiles command:

You can use the **manageprofiles** utility to create profiles with the WebSphere eXtreme Scale template, and augment and unaugment existing application server profiles with the eXtreme Scale augment templates. To use the features of the product, your environment must contain at least one profile augmented for the product.
- Before you can create and augment profiles, you must install eXtreme Scale . See 7.1.1 "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server" on page 186 for more information.

Purpose

The **manageprofiles** command creates the runtime environment for a product process in a set of files called a profile. The profile defines the runtime environment. You can perform the following actions with the **manageprofiles** command:
- Create and augment a deployment manager profile
- Create and augment a custom profile
- Create and augment stand-alone application server profile
- Create and augment a cell profile
- Unaugment any type of profile

When you augment an existing profile, you change the profile by applying a product-specific augmentation template.
- Augment the profile with the xs_augment template if you installed the eXtreme Scale client or both the client and server.
- Augment the profile with the pf_augment template if you installed only the partitioning facility.
- Apply both templates if your environment contains the eXtreme Scale client and the partitioning facility.

Location

The command file is in the `install_root/bin` directory.

Usage

For detailed help, use the **-help** parameter:
```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/dmgr -help
```

In the following sections, each task that you can perform using the **manageprofiles** command, along with a list of required parameters, is described. For details on the optional parameters to specify for each task, see the **manageprofiles** command in the WebSphere Application Server Information Center.

Create a deployment manager profile

You can use the **`manageprofiles`** command to create a deployment manager profile. The deployment manager administers the application servers that are federated into the cell.

Parameters

`-create`
: Creates a profile. (Required)

`-templatePath` *`template_path`*
: Specifies the file path to the template. (Required)

  Use the following format:

  `-templatePath` *`install_root`*`/profileTemplates/`*`template_type`*`/dmgr`

  where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:

  ```
  ./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/dmgr
  ```
- Using the pf_augment template:

  ```
  ./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/dmgr
  ```

Create a custom profile

You can use the **`manageprofiles`** command to create a custom profile. A custom profile is an empty node that you customize through the deployment manager to include application servers, clusters, or other Java processes.

Parameters

`-create`
: Creates a profile. (Required)

`-templatePath` *`template_path`*
: Specifies the file path to the template. (Required)

  Use the following format:

  `-templatePath` *`install_root`*`/profileTemplates/`*`template_type`*`/managed`

  where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:

  ```
  ./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/managed
  ```
- Using the pf_augment template:

  ```
  ./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/managed
  ```

Create a stand-alone application server profile

You can use the **`manageprofiles`** command to create a stand-alone application server profile.

Parameters

-create
Creates a profile. (Required)

-templatePath *template_path*
Specifies the file path to the template. (Required)

Use the following format:

```
-templatePath install_root/profileTemplates/template_type/default
```

where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/xs_augment/default
```

- Using the pf_augment template:

```
./manageprofiles.sh|bat -create -templatePath install_root/profileTemplates/pf_augment/default
```

Create a cell profile

You can use the **manageprofiles** command to create a cell profile, which consists of a deployment manager and an application server.

Parameters

Specify the following parameters in the deployment manager template:

-create
Creates a profile. (Required)

-templatePath *template_path*
Specifies the file path to the template. (Required)

Use the following format:

```
-templatePath install_root/profileTemplates/template_type/cell/dmgr
```

where *template_type* is `xs_augment` or `pf_augment`.

Specify the following parameters with the application server template:

-create
Creates a profile. (Required)

-templatePath *template_path*
Specifies the file path to the template. (Required)

Use the following format:

```
-templatePath install_root/profileTemplates/template_type/cell/default
```

where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:

```
./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
```

- Using the pf_augment template:

```
./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
```

Augment a deployment manager profile

You can use the **manageprofiles** command to augment a deployment manager profile.

Parameters

-augment
: Augments the existing profile. (Required)

-profileName
: Specifies the name of the profile. (Required)

-templatePath *template_path*
: Specifies the path to the template files that are located in the installation root directory. (Required)

  Use the following format:

  ```
  -templatePath install_root/profileTemplates/template_type/dmgr
  ```

  where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:

```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/xs_augment/dmgr
```

- Using the pf_augment template:

```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/pf_augment/dmgr
```

Augment a custom profile

You can use the **manageprofiles** command to augment a custom profile.

Parameters

-augment
: Augments the existing profile. (Required)

-profileName
: Specifies the name of the profile. (Required)

-templatePath *template_path*
: Specifies the path to the template files that are located in the installation root directory. (Required)

  Use the following format:

  ```
  -templatePath install_root/profileTemplates/template_type/managed
  ```

where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:
```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/xs_augment/managed
```
- Using the pf_augment template:
```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/pf_augment/managed
```

Augment a stand-alone application server profile

You can use the **`manageprofiles`** command to augment a stand-alone application server profile.

Parameters

`-augment`
: Augments the existing profile. (Required)

`-profileName`
: Specifies the name of the profile. (Required)

`-templatePath` *`template_path`*
: Specifies the path to the template files that are located in the installation root directory. (Required)

  Use the following format:
```
-templatePath install_root/profileTemplates/template_type/default
```

  where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:
```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/xs_augment/default
```
- Using the pf_augment template:
```
./manageprofiles.sh|bat -augment -profileName profile01
-templatePath install_root/profileTemplates/pf_augment/default
```

Augment a cell profile

You can use the **`manageprofiles`** command to augment a cell profile.

Parameters

Specify the following parameters for the deployment manager profile:

`-augment`
: Augments the existing profile. (Required)

`-profileName`
: Specifies the name of the profile. (Required)

`-templatePath` *`template_path`*
: Specifies the path to the template files that are located in the installation root directory. (Required)

  Use the following format:

```
-templatePath install_root/profileTemplates/template_type/cell/dmgr
```

where *template_type* is `xs_augment` or `pf_augment`.

Specify the following parameters for the application server profile:

-augment
Augments the existing profile. (Required)

-profileName
Specifies the name of the profile. (Required)

-templatePath *template_path*
Specifies the path to the template files that are located in the installation root directory. (Required)

Use the following format:

```
-templatePath install_root/profileTemplates/template_type/cell/default
```

where *template_type* is `xs_augment` or `pf_augment`.

Example

- Using the xs_augment template:

```
./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
```

- Using the pf_augment template:

```
./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
```

Unaugment a profile

To unaugment a profile, specify the **-ignoreStack** parameter with the **-templatePath** parameter in addition to specifying the required **-unaugment** and **-profileName** parameters.

Parameters

-unaugment
Unaugments a previously augmented profile. (Required)

-profileName
Specifies the name of the profile. The parameter is issued by default if no values are specified. (Required)

-templatePath *template_path*
Specifies the path to the template files that are located in the installation root directory. (Optional)

Use the following format:

```
-templatePath install_root/profileTemplates/template_type/profile_type
```

where *template_type* is `xs_augment` or `pf_augment` and *profile_type* is one of four profile types:

- `dmgr`: deployment manager profile
- `managed`: custom profile
- `default`: stand-alone application server profile

- `cell`: cell profile

-ignoreStack
: Used with the **-templatePath** parameter to unaugment a particular profile that has been augmented. (Optional)

Example

- Using the xs_augment template:
```
./manageprofiles.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath install_root/profileTemplates/xs_augment/profile_type
```
- Using the pf_augment template:
```
./manageprofiles.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath install_root/profileTemplates/pf_augment/profile_type
```

Related tasks:

"Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203
After you install the product, create unique types of profiles and augment existing profiles for WebSphere eXtreme Scale.

"Using the graphical user interface to create profiles" on page 204
Use the graphical user interface (GUI), which is provided by the Profile Management Tool plug-in, to create profiles for WebSphere eXtreme Scale. A profile is a set of files that define the runtime environment.

"Using the graphical user interface to augment profiles" on page 205
After you install the product, you can augment an existing profile to make it compatible with WebSphere eXtreme Scale.

Non-root profiles:

Give a non-root user permissions for files and directories so that the non-root user can create a profile for the product. The non-root user can also augment a profile that was created by a root user, a different non-root user, or the same non-root user.

In a WebSphere Application Server environment, non-root (non-administrator) users are limited in being able to create and use profiles in their environment. Within the Profile Management tool plug-in, unique names and port values are disabled for non-root users. The non-root user must change the default field values in the Profile Management tool for the profile name, node name, cell name, and port assignments. Consider assigning non-root users a range of values for each of the fields. You can assign responsibility to the non-root users for adhering to their proper value ranges and for maintaining the integrity of their own definitions.

The term *installer* refers to either a root or non-root user. As an installer, you can grant non-root users permissions to create profiles and establish their own product environments. For example, a non-root user might create a product environment to test application deployment with a profile that the user owns. Specific tasks that you can complete to allow non-root profile creation include the following items:

- Creating a profile and assigning ownership of the profile directory to a non-root user so that the non-root user can start WebSphere Application Server for a specific profile.
- Granting write permission of the appropriate files and directories to a non-root user, which allows the non-root user to then create the profile. With this task, you can create a group for users who are authorized to create profiles, or give individual users the ability to create profiles.

- Installing maintenance packages for the product, which includes required services for existing profiles that are owned by a non- user. As the installer, you are the owner of any new files that the maintenance package creates.

For more information about creating profiles for non-root users, see Creating profiles for non-root users .

As an installer, you can also grant permissions for a non-root user to augment profiles. For example, a non-root user can augment a profile that is created by an installer, or augment a profile that they create. Follow the WebSphere Application Server Network Deployment non-root user augmentation process.

However, when a non-root user augments a profile that is created by the installer, the non-root user does not need to create the following files before augmentation. The following files were established during the profile creation process:
- *was_root*`/logs/manageprofiles.xml`
- *was_root*`/properties/fsdb.xml`
- *was_root*`/properties/profileRegistry.xml`

When a non-root user augments a profile that the user creates, the non-root user must modify the permissions for the documents that are located within the eXtreme Scale profile templates.

Attention: You can also use a non-root (non-administrator) profile for WebSphere eXtreme Scale in a stand-alone environment, one outside of WebSphere Application Server. You must change the owner of the ObjectGrid directory to the non-root profile. Then you can log in with that non-root profile and operate eXtreme Scale as you normally would for a root (administrator) profile.

Runtime files for WebSphere eXtreme Scale integrated with WebSphere Application Server

Java archive (JAR) files are included in the installation. You can see the JAR files that are included and the location to which they are installed.

Table 4. Runtime files for WebSphere eXtreme Scale. The following table lists the Java archive (JAR) files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
`wxsdynacache.jar`	Client and server	`lib`	The `wxsdynacache.jar` file contains the necessary classes to use with the dynamic cache provider.
`wsobjectgrid.jar`	Local and client	`lib`	The `wsobjectgrid.jar` contains the eXtreme Scale local, client, and server run times.
`ogagent.jar`	Local, client, and server	`lib`	The `ogagent.jar` file contains the runtime classes that are required to run the Java instrumentation agent that is used with the EntityManager API.
`ogsip.jar`	Server	`lib`	The `ogsip.jar` file contains the eXtreme Scale Session Initiation Protocol (SIP) session management runtime that is compatible with WebSphere Application Server Version 6.1.x and later.
`sessionobjectgrid.jar`	Client and server	`lib`	The `sessionobjectgrid.jar` file contains the eXtreme Scale HTTP session management runtime.
`sessionobjectgridsip.jar`	Server	`lib`	The `sessionobjectgridsip.jar` file contains the eXtreme Scale SIP session management runtime that is compatible with with WebSphere Application Server Version 6.1.x and later.
`wsogclient.jar`	Local and client	`lib`	The `wsogclient.jar` file installed when you use an environment that contains WebSphere Application Server Version6.1.x and later. This file contains only the local and client runtime environments.

Table 4. Runtime files for WebSphere eXtreme Scale (continued). The following table lists the Java archive (JAR) files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
wxssizeagent.jar	Local, client, and server	lib	The wxssizeagent.jar file is used to provide more accurate cache entry sizing information when using Java runtime environment (JRE) Version 1.5 or later.
oghibernate-cache.jar	Client and server	optionalLibraries/ObjectGrid	The oghibernate-cache.jar file contains the eXtreme Scale level 2 cache plug-in for JBoss Hibernate.
ogspring.jar	Local, client, and server	optionalLibraries/ObjectGrid	The ogspring.jar file contains support classes for the SpringSource Spring framework integration.
xsadmin.jar	Utility	optionalLibraries/ObjectGrid	The xsadmin.jar file contains the eXtreme Scale administration sample utility.
ibmcfw.jar ibmorb.jar ibmorbapi.jar	Client and server	optionalLibraries/ObjectGrid/endorsed	This set of files includes the Object Request Broker (ORB) runtime that is used for running applications in Java SE processes.
wxshyperic.jar	Utility	optionalLibraries/ObjectGrid/hyperic/lib	The WebSphere eXtreme Scale server detection plug-in for the SpringSource Hyperic monitoring agent.
restservice.ear	Client	optionalLibraries/ObjectGrid/restservice/lib	The restservice.ear file contains the eXtreme Scale REST data service application enterprise archive for WebSphere Application Server environments.
restservice.war	Client	optionalLibraries/ObjectGrid/restservice/lib	The restservice.war file contains the eXtreme Scale REST data service Web archive for application servers acquired from another vendor.
splicerlistener.jar	Utility	optionalLibraries/ObjectGrid/session/lib	The splicerlistener.jar file contains the splicer utility for the eXtreme Scale HTTP session manager filter.
splicer.jar	Utility	optionalLibraries/ObjectGrid/legacy/session/lib	The splicer.jar contains the Version 7.0 splicer utility for the eXtreme Scale HTTP session manager filter.
wxsra.rar	Utility	optionalLibraries/ObjectGrid/wxsra.rar	The wxsra.rar contains the eXtreme Scale resource adapter to connect to the grid using a connection factory.

Table 5. Runtime files for WebSphere eXtreme Scale Client. The following table lists the Java archive (JAR) files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
wxsdynacache.jar	Client and server	lib	The wxsdynacache.jar file contains the necessary classes to use with the dynamic cache provider.
ogagent.jar	Local, client, and server	lib	The ogagent.jar file contains the runtime classes that are required to run the Java instrumentation agent that is used with the EntityManager API.
ogsip.jar	Server	lib	The ogsip.jar file contains the eXtreme Scale Session Initiation Protocol (SIP) session management runtime that is compatible with WebSphere Application Server Version6.1.x and later.
sessionobjectgrid.jar	Client and server	lib	The sessionobjectgrid.jar file contains the eXtreme Scale HTTP session management runtime.
sessionobjectgridsip.jar	Server	lib	The sessionobjectgridsip.jar file contains the eXtreme Scale SIP session management runtime that is compatible with WebSphere Application Server Version6.1.x and later.
wsogclient.jar	Local and client	lib	The wsogclient.jar file installed when you use an environment that contains WebSphere Application Server Version6.1.x and later. This file contains only the local and client runtime environments.
wxssizeagent.jar	Local, client, and server	lib	The wxssizeagent.jar file is used to provide more accurate cache entry sizing information when using Java runtime environment (JRE) Version1.5 or later

Table 5. Runtime files for WebSphere eXtreme Scale Client (continued). The following table lists the Java archive (JAR) files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
`oghibernate-cache.jar`	Client and server	`optionalLibraries/ObjectGrid`	The `oghibernate-cache.jar` file contains the eXtreme Scale level 2 cache plug-in for JBoss Hibernate.
`ogspring.jar`	Local, client, and server	`optionalLibraries/ObjectGrid`	The `ogspring.jar` file contains support classes for the SpringSource Spring framework integration.
`xsadmin.jar`	Utility	`optionalLibraries/ObjectGrid`	The `xsadmin.jar` file contains the eXtreme Scale administration sample utility.

`ibmcfw.jar`
`ibmorb.jar`
`ibmorbapi.jar`
	Client and server	`optionalLibraries/ObjectGrid/endorsed`	This set of files includes the Object Request Broker (ORB) runtime that is used for running applications in Java SE processes.

`wxshyperic.jar`	Utility	`optionalLibraries/ObjectGrid/hyperic/lib`	The WebSphere eXtreme Scale server detection plug-in for the SpringSource Hyperic monitoring agent.
`restservice.ear`	Client	`optionalLibraries/ObjectGrid/restservice/lib`	The `restservice.ear` file contains the eXtreme Scale REST data service application enterprise archive for WebSphere Application Server environments.
`restservice.war`	Client	`optionalLibraries/ObjectGrid/restservice/lib`	The `restservice.war` file contains the eXtreme Scale REST data service Web archive for application servers acquired from another vendor.
`splicerlistener.jar`	Utility	`optionalLibraries/ObjectGrid/session/lib`	The `splicerlistener.jar` file contains the splicer utility for the eXtreme Scale HTTP session manager filter.
`splicer.jar`	Utility	`optionalLibraries/ObjectGrid/legacy/session/lib`	The `splicer.jar` contains the Version 7.0 splicer utility for the eXtreme Scale HTTP session manager filter.

Runtime files for WebSphere eXtreme Scale stand-alone installation

Java archive (JAR) files are included in the installation. You can see the JAR files that are included and the location to which they are installed.

Table 6. Runtime files for WebSphere eXtreme Scale full installation. WebSphere eXtreme Scale relies on ObjectGrid processes and related APIs. The following table lists the JAR files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
`wxsdynacache.jar`	Client and server	`dynacache/lib`	The `wxsdynacache.jar` file contains the necessary classes to use with the dynamic cache provider. The file is automatically included in the server runtime environment when you use the supplied scripts.
`wxshyperic.jar`	Utility	`hyperic/lib`	The WebSphere eXtreme Scale server detection plug-in for the SpringSource Hyperic monitoring agent.
`objectgrid.jar`	Local, client, and server	`lib`	The `objectgrid.jar` file is an OSGi bundle that is used by the server runtime environment of Java SE 1.5 and later. The file is automatically included in the server runtime environment when you use the supplied scripts.
`ogagent.jar`	Local, client, and server	`lib`	The `ogagent.jar` file contains the runtime classes that are required to run the Java instrumentation agent that is used with the EntityManager API.
`ogclient.jar`	Local and client	`lib`	The `ogclient.jar` file is an OSGi bundle that contains only the local and client runtime environments. You can use this file with Java SE 1.5 and later.
`ogspring.jar`	Local, client, and server	`lib`	The `ogspring.jar` file contains support classes for the SpringSource Spring framework integration.
`wsogclient.jar`	Local and client	`lib`	The `wsogclient.jar` file installed when you use an environment that contains WebSphere Application Server Version 6.1 and later. This file contains only the local and client runtime environments.

Table 6. Runtime files for WebSphere eXtreme Scale full installation (continued). WebSphere eXtreme Scale relies on ObjectGrid processes and related APIs. The following table lists the JAR files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
wxssizeagent.jar	Local, client, and server	lib	The wxssizeagent.jar file is used to provide more accurate cache entry sizing information when using Java runtime environment (JRE) Version 1.5 and later.
ibmcfw.jar ibmorb.jar ibmorbapi.jar	Client and server	lib/endorsed	This set of files includes the Object Request Broker (ORB) runtime that is used for running applications in Java SE processes.
restservice.ear	Client	restservice/lib	The restservice.ear file contains the eXtreme Scale REST data service application enterprise archive for WebSphere Application Server environments.
restservice.war	Client	restservice/lib	The restservice.war file contains the eXtreme Scale REST data service Web archive for application servers acquired from another vendor.
xsadmin.jar	Utility	samples	The xsadmin.jar file contains the eXtreme Scale administration sample utility.
sessionobjectgrid.jar	Client and server	session/lib	The sessionobjectgrid.jar file contains the eXtreme Scale HTTP session management runtime.
splicerlistener.jar	Utility	session/lib	The splicerlistener.jar file contains the splicer utility for the eXtreme Scale Version 7.1 and later HTTP session listener.
xsgbean.jar	Server	wasce/lib	The xsgbean.jar file contains the GBean for embedding eXtreme Scale servers in WebSphere Application Server Community Edition application servers.
splicer.jar	Utility	legacy/session/lib	The splicer utility for the WebSphere eXtreme Scale Version 7.0 HTTP session manager filter.
wxsra.rar	Client and server	session/lib	The wxsra.rar contains the eXtreme Scale resource adapter to connect to the grid using a connection factory.

Table 7. Runtime files for WebSphere eXtreme Scale Client. WebSphere eXtreme Scale Client relies on ObjectGrid processes and related APIs. The following table lists the JAR files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
wxsdynacache.jar	Client and server	dynacache/lib	The wxsdynacache.jar file contains the necessary classes to use with the dynamic cache provider. The file is automatically included in the server runtime environment when you use the supplied scripts.
wxshyperic.jar	Utility	hyperic/lib	The WebSphere eXtreme Scale server detection plug-in for the SpringSource Hyperic monitoring agent.
ogagent.jar	Local, client, and server	lib	The ogagent.jar file contains the runtime classes that are required to run the Java instrumentation agent that is used with the EntityManager API.
ogclient.jar	Local and client	lib	The ogclient.jar file is an OSGi bundle that contains only the local and client runtime environments. You can use this file with Java SE 1.5 and later.
ogspring.jar	Local, client, and server	lib	The ogspring.jar file contains support classes for the SpringSource Spring framework integration.
wsogclient.jar	Local and client	lib	The wsogclient.jar file installed when you use an environment that containsWebSphere Application Server Version 6.1 and later. This file contains only the local and client runtime environments.
wxssizeagent.jar	Local, client, and server	lib	The wxssizeagent.jar file is used to provide more accurate cache entry sizing information when using Java runtime environment (JRE) Version 1.5 and later.
ibmcfw.jar ibmorb.jar ibmorbapi.jar	Client and server	lib/endorsed	This set of files includes the Object Request Broker (ORB) runtime that is used for running applications in Java SE processes.

Table 7. Runtime files for WebSphere eXtreme Scale Client (continued). WebSphere eXtreme Scale Client relies on ObjectGrid processes and related APIs. The following table lists the JAR files that are included in the installation. The installation location is relative to the *wxs_home* directory that you choose during the installation.

File name	Environment	Installation location	Description
restservice.ear	Client	restservice/lib	The restservice.ear file contains the eXtreme Scale REST data service application enterprise archive for WebSphere Application Server environments.
restservice.war	Client	restservice/lib	The restservice.war file contains the eXtreme Scale REST data service Web archive for application servers acquired from another vendor.
xsadmin.jar	Utility	samples	The xsadmin.jar file contains the eXtreme Scale administration sample utility.
sessionobjectgrid.jar	Client and server	session/lib	The sessionobjectgrid.jar file contains the eXtreme Scale HTTP session management runtime.
splicerlistener.jar	Utility	session/lib	The splicerlistener.jar file contains the splicer utility for the eXtreme Scale Version 7.1 and later HTTP session listener.
splicer.jar	Utility	legacy/session/lib	The splicer utility for the WebSphere eXtreme Scale Version 7.0 HTTP session manager filter.
wxsra.rar	Client and server	session/lib	The wxsra.rar contains the eXtreme Scale resource adapter to connect to the grid using a connection factory.

Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with silent mode

Use a fully qualified response file, which you configure specifically to your needs, or pass parameters to the command line to silently install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client.

Before you begin

- Stop all processes that are running in your WebSphere Application Server or WebSphere Application Server Network Deployment environment. See Command-line utilities for more information about the **stopManager**, **stopNode**, and **stopServer** commands.

  CAUTION:
  Ensure that any running processes are stopped. If the running processes are not stopped, the installation proceeds, creating unpredictable results and leaving the installation in an undetermined state on some platforms.
- Verify that the target installation directory is empty or does not exist.

  Important: If a previous version of WebSphere eXtreme Scale or the ObjectGrid component exists in the directory that you specify to install Version 7.1.1, the product is not installed. For example, you might have a previously existing *was_root*/ObjectGrid folder. You can either select a different installation directory or cancel the installation. Next, uninstall the previous installation and run the wizard again.

About this task

A silent installation uses the same installation program that the graphical user interface (GUI) version uses. However, instead of displaying a wizard interface, the silent installation reads all of your responses from a file that you customize, or from parameters that you pass to the command line. See an example of a "wxssetup.response.txt file" on page 196, which includes a description of each option.

Procedure

1. Optional: If you choose to install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client using a response file, first customize the `wxssetup.response.txt` file.

   Remember: You must specify the fully-qualified response file name. Specifying the relative path causes the installation to fail with no indication that an error occurred.

   a. Make a copy of the response file to customize.

      For the WebSphere eXtreme Scale full installation, copy the response file from the product DVD to your disk drive.

      For the WebSphere eXtreme Scale Client, unzip the WebSphere eXtreme Scale Client compressed file onto your hard drive and find the response file.

   b. Open and edit the response file in the text editor of your choice. The previous example response file provides details on how to specify each of the parameters. You must specify the following parameters:

      - The license agreement
      - The installation directory

      Tip: When you install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in a WebSphere Application Server environment, the installer uses the installation directory to determine where the existing WebSphere Application Server instance is installed. If you install on a node that contains multiple WebSphere Application Server instances, clearly define your location.

   c. Run the following script to start the installation.

      For the WebSphere eXtreme Scale full installation:

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

      For the WebSphere eXtreme Scale Client installation:

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

      You can also use the response file when you run a GUI installation. You can use the response file with a GUI installation to debug problems that are hidden with the silent installation. When you specify the `wxssetup.response` file for GUI or silent installations, you must use the fully qualified path. Run the following script to run the GUI installation with your response file:

      - 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. Optional: If you choose to install eXtreme Scale by passing certain parameters to the command line, run the following script to start the installation:

   For the WebSphere eXtreme Scale full installation:

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

   For the WebSphere eXtreme Scale Client installation:

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

Related reference:

"Response file for silent installation" on page 219
Specify parameters at the command line to customize and configure your product installation.

Response file for silent installation

Specify parameters at the command line to customize and configure your product installation.

Note: You must specify the fully-qualified response file name. Specifying the relative path causes the installation to fail with no indication that an error occurred.

Parameters

You can pass the following parameters during a command-line or options file installation of the product:

`-silent`
Suppresses the graphical user interface (GUI). Specify the **`-options`** parameter to indicate that the installer completes the installation according to a customized options file. If you do not specify the **`-options`** parameter, the default values are used instead.

Example usage

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

`-options` *path_name/file_name*
Specifies an options file that the installer uses to complete a silent installation. Properties on the command line take precedence.

Example usage

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

`-log` *#* **`!file_name @`***event_type*
Generates an installation log file that logs the following event types:

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

Example usage

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

`-is:log` *path_name/file_name*
Creates a log file that contains the Java Virtual Machine (JVM) searches of the installer while attempting to start the GUI. The log file is not created unless specified.

Example usage

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

`-is:javaconsole`
Displays a console window during the installation process.

Example usage

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

`-is:silent`
Suppresses the Java initialization window that is displayed as the installer starts.

Example usage

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

-is:tempdir *path_name*
Specifies the temporary directory that the installer uses during the installation.

Example usage

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

Related tasks:

7.1.1 "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in a stand-alone environment" on page 184
You can install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in an environment that does not contain WebSphere Application Server or WebSphere Application Server Network Deployment. This type of installation is called a stand-alone installation.

"Uninstalling WebSphere eXtreme Scale" on page 230
To remove WebSphere eXtreme Scale from your environment, you can use the wizard or you can silently uninstall the product.

7.1.1 "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server" on page 186
You can install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in an environment in which WebSphere Application Server or WebSphere Application Server Network Deployment is installed. You can use the existing features of WebSphere Application Server or WebSphere Application Server Network Deployment to enhance your eXtreme Scale applications.

"Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203
After you install the product, create unique types of profiles and augment existing profiles for WebSphere eXtreme Scale.

"Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with silent mode" on page 217
Use a fully qualified response file, which you configure specifically to your needs, or pass parameters to the command line to silently install WebSphere eXtreme Scale or WebSphere eXtreme Scale Client.

7.1.1 "Installing WebSphere eXtreme Scale with the installation wizard" on page 183
You can use the installation wizard to install WebSphere eXtreme Scale for stand-alone or WebSphere Application Server configurations.

"Silently installing a CIP or an IIP" on page 195
You can silently install a customized installation package (CIP) or an integrated installation package (IIP) for the product by using either a fully-qualified response file, which you configure specifically to your needs, or parameters that you pass to the command line.

Installing the REST data service

This topic describes how to install the WebSphere eXtreme Scale REST data service into a Web server.

Before you begin

Software requirements

The WebSphere eXtreme Scale REST data service is a Java Web application that can be deployed to any application server that supports Java servlet specification, Version 2.3 and a Java runtime environment, Version 5 or later.

The following software is required:
- Java Standard Edition 5 or later
- Web servlet container, Version 2.3 or later, which includes one of the following:
  - WebSphere Application Server Version 6.1.0.25 or later
  - WebSphere Application Server Version 7.0.0.5 or later
  - WebSphere Community Edition Version 2.1.1.3 or later
  - Apache Tomcat Version 5.5 or later

  WebSphere eXtreme Scale, Version 7.1 or later, including the trial.

About this task

The WebSphere eXtreme Scale REST data service includes a single `wxsrestservice.war` file. The `wxsrestservice.war` file includes a single servlet that acts as a gateway between your WCF Data Services client applications or any other HTTP REST client and a data grid.

The REST data service includes a sample that allows you to quickly create a data grid and interact with it using an eXtreme Scale client or the REST data service. See "Configuring REST data services" on page 423 for details on using the sample.

When WebSphere eXtreme Scale 7.1 is installed or the eXtreme Scale Version 7.1 trial is extracted, the following directories and files are included:
- `restservice_home/lib`

  The lib directory contains these files:
  - `wxsrestservice.ear` – The REST data service enterprise application archive for use with WebSphere Application Server and WebSphere Application Server CE.
  - `wxsrestservice.war` – The REST data service web module for use with Apache Tomcat.

  The `wxsrestservice.ear` file includes the `wxsrestservice.war` file and are both tightly coupled with the WebSphere WebSphere eXtreme Scale runtime. If WebSphere eXtreme Scale is upgraded to a new version or a fix pack applied, the `wxsrestservice.war` file or `wxsrestservice.ear` file will need to be manually upgraded to the version installed in this directory.
- `restservice_home/gettingstarted`

  The `gettingstarted` directory contains a simple sample that demonstrates how to use the WebSphere eXtreme Scale REST data service with a data grid.

Procedure

Package and deploy the REST data service.
The REST data service is designed as a self-contained WAR module. To configure the REST data service, you must first package the REST data service configuration and optional WebSphere eXtreme Scale configuration files into a JAR file or directory. This application packaging is then referenced by the web container server runtime. The following diagram illustrates files used by the eXtreme Scale REST data service.

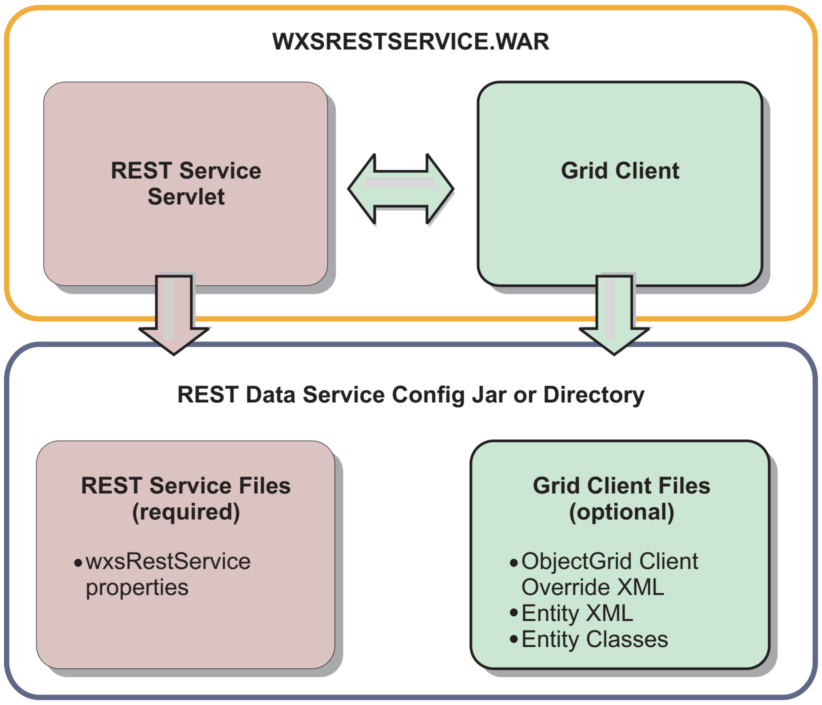

WXSRESTSERVICE.WAR
REST Service Servlet
Grid Client
REST Data Service Config Jar or Directory
REST Service Files (required)
•wxsRestService properties
Grid Client Files (optional)
•ObjectGrid Client Override XML
•Entity XML
•Entity Classes

Figure 28. WebSphere eXtreme Scale REST Data Service Files

The REST service configuration JAR or directory must contain the following file: `wxsRestService.properties`: The `wxsRestService.properties` file includes the configuration options for the REST data service. This includes the catalog service endpoints, ObjectGrid names to expose, trace options and more. See REST data service properties file.

The following ObjectGrid client files are optional:

- `META-INF/objectGridClient.xml`: The ObjectGrid client override XML file is used to connect to the remote data grid. By default this file is not required. If this file is not present, the REST service uses the server configuration, disabling the near cache.

  The name of the file can be overridden using the objectGridClientXML REST data service configuration property. If provided, this XML file should include:

  1. Any ObjectGrids that you want to expose to the REST data service.
  2. Any reference to the entity descriptor XML file associated with each ObjectGrid configuration.
- `META-INF/`*`entity descriptor XML files`*: One or more entity descriptor XML files are required only if the client needs to override the entity definition of the client. The entity descriptor XML file must be used in conjunction with the ObjectGrid client override XML descriptor file.
- **Entity classes** Annotated entity classes or an entity descriptor XML file can be used to describe the entity metadata. The REST service only requires entity classes in the classpath if the eXtreme Scale servers are configured with entity metadata classes and a client override entity XML descriptor is not used.

  An example with the minimum required configuration file, where the entities are defined in XML on the servers:

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

  The property file contains:

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

An example with one entity, override XML files and entity classes:

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

The property file contains:

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

```
com/acme/entities/Customer.class
META-INF/objectGridClient.xml
```

The client ObjectGrid descriptor XML file contains:

```
<objectGrid name="CustomerGrid" entityMetadataXMLFile="emd.xml"/>
META-INF/emd.xml
```

The entity metadata descriptor XML file contains:

```
<entity class-name="com.acme.entities.Customer" name="Customer"/>
```

Installing the Eclipse Equinox OSGi framework with Eclipse Gemini for clients and servers

If you want to deploy WebSphere eXtreme Scale in the OSGi framework, then you must set up the Eclipse Equinox Environment.

About this task

The task requires that you download and install the Blueprint framework, which allows you to later configure JavaBeans and expose them as services. The use of services is important because you can expose plug-ins as OSGi services so they can be used by the eXtreme Scale run time environment. The product supports two blueprint containers within the Eclipse Equinox core OSGi framework: Eclipse Gemini and Apache Aries. Use this procedure to set up the Eclipse Gemini container.

Procedure

1. Download Eclipse Equinox SDK Version 3.6.1 or later from the Eclipse website. Create a directory for the Equinox framework, for example: `/opt/equinox`. These instructions refer to this directory as `equinox_root`. Extract the compressed file in the `equinox_root` directory.
2. Download the gemini-blueprint incubation 1.0.0 compressed file from the Eclipse website. Extract the file contents into a temporary directory, and copy the following extracted files to the `equinox_root/plugins` directory:

   ```
   dist/gemini-blueprint-core-1.0.0.jar
   dist/gemini-blueprint-extender-1.0.0.jar
   dist/gemini-blueprint-io-1.0.0.jar
   ```

   Attention: Depending on the location where you download the compressed Blueprint file, the extracted files might have the extension, `RELEASE.jar`, much like the Spring framework JAR files in the next step. You must verify that the file names match the file references in the `config.ini` file.
3. Download the Spring Framework Version 3.0.5 from the following SpringSource web page: http://www.springsource.com/download/community. Extract it into a temporary directory, and copy the following extracted files to the `equinox_root/plugins` directory:

```
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. Download the AOP Alliance Java archive (JAR) file from the SpringSource web page. Copy the com.springsource.org.aopalliance-1.0.0.jar to the `equinox_root/plugins` directory.
5. Download the Apache commons logging 1.1.1 JAR file from the SpringSource web page. Copy the `com.springsource.org.apache.commons.logging-1.1.1.jar` file to the `equinox_root/plugins` directory.
6. Download the Luminis OSGi Configuration Admin command-line client. Use this JAR file bundle to manage OSGi administrative configurations. Copy the `net.luminis.cmc-0.2.5.jar` to the `equinox_root/plugins` directory.
7. Download the Apache Felix file installation Version 3.0.2 bundle from the following web page: http://felix.apache.org/site/index.html. Copy the `org.apache.felix.fileinstall-3.0.2.jar` file to the `equinox_root/plugins` directory.
8. Create a configuration directory inside `equinox_root/plugins` directory; for example:

   ```
   mkdir equinox_root/plugins/configuration
   ```
9. Create the following `config.ini` file in the `equinox_root/plugins/configuration` directory, replacing *equinox_root* with the absolute path to your `equinox_root` directory and removing all trailing spaces after the backslash on each line. You must include a blank line at the end of the file; for example:

   ```
   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
   ```

   If you have already set up the environment, you can clean up the Equinox plug-in repository by removing the following directory: `equinox_root\plugins\configuration\org.eclipse.osgi`.
10. Run the following commands to start equinox console.

    If you are running a different version of Equinox, then your JAR file name is different from the one in the following example:

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

Related concepts:

OSGi framework overview
OSGi defines a dynamic module system for Java. The OSGi service platform has a layered architecture, and is designed to run on various standard Java profiles. You can start WebSphere eXtreme Scale servers and clients in an OSGi container.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Related information:

"Introduction: Starting and configuring the eXtreme Scale server and container to run plug-ins in the OSGi framework" on page 154
In this tutorial you start an eXtreme Scale server in the OSGi framework, start an eXtreme Scale container, and wire the sample plug-ins with eXtreme Scale runtime environment.

Installing eXtreme Scale bundles

WebSphere eXtreme Scale includes bundles that can be installed into an Eclipse Equinox OSGi framework. These bundles are required to start eXtreme Scale servers or use eXtreme Scale clients in OSGi. You can install the eXtreme Scale bundles using the Equinox console or using the config.ini configuration file.

Before you begin

This task assumes that you have installed the following products:

- Eclipse Equinox OSGi framework
- eXtreme Scale stand-alone client or server

About this task

eXtreme Scale includes two bundles. Only one of the following bundles is required in an OSGi framework:

objectgrid.jar
The server bundle is the `objectgrid.jar` file and is installed with the eXtreme Scale stand-alone server installation and is required for running eXtreme Scale servers and can also be used for running eXtreme Scale clients, or local, in-memory caches. The bundle ID for the `objectgrid.jar` file is com.ibm.websphere.xs.server_`<version>`, where the version is in the format: `<Version>.<Release>.<Modification>`. For example, the server bundle for this release is `com.ibm.websphere.xs.server_8.5.0`.

ogclient.jar
The `ogclient.jar` bundle is installed with the eXtreme Scale stand-alone and client installations and is used to run eXtreme Scale clients or local, in-memory caches. The bundle ID for `ogclient.jar` file is com.ibm.websphere.xs.client_`<version>`, where the version is in the format: `<Version>_<Release>_<Modification`. For example, the client bundle for this release is `com.ibm.websphere.xs.server_8.5.0`.

For more information about developing eXtreme Scale plug-ins, see the System APIs and Plug-ins topic.

Related concepts:

"Embedded server API" on page 479
WebSphere eXtreme Scale includes application programming interfaces (APIs) and system programming interfaces for embedding eXtreme Scale servers and clients within your existing Java applications.

Install the eXtreme Scale client or server bundle into the Eclipse Equinox OSGi framework using the Equinox console

Procedure

1. Start the Eclipse Equinox framework with the console enabled; for example:
   ```
   java_home/bin/java -jar <equinox_root>/plugins/
   org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
2. Install the eXtreme Scale client or server bundle in the Equinox console:
   ```
   osgi> install file:///<path to bundle>
   ```
3. Equinox displays the bundle ID for the newly installed bundle:
   ```
   Bundle id is 25
   ```
4. Start the bundle in the Equinox console, where `<id>` is the bundle ID assigned when the bundle was installed:
   ```
   osgi>  start <id>
   ```
5. Retrieve the service status in the Equinox console to verify that the bundle has started; for example:
   ```
   osgi> ss
   ```

   When the bundle starts successfully, the bundle displays the ACTIVE state; for example:
   ```
   25      ACTIVE      com.ibm.websphere.xs.server_8.5.0
   ```

Install the eXtreme Scale client or server bundle into the Eclipse Equinox OSGi framework using the `config.ini` file

Procedure

1. Copy the eXtreme Scale client or server (objectgrid.jar or ogclient.jar) bundle from the `<wxs_install_root>/ObjectGrid/lib` to the Eclipse Equinox plug-ins directory; for example: <equinox_root>/plugins
2. Edit the Eclipse Equinox `config.ini` configuration file, and add the bundle to the osgi.bundles property; for example:
   ```
   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
   ```

   Important: Verify that a blank line exists after the last bundle name. Each bundle is separated by a comma.
3. Start the Eclipse Equinox framework with the console enabled; for example:
   ```
   java_home/bin/java -jar <equinox_root>/plugins/
   org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
4. Retrieve the service status in the Equinox console to verify that the bundle has started:
   ```
   osgi> ss
   ```

   When the bundle starts successfully, the bundle displays the ACTIVE state; for example:
   ```
   25      ACTIVE      com.ibm.websphere.xs.server_8.5.0
   ```

Results

The eXtreme Scale server or client bundle is installed and started in your Eclipse Equinox OSGi framework.

Verifying the installation

After the installation wizard completes, you can verify the installation by checking several aspects of the installation.

Procedure

- **For an installation that is integrated with WebSphere Application Server or a stand-alone installation:**

  Use one of the following methods to verify that your installation completed successfully:

  – For WebSphere eXtreme Scale installations integrated with WebSphere Application Server, enter the following version information command:

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

    The product name, version number, and build number are displayed as a result.

  – For stand-alone WebSphere eXtreme Scale installations, enter the following version information command:

    ```
    wxs_install_root/ObjectGrid/lib> java -jar objectgrid.jar version
    ```

    The product name, version number, and build number are displayed as a result.

  – Check properties files for the proper version number.

    - Signature files: The signature files are in the *was_root*/`properties/version` directory. If a fix pack has been installed, additional `fxtg` files are also included. Some examples of signature file names follow:

      ```
      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 product file:

      The product file is in the *was_root*/`properties/version` directory. Look for the `WXS.product` file. An example of the contents of this file follows:

      ```
      <?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>
      ```

  – Verify that the runtime files are installed. Lists of the runtime files for each installation type are documented in the following topics:

    - "Runtime files for WebSphere eXtreme Scale stand-alone installation" on page 215
    - "Runtime files for WebSphere eXtreme Scale integrated with WebSphere Application Server" on page 213

- **For an installation that is integrated with WebSphere Application Server, you have the following additional ways to check that the installation has completed successfully:**

  – Run the version info command for WebSphere Application Server:

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

The output displays a list of the installed products, including installation directories, products installed, versions, build level, build date, and so on.

Tip: Add the **-maintenancePackages** parameter to see further details:

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

– Check the Welcome panel for the WebSphere Application Server administrative console. Go to `http://localhost:9060/ibm/console`. Log in to the console. The version of WebSphere eXtreme Scale displays on the Welcome panel.
– Use the first steps console to augment the WebSphere Application Server installation with WebSphere eXtreme Scale:

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

For more information, see "Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203.

What to do next

If you see that the installation did not complete as you suspected, you must troubleshoot the installation. For more information, see "Troubleshooting installation."

Taking the first steps after installation

After complete and verify the installation, you can begin to use WebSphere eXtreme Scale to create your data grid.

Procedure

1. Update your installation by applying maintenance.

   More information: "Updating eXtreme Scale servers" on page 235.
2. If you are using WebSphere eXtreme Scale for the first time, you can use the Getting started information to learn more about how to use the product.

   More information: Chapter 1, "Getting started," on page 1
3. Configure the product. Create properties and XML files to define the configuration for data grids, servers, and clients. You can also configure cache or database integration, REST data services, or OSGi plug-ins.

   More information: Chapter 6, "Configuring," on page 257
4. Develop an application that accesses the data grid.

   More information: Developing applications
5. Start and administer container and catalog servers with your configuration files and data grid application.

   More information: Chapter 7, "Administering," on page 459
6. Monitor the performance of your configuration with the various monitoring tools.

   More information: Chapter 8, "Monitoring," on page 511

Troubleshooting installation

Use this information to troubleshoot issues with your installation and updates.

Procedure

- **Problem:** When you run the installation command from a remote computer, such as `\\mymachine\downloads\`, the following message displays: `CMD.EXE was started with the above path as the current directory. UNC paths are not supported. Defaulting to Windows directory.` As a result, the installation does not complete correctly.

  Solution: Map the remote computer to a network drive. For example, in Windows, you can right-click **My computer** and choose **Map Network Drive** and include the uniform naming conventions (UNC) path to the remote computer. You can then run the installation script from the network drive successfully, for example: `y:\mymachine\downloads\WXS\install.bat`.
- **Problem:** The installation completes unsuccessfully.

  Solution: Check the log files to see where the installation failed. When the installation completes unsuccessfully, the logs are in the *wxs_install_root*`/logs/wxs` directory.
- **Problem:** A catastrophic failure occurs during the installation.

  Solution: Check the log files to see where the installation failed. When the installation fails when it is partially completed, the logs can generally be found in the *`user_root`*`/wxs_install_logs/` directory.
- Windows **Problem:** If you are installing the WebSphere eXtreme Scale Client on Windows, you might see the following text in the results of the installation:

  ```
  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.
  ```

  Solution: If you see a failure with the `iscdeploy.sh` file, you can ignore the error. This error does not cause any problems.
- Linux **Problem:**

  If you have a full installation and try to apply WebSphere eXtreme Scale Client only maintenance with the update installer, you see the following message:

  ```
  Prerequisite checking has failed. Click Back to select a different package,
  or click Cancel to exit.

  Failure messages are:

  Required feature wxs.client.primary is not found.
  ```

  If you have WebSphere eXtreme Scale Client installed and try to apply a full maintenance package with the update installer, you see the following message:

  ```
  Prerequisite checking has failed. Click Back to select a different package,
  or click Cancel to exit.

  Failure messages are:

  Required feature wxs.primary is not found.
  ```

  Solution: The maintenance package that you install must match the type of installation. Download and apply the maintenance package that applies to your installation type.
- Linux **Problem:** The installation hangs.

  Solution: Sometimes, when installing WebSphere eXtreme Scale on Linux as a non-root user, the installer can hang. This is likely because the maximum number of open files is set too low on your Linux operating system. You will need to raise the allowed limit in the `/etc/limits.conf` or `/etc/security/limits.conf` file (where the file is located depends on your specific Linux distribution) to at least 8192.

Uninstalling WebSphere eXtreme Scale

To remove WebSphere eXtreme Scale from your environment, you can use the wizard or you can silently uninstall the product.

Before you begin

Attention: The uninstaller removes all binary files and all maintenance, such as fix packs and interim fixes, at the same time.

Procedure

1. Stop all processes that are running eXtreme Scale.

   CAUTION:
   Ensure that any running processes are stopped. If the running processes are not stopped, the uninstallation proceeds, creating unpredictable results and leaving the uninstallation in an undetermined state on some platforms.

   - If you installed stand-alone eXtreme Scale, read about stopping stand-alone servers to stop processes.
   - If you installed eXtreme Scale with an existing installation of WebSphere Application Server, read about command-line utilities for more information about stopping WebSphere Application Server processes.
   - If you are running the web console, use the `stopConsoleServer` command to stop the web console server. The `stopConsoleServer` script is in the *wxs_install_root*/`ObjectGrid/bin` directory. If you do not stop this server before running the uninstallation, the process is automatically stopped during the uninstallation process.
2. Uninstall the product. You can run the uninstallation in a GUI or silently.

   Note: When specifying the response file `wxssetup.response` file for silent or GUI uninstall or installations, the fully qualified path must always be specified. The response file is optional for the GUI uninstallation.

   - **To run the uninstallation using the GUI:**
     - Linux UNIX `<`*`install_home`*`>/uninstall_wxs/uninstall`
     - Windows `<`*`install_home`*`>\uninstall_wxs\uninstall.exe`

     If you want to run the uninstallation using the GUI and the `wxssetup.response` file, use one of the following commands:

     - 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
       ```
   - **To run the uninstallation silently using the response file `wxssetup.response` script:**
     - 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
```

Results

You removed eXtreme Scale from your environment.

Related reference:

"Response file for silent installation" on page 219
Specify parameters at the command line to customize and configure your product installation.

Customizing WebSphere eXtreme Scale for z/OS

Using the WebSphere Customization Toolbox, you can generate and run customized jobs to customize WebSphere eXtreme Scale for z/OS®.

Before you begin

- Verify that your system contains the latest level of WebSphere Application Server Network Deployment:
  - If you are running Version 6.1, your system must contain fix pack 39 at a minimum. See Installing your Version 6.1 application serving environment for more information.
  - If you are running Version 7.0, your system must contain fix pack 19 at a minimum. See Installing your Version 7.0 application serving environment for more information.
  - If you are running Version 8.0, your system must contain fix pack 4 at minimum. See Installing your Version 8.0 application serving environment for more information.
- Install WebSphere eXtreme Scale for z/OS. See the *WebSphere eXtreme Scale Program Directory* on the Library Page for more information.

About this task

Using the WebSphere Customization Toolbox, generate customization definitions and upload and run customized jobs to customize WebSphere eXtreme Scale for z/OS. See the following topics for more information:

Procedure

- "Installing the WebSphere Customization Toolbox"
- "Generating customization definitions" on page 232
- "Uploading and running customized jobs" on page 233

Installing the WebSphere Customization Toolbox

Install the WebSphere Customization Toolbox to customize your WebSphere eXtreme Scale for z/OS environment.

Before you begin

- Install WebSphere eXtreme Scale for z/OS. See the *WebSphere eXtreme Scale Program Directory* on the Library Page for more information.

- You must use the latest version of the WebSphere Customization Toolbox to successfully install the product extension files.

About this task

The WebSphere Customization Tools is a workstation-based graphical tool you use to create customized jobs that build WebSphere eXtreme Scale for z/OS runtime environments.

Procedure

1. Use FTP to copy the `xs.wct` extension file from your z/OS system to the workstation on which you are installing the WebSphere Customization Tools.

   7.1.1 The extension files are in the `/usr/lpp/zWebSphereXS/util/WCT` directory on your z/OS operating system.
2. Download and install the latest version of the WebSphere Customization Toolbox.
3. Upload the `xs.wct` file to the WebSphere Customization Toolbox application.
   a. Start the WebSphere Customization Toolbox application on your workstation.
   b. Click **Help** > **Software Updates** > **Manage Extension**.
   c. From the WebSphere Customization Toolbox Extension panel, click **Install**.
   d. From the Source Archive File panel, click **Browse**, navigate to the directory in which you copied the `xs.wct` file in step 1, and click **Open**.
   e. Click **Next** on the Source Archive panel.
   f. Click **Next** on the Extension summary panel, and click **Finish** on the WebSphere Customization Toolbox Extension panel.
4. 7.1.1+ From the WebSphere Customization Toolbox Extension panel, click **Install**.
5. 7.1.1+ From the Source Archive File panel, click **Browse**, navigate to the directory in which you copied the `xspf.wct` file in step 1, and click **Open**.
6. 7.1.1+ Click **Next** on the Summary panel.
7. 7.1.1+ Click **Next** on the Extension summary panel, and click **Finish** on the WebSphere Customization Toolbox Extension panel

What to do next

After you upload both extension files and restart the WebSphere Customization Toolbox, you can use the Profile Management Tool to generate customization definitions for WebSphere eXtreme Scale for z/OS. See "Generating customization definitions" for more information.

Generating customization definitions

Use the Profile Management Tool function within the WebSphere Customization Toolbox to generate customization definitions and create customized jobs for WebSphere eXtreme Scale for z/OS.

Before you begin

- Verify that your system contains the latest level of WebSphere Application Server Network Deployment. See "Customizing WebSphere eXtreme Scale for z/OS" on page 231.

About this task

You can generate customization definitions using the Profile Management Tool, which is provided in the WebSphere Customization Tools. A *customization definition* is a set of files used to create customized jobs for configuring WebSphere eXtreme Scale for z/OS.

Procedure

1. Start the Profile Management Tool.
   - Windows Click **Start** > **Programs** > **IBM WebSphere** > **WebSphere Customization Toolbox** > **WebSphere Customization Toolbox**. After the application starts, click the **Profile Management Tool** tab.
   - Linux Run the shell script, `wct.sh`, which is in the `/opt/IBM/WebSphere/Toolbox/` directory. After the application starts, click the **Profile Management Tool** tab.
2. Using the customization locations that you have added or created for WebSphere Application Server for z/OS, select the profile that you want to augment from the Customization Locations list of the existing WebSphere Application Server product version that is installed on your z/OS operating system.

   Note: Do not use the same location that you are using for other WebSphere eXtreme Scale customization definitions.
3. Select an environment from the Customization Definitions list for WebSphere Application Server for z/OS. Click **Augment** to create the response file and list of instructions for augmenting the WebSphere Application Server for z/OS runtime environment. The Profile Management Tool Environment Selection panel is displayed.
4. Select the environment to augment from the Environments list, and click **Next**.
   - Management
   - Application server
   - Managed (custom) node

   The Profile Management Tool Augment Selection panel is displayed.
5. Select the type of augmentation to apply from the list of Augment types, and click **Next**.
6. Complete the fields on the panels. Specify the values for the parameters that are used to create your WebSphere Application Server for z/OS runtime environment.
7. Click **Augment** to generate the customization definition.
8. Click **Finish** to close the dialog, and continue.

What to do next

Upload the customized job to your target z/OS system. See "Uploading and running customized jobs" for more information.

Uploading and running customized jobs

After you generate the customization definitions, you can upload and run the customized jobs that are associated with the definitions to your WebSphere eXtreme Scale for z/OS system.

Before you begin

Generate the customization definitions for the jobs that you want to upload to your z/OS system. For more information, see “Generating customization definitions” on page 232.

About this task

Upload and run the customized jobs that you created using the WebSphere Customization Tools to administer and monitor your WebSphere eXtreme Scale for z/OS environment.

Procedure

1. Upload the customized jobs. On the **Customization Definitions** tab, select the jobs that you want to upload and click **Process**. The Profile Management Tool Select Process Type panel is displayed.
2. Select the **FTP upload type** for the target z/OS operating system, and click **Next**. Specify the required information on the Upload Customization Definition panel.
3. Click **Finish**.
4. Run the customized jobs. Click the **Customization Instructions** tab, and follow the customization instructions for each job.

Chapter 5. Upgrading and migrating WebSphere eXtreme Scale

You can migrate to Version 7.1.1 from previous versions, or you can apply maintenance packages. To avoid outages, you must consider the order in which you apply the updates to the servers in your configuration.

Updating eXtreme Scale servers

You can upgrade WebSphere eXtreme Scale to a new version, either by applying maintenance or installing a new version, without interrupting service.

Before you begin

You must have the binary file for the major version release or maintenance that you want to apply. You can get the latest information about the available releases and maintenance packages from the IBM support portal for WebSphere eXtreme Scale.

About this task

To upgrade without service interruption, first upgrade catalog servers, then upgrade the container servers and client servers. If you have a server and client installation on the same physical server, you can upgrade the full installations on each physical server. If you have client-only installations, upgrade these installations last.

7.1.1 Before you can upgrade to WebSphere eXtreme Scale Version 7.1.0 or earlier to Version 7.1.1.x, you must first apply an interim fix to each installation used by your client and container servers. The fix is available at IBM support portal. After applying the fix, you can proceed to upgrade your version of WebSphere eXtreme Scale to version 7.1.1.x.

Procedure

1. Upgrade the catalog service tier, repeating the following steps for each catalog server in the data grid. Upgrade the catalog service tier before upgrading any container servers or clients. Individual catalog servers can interoperate with version compatibility, so you can apply upgrades to one catalog server at a time without interrupting service.
   a. If you are running with quorum mechanism enabled, check for a healthy quorum status. Run the following command:
      ```
      xsadmin -quorumStatus
      ```
      ```
      xscmd -c showQuorumStatus
      ```
      This result indicates that all the catalog servers are connected.
   b. If you are using multi-master replication between two catalog service domains, dismiss the link between the two catalog service domains while you are upgrading the catalog servers.
      ```
      xsadmin –ch host -p 1099 -dismissLink domain_name
      ```

7.1.1+

```
xscmd –c dismissLink –cep host:2809 -fd domain_name
```

You only need to run this command from one of the catalog service domains to remove the link between two catalog service domains.

c. Shut down one of the catalog servers. You can use the **stopOgServer** command, the **xscmd -c teardown** command, or shut down the application server that is running the catalog service in WebSphere Application Server. There are no requirements for the order in which you stop the catalog servers, but shutting down the primary catalog server last reduces turnover. To determine which catalog server is the primary, look for the CWOBJ8106 message in the log files. Under normal conditions when the quorum mechanism is enabled, quorum is maintained when a catalog server is shut down, but it is a best practice to query quorum status after each shutdown with the **xscmd -c showQuorumStatus** command.

   You can provide a specific list of servers to stop to the **stopOgServer** command, or the **xscmd -c teardown** commands:

```
stopOgServer server_name
```

```
xsadmin –teardown server_name
```

   7.1.1+

```
xscmd –c teardown -sl server_name
```

   With the previous examples, the **stopOgServer** , or **xscmd -c teardown** commands are completing the same shutdown tasks. However, you can filter the servers to stop with the **xscmd -c teardown** command. See "Stopping servers gracefully with the **xscmd** utility" on page 474 for more information about filtering the servers by zone or host name. The teardown command filters out the matching servers and asks if the selected servers are correct.

d. Install the updates on the catalog server. You can either migrate the catalog server to a new major release of the product or apply a maintenance package. See the following topics for more information:
   - To migrate from a installation: "Migrating to WebSphere eXtreme Scale Version 7.1.1" on page 238

e. Restart the catalog server.

   If you are using a stand-alone environment, see "Starting a stand-alone catalog service" on page 461 for more information. If you are using a WebSphere Application Server environment, see "Starting and stopping servers in a WebSphere Application Server environment" on page 475 for more information.

   The catalog server runs in compatibility mode until all the catalog servers are moved to the same level. Compatibility mode mostly applies to major release migrations because new functions are not available on the servers that are not migrated. No restrictions exist on how long catalog servers can run in compatibility mode, but the best practice is to migrate all catalog servers to the same level as soon as possible.

f. Verify that the catalog server started successfully. Ensure that the following **xscmd** commands return valid results:

```
xscmd -c routetable -cep cathost:2809
xscmd -c showMapSizes -cep cathost:2809
```

Important: These commands must contain the -cep *<catalog_server_host>*:*<listener_port>* value for the restarted catalog server.

   g. Apply updates to the remaining catalog servers in your configuration.

2. Upgrade the container servers, repeating the following steps for each container server in the data grid. You can upgrade container servers in any order.
   a. Stop the container servers that you want to upgrade. You can stop the container servers that you want to updgrade with the **stopOgserver** command, or with the **xscmd -c teardown** command. For more information, see "Stopping stand-alone servers" on page 472 and "Stopping servers gracefully with the **xscmd** utility" on page 474.

      By running the **xscmd -c teardown** or **stopOgserver** commands to handle multiple servers in parallel, the placement mechanism can move shards in larger groups. However, do not to take down too many servers at the same time. The resources of servers that remain might become overloaded.
   b. Verify that the container servers were stopped and removed from the data grid. . Run the following **xscmd** commands and verify that the results do not contain the stopped container servers.
      ```
       xscmd -c routetable
      xscmd -c showMapSizes
      ```
      If these commands are run too soon after container servers are stopped, correct results might not be returned. Wait a few minutes and try running the commands again.
   c. Install the updates on the container servers. You can either migrate the container servers to a new major release of the product or apply a maintenance package. See the following topics for more information:
      - To migrate from a installation: "Migrating to WebSphere eXtreme Scale Version 7.1.1" on page 238
   d. Restart your container servers.
   e. Verify that the container servers were restarted and added to the data grid. Run the following **xscmd** commands and verify that the results contain the restarted container servers.
      ```
       xscmd -c routetable
      xscmd -c showMapSizes
      ```
      If these commands are run too soon after container servers are started, correct results might not be returned. Wait a few minutes and try running the commands again.
   f. Upgrade any remaining container servers in your configuration.
3. If you are using multi-master replication, reconnect your catalog service domains. Use the **xscmd -c establishLink** command to reconnect the catalog service domains.
   ```
   xsadmin –ch host –p 1099 –establishLink dname fdHostA:2809,fdHostB:2809
   ```
   ```
   xscmd –c establishLink -cep host:2809 -fd dname -fe fdHostA:2809,fdHostB:2809
   ```
4. Upgrade the client-only installations.

What to do next

- You can also use these steps to revert to an older version or to uninstall maintenance packages. However, if you revert to Version 7.1.0 when you are using multi-master replication, the two-way replication might not function correctly when you re-establish the links. In this situation, restart both catalog service domains and re-link the catalog service domains with the **establishLink** command.

Related concepts:

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

Migrating to WebSphere eXtreme Scale Version 7.1.1

With the WebSphere eXtreme Scale installer, you cannot upgrade or modify a previous installation. You must uninstall the previous version before you install the new version. You do not need to migrate your configuration files because they are backward compatible. However, if you changed any of the script files that are shipped with the product, you must reapply these changes to the updated script files.

Before you begin

Verify that your systems meet the minimum requirements for the product versions you plan to migrate and install. See "Hardware and software requirements" on page 59 for more information.

About this task

Merge any modified product script files with new product script files in the `/bin` directory to maintain your changes.

Tip: If you did not modify the script files that are installed with the product, you are not required to complete the following migration steps. Instead, you can upgrade to Version 7.1.1 by uninstalling the previous version and installing the new version in the same directory.

Procedure

1. Stop all processes that are using WebSphere eXtreme Scale.
   - Stop all processes that are running in your stand-alone WebSphere eXtreme Scale environment. For more information, see "Stopping stand-alone servers" on page 472.
   - Read about command-line utilities to stop all processes that are running in your WebSphere Application Server or WebSphere Application Server Network Deployment environment.
2. Save any modified scripts from your current installation directory to a temporary directory.
3. Uninstall the product. For more information, see 7.1.1 "Uninstalling WebSphere eXtreme Scale" on page 230.
4. Install WebSphere eXtreme Scale Version 7.1.1. See Chapter 4, "Installing," on page 171 for more information.
5. Merge your changes from the files in the temporary directory to the new product script files in the `/bin` directory.
6. Start all of your WebSphere eXtreme Scale processes to begin using the product. For more information, see Chapter 7, "Administering," on page 459.

Related concepts:

"Hardware and software requirements" on page 59
Browse an overview of hardware and operating system requirements. Although you are not required to use a specific level of hardware or operating system for WebSphere eXtreme Scale, formally supported hardware and software options are available on the Systems Requirements page of the product support site. If a conflict exists between the information center and the System Requirements page, the information at the website takes precedence. Prerequisite information in the information center is provided as a convenience only.

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

Installing fix packs using IBM Installation Manager

You can use IBM Installation Manager to update the product with the fix packs that are available for WebSphere eXtreme Scale product offerings. Fix packs can be installed from the GUI, the command line, or using response files.

Installing fix packs using the GUI

You can update this product to a later version using the IBM Installation Manager wizard.

Before you begin

Contact the IBM Software Support Center for information about upgrades for WebSphere eXtreme Scale stand-alone or WebSphere eXtreme Scale for WebSphere Application Server product offerings. The most current information is available from the IBM Software Support Center and Fix Central.

IBM Installation Manager is used to apply product maintenance to the following product offerings:

- WebSphere eXtreme Scale in a stand-alone environment
- WebSphere eXtreme Scale Client in a stand-alone environment
- WebSphere eXtreme Scale for WebSphere Application Server Version 7
- WebSphere eXtreme Scale for WebSphere Application Server Version 8
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

Make sure that the web-based or local service repository location is listed and checked or that the **Search service repositories during installation and updates** option is selected on the Repositories panel in your Installation Manager preferences. For more information on using service repositories with Installation Manager, read the IBM Installation Manager Version 1.5 Information Center.

About this task

Restriction: You cannot use the Installation Manager to upgrade an installation and add or remove the full WebSphere Application Server profile feature .

Procedure

1. Stop all processes that are running in your environment.
   - To stop all processes that are running in your stand-alone eXtreme Scale environment, see "Stopping stand-alone servers" on page 472.

- To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.

2. Log on to your system.
3. Stop all servers and applications on the WebSphere Application Server installation that is being updated.
4. Start Installation Manager.
5. Click **Update**.

   Note: If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.
6. Select the package group to update.

   Tip: If you select **Update all**, Installation Manager will search all of the added and predefined repositories for updates to all of the package groups that it has installed. Use this feature only if you have full control over which fixes are contained in the targeted repositories. If you create and point to a set of custom repositories that include only the specific fixes that you want to install, you should be able to use this feature confidently. If you enable searching service repositories or install fixes directly from other live web-based repositories, then you might not want to select this option so that you can select only the fixes that you want to install for each offering on subsequent panels.
7. Click **Next**.
8. Select the version to which you want to update under:
   - WebSphere eXtreme Scale in a stand-alone environment
   - WebSphere eXtreme Scale Client in a stand-alone environment
   - WebSphere eXtreme Scale for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale for WebSphere Application Server Version 8
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
9. Select any fixes that you want to install.

   Any recommended fixes are selected by default.

   If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.
10. Click **Next**.
11. Accept the terms in the license agreements, and click **Next**.
12. Select the optional features that you want in your updated installation.
13. Review the summary information, and click **Update**.
    - If the installation is successful, the program displays a message indicating that installation is successful.
    - If the installation is not successful, click **View Log File** to troubleshoot the problem.
14. Click **Finish**.
15. Click **File > Exit** to close Installation Manager.

Installing fix packs using the command line

You can use the IBM Installation Manager from the command line to update the product with the fix packs that are available for WebSphere eXtreme Scale product offerings.

Before you begin

. Contact the IBM Software Support Center for information about upgrades for WebSphere eXtreme Scale stand-alone or WebSphere eXtreme Scale for WebSphere Application Server product offerings. The most current information is available from the IBM Software Support Center and Fix Central.

IBM Installation Manager is used to apply product maintenance to the following product offerings:

- WebSphere eXtreme Scale in a stand-alone environment
- WebSphere eXtreme Scale Client in a stand-alone environment
- WebSphere eXtreme Scale for WebSphere Application Server Version 7
- WebSphere eXtreme Scale for WebSphere Application Server Version 8
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

About this task

Restriction: You cannot use the Installation Manager to upgrade an installation and add or remove the full WebSphere Application Server profile feature .

Procedure

1. For a list of interim fixes and fix packs that are available forWebSphere eXtreme Scale 8.5 and specific information about each fix, perform the following actions.
   a. Go to Fix Central.
   b. Select **WebSphere** as the product group.
   c. Select WebSphere eXtreme Scale as the product.
   d. Select **8.5** as the installed version.
   e. Select your operating system as the platform, and click **Continue**.
   f. Select **Browse for fixes**, and click **Continue**.
   g. Click **More Information** under each fix to view information about the fix.
   h. **Recommendation:** Make a note of the name of the fix pack that you would like to install.
2. Update WebSphere eXtreme Scale Version 8.5 with the fix pack using the following procedure.
   - Download the file that contains the fix pack from Fix Central, and use local updating.

     You can download a compressed file that contains the fix pack from Fix Central. Each compressed fix-pack file contains an Installation Manager repository for the fix pack and usually has a .zip extension. After downloading and extracting the fix-pack file, use Installation Manager to update WebSphere Application Server Version 8.x with the fix pack.

     a. To download the fix pack, perform the following actions:
        1) Go to Fix Central.
        2) Select **WebSphere** as the product group.

3) Select **WebSphere eXtreme Scale** as the product.
4) Select **8.x** as the installed version.
5) Select your operating system as the platform, and click **Continue**.
6) Select **Browse for fixes**, and click **Continue**.
7) Select the fix pack that you want to download, and click **Continue**.
8) Select your download options, and click **Continue**.
9) Click **I agree** to agree to the terms and conditions.
10) Click **Download now** to download the fix pack.
11) Transfer the compressed file in binary format to the system on which it will be installed.
12) Extract the compressed repository files to a directory on your system.

b. To install a fix pack from a downloaded file, perform the following actions:
   1) Log on to your system.
   2) Stop all processes that are running in your environment. To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.
   3) Change to the `Installation_Manager_binaries/eclipse/tools` directory, where `Installation_Manager_binaries` is the installation root directory for the Installation Manager.
   4) Install the fix pack.

UNIX Linux

```
./imcl install offering_ID_offering_version,optional_feature_ID
  -installationDirectory product_installation_location
  -repositories location_of_expanded_files
  -acceptLicense
```

Windows

```
imcl.exe install offering_ID_offering_version,optional_feature_ID
  -installationDirectory product_installation_location
  -repositories location_of_expanded_files
  -acceptLicense
```

Tips:

– The *offering_ID* is the offering ID that is listed in Offering IDs for WebSphere eXtreme Scale product offerings.

– The *offering_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to install (`8.5.0.20110503_0200` for example).
  - If *offering_version* is **not** specified, the latest version of the offering and **all** interim fixes for that version are installed.
  - If *offering_version* is specified, the specified version of the offering and **no** interim fixes for that version are installed.

  The offering version can be found attached to the end of the offering ID with an underscore when you run the following command against the repository:

  ```
  imcl listAvailablePackages -repositories source_repository
  ```

– You can also specify `none`, `recommended` or `all` with the `-installFixes` argument to indicate which interim fixes you want installed with the offering.

- If the offering version is **not** specified, the `-installFixes` option defaults to `all`.
- If the offering version is specified, the `-installFixes` option defaults to `none`.

– You can add a list of features that are separated by commas. If a list of features is not specified, the default features are installed.

5) **Optional:** List all installed packages to verify the installation:

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Installing fix packs using a response file

You can update this product to a later version using IBM Installation Manager with a response file.

Before you begin

Tip: As an alterative to the procedure that is described in this article, Installation Manager allows you to use the **`updateAll`** command in a response file or on the command line to search for and update all installed packages. Use this command only if you have full control over which fixes are contained in the targeted repositories. If you create and point to a set of custom repositories that include only the specific fixes that you want to install, you should be able to use this command confidently. If you enable searching service repositories or install fixes directly from other live web-based repositories, then you might not want to select this option so that you can select only the fixes that you want to install using the -installFixes option with the **`install`** command on the command line or the installFixes attribute in a response file.

Procedure

1. For a list of interim fixes and fix packs that are available for WebSphere eXtreme Scale and specific information about each fix, perform the following actions.
   a. Go to Fix Central.
   b. Select **WebSphere** as the product group.
   c. Select WebSphere eXtreme Scale as the product.
   d. Select **8.x** as the installed version.
   e. Select your operating system as the platform, and click **Continue**.
   f. Select **Browse for fixes**, and click **Continue**.
   g. Click **More Information** under each fix to view information about the fix.
   h. **Recommendation:** Make a note of the name of the fix pack that you would like to install.
2. Update WebSphere eXtreme Scale with the fix pack using the following procedure.
   - Download the file that contains the fix pack from Fix Central, and use local updating.

     You can download a compressed file that contains the fix pack from Fix Central. Each compressed fix-pack file contains an Installation Manager repository for the fix pack and usually has a .zip extension. After downloading and extracting the fix-pack file, use Installation Manager to update WebSphere eXtreme Scale with the fix pack.

a. To download the fix pack, perform the following actions:
   1) Go to Fix Central.
   2) Select **WebSphere** as the product group.
   3) Select **WebSphere eXtreme Scale** as the product.
   4) Select **8.6** as the installed version.
   5) Select your operating system as the platform, and click **Continue**.
   6) Select **Browse for fixes**, and click **Continue**.
   7) Select the fix pack that you want to download, and click **Continue**.
   8) Select your download options, and click **Continue**.
   9) Click **I agree** to agree to the terms and conditions.
   10) Click **Download now** to download the fix pack.
   11) Transfer the compressed file in binary format to the system on which it will be installed.
   12) Extract the compressed repository files to a directory on your system.

b. Perform the following actions:
   1) Log on to your system.
   2) If the repository requires a user name and password, create a keyring file to access this repository.

      For more information on creating a keyring file for Installation Manager, read the IBM Installation Manager Version 1.5 Information Center.

      Tip: When creating a keyring file, append `/repository.config` at the end of the repository URL location if the **`imutilsc`** command is unable to find the URL that is specified.
   3) To stop all processes that are running in your stand-alone eXtreme Scale environment, see "Stopping stand-alone servers" on page 472To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.
   4) Change to the *`Installation_Manager_binaries`*`/eclipse/tools` directory, where *`Installation_Manager_binaries`* is the installation root directory for the Installation Manager.
   5) Install the fix pack using a response file.

      For example:

      – Windows **Administrator or non-administrator:**

      ```
      imcl.exe -acceptLicense
        input C:\temp\update_response_file.xml
        -log C:\temp\update_log.xml
        -keyring C:\IM\im.keyring
      ```

      – UNIX Linux **Administrator:**

      ```
      ./imcl -acceptLicense
        input /var/temp/update_response_file.xml
        -log /var/temp/update_log.xml
        -keyring /var/IM/im.keyring
      ```

      – UNIX Linux **Non-administrator:**

```
./imcl -acceptLicense
  input user_home/var/temp/update_response_file.xml
  -log user_home/var/temp/update_log.xml
  -keyring user_home/var/IM/im.keyring
```

Uninstalling fix packs using IBM Installation Manager

You can use IBM Installation Manager to rollback WebSphere eXtreme Scale product offerings to an earlier version. You can uninstall fix packs from the GUI, the command line, or using response files.

Uninstalling fix packs using the GUI

You can roll back this product to an earlier version using the IBM Installation Manager GUI.

Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

About this task

Restriction: You cannot use the Installation Manager to roll back an installation and add or remove a feature.

Procedure

1. Stop all processes that are running in your environment.
   - To stop all processes that are running in your stand-alone eXtreme Scale environment, see “Stopping stand-alone servers” on page 472.
   - To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.
2. Start Installation Manager.
3. Click **Roll Back**.
4. Select the package group to roll back.
5. Click **Next**.
6. Select the version to which you want to roll back under.
7. Click **Next**.
8. Review the summary information, and click **Roll Back**.
   - If the rollback is successful, the program displays a message indicating that the rollback is successful.
   - If the rollback is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

Uninstalling fix packs using the command line

You can roll back this product to an earlier version using IBM Installation Manager from the command line.

Before you begin

Restriction: In order to use this procedure, you must have Installation Manager Version 1.5 or later installed on your system.

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

About this task

Restriction: You cannot use the Installation Manager to roll back an installation and add or remove the full WebSphere Application Server profile feature .

Procedure

1. Optional: If the repository requires a user name and password, create a keyring file to access this repository.

   For more information on creating a keyring file for Installation Manager, read the IBM Installation Manager Version 1.5 Information Center.

   Tip: When creating a keyring file, append `/repository.config` at the end of the repository URL location if the **`imutilsc`** command is unable to find the URL that is specified.
2. Log on to your system.
3. Stop all processes that are running in your environment.
   - To stop all processes that are running in your stand-alone eXtreme Scale environment, see "Stopping stand-alone servers" on page 472.
   - To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.
4. Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager.
5. Use the **`imcl`** command to roll back the product.

   UNIX Linux

   ```
   ./imcl rollback offering_ID_offering_version
     -repositories source_repository
     -installationDirectory installation_directory
     -preferences preference_key=value
     -properties property_key=value
     -keyring keyring_file -password password
     -acceptLicense
   ```

   Windows

   ```
   imcl.exe rollback offering_ID_offering_version
     -repositories source_repository
     -installationDirectory installation_directory
     -preferences preference_key=value
     -properties property_key=value
     -keyring keyring_file -password password
     -acceptLicense
   ```

   Tips:
   - The *offering_ID* is the offering ID that is listed in Offering IDs for WebSphere eXtreme Scale product offerings.

- The *offering_version*, which optionally can be attached to the offering ID with an underscore, is a specific version of the offering to which to roll back (`8.5.0.20110503_0200` for example).
  - If *offering_version* is **not** specified, the installation rolls back to the previously installed version of the offering and **all** interim fixes for that version are installed.
  - If *offering_version* is specified, the installation rolls back to the specified earlier version of the offering and **no** interim fixes for that version are installed.

  The offering version can be found attached to the end of the offering ID with an underscore in the Package section of the report that is generated when you run the **`historyInfo`** or **`genHistoryReport`** command from the *`app_server_root`*`/bin` directory.

For more information on using Installation Manager, read the IBM Installation Manager Version 1.5 Information Center.

6. Optional: List all installed packages to verify the roll back.

   UNIX Linux

   ```
   ./imcl listInstalledPackages -long
   ```

   Windows

   ```
   imcl.exe listInstalledPackages -long
   ```

Uninstalling fix packs using response files

You can roll back this product to an earlier version using IBM Installation Manager with a response file.

Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the saved files, Installation Manager requires access to the repository that was used to install the earlier version.

About this task

Restriction: You cannot use the Installation Manager to roll back an installation and add or remove the full WebSphere Application Server profile feature .

Procedure

1. Optional: If the repository requires a username and password, create a keyring file to access this repository.

   For more information on creating a keyring file for Installation Manager, read the IBM Installation Manager Version 1.5 Information Center.

   Tip: When creating a keyring file, append `/repository.config` at the end of the repository URL location if the **`imutilsc`** command is unable to find the URL that is specified.
2. Log on to your system.
3. Stop all processes that are running in your environment.
   - To stop all processes that are running in your stand-alone eXtreme Scale environment, see “Stopping stand-alone servers” on page 472.

- To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.

4. Use a response file to roll back the product.

   Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and roll back the product.

   For example:

   - Windows **Administrator or non-administrator:**

     ```
     imcl.exe
       input C:\temp\rollback_response_file.xml
       -log C:\temp\rollback_log.xml
       -keyring C:\IM\im.keyring
     ```

   - UNIX Linux **Administrator:**

     ```
     ./imcl
       input /var/temp/rollback_response_file.xml
       -log /var/temp/rollback_log.xml
       -keyring /var/IM/im.keyring
     ```

   - UNIX Linux **Non-administrator:**

     ```
     ./imcl
       input user_home/var/temp/rollback_response_file.xml
       -log user_home/var/temp/rollback_log.xml
       -keyring user_home/var/IM/im.keyring
     ```

   Note: The program might write important post-installation instructions to standard output.

   For more information on using Installation Manager, read the IBM Installation Manager Version 1.5 Information Center.

5. Optional: List all installed packages to verify the roll back.

   UNIX Linux

   ```
   ./imcl listInstalledPackages -long
   ```

   Windows

   ```
   imcl.exe listInstalledPackages -long
   ```

Using the Update Installer to install maintenance packages

Use the IBM Update Installer to update your WebSphere eXtreme Scale or WebSphere eXtreme Scale Client environment with various types of maintenance, such as interim fixes, fix packs, and refresh packs.

About this task

Use the IBM Update Installer to install and apply various types of maintenance packages for WebSphere eXtreme Scale or WebSphere eXtreme Scale Client. Because the Update Installer undergoes regular maintenance, you must use the most current version of the tool.

Important: If a non-root user launches the Update Installer program, then that user account must be able to run the **`slibclean`** command; otherwise, a root user must run the **`slibclean`** command whenever the Update Installer program is used.

Procedure

1. Stop all processes that are running in your environment.

- To stop all processes that are running in your stand-alone eXtreme Scale environment, see "Stopping stand-alone servers" on page 472.
- To stop all processes that are running in your WebSphere Application Server environment, see Command-line utilities.

2. Download the latest version of the Update Installer. See Recommended fixes for more information.
3. Install the Update Installer. See Installing the Update Installer for WebSphere Software in the WebSphere Application Server Information Center for more information.
4. Download into the *updi_root*`/maintenance` directory the maintenance packages that you intend to install. See the Support site for more information.
5. Use the Update Installer to install the interim fix, fix pack, or refresh pack. You can install the maintenance package by running the graphical user interface (GUI), or by running the Update Installer in silent mode.

   Run the following command from the *updi_root* directory to start the GUI:

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

   Run the following command from the *updi_root* directory to run the Update Installer in silent mode:

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

Results

If the installation process fails, see the temporary log file, which is in the *updi_root*`/logs/update/tmp` directory. The Update Installer creates the *install_root*`/logs/update/`*maintenance_package*`.install` directory in which the installation log files are located.

Related concepts:

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

xsadmin tool to xscmd tool migration

In previous releases, the **xsadmin** tool was a sample command-line utility to monitor the state of the environment. The **xscmd** tool has been introduced as an officially supported administrative and monitoring command-line tool. If you were previously using the **xsadmin** tool, consider migrating your commands to the new **xscmd** tool.

xsadmin and xscmd command equivalents

Important: The **xsadmin** utility has now been deprecated. Use the **xscmd** utility instead. The **xscmd** utility is provided as a supported utility for monitoring and administering your environment. For more information, see "Administering with the **xscmd** utility" on page 482.

Table 8. Arguments for the **xsadmin** *utility and* **xscmd** *equivalent commands*. Some **xscmd** commands have a short form and a long form. The short form commands have one dash (-), and the long form commands have two dashes (--). You can use either form interchangeably.

xsadmin Command Line Argument	**xscmd** Equivalent Command	**xscmd** Command Parameters
-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 showPlacement -container***containerName* • **-c showPlacement -server** *serverName*	**-e, -i, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms**
-continuous	n/a	n/a
-coregroups	• • **-c listCoreGroupMembers -cg** *core_group*	n/a
-dismissLink *<catalog_service_domain>*	**-c dismissLink**	• **-fd** *<foreignCatalogServiceDomain>* • **--foreignCatalogServiceDomain** *<foreignCatalogServiceDomain>*
-dmgr	n/a - this argument is automatically determined with **xscmd**	n/a
-empties	arg specific to a new command	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

Table 8. Arguments for the **xsadmin** *utility and* **xscmd** *equivalent commands (continued).* Some **xscmd** commands have a short form and a long form. The short form commands have one dash (-), and the long form commands have two dashes (--). You can use either form interchangeably.

xsadmin Command Line Argument	**xscmd Equivalent Command**	**xscmd Command Parameters**
-fst	• **-st** *<shard_type>* • **--shardType** *<shard_type>* Shard values: P=primary A=asyncReplica S=syncReplica	n/a
-fz	• **-z** • **--zone**	n/a
-force	arg specific to a new command	
-g	• **-g** • **--objectGrid**	n/a
-getstatsspec	**-c getStatsSpec**	n/a
-getTraceSpec	**-c getTraceSpec**	n/a
-h	You can run help with or without a specific command name: • **-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, -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**

Table 8. Arguments for the **xsadmin** *utility and* **xscmd** *equivalent commands (continued).* Some **xscmd** commands have a short form and a long form. The short form commands have one dash (-), and the long form commands have two dashes (--). You can use either form interchangeably.

xsadmin Command Line Argument	**xscmd Equivalent Command**	**xscmd Command Parameters**
-profile	To save the current security settings as a security profile: • **-ssp** *profile_name* • **--saveSecProfile** *profile_name* To use a specified security profile: • **-sp** *profile_name* • **--securityProfile** *profile_name*	
-quorumstatus	**-c showQuorumStatus**	n/a
-releaseShard *<container_server_name> <objectgrid_name> <map_set_name> <partition_name>*	**-c releaseShard**	**-c, -g, -ms, -p**
-reserved	• **-sf R** • **--shardFilter R**	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

Table 8. Arguments for the **xsadmin** *utility and* **xscmd** *equivalent commands (continued).* Some **xscmd** commands have a short form and a long form. The short form commands have one dash (-), and the long form commands have two dashes (--). You can use either form interchangeably.

`xsadmin` Command Line Argument	`xscmd` Equivalent Command	`xscmd` Command Parameters
`-unassigned`	`-c showPlacement -sf U`	`-e, -i, , -st, -snp, -ct, -s, -p, -hf, -z, -g, -m, -ms`
`-username`	• `-user` • `--username`	n/a
`-v`	• `-v` • `--verbose`	n/a
`-xml`	`-c showPlacement`	n/a

Related tasks:

"Configuring security profiles for the **xscmd** utility" on page 607
By creating a security profile, you can use saved security parameters to use the **xscmd** utility with secure environments.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

"Monitoring with the **xscmd** utility" on page 535
The **xscmd** utility replaces the **xsadmin** sample utility as a fully supported monitoring and administration tool. With the **xscmd** utility, you can display textual information about your WebSphere eXtreme Scale topology.

Deprecated properties and APIs

The following list of properties and APIs were deprecated in the specified releases. Use the recommended migration action to determine how to update your configuration.

7.1.1+
Deprecated items in Version 7.1.1

Table 9. Deprecated properties and APIs

Deprecation	Recommended migration action
com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener class This class was used to propagate successful ObjectGrid transaction commit processes to other WebSphere application servers hosting the same ObjectGrid instance, based upon the ObjectGrid name.	7.1.1+ The TranPropListener interface has been replaced by the JMSObjectGridEventListener interface, which is a JMS-based implementation of the ObjectGridEventListener interface. It supports client-side, near cache invalidation and peer-to-peer replication.
com.ibm.websphere.objectgrid.plugins.OptimisticCallback class This class was used to provide optimistic comparison operations for the values of a map.	7.1.1+ The OptimisticCallback plug-in has been replaced by the ValueDataSerializer.Versionable interface, which you can implement when you use the DataSerializer plug-in with the COPY_TO_BYTES copy mode or when you use the @Version annotation with the EntityManager API. See the API documentation for more information.
com.ibm.websphere.objectgrid.plugins.NoVersioningOptimisticCallback plug-in This plug-in was used for optimistic locking without doing version checking. With this built-in OptimisticCallback handler, the loader handled version checking, but optimistic locking was used to ensure that committed data is always returned on a read.	7.1.1+ The NoVersioningOptimisticCallback interface extends the OptimisticCallback interface. Therefore, use the pessimistic locking strategy with a default transaction isolation of READ_COMMITTED or lower. See Tuning locking performance for more information.

Table 9. Deprecated properties and APIs (continued)

Deprecation	Recommended migration action
com.ibm.websphere.objectgrid.plugins.ObjectTransformer class This plug-iin was used to serialize, deserialize, and copy objects into the cache.	7.1.1+ The ObjectTransformer interface has been replaced by the DataSerializer plug-ins, which you can use to efficiently store arbitrary data in WebSphere eXtreme Scale so that existing product APIs can efficiently interact with your data.
com.ibm.websphere.objectgrid.BackingMap.setMapEventListeners method This method was used to set the list of MapEventListener objects.	7.1.1+ Use either the addMapEventListener(EventListener) or removeMapEventListener(EventListener) methods to add or remove event listeners from a backing map.
com.ibm.websphere.objectgrid.ObjectGrid.setEventListeners method This method was used to overwrite the current list of ObjectGridEventListener objects and replace it with the supplied list of ObjectGridEventListeners objects.	7.1.1+ Use either the addEventListener(EventListener) or removeEventListener(EventListener) methods to add or remove event listeners or life cycle listeners from the data grid.

7.1.1+
Stabilized features in Version 7.1.1

If a feature is listed as stabilized, IBM does not currently plan to deprecate or remove this capability in a subsequent release of the product; but future investment will be focused on the alternative function. Users do not need to change any existing applications and scripts that use a stabilized function; but they should consider using the strategic alternative for new applications.

Table 10. Deprecated properties and APIs

Stabilized feature	Recommended migration action
xsadmin The xsadmin utility is provided as a sample of how you can create custom utilities for your deployment.	7.1.1+ Use the `xscmd` utility to complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Deprecated items in Version 7.1

Table 11. Deprecated properties and APIs

Deprecation	Recommended migration action
catalog.services.cluster cell and server property: This custom property was used to define a group of catalog servers in the WebSphere Application Server configuration.	This custom property is deprecated starting in the Version 7.1 release. Create a catalog service domain in the WebSphere Application Server administrative console, which creates the same configuration as using the custom property. See for more information.
CoreGroupServicesMBean MBean and interface	This MBean is deprecated starting in the Version 7.1 release. Use the CatalogServiceManagementMBean instead.
ServerMBean.updateTraceSpec() MBean operation	This operation is deprecated starting in the Version 7.1 release. Use the TraceSpec attribute on the DynamicServerMBean instead.
CoreGroupServicesMBean MBean	This MBean is deprecated starting in the Version 7.1 release. Use the CatalogServiceManagementMbean MBean instead.
ServiceUnavailableException exception	This exception is deprecated starting in the Version 7.1 release. Use the TargetNotAvailableException exception instead.

Table 11. Deprecated properties and APIs (continued)

Deprecation	Recommended migration action
Partitioning facility (WPF): The partitioning facility is a set of programming APIs that allow Java EE applications to support asymmetric clustering.	The capabilities of WPF can be alternatively realized in WebSphere eXtreme Scale.
StreamQuery: A continuous query over in-flight data stored in ObjectGrid maps.	None
Static grid configuration: A static, cluster-based topology using the cluster deployment XML file.	Replaced with the improved, dynamic deployment topology for managing large data grids.
Deprecated system properties: System properties to specify the server and client properties files are deprecated.	You can still use these arguments, but change your system properties to the new values. **`-Dcom.ibm.websphere.objectgrid.CatalogServerProperties`** The property was deprecated in WebSphere eXtreme Scale Version 7.0. Use the **`-Dobjectgrid.server.props`** property. **`-Dcom.ibm.websphere.objectgrid.ClientProperties`** The property was deprecated in WebSphere eXtreme Scale Version 7.0. Use the **`-Dobjectgrid.client.props`** property. **`-Dobjectgrid.security.server.prop`** The property was deprecated in WebSphere eXtreme Scale Version 6.1.0.3. Use the **`-Dobjectgrid.server.prop`** property. **`-serverSecurityFile`** This argument was deprecated in WebSphere eXtreme Scale Version 6.1.0.3. This option is passed into the `startOgServer` script. Use the **`-serverProps`** argument.

Removed properties and APIs

If you are migrating your configuration from an earlier release of WebSphere eXtreme Scale, some features might be removed from this and earlier releases. Use the recommended migration action to determine how to update your configuration.

If a feature is listed as deprecated in Deprecated features, IBM® might remove this capability in a subsequent release of the product. Future investment will be focused on the strategic function listed under Recommended Migration Actions in Deprecated features. Typically, a feature is not removed until at least two major releases or three full years (whichever time period is longer) after the release in which that feature is deprecated. In rare cases, it might become necessary to remove features sooner; such cases are indicated clearly and explicitly in the descriptions of these deprecated features in Deprecated features.

The following information describes removed features, APIs, scripting interfaces, tools, and publicly exposed configuration data. Where possible, the recommended replacement is identified.

Chapter 6. Configuring

You can configure WebSphere eXtreme Scale to run in a stand-alone environment, or you can configure eXtreme Scale to run in an environment with WebSphere Application Server or WebSphere Application Server Network Deployment. For a WebSphere eXtreme Scale deployment to pick up configuration changes on the server side of the data grid, you must restart processes to make these changes take effect rather than being applied dynamically. However, on the client side, although you may not alter the configuration settings for an existing client instance, you can create a new client with the settings you require by using an XML file or doing so programmatically. When creating a client, you can override the default settings that come from the current server configuration.

Configuration methods

You can configure most aspects of the product with XML files and property files. You can also use programmatic methods, including application and system programming interfaces, plug-ins, and managed beans.

About this task

Use the following files to create a basic configuration:

Server properties file
Use the server properties file to define settings for catalog and container servers, such as trace, logging, security, ports, and so on. You can pass a server properties file to the server start script, put the file in your classpath, or define the file with system properties.

Client properties file
Use the client properties file to set properties on your clients, including ports and security settings. You can specify the client properties file to use with a system property, by placing the file in the classpath, or by using the ClientClusterContext.getClientProperties method.

ObjectGrid descriptor XML file
The ObjectGrid descriptor XML file describes the data grid and map configuration. Specify the file to use with the server start script for stand-alone configurations, or add the file to the application module for WebSphere Application Server configurations.

Deployment policy descriptor XML file
The deployment policy XML file controls shard and placement of data on the various container servers in the configuration. Specify the file to use with the server start script for stand-alone configurations, or add the file to the application module for WebSphere Application Server configurations.

Related concepts:

"Embedded server API" on page 479
WebSphere eXtreme Scale includes application programming interfaces (APIs) and system programming interfaces for embedding eXtreme Scale servers and clients within your existing Java applications.

Interacting with an ObjectGrid using the ObjectGridManager interface
The ObjectGridManagerFactory class and the ObjectGridManager interface provide a mechanism to create, access, and add data to ObjectGrid instances. The ObjectGridManagerFactory class is a static helper class to access the ObjectGridManager interface, a singleton. The ObjectGridManager interface includes several convenience methods to create instances of an ObjectGrid object. The ObjectGridManager interface also facilitates creation and caching of ObjectGrid instances that can be accessed by several users.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Related information:

ObjectGridManager interface

ClientClusterContext interface

DeploymentPolicy interface

Operational checklist

Use the operational checklist to prepare your environment for deploying WebSphere eXtreme Scale.

Table 12. Operational checklist

Checklist item	For more information

If you are using AIX, tune the following operating system settings:
 	
TCP_KEEPINTVL
The TCP_KEEPINTVL setting is part of a socket keep-alive protocol that enables detection of network outage. The property specifies the interval between packets that are sent to validate the connection. When you are using WebSphere eXtreme Scale, set the value to `10`. To check the current setting, run the following command:
`# no –o tcp_keepintvl`
To change the current setting, run the following command:
`# no –o tcp_keepintvl=10`
The TCP_KEEPINTVL setting is in half seconds.
 	
TCP_KEEPINIT
The TCP_KEEPINIT setting is part of a socket keep-alive protocol that enables detection of network outage. The property specifies the initial timeout value for TCP connection. When you are using WebSphere eXtreme Scale, set the value to `40`. To check the current setting, run the following commands:
`# no –o tcp_keepinit`
To change the current setting, run the following command:
`# no –o tcp_keepinit=40`
The TCP_KEEPINIT setting is in half seconds.
	• For AIX tuning information, see Tuning AIX systems.

Update the `orb.properties` file to modify the transport behavior of the grid. The `orb.properties` file is in the `java/jre/lib` directory.	“ORB properties” on page 566

Table 12. Operational checklist (continued)

Checklist item	For more information
Use parameters in the **startOgServer** script. In particular, use the following parameters: • Set heap settings with the **-jvmArgs** parameter. • Set application class path and properties with the **-jvmArgs** parameter. • Set **-jvmArgs** parameters for configuring agent monitoring. **Port settings** WebSphere eXtreme Scale has to open ports for communications for some transports. These ports are all dynamically defined. However, if a firewall is in use between containers then you must specify the ports. Use the following information about the ports: **Listener port** You can use the **-listenerPort** argument to specify the port that is used for communication between processes. **Core group port** You can use the **-haManagerPort** argument to specify the port that is used for failure detection. This argument is the same as peerPort. Note that core groups do not need to communicate across zones, so you might not need to set this port if the firewall is open to all the members of a single zone. **JMX service port** You can use the **-JMXServicePort** argument to specify the port that the JMX service should use. **SSL port** Passing `-Dcom.ibm.CSI.SSLPort=1234` as a **-jvmArgs** argument sets the SSL port to `1234`. The SSL port is the secure port peer to the listener port. **Client port** Used in the catalog service only. You can specify this value with the **-catalogServiceEndPoints** argument. The format of the value of this parameter is in the format: `serverName:hostName:clientPort:peerPort`	"**startOgServer** script" on page 466
Verify that security settings are configured correctly: • Transport (SSL) • Application (Authentication and Authorization) To verify your security settings, you can try to use a malicious client to connect to your configuration. For example, when the SSL-Required setting is configured, a client that has a TCP_IP setting with or a client with the wrong trust store should not be able to connect to the server. When authentication is required, a client with no credential, such as a user ID and password, should not be able to connect to the sever. When authorization is enforced, a client with no access authorization should not be granted the access to the server resources.	"Security integration with external providers" on page 597
Choose how you are going to monitor your environment. • **xscmd** tool: – The JMX ports of the catalog servers need to be visible to the**xscmd** tool. The container server ports also need to be accessible for some commands that gather information from the containers. • Monitoring console: With the monitoring console, you can chart current and historical statistics. • Vendor monitoring tools: – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• "Monitoring with the **xscmd** utility" on page 535 • "Java Management Extensions (JMX) security" on page 595 • "Monitoring with the web console" on page 514 • "Monitoring with the IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale" on page 551 • "Monitoring eXtreme Scale with Hyperic HQ" on page 560 • "Monitoring eXtreme Scale applications with CA Wily Introscope" on page 557

Configuring data grids

Use an ObjectGrid descriptor XML file to configure data grids, backing maps, plug-ins, and so on. To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API. For a distributed topology, you need an ObjectGrid descriptor XML file and a deployment policy XML file.

Related concepts:

"Embedded server API" on page 479
WebSphere eXtreme Scale includes application programming interfaces (APIs) and system programming interfaces for embedding eXtreme Scale servers and clients within your existing Java applications.

Interacting with an ObjectGrid using the ObjectGridManager interface
The ObjectGridManagerFactory class and the ObjectGridManager interface provide a mechanism to create, access, and add data to ObjectGrid instances. The ObjectGridManagerFactory class is a static helper class to access the ObjectGridManager interface, a singleton. The ObjectGridManager interface includes several convenience methods to create instances of an ObjectGrid object. The ObjectGridManager interface also facilitates creation and caching of ObjectGrid instances that can be accessed by several users.

"Distributing changes between peer JVMs" on page 267
The LogSequence and LogElement objects distribute changes between peer JVMs and communicate the changes that have occurred in an eXtreme Scale transaction with an ObjectGridEventListener plug-in.

"JMS event listener" on page 271
The JMSObjectGridEventListener is designed to support client-side near cache invalidation and a peer-to-peer replication mechanism. It is a Java Message Service (JMS) implementation of the ObjectGridEventListener interface.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Related information:

ObjectGridManager interface

ClientClusterContext interface

DeploymentPolicy interface

Configuring local deployments

A local in-memory eXtreme Scale configuration can be created by using an ObjectGrid descriptor XML file or APIs.

About this task

To create a local deployment, you create an ObjectGrid descriptor XML file and then pass the file to the createObjectGrid methods in the ObjectGridManager interface.

As an alternative, you can also create the entire deployment programmatically with the ObjectGridManager interface.

Procedure

1. Create an ObjectGrid descriptor XML file.

   The following `companyGrid.xml` file is an example of an ObjectGrid descriptor XML. The first few lines of the file include the required header for each ObjectGrid XML file. The file defines an ObjectGrid instance named "CompanyGrid" and several BackingMaps named "Customer," "Item," "OrderLine," and "Order."

   companyGrid.xml file

   ```
   <?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. Pass the XML file to one of the createObjectGrid methods in the ObjectGridManager interface.

   The following code sample validates the `companyGrid.xml` file against the XML schema, and creates the ObjectGrid instance named "CompanyGrid." The newly created ObjectGrid instance is not cached.

   ```
   ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
   ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid",
    new URL("file:etc/test/companyGrid.xml"), true, false);
   ```

What to do next

See Creating ObjectGrid instances with the ObjectGridManager interface for more information about defining all of the maps programmatically with the createObjectGrid methods on the ObjectGridManager interface.

Related concepts:

"Distributing changes between peer JVMs" on page 267
The LogSequence and LogElement objects distribute changes between peer JVMs and communicate the changes that have occurred in an eXtreme Scale transaction with an ObjectGridEventListener plug-in.

"JMS event listener" on page 271
The JMSObjectGridEventListener is designed to support client-side near cache invalidation and a peer-to-peer replication mechanism. It is a Java Message Service (JMS) implementation of the ObjectGridEventListener interface.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Configuring evictors with XML files

In addition to programmatically setting a time-to-live (TTL) evictor with the BackingMap interface, you can use an XML file to configure an evictor on each BackingMap instance.

Before you begin

Before you begin, decide on the type of evictor you are going to use:

- **The default time-based TTL evictor:** The default evictor uses a time-to-live (TTL) eviction policy for each BackingMap instance.
- **A pluggable evictor mechanism:** Pluggable evictors typically use an eviction policy that is based on the number of entries instead of on time.

Set the evictor settings before you start your container servers.

Procedure

- To set the default TTL evictor, add the **`ttlEvictorType`** attribute to the ObjectGrid descriptor XML file.

  The following example shows that the map1 BackingMap instance uses a NONE TTL evictor type. The map2 BackingMap instance uses either a LAST_ACCESS_TIME or LAST_UPDATE_TIME TTL evictor type. Specify only one or the other of these settings. The map2 BackingMap instance has a time-to-live value of 1800 seconds, or 30 minutes. The map3 BackingMap instance is defined to use a CREATION_TIME TTL evictor type and has a time-to-live value of 1200 seconds, or 20 minutes.

```
<?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>
```

Figure 29. Enable TimeToLive evictor with XML

- To set a pluggable evictor, use the following 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="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>
```

Figure 30. Plugging in an evictor using XML

Related concepts:

"Distributing changes between peer JVMs" on page 267
The LogSequence and LogElement objects distribute changes between peer JVMs and communicate the changes that have occurred in an eXtreme Scale transaction with an ObjectGridEventListener plug-in.

"JMS event listener" on page 271
The JMSObjectGridEventListener is designed to support client-side near cache invalidation and a peer-to-peer replication mechanism. It is a Java Message Service (JMS) implementation of the ObjectGridEventListener interface.

Evictors
Evictors remove data from the data grid. You can either set a time-based evictor or because evictors are associated with BackingMaps, use the BackingMap interface to specify the pluggable evictor.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Configuring a locking strategy in the ObjectGrid descriptor XML file

You can define an optimistic, a pessimistic, or no locking strategy on each BackingMap in the WebSphere eXtreme Scale configuration.

Before you begin

Decide which locking strategy you want to use. For more information, see Locking strategies.

About this task

You can configure each BackingMap instance to use one of the following locking strategies:

- Optimistic locking mode (default)
- Pessimistic locking mode
- None

Procedure

- **Configure a pessimistic locking strategy**
  - With the lockStrategy attribute in the ObjectGrid descriptor XML file:

```
<?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>
```

- **Configure an optimistic locking strategy**
  - With the lockStrategy attribute in the ObjectGrid descriptor XML file:

```
<?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>
```

- **Configure a no locking strategy**
  - With the lockStrategy attribute in the ObjectGrid descriptor XML file:

```
<?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>
```

Related concepts:

"Distributing changes between peer JVMs" on page 267
The LogSequence and LogElement objects distribute changes between peer JVMs and communicate the changes that have occurred in an eXtreme Scale transaction with an ObjectGridEventListener plug-in.

"JMS event listener" on page 271
The JMSObjectGridEventListener is designed to support client-side near cache invalidation and a peer-to-peer replication mechanism. It is a Java Message Service

(JMS) implementation of the ObjectGridEventListener interface.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Configuring the lock timeout value in the ObjectGrid descriptor XML file

The lock timeout value on a BackingMap instance is used to ensure that an application does not wait endlessly for a lock mode to be granted because of a deadlock condition that occurs due to an application error.

Before you begin

To configure the lock timeout value, the locking strategy must be set to either OPTIMISTIC or PESSIMISTIC. See “Configuring a locking strategy in the ObjectGrid descriptor XML file” on page 264 for more information.

About this task

When a LockTimeoutException exception occurs, the application must determine if the timeout is occurring because the application is running slower than expected, or if the timeout occurred because of a deadlock condition. If an actual deadlock condition occurred, then increasing the lock wait timeout value does not eliminate the exception. Increasing the timeout results in the exception taking longer to occur. However, if increasing the lock wait timeout value does eliminate the exception, then the problem occurred because the application was running slower than expected. The application in this case must determine why performance is slow.

To prevent deadlocks from occurring, the lock manager has a default timeout value of 15 seconds. If the timeout limit is exceeded, a LockTimeoutException exception occurs. If your system is heavily loaded, the default timeout value might cause the LockTimeoutException exceptions to occur when no deadlock exists. In this situation, you can increase the lock timeout value programmatically or in the ObjectGrid descriptor XML file.

Procedure

Configure the lock timeout value with the lockTimeout attribute in the ObjectGrid descriptor XML file.

```
<?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"
                lockTimeout="60"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Configuring peer-to-peer replication with JMS

The Java Message Service (JMS) based peer-to-peer replication mechanism is used in both the distributed and local WebSphere eXtreme Scale environment. JMS is a core-to-core replication process and allows data updates to flow among local ObjectGrids and distributed ObjectGrids. For example, with this mechanism you can move data updates from a distributed eXtreme Scale data grid to a local eXtreme Scale grid, or from a grid to another grid in a different system domain.

Before you begin

The JMS-based peer-to-peer replication mechanism is based on the built-in JMS-based ObjectGridEventListener, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener. For detailed information regarding enabling peer-to-peer replication mechanism, see "JMS event listener" on page 271.

See "Configuring Java Message Service (JMS)-based client synchronization" on page 355 for more information.

The following is an XML configuration example to enable a peer-to-peer replication mechanism on an eXtreme Scale configuration:

peer-to-peer replication configuration - XML example
```
<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>
```

Related concepts:

"Distributing changes between peer JVMs"
The LogSequence and LogElement objects distribute changes between peer JVMs and communicate the changes that have occurred in an eXtreme Scale transaction with an ObjectGridEventListener plug-in.

"JMS event listener" on page 271
The JMSObjectGridEventListener is designed to support client-side near cache invalidation and a peer-to-peer replication mechanism. It is a Java Message Service (JMS) implementation of the ObjectGridEventListener interface.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Distributing changes between peer JVMs

The LogSequence and LogElement objects distribute changes between peer JVMs and communicate the changes that have occurred in an eXtreme Scale transaction with an ObjectGridEventListener plug-in.

For more information about how Java Message Service (JMS) can be used to distribute transactional changes, see JMS for distributed transaction changes.

A prerequisite is that the ObjectGrid instance must be cached by the ObjectGridManager. See createObjectGrid methods for more information. The cacheInstance boolean value must be set to true.

It is not necessary for you to implement this mechanism. There is a built-in peer-to-peer replication mechanism for you to use this function. See "Configuring peer-to-peer replication with JMS" on page 267.

The objects provide a means for an application to easily publish changes that have occurred in an ObjectGrid using a message transport to peer ObjectGrids in remote Java virtual machines and then apply those changes on that JVM. The LogSequenceTransformer class is critical to enabling this support. This article examines how to write a listener using a Java Message Service (JMS) messaging system for propagating the messages. To that end, eXtreme Scale supports transmitting LogSequences that result from an eXtreme Scale transaction commit across WebSphere Application Server cluster members with an IBM-provided plug-in. This function is not enabled by default, but can be configured to be operational. However, when either the consumer or producer is not a WebSphere Application Server, using an external JMS messaging system might be required.

Implementing the mechanism

The LogSequenceTransformer class, and the ObjectGridEventListener, LogSequence and LogElement APIs allow any reliable publish-and-subscribe to be used to distribute the changes and filter the maps you want to distribute. The snippets in this topic show how to use these APIs with JMS to build a peer-to-peer ObjectGrid shared by applications that are hosted on a diverse set of platforms sharing a common message transport.

Initialize the plug-in

The ObjectGrid calls the initialize method of the plug-in, part of the ObjectGridEventListener interface contract, when the ObjectGrid starts. The initialize method must obtain its JMS resources, including connections, sessions, and publishers, and start the thread that is the JMS listener.

The following examples show the initialize method:

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);
    }
}
```

The code to start the thread uses a Java 2 Platform, Standard Edition (Java SE) thread. If you are running a WebSphere Application Server Version 6.x or a WebSphere Application Server Version 5.x Enterprise server, use the asynchronous bean application programming interface (API) to start this daemon thread. You can also use the common APIs. Following is an example replacement snippet showing the same action using a work manager:

```
// start the listener thread.
listenerRunning = true;
workManager.startWork(this, true);
```

The plug-in must also implement the Work interface instead of the Runnable interface. You also need to add a release method to set the listenerRunning variable to false. The plug-in must be provided with a WorkManager instance in its constructor or by injection if using an Inversion of Control (IoC) container.

Transmit the changes

The following is a sample transactionEnd method for publishing the local changes that are made to an ObjectGrid. This sample uses JMS, although you can use any message transport that is capable of reliable publish-and subscribe-messaging.

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);
        }
    }
```

This method uses several instance variables:

- `jmsSession` variable: A JMS session that is used to publish messages. It is created when the plug-in initializes.
- `mode` variable: The distribution mode.
- `publishMaps` variable: A set that contains the name of each map with changes to publish. If the variable is empty, then all the maps are published.
- `publisher` variable: A TopicPublisher object that is created during the plug-in initialize method

Receive and apply update messages

Following is the run method. This method runs in a loop until the application stops the loop. Each loop iteration attempts to receive a JMS message and apply it to the ObjectGrid.

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");
   }
```

Related tasks:

"Configuring data grids" on page 261
Use an ObjectGrid descriptor XML file to configure data grids, backing maps, plug-ins, and so on. To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API. For a distributed topology, you need an ObjectGrid descriptor XML file and a deployment policy XML file.

"Configuring local deployments" on page 261
A local in-memory eXtreme Scale configuration can be created by using an ObjectGrid descriptor XML file or APIs.

"Configuring a locking strategy in the ObjectGrid descriptor XML file" on page 264
You can define an optimistic, a pessimistic, or no locking strategy on each BackingMap in the WebSphere eXtreme Scale configuration.

"Configuring peer-to-peer replication with JMS" on page 267
The Java Message Service (JMS) based peer-to-peer replication mechanism is used in both the distributed and local WebSphere eXtreme Scale environment. JMS is a core-to-core replication process and allows data updates to flow among local ObjectGrids and distributed ObjectGrids. For example, with this mechanism you can move data updates from a distributed eXtreme Scale data grid to a local eXtreme Scale grid, or from a grid to another grid in a different system domain.

"Configuring evictors with XML files" on page 263
In addition to programmatically setting a time-to-live (TTL) evictor with the BackingMap interface, you can use an XML file to configure an evictor on each BackingMap instance.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

JMS event listener

The JMSObjectGridEventListener is designed to support client-side near cache invalidation and a peer-to-peer replication mechanism. It is a Java Message Service (JMS) implementation of the ObjectGridEventListener interface.

The client invalidation mechanism can be used in a distributed eXtreme Scale environment to ensure client near cache data to be synchronized with servers or other clients. Without this function, the client near cache could hold stale data. However, even with this JMS-based client invalidation mechanism, you have to take into consideration the timing window for updating a client near cache because of the delay for the run time in publishing updates.

The peer-to-peer replication mechanism can be used in both distributed and local eXtreme Scale environments. It is an ObjectGrid core-to-core replication process and allows data updates to flow among local ObjectGrids and distributed ObjectGrids. For example, with this mechanism you can move data updates from a distributed grid to a local ObjectGrid, or from any grid to another grid in a different system domain.

The JMSObjectGridEventListener requires the user to configure JMS and Java Naming and Directory Interface (JNDI) information in order to obtain required JMS resources. Additionally, replication-related properties must be set correctly. In a JEE environment, the JNDI should be available in both Web and Enterprise JavaBean (EJB) containers. In this case, the JNDI property is optional unless you want to obtained external JMS resources.

This event listener has properties you can configure with XML or programmatic approaches, which can be used for only client invalidation, only peer-to-peer replication, or both. Most properties are optional for customizing the behavior to achieve your required functionality.

.

For more information see the JMSObjectGridEventListener API.

Extending the JMSObjectGridEventListener plug-in

The JMSObjectGridEventListener plug-in allows peer ObjectGrid instances to receive updates when data in the grid has been changed or evicted. It also allows clients to be notified when entries are updated or evicted from an eXtreme Scale grid. This topic describes how to extend the JMSObjectGridEventListener plug-in to allow applications to be notified when a JMS message is received. This is most useful when using the CLIENT_SERVER_MODEL setting for client invalidation.

When running in the receiver role, the overridden JMSObjectGridEventListener.onMessage method is automatically called by the eXtreme Scale runtime when the JMSObjectGridEventListener instance receives JMS message updates from the grid. These messages wrap a collection of LogSequence. Objects. The LogSequence objects are passed to the onMessage method and the application uses the LogSequence to identify which cache entries have been inserted, deleted, updated or invalidated.

To use the onMessage extension point, applications perform the following steps.

1. Create a new class, extending the JMSObjectGridEventListener class, overriding the onMessage method.
2. Configure the extended JMSObjectGridEventListener the same way as the ObjectGridEventListener for ObjectGrid.

The extended JMSObjectGridEventListener class is a child class of the JMSObjectGridEventListener class and can only override two methods: the initialize (optional) and onMessage methods. If a child class of the JMSObjectGridEventListener class needs to use any ObjectGrid artifacts such as ObjectGrid or Session in the onMessage method, it can get these artifacts in the initialize method and cache them as instance variables. Also, in the onMessage method, cached ObjectGrid artifacts can be used to process a passed collection of LogSequences.

Note: The overridden initialize method has to invoke super.initialize method in order to initialize parent JMSObjectGridEventListener appropriately.

The following is a sample for an extended JMSObjectGridEventListener class.

```
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 + "]";
 }
}
```

Configuration

The extended JMSObjectGridEventListener class must be configured the same way for both client invalidation and peer-to-peer replication mechanism. The following is the XML configuration example.

```
<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>
```

Note: The className of ObjectGridEventListener bean is configured with the extended JMSObjectGridEventListener class with the same properties as the generic JMSObjectGridEventListener.

Related tasks:

"Configuring data grids" on page 261
Use an ObjectGrid descriptor XML file to configure data grids, backing maps, plug-ins, and so on. To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API. For a distributed topology, you need an ObjectGrid descriptor XML file and a deployment policy XML file.

"Configuring local deployments" on page 261
A local in-memory eXtreme Scale configuration can be created by using an ObjectGrid descriptor XML file or APIs.

"Configuring a locking strategy in the ObjectGrid descriptor XML file" on page 264
You can define an optimistic, a pessimistic, or no locking strategy on each BackingMap in the WebSphere eXtreme Scale configuration.

"Configuring peer-to-peer replication with JMS" on page 267
The Java Message Service (JMS) based peer-to-peer replication mechanism is used in both the distributed and local WebSphere eXtreme Scale environment. JMS is a core-to-core replication process and allows data updates to flow among local ObjectGrids and distributed ObjectGrids. For example, with this mechanism you can move data updates from a distributed eXtreme Scale data grid to a local eXtreme Scale grid, or from a grid to another grid in a different system domain.

"Configuring evictors with XML files" on page 263
In addition to programmatically setting a time-to-live (TTL) evictor with the BackingMap interface, you can use an XML file to configure an evictor on each BackingMap instance.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Configuring dynamic maps

You can dynamically create maps that are based on a set of map templates. You can create your own map templates.

About this task

WebSphere eXtreme Scale uses Java regular expression pattern matching. For more information about the regular expression engine in Java, see the API documentation for the java.util.regex package and classes.

Limitations:

- The QuerySchema element does not support the template for mapName.
- You cannot use entities with dynamic maps.
- An entity BackingMap is implicitly defined, mapped to the entity through the class name.

Considerations:

- Many plug-ins have no way of determining the map with which each plug-in is associated.
- Other plug-ins differentiate themselves by using a map name or BackingMap name as an argument.

Procedure

Define a template map in your ObjectGrid XML file. To define a template map in the ObjectGrid XML file, set the **`template`** attribute on the backingMap element to `true`. As a result, the name of the backingMap is interpreted as a regular expression.

Attention: When you are defining template maps, verify that the map names are unique enough so that the application can match to only one of the template maps

with the Session.getMap(String mapName) method. If the getMap() method matches more than one template map pattern, an `IllegalArgumentException` exception results. With dynamic maps, every name that matches the regular expression for a template map results in map creation. Be sure to note the number of maps that your application creates, particularly if your regular expression is generic.
An example of an ObjectGrid XML file with a template map defined follows. This XML file defines one template map and one non-template map. The name of the template map is the following regular expression: `templateMap.*`. When the Session.getMap(String) method is called with a map name that matches this regular expression, the application creates a map.

```
<?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" />
   <backingMap name="templateMap.*" template="true"
    pluginCollectionRef="templatePlugins" lockStrategy="PESSIMISTIC" />
  </objectGrid>
 </objectGrids>

 <backingMapPluginCollections>
  <backingMapPluginCollection id="templatePlugins">
   <bean id="Evictor"
    className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor" />
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Note: A template is not an actual BackingMap. That is, the "accounting" ObjectGrid does not contain an actual “templateMap.*” map. The template is only used as a basis for dynamic map creation. However, you must include the dynamic map in the mapRef element of the deployment policy XML file that is named exactly as in the ObjectGrid XML. This element identifies which mapSet in which the dynamic maps are defined.

Example

objectGrid.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="session">
    <backingMap name="objectgrid.session.metadata.dynamicmap.*" template="true"
        lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME">
    <backingMap name="objectgrid.session.attribute.dynamicmap.*"
        template="true" lockStrategy="OPTIMISTIC"/>
    <backingMap name="datagrid.session.global.ids.dynamicmap.*"
        lockStrategy="PESSIMISTIC"/>
 </objectGrid>
</objectGrids>
</objectGridConfig>
```

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="session">
   <mapSet name="mapSet2" numberOfPartitions="5" minSyncReplicas="0"
     maxSyncReplicas="0" maxAsyncReplicas="1" developmentMode="false"
     placementStrategy="PER_CONTAINER">
    <map ref="logical.name"/>
    <map ref="objectgrid.session.metadata.dynamicmap.*"/>
    <map ref="objectgrid.session.attribute.dynamicmap.*"/>
    <map ref="datagrid.session.global.ids"/>
   </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

What to do next

Create a dynamic map with your defined templates:

- **With Java APIs:** See Creating dynamic maps with Java APIs for an example of calling the Session.getMap(String) method to define your dynamic map.

Note: The ObjectGridPermission.DYNAMIC_MAP is required for dynamic map creation when eXtreme Scale security is enabled. This permission is checked when the Session.getMap(String) method is called. For more information, see "Authorizing application clients" on page 589.

Configuring deployment policies

Use the deployment policy descriptor XML file and the objectgrid descriptor XML file to manage a distributed topology. The deployment policy is encoded as an XML file that is provided to the container server. The deployment policy provides information about maps, map sets, partitions, replicas, and so on. It also controls shard placement behaviors.

Related concepts:

"Embedded server API" on page 479
WebSphere eXtreme Scale includes application programming interfaces (APIs) and system programming interfaces for embedding eXtreme Scale servers and clients within your existing Java applications.

Interacting with an ObjectGrid using the ObjectGridManager interface
The ObjectGridManagerFactory class and the ObjectGridManager interface provide a mechanism to create, access, and add data to ObjectGrid instances. The ObjectGridManagerFactory class is a static helper class to access the ObjectGridManager interface, a singleton. The ObjectGridManager interface includes several convenience methods to create instances of an ObjectGrid object. The ObjectGridManager interface also facilitates creation and caching of ObjectGrid instances that can be accessed by several users.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both

stand-alone servers and servers that are hosted in WebSphere Application Server.

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Related information:

ObjectGridManager interface

ClientClusterContext interface

DeploymentPolicy interface

Configuring distributed deployments

Use the deployment policy descriptor XML file and the ObjectGrid descriptor XML file to manage your topology.

The deployment policy is encoded as an XML file that is provided to the eXtreme Scale container server. The XML file specifies the following information:

- The maps that belong to each map set
- The number of partitions
- The number of synchronous and asynchronous replicas

The deployment policy also controls the following placement behaviors.

- The minimum number of active container servers before placement occurs
- Automatic replacement of lost shards
- Placement of each shard from a single partition onto a different machine

Endpoint information is not pre-configured in the dynamic environment. There are no server names or physical topology information found in the deployment policy. All shards in a data grid are automatically placed into container servers by the catalog service. The catalog service uses the constraints that are defined by the deployment policy to automatically manage shard placement. This automatic shard placement leads to easy configuration for large data grids. You can also add servers to your environment as needed.

Restriction: In a WebSphere Application Server environment, a core group size of more than 50 members is not supported.

A deployment policy XML file is passed to a container server during startup. A deployment policy must be used along with an ObjectGrid XML file. The deployment policy is not required to start a container server, but is recommended. The deployment policy must be compatible with the ObjectGrid XML file that is used with it. For each objectgridDeployment element in the deployment policy, you must include a corresponding objectGrid element in your ObjectGrid XML file. The maps in the objectgridDeployment must be consistent with the backingMap elements found in the ObjectGrid XML. Every backingMap must be referenced within only one mapSet element.

In the following example, the `companyGridDpReplication.xml` file is intended to be paired with the corresponding `companyGrid.xml` file.

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>
```

The `companyGridDpReplication.xml` file has one mapSet element that is divided into 11 partitions. Each partition must have exactly one synchronous replica. The number of synchronous replicas is specified by the minSyncReplicas and maxSyncReplicas attributes. Because the minSyncReplicas attribute is set to 1, each partition in the mapSet element must have at least one synchronous replica available to process write transactions. Because the maxSyncReplicas attribute is set to 1, each partition cannot exceed one synchronous replica. The partitions in this mapSet element have no asynchronous replicas.

The numInitialContainers attribute instructs the catalog service to defer placement until four container servers are available to support this ObjectGrid instance. The numInitialContainers attribute is ignored after the specified number of container servers has been reached.

7.1.1+ You can also use the **`placementDeferralInterval`** property and **`xscmd -c suspendBalancing`** command to delay the placement of shards on the container servers.

Although the `companyGridDpReplication.xml` file is a basic example, a deployment policy can offer you full control over your environment.

Distributed topology

Distributed coherent caches offer increased performance, availability, and scalability, which you can configure.

WebSphere eXtreme Scale automatically balances servers. You can include additional servers without restarting WebSphere eXtreme Scale. Adding additional servers without having to restart eXtreme Scale allows you to have simple deployments and also large, terabyte-sized deployments in which thousands of servers are needed.

This deployment topology is flexible. Using the catalog service, you can add and remove servers to better use resources without removing the entire cache. You can use the **`startOgServer`** and **`stopOgServer`** commands to start and stop container servers. Both of these commands require you to specify the **`-catalogServiceEndPoints`** option. All distributed topology clients communicate to the catalog service through the Internet Interoperability Object Protocol (IIOP). All clients use the ObjectGrid interface to communicate with servers.

The dynamic configuration capability of WebSphere eXtreme Scale makes it easy to add resources to the system. Containers host the data and the catalog service allows clients to communicate with the grid of container servers. The catalog service forwards requests, allocates space in host container servers, and manages the health and availability of the overall system. Clients connect to a catalog service, retrieve a description of the container server topology, and then communicate directly to each server as needed. When the server topology changes due to the addition of new servers, or due to the failure of others, the catalog service automatically routes client requests to the appropriate server that hosts the data.

A catalog service typically exists in its own grid of Java virtual machines. A single catalog server can manage multiple servers. You can start a container server in a JVM by itself or load the container server into an arbitrary JVM with other container servers for different servers. A client can exist in any JVM and communicate with one or more servers. A client can also exist in the same JVM as a container server.

You can also create a deployment policy programmatically when you are embedding a container server in an existing Java process or application. For more information, see the DeploymentPolicy API documentation.

Related tasks:

"Starting container servers" on page 463
You can start container servers from the command line using a deployment topology or using a `server.properties` file.

"Configuring WebSphere Application Server applications to automatically start container servers" on page 327
Container servers in a WebSphere Application Server environment start automatically when a module starts that has the eXtreme Scale XML files included.

"Controlling placement" on page 486
You can use several different options to control when shards are placed on various container servers in the configuration. During startup, you might choose to delay the placement of shards. When you are running all of your container servers, you might need to suspend, resume, or change placement while you maintain servers.

Controlling shard placement with zones

Use your deployment policy to define zones. Zones give you control over shard placement in WebSphere eXtreme Scale. Zones are a logical, user-defined concept used to represent logical groupings of physical servers.

Related concepts:

Zones
Zones give you control over shard placement. Zones are user-defined logical groupings of physical servers. The following are examples of different types of zones: different blade servers, chassis of blade servers, floors of a building, buildings, or different geographical locations in a multiple data center environment. Another use case is in a virtualized environment where many server instances, each with a unique IP address, run on the same physical server.

"Zone-preferred routing" on page 283
With zone-preferred routing, you can define how WebSphere eXtreme Scale directs transactions to zones.

Related reference:

"Example: Zone definitions in the deployment policy descriptor XML file" on page 288
You can specify zones and zone rules with the deployment policy descriptor XML file.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Zones for replica placement

With zones, you can place replicas across data centers. A zone can be defined as different floors of a building, different buildings, or even different cities or other distinctions as configured with zone rules. With this capability, data grids of thousands of partitions can be managed with optional placement rules.

Zone rules

An eXtreme Scale partition has one primary shard and zero or more replica shards. For this example, consider the following naming convention for these shards. `P` is the primary shard, `S` is a synchronous replica and `A` is an asynchronous replica. A zone rule has three components:

- A rule name
- A list of zones
- An inclusive or exclusive flag

For more information about defining a zone name for a container server, see "Defining zones for container servers" on page 287. A zone rule specifies the possible set of zones in which a shard can be placed. The inclusive flag indicates that after a shard is placed in a zone from the list, then all other shards are also placed in that zone. An exclusive setting indicates that each shard for a partition is placed in a different zone in the zone list. For example, using an exclusive setting means that if there are three shards (primary, and two synchronous replicas), then the zone list must have three zones.

Each shard can be associated with one zone rule. A zone rule can be shared between two shards. When a rule is shared then the inclusive or exclusive flag extends across shards of all types sharing a single rule.

Examples

A set of examples showing various scenarios and the deployment configuration to implement the scenarios follows.

Striping primaries and replicas across zones

You have three blade chassis, and want primaries that are distributed across all three, with a single synchronous replica placed in a different chassis than the primary. Define each chassis as a zone with chassis names `ALPHA`, `BETA`, and `GAMMA`. An example deployment XML follows:

```
<?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>
```

This deployment XML contains a grid called library with a single Map called book. It uses four partitions with a single synchronous replica. The zone metadata clause shows the definition of a single zone rule and the association of zone rules with shards. The primary and synchronous shards are both associated with the zone rule "stripeZone". The zone rule has all three zones in it and it uses exclusive placement. This rule means that if the primary for partition 0 is placed in `ALPHA` then the replica for partition 0 is placed in either `BETA` or `GAMMA`. Similarly, primaries for other partitions are placed in other zones and the replicas are placed.

Asynchronous replica in a different zone than primary and synchronous replica

In this example, two buildings exist with a high latency connection between them. You want no data loss high availability for all scenarios. However, the performance impact of synchronous replication between buildings leads you to a trade-off. You want a primary with synchronous replica in one building and an asynchronous replica in the other building. Normally, the failures are JVM crashes or computer failures rather than large-scale issues. With this topology, you can survive normal failures with no data loss. The loss of a building is rare enough that some data loss is acceptable in that case. You can make two zones, one for each building. The deployment XML file follows:

```
<?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>
```

The primary and synchronous replica share a primaySync zone rule with an exclusive flag setting of false. So, after the primary or sync gets placed in a zone, then the other is also placed in the same zone. The asynchronous replica uses a second zone rule with the same zones as the primarySync zone rule but it uses the **`exclusivePlacement`** attribute set to true. This attribute indicates that means a shard cannot be placed in a zone with another shard from the same partition. As a result, the asynchronous replica does not get placed in the same zone as the primary or synchronous replicas.

Placing all primaries in one zone and all replicas in another zone

Here, all primaries are in one specific zone and all replicas in a different zone, a primary and a single asynchronous replica. All replicas are in zone A and primaries in B.

```
<?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>
```

Here, you can see two rules, one for the primaries (P) and another for the replica (A).

Zones over wide area networks (WAN)

You might want to deploy a single data grid over multiple buildings or data centers with slower network interconnections. Slower network connections lead to lower bandwidth and higher latency connections. The possibility of network partitions also increases in this mode due to network congestion and other factors. eXtreme Scale approaches this harsh environment by limiting heartbeating between zones.

Java virtual machines grouped into core groups do heartbeat each other. When the catalog service organizes Java virtual machines into core groups, those groups do not span zones. A leader within that group pushes membership information to the catalog service. The catalog service verifies any reported failures before taking action. It does this by attempting to connect to the suspect Java virtual machines. If the catalog service sees a false failure detection, then it takes no action as the core group partition heals in a short time.

The catalog service also heartbeats core group leaders periodically at a slow rate to handle the case of core group isolation.

Zone-preferred routing

With zone-preferred routing, you can define how WebSphere eXtreme Scale directs transactions to zones.

You have control over where the shards of a data grid are placed. See "Zones for replica placement" on page 281 to get more information about some basic scenarios and how to configure your deployment policy accordingly.

Zone-preferred routing givesWebSphere eXtreme Scale clients the capability to specify a preference for a particular zone or set of zones. As a result, client transactions are routed to preferred zones before attempting to route to any other zone.

Requirements for zone-preferred routing

Before attempting zone-preferred routing, ensure that the application is able to satisfy the requirements of your scenario.

Per-container partition placement is necessary to use zone-preferred routing. This placement strategy is a good fit for applications that are storing session data in the ObjectGrid. The default partition placement strategy for WebSphere eXtreme Scale is `fixed-partition`. Keys are hashed at transaction commit time to determine which partition houses the key-value pair of the map when using fixed-partition placement.

Per-container placement assigns your data to a random partition when the transaction commits time through the SessionHandle object. You must be able to reconstruct the SessionHandle object to retrieve your data from the data grid.

You can use zones to have more control over where primary shards and replica shards are placed in your domain. Using multiple zones in your deployment is advantageous when your data is in multiple physical locations. Geographically separating primaries and replicas is a way to ensure that catastrophic loss of one data center does not affect the availability of the data.

When data is spread across multiple zones, it is likely that clients are also spread across the topology. Routing clients to their local zone or data center has the obvious performance benefit of reduced network latency. Route clients to local zones or data centers when possible.

Configuring your topology for zone-preferred routing

Consider the following scenario. You have two data centers: Chicago and London. To minimize response time of clients, you want clients to read and write data to their local data center.

Primary shards must be placed in each data center so that transactions can be written locally from each location. Clients must be aware of zones to route to the local zone.

Per-container placement locates new primary shards on each container that is started. Replicas are placed according to zone and placement rules that are specified by the deployment policy. By default, a replica is placed in a different zone than its primary shard. Consider the following deployment policy for this scenario.

```
<?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>
```

Each container that starts with the deployment policy receives three new primaries. Each primary has one asynchronous replica. Start each container with the appropriate zone name. Use the **`-zone`** parameter if you are starting your containers with the **`startOgServer`** script.

For a Chicago container server:

- 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
  ```

If your containers are running in WebSphere Application Server, you must create a node group and name it with the prefix `ReplicationZone`. Servers that are running on the nodes in these node groups are placed in the appropriate zone. For example, servers running on a Chicago node might be in a node group named `ReplicationZoneChicago`.

See "Zones for replica placement" on page 281 for more information.

Primary shards for the Chicago zone have replicas in the London zone. Primary shards for the London zone have replicas in the Chicago zone.

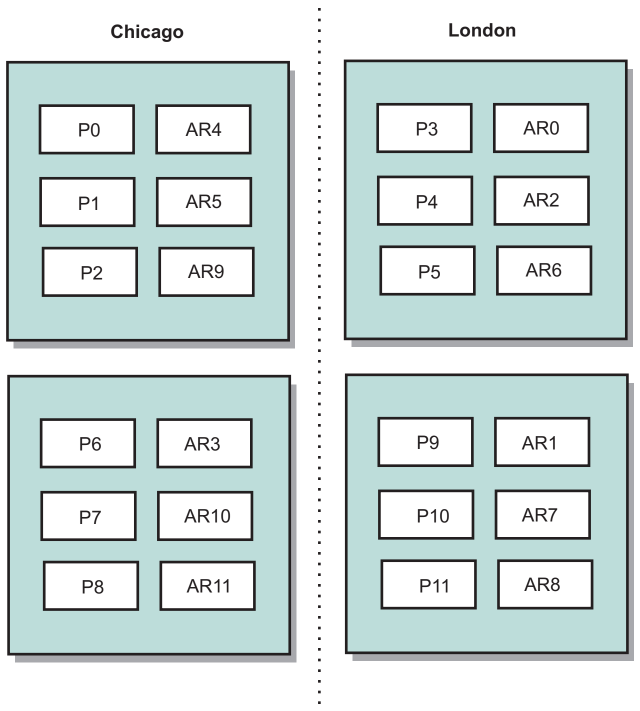


Figure 31. Primaries and replicas in zones

Set the preferred zones for the clients. Provide a client properties file to your client Java virtual machine (JVM). Create a file named `objectGridClient.properties` and ensure that this file is in your classpath.

Include the **`preferZones`** property in the file. Set the property value to the appropriate zone. Clients in Chicago must have the following value in the `objectGridClient.properties` file:

```
preferZones=Chicago
```

The property file for London clients must contain the following value:

```
preferZones=London
```

This property instructs each client to route transactions to its local zone if possible. The topology asynchronously replicates data that is inserted into a primary shard in the local zone into the foreign zone.

Using the SessionHandle interface to route to the local zone

The per-container placement strategy does not use a hash-based algorithm to determine the location of your key-value pairs in the data grid. You must use SessionHandle objects to ensure that transactions are routed to the correct location when you are using this placement strategy. When a transaction is committed, a SessionHandle object is bound to the session if one has not already been set. The SessionHandle object can also be bound to the Session by calling the Session.getSessionHandle method before committing the transaction. The following code snippet shows a SessionHandle being bound before committing the transaction.

```
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();
```

Assume that the prior code was running on a client in your Chicago data center. The **preferZones** attribute is set to `Chicago` for this client. As a result, your deployment would route transactions to one of the primary partitions in the Chicago zone: partition 0, 1, 2, 6, 7, or 8.

The SessionHandle object provides a path back to the partition that is storing this committed data. The SessionHandle object must be reused or reconstructed and set on the Session to get back to the partition containing the committed data.

```
ogSession.setSessionHandle(sessionHandle);
ogSession.begin();

// value returned will be "mercury"
String value = map.get("planet1");
ogSession.commit();
```

The transaction in this code reuses the SessionHandle object that was created during the insert transaction. The get transaction then routes to the partition that holds the inserted data. Without the SessionHandle object, the transaction cannot retrieve the inserted data.

How container and zone failures affect zone-based routing

Generally, a client with the **preferZones** property set routes all transactions to the specified zone or zones. However, the loss of a container results in the promotion of a replica shard to a primary shard. A client that was previously routing to partitions in the local zone must retrieve previously inserted data from the remote zone.

Consider the following scenario. A container in the Chicago zone is lost. It previously contained primaries for partitions 0, 1, and 2. The new primary shards for these partitions are then placed in the London zone because the London zone hosted the replicas for these partitions.

Any Chicago client that is using a SessionHandle object that points to one of the failed-over partitions now reroutes to London. Chicago clients that are using new SessionHandle objects route to Chicago-based primaries.

Similarly, if the entire Chicago zone is lost, all replicas in the London zone are promoted to primaries. In this scenario, all Chicago clients route their transactions to London.

Related tasks:

"Controlling shard placement with zones" on page 280
Use your deployment policy to define zones. Zones give you control over shard placement in WebSphere eXtreme Scale. Zones are a logical, user-defined concept used to represent logical groupings of physical servers.

"Defining zones for container servers"
Zones are collections of container servers. A container server can belong only one zone. A container server is assigned to a zone when it starts.

"Viewing zone information with the **xscmd** utility" on page 292
You can use the **xscmd** utility to view information about your current zone deployment, including shard placement data.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related reference:

"Example: Zone definitions in the deployment policy descriptor XML file" on page 288
You can specify zones and zone rules with the deployment policy descriptor XML file.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Defining zones for container servers

Zones are collections of container servers. A container server can belong only one zone. A container server is assigned to a zone when it starts.

About this task

You must plan your zones before you start your container servers because container servers define their zone membership at startup. If you want to change the zone membership of a container server, you must restart the server with the new zone information.

Procedure

How you define zones depends on if you are using stand-alone container servers or container servers that are running within WebSphere Application Server:

- Define zones for stand-alone container servers.
  1. Use the **-zone** parameter of the **startOgServer** script to specify the zone for all the containers in the started server. For more information about starting servers, see "Starting and stopping stand-alone servers" on page 459.

2. You can also zone names when you are starting container servers programmatically with the embedded server API. For more information, see "Using the embedded server API to start and stop servers" on page 476.

- Define zones for container servers that are running within WebSphere Application Server.

  You can use node groups to place container servers in specific zones. Use the following syntax to name your node group to assign it a zone: `ReplicationZone<identifier>`. When you define zones in the deployment policy, you must name the zones exactly as you named the node groups. The node group name and the zone name in the deployment policy descriptor XML file must be identical

  Important: WebSphere Application Server does not prohibit nodes from being in multiple node groups. Because container servers can be only one zone, ensure that your nodes are in exactly one ReplicationZone node group.

  For example, divide four nodes into two zones, A and B.

  1. Configure four nodes: node1, node2, node3, and node4, each node having two servers.
  2. Create a node group named ReplicationZoneA and a node group named ReplicationZoneB.
  3. Add node1 and node2 to ReplicationZoneA and add node3 and node4 to ReplicationZoneB.
  4. Define ReplicationZoneA and ReplicationZoneB in your deployment policy descriptor XML file. See "Example: Zones in a WebSphere Application Server environment" on page 291 for an example.
  5. When the servers on node1 and node2 are started, they join ReplicationZoneA, or zone A in the WebSphere eXtreme Scale configuration. The servers on node3 and node4 join ReplicationZoneB, as zone B in the WebSphere eXtreme Scale configuration.

Related concepts:

Zones
Zones give you control over shard placement. Zones are user-defined logical groupings of physical servers. The following are examples of different types of zones: different blade servers, chassis of blade servers, floors of a building, buildings, or different geographical locations in a multiple data center environment. Another use case is in a virtualized environment where many server instances, each with a unique IP address, run on the same physical server.

"Zone-preferred routing" on page 283
With zone-preferred routing, you can define how WebSphere eXtreme Scale directs transactions to zones.

Related reference:

"Example: Zone definitions in the deployment policy descriptor XML file"
You can specify zones and zone rules with the deployment policy descriptor XML file.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Example: Zone definitions in the deployment policy descriptor XML file

You can specify zones and zone rules with the deployment policy descriptor XML file.

Example: Primary and replica shards in different zones

This example places primary shards in one zone, and replica shards in a different zone, with a single asynchronous replica. All primary shards start in the DC1 zone. Replica shards start in zone DC2.

```
<?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>
```

One asynchronous replica is defined in the ms1 mapSet element. Therefore, two shards exist for each partition: a primary and one asynchronous replica. In the zoneMetadata element, a shardMapping element is defined for each shard: P for the primary, and DC1 for the asynchronous replica. The primaryRule attribute defines the zone set for the primary shards, which is just zone DC1, and this rule is to be used for primary shard placement. Asynchronous replicas are placed in the DC2 zone.

However, if the DC2 zone is lost, the replica shards become unavailable. The loss or failure of a container server in the DC1 zone can result in data loss, even though a replica has been specified.

To address this possibility, you can either add a zone or add a replica, as described in the following sections.

Example: Add a zone, striping shards

The following code configures a new zone:

```
<?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>
```

Three total zones have been defined in this code: A, B, and C. Instead of separate primary and replica zone rules, a shared zone rule named stripeRule is defined. This rule includes all of the zones, with the exclusivePlacement attribute set to `true`. The eXtreme Scale placement policy ensures that primary and replica shards are in separate zones. This striping of placement causes primary and replica shards to spread across both zones to conform to this policy. Adding a third zone C ensures that losing any one zone does not result in data loss, and still leaves primary and replica shards for each partition. A zone failure results in the loss of either the primary shard, the replica shard, or neither. Any lost shard is replaced from the surviving shard in a surviving zone, placing it in the other surviving zone.

Example: Add a replica and define multiple data centers

The classic two data-center scenario has high speed, low latency networks in each data center, but high latency between the data centers. Synchronous replicas are used in each data center where the low latency minimizes the impact of replication on response times. Asynchronous replication is used between data centers, so the high latency network has no impact on response time.

```
<?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>
```

The primary and synchronous replica share the primarySync rule with an exclusivePlacement attribute setting of `false`. The exclusivePlacement attribute set to false creates a configuration with the primary and synchronous replica shards of each partition placed in the same zone. The asynchronous replica shard uses a second zone rule with mostly the same zones as the primarySync zone rule. However the asynchronous replica uses the exclusivePlacement attribute set to `true`. The exclusivePlacement attribute, when set to true, means that a shard cannot be placed in a zone with another shard from the same partition. As a result, the asynchronous replica shard does not get placed in the same zone as the primary or synchronous replica shard. There are three shards per partition in this mapSet: a primary, and both a synchronous and asynchronous replica, so there are three shardMapping elements, one for each shard.

If a zone is lost, any asynchronous replicas are lost, and not regenerated, because they have no separate zone. If the primary and replica shards are lost, then the surviving asynchronous replica is promoted to primary, and a new synchronous replica is created in the zone. The primaries and replicas are striped across each zone.

With exclusive placement, each shard has its own zone: You must have enough zones for all the shards you want to place in their own zones. If a rule has one zone, only one shard can be placed in the zone. With two zones, you can have up to two shards in the zone.

Example: Zones in a WebSphere Application Server environment

The following code configures a new zone:

```
<?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>
```

For this example, three node groups are defined in the WebSphere Application Server environment: ReplicationZoneA, ReplicationZoneB, and ReplicationZoneC. The node group name and the zone name in the deployment policy descriptor XML file must be identical, and must contain the text `ReplicationZone<identifier>`. This file defines a similar configuration to the striping shards example, but shows the required naming for a WebSphere Application Server configuration.

Related concepts:

Zones
Zones give you control over shard placement. Zones are user-defined logical groupings of physical servers. The following are examples of different types of zones: different blade servers, chassis of blade servers, floors of a building, buildings, or different geographical locations in a multiple data center environment. Another use case is in a virtualized environment where many server instances, each with a unique IP address, run on the same physical server.

"Zone-preferred routing" on page 283
With zone-preferred routing, you can define how WebSphere eXtreme Scale directs transactions to zones.

Related tasks:

"Controlling shard placement with zones" on page 280
Use your deployment policy to define zones. Zones give you control over shard placement in WebSphere eXtreme Scale. Zones are a logical, user-defined concept used to represent logical groupings of physical servers.

"Defining zones for container servers" on page 287
Zones are collections of container servers. A container server can belong only one zone. A container server is assigned to a zone when it starts.

"Viewing zone information with the **xscmd** utility"
You can use the **xscmd** utility to view information about your current zone deployment, including shard placement data.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Viewing zone information with the xscmd utility

You can use the **xscmd** utility to view information about your current zone deployment, including shard placement data.

Before you begin

- Deploy a distributed data grid with multiple data centers. See "Zone-preferred routing" on page 283 for more information.

About this task

You can determine information about your configuration related to zone settings by using the **xscmd** utility that ships with the product.

Procedure

Use the **xscmd** utility to determine information about the shards of data. Run the following command:

```
xscmd -c showPlacement -z zone_name
```

Example

You can also run a simpler scenario by using the getting started sample: *wxs_install_root*/ObjectGrid/gettingstarted. See "Tutorial: Getting started with WebSphere eXtreme Scale" on page 1 for more information.

1. Start a catalog server:
   ```
   runcat.bat
   ```
2. Determine your required number of replicas, zone rules, containers, and other settings such as with the following command:
   ```
   startOgServer.bat serverA0 -objectgridFile xml\objectgrid.xml
   -deploymentPolicyFile xml\deployment.xml -zone zoneA
   ```
3. You can stop container processes to simulate failure in the data grid:
   ```
   stopOgServer.bat serverA0,serverA1,serverB0 -catalogServiceEndPoints localhost:2809
   ```
   .

   If the server that contains the last shard of a partition is stopped, eXtreme Scale allocates a new primary shard. You can check for data loss:
   - The **runclient** script inserts and reads item in your data grid.

- The **xscmd -c showMapSizes** command shows the number of items in the data grid.

4. Show active container servers with the following command:

```
xscmd -c showPlacement -z zone_name
```

Related concepts:

Zones
Zones give you control over shard placement. Zones are user-defined logical groupings of physical servers. The following are examples of different types of zones: different blade servers, chassis of blade servers, floors of a building, buildings, or different geographical locations in a multiple data center environment. Another use case is in a virtualized environment where many server instances, each with a unique IP address, run on the same physical server.

"Zone-preferred routing" on page 283
With zone-preferred routing, you can define how WebSphere eXtreme Scale directs transactions to zones.

Related reference:

"Example: Zone definitions in the deployment policy descriptor XML file" on page 288
You can specify zones and zone rules with the deployment policy descriptor XML file.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Configuring catalog and container servers

WebSphere eXtreme Scale has two types of servers: catalog servers and container servers. Catalog servers control the placement of shards and discover and monitor the container servers. Multiple catalog servers can join a catalog service domain to provide high availability to the environment. A container server is a Java virtual machine (JVM) that stores the application data for the data grid.

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Configuring catalog servers and catalog service domains

The catalog service hosts logic that is typically idle during steady states. As a result, the catalog service minimally influences scalability. The service is built to service hundreds of container servers that become available simultaneously. For high availability, configure the catalog service into a catalog service domain.

Before you begin

After a catalog service domain is started, the members of the data grid bind together. Carefully plan your catalog service domain topology, because you cannot modify your catalog service domain configuration at run time. Spread out the data grid as diversely as possible to prevent errors.

The best practice to avoid a single point of failure for your catalog service domain is to start a minimum of three catalog servers on three different nodes.

If you are using only two nodes, configure two catalog servers on each of the two nodes for a total of four catalog server processes. Creating this configuration ensures that when only one of the nodes is started, the required two catalog servers are running. You must start at least two catalog servers at the same time. When catalog servers start, they look for other catalog servers in the configuration, and do not start successfully until at least one other catalog sever is found.

Procedure

- Configure stand-alone catalog servers and catalog service domains.

  You configure stand-alone catalog server and catalog service domains with parameters and property files that you pass to the start server command or to the embedded server API.
  - "Example: Configuring catalog service domains" on page 295
  - "Starting and stopping stand-alone servers" on page 459
  - Catalog server properties
- Configure catalog servers and catalog service domains in WebSphere Application Server

  Configure catalog servers that run in WebSphere Application Server with the WebSphere Application Server administrative console, administrative tasks, and the server properties file. The server life cycle is controlled by the process life cycle within WebSphere Application Server. When processes start or stop in WebSphere Application Server, the catalog servers that are running on these processes also start or stop.
  - "Creating catalog service domains in WebSphere Application Server" on page 299
  - "Configuring the catalog service in WebSphere Application Server" on page 298

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Example: Configuring catalog service domains

When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Example: Starting four catalog servers on two nodes in a stand-alone environment

The following script starts catalog servers cs0 and cs1 on the host1 node, and starts catalog servers cs2 and cs3 on the host2 node.

```
./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
```

Remember: You must use the **-listenerPort** option because the catalog servers that are running on a node each require a unique port number.

Example: Starting multiple catalog servers in a WebSphere Application Server environment

Catalog servers start automatically in a WebSphere Application Server environment. You can define multiple catalog servers to start by creating a catalog service domain. After you specify multiple endpoints in the catalog service domain, restart the included application servers so that the catalog servers start in parallel.

- **WebSphere Application Server Network Deployment**: You can choose multiple existing application servers from the cell to be members of your catalog service domain.
- **Base WebSphere Application Server**: You can start the catalog service on multiple stand-alone nodes. By defining multiple profiles on the same installation image with the profile management tool, you can create a set of stand-alone nodes that each have unique ports assigned. In each application server, define the catalog service domain. You can specify any other application servers by adding remote servers to the configuration. After you create this configuration on all of the stand-alone servers, you can start the set of base application servers in parallel by running the `startServer` script or by using a Windows service to start the servers.

Related tasks:

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

"Configuring the catalog service in WebSphere Application Server" on page 298
Catalog service processes can run in WebSphere Application Server. The server life cycle in WebSphere Application Server determines when the catalog service starts and stops.

"Creating catalog service domains in WebSphere Application Server" on page 299
Catalog service domains define a group of catalog servers that manage the placement of shards and monitor the health of container servers in your data grid.

"Starting and stopping servers in a WebSphere Application Server environment" on page 475
Catalog and container servers can automatically start in a WebSphere Application Server or WebSphere Application Server Network Deployment environment.

"Troubleshooting administration" on page 631
Use the following information to troubleshoot administration, including starting and stopping servers, using the `xscmd` utility, and so on.

"Administering with the `xscmd` utility" on page 482
With the `xscmd` utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

"Starting and stopping stand-alone servers" on page 459
You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

"Starting a stand-alone catalog service" on page 461
You must start the catalog service manually when you are using a distributed WebSphere eXtreme Scale environment that is not running in WebSphere Application Server.

Related reference:

"Catalog service domain administrative tasks" on page 301
You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Configuring WebSphere eXtreme Scale with WebSphere Application Server

You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

About this task

Attention: Do not collocate your container servers with catalog servers in a production environment. Include the catalog service in multiple node agent processes or in an application server that is not hosting an eXtreme Scale application.

Related concepts:

"Interoperability with other products" on page 49
You can integrate WebSphere eXtreme Scale with other products, such as WebSphere Application Server and WebSphere Application Server Community Edition.

"Monitoring with vendor tools" on page 550
WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli Monitoring and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

"Tuning garbage collection with WebSphere Real Time" on page 577
Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary. WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

High availability catalog service
A catalog service domain is the data grid of catalog servers you are using, which retain topology information for all of the container servers in your eXtreme Scale environment. The catalog service controls balancing and routing for all clients.

Chapter 7, "Administering," on page 459

Related reference:

"Catalog service domain administrative tasks" on page 301
You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Configuring the catalog service in WebSphere Application Server:

Catalog service processes can run in WebSphere Application Server. The server life cycle in WebSphere Application Server determines when the catalog service starts and stops.

Procedure

1. Choose one or more WebSphere Application Server processes to augment with the WebSphere eXtreme Scale profile. See "Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203 for more information. If you want the catalog service to start automatically in WebSphere Application Server Network Deployment on the deployment manager, install WebSphere eXtreme Scale on the deployment manager node and augment the deployment manager profile.
2. Configure server properties files for the WebSphere Application Server processes and add to the class path for the node. See Server properties file for more information.
3. Configure a catalog service domain. The catalog service domain is a group of catalog servers within your environment. See "Creating catalog service domains in WebSphere Application Server" on page 299 for more information.
4. Start the WebSphere Application Server processes that are hosting the catalog servers. See "Starting and stopping servers in a WebSphere Application Server environment" on page 475 for more information.

Related concepts:

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

High availability catalog service
A catalog service domain is the data grid of catalog servers you are using, which retain topology information for all of the container servers in your eXtreme Scale environment. The catalog service controls balancing and routing for all clients.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

Related reference:

"Catalog service domain administrative tasks" on page 301
You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Creating catalog service domains in WebSphere Application Server:

Catalog service domains define a group of catalog servers that manage the placement of shards and monitor the health of container servers in your data grid.

Before you begin

- Install WebSphere eXtreme Scale on WebSphere Application Server. See 7.1.1 "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server" on page 186 for more information.

About this task

By creating a catalog service domain, you are defining a highly available collection of catalog servers.

These catalog servers can run in WebSphere Application Server within a single cell and core group. The catalog service domain can also define a remote group of servers that run in different Java SE processes or other WebSphere Application Server cells.

For catalog servers that run on existing application servers within the cell: When you define a catalog service domain that places catalog servers on the application servers within the cell, the core group mechanisms of WebSphere Application Server are used. The catalog service automatically starts on the application servers in the cell. As a result, the members of a single catalog service domain cannot span the boundaries of a core group, and a catalog service domain therefore cannot span cells. However, WebSphere eXtreme Scale container servers and clients can span cells by connecting to a catalog server across cell boundaries, such as a stand-alone catalog service domain or a catalog service domain embedded in another cell.

For remote catalog servers: You can connect WebSphere eXtreme Scale containers and clients to a catalog service domain that is running in another WebSphere Application Server cell or that are running as stand-alone processes. Because remotely configured catalog servers do not automatically start in the cell, you must manually start any remotely configured catalog servers. When you configure a remote catalog service domain, the domain name should match the domain name that you specified when you start the remote catalog servers. The default catalog service domain name for stand-alone catalog servers is `DefaultDomain`. Specify a catalog service domain name with the **`startOgServer`** command **`-domain`** parameter, a server properties file, or with the embedded server API. You must start each remote catalog server process in the remote domain with the same domain name. See "Starting a stand-alone catalog service" on page 461 for more information about starting catalog servers.

Attention: Do not collocate the catalog services with WebSphere eXtreme Scale container servers in a production environment. Include the catalog service in multiple node agent processes or in an application server that is not hosting a WebSphere eXtreme Scale application.

Procedure

1. Create the catalog service domain.
   a. In the WebSphere Application Server administrative console, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains** > **New**.
   b. Define a name, default value, and JMX authentication credentials for your catalog service domain. If you are configuring remote endpoints for the catalog service domain, the name of the catalog service domain should match the name of the catalog service domain that you specify when you start the catalog servers.
   c. Add catalog server endpoints. You can either select existing application servers or add remote servers that are running a catalog service.
2. Test the connection to the catalog servers within your catalog service domain. For existing application servers, catalog servers start when the associated application server is started. For remote application servers, you must start the servers manually using the **`startOgServer`** command or embedded server API.
   a. In the WebSphere Application Server administrative console, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains**.
   b. Select the catalog service domain that you want to test and click **Test connection**. When you click this button, all of the defined catalog service

domain end points are queried one by one, if any one end point is available, returns a message that indicates that the connection to the catalog service domain was successful.

Related concepts:

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

High availability catalog service
A catalog service domain is the data grid of catalog servers you are using, which retain topology information for all of the container servers in your eXtreme Scale environment. The catalog service controls balancing and routing for all clients.

Related reference:

"Catalog service domain administrative tasks"
You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Catalog service domain administrative tasks:

You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

Requirements

You must have the WebSphere eXtreme Scale Client installed in your WebSphere Application Server environment.

List all administrative tasks

To get a list of all of the administrative tasks that are associated with catalog service domains, run the following command with **`wsadmin`**:

- Using Jacl:
```
wsadmin>$AdminTask help XSDomainManagement
```
- Using a Jython string:
```
wsadmin>print AdminTask.help ('XSDomainManagement')
```

Commands

The administrative tasks for catalog service domains include the following commands:

- "createXSDomain"
- "deleteXSDomain" on page 305
- "getDefaultXSDomain" on page 306
- "listXSDomains" on page 306
- "modifyXSDomain" on page 306
- "testXSDomainConnection" on page 313
- "testXSServerConnection" on page 313

List all administrative task command arguments

To get a list of all of the command arguments associated with catalog service domain administrative tasks, run the following command with **wsadmin**:

- Using Jacl:

```
wsadmin>$AdminTask help <command>
wsadmin>$AdminTask help <command> <commandStep>
Example: wsadmin>$AdminTask help createXSDomain defineDomainServers
```

- Using a Jython string:

```
wsadmin>print AdminTask.help ('<command>')
Example: wsadmin>print AdminTask.help ('createXSDomain')
```

createXSDomain

The **createXSDomain** command registers a new catalog service domain.

Table 13. createXSDomain command arguments

Argument	Description
-name (required)	Specifies the name of the catalog service domain that you want to create.
-default	Specifies whether the catalog service domain is the default for the cell. The default value is `true`. (Boolean: set to `true` or `false`)

Table 14. defineDomainServers step arguments

Argument	Description
name_of_endpoint	Specifies the name of the catalog service domain endpoint. • **For existing application servers**: Specifies the name of the endpoint must be in the following format, using backslashes: *cell_name\node_name\server_name* • **For remote servers**: Specifies the host name of the remote server. You can have the same name for multiple endpoints, but the client port values must be unique for each endpoint.
custom_properties	Specifies custom properties for the catalog service domain endpoint. If you do not have any custom properties, use a set of double quotation marks ("") for this argument.

Table 14. defineDomainServers step arguments (continued)

Argument	Description
endpoint_ports	Specifies the port numbers for the catalog service domain endpoint. The ports must be specified in the following order: `<client_port>,<listener_port>` For existing application servers where only a client port is required, enter the client port value as either `"2809"` or `"2809,"`. For remote servers where only a listener port is required, enter the listener port value as: `",9810"` **Client Port** Specifies the port that is used for communication between the catalog servers in the catalog service domain. This value is only required for existing application servers (catalog servers that are running in WebSphere Application Server processes) and can be set to any port that is not being used elsewhere. **Listener Port** Specifies the port that is used for communication with clients. This value is required for remote endpoints and must match the value used when the catalog service was started. The listener port is used by clients and containers to communicate with the catalog service. **For WebSphere eXtreme Scale remote endpoints**: Defines the Object Request Broker (ORB) listener port for containers and clients to communicate with the catalog service through the ORB. For WebSphere Application Server endpoints, the listener port value is optional because the value is inherited from the BOOTSTRAP_ADDRESS port configuration.

Table 15. configureClientSecurity step arguments

Argument	Description
`-securityEnabled`	Specifies that client security is enabled for the catalog server. The server properties file that is associated with the selected catalog server must have a matching **`securityEnabled`** setting in the server properties file. If these settings do not match, an exception results. (Boolean: set to `true` or `false`)

Table 15. configureClientSecurity step arguments (continued)

Argument	Description
-credentialAuthentication (optional)	Indicates if credential authentication is enforced or supported. **Never** No client certificate authentication is enforced. **Required** Credential authentication is always enforced. If the server does not support credential authentication, the client cannot to connect to the server. **Supported** (Default) Credential authentication is enforced only if both the client and server support credential authentication.
-authenticationRetryCount (optional)	Specifies the number of times that authentication gets tried again if the credential is expired. If you do not want to try authentication again, set the value to `0`. The default value is `0`.
-credentialGeneratorClass	Indicates the com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator implementation class, so the client retrieves the security tokens from the thread.

-credentialGeneratorProps	Specifies the properties for the CredentialGenerator implementation class. The properties are sent to the object with the setProperties(String) method. The credential generator properties value is used only when a value is specified for the **Credential generator class** field.
	Properties for
	`com.ibm.websphere.objectgrid.security.plugins.`
	`UserPasswordCredentialGenerator`
	includes `userid_password` which can be defined as `"userid password"`.
	Note: Because parsing of the `userid_password` property depends upon the space character as the value separator, userids and passwords which contain spaces must use the `"\20"` escape character to represent a space. For example: If the userid is `"Test User Id"` and the password is `"Test Password"`, the `userid_password` property should be entered as: `"Test\20User\20Id Test\20Password"`.
	Properties for
	`com.ibm.websphere.objectgrid.security.plugins.`
	`builtins.WSTokenCredentialGenerator`
	includes the property `subject_type`, which can be defined as either`"runAs"` or `"caller"`.

Return value:

Batch mode example usage

Batch mode requires correct formatting of the command entry. Consider using interactive mode to ensure the values that you enter are processed correctly. When you use batch mode, you must define the **-defineDomainServers** step arguments using a specific array of properties. This array of properties is in the format *name_of_endpoint custom_properties endpoint_ports*. The *endpoint_ports* value is a list of ports that must be specified in the following order: *<client_port>*,*<listener_port>*.

- Create a catalog service domain of remote endpoints using Jacl:

```
$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"}}
```

- Create a catalog service domain of remote endpoints using Jython string:

```
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"] ]')
```

- Create a catalog service domain of existing application server endpoints using Jacl:

```
$AdminTask createXSDomain {-name TestDomain -default true -defineDomainServers
 {{cellName/nodeName/serverName "" 1109}}}
```

Interactive mode example usage

- Using Jacl:

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

- Using Jython string:

```
AdminTask.createXSDomain ('[-interactive]')
```

deleteXSDomain

The **deleteXSDomain** command deletes a catalog service domain.

Required parameters:

-name
Specifies the name of the catalog service domain to delete.

Return value:

Batch mode example usage

- Using Jacl:

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

- Using Jython string:

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

Interactive mode example usage

- Using Jacl:

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

- Using Jython string:

```
AdminTask.deleteXSDomain ('[-interactive]')
```

getDefaultXSDomain

The **getDefaultXSDomain** command returns the default catalog service domain for the cell.

Required parameters: None

Return value: The name of the default catalog service domain.

Batch mode example usage

- Using Jacl:
  ```
  $AdminTask getDefaultXSDomain
  ```
- Using Jython string:
  ```
  AdminTask.getDefaultXSDomain
  ```

Interactive mode example usage

- Using Jacl:
  ```
  $AdminTask getDefaultXSDomain {-interactive}
  ```
- Using Jython string:
  ```
  AdminTask.getDefaultXSDomain ('[-interactive]')
  ```

listXSDomains

The **listXSDomains** command returns a list of the existing catalog service domains.

Required parameters: None

Return value: A list of all of the catalog service domains in the cell.

Batch mode example usage

- Using Jacl:
  ```
  $AdminTask listXSDomains
  ```
- Using Jython string:
  ```
  AdminTask.listXSDomains
  ```

Interactive mode example usage

- Using Jacl:
  ```
  $AdminTask listXSDomains {-interactive}
  ```
- Using Jython string:
  ```
  AdminTask.listXSDomains ('[-interactive]')
  ```

modifyXSDomain

The **modifyXSDomain** command modifies an existing catalog service domain.

Batch mode requires correct formatting of the command entry. Consider using interactive mode to ensure the values that you enter are processed correctly. When you use batch mode, you must define the **-modifyEndpoints**, **-addEndpoints** and **-removeEndpoints** step arguments using a specific array of properties. This array of properties is in the format *name_of_endpoint host_name custom_properties endpoint_ports*. The *endpoint_ports* value is a list of ports that must be specified in the following order: <*client_port*>**,**<*listener_port*>.

Table 16. modifyXSDomain command arguments

Argument	Description
-name (required)	Specifies the name of the catalog service domain that you want to edit.
-default	If set to `true`, specifies that the selected catalog service domain is the default for the cell. (Boolean)

Table 17. modifyEndpoints step arguments

Argument	Description
name_of_endpoint	Specifies the name of the catalog service domain endpoint. • **For existing application servers**: Specifies the name of the endpoint in the following format, using backslashes: `cell_name\node_name\server_name` • **For remote servers**: Specifies the host name of the remote server. You can have the same name for multiple endpoints, but the listener port values must be unique for each endpoint.

Table 17. modifyEndpoints step arguments (continued)

Argument	Description
endpoint_ports	Specifies the port numbers for the catalog service domain endpoint. The endpoints must be specified in the following order: `<client_port>,<listener_port>` For existing application servers where only a client port is required, enter the client port value as either: `"2809"` or `"2809,"`. For remote servers where only a listener port is required, enter the listener port value as: `",9810"`. **Client Port** Specifies the port that is used for communication between the catalog servers in the catalog service domain. This value is only required for existing application servers (catalog servers that are running in WebSphere Application Server processes) and can be set to any port that is not being used elsewhere. **Listener Port** Specifies the port that is used for communication with clients. This value is required for remote endpoints and must match the value used when the catalog service was started. The listener port is used by clients and containers to communicate with the catalog service. **For WebSphere eXtreme Scale remote endpoints**: Defines the Object Request Broker (ORB) listener port for containers and clients to communicate with the catalog service. For WebSphere Application Server endpoints, specifying the listener port value is optional. The value is inherited from the BOOTSTRAP_ADDRESS port configuration.

Table 18. addEndpoints step arguments

Argument	Description
name_of_endpoint	Specifies the name of the catalog service domain endpoint. • **For existing application servers**: Specifies the name of the endpoint in the following format, using backslashes: `cell_name\node_name\server_name` • **For remote servers**: Specifies the host name of the remote server. You can have the same name for multiple endpoints, but the listener port values must be unique for each endpoint.
custom_properties	Specifies custom properties for the catalog service domain endpoint. If you do not have any custom properties, use a set of double quotation marks ("") for this argument.

Table 18. addEndpoints step arguments (continued)

Argument	**Description**
endpoint_ports	Specifies the port numbers for the catalog service domain endpoint. The endpoints must be specified in the following order: `<client_port>,<listener_port>` For existing application servers where only a client port is required, enter the client port value as either: `"2809"` or `"2809,"`. For remote servers where only a listener port is required, enter the listener port value as : `",9810"`. **Client Port** Specifies the port that is used for communication between the catalog servers in the catalog service domain. This value is only required for existing application servers (catalog servers that are running in WebSphere Application Server processes) and can be set to any port that is not being used elsewhere. **Listener Port** Specifies the port that is used for communication with clients. This value is required for remote endpoints and must match the value used when the catalog service was started. The listener port is used by clients and containers to communicate with the catalog service. **For WebSphere eXtreme Scale remote endpoints**: Defines the Object Request Broker (ORB) listener port for containers and clients to communicate with the catalog service through the ORB. For WebSphere Application Server endpoints, specifying the listener port value is optional because the value is inherited from the BOOTSTRAP_ADDRESS port configuration.

Table 19. removeEndpoints step arguments

Argument	**Description**
name_of_endpoint	Specifies the name of the catalog service endpoint to delete.

Table 20. configureClientSecurity step arguments

Argument	Description
`-securityEnabled`	Specifies that client security is enabled for the catalog server. The server properties file that is associated with the selected catalog server must have a matching **`securityEnabled`** setting in the server properties file. If these settings do not match, an exception results. (Boolean: set to `true` or `false`)
`-credentialAuthentication` (optional)	Indicates whether credential authentication is enforced or supported. **Never** No client certificate authentication is enforced. **Required** Credential authentication is always enforced. If the server does not support credential authentication, the client cannot to connect to the server. **Supported** (Default) Credential authentication is enforced only if both the client and server support credential authentication.
`-authenticationRetryCount` (optional)	Specifies the number of times that authentication gets tried again if the credential is expired. If you do not want to try authentication again, set the value to `0`. The default value is `0`.
`-credentialGeneratorClass`	Indicates the com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator implementation class, so the client retrieves the security tokens from the thread.

Table 20. configureClientSecurity step arguments (continued)

Argument	Description

`-credentialGeneratorProps`	Specifies the properties for the CredentialGenerator implementation class. The properties are sent to the object with the setProperties(String) method. The credential generator properties value is used only when a value is specified for the **Credential generator class** field.
	
	Properties for
	`com.ibm.websphere.objectgrid.security.plugins.`
	`UserPasswordCredentialGenerator`
	
	includes `userid_password` which can be defined as `"userid password"`.
	Note: Because parsing of the `userid_password` property depends upon the space character as the value separator, userids and passwords which contain spaces must use the `"\20"` escape character to represent a space. For example: If the userid is `"Test User Id"` and the password is `"Test Password"`, the `userid_password` property should be entered as: `"Test\20User\20Id Test\20Password"`.
	
	Properties for
	`com.ibm.websphere.objectgrid.security.plugins.`
	`builtins.WSTokenCredentialGenerator`
	
	includes the property `subject_type`, which can be defined as either`"runAs"` or `"caller"`.

Return value:

Batch mode example usage

- Using Jacl:

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

- Using Jython string:

```
AdminTask.modifyXSDomain('[-name TestDomain
-default false -modifyEndpoints [[xhost1.ibm.com "" ,2809]]
-addEndpoints [[xhost3.ibm.com "" ,2809]]
-removeEndpoints [[xhost2.ibm.com]]]')
```

- Using the client security specification during the modify command:

```
$AdminTask modifyXSDomain {-name myDomain -default false
-configureClientSecurity {-securityEnabled true -
Supported -authenticationRetryCount 1 -credentialGeneratorClass
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
 -credentialGeneratorProps "manager manager1"}}
```

Interactive mode example usage

- Using Jacl:

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

- Using Jython string:

```
AdminTask.modifyXSDomain ('[-interactive]')
```

testXSDomainConnection

The **`testXSDomainConnection`** command tests the connection to a catalog service domain.

Required parameters:

`-name`
: Specifies the name of the catalog service domain to which to test the connection.

Optional parameters

`-timeout`
: Specifies the maximum amount of time to wait for the connection, in seconds.

Return value: If a connection can be made, returns `started`, otherwise, returns `stopped`.

Batch mode example usage

- Using Jacl:
  ```
  $Admintask testXSDomainConnection
  ```
- Using Jython string:
  ```
  AdminTask.testXSDomainConnection
  ```

Interactive mode example usage

- Using Jacl:
  ```
  $AdminTask testXSDomainConnection {-interactive}
  ```
- Using Jython string:
  ```
  AdminTask.testXSDomainConnection ('[-interactive]')
  ```

testXSServerConnection

The **`testXSServerConnection`** command tests the connection to a catalog server. This command works for both stand-alone servers and servers that are a part of a catalog service domain.

Required parameters:

`host`
: Specifies the host on which the catalog server resides.

`listenerPort`
: Specifies the listener port for the catalog server.

Optional parameters

`timeout`
: Specifies the maximum amount of time to wait for a connection to the catalog server, in seconds.

`domain`
: Specifies the name of a catalog service domain. If you define a value for this parameter, the client security properties for the specified catalog service domain are used to test the connection. Otherwise, a search occurs to find the catalog service domain for the specified host and listener port. If a catalog

service domain is found, the client security properties that are defined for the catalog service domain are used to test the server. Otherwise, no client security properties are used during the test.

Return value: If a connection can be made, returns `started`, otherwise returns `stopped`.

Batch mode example usage

- Using Jacl:
  ```
  $Admintask testXSServerConnection {-host xhost1.ibm.com -listenerPort 2809}
  ```
- Using Jython string:
  ```
  AdminTask.testXSServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
  ```

Interactive mode example usage

- Using Jacl:
  ```
  $AdminTask testXSServerConnection {-interactive}
  ```
- Using Jython string:
  ```
  AdminTask.testXSServerConnection ('[-interactive]')
  ```

Related concepts:

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

High availability catalog service
A catalog service domain is the data grid of catalog servers you are using, which retain topology information for all of the container servers in your eXtreme Scale environment. The catalog service controls balancing and routing for all clients.

Related tasks:

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

"Configuring the catalog service in WebSphere Application Server" on page 298
Catalog service processes can run in WebSphere Application Server. The server life cycle in WebSphere Application Server determines when the catalog service starts and stops.

"Creating catalog service domains in WebSphere Application Server" on page 299
Catalog service domains define a group of catalog servers that manage the placement of shards and monitor the health of container servers in your data grid.

"Starting and stopping servers in a WebSphere Application Server environment" on page 475
Catalog and container servers can automatically start in a WebSphere Application Server or WebSphere Application Server Network Deployment environment.

Catalog service domain collection:

Use this page to manage catalog service domains. Catalog service domains define a group of catalog servers that manage the placement of shards and monitors the health of container servers in your data grid.

To view this administrative console page, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains**. To create a new catalog service domain, click **New**. To delete a catalog service domain, select the catalog service domain you want to remove and click **Delete**.

Test Connection:

When you click the **Test connection** button, all of the defined catalog service domain end points are queried one by one, if any one end point is available, returns a message that indicates that the connection to the catalog service domain was successful. You can use this button to test that you have configured the connection and security information correctly.

Set Default:

Defines the catalog service domain that is used as the default. Select one catalog service domain as the default and click **Set default**. Only one catalog service domain can be selected as the default.

Name:

Specifies the name for the catalog service domain.

Default:

Specifies which catalog service domain in the list is the default. The default catalog service domain is indicated with the following icon: ✓.

Catalog service domain settings:

Use this page to manage the settings for a specific catalog service domain. Catalog service domains define a group of catalog servers that manage the placement of shards and monitors the health of container servers in your data grid. You can define a catalog service domain that is in the same cell as your deployment manager. You can also define remote catalog service domains if your WebSphere eXtreme Scale configuration is in a different cell or your data grid is made up of Java SE processes.

To view this administrative console page, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains** > ***catalog_service_domain_name***.

Test connection:

When you click the **Test connection** button, all of the defined catalog service domain end points are queried one by one, if any one end point is available, returns a message that indicates that the connection to the catalog service domain was successful. You can use this button to test that you have configured the connection and security information correctly.

Name:

Specifies the name of the catalog service domain.

Enable this catalog service domain as the default unless another catalog service domain is explicitly specified:

If you select this check box, the selected catalog service domain becomes the default catalog service domain for the cell. Each server profile in the cell that is augmented with the WebSphere eXtreme Scale profile belongs to the selected catalog service domain.

For WebSphere eXtreme Scale, all eXtreme Scale containers that are embedded in Java EE application modules connect to the default domain. Clients can connect to the default domain using the ServerFactory.getServerProperties().getCatalogServiceBootstrap() API to retrieve the catalog service endpoints to use when calling the ObjectGridManager.connect() API.

If you change the default domain to point to a different set of catalog servers, then all containers and clients refer to the new domain after they are restarted.

Catalog servers:

Specifies a list of catalog servers that belong to this catalog service domain.

Click **New** to add a catalog server to the list. This catalog server must already exist in the eXtreme Scale configuration. You can also edit or delete a server from the list by selecting the endpoint and then clicking **Edit** or **Delete**. Define the following properties for each catalog server endpoint:

Catalog server endpoint
Specifies the name of the existing application server or remote server on which the catalog service is running. A catalog service domain cannot contain a mix of existing application servers and remote server endpoints.

- **Existing application server**: Specifies the path of an application server, node agent, or deployment manager in the cell. A catalog service starts automatically in the selected server. Select from the list of the existing application servers. All of the application servers that you define within the catalog service domain must be in the same core group.
- **Remote server**: Specifies the host name of the remote catalog server.

  For WebSphere eXtreme Scale remote endpoints: Specifies the host name of the remote catalog server process. You must start the remote servers with the `startOgServer` script or the embedded server API.

Client Port

Specifies the port that is used for communication between the catalog servers in the catalog service domain. This value is required for catalog servers that are running in WebSphere Application Server processes. You can set the value to any port that is not being used by another process.

Listener Port

Specifies the port that is used for communication with clients. This value is required for remote endpoints and must match the value used when the catalog service was started. The listener port is used by clients and containers to communicate with the catalog service.

For WebSphere eXtreme Scale remote endpoints: Defines the Object Request Broker (ORB) listener port for containers and clients to communicate with the catalog service through the ORB. For WebSphere Application Server endpoints, the listener port value is inherited from the BOOTSTRAP_ADDRESS port configuration.

Status

Table 21. Catalog server endpoint status

Icon	Definition
❓	Unknown
➡	Started
❌	Stopped

Client security properties:

Use this page to configure client security for a catalog service domain. These settings apply to all the servers in your catalog service domain. These properties can be overridden by specifying a `splicer.properties` file with the com.ibm.websphere.xs.sessionFilterProps custom property or by splicing the application EAR file.

To view this administrative console page, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains** > ***catalog_service_domain_name*** > **Client security properties**.

Enable client security:

Specifies that client security is enabled for the catalog server. The server properties file that is associated with the selected catalog server must have a matching **`securityEnabled`** setting in the server properties file. If these settings do not match, an exception results.

Credential authentication:

Indicates if credential authentication is enforced or supported.

Never

No client credential authentication is enforced.

Required
: Credential authentication is always enforced. If the server does not support credential authentication, the client cannot to connect to the server.

Supported
: Credential authentication is enforced only if both the client and server support credential authentication.

Authentication retry count:

Specifies the number of times that authentication gets tried again if the credential is expired.

If you do not want to try authentication again, set the value to `0`.

Credential generator class:

Indicates the com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator implementation class, so the client retrieves the credential from the CredentialGenerator object.

You can choose from two predefined credential generator classes, or you can specify a custom credential generator. If you choose a custom credential generator, you must indicate the name of the credential generator class.

- com.ibm.websphere.objectgrid.security.plugins.builtins. UserPasswordCredentialGenerator
- com.ibm.websphere.objectgrid.security.plugins. UserPasswordCredentialGenerator
- Custom credential generator

Subject type:

Specifies if you are using the J2EE caller or the J2EE runAs subject type. You must specify this value when you choose the WSTokenCredentialGenerator credential generator.

- **runAs**: The subject contains the principal of the J2EE run as identity and the J2EE run as credential.
- **caller**: The subject contains the principal of the J2EE caller and the J2EE caller credential.

User ID:

Specify a user ID when you are using the UserPasswordCredentialGenerator credential generator implementation.

Password:

Specify a password when you are using the UserPasswordCredentialGenerator credential generator implementation.

Credential generator properties:

Specifies the properties for a custom CredentialGenerator implementation class. The properties are set in the object with the setProperties(String) method. The credential generator properties value is used only when a value is specified for the **Credential generator class** field.

Catalog service domain custom properties:

You can further edit the configuration of the catalog service domain by defining custom properties.

To view this administrative console page, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains** > **Custom properties**. To create a new custom property, click **New**.

Name:

Specifies the name of the custom property for the catalog service domain.

Value:

Specifies a value for the custom property for the catalog service domain.

Configuring the quorum mechanism

The quorum mechanism is configured for each catalog service. You must enable the quorum mechanism on all of the catalog servers in the catalog service domain.

Before you begin

Before you enable the quorum mechanism, you must configure a topology that supports this type of configuration. The configuration must support the following requirements:

- **Flat IP address space:** Any addressable element on the network must be able to connect to any other addressable element on the network unimpeded. You must use a flat IP address naming space. All the firewalls in the configuration must allow all traffic to flow between the IP addresses and ports that are being used to host catalog servers and container servers.
- **Number of catalog servers:** You must start at least one catalog server for each data center in the configuration.
- **Heartbeat interval setting:** If you do not define the heartbeat interval, the default value is 30 seconds. WebSphere eXtreme Scale checks on the JVMs in a single zone at the defined interval. For example, if a heartbeat on a container server is missed, and quorum is established, a failover event occurs to place a new container server. See "Tuning the heartbeat interval setting for failover detection" on page 321 for more information.
- **Transport security:** Because data centers are normally deployed in different geographical locations, you might want to enable transport security between the data centers for security reasons. Read about transport layer security in the *Administration Guide*.

About this task

The quorum mechanism is disabled by default. Enable the quorum mechanism in the following scenarios:

- When your catalog service domain spans a network that is unpredictable or unstable. This type of network might span multiple data centers.
- When you want to prevent the data grid from self-healing during a brownout on the network, and instead temporarily pause data grid operations from occurring.

You can leave the quorum mechanism disabled if your catalog service domain is contained within a single data center, or is on a local area network (LAN). In this type of configuration, default heart beating is used and brownouts are assumed to be shorter than 10 seconds. Because the detection period is approximately 30 seconds, any short brownouts that occur do not cause placement changes to occur in the data grid.

If you enable quorum, all the catalog servers must be available and communicating with the data grid to conduct placement operations. If a network brownout occurs, placement is paused until all the catalog servers are available. If a data center failure occurs, manual administrator actions are required to remove the failed catalog server from the quorum.

Procedure

1. **Enable quorum on the catalog servers.** In WebSphere Application Server, you must configure quorum with the server properties file. In a stand-alone environment you can either use the properties method or enable quorum when you start the server:
   - **Set the `enableQuorum=true` property in the server properties file.**

     You can use this configuration in a WebSphere Application Server or stand-alone environment. See the following example `objectGridServer.properties` file:

     ```
     catalogClusterEndPoints=cat0:cat0.domain.com:6600:6601,
     cat1:cat1.domain.com:6600:6601
     catalogServiceEndPoints= cat0.domain.com:2809, cat1.domain.com:2809
     enableQuorum=true
     ```

     For more information about configuring the properties file, see Server properties file.
   - **Pass the `-quorum` enabled flag on the `startOgServer` command.**

     You can use this configuration method when you start stand-alone servers only.

     ```
     # bin/startOgServer cat0 –serverProps objectGridServer.properties -quorum true
     ```

     For more information about the **`startOgServer`** command, see “Starting and stopping stand-alone servers” on page 459.
2. **Start container servers in the same zone.**

   When you are running a data grid across data centers, the servers must use the zone information to identify the data center in which they reside. Setting the zone on the container servers allows WebSphere eXtreme Scale to monitor health of the container servers that are scoped to the data center, minimizing cross-data-center traffic. The container server JVMs in a core group must never span multiple LANs that are connected with links, like in a wide area network. See “Defining zones for container servers” on page 287 for more information about defining zones for container servers.

   Container server JVMs are tagged with a zone identifier. The data grid of container JVMs is automatically broken in to small core groups of JVMs. A core group only includes JVMs from the same zone. JVMs from different zones are never in the same core group.

   A core group aggressively tries to detect the failure of its member JVMs.

Results

By setting the quorum mechanism to be enabled on the catalog servers within a catalog service domain, all the catalog servers must be available for data grid placement operations to occur. In the event of a short network brownout, placement operations are temporarily stopped until all the catalog servers in the quorum are available.

You can add additional catalog servers to the quorum by repeating these steps.

What to do next

- You can remove a catalog server from the quorum by stopping the catalog server with the administrative method that is required by the configuration. When a catalog server is stopped through administrative actions, quorum is automatically reestablished among the remaining catalog servers, and placement can continue. If you restart the catalog server with the steps described in this topic, the catalog server can rejoin the quorum.
- If a long-term or permanent failure of a catalog server that is in the currently defined quorum occurs, you must override the quorum mechanism so that placement can continue. See "Managing data center failures when quorum is enabled" on page 491 for more information about overriding the quorum mechanism.

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Catalog server quorums
When the quorum mechanism is enabled, all the catalog servers in the quorum must be available for placement operations to occur in the data grid.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Tuning the heartbeat interval setting for failover detection

You can configure the amount of time between system checks for failed servers with the heartbeat interval setting. This setting applies to catalog servers only.

About this task

Configuring failover varies depending on the type of environment you are using. If you are using a stand-alone environment, you can configure failover with the command line. If you are using a WebSphere Application Server Network Deployment environment, you must configure failover in the WebSphere Application Server Network Deployment administrative console.

Procedure

- Configure failover for stand-alone environments.
  - With the **`-heartbeat`** parameter in the **`startOgServer`** script when you start the catalog server.
  - With the heartBeatFrequencyLevel property in the server properties file for the catalog server.

  Use one of the following values:

Table 22. Valid heartbeat values

Value	Action	Description
-1	Aggressive	Specifies an aggressive heartbeat level. With this value, failures are detected more quickly, but more processor and network resources are used. This level is more sensitive to missing heartbeats when the server is busy. Failovers are typically detected within 5 seconds.
0	Typical (default)	Specifies a heartbeat level at a typical rate. With this value, failover detection occurs at a reasonable rate without overusing resources. Failovers are typically detected within 30 seconds.
1	Relaxed	Specifies a relaxed heartbeat level. With this value, a decreased heartbeat frequency increases the time to detect failures, but also decreases processor and network use. Failovers are typically detected within 180 seconds.

  An aggressive heartbeat interval can be useful when the processes and network are stable. If the network or processes are not optimally configured, heartbeats might be missed, which can result in a false failure detection.

- Configure failover for WebSphere Application Server environments.

  You can configure WebSphere Application Server Network Deployment

  7.1.1 Version 6.1 and later to allow WebSphere eXtreme Scale to fail over very quickly. The default failover time for hard failures is approximately 200 seconds. A hard failure is a physical computer or server crash, network cable disconnection or operating system error. Failures because of process crashes or soft failures typically fail over in less than one second. Failure detection for soft failures occurs when the network sockets from the dead process are closed automatically by the operating system for the server hosting the process.

  Core group heartbeat configuration

  WebSphere eXtreme Scale running in a WebSphere Application Server process inherits the failover characteristics from the core group settings of the application server. The following sections describe how to configure the core group heartbeat settings for different versions of WebSphere Application Server Network Deployment:

  - **Update the core group settings for WebSphere Application Server Network Deployment 7.1.1 Version 6.1 and 7.0:**

    Specify the heartbeat interval in seconds on WebSphere Application Server versions from Version 6.0 through Version 6.1.0.12 or in milliseconds starting with Version 6.1.0.13. You must also specify the number of missed heartbeats. This value indicates how many heartbeats can be missed before a peer Java virtual machine (JVM) is considered as failed. The hard failure detection time is approximately the product of the heartbeat interval and the number of missed heartbeats.

    These properties are specified using custom properties on the core group using the WebSphere administrative console. See Core group custom properties for configuration details. These properties must be specified for all core groups used by the application:

- The heartbeat interval is specified using either the IBM_CS_FD_PERIOD_SEC custom property for seconds or the IBM_CS_FD_PERIOD_MILLIS custom property for milliseconds (requires Version 6.1.0.13 or later).
- The number of missed heartbeats is specified using the IBM_CS_FD_CONSECUTIVE_MISSED custom property.

The default value for the IBM_CS_FD_PERIOD_SEC property is 20 and for the IBM_CS_FD_CONSECUTIVE_MISSED property is 10. If the IBM_CS_FD_PERIOD_MILLIS property is specified, then it overrides any of the set IBM_CS_FD_PERIOD_SEC custom properties. The values of these properties are positive integer values.

Use the following settings to achieve a 1500 ms failure detection time for WebSphere Application Server Network Deployment Version 6.x servers:

- Set `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment V6.1.0.13 and later)
- Set `IBM_CS_FD_CONSECUTIVE_MISSED = 2`

– **Update the core group settings for WebSphere Application Server Network Deployment Version 7.0**

WebSphere Application Server Network Deployment Version 7.0 provides two core group settings that can be adjusted to increase or decrease failover detection:

- **Heartbeat transmission period.** The default is 30000 milliseconds.
- **Heartbeat timeout period.** The default is 180000 milliseconds.

For more details on how change these settings, see the WebSphere Application Server Network Deployment Information center: Discovery and failure detection settings.

Use the following settings to achieve a 1500 ms failure detection time for WebSphere Application Server Network Deployment Version 7 servers:

- Set the heartbeat transmission period to 750 milliseconds.
- Set the heartbeat timeout period to 1500 milliseconds.

What to do next

When these settings are modified to provide short failover times, there are some system-tuning issues to be aware of. First, Java is not a real-time environment. It is possible for threads to be delayed if the JVM is experiencing long garbage collection times. Threads might also be delayed if the machine hosting the JVM is heavily loaded (due to the JVM itself or other processes running on the machine). If threads are delayed, heartbeats might not be sent on time. In the worst case, they might be delayed by the required failover time. If threads are delayed, false failure detections occur. The system must be tuned and sized to ensure that false failure detections do not happen in production. Adequate load testing is the best way to ensure this.

Note: The current version of eXtreme Scale supports WebSphere Real Time.

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Configuring container servers

The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions, which are hosted across multiple container servers. Each container server in turn hosts a subset of the complete data.

About this task

- **Stand-alone container servers**:

  Configure stand-alone container servers with a server properties file and a deployment policy XML file. Control the life cycle of a container server with the start and stop scripts or with the embedded server API.
- **Container servers that start in WebSphere Application Server:**

  Configure container servers in WebSphere Application Server with a server properties file and deployment policy XML file that is embedded into a Java EE application module. The life cycle of the container servers is controlled by the application. Container servers start and stop with the application.

Container server reconnect properties

Use Java virtual machine (JVM) properties to configure how your container server reconnects to the data grid if the container server becomes disconnected.

JVM system properties

If a container server becomes disconnected from the data grid, WebSphere eXtreme Scale attempts to reconnect those container servers. By setting system properties, you can control how the container reconnects. You can set these properties when you start a container server. Some properties are applicable to WebSphere eXtreme Scale in a stand-alone environment, while others are only applicable in an integrated WebSphere Application Server environment. For example, when a container server is started in a stand-alone environment, you can set these properties as an option from a command console:

```
startOgServer.sh server01 -objectgridFile objectgrid.xml
-deploymentPolicyFile deployment.xml
-Dcom.ibm.websphere.objectgrid.container.reconnect.restart=false
```

For more information, see "Starting and stopping stand-alone servers" on page 459. If you want to set the appropriate property for WebSphere eXtreme Scale for

WebSphere Application Server, you can use the WebSphere Integrated Solutions Console tool. This tool is a graphical user interface that is accessible from the WebSphere Application Server environment, and is installed as an extension to the WebSphere ISC.

`com.ibm.websphere.objectgrid.container.reconnect.block.reconnect.time`
: Defines the amount of time (in milliseconds) to block another container reconnect call. Only valid when a container server is started for the product offering:WebSphere eXtreme Scale for WebSphere Application Server.

  Default: 30000 milliseconds

`com.ibm.websphere.objectgrid.container.reconnect.min.successful.heartbeats`
: Defines the minimum number of successful heartbeats before a container can be stopped. Only valid when a container server is started for the product offering: WebSphere eXtreme Scale for WebSphere Application Server.

  Default: 10

`com.ibm.websphere.objectgrid.container.reconnect.restart`
: Defines whether container reconnect can restart the JVM. Only valid when a container server is started for WebSphere eXtreme Scale in a stand-alone environment.

  Default: true

`com.ibm.websphere.objectgrid.container.reconnect.restart.delay`
: Defines the time (in milliseconds) to delay before you proceed with the startup on the newly created child container when the JVM is restarted. Only valid when a container server is started for the product offering: WebSphere eXtreme Scale in a stand-alone environment.

  Default: 2000 milliseconds

`com.ibm.websphere.objectgrid.container.reconnect.restart.parent.timeout`
: Defines the time (in milliseconds) for the newly created child container to wait for parent death before timeout when the JVM is restarted. Only valid when a container server is started for the product offering: WebSphere eXtreme Scale in a stand-alone environment.

  Default: 180000 milliseconds

`com.ibm.websphere.objectgrid.container.reconnect.retry.forever`
: Defines whether the container attempts to reconnect to the container server forever. Only valid when a container server is started for the product offering:WebSphere eXtreme Scale for WebSphere Application Server.

  Default: false

Configuring container servers in WebSphere Application Server

Configure container servers in WebSphere Application Server by using a server properties file and deployment policy XML file that is embedded into a Java EE application module. Container servers stop and start when the application is stopped and started.

Before you begin

Configure a catalog service domain. See "Creating catalog service domains in WebSphere Application Server" on page 299 for more information.

About this task

To create container servers in WebSphere Application Server, you must embed the WebSphere eXtreme Scale configuration XML files to create the container servers within the application module.

Procedure

1. Identify the application servers on which you want to deploy the Java EE application that contains the WebSphere eXtreme Scale container server definitions. Verify that the target application server profiles have been augmented with the WebSphere eXtreme Scale profile. In a production environment, do not collocate the servers that you use for container servers with the catalog servers. See "Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203 for more information.
2. Configure a server properties file and add the server properties file to the class path for each target application server node. See Server properties file for more information.
3. Add the ObjectGrid descriptor XML file and deployment policy XML file to the application module. See "Configuring WebSphere Application Server applications to automatically start container servers" on page 327 for more information.

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

Related reference:

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Configuring WebSphere Application Server applications to automatically start container servers:

Container servers in a WebSphere Application Server environment start automatically when a module starts that has the eXtreme Scale XML files included.

Before you begin

WebSphere Application Server and WebSphere eXtreme Scale must be installed, and you must be able to access the WebSphere Application Server administrative console.

About this task

Java Platform, Enterprise Edition applications have complex class loader rules that greatly complicate loading classes when using a shared data grid within a Java EE server. A Java EE application is typically a single Enterprise Archive (EAR) file. The EAR file contains one or more deployed Enterprise JavaBeans (EJB) or web archive (WAR) modules.

WebSphere eXtreme Scale watches for each module start and looks for eXtreme Scale XML files. If the catalog service detects that a module starts with the XML files, the application server is registered as a container server Java virtual machine (JVM). By registering the container servers with the catalog service, the same application can be deployed in different data grids, but used as a single data grid by the catalog service. The catalog service is not concerned with cells, grids, or dynamic grids. A single data grid can span multiple cells if required.

Procedure

1. Package your EAR file to have modules that include the eXtreme Scale XML files in the `META-INF` folder. WebSphere eXtreme Scale detects the presence of the `objectGrid.xml` and `objectGridDeployment.xml` files in the `META-INF` folder of EJB and WEB modules when they start. If only an `objectGrid.xml` file is found, then the JVM is assumed to be client. Otherwise, it is assumed this JVM acts as a container for the data grid that is defined in the `objectGridDeployment.xml` file.

   You must use the correct names for these XML files. The file names are case-sensitive. If the files are not present, then the container does not start. You can check the `systemout.log` file for messages that indicate that shards are being placed. An EJB module or WAR module using eXtreme Scale must have eXtreme Scale XML files in its `META-INF` directory.

   The eXtreme Scale XML files include:

   - An ObjectGrid descriptor XML file, named `objectGrid.xml`. See ObjectGrid descriptor XML file for more information.
   - A deployment descriptor XML file named `objectGridDeployment.xml`. See Deployment policy descriptor XML file for more information.
   - (Optional) An entity metadata descriptor XML file, if entities are used. The `entity.xml` file name must match the name that is specified in the `objectGrid.xml` file. See Entity metadata descriptor XML file for more information.

   The run time detects these files, then contacts the catalog service to inform it that another container is available to host shards for that eXtreme Scale.

Tip: If your application has entities and you are planning to use one container server, set the **`minSyncReplicas`** value to `0` in the deployment descriptor XML file. Otherwise, you might see one of the following messages in the `SystemOut.log` file because placement cannot occur until another server starts to meet the minSyncReplica policy:

```
CWPRJ1005E: Error resolving entity association. Entity=entity_name,
association=association_name.
```

```
CWOBJ3013E: The EntityMetadata repository is not available.  Timeout
threshold reached when trying to register the entity: entity_name.
```

2. Deploy and start your application.

   The container starts automatically when the module is started. The catalog service starts to place partition primaries and replicas (shards) as soon as possible. This placement occurs immediately unless you configure the environment to delay placement. For more information, see "Controlling placement" on page 486.

What to do next

Applications within the same cell as the containers can connect to these data grids by using a ObjectGridManager.connect(null, null) method and then call the getObjectGrid(ccc, "object grid name") method. The connect or getObjectGrid methods might be blocked until the containers have placed the shards, but this blocking is only an issue when the data grid is starting.

ClassLoaders

Any plug-ins or objects stored in an eXtreme Scale are loaded on a certain class loader. Two EJB modules in the same EAR can include these objects. The objects are the same but are loaded with different ClassLoaders. If application A stores a Person object in a map that is local to the server, application B receives a `ClassCastException` if it tries to read that object. This exception occurs because application B loaded the Person object on a different class loader.

One approach to resolve this problem is to have a root module contain the necessary plug-ins and objects that are stored in the eXtreme Scale. Each module that uses eXtreme Scale must reference that module for its classes. Another resolution is to place these shared objects in a utility JAR file that is on a common class loader shared by both modules and applications. The objects can also be placed in the WebSphere classes or `lib/ext` directory, however this placement complicates the deployment.

EJB modules in an EAR file typically share the same ClassLoader and are not affected by this problem. Each WAR module has its own ClassLoader and is affected by this problem.

Connecting to a data grid client-only

If the **`catalog.services.cluster`** property is defined in the cell, node or server custom properties, any module in the EAR file can call the ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null) method to get a ClientClusterContext. The module can also call the ObjectGridManager.getObjectGrid(ccc, "grid name") method to gain a reference to the data grid. If any application objects are stored in Maps, verify that those objects are present in a common ClassLoader.

Java clients or clients outside the cell can connect with the bootstrap IIOP port of the catalog service. In WebSphere Application Server, the deployment manager hosts the catalog service by default. The client can then obtain a ClientClusterContext and the named data grid.

Entity manager

With the entity manager, the tuples are stored in the maps instead of application objects, resulting in fewer class loader problems. Plug-ins can be a problem, however. Also note that a client override ObjectGrid descriptor XML file is always required when connecting to a data grid that has entities defined: ObjectGridManager.connect("host:port[,host:port], null, objectGridOverride) or ObjectGridManager.connect(null, objectGridOverride).

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

Related reference:

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

"Configuring distributed deployments" on page 278
Use the deployment policy descriptor XML file and the ObjectGrid descriptor XML file to manage your topology.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Configuring multiple data center topologies

With the multi-master asynchronous replication, you link a set of catalog service domains. The connected catalog service domains are then synchronized using replication over the links. You can define the links using properties files, at run time with Java Management Extensions (JMX) programs, or with command-line utilities. The set of current links for a domain is stored in the catalog service. You can add and remove links without restarting the catalog service domain that hosts the data grid.

Before you begin

- See “Planning multiple data center topologies” on page 35 for more information about multi-master replication topologies and design considerations. You can configure links among catalog service domains with the server properties file to form the topology during server startup. You can also configure links at run time.
- If you are using loaders in your multi-master replication topology, you must plan how you are going to maintain accurate data between the data centers. The approaches that you can use vary depending on the topology you are using. For more information, see “Loader considerations in a multi-master topology” on page 42.

Procedure

- Define links in the server properties file for the catalog server of each catalog service domain in the topology, for bootstrap purposes.

  See Server properties file for more information about defining this file for the catalog server.

  Important: Property names are case-sensitive.

  Local Domain name:
  Specify the name of the catalog service domain for the current catalog server:

  ```
  domainName=domain1
  ```

  An optional list of foreign domain names:
  Specify the names of catalog service domains to which you want to link in the multi-master replication topology:

  ```
  foreignDomains=domain2,domain3,domain4
  ```

  An optional list of endpoints for the foreign domain names:
  Specifies the connection information for the catalog servers of the foreign domains:

  ```
  domain2.endPoints=hostB1:2809, hostB2:2809
  ```

  If a foreign domain has multiple catalog servers, specify all of them.

- Use the **xscmd** utility or JMX programming to add or remove links at run time.

  The links for a domain are kept in the catalog service in replicated memory. This set of links can be changed at any time by the administrator without requiring a restart of this domain or any other domain. The **xscmd** utility includes several options for working with links.

  The **xscmd** utility connects to a catalog service and thus a single catalog service domain. Therefore, the **xscmd** utility can be used to create and destroy links between the domain it attaches to and any other domain.

  Use the command line to create a link, for example:

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

  The command establishes a new link between the local domain and the foreign domain named `dname`. The `dname` catalog service is running at `fdHostA:2809` and `fdHostB:2809`. The local catalog service domain has a catalog service listener host and port of `host:2809`. Specify all catalog service endpoints from the foreign domain so that fault tolerance connectivity to the domain is possible. Do not use a single `host:port` pair for the catalog service of the foreign catalog service domain.

You can use any local catalog service JVM with **xscmd** and the using the **-cep** option. If the catalog server is hosted in a WebSphere Application Server deployment manager, then the port is usually 9809.

The ports specified for the foreign domain are not JMX ports. They are the usual ports you would use for eXtreme Scale clients.

After the command to add a new link is issued, the catalog service instructs all containers under its management to begin replicating to the foreign domain. A link is not needed on both sides. It is only necessary to create a link on one side.

Use the command line to remove a link, for example:

```
xscmd –c dismissLink -cep host:2809 -fd dname
```

The command connects to the catalog service for a domain and instructs it to stop replicating to a specific domain. A link needs to be dismissed from one side only.

Attention: You can run the establish or dismiss link commands multiple times. If the link does not enter the correct status or is disjoint, run the command again.

Examples

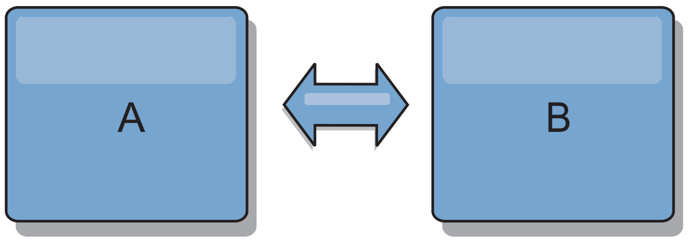


Figure 32. Example: Link between catalog service domains

Suppose that you want to configure a two-domain setup involving catalog service domains A and B.

Here is the server properties file for the catalog server in domain A:

```
domainName=A
foreignDomains=B
B.endPoints=hostB1:2809, hostB2:2809
```

Here is the server properties file for the catalog server in domain B. Notice the similarity between the two property files.

```
domainName=B
foreignDomains=A
A.endPoints=hostA1:2809,hostA2:2809
```

After the two domains are started, then any data grids with the following characteristics are replicated between the domains.

- Has a private catalog service with a unique domain name
- Has the same data grid name as other grids in the domain
- Has the same number of partitions as other data grids in the domain
- Is a FIXED_PARTITION data grid (PER_CONTAINER data grids cannot be replicated)

- Has the same number of partitions (it might or might not have the same number and types of replicas)
- Has the same data types being replicated as other data grids in the domain
- Has the same map set name, map names, and dynamic map templates as other data grids in the domain

The replication policy of a catalog service domain is ignored.

The preceding example shows how to configure each domain to have a link to the other domain, but it is necessary only to define a link in one direction. This fact is especially useful in hub and spoke topologies, allowing a much simpler configuration. The hub property file does not require updates as spokes are added, and each spoke file needs only to include hub information. Similarly, a ring topology requires each domain to have only a link to the previous and next domain in the ring.

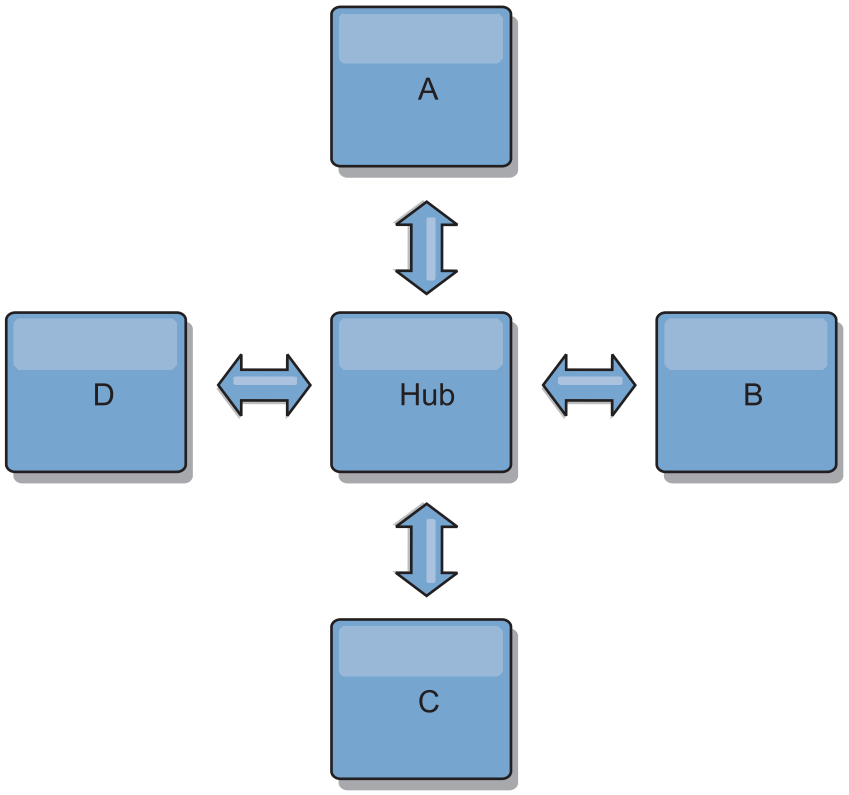


Figure 33. Example: Hub and spoke topology

The hub and four spokes (domains A, B, C, and D) has server property files like the following examples.

```
domainName=Hub
```

Spoke A has the following server properties:

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

Spoke B has the following server properties:

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

Spoke C has the following server properties:

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

Spoke D has the following properties:

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

What to do next

- If you need to check or troubleshoot problems with the links between your catalog service domains, you can use the **xscmd** utility. For more information about commands to help you troubleshoot your multiple data center configuration, see "Troubleshooting multiple data center configurations" on page 632.
- You can provide a custom collision arbiter to resolve collisions between the catalog service domains. See Developing custom arbiters for multi-master replication for more information.

Related concepts:

"Planning multiple data center topologies" on page 35
Using multi-master asynchronous replication, two or more data grids can become exact copies of each other. Each data grid is hosted in an independent catalog service domain, with its own catalog service, container servers, and a unique name. With multi-master asynchronous replication, you can use links to connect a collection of catalog service domains. The catalog service domains are then synchronized using replication over the links. You can construct almost any topology through the definition of links between the catalog service domains.

"Topologies for multi-master replication" on page 36
You have several different options when choosing the topology for your deployment that incorporates multi-master replication. Multi-master replication topologies can be implemented in the DataPower XC10 Appliance by creating multiple collectives and linking them.

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

"Design considerations for multi-master replication" on page 44
When implementing multi-master replication, you must consider aspects in your design such as: arbitration, linking, and performance.

"Loader considerations in a multi-master topology" on page 42
When you are using loaders in a multi-master topology, you must consider the possible collision and revision information maintenance challenges. The data grid maintains revision information about the items in the data grid so that collisions can be detected when other primary shards in the configuration write entries to the data grid. When entries are added from a loader, this revision information is not included and the entry takes on a new revision. Because the revision of the entry seems to be a new insert, a false collision could occur if another primary shard also changes this state or pulls the same information in from a loader.

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use

multi-master replication topologies.

Related information:

Improve response time and data availability with WebSphere eXtreme Scale multi-master capability

Configuring ports

You must open ports to communicate among servers and with remote servers.

Related concepts:

"Planning for network ports" on page 51
WebSphere eXtreme Scale servers require several ports to operate.

"Servers with multiple network cards" on page 338
You can run eXtreme Scale processes on a server that has more than one network card.

Configuring ports in stand-alone mode

You can configure the necessary ports for servers and clients in an eXtreme scale deployment by using command-line parameters, property files or programmatically. Most examples included in the following sections describe command-line parameters to the **`startOgServer`** script. Equivalent configuration options can also be set in properties files, using the embedded server API or the client API.

Procedure

1. Start catalog service endpoints.

   WebSphere eXtreme Scale uses IIOP to communicate between Java virtual machines. The catalog service JVMs are the only processes that require the explicit configuration of ports for the IIOP services and group services ports. Other processes dynamically allocate ports.

   a. Specify client and peer ports. The client port and peer port are used for communication between catalog services in a catalog service domain. To specify the client port and peer port, use the following command-line option:

      `-catalogServiceEndPoints <serverName:hostName:clientPort:peerPort>`
      Specifies a list of catalog servers to link together into a catalog service domain. Each attribute is defined as follows:

      serverName
      Specifies the name of the catalog server.

      hostName
      Specifies the host name for the computer where the server is launched.

      clientPort
      Specifies the port that is used for peer catalog service communication.

      peerPort
      This value is the same as the haManagerPort. Specifies the port that is used for peer catalog service communication.

      The following example starts the cs1 catalog server, which is in the same catalog service domain as the cs2 and cs3 servers:

```
startOgServer.bat|sh cs1 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,cs2:MyServer2.company.com:6601:6602,cs3:MyServer3.company.com:6601:6602
```

If you start additional catalog servers, they must include the same servers in the **-catalogServiceEndPoints** argument. The order of the list can be different, but the servers contained in the list must be the same for each catalog server. Do not put any spaces in the list.

You can also set the catalog service end points with the catalogClusterEndPoints server property.

b. Set the listener host and port. The listener port is used for communication between catalog services in a catalog service domain, and for communication between catalog services and container servers and clients. To specify the listener port and listener host, use the following command-line options:

-listenerHost <host name>

Specifies the host name to which the Object Request Broker (ORB) transport protocol binds for communication. The value must be a fully qualified domain name or IP address. If your configuration involves multiple network cards, set the listener host and port to the IP address for which to bind. By setting the listener and host port, it allows the transport mechanism in the JVM know which IP address to use. If you do not specify which IP address to use, symptoms such as connection timeouts, unusual API failures, and clients that seem to hang can occur.**Default:** `localhost`

-listenerPort <port>
Specifies the port number to which the ORB transport protocol binds for communication. **Default:** `2809`

You can also set the listener port and listener host with the listenerHost and listenerPort server properties.

c. Optional: Set the JMX service port.

The JMX service port is used for communication from JMX clients. To specify the JMX service port, use the following command-line option:

-JMXServicePort <port>
Required only for WebSphere eXtreme Scale in a stand-alone environment. Specifies the port number on which the MBean server listens for communication with Java Management Extensions (JMX).

Default: `1099` for catalog servers

You can also set the JMX service port with the JMXServicePort server property.

d. Optional: Set the JMX connector port.

The JMX connector port is used for communication from JMX clients. To specify the JMX connector port, use the following command-line option:

-JMXConnectorPort <port>
Defines the Secure Sockets Layer (SSL) port to which the Java Management Extensions (JMX) service binds. Only required if an SSL transport protocol is needed for JMX data.

You can also set the JMX connector port with the JMXConnectorPort server property.

e. Set the Secure Socket Layer (SSL) port.

When security is enabled, an SSL port is also required. To specify the SSL port, use the following command-line option:

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

An example of the command follows:

```
./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809
-catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
```

Figure 34. Example using the command line. Start the first catalog server on hostA.

Start the second catalog server on hostB. An example of the command follows:

```
./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809
-catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
```

2. Start container server endpoints.

   The following command starts a container server to use with the example catalog service:

   ```
   ./startOgServer.sh c0 -catalogServiceEndPoints hostA:2809,hostB:2809
   ```

   The container server Java virtual machines use two ports. The HA manager port is used for internal communication between peer container servers and catalog servers. The listener port is used for IIOP communication between peer container servers, catalog servers, and clients. The listener host is used to bind the ORB to a specific network adapter. If you do not specify, both ports are dynamically selected. However, if you want to explicitly configure ports, such as in a firewall environment, you can use a command-line options to specify the ORB port.

   a. Specify the listener host and port. To specify the listener port and listener host, use the following command-line options:

      `-listenerHost <host name>`

      Specifies the host name to which the Object Request Broker (ORB) transport protocol binds for communication. The value must be a fully qualified domain name or IP address. If your configuration involves multiple network cards, set the listener host and port to the IP address for which to bind. By setting the listener and host port, it allows the transport mechanism in the JVM know which IP address to use. If you do not specify which IP address to use, symptoms such as connection timeouts, unusual API failures, and clients that seem to hang can occur.**Default:** `localhost`

      `-listenerPort <port>`
      Specifies the port number to which the ORB transport protocol binds for communication. **Default:** `2809`

      You can also set listener port and listener host with the listenerHost and listenerPort server properties.

   b. Specify the HA manager port. To specify the HA manager port, use the following command-line option:

      `-haManagerPort <port>`
      Specifies the port that is used by the high availability (HA) manager for heartbeat communication between peer container servers. The **`haManagerPort`** port is only used for peer-to-peer communication between container servers that are in same domain. If the haManagerPort property is not defined, then an ephemeral port is used. In WebSphere Application Server, this setting is inherited by the high availability manager port configuration.

You can also set the HA manager port with the HAManagerPort server property.

c. Optional: Specify the SSL port.

   When security is enabled, a Secure Socket Layer (SSL) port is also required. To specify the SSL port, use the following command-line option:

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

d. Optional: Specify the JMX service port.

   -JMXServicePort <port>
   Required only for WebSphere eXtreme Scale in a stand-alone environment. Specifies the port number on which the MBean server listens for communication with Java Management Extensions (JMX).

   Default: `1099` for catalog servers

   You can also set the JMX service port with the JMXServicePort server property.

e. Optional: Set the JMX connector port.

   The JMX connector port is used for communication from JMX clients. To specify the JMX connector port, use the following command-line option:

   -JMXConnectorPort <port>
   Defines the Secure Sockets Layer (SSL) port to which the Java Management Extensions (JMX) service binds. Only required if an SSL transport protocol is needed for JMX data.

   You can also set the JMX connector port with the JMXConnectorPort server property.

3. Start client endpoints.

   Clients must know the catalog service listener end points only. Clients retrieve end points for container server Java virtual machines, which are the Java virtual machines that hold the data, automatically from the catalog service. To connect to the catalog service in the previous example, the client must pass the following list of `host:port` pairs to the connect API:

   ```
   hostA:2809,hostB:2809
   ```

   The client can also receive callbacks from container servers when using the DataGrid API. These callbacks communicate using IIOP with the ORB listener port. To specify the port and network adapter to receive callbacks, set the **`listenerHost`** and **`listenerPort`** properties in the client properties file.

   When security is enabled, a Secure Socket Layer (SSL) port is also required. To specify the SSL port, use the following system property when starting the client process:

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

Related concepts:

"Planning for network ports" on page 51
WebSphere eXtreme Scale servers require several ports to operate.

"Servers with multiple network cards" on page 338
You can run eXtreme Scale processes on a server that has more than one network card.

Configuring ports in a WebSphere Application Server environment

WebSphere eXtreme Scale catalog services, container servers and clients, when running in WebSphere Application Server processes, utilize ports and services already defined for the process.

About this task

The following sections explain details relating to using ports in your deployment.

1. Catalog service endpoints

   WebSphere eXtreme Scale catalog services run in any WebSphere Application Server process and are configured using the administrative console or using administrative tasks. All ports are inherited by the process except for the client port, which is explicitly configured. For details on which ports are used by the catalog service, see "Planning for network ports" on page 51. For details on configuring a catalog service domain, see High availability catalog service.
2. Container server endpoints

   WebSphere eXtreme Scale container servers are hosted within Java EE modules. The container servers use the ports defined for the application server process. For details on which ports are used by the container service, see "Planning for network ports" on page 51. For details on starting a container within a Java EE module such as an Enterprise JavaBeans™ (EJB) or Web module, see "Configuring WebSphere Application Server applications to automatically start container servers" on page 327.
3. Client endpoints

   WebSphere eXtreme Scale clients are hosted within Java EE web or EJB modules.

   Clients programmatically connect to the catalog service domain using the ObjectGridManager.connect() API. When connecting to a catalog service domain hosted within the same cell, the client connection will automatically find the default catalog service domain by using the following API call on the ObjectGridManager:

   ```
   connect(securityProps, overRideObjectGridXML)
   ```

   If the default catalog service domain is hosted remotely (external to the cell), the catalog service endpoints must be specified using the following method on the ObjectGridManager API:

   ```
   connect(catalogServerEndpoints, securityProps, overRideObjectGridXml)
   ```

   If the default catalog service domain is defined in the cell, then the CatalogServerProperties API can be used to retrieve the catalog server addresses. The XSDomainManagement administrative task can also be used to retrieve any configured catalog service domain endpoints.

Related concepts:

"Planning for network ports" on page 51
WebSphere eXtreme Scale servers require several ports to operate.

"Servers with multiple network cards"
You can run eXtreme Scale processes on a server that has more than one network card.

Servers with multiple network cards

You can run eXtreme Scale processes on a server that has more than one network card.

If a server or client is running on a server that contains more than one network card, then you must specify the network port and host name in your eXtreme Scale configuration to bind to a specified card. If this configuration is not specified, then the eXtreme Scale runtime automatically chooses a network post and host name, which may result in connection failures or slower performance.

When you are setting the host name for eXtreme Scale processes that are embedded in WebSphere Application Server, you might need to consider the WebSphere Application Server or other stack products in your configuration. For an example, see Technote: Configuring the node agent on one NIC and its application server on another NIC, which is on a different subnet, can lead to ORB errors .

For catalog or container servers, you must set the listener host and listener port in one of the following ways:

- In the server properties file.
- Command-line parameter on the **startOgServer** script.

For clients, you cannot use the command line, and must use client properties.

Related tasks:

"Configuring ports" on page 334
You must open ports to communicate among servers and with remote servers.

"Configuring ports in stand-alone mode" on page 334
You can configure the necessary ports for servers and clients in an eXtreme scale deployment by using command-line parameters, property files or programmatically. Most examples included in the following sections describe command-line parameters to the **startOgServer** script. Equivalent configuration options can also be set in properties files, using the embedded server API or the client API.

"Configuring ports in a WebSphere Application Server environment" on page 337
WebSphere eXtreme Scale catalog services, container servers and clients, when running in WebSphere Application Server processes, utilize ports and services already defined for the process.

Configuring transports

Transports enable the exchange of objects and data between different server processes in your configuration.

About this task

The main transport mechanism is the Object Request Broker (ORB). This mechanism stores cache entries on the Java heap. Using the ORB as the transport mechanism is required in the following configuration scenarios:

- When you are using a system other than x86 64-bit Linux.
- When you are using container servers that are running in a WebSphere Application Server environment.
- When you are using evictor plug-ins or composite indexes.

7.1.1 If you are using eXtremeMemory, a new transport that is called eXtremeIO is used. With eXtremeMemory, cache entries are stored in native memory. Native memory does not go through garbage collection, leading to more constant performance and predicable response times. Objects are serialized into bytes on the container server. For more information, see "Configuring IBM eXtremeIO (XIO)" on page 340.

Configuring IBM eXtremeIO (XIO)

IBM eXtremeIO (XIO) is a transport mechanism that replaces the Object Request Broker (ORB).

Before you begin

- Linux XIO is supported on x86 64-bit Linux systems that are using a 64-bit SDK only.
- You cannot use XIO in the following configuration scenarios:
  - 7.1.1 When you are using container servers that are running in a WebSphere Application Server environment.
  - 7.1.1 When you are using the ReplicationMapListener interface to create an implementation of an event listener for client-side maps that are in replication mode.
  - When you are using custom evictor plug-ins.
  - When you are using composite indexes.
  - When you are using built-in write-behind loaders.

Procedure

- 7.1.1 Enable the container servers to start using the new transport mechanism. You can use one of the following methods to pick up the new property values:
  - Place a well-named `objectGridServer.properties` file in the class path. For more information, see Server properties file.
  - Set the properties from your application with the ServerProperties interface. For more information, see ServerProperties interface.
  - Start an OSGi server bundle. For more information, see "Starting eXtreme Scale servers using the Eclipse Equinox OSGi framework" on page 493.
  - Restart the container servers. For more information, see 7.1.1 "Starting container servers" on page 463 and "Configuring container servers in WebSphere Application Server" on page 325.
- To prevent the container servers from running out of direct memory, set the maximum direct memory value to 512 MB. Set the **`-XX:MaxDirectMemorySize=512M`** Java virtual machine (JVM) argument on the container servers in the environment.
  - Stand-alone container servers:

    Use the **`-jvmArgs`** parameter when you run the 7.1.1 **`startOgServer`** command.
  - Container servers that are running in WebSphere Application Server:

    Add the JVM argument to the process definition for the container server JVM. For more information, see WebSphere Application Server information center: Configuring the JVM.
  - Container servers that are running the Liberty profile:

    For more information about setting JVM options in Liberty servers, see WebSphere Application Server information center: Customizing the Liberty profile environment.
- Optional: Use command-line arguments and server properties to configure XIO behavior. Update the server properties file for each container server in the configuration to enable XIO properties. After you decide on the properties that

you want to set, you can set the values in the server properties file or programmatically with the ServerProperties interface. For more information about the properties you can set, see “Tuning IBM eXtremeIO (XIO)” on page 570.

What to do next

You can also use IBM eXtremeMemory to help you avoid garbage collection pauses, leading to more constant performance and predicable response times. For more information, see “Configuring IBM eXtremeMemory” on page 346.

Related concepts:

Tuning the copy mode
WebSphere eXtreme Scale makes a copy of the value based on the available CopyMode settings. Determine which setting works best for your deployment requirements.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Configuring Object Request Brokers

The Object Request Broker (ORB) is used by WebSphere eXtreme Scale to communicate over a TCP stack. Use the `orb.properties` file to pass the properties that are used by the ORB to modify the transport behavior of the data grid. No action is required to use the ORB provided by WebSphere eXtreme Scale or WebSphere Application Server for your WebSphere eXtreme Scale servers.

Related reference:

“ORB properties” on page 566
Object Request Broker (ORB) properties modify the transport behavior of the data grid. These properties can be set with an `orb.properties` file, as settings in the WebSphere Application Server administrative console, or as custom properties on the ORB in the WebSphere Application Server administrative console.

Configuring the Object Request Broker in a WebSphere Application Server environment

You can use WebSphere eXtreme Scale with applications that use the Object Request Broker (ORB) directly in WebSphere Application Server or WebSphere Application Server Network Deployment environments.

Procedure

1. Name your application servers appropriately.

   You cannot have servers in a WebSphere Application Server environment with the same name when the servers are using the ORB to communicate with each other. You can resolve this restriction by specifying the system property **`-Dcom.ibm.websphere.orb.uniqueServerName=true`** for the processes that have the same name. For example, when servers with the name `server1` on each node are used as a catalog service domain, or where multiple node agents are used to form a catalog service domain.
2. Tune the ORB properties within the WebSphere Application Server configuration.

See “ORB properties” on page 566 for more information about the properties that you can tune. Depending on the property, you might change a setting in the administrative console or in the *was_root*`properties/orb.properties` file.

3. If you are using multiple network interface cards, you must set the ORB_LISTENER_ADDRESS value in the ports panel in the WebSphere Application Server administrative console. Repeat this step for each application server in the configuration.
   a. For an application server, click **Servers** > **Application servers** > ***server_name***. Under Communications, click **Ports**. The Ports panel is displayed for the specified server.
   b. Click **Details** and edit the ORB_LISTENER_ADDRESS value.
   c. Enter the IP address in the **Host** field. This value must be a private address for a multiple network interface environment.

      Note: DNS host names are not supported for the ORB_LISTENER_ADDRESS value.
   d. Enter the port number in the **Port** field. The port number specifies the port for which the service is configured to accept client requests. The port value is used with the host name.

What to do next

7.1.1+ You can use the **`wxsLogAnalyzer`** tool to verify the ORB settings across your environment. See “Analyzing log and trace data” on page 623 for more information.

Configuring the Object Request Broker with stand-alone WebSphere eXtreme Scale processes

You can use WebSphere eXtreme Scale with applications that use the Object Request Broker (ORB) directly in environments that do not contain WebSphere Application Server or WebSphere Application Server Network Deployment.

Before you begin

Deprecated: The Object Request Broker (ORB) is deprecated. If you were not using the ORB in a previous release, use IBM eXtremeIO (XIO) for your transport mechanism. If you are using the ORB, consider migrating your configuration to use XIO.

If you use the ORB within the same process as eXtreme Scale when you are running applications, or other components and frameworks, that are not included with eXtreme Scale, you might need to complete additional tasks to ensure that eXtreme Scale runs correctly in your environment.

About this task

Add the **`ObjectGridInitializer`** property to the `orb.properties` file to initialize the use of the ORB in your environment. Use the ORB to enable communication between eXtreme Scale processes and other processes that are in your environment.

Procedure

1. The stand-alone installation does not include an `orb.properties` file. You must put an `orb.properties` file is in the `java/jre/lib` directory. For descriptions of the properties and settings, see “ORB properties” on page 566.

2. In the `orb.properties` file, type the following line, and save your changes:

```
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
```

Results

eXtreme Scale correctly initializes the ORB and coexists with other applications for which the ORB is enabled.

To use a custom version of the ORB with eXtreme Scale, see "Configuring a custom Object Request Broker."

What to do next

7.1.1+ You can use the **`xsLogAnalyzer`** tool to verify the ORB settings across your environment. See "Analyzing log and trace data" on page 623 for more information.

Related reference:

"ORB properties" on page 566
Object Request Broker (ORB) properties modify the transport behavior of the data grid. These properties can be set with an `orb.properties` file, as settings in the WebSphere Application Server administrative console, or as custom properties on the ORB in the WebSphere Application Server administrative console.

Configuring a custom Object Request Broker

WebSphere eXtreme Scale uses the Object Request Broker (ORB) to enable communication among processes. No action is required to use the Object Request Broker (ORB) provided by WebSphere eXtreme Scale or WebSphere Application Server for your WebSphere eXtreme Scale servers. Little effort is required to use the same ORBs for your WebSphere eXtreme Scale clients. If instead you must use a custom ORB, the ORB supplied with the IBM SDK is a good choice, although you must configure the ORB. ORBs from other vendors can be used, also with configuration.

Before you begin

Deprecated: The Object Request Broker (ORB) is deprecated. If you were not using the ORB in a previous release, use IBM eXtremeIO (XIO) for your transport mechanism. If you are using the ORB, consider migrating your configuration to use XIO.

Determine if you are using the ORB provided with WebSphere eXtreme Scale or WebSphere Application Server, the ORB provided with the IBM SDK, or an external vendor ORB.

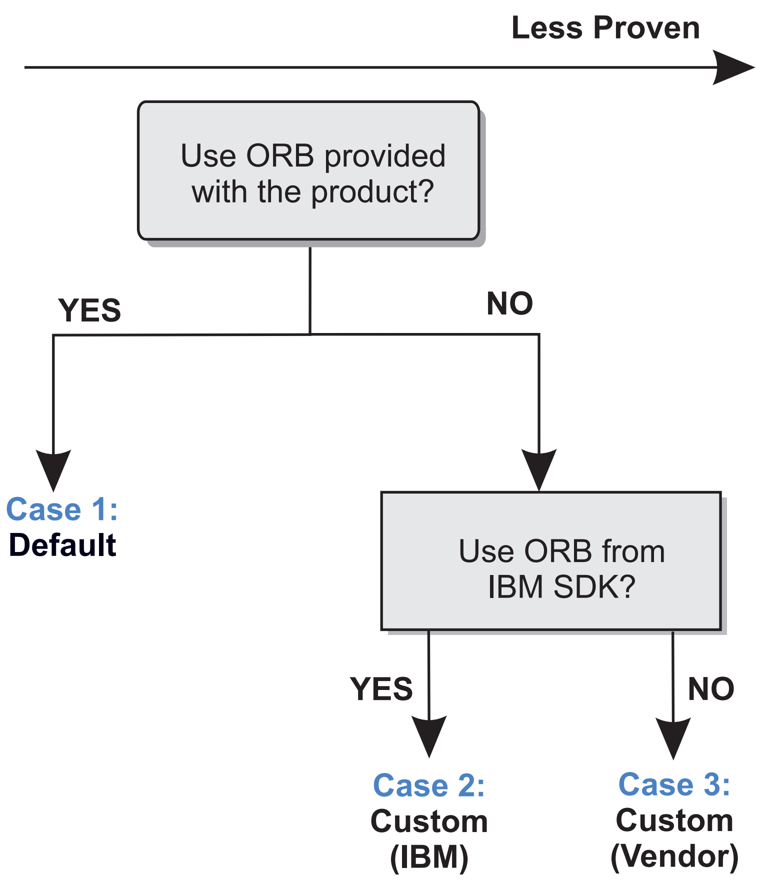


Figure 35. Choosing an ORB

You can make separate decisions for the WebSphere eXtreme Scale server processes and WebSphere eXtreme Scale client processes. While eXtreme Scale supports developer kits from most vendors, it is recommended you use the ORB that is supplied with eXtreme Scale for both your server and client processes. eXtreme Scale does not support the ORB that is supplied with the Oracle Java Development Kit (JDK).

About this task

Become familiar with the configuration that is required to use the ORB of your choice.

Case 1: Default ORB

- For your WebSphere eXtreme Scale server processes, no configuration is required to use the ORB provided with WebSphere eXtreme Scale or WebSphere Application Server.
- For your WebSphere eXtreme Scale client processes, minimal classpath configuration is required to use the ORB provided with WebSphere eXtreme Scale or WebSphere Application Server.

Case 2: Custom ORB (IBM)

To configure your WebSphere eXtreme Scale client processes to use the ORB provided with the IBM SDK, see the instructions later in this topic. You can use the IBM ORB whether you are using the IBM SDK or another development kit. You can use IBM SDK Version 5 or later.

Case 3: Custom ORB (supplied by an external vendor)
Using a vendor ORB for your WebSphere eXtreme Scale client processes is the least tested option. Any problems that you encounter when you use ORBs from independent software vendors must be reproducible with the IBM ORB and compatible JRE before you contact support.

The ORB supplied with the Oracle Java Development Kit (JDK) is not supported.

Procedure

- Configure your client processes to use one of the default ORBs (**Case 1**). Use the following JVM argument :

```
-jvmArgs –Djava.endorsed.dirs=default_ORB_directory${pathSeparator}JRE_HOME/lib/endorsed
```

  The default ORB directory is: *wxs_home*/`lib/endorsed`. Updating the following properties in the `orb.properties` file might also be necessary:

```
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
```

- Configure client or server processes to use IBM SDK Version 5 **(Case 2)**.
  1. Copy the ORB Java archive (JAR) files into an empty directory, or the *custom_ORB_directory*.
     – `ibmorb.jar`
     – `ibmorbapi.jar`
  2. Specify the *`custom_ORB_directory`* directory as an endorsed directory in the scripts that start the Java command.

     Tip: If your Java commands already specify an endorsed directory, another option is to place the *`custom_ORB_directory`* directory under the existing endorsed directory. By placing the *`custom_ORB_directory`* directory under the existing endorsed directory, updating the scripts is not necessary. If you decide to update the scripts anyway, be sure to add the *`custom_ORB_directory`* directory as a prefix to your existing `–Djava.endorsed.dirs=` argument, rather than completely replacing the existing argument.
     – Update scripts for a stand-alone eXtreme Scale environment.
       Edit the path for the *OBJECTGRID_ENDORSED_DIRS* variable in the `setupCmdLine.bat|sh` file to specify the *custom_ORB_directory*. Save your changes.
     – Update scripts when eXtreme Scale is embedded in a WebSphere Application Server environment.
       Add the following system property and parameters to the `startOgServer` script:
       `-jvmArgs –Djava.endorsed.dirs=`*`custom_ORB_directory`*
     – Update custom scripts that you use to start a client application process or a server process.
       `-Djava.endorsed.dirs=`*`custom_ORB_directory`*

Related reference:
"ORB properties" on page 566
Object Request Broker (ORB) properties modify the transport behavior of the data grid. These properties can be set with an `orb.properties` file, as settings in the WebSphere Application Server administrative console, or as custom properties on the ORB in the WebSphere Application Server administrative console.

Configuring IBM eXtremeMemory

By configuring eXtremeMemory, you can store objects in native memory instead of on the Java heap. Configuring eXtremeMemory enables the IBM eXtremeIO transport mechanism.

Before you begin

- To learn more about eXtremeMemory and for more information about determining the maximum amount of memory to use for eXtremeMemory, see "Planning to use IBM eXtremeMemory" on page 55.

About this task

For more information about eXtremeMemory and its benefits versus the Java heap to store your data grid objects, see IBM eXtremeMemory. When you are using eXtremeMemory, eXtremeIO is used for communication between container servers. Objects are serialized into bytes on the container server. To enable eXtremeMemory, you set the required server properties on all of the container servers in the data grid and restart the servers.

eXtremeMemory is not used on catalog servers. If you have a catalog server and a container server that are collocated, the container servers use eXtremeMemory, but the catalog server does not.

Procedure

1. Configure the native libraries by setting the appropriate environment variables. Add the *wxs_install_root*/`ObjectGrid/native` directory to the native library path.

   You can set the environment variable with one of the following ways:

   - Linux Set the *LD_LIBRARY_PATH* environment variable before calling the Java program:

     ```
     LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:wxs_install_root/ObjectGrid/native
     export LD_LIBRARY_PATH
     ```

   - Set the *java.library.path* Java system property to the location where the native libraries are located:

     ```
      java -Djava.library.path=wxs_install_root/ObjectGrid/native <other args>
     ```

2. Update the server properties file for each container server in the configuration to enable eXtremeMemory. To enable eXtremeMemory, you must set the enableXM property. If you do not want the default value of 25% of the entire system to be used for eXtremeMemory, you must also set the maxXMSize property. After you decide on the properties that you want to set, you can set the values in the server properties file or programmatically with the ServerProperties interface.

   For more information about determining the maxXMSize value to set, see "Planning to use IBM eXtremeMemory" on page 55.

   Required properties

   7.1.1+ **enableXM**

   When set to `true`, enables IBM eXtremeMemory on the server and configures the server to use IBM eXtremeIO for synchronous and asynchronous replication. Cache entries for maps that are compatible with eXtremeMemory are stored in native memory instead of on the Java heap. All container servers in the data grid must use the same value for the **`enableXM`** property.

Default:`false`

Suggested properties

7.1.1+ **maxXMSize**
Sets the maximum amount of memory, in megabytes, used by the server for eXtremeMemory storage.

Default: 25% of the total memory on the system.

3. Enable the container servers to start using eXtremeMemory. You can use one of the following methods to pick up the new property values:
   - Place a well-named `objectGridServer.properties` file in the class path. For more information, see Server properties file.
   - Set the properties from your application with the ServerProperties interface. For more information, see ServerProperties interface.
   - Start an OSGi server bundle. For more information, see "Starting eXtreme Scale servers using the Eclipse Equinox OSGi framework" on page 493.
   - Restart the container servers. For more information, see 7.1.1 "Starting container servers" on page 463 and "Configuring container servers in WebSphere Application Server" on page 325.

What to do next

You can also set properties to configure eXtremeIO. For more information, see "Configuring IBM eXtremeIO (XIO)" on page 340.

Related concepts:

IBM eXtremeMemory
IBM eXtremeMemory enables objects to be stored in native memory instead of the Java heap. By moving objects off the Java heap, you can avoid garbage collection pauses, leading to more constant performance and predicable response times.

Configuring Java clients

You can configure WebSphere eXtreme Scale to run in a stand-alone environment, or in an environment with WebSphere Application Server. For a WebSphere eXtreme Scale deployment to pick up configuration changes on the server grid side, you must restart processes to make these changes take effect rather than being applied dynamically. However, on the client side, although you cannot alter the configuration settings for an existing client instance, you can create a new client instance with the settings you require by using an XML file or doing so programmatically. When creating a client, you can override the default settings that come from the current server configuration.

You can configure an eXtreme Scale client (Java client only) in the following ways, each of which can be done with a client override XML file or programmatically:

- XML configuration
- Programmatic configuration
- Spring Framework configuration
- Disabling the near cache

Java client overrides

You can configure a WebSphere eXtreme Scale client based on your requirements by overriding the server settings. You can override several plug-ins and attributes.

To override settings on a client, you can use either XML or programmatic configuration. For more information about overriding client settings, see "Configuring Java clients with an XML configuration" and "Configuring Java clients programmatically" on page 352.

You can override the following plug-ins on a client:

- **BackingMap plug-ins**
  - Evictor plug-in
  - MapEventListener plug-in
  - BackingMapLifecycleListener plug-in
  - MapSerializerPlugin plug-in
- **BackingMap attributes**
  - numberOfBuckets attribute

    Deprecated: This property has been deprecated. Use the nearCacheEnabled attribute to enable the near cache.
  - timeToLive attribute
  - ttlEvictorType attribute
  - evictionTriggers attribute
- **ObjectGrid plug-ins**
  - TransactionCallback plug-in
  - ObjectGridEventListener plug-in
  - ObjectGridLifecycleListener plug-in
- **ObjectGrid attributes**
  - entityMetadataXMLFile attribute
  - txTimeout attribute
  - txIsolation attribute

Related tasks:

"Configuring Java clients with an XML configuration"
You can use an ObjectGrid configuration XML file to override settings on the client side.

Configuring clients in the Spring framework
You can override client-side ObjectGrid settings with the Spring Framework.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Configuring Java clients with an XML configuration

You can use an ObjectGrid configuration XML file to override settings on the client side.

About this task

To change the settings on a WebSphere eXtreme Scale client, create an ObjectGrid XML file that is similar in structure to the file that was used for the container server.

For a list of the plug-ins and attributes that you can override on the client, see "Java client overrides" on page 347.

Procedure

1. Create an ObjectGrid configuration XML file for the client. This file is similar in structure to the file for the container server.

   Assume that the following XML file was paired with a deployment policy XML file, and these files were used to start a container server.

   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>
```

   On a container server, the ObjectGrid instance named CompanyGrid behaves as defined by the `companyGridServerSide.xml` file. By default, the CompanyGrid client has the same settings as the CompanyGrid instance that is running on the server. The following ObjectGrid XML file can be used to specify some of the attributes and plug-ins on the CompanyGrid client:

   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>
```

The XML file defines the following overrides:

- The TransactionCallback bean on the client is com.company.MyClientTxCallback instead of the server-side setting of com.company.MyTxCallback.
- The client does not have an ObjectGridEventListener plug-in because the className value is the empty string.
- The client sets the numberOfBuckets to 1429 for the Customer backingMap, retains its Evictor plug-in, and removes the MapEventListener plug-in.
- The numberOfBuckets and timeToLive attributes of the OrderLine backingMap changed.
- Although a different lockStrategy attribute is specified, there is no effect because the lockStrategy attribute is not supported for a client override.

2. Create the client with the XML file.

   To create the CompanyGrid client with the `companyGridClientSide.xml` file, pass the ObjectGrid XML file as a URL to one of the connect methods on the ObjectGridManager interface:

```
ObjectGridManager ogManager =
 ObjectGridManagerFactory.ObjectGridManager();
ClientClusterContext clientClusterContext =
 ogManager.connect("MyServer1.company.com:2809", null, new URL(
                "file:xml/companyGridClientSide.xml"));
```

Related concepts:

"Java client overrides" on page 347
You can configure a WebSphere eXtreme Scale client based on your requirements by overriding the server settings. You can override several plug-ins and attributes.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Configuring the REST gateway with an XML configuration

You can use a `wxsRestGateway.properties` file to override data grid settings on the client side.

About this task

To override data grid settings on the client side, create an `wxsRestGateway.properties` file in the REST gateway for WebSphere Application

Server and other WebSphere servers to override default behavior.

Procedure

1. Create an ObjectGrid configuration file for the client that is similar in structure to the file for the container server.

   Assume that the following XML file was paired with a deployment policy XML file, and these files were used to start a container server.

   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>
```

   On a container server, the ObjectGrid instance named CompanyGrid behaves as defined by the `companyGridServerSide.xml` file. By default, the CompanyGrid client has the same settings as the CompanyGrid instance that is running on the server.

   The following ObjectGrid XML file can be used to specify some of the attributes and plug-ins on the CompanyGrid client.

   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>
```

The XML file defines the following overrides:

- The TransactionCallback bean on the client is com.company.MyClientTxCallback instead of the server-side setting of com.company.MyTxCallback.
- The client does not have an ObjectGridEventListener plug-in because the className value is the empty string.
- The client sets the numberOfBuckets to 1429 for the Customer backingMap, retains its Evictor plug-in, and removes the MapEventListener plug-in.
- The numberOfBuckets and timeToLive attributes of the OrderLine backingMap changed.
- Although a different lockStrategy attribute is specified, there is no effect because the lockStrategy attribute is not supported for a client override.

2. Create a `wxsRestGateway.properties` file.

   In this properties file, specify the client-side data grid configuration file, `companyGridServerSide.xml`, from step 1. See the following example of a `wxsRestGateway.properties` file:

   ```
   objectGridClientXML=D:\\wxsRestConfig\\companyGridClientSide.xml
   ```

3. In either a stand-alone or WebSphere Application Server configuration, use the system properties, `wxs.restgateway.props` to specify the `wxsRestGateway.properties` file; for example:

   ```
   -Dwxs.restgateway.props=D:\wxsRestConfig\wxsRestGateway.properties
   ```

Configuring Java clients programmatically

You can override client-side settings programmatically. Create an ObjectGridConfiguration object that is similar in structure to the server-side ObjectGrid instance.

About this task

The following code example creates the same overrides that are described in “Configuring Java clients with an XML configuration” on page 348.

For a list of the plug-ins and attributes that you can override on the client, see “Java client overrides” on page 347.

Procedure

The following code creates a client-side ObjectGrid instance.

```
ObjectGridConfiguration companyGridConfig = ObjectGridConfigFactory
    .createObjectGridConfiguration("CompanyGrid");
Plugin txCallbackPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.TRANSACTION_CALLBACK, "com.company.MyClientTxCallback");
companyGridConfig.addPlugin(txCallbackPlugin);

Plugin ogEventListenerPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.OBJECTGRID_EVENT_LISTENER, "");
companyGridConfig.addPlugin(ogEventListenerPlugin);

BackingMapConfiguration customerMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("Customer");
customerMapConfig.setNumberOfBuckets(1429);
```

```
Plugin evictorPlugin = ObjectGridConfigFactory.createPlugin(PluginType.EVICTOR,
    "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor");
customerMapConfig.addPlugin(evictorPlugin);

companyGridConfig.addBackingMapConfiguration(customerMapConfig);

BackingMapConfiguration orderLineMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("OrderLine");
orderLineMapConfig.setNumberOfBuckets(701);
orderLineMapConfig.setTimeToLive(800);
orderLineMapConfig.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);

companyGridConfig.addBackingMapConfiguration(orderLineMapConfig);

ClientClusterContext client = ogManager.connect(catalogServerEndpoints, null, null);
ObjectGrid companyGrid = ogManager.getObjectGrid(client, objectGridName, companyGridConfig);
```

The ogManager instance of the ObjectGridManager interface checks for overrides only in the ObjectGridConfiguration and BackingMapConfiguration objects that you include in the overrideMap Map. For instance, the previous code overrides the number of buckets on the OrderLine Map. However, the Order map remains unchanged on the client side because no configuration for that map is included.

Configuring the near cache

Clients can optionally have a local, in-line cache when eXtreme Scale is used in a distributed topology. This optional cache is called a near cache, an independent data grid on each client, serving as a cache for the remote, server-side cache. The near cache is enabled by default when locking is disabled, or is configured as optimistic, and cannot be used when configured as pessimistic.

About this task

A near cache is fast because it provides local in-memory access to a subset of the entire cached data set that is stored remotely.

To configure the near cache, you can edit the necessary settings in the ObjectGrid XML file for your container server. The settings in this file apply to all clients, unless you override the settings. You can override the **nearCacheEnabled** setting for the near cache with either XML or programmatic configuration.

Procedure

1. Near cache is enabled if you are using default settings. To enable the near cache, you must set the **lockStrategy** attribute in the ObjectGrid descriptor XML file for the container servers to `NONE` or `OPTIMISTIC`. The default value is `OPTIMISTIC`. For more information about the **lockStrategy** attribute, see ObjectGrid descriptor XML file. Clients do not maintain a near cache when the locking setting is configured as `PESSIMISTIC`.
2. 7.1.1 To enable or disable the near cache, set the **numberOfBuckets** attribute in the ObjectGrid descriptor XML file.

   The near cache is enabled by default when the **lockStrategy** attribute in the ObjectGrid descriptor XML file is set to `NONE` or `OPTIMISTIC`. Clients do not maintain a near cache when the locking setting is configured as `PESSIMISTIC`. To disable the near cache, you must set the **numberOfBuckets** attribute to `0` in the client override ObjectGrid descriptor file.

   numberOfBuckets

Deprecated: This property has been deprecated. Use the **`nearCacheEnabled`** attribute to enable the near cache.
The BackingMap instance uses a hash map for implementation. The **`numberOfBuckets`** attribute specifies the number of buckets for the BackingMap instance to use. If multiple entries exist in the BackingMap, more buckets lead to better performance because the risk of collisions is lower as the number of buckets increases. More buckets also lead to more concurrency. Specify a value of `0` to disable the near cache on a client. When you set the value to `0` for a client, set the value in the client override ObjectGrid XML descriptor file only. (Optional)

3. Restart the container servers and clients. For more information, see "Starting and stopping stand-alone servers" on page 459 and "Configuring container servers in WebSphere Application Server" on page 325.

Results

To check whether a near cache is enabled, run the BackingMap.isNearCacheEnabled() method in your client. You can also look for the CWOBJ1128I message in the log files on the client to see if the near cache is enabled.

What to do next

By default, the client-side near cache does not have a maximum size and out of memory errors in the client can occur. To control the size of the near cache, configure a client-side override that enables a time-to live (TTL) or least recently used (LRU) evictor on the client. For more information about configuring an evictor for the near cache, see "Configuring an evictor for the near cache."

Related concepts:

"Configuring Java Message Service (JMS)-based client synchronization" on page 355
You can use JMS-based client synchronization to keep data from the client near cache synchronized with other severs and clients.

"Distributed cache" on page 19
WebSphere eXtreme Scale is most often used as a shared cache, to provide transactional access to data to multiple components where a traditional database would otherwise be used. The shared cache eliminates the need configure a database.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Configuring an evictor for the near cache

To control the size of the near cache, configure a client-side override that enables an evictor on the client.

About this task

You can use an XML or programmatic override to configure the evictor. For more information, see "Configuring Java clients with an XML configuration" on page 348 and "Configuring Java clients programmatically" on page 352.

Procedure

- XML file configuration:

  Use the following ObjectGrid XML file as an example to create a client-side configuration file to configure an evictor for the near cache.

```
<?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" nearCacheEnabled="true"
        pluginCollectionRef="customerPlugins" />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="customerPlugins">
      <!-- Limit the near cache size to 53*1000=53,000 entries using a least recently used algorithm -->
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
       <property name="numberOfLRUQueues" type="int" value="53" description="set number of LRU queues" />
       <property name="maxSize" type="int" value="1000" description="set max size for each LRU queue" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

- Programmatic configuration:

  Use the following code snippet to programmatically configure an LRU evictor for the near cache:

```
ObjectGridConfiguration companyGridConfig = ObjectGridConfigFactory
  .createObjectGridConfiguration("CompanyGrid");
BackingMapConfiguration customerMapConfig = ObjectGridConfigFactory
  .createBackingMapConfiguration("Customer");
Plugin evictorPlugin = ObjectGridConfigFactory.createPlugin(PluginType.EVICTOR,
  "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor");
ConfigProperty numQueues=ObjectGridConfigFactory.createConfigProperty(ConfigPropertyType.INT_PRIM,
  "numberOfLRUQueues", "53");
evictorPlugin.addConfigProperty(numQueues);
ConfigProperty maxSize=ObjectGridConfigFactory.createConfigProperty(ConfigPropertyType.INT_PRIM,
  "maxSize", "1000");
evictorPlugin.addConfigProperty(maxSize);
customerMapConfig.addPlugin(evictorPlugin);
companyGridConfig.addBackingMapConfiguration(customerMapConfig);
ClientClusterContext client = ogManager.connect(catalogServerEndpoints, null, null);
ObjectGrid companyGrid = ogManager.getObjectGrid(client, objectGridName,companyGridConfig);
```

Configuring Java Message Service (JMS)-based client synchronization

You can use JMS-based client synchronization to keep data from the client near cache synchronized with other severs and clients.

Near cache

You can use the built-in Java Message Service (JMS)-based com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener class to enable the client invalidation mechanism within a distributed eXtreme Scale environment.

The client invalidation mechanism is the solution for the issue of stale data in client near cache in distributed eXtreme Scale environment. This mechanism ensures that the client near cache is synchronized with servers or other clients. However, even with this JMS-based client invalidation mechanism, the client near cache does not immediately update. A delay occurs when the run time publishes updates.

Two models are available for the client invalidation mechanism in a distributed eXtreme Scale environment:

- Client-server model: In this model, all server processes are in a publisher role that publishes all the transaction changes to the designated JMS destination. All client processes are in receiver roles and receive all transactional changes from the designated JMS destination.
- Client as dual roles model: In this model, all server processes have nothing to do with the JMS destination. All client processes are both JMS publisher and

receiver roles. Transactional changes that occur on the client are published to the JMS destination and all the clients receive these transactional changes.

For more information, see “JMS event listener” on page 271.

Client-server model

In a client-server model, the servers are in a JMS publisher role and the client is in JMS receiver role.

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>
```

Client as dual roles model

In client as dual roles model, each client has both JMS publisher and receiver roles. The client publishes every committed transactional change to a designated JMS destination and receives all the committed transactional changes from other clients. The server has nothing to do with JMS in this model.

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>
```

Related tasks:

"Configuring the near cache" on page 353
Clients can optionally have a local, in-line cache when eXtreme Scale is used in a distributed topology. This optional cache is called a near cache, an independent data grid on each client, serving as a cache for the remote, server-side cache. The near cache is enabled by default when locking is disabled, or is configured as optimistic, and cannot be used when configured as pessimistic.

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Configuring request and retry timeout values

About this task

You can configure settings for the eXtreme Scale client that control how long the client attempts to create network connections, how long the client attempts to process a data grid request to a partition, and how long it attempts to retry that request to the partition, before it returns an exception to your application.

Factors for tuning request and retry timeout values in IBM eXtremeIO (XIO) and Object Request Broker (ORB)

For some tuning options, where you set the values depends on which transport you are using, either XIO or ORB. These transport-level tuning options have the initial impact on interactions with your client because they govern how long the transport attempts network socket connections and how long an individual remote procedure call (RPC) analogous to a data grid operation is given to complete.

When you tune these values, consider what your environment can tolerate under peak load conditions as well as steady state conditions. If you tune the intervals too far under the default values (30 seconds for request timeout, for example), your operations might fail prematurely. Consider the following factors:
- Network latencies
- Coupling of grid interactions with external resources like databases
- Garbage collection pauses resulting from your combination of heap size, heap usage, and garbage collection tuning policies

ORB settings for tuning request and retry timeout values

The following timeout settings exist for the ORB:

com.ibm.CORBA.ConnectionTimeout
Specifies the amount of time that the ORB attempts to create a socket connection with the remote location before the attempts time out. The ORB caches these connections, and therefore, this operation is not done on every request.

com.ibm.CORBA.RequestTimeout
Specifies the amount of time that the ORB waits for an RPC to complete before timing out.

com.ibm.CORBA.FragmentTimeout
Reference the IBM ORB documentation for precise details. The product provides default settings for this value.

com.ibm.CORBA.LocateRequestTimeout
Reference the IBM ORB documentation for precise details. The product provides default settings for this value.

com.ibm.CORBA.SocketWriteTimeout
Specifies how many seconds a socket write waits before giving up.

As you tune the RequestTimeout and ConnectionTimeout settings, adjusting them based on the default recommendations can be appropriate. You can also set these settings with the same value, where you define these settings that are based on how long you want the request timeout to be.

XIO settings for tuning request and retry timeout values

With XIO, the following consolidated settings exist:
- The `xioTimeout` setting determines how long the XIO transport attempts to establish a network socket connection.

- There is no equivalent to the **`LocateRequest`** setting and the **`FragmentTimeout`** setting in the ORB.
- The **`xioRequestTimeout`** value specifies how many seconds any request waits for a response before giving up. This property influences the amount of time a client takes to fail over if a network outage failure occurs. If you set this property too low, requests might time out inadvertently. Carefully consider the value of this property to prevent inadvertent timeouts.

Common settings for tuning request and retry timeout values

The next level of tuning is the requestRetryTimeout. With each transport type, after it throws a system exception because an RPC did not complete in time, the eXtreme Scale client can use the additional time that is defined by the requestRetryTimeout setting (for example, the request timeout is 10 seconds, and the retry request timeout is 20 seconds) to specify how long it takes to complete the following actions:
- Asynchronously asks the catalog server for the latest routing table in case partitions are located elsewhere because of a failover.
- Takes new routes and retries the request, or stops trying and throws an exception to your application.

The requestRetryTimeout property is set in milliseconds. Set the value greater than zero for the request to be retried on exceptions for which retry is available. Set the value to `0` to fail without retries on exceptions. To use the default behavior, remove the property or set the value to `-1`.

Ways to set request retry timeout

You can configure the request retry timeout value on the client properties file or in a session. The session value overrides the client properties setting. If the value is set to greater than zero, the request is tried until either the timeout condition is met or a permanent failure occurs. A permanent failure might be a `DuplicateKeyException` exception. A value of zero indicates the fail-fast mode setting and the data grid does not attempt to try the transaction again after any type of transaction.

Transaction timeout and request retry timeout

During run time, the transaction timeout value is used with the request retry timeout value, ensuring that the request retry timeout does not exceed the transaction timeout.

Two types of transactions exist: Autocommit transactions, and transactions that use explicit begin and commit methods. The valid exceptions for retry differ between these two types of transactions:
- For transactions that are called within a session, transactions are tried again for ORB CORBA SystemException (TransportException for XIO) and eXtreme Scale client TargetNotAvailable exceptions.
- Autocommit transactions are tried again for CORBA SystemException and eXtreme Scale client availability exceptions. These exceptions include the ReplicationVotedToRollbackTransactionException, TargetNotAvailable, and AvailabilityException exceptions.

Application or other permanent failures return immediately and the client does not try the transaction again. These permanent failures include the

DuplicateKeyException and KeyNotFoundException exceptions. Use the fail-fast setting to return all exceptions without trying transactions again after any exceptions.

Exceptions where the client tries the transaction again:
- ReplicationVotedToRollbackTransactionException (only on autocommit)
- TargetNotAvailable
- org.omg.CORBA.SystemException (TransportException is the XIO equivalent of this ORB system exception.)
- AvailabilityException (only on autocommit)
- LockTimeoutException (only on autocommit)
- UnavailableServiceException (only on autocommit)

Permanent exceptions, where the transaction is not tried again:
- DuplicateKeyException
- KeyNotFoundException
- LoaderException
- TransactionAffinityException
- LockDeadlockException
- OptimisticCollisionException

Procedure

- Set the request retry timeout value in a client property file.

  To set the requestRetryTimeout value on a client, add or modify the requestRetryTimeout property in the Client properties file. The client properties is the `objectGridClient.properties` file by default. The requestRetryTimeout property is set in milliseconds. Set the value greater than zero for the request to be retried on exceptions for which retry is available. Set the value to `0` to fail without retries on exceptions. To use the default behavior, remove the property or set the value to `-1`. An example of the value in the `objectGridClient.properties` file follows:

  ```
  requestRetryTimeout = 30000
  ```

  The requestRetryTimeout value is specified in milliseconds. In the example, if the value is used on an ObjectGrid instance, the requestRetryTimeout value is 30 seconds.
- Set the request retry timeout value programmatically.

  To set the client properties programmatically, first create a client properties file in an appropriate `<location>` for your application. In the following example, the client properties file refers to the `objectGridClient.properties` snippet in the previous section. After you connect to ObjectGridManager instance, set the client properties as described. Then, when you have an ObjectGrid instance, the instance has the client properties that you defined in the file. If you change the client properties file, you must explicitly get a new ObjectGrid instance each time.

  ```
  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);
  ```
- Set the override file during a session commit.

To set the request retry timeout on a session or to override the requestRetryTimeout client property, call the setRequestRetryTimeout(long) method on the Session interface.

```
Session sessionA = objectGrid.getSession();
sessionA.setRequestRetryTimeout(30000);
ObjectMap mapA = sessionA.getMap("payroll");
String key = "key:" + j;
mapA.insert(key, "valueA");
```

This session now uses a requestRetryTimeout value of 30000 ms or 30 seconds, regardless of the value that is set in the client properties file. For more information about the session interface, see Using Sessions to access data in the grid.

Example

Consider the following example, where the client can handle network latency, garbage collection, general contention on the server as a result of setting short timeout values. The **`requestRetryTimeout`** property is `10` seconds, and the **`xioTimeout`** property matches the ORB **`ConnectionTimeout`** value, which is `5` seconds.

Table 23. Data grid configurations for ORB and eXtremeIO transport types

Grid Type	ORB	XIO
A Java or .NET client application that accesses an eXtreme Scale API directly	• Modify the `orb.properties` file for your client application. Set the following values: – com.ibm.CORBA.RequestTimeout=5 – com.ibm.CORBA.ConnectTimeout=5 – com.ibm.CORBA.FragmentTimeout=5 – com.ibm.CORBA.LocateRequestTimeout=5 – com.ibm.CORBA.SocketWriteTimeout=5 **Note:** With WebSphere Application Server, you control the ORB settings through the deployment manager, and not through an `orb.properties` file. • Modify the `objectGridClient.properties` file for the client application with requestRetryTimeout=10000.	Modify the `objectGridClient.properties` file for your client application with the following values: • xioRequestTimeout=5000. This value is in milliseconds and is equivalent to com.ibm.CORBA.RequestTimeout. • xioTimeout=5. This value is in seconds and is equivalent to com.ibm.CORBA.ConnectTimeout. • requestRetryTimeout=10000. This value is in milliseconds and is also used for the ORB transport. • ORB FragmentTimeout and LocateRequestTimeout have no XIO equivalent values.
HTTP session	Same ORB configuration as a Java or .NET client application that accesses an eXtreme Scale API directly	Same XIO configuration as a Java or .NET client application that accesses an eXtreme Scale API directly
Dynamic cache	Same ORB configuration as a Java or .NET client application that accesses an eXtreme Scale API directly. For dynamic cache instances, you can set the following additional property on the cache instance: • `com.ibm.websphere.xs.dynacache.request_retry_timeout_override=10000` You can use the **`requestRetryTimeout`** setting with the client properties in the class path instead of this cache instance property, if you want it to be the same for every dynamic cache instance.	Same XIO configuration as a Java or .NET client application that accesses an eXtreme Scale API directly. For dynamic cache instances, you can set the following additional property on the cache instance: • `com.ibm.websphere.xs.dynacache.request_retry_timeout_override=10000` You can use the **`requestRetryTimeout`** setting with the client properties in the class path instead of this cache instance property, if you want it to be the same for every dynamic cache instance.

Related concepts:

Using Sessions to access data in the grid
Your applications can begin and end transactions through the Session interface. The Session interface also provides access to the application-based ObjectMap and JavaMap interfaces.

Configuring eXtreme Scale connection factories

Before you begin

Before you create the connection factories, you must install the resource adapter.

About this task

After you install the resource adapter, you can create one or more resource adapter connection factories that represent eXtreme Scale client connections to remote data grids. Complete the following steps to configure a resource adapter connection factory and use it within an application.

You can create an eXtreme Scale connection factory at the node scope for stand-alone resource adapters or within the application for embedded resource adapters. See the related topics for information about how to create connection factories in WebSphere Application Server.

Procedure

1. Using the WebSphere Application Server administrative console to create an eXtreme Scale connection factory that represents an eXtreme Scale client connection. See Configuring Java EE Connector connection factories in the administrative console. After you specify properties for the connection factory in the General Properties panel, you must click **Apply** for the Custom properties link to become active.
2. Click **Custom properties** in the administrative console. Set the following custom properties to configure the client connection to the remote data grid.

Table 24. Custom properties for configuring connection factories

Property Name	Type	Description
ConnectionName	String	(Optional) The name of the eXtreme Scale client connection. The ConnectionName helps identify the connection when exposed as a managed bean. This property is optional. If not specified, the ConnectionName is undefined.
CatalogServiceEndpoints	String	(Optional) The catalog service domain end points in the format: `<host>:<port>[,<host><port>]`. For more information, see "Catalog service domain settings" on page 315. This property is required if the catalog service domain is not set.
CatalogServiceDomain	String	(Optional) The catalog service domain name that is defined in WebSphere Application Server. For more information, see "Configuring catalog servers and catalog service domains" on page 294. This property is required if the CatalogServiceEndpoints property is not set.
ObjectGridName	String	(Optional) The name of the data grid that this connection factory connects to. If not specified, then the application must supply the name when obtaining the connection from the connection factory.
ObjectGridURL	String	(Optional) The URL of the client data grid, override XML file. This property is not valid if the ObjectGridResource is also specified. For more information, see "Configuring Java clients" on page 347.
ObjectGridResource	String	The resource path of the client data grid, override XML file. This property is optional and invalid if ObjectGridURL is also specified. For more information, see "Configuring Java clients" on page 347.

Table 24. Custom properties for configuring connection factories (continued)

Property Name	Type	Description
ClientPropertiesURL	String	(Optional) The URL of the client properties file. This property is not valid if the ClientPropertiesResource is also specified. For more information, see Client properties file for more information.
ClientPropertiesResource	String	(Optional) The resource path of the client properties file. This property is not valid if the ClientPropertiesURL is also specified. For more information, see Client properties file for more information.

WebSphere Application Server also allows other configuration options for adjusting connection pools and managing security. See the related information for links to WebSphere Application Server Information Center topics.

What to do next

Create an eXtreme Scale connection factory reference in the application. See Configuring applications to connect with eXtreme Scale for more information.

Configuring Eclipse environments to use eXtreme Scale connection factories

Before you begin

- You must install Rational® Application Developer Version 7 or later or Eclipse Java EE IDE for Web Developers Version 1.4 or later.
- A server runtime environment must be configured.

Procedure

1. Import the `wxsra.rar` file into your project by selecting **File** > **Import**. The Import window is displayed.
2. Select **Java EE** > **RAR file**. The Connector Import window is displayed.
3. To specify the connector file, click **Browse** to locate the `wxsra.rar` file. The `wxsra.rar` file is installed when you install a resource adapter. You can find the resource adapter archive (RAR) file in the following location:
   - For WebSphere Application Server installations: *`wxs_install_root`*`/optionalLibraries/ObjectGrid`
   - For stand-alone installations: *`wxs_install_root`*`/ObjectGrid/lib directory`
4. Create a name for the new connector project in the **Connector project** field. You can use `wxsra`, which is the default name.
5. Choose a Target runtime, which references a Java EE server runtime environment.
6. Optionally select **Add project to EAR** to embed the RAR into an existing EAR project.

Results

The RAR file is now imported into your Eclipse workspace.

What to do next

You can reference the RAR project from your other Java EE projects using the following steps:

1. Right click on the project and click **Properties**.
2. Select **Java Build Path**.
3. Select the Projects tab.

4. Click **Add**.
5. Select the **wxsra** connector project, and click **OK**.
6. Click **OK** again to close the Properties window.

The eXtreme Scale resource adapter classes are now in the classpath. To install product runtime JAR files using the Eclipse console, see Setting up a stand-alone development environment in Eclipse for more information.

Configuring cache integration

WebSphere eXtreme Scale can integrate with other caching-related products. You can also use the WebSphere eXtreme Scale dynamic cache provider to plug WebSphere eXtreme Scale into the dynamic cache component in WebSphere Application Server. Another extension to WebSphere Application Server is the WebSphere eXtreme Scale HTTP session manager, which can help to cache HTTP sessions.

Related tasks:

"Troubleshooting loaders" on page 633
Use this information to troubleshoot issues with your database loaders.

"Configuring JPA loaders" on page 419
A Java Persistence API (JPA) Loader is a plug-in implementation that uses JPA to interact with the database.

Configuring HTTP session managers

The HTTP session manager provides session replication capabilities for an associated application. The session manager works with the Web container to create and manage the life cycles of HTTP sessions that are associated with the application.

Related concepts:

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related reference:

"XML files for HTTP session manager configuration" on page 382
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Configuring the HTTP session manager with WebSphere Application Server

While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

Before you begin

- WebSphere eXtreme Scale must be installed on your WebSphere Application Server or WebSphere Application Server Network Deployment cell to use the eXtreme Scale session manager. For more information, see 7.1.1 “Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server” on page 186.
- When WebSphere eXtreme Scale for HTTP session replication is used on WebSphere Application Server, the **Allow overflow session management** setting must be checked for every applicable web application and application server that hosts that web application. For more information, see Session management settings.
- Global security must be enabled in the WebSphere Application Server administrative console, if the catalog servers within your catalog service domain have Secure Sockets Layer (SSL) enabled. It must also be enabled if you want to use SSL for a catalog service domain with SSL supported. You require SSL for a catalog server by setting the transportType attribute to `SSL-Required` in the Server properties file. For more information about configuring global security, see Global security settings.

About this task

The WebSphere eXtreme Scale HTTP session manager supports both embedded and remote servers for caching.

- **Embedded scenario**

  In the embedded scenario, the data grid servers are collocated in the same processes where the servlets run. The session manager can communicate directly with the local ObjectGrid instance, avoiding costly network delays.

  If you are using WebSphere Application Server, place the supplied *wxs_home*`/session/samples/objectGrid.xml` and *wxs_home*`/session/samples/objectGridDeployment.xml` files into the `META-INF` directories of your web archive (WAR) files. eXtreme Scale automatically detects these files when the application starts and automatically starts the eXtreme Scale containers in the same process as the session manager.

  You can modify the `objectGridDeployment.xml` file. Modifying this file depends on whether you want to use synchronous or asynchronous replication and how many replicas you want configured.
- **Remote servers scenario**

  In the remote servers scenario, the container servers that are run are in different processes than the servlets. The session manager communicates with a remote container server. To use a remote, network-attached container server, the session manager must be configured with the host names and port numbers of the catalog service domain. The session manager then uses an eXtreme Scale client connection to communicate with the catalog server and the container servers.

If the container servers are starting in independent, stand-alone processes, start the data grid containers with the `objectGridStandAlone.xml` and `objectGridDeploymentStandAlone.xml` files that are supplied in the session manager samples directory.

Procedure

1. Splice your application so that it can use the session manager. To use the session manager, you must add the appropriate filter declarations to the web deployment descriptors for the application. In addition, session manager configuration parameters are passed in to the session manager in the form of servlet context initialization parameters in the deployment descriptors. There are multiple ways in which you can introduce this information into your application:
   - **Auto-splice with WebSphere Application Server**

     You can configure your application to use the HTTP session manager for the data grid when you install your application. You can also edit the application or server configuration to use the WebSphere eXtreme Scale HTTP session manager. For more information, see "Configuring WebSphere Application Server HTTP session persistence to a data grid" on page 368.
   - **Splice the application with the `addObjectGridFilter` script**

     For more information about running this script, see "Splicing a session data grid application with the **`addObjectGridFilter`** script" on page 374.
   - **Auto-splice the application with custom properties**

     You do not need to manually splice your applications when the application is running in WebSphere Application Server or WebSphere Application Server Network Deployment.

     You can use this auto-splice option when your environment meets the following conditions:
     - You are using a deployment manager. The cell, server, and application scope are available scopes and are only available when you are running in a deployment manager. If you require a different scope, manually splice your web applications.
     - The `splicer.properties` file must be in at the same path on all nodes. The nodes are hosting an application server or applications that are being spliced for session replication. For mixed environments containing Windows and UNIX nodes, this option is not possible, so you must manually splice the application.

     Add the `com.ibm.websphere.xs.sessionFilterProps` custom property to either a cell or a server to set the `splicer.properties` file location for all of the web applications at that scope. The file exists on the deployment manager. If you want to indicate the `splicer.properties` file for a specific application with a cell-level custom property, enter the name of the custom property as: `<`*`application_name`*`>,com.ibm.websphere.xs.sessionFilterProps`, where *application_name* indicates the name of the application for which you want to apply the custom property. The value is the location of the `splicer.properties` file your applications require. An example path for the location of a file follows: `/opt/splicer.properties`.
   - **Manually splice the application with the Ant build script**

     WebSphere eXtreme Scale ships with a `build.xml` file that can be used by Apache Ant, which is included in the *was_root*`/bin` folder of a WebSphere Application Server installation. You can modify the `build.xml` file to change the session manager configuration properties. The configuration properties are identical to the property names in the `splicer.properties` file. You modify the `build.xml` file, start the Ant process by running the following command:

- UNIX `ant.sh`, `ws_ant.sh`
- Windows `ant.bat`, `ws_ant.bat`

(UNIX) or (Windows).

- **Manually update the web descriptor**

  Edit the `web.xml` file that is packaged with the web application to incorporate the filter declaration, its servlet mapping, and servlet context initialization parameters. Do not use this method because it is prone to errors.

For a list of the parameters that you can use, see “Servlet context initialization parameters” on page 388.

2. Deploy the application. Deploy the application with your normal set of steps for a server or cluster. After you deploy the application, you can start the application.
3. Access the application. You can now access the application, which interacts with the session manager and WebSphere eXtreme Scale.

What to do next

You can change most of the configuration attributes for the session manager when you instrument your application to use the session manager. These attributes include: synchronous or asynchronous replication, in-memory session table size, and so on. Apart from the attributes that can be changed at application instrumentation time, the only other configuration attributes that you can change after the application deployment are the attributes that are related to the WebSphere eXtreme Scale server cluster topology and the way that their clients (session managers) connect to them.

Remote scenario behavior: If the entire data grid that is hosting the application session data is unreachable from the web container client, the client instead uses the base web container in WebSphere Application Server for session management. The data grid might be unreachable in the following scenarios:

- A network problem between the web container and the remote container servers.
- The remote container server processes have been stopped.

The number of session references kept in memory, which is specified by **`sessionTableSize`** parameter, is still maintained when the sessions are stored in the base web container. The least recently used sessions are invalidated from the web container session cache when the **`sessionTableSize`** value is exceeded. If the remote data grid becomes available, sessions that were invalidated from the web container cache can retrieve data from the remote data grid and load the data into a new session. If the entire remote data grid is not available and the session is invalidated from the session cache, the user session data is lost. Because of this issue, do not shut down the entire production remote data grid when the system is running under load.

Related concepts:

“Interoperability with other products” on page 49
You can integrate WebSphere eXtreme Scale with other products, such as WebSphere Application Server and WebSphere Application Server Community Edition.

"Monitoring with vendor tools" on page 550
WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli Monitoring and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

"Tuning garbage collection with WebSphere Real Time" on page 577
Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary. WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related reference:

"XML files for HTTP session manager configuration" on page 382
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Configuring WebSphere Application Server HTTP session persistence to a data grid:

You can configure your WebSphere Application Server application to persist sessions to a data grid. This data grid can be in an embedded container server that runs within WebSphere Application Server, or it can be in a remote data grid.

Before you begin

Before you change the configuration in WebSphere Application Server, you must have:

- The name of the session data grid that you want to use. See "Configuring the HTTP session manager with WebSphere Application Server" on page 365 for information about creating a session data grid.
- If the catalog service to manage your sessions is outside of the cell in which you are installing your session application, you must create a catalog service domain. For more information, see "Creating catalog service domains in WebSphere Application Server" on page 299.
- If you are configuring a catalog service domain, you might must enable client security on the catalog service domain if the container servers require authentication. These settings inform the run time which CredentialGenerator implementation to use. This implementation generates a credential to pass to the remote data grid. For more information, see "Configuring client security on a catalog service domain" on page 604.
- Enable global security in the WebSphere Application Server administrative console, if you support one of these scenarios:
  - The catalog servers within your catalog service domain have Secure Sockets Layer (SSL) enabled.
  - You want to use SSL for a catalog service domain with SSL supported.

  You require SSL for a catalog server by setting the **`transportType`** attribute to `SSL-Required` in the Server properties file. For more information about configuring global security, see Global security settings.
- If you are using Version 7.1.0.3 or later, you can persist sessions that use URL rewriting or cookies as a session tracking mechanism to the data grid. For releases before Version 7.1.0.3, you cannot persist sessions that use URL rewriting as a session tracking mechanism. To enable the persistence of sessions that use URL rewriting, set the **`useURLEncoding`** property to `true` in the `splicer.properties` file after you automatically splice the application.
- 7.1.1+ When you are automatically splicing applications for HTTP session management in WebSphere Application Server , all of the application servers that host the web application have the **`HttpSessionIdReuse`** web container custom property that is set to `true`. This property enables sessions that fail over from one application server to another, or are invalidated from the in-memory session cache in a remote scenario to preserve its session ID across requests. If you do not want this behavior, set the web container custom property to `false` on all of the applicable application servers before you configure session management for the applications. For more information about this custom property, see "Troubleshooting cache integration" on page 628.

Procedure

- **To configure session management when you are installing the application, complete the following steps:**
  1. In the WebSphere Application Server administrative console, click **Applications** > **New application** > **New Enterprise Application**. Choose the **Detailed** path for creating the application and complete the initial wizard steps.
  2. In the **eXtreme Scale session management settings** step of the wizard, configure the data grid that you want to use. Choose either the **Remote eXtreme Scale data grid** or the **Embedded eXtreme Scale data grid**.

- For the **Remote eXtreme Scale data grid** option, choose the catalog service domain that manages the session data grid, and choose a data grid from the list of active session data grids.
- For the **Embedded eXtreme Scale data grid** option, choose either the default ObjectGrid configuration or specify the specific location of your ObjectGrid configuration files.

3. Complete the wizard steps to finish installing your application.

You can also install the application with a wsadmin script. In the following example, the **-SessionManagement** parameter creates the same configuration that you can in the administrative console:

For the remote eXtreme Scale data grid configuration:

```
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]]]')
```

For the eXtreme Scale embedded scenario with default configuration:

```
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]]]')
```

For the eXtreme Scale embedded scenario with a custom configuration:

```
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]]]')
```

- **To configure session management on an existing application in the WebSphere Application Server administrative console:**

  Note: The **Override session management** box is checked when the application is set to use WebSphere eXtreme Scale. This means any server level session settings that were made to WebSphere Application Server configuration are overwritten by the application-level session settings. If you do not want to override settings, you can enable WebSphere eXtreme Scale at the server level.

  1. In the WebSphere Application Server administrative console, click **Applications** > **Application Types** > **WebSphere enterprise applications** >

application_name > **Web Module properties** > **Session management** > **eXtreme Scale session management settings**.
2. Update the fields to enable session persistence to a data grid.

You can also update the application with a wsadmin script. In the following example, the **-SessionManagement** parameter creates the same configuration that you can in the administrative console:

For the remote eXtreme Scale data grid configuration:

```
AdminApp.edit('DefaultApplication','[-SessionManagement[[true
XSRemoteSessionManagement cs0:!:grid0]]]')
```

The :!: characters that are passed are used as delimiters. The values that are passed are:

```
catalogServiceName:!:gridName
```

For the eXtreme Scale embedded scenario with default configuration:

```
AdminApp.edit('DefaultApplication','[-SessionManagement[[true
XSEmbeddedSessionManagement :!:!:!:default]]]')
```

The :!: characters that are passed are used as delimiters. The values that are passed are:

```
catalogServiceName:!:gridName:!:default:!:
absolutePath_to_objectGridXmlfile:!:absolutePath_to_DeploymentXmlfile
```

For the eXtreme Scale embedded scenario with a custom configuration:

```
AdminApp.edit('DefaultApplication','[-SessionManagement[[true
XSEmbeddedSessionManagement
:!:!:!:custom:!:c:\XS\objectgrid.xml:!:c:\XS\objectgriddeployment.xml]]]')
```

The :!: characters that are passed are used as delimiters. The values that are passed are:

```
catalogServiceName:!:gridName:!:custom:!:
absolutePath_to_objectGridXmlfile:!:absolutePath_to_DeploymentXmlfile
```

When you save the changes, the application uses the configured data grid for session persistence on the appliance.

- **To configure session management on an existing server:**
  1. In the WebSphere Application Server administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Session management** > **eXtreme Scale session management settings**.
  2. Update the fields to enable session persistence.

  You can also configure session management on an existing server with the following wsadmin tool commands:

  For the remote eXtreme Scale data grid configuration:

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSRemoteSessionManagement -XSRemoteSessionManagement
[-catalogService cs0 -csGridName grid0]]')
```

  For the eXtreme Scale embedded configuration:

  – The default configuration, if you are using the default XML files:

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSEmbeddedSessionManagement
-XSEmbeddedSessionManagement [-embeddedGridType default -objectGridXML -objectGridDeploymentXML ]]')
```

  – The custom configuration, if you are using customized XML files:

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSEmbeddedSessionManagement
-XSEmbeddedSessionManagement
[-embeddedGridType custom -objectGridXML c:\XS\objectgrid.xml -objectGridDeploymentXML
c:\XS\objectgriddeployment.xml]]')
```

When you save the changes, the server now uses the configured data grid for session persistence with any applications that are running on the server.

Results

You configured HTTP session manager to persist the sessions to a data grid. Entries are removed from the data grid when the sessions time out. See Session management settings for more information about updating the session timeout value in the WebSphere Application Server administrative console.

Related concepts:

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related reference:

"XML files for HTTP session manager configuration" on page 382
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

eXtreme Scale session management settings:

You can configure your WebSphere Application Server applications to use WebSphere eXtreme Scale or a WebSphere DataPower XC10 Appliance for session persistence.

You can edit these settings in the enterprise application installation wizard, or on the application or server detail pages:

- Version 6.1: **Applications** > **Install new application**
- Version 6.1: **Applications** > **Enterprise Applications** > *application_name*
- Version 6.1: **Servers** > **Application servers** > *server_name* > **Web container settings** > **Session management**
- Version 7.0: **Applications** > **New application** > **New Enterprise Application**, and choose the Detailed path for creating the application.
- Version 7.0: **Applications** > **Application Types** > **WebSphere enterprise applications** > *application_name* > **Web Module properties** > **Session management** > **Session management settings**
- Version 7.0: **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Container settings** > **Session management settings**

Enable session management:

Enables session management to use WebSphere eXtreme Scale embedded or remote data grid or a WebSphere DataPower XC10 Appliance for session persistence.

Manage session persistence by:

Specifies how session persistence is managed. You can choose one of the following options:
- WebSphere DataPower XC10 Appliance
- Remote eXtreme Scale data grid
- Embedded eXtreme Scale data grid

The remaining settings that you configure depend on the session persistence mechanism you choose.

WebSphere DataPower XC10 Appliance specific settings:

The following settings are specific to configuring the WebSphere DataPower XC10 Appliance for session persistence.

IP or host name of the WebSphere DataPower XC10 Appliance:

Specifies the IP or host name of the appliance to use for persisting sessions.

IBM WebSphere DataPower XC10 Appliance administrative credentials:

Specifies the **User name** and **Password** that you use to log in to the DataPower XC10 Appliance user interface. Click **Test Connection...** to test the connection to your appliance.

Session persistence preference:

Specifies the data grid on which sessions are persisted. You can choose one of the following options:
- **Persist sessions in a new data grid on the IBM WebSphere DataPower XC10 Appliance**. You can then specify a **Data grid name**.
- **Persist sessions in an existing data grid on the IBM WebSphere DataPower XC10 Appliance**. You can then enter or browse for an **Existing data grid name**.

Remote eXtreme Scale data grid configuration:

The following settings are specific to configuring the remote eXtreme Scale grid for session persistence.

Catalog service domain that manages the remote session data grid:

Specifies the catalog service domain that you want to use to manage your sessions.

If no catalog service domains are displayed, or you want to create a new catalog service domain, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains**.

Remote data grid in which to store session information:

Specifies the name of the data grid in the catalog service domain in which you want to store your session information. The list of active remote grids is populated when you select a catalog service. The remote data grid must already exist in the eXtreme Scale configuration.

Embedded eXtreme Scale data grid configuration:

The following settings are specific to configuring an embedded eXtreme Scale configuration. In the embedded eXtreme Scale scenario, the eXtreme Scale processes are hosted by WebSphere Application Server processes.

eXtreme Scale embedded data grid configuration:

- **Use default ObjectGrid configuration**
- **Specify custom ObjectGrid configuration files**

Full path to `objectgrid.xml` file to copy into configuration
: Specifies the full path to the `objectgrid.xml` file for the configuration that you want to use.

Full path to `objectgriddeployment.xml` file to copy into configuration
: Specifies the full path to the `objectgriddeployment.xml` file for the configuration that you want to use.

Splicing a session data grid application with the `addObjectGridFilter` script:

Use the **`addObjectGridFilter`** command-line script to splice an application with filter declarations and configuration in the form of servlet context initialization parameters.

About this task

For a WebSphere Application Server deployment, the script is in the following location: *was_root*`/optionalLibraries/ObjectGrid/session/bin/addObjectGridFilter.bat/sh` . For a stand-alone deployment, the script is in the following location: *wxs_home*`/ObjectGrid/session/bin/addObjectGridFilter.sh/bat`.

Procedure

Run the **`addObjectGridFilter`** on your application.

Windows

```
addObjectGridFilter.bat [ear_file] [splicer_properties_file]
```

UNIX

```
addObjectGridFilter.sh [ear_file] [splicer_properties_file]
```

- [*ear_file*] - Specifies the absolute path to the enterprise archive file to be spliced.
- [*splicer_properties_file*] : Specifies the absolute path to the splicer properties file that contains various configuration properties.

Results

The servlet filter that is spliced maintains defaults for configuration values. You can override these default values with configuration options that you specify in the properties file in the second argument. For a list of the parameters that you can use, see "Servlet context initialization parameters" on page 388.

You can modify and use the sample `splicer.properties` file that is provided with a eXtreme Scale installation. You can also use the **`addObjectGridServlets`** script, which inserts the session manager by extending each servlet. However, the recommended script is the **`addObjectGridFilter`** script.

Example

UNIX **Example using eXtreme Scale installed on WebSphere Application Server on UNIX:**

1. `cd` *`wxs_home`*`/optionalLibraries/ObjectGrid/session/bin`
2. `addObjectGridFilter.sh /tmp/mySessionTest.ear` *`was_root`*`/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Example using eXtreme Scale installed in a stand-alone directory on UNIX:**

1. `cd` *`was_root`*`/session/bin`
2. `addObjectGridFilter.sh /tmp/mySessionTest.ear` *`was_root`*`/session/samples/splicer.properties`

Editing the `splicer.properties` file:

After you have configured the appliance to store HTTP sessions, you can edit other aspects of the HTTP session configuration with the `splicer.properties` file.

Procedure

1. Find the `splicer.properties` file to edit.

   In a WebSphere Application Server environment:You can get the path location of the `splicer.properties` file by locating the **`sessionFilterProps`** custom property. If you configured session persistence at the server level, the name of the custom property is: `com.ibm.websphere.xs.sessionFilterProps`. If you configured session persistence at the application level, the name of the custom property is: `<`*`application_name`*`>`**,**`com.ibm.websphere.xs.sessionFilterProps`. These custom properties might be in one of the following locations:
   - In a WebSphere Application Server Network Deployment environment: The `splicer.properties` file is on the deployment manager profile path.
   - In a stand-alone WebSphere Application Server environment: A custom property on the application server.
2. Edit the properties in the file. For more information, see "`splicer.properties` file" on page 392.

Using WebSphere eXtreme Scale for SIP session management

You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

SIP session management configuration

To use WebSphere eXtreme Scale as the SIP replication mechanism, set the com.ibm.sip.ha.replicator.type custom property. In the administrative console, select **Application servers** > *my_application_server* > **SIP container** > **Custom**

properties for each server to add the custom property. Type `com.ibm.sip.ha.replicator.type` for the Name and `OBJECTGRID` for the Value.

Use the following properties to customize the behavior of the ObjectGrid that is used to store SIP sessions. In the administrative console, click **Application servers** > *my_application_server* > **SIP container** > **Custom properties** for each server to add the custom property. Type the **Name** and **Value**. Each server must have the same properties set to function properly.

Table 25. Custom properties for SIP session management with ObjectGrid

Property	Value	Default
com.ibm.sip.ha.replicator.type	`OBJECTGRID`: use ObjectGrid as SIP session store	
min.synchronous.replicas	Minimum number of synchronous replicas	0
max.synchronous.replicas	Maximum number of synchronous replicas	0
max.asynchronous.replicas	Maximum number of asynchronous replicas	1
auto.replace.lost.shards	See "Configuring distributed deployments" on page 278 for more information.	true
development.mode	• `true` - allow replicas to be active on same node as primaries • `false` - replicas must be on different node than primaries	false

Related tasks:

"Configuring HTTP session managers" on page 364
The HTTP session manager provides session replication capabilities for an associated application. The session manager works with the Web container to create and manage the life cycles of HTTP sessions that are associated with the application.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere Application Server HTTP session persistence to a data grid" on page 368
You can configure your WebSphere Application Server application to persist sessions to a data grid. This data grid can be in an embedded container server that runs within WebSphere Application Server, or it can be in a remote data grid.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

Related reference:

"XML files for HTTP session manager configuration" on page 382
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Configuring HTTP session manager with WebSphere Portal

You can persist HTTP sessions from WebSphere Portal into a data grid.

Before you begin

Your WebSphere eXtreme Scale and WebSphere Portal environment must meet the following requirements:

- How you install WebSphere eXtreme Scale depends on your deployment scenario. You can run the container servers, which host the data grids, either inside or outside of the WebSphere Application Server cell:
  - If you are running container servers in the WebSphere Application Server cell **(embedded scenario)**: Install both the WebSphere eXtreme Scale client and server on your WebSphere Application Server and WebSphere Portal nodes.
  - If you are running container servers outside of the WebSphere Application Server cell **(remote scenario)**: Install WebSphere eXtreme Scale Client on your WebSphere Application Server and WebSphere Portal nodes.

  See "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server" on page 186 for more information.
- WebSphere Portal Version 7 or later.
- Custom portlets must be configured within WebSphere Portal. The administrative portlets that come with WebSphere Portal cannot currently be integrated with data grids.

About this task

Introducing WebSphere eXtreme Scale into a WebSphere Portal environment can be beneficial in the following scenarios:

Important: Although the following scenarios introduce benefits, increased processor usage in the WebSphere Portal tier can result from introducing WebSphere eXtreme Scale into the environment.

- **When session persistence is required.**

  For example, if the session data from your custom portlets must stay available during a WebSphere Portal Server failure, you can persist the HTTP sessions to the WebSphere eXtreme Scale data grid. Data replicates among many servers, increasing data availability.
- **In a multiple data center topology.**

  If your topology spans multiple data centers across different physical locations, you can persist the WebSphere Portal HTTP sessions to the WebSphere eXtreme Scale data grid. The sessions replicate across data grids in the data centers. If a data center fails, the sessions are rolled over to another data center that has a copy of the data grid data.

- **To lower memory requirements on the WebSphere Portal Server tier.**

  By offloading session data to a remote tier of container servers, a subset of sessions are on the WebSphere Portal servers. This offload of data reduces the memory requirements on the WebSphere Portal Server tier.

Procedure

1. Splice the `wps` WebSphere Portal application and any custom portlets to enable the sessions to be stored in the data grid.

   See "Configuring WebSphere Application Server HTTP session persistence to a data grid" on page 368 for more information. This action results in the splicing of the custom portlets to enable session persistance to your data grid.

   You can splice the application by configuring HTTP session management when you deploy the application, or you can use custom properties to automatically splice your applications. See "Configuring the HTTP session manager with WebSphere Application Server" on page 365 for more information about splicing the application.
2. If you are using the remote scenario, where your container servers are outside of the WebSphere Application Server, explicitly start remote eXtreme Scale containers for remote HTTP session persistence scenarios. Start the containers with the `XS/ObjectGrid/session/samples/objectGridStandAlone.xml` and `objectGridDeploymentStandAlone.xml` configuration files. For example, you might use the following command: UNIX Linux

   ```
   startOgServer.sh xsContainer1 -catalogServiceEndPoints <host>:<port>
   -objectgridFile XS/ObjectGrid/session/samples/objectGridStandAlone.xml -deploymentPolicyFile
   XS/ObjectGrid/session/samples/objectGridDeploymentStandAlone.xml
   ```

   For more information about starting container servers, see "Starting container servers" on page 463. If you are using an embedded scenario, see "Configuring container servers in WebSphere Application Server" on page 325 for more information about configuring and starting container servers.
3. Some versions of WebSphere Portal server can have runtime errors when cookies are added to an HTTP response. Since adds cookies for failover and other purposes, these cookies need to be added to WebSphere Portal server cookie ignore list. For more information, see the cookie.ignore.regex parameter section of Caching pages shared by multiple users on the IBM WebSphere Portal wiki. The two cookies that need to be added to the list are IBMID.* and IBMSessionHandle.*. The updated list may look like this for example `"digest\\ .ignore.*|LtpaToken|LtpaToken2|JSESSIONID|IBMID.*|IBMSessionHandle.*"`. For more information, see Caching pages shared by multiple users on the IBM WebSphere Portal wiki.
4. Restart the WebSphere Portal servers. See WebSphere Portal Version 7: Starting and stopping servers, deployment managers, and node agents for more information.

Results

You can access the WebSphere Portal Server, and HTTP session data for the configured custom portlets is persisted to the data grid.

If the entire data grid that is hosting the application session data is unreachable from the web container client, the client instead uses the base web container in WebSphere Application Server for session management. The data grid might be unreachable in the following scenarios:

- A network problem between the Web container and the remote container servers.
- The remote container server processes have been stopped.

The number of session references kept in memory, specified by **`sessionTableSize`** parameter, is still maintained when the sessions are stored in the base web container. The least recently used sessions are invalidated from the web container session cache when the **`sessionTableSize`** value is exceeded. If the remote data grid becomes available, sessions that were invalidated from the web container cache can retrieve data from the remote data grid and load the data into a new session. If the entire remote data grid is not available and the session is invalidated from the session cache, the user's session data is lost. Because of this issue, you should not shut down the entire production remote data grid when the system is running under load.

Related concepts:

"Interoperability with other products" on page 49
You can integrate WebSphere eXtreme Scale with other products, such as WebSphere Application Server and WebSphere Application Server Community Edition.

"Monitoring with vendor tools" on page 550
WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli Monitoring and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

"Tuning garbage collection with WebSphere Real Time" on page 577
Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary. WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related reference:

"XML files for HTTP session manager configuration" on page 382
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Configuring the HTTP session manager for various application servers

WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

About this task

You can use the HTTP session manager with other application servers that are not running WebSphere Application Server, such as WebSphere Application Server Community Edition. To configure other application servers to use the data grid, you must splice your application and incorporate WebSphere eXtreme Scale Java archive (JAR) files into your application.

Procedure

1. Splice your application so that it can use the session manager. To use the session manager, you must add the appropriate filter declarations to the web deployment descriptors for the application. In addition, session manager configuration parameters are passed in to the session manager in the form of servlet context initialization parameters in the deployment descriptors. There are three ways in which you can introduce this information into your application:
   - **`addObjectGridFilter`** script: For more information, see "Splicing a session data grid application with the **`addObjectGridFilter`** script" on page 374.
   - Ant build script:

     WebSphere eXtreme Scale ships with a `build.xml` file that can be used by Apache Ant, which is included in the *was_root*`/bin` folder of a WebSphere Application Server installation. You can modify the `build.xml` file to change the session manager configuration properties. The configuration properties are identical to the property names in the `splicer.properties` file. After the `build.xml` file has been modified, invoke the Ant process by running `ant.sh`, `ws_ant.sh` (UNIX) or `ant.bat`, `ws_ant.bat` (Windows).
   - Update the web descriptor manually:

     Edit the `web.xml` file that is packaged with the web application to incorporate the filter declaration, its servlet mapping, and servlet context initialization parameters. Do not use this method because it is prone to errors.

   For a list of the parameters that you can use, see "Servlet context initialization parameters" on page 388.

2. Incorporate the WebSphere eXtreme Scale session replication manager JAR files into your application. You can embed the files into the application module `WEB-INF/lib` directory or in the application server classpath. The required JAR files vary depending on the type of containers that you are using:
   - Remote container servers: `ogclient.jar` and `sessionobjectgrid.jar`
   - Embedded container servers: `objectgrid.jar` and `sessionobjectgrid.jar`
3. Optional: If you use remote container servers, start the container servers. See “Starting a stand-alone catalog service” on page 461 for details.
4. Deploy the application. Deploy the application with your normal set of steps for a server or cluster. After you deploy the application, you can start the application.
5. Access the application. You can now access the application, which interacts with the session manager and WebSphere eXtreme Scale.

What to do next

You can change a majority of the configuration attributes for the session manager when you instrument your application to use the session manager. These attributes include variations to the replication type (synchronous or asynchronous), in-memory session table size, and so on. Apart from the attributes that can be changed at application instrumentation time, the only other configuration attributes that you can change after the application deployment are the attributes that are related to the WebSphere eXtreme Scale server cluster topology and the way that their clients (session managers) connect to them.

Remote scenario behavior: If the entire data grid that is hosting the application session data is unreachable from the web container client, the client instead uses the base web container of the application server for session management. The data grid might be unreachable in the following scenarios:
- A network problem between the Web container and the remote container servers.
- The remote container server processes have been stopped.

The number of session references kept in memory, specified by **`sessionTableSize`** parameter, is still maintained when the sessions are stored in the base web container. The least recently used sessions are invalidated from the web container session cache when the **`sessionTableSize`** value is exceeded. If the remote data grid becomes available, sessions that were invalidated from the web container cache can retrieve data from the remote data grid and load the data into a new session. If the entire remote data grid is not available and the session is invalidated from the session cache, the user session data is lost. Because of this issue, do not shut down the entire production remote data grid when the system is running under load.

Related concepts:

“Interoperability with other products” on page 49
You can integrate WebSphere eXtreme Scale with other products, such as WebSphere Application Server and WebSphere Application Server Community Edition.

“Monitoring with vendor tools” on page 550
WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli Monitoring and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

"Tuning garbage collection with WebSphere Real Time" on page 577
Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary. WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related reference:

"XML files for HTTP session manager configuration"
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

XML files for HTTP session manager configuration

When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

Sample files location

These XML files are packaged in *wxs_install_root*/`ObjectGrid/session/samples` for a stand-alone installation or *was_root*/`optionalLibraries/ObjectGrid/session/samples` for WebSphere eXtreme Scale installed in a WebSphere Application Server cell.

Embedded XML package

If you are configuring an embedded scenario, the container server starts in the web container tier. Use the `objectGrid.xml` file and `objectGridDeployment.xml` file, which are provided by default. You can update these files to customize the behavior of the HTTP session manager.

```
<?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>
```

Figure 36. `objectGrid.xml` file

Values you can change:

ObjectGrid name attribute
The value must match the following values in other configuration files:

- The **`objectGridName`** property in the `splicer.properties` file that is used to splice the web application.
- The **`objectgridName`** attribute in the `objectGridDeployment.xml` file.

If you have multiple applications, and you want the session data to be stored in different data grids, those applications must have different ObjectGrid name attribute values.

7.1.1+ **ObjectGrid txTimeout attribute**
This value determines how many seconds a transaction can be open before the container server triggers the transaction to time out. The default is 30 seconds, and can be changed depending on the environment. If the HTTP session persistence is configured with the **`replicationInterval`** servlet context initialization parameter value set greater than zero, transactions are batched on a thread. If the **`replicationInterval`** property is set to `0`, a transaction typically starts when a web application retrieves a valid HttpSession object. The transaction commits at the end of the web application request. If your environment has requests that take longer than 30 seconds, set this value accordingly.

Values you cannot change:

ObjectGridEventListener
The ObjectGridEventListener line cannot be changed and is used internally.

objectgridSessionMetadata
The objectgridSessionMetadata line refers to the map where the HTTP session metadata is stored. There is one entry for every HTTP session stored in the data grid in this map.

objectgridSessionTTL.*
This value cannot be changed and is for future use.

objectgridSessionAttribute.*
The `objectgridSessionAttribute.*` text defines a dynamic map. This value is used to create the map in which HTTP session attributes are stored when the **fragmentedSession** parameter is set to `true` in the `splicer.properties` file. This dynamic map is called `objectgridSessionAttribute`. Another map is created based on this template called `objectgridSessionAttributeEvicted`, which stores sessions that have timed out, but the web container has not invalidated.

A time to live policy (TTL) is defined for the `objectgridSessionMetadata` map definition. The other map, `objectgridSessionAttribute` is dependant on this map and does not require a TTL parameter. For each active HTTP session, an entry gets created in the `objectgridSessionMetadata` map, and one entry in the `objectgridSessionAttribute` map for every session attribute. If an in-memory session does not exist due to an application server failure or a session is removed from the in-memory cache on the application server, the grid must initiate the session invalidation using the TTL eviction policy. At the time of eviction, the attributes are removed from the `objectgridSessionAttribute` map and inserted into a dynamically created map called `objectgridSessionAttributeEvicted` map. The data is stored in this map until an application server can remove the session and complete session invalidation. Therefore, the TTL parameter is only required in the `objectgridSessionMetadata` map definition.

Note: The `objectgridSessionTTL` is not used by WebSphere eXtreme Scale in the current release.

The **MapEventListener** line is internal and cannot be modified.

```
<?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>
```

Figure 37. `objectGridDeployment.xml` file

Values you can change:

ObjectGrid name attribute
The value must match the following values in other configuration files:
- The **objectGridName** property in the `splicer.properties` file that is used to splice the web application.
- The ObjectGrid **name** attribute in the `objectGrid.xml` file.

If you have multiple applications, and you want the session data to be stored in different data grids, those applications must have different ObjectGrid name attribute values.

mapSet element attributes
You can change all mapSet properties except for the placementStrategy attribute.

Name Can be updated to any value.

numberOfPartitions
Specifies the number of primary partitions that are started in each server that is hosting the web application. As you add partitions, the data becomes more spread out in the event of a failover. The default value is 5 partitions, and is fine for most applications.

minSyncReplicas, maxSyncReplicas, and maxAsyncReplicas
Specifies the number and type of replicas that store the HTTP session data. The default is 1 asynchronous replica, which is fine for most applications. Synchronous replication occurs during the request path, which can increase the response times for your web application.

developmentMode
Informs the eXtreme Scale placement service whether the replica shards for a partition can be placed on the same node as its primary shard. You can set the value to true in a development environment, but disable this function in a production environment because a node failure could cause the loss of session data.

placementStrategy
Do not change the value of this attribute.

The rest of the file refers to the same map names as in the `objectGrid.xml` file. These names cannot be changed.

Values you cannot change:

- The placementStrategy attribute on the mapSet element.

Remote XML package

When you are using the remote mode, where the containers run as stand-alone processes, you must use the `objectGridStandAlone.xml` file and the `objectGridDeploymentStandAlone.xml` file to start the processes. You can update these files to modify the configuration.

```
<?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>
```

Figure 38. `objectGridStandAlone.xml` file

Values you can change:

ObjectGrid name attribute
The value must match the following values in other configuration files:

- The **`objectGridName`** property in the `splicer.properties` file that is used to splice the web application.
- The **`objectgridName`** attribute in the `objectGridStandAlone.xml` file.

If you have multiple applications, and you want the session data to be stored in different data grids, those applications must have different ObjectGrid name attribute values.

7.1.1+ **ObjectGrid txTimeout attribute**

This value determines how many seconds a transaction can be open before the container server triggers the transaction to time out. The default is 30 seconds, and can be changed depending on the environment. If the HTTP session persistence is configured with the **`replicationInterval`** servlet context initialization parameter value set greater than zero, transactions are batched on a thread. If the **`replicationInterval`** property is set to `0`, a transaction typically starts when a web application retrieves a valid HttpSession object. The transaction commits at the end of the web application request. If your environment has requests that take longer than 30 seconds, set this value accordingly.

Values you cannot change:

ObjectGridEventListener

The ObjectGridEventListener line cannot be changed and is used internally.

objectgridSessionMetadata

The objectgridSessionMetadata line refers to the map where the HTTP session metadata is stored. There is one entry for every HTTP session stored in the data grid in this map.

objectgridSessionTTL.*

This value cannot be changed and is for future use.

objectgridSessionAttribute.*

The `objectgridSessionAttribute.*` text defines a dynamic map. This value is used to create the map in which HTTP session attributes are stored when the **`fragmentedSession`** parameter is set to `true` in the `splicer.properties` file. This dynamic map is called `objectgridSessionAttribute`. Another map is created based on this template called `objectgridSessionAttributeEvicted`, which stores sessions that have timed out, but the web container has not invalidated.

A time to live policy (TTL) is defined for the `objectgridSessionMetadata` map definition. The other map, `objectgridSessionAttribute` is dependant on this map and does not require a TTL parameter. For each active HTTP session, an entry gets created in the `objectgridSessionMetadata` map, and one entry in the `objectgridSessionAttribute` map for every session attribute. If an in-memory session does not exist due to an application server failure or a session is removed from the in-memory cache on the application server, the grid must initiate the session invalidation using the TTL eviction policy. At the time of eviction, the attributes are removed from the `objectgridSessionAttribute` map and inserted into a dynamically created map called `objectgridSessionAttributeEvicted` map. The data is stored in this map until an application server can remove the session and complete session invalidation. Therefore, the TTL parameter is only required in the `objectgridSessionMetadata` map definition.

Note: The `objectgridSessionTTL` is not used by WebSphere eXtreme Scale in the current release.

The **MetadataMapListener** line is internal and cannot be modified.

```
<?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="47" minSyncReplicas="0" maxSyncReplicas="0"
   maxAsyncReplicas="1" developmentMode="false" placementStrategy="FIXED_PARTITIONS">
           <map ref="objectgridSessionMetadata"/>
           <map ref="objectgridSessionAttribute.*"/>
           <map ref="objectgridSessionTTL.*"/>
     </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

Figure 39. `objectGridDeploymentStandAlone.xml` file

Values you can change:

objectgridName attribute
The value must match the following values in other configuration files:
- The **objectGridName** property in the `splicer.properties` file that is used to splice the web application.
- The ObjectGrid **name** attribute in the `objectGrid.xml` file.

If you have multiple applications, and you want the session data to be stored in different data grids, those applications must have different ObjectGrid name attribute values.

mapSet element attributes
You can change all mapSet properties.

Name Can be updated to any value.

numberOfPartitions
When using the default FIXED_PARTITIONS placement strategy, this specifies the number of total partitions that will be spread across all running grid containers. The default value is 47 partitions, and is fine for most applications. If a PER_CONTAINER placement strategy is used, then this specifies the number of primary partitions started in each grid container. As you add partitions, the data becomes more spread out in the event of a failover. The recommended value is 5 for the PER_CONTAINER strategy.

minSyncReplicas, maxSyncReplicas, and maxAsyncReplicas
Specifies the number of primary partitions that are started in each server that is hosting the web application. As you add partitions, the data becomes more spread out in the event of a failover. The default value is 5 partitions, and is fine for most applications.

developmentMode
Informs the eXtreme Scale placement service whether the replica shards for a partition can be placed on the same node as its primary shard. You can set the value to true in a development environment, but disable this function in a production environment because a node failure could cause the loss of session data.

placementStrategy
You can change this attribute to one of the following:

- FIXED_PARTITIONS This is the default value and is the preferred approach for using a remote HTTP Session topology. It is required if you are using Multi-Master replication
- PER_CONTAINER This is still a supported configuration in a remote topology.

Related concepts:

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related tasks:

"Configuring HTTP session managers" on page 364
The HTTP session manager provides session replication capabilities for an associated application. The session manager works with the Web container to create and manage the life cycles of HTTP sessions that are associated with the application.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere Application Server HTTP session persistence to a data grid" on page 368
You can configure your WebSphere Application Server application to persist sessions to a data grid. This data grid can be in an embedded container server that runs within WebSphere Application Server, or it can be in a remote data grid.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Troubleshooting cache integration" on page 628
Use this information to troubleshoot issues with your cache integration configuration, including HTTP session and dynamic cache configurations.

Servlet context initialization parameters

The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

Parameters

`applicationQualifiedCookies`

A string value of either `true` or `false`. Set to `true` if your environment contains multiple applications that use unique cookie names. Default is `false`, which assumes all applications are using the same cookie name.

`authenticationRetryCount`

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.

`catalogHostPort`

The catalog server can be contacted to obtain a client side ObjectGrid instance. The value must be of the form `host:port<,host:port>`. The host is the listener host on which the catalog server is running. The port is the listener port for that catalog server process. This list can be arbitrarily long and is used for bootstrapping only. The first viable address is used. It is optional inside WebSphere Application Server if the **`catalog.services.cluster`** property is configured.

`credentialAuthentication`

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.

`cookieDomain`

Specifies if you require sessions to be accessible across hosts. Set the value to the name of the common domain between the hosts.

`cookiePath`

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`

The name of the class that implements the com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator interface. This class is used to obtain credentials for clients.

`credentialGeneratorProps`

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.

`enableSessionStats`

A string value of either `true` or `false`. Enables eXtreme Scale client HTTP Sessions statistics tracking.

`fragmentedSession`

A string value of either `true` or `false`. The default value is `true`. Use this setting to control whether the product stores session data as a whole entry, or stores each attribute separately.

Set the fragmentedSession parameter to `true` if the web application session has many attributes or attributes with large sizes. Set fragmentedSession to `false` if a session has few attributes, because all the attributes are stored in the same key in the data grid.

In the previous, filter-based implementation, this property was referred to as persistenceMechanism, with the possible values of ObjectGridStore (fragmented) and ObjectGridAtomicSessionStore (not fragmented).

objectGridType

A string value of either `REMOTE` or `EMBEDDED`. The default is `REMOTE`.

If it is set to `REMOTE`, the session data is stored outside of the server on which the web application is running.

If it is set to `EMBEDDED`, an embedded eXtreme Scale container starts in the application server process on which the web application is running.

objectGridName

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 container servers.

objectGridXML

The file location of the `objectgrid.xml` file. The built-in XML file packaged in the eXtreme Scale library is loaded automatically if `objectGridType=EMBEDDED` and the **objectGridXML** property is not specified.

objectGridDeploymentXML

Specifies the location of the objectGrid deployment policy XML file. The built-in XML file packaged in the eXtreme Scale library is loaded automatically if `objectGridType=EMBEDDED` and the **objectGridDeploymentXML** property is not specified.

replicationInterval

An integer value (in seconds) that defines the time between writing of updated sessions to ObjectGrid. The default is `10` seconds. Possible values are from 0 to 60. 0 means that updated sessions are written to the ObjectGrid at the end of servlet service method call for each request. A higher **replicationInterval** value improves performance because fewer updates are written to the data grid. However, a higher value makes the configuration less fault tolerant.

This setting applies only when objectGridType is set to `REMOTE`.

reuseSessionID

A string value of either `true` or `false`. The default is `false`. Set to `true` if the underlying web container reuses session IDs across requests to different hosts. The value of this property must be the same as the value in the web container. If you are using WebSphere Application Server and configuring eXtreme Scale HTTP session persistence using the administrative console or **wsadmin** tool scripting, the web container custom property `HttpSessionIdReuse=true` is added by default. The **reuseSessionID** is also set to `true`. If you do not want the session IDs to be reused, set the `HttpSessionIdReuse=false` custom property on the web container custom property before you configure eXtreme Scale session persistence.

sessionIdOverrideClass

The name of the class that implements the com.ibm.websphere.xs.sessionmanager.SessionIDOverride interface. This class is used to override the unique session identifier retrieved with the HttpSession.getId() method so that all applications have the same ID. The default is to use the user ID derived from the HttpSession.getId().

`sessionStatsSpec = session.all = enabled`

A string of eXtreme Scale client HTTP statistics specification.

`shareSessionsAcrossWebApps`

A string value of either `true` or `false`. The default is `false`. Specifies if sessions are shared across web applications, specified as a string value of either `true` or `false`. The servlet specification states that HTTP Sessions cannot be shared across web applications. An extension to the servlet specification is provided to allow this sharing.

`sessionTableSize`

An integer value that defines the number of session references kept in memory. The default is `1000`.

This setting pertains only to a `REMOTE` topology because the `EMBEDDED` topology already has the session data in the same tier as the web container.

Sessions are evicted from the in-memory table based on least recently used (LRU) logic. When a session is evicted from the in-memory table, it is invalidated from the web container. However, the data is not removed from the grid, so subsequent requests for that session can still retrieve the data. This value must be set higher than the web container maximum thread pool value, which reduces contention on the session cache.

`securityEnabled`

A string value of either `true` or `false`. The default value is `false`. This setting enables eXtreme Scale client security. It must match the **`securityEnabled`** setting in the eXtreme Scale server properties file. If the settings do not match, an exception occurs.

`sessionIdOverrideClass`

Overrides the retrieved session ID of an application. The default is to use the ID derived from the HttpSession.getId() method. Enables eXtreme Scale client HTTP Sessions to override the unique session ID of an application so that all applications are retrieved with the same ID. Set to the implementation of the com.ibm.websphere.xs.sessionmanager.SessionIDOverride interface. This interface determines the HttpSession ID based on the HttpServletRequest object.

`traceSpec`

Specifies the IBM WebSphere trace specification as a string value. Use this setting for application servers other than WebSphere Application Server.

`traceFile`

Specifies the trace file location as a string value. Use this setting for application servers other than WebSphere Application Server.

`useURLEncoding`

A string value of either `true` or `false`. The default is `false`. Set to `true` if you want to enable URL rewriting. The default value is `false`, which indicates that cookies are used to store session data. The value of this parameter must be the same as the web container settings for session management.

`useCookies`

A string value of either `true` or `false`. Set to true if the underlying web container will reuse session ID's across requests to different hosts. The default is `false`. The value of this should be the same as what is set in the web container.

Related concepts:

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related tasks:

"Configuring HTTP session managers" on page 364
The HTTP session manager provides session replication capabilities for an associated application. The session manager works with the Web container to create and manage the life cycles of HTTP sessions that are associated with the application.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere Application Server HTTP session persistence to a data grid" on page 368
You can configure your WebSphere Application Server application to persist sessions to a data grid. This data grid can be in an embedded container server that runs within WebSphere Application Server, or it can be in a remote data grid.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Troubleshooting cache integration" on page 628
Use this information to troubleshoot issues with your cache integration configuration, including HTTP session and dynamic cache configurations.

`splicer.properties` file

The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Sample splicer properties

If you choose to use any of the additional properties that are described in this file, be sure to uncomment the lines for the properties that you want to enable.

```
# Properties file that contains all the configuration
# options that the servlet filter based ObjectGrid session
# manager can be configured to use.
#
# This properties file can be made to hold all the default
# values to be assigned to these configuration settings, and
# individual settings can be overridden using ANT Task
# properties, if this properties file is used in conjunction
# with the filtersplicer ANT task.

# A string value of either "REMOTE" or "EMBEDDED".  The default is REMOTE.
# 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

# Catalog Server can be contacted to obtain a client side
# ObjectGrid instance.  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.
# This list can be arbitrarily long and is used for bootstrapping only.
# The first viable address will be used.  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

# An integer value that defines the number of session references
# kept in memory. The default is 1000. 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. If the settings do not match, an exception occurs.

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 =

# A string of IBM WebShere trace specification,
# useful for all other application servers besides WebSphere.

# traceSpec =

# A string of trace file location.
# 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=

# This property should be set to the same path you have configured
# for your application server cookie settings. The default path
# is /.

# cookiePath

# 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=

# A string value of either "true" or "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

# A string value of either "true" or "false", default is "false".
# Set to true if you want to enable urlRewriting.  Default is
# false. The value of this should reflect what is set in the
# web container settings for session management.

# useURLEncoding = false

# Set to false if you want to disable cookies as the session tracking
# mechanism.  Default is true. The value of this should reflect what
# is set in the web container settings for session management.

# useCookies = true


# A string of eXtreme Scale client Http session statistics specification,

# sessionStatsSpec = session.all=enabled

# Set to true if your environment contains multiple applications that
# use unique cookie names. Default is false, which assumes all applications
# are using the same cookie name.

# applicationQualifiedCookies = false



# The prefix of the two cookies that are added to the HTTP response that
# represent the ID of the session object in the data grid and the session
# handle that contains the session's data. Default is IBM

# cookieNamePrefix = IBM

# When listenerMode = true (default), use the web container to generate sessions.
# if it is set to false, the web container will not be used. Only supported when
# reuseSessionId = true, sessionTableSize > 0, and when installed on WebSphere
# Application Server.

# listenerMode = true


# Only applies when listenerMode=false. When this property is set to true, all
# listeners configured for this web application will get the HttpSessionListener.sessionCreated
# call whenever a session is created, or a session is retrieved from the remote grid.
# Examples of this would be when an application server fails, or the
# sessionTableSize is exceeded and a session has to be brought back into the
# application server from the remote grid. HttpSessionListener.sessionDestroyed will also be
# called when a session is invalidated from the in-memory session cache if the sessionTableSize
# limit is exceeded.

# sessionCreatedOnFailover = false
```

Related concepts:

HTTP session management
The session replication manager that is shipped with WebSphere eXtreme Scale can work with the default session manager in WebSphere Application Server. Session data is replicated from one process to another process to support user session data high availability.

"Using WebSphere eXtreme Scale for SIP session management" on page 375
You can use WebSphere eXtreme Scale as a Session Initiation Protocol (SIP) replication mechanism as a reliable alternative to the data replication service (DRS) for SIP session replication.

Related tasks:

"Configuring HTTP session managers" on page 364
The HTTP session manager provides session replication capabilities for an associated application. The session manager works with the Web container to create and manage the life cycles of HTTP sessions that are associated with the application.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere Application Server HTTP session persistence to a data grid" on page 368
You can configure your WebSphere Application Server application to persist sessions to a data grid. This data grid can be in an embedded container server that runs within WebSphere Application Server, or it can be in a remote data grid.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Troubleshooting cache integration" on page 628
Use this information to troubleshoot issues with your cache integration configuration, including HTTP session and dynamic cache configurations.

Configuring the dynamic cache provider for WebSphere eXtreme Scale

7.1.1

Installing and configuring the dynamic cache provider for eXtreme Scale depends on what your requirements are and the environment you have set up.

Before you begin

- To use the dynamic cache provider, WebSphere eXtreme Scale must be installed on top of the WebSphere Application Server node deployments, including the deployment manager node. See 7.1.1 "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client with WebSphere Application Server" on page 186 for more information.
- Global security must be enabled in the WebSphere Application Server administrative console, if the catalog servers within your catalog service domain have Secure Sockets Layer (SSL) enabled or you want to use SSL for a catalog service domain with SSL supported. You require SSL for a catalog server by setting the transportType attribute to `SSL-Required` in the Server properties file. For more information about configuring global security, see Global security settings.

About this task

For information about using the eXtreme Scale dynamic cache provider with IBM WebSphere Commerce, see the following topics in the IBM WebSphere Commerce documentation:

- Enabling the dynamic cache service and servlet caching
- Enabling WebSphere Commerce data cache

If you are not specifically directing your caching to a defined Object Cache or Servlet Cache instance, then it is likely that the Dynamic Cache API calls are being serviced by the baseCache. If you want to use the eXtreme Scale dynamic cache provider for JSP, Web services or command caching, then you must set the baseCache instance to use the eXtreme Scale dynamic cache provider. The same configuration properties are used to configure the baseCache instance. Remember that these configuration properties need to be set as Java Virtual Machine (JVM) custom properties. This caveat applies to any cache configuration property discussed in this section except for servlet caching. To use eXtreme Scale with the dynamic cache provider for servlet caching, be sure to configure enablement in system properties rather than custom properties.

Procedure

1. Enable the eXtreme Scale dynamic cache provider.
   - **WebSphere Application Server Version 7.0 and later:**

     You can configure the dynamic cache service to use the eXtreme Scale dynamic cache provider with the administrative console. After you install eXtreme Scale, the eXtreme Scale dynamic cache provider is immediately available as a **Cache Provider** option in the administrative console. For more information, see WebSphere Application Server Version 7.0 information center: Selecting a cache service provider.
   - **WebSphere Application Server Version 6.1:**

     Use a custom property to configure the dynamic cache service to use the eXtreme Scale dynamic cache provider. You can also use these custom properties in WebSphere Application Server Version 7.0 and later. To create a custom property on a cache instance, click **Resources** > **Cache instances** >

cache_instance_type > **cache_instance_name** > **Custom properties** > **New**. If you are using the base cache instance, create the custom properties on the JVM.

com.ibm.ws.cache.CacheConfig.cacheProviderName
: To use the eXtreme Scale dynamic cache provider, set the value to `com.ibm.ws.objectgrid.dynacache.CacheProviderImpl`. You can create this custom property on a dynamic cache instance, or the base cache instance. If you choose to set the custom property on the base cache instance, then all other cache instances on the server use the eXtreme Scale provider by default. Any eXtreme Scale dynamic cache provider configuration properties set for the baseCache are the default configuration properties for all cache instances backed by eXtreme Scale. To override the base cache instance and make a particular dynamic cache instance use the default dynamic cache provider, create the com.ibm.ws.cache.CacheConfig.cacheProviderName custom property on the dynamic cache instance and set the value to `default`.

2. Optional: If you are using replicated cache instances, configure the replication setting for the cache.

   With the eXtreme Scale dynamic cache provider, you can have local cache instances or replicated cache instances. If you are only using local cache instances, you can skip this step.

   Use one of the following methods to configure the replicated cache:

   - Enable cache replication with the administrative console. You can enable cache replication at any time in WebSphere Application Server Version 7.0. In WebSphere Application Server Version 6.1, you must create a DRS replication domain.
   - Enable cache replication with the com.ibm.ws.cache.CacheConfig.enableCacheReplication custom property to force the cache to report that it is a replicated cache, even though a DRS replication domain has not been assigned to it. Set the value of this custom property to `true`. Set this custom property on the cache instance if you are using an object cache or servlet cache, or on the JVM if you are using the baseCache instance.
3. Optional: If you are using eXtreme Scale as a JSP fragment cache, set the com.ibm.ws.cache.CacheConfig.disableTemplateInvalidation custom property to `true` to disable template-based invalidations during JSP reloads.
4. Configure the topology for the dynamic cache service.

   The only required configuration parameter for the eXtreme Scale dynamic cache provider is the cache topology. Set the custom property on the cache instance or for the dynamic cache service if you are using baseCache instance. Enter the name of the custom property as: `com.ibm.websphere.xs.dynacache.topology`.

   The three possible values for this property follow. You must use one of the allowed values:

   - `embedded`
   - `embedded_partitioned`
   - `remote`

   If you are using embedded or embedded partitioned topologies, consider setting the `com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent` custom property to `true` to save some serialization costs. Set this custom property on the cache instance or the JVM if you are using the baseCache instance.

5. Optional: If you are using an embedded partitioned topology, configure the number of initial containers for the dynamic cache service.

   You can maximize the performance of caches that are using the embedded partitioned topology by configuring the number of initial containers. Set the variable as a system property on the WebSphere Application Server Java virtual machine.

   Enter the name of the property as: `com.ibm.websphere.xs.dynacache.num_initial_containers`.

   The recommended value of this configuration property is an integer that is equal to or slightly less than the total number of WebSphere Application Server instances that are accessing this distributed cache instance. For example, if a dynamic cache service is shared between data grid members, then the value should be set to the number of data grid members.

   For embedded or embedded_partitioned topologies, you must be using Version 7.0 of WebSphere Application Server. Set the following custom property on the JVM process to ensure that the initial containers are available right away.

   ```
   com.ibm.ws.cache.CacheConfig.createCacheAtServerStartup=true
   ```
6. Configure the eXtreme Scale catalog service grid.

   When you are using eXtreme Scale as the dynamic cache provider for a distributed cache instance, you must configure an eXtreme Scale catalog service domain.

   A single catalog service domain can service multiple dynamic cache service providers backed by eXtreme Scale.

   A catalog service can run inside or outside of WebSphere Application Server processes. Starting with eXtreme Scale Version 7.1, when you use the administrative console to configure catalog service domains, the dynamic cache uses these settings. It is not necessary to take additional steps to set up a catalog service. For more information, see “Creating catalog service domains in WebSphere Application Server” on page 299.
7. Configure custom key objects.

   When you are using custom objects as keys the objects must implement the Serializable or Externalizable interface. When you are using the embedded or embedded partitioned topologies, you must place objects on the WebSphere shared library path, just like if they were being used with the default dynamic cache provider. See Using the DistributedMap and DistributedObjectCache interfaces for the dynamic cache in the WebSphere Application Server Network Deployment information center for more details.

   If you are using the remote topology, you must place the custom key objects on the CLASSPATH for the standalone eXtreme Scale containers. See “Starting container servers” on page 463 for more information.
8. Optional: If you are using a remote topology, configure the eXtreme Scale container servers.
   - **Embedded or embedded partitioned topology:**

     The cached data is stored in WebSphere eXtreme Scale container servers. Container servers can run inside or outside of WebSphere Application Server processes. The eXtreme Scale provider automatically creates containers inside the WebSphere process when you are using embedded or embedded partitioned topologies for a cache instance. No further configuration is needed for these topologies.
   - **Remote topology:**

     When you are using the remote topology, you must start up stand-alone eXtreme Scale container servers before the WebSphere Application Server instances that access the cache instance start. To start stand-alone container servers, see "Starting and stopping stand-alone servers" on page 459. Verify that all the container servers for a specific dynamic cache service point to the same catalog service endpoints.

The XML configuration files for the stand-alone eXtreme Scale dynamic cache provider containers are in either the *was_root*/optionalLibraries/ObjectGrid/dynacache/etc directory for installations on top of WebSphere Application Server, or the *wxs_install_root*/ObjectGrid/dynacache/etc directory for stand-alone installations. The files are named dynacache-remote-objectgrid.xml and dynacache-remote-definition.xml. Make copies of these files to edit and use when you are starting stand-alone containers for the eXtreme Scale dynamic cache provider. The **numInitialContainers** parameter in the **dynacache-remote-deployment.xml** file must match the number of container processes that are running. Note that the **numberOfPartitions** attribute in the dynacache-remote-objectgrid.xml file has a default value of 47.

Note: The set of container server processes must have enough free memory to service all the dynamic cache instances that are configured to use the remote topology. Any WebSphere Application Server process that shares the same or equivalent values for the catalog.services.cluster custom property must use the same set of stand-alone containers. The number of containers and number of servers on which they reside must be sized appropriately. See "Dynamic cache capacity planning" on page 68 for additional details.

A command line entry that starts a stand-alone container for the eXtreme Scale dynamic cache provider follows:

UNIX

```
startOgServer.sh container1 -objectGridFile
../dynacache/etc/dynacache-remote-objectgrid.xml -deploymentPolicyFile
../dynacache/etc/dynacache-remote-deployment.xml -catalogServiceEndPoints
MyServer1.company.com:2809
```

9. For distributed or embedded topologies, enable the sizing agent to improve memory consumption estimates.

   The sizing agent estimates memory consumption (usedBytes statistic). The agent requires a Java 5 or higher JVM.

   Load the agent by adding the following argument to the JVM command line:

   `-javaagent:WXS lib directory/wxssizeagent.jar`

   For an embedded topology, add the argument to the command line of the WebSphere Application Server process.

   For a distributed topology, add the argument to command line of the eXtreme Scale processes (containers) and the WebSphere Application Server process.

Related concepts:

"Dynamic cache capacity planning" on page 68
The Dynamic Cache API is available to Java EE applications that are deployed in WebSphere Application Server. You can use the dynamic cache to cache business data, generated HTML, or to synchronize the cached data in the cell by using the data replication service (DRS).

Dynamic cache provider overview
The WebSphere Application Server provides a dynamic cache service that is available to deployed Java EE applications. This service is used to cache data such as output from servlet, JSP, or commands, and object data programmatically specified within an enterprise application with the DistributedMap APIs. .

"Data invalidation" on page 32
To remove stale cache data, you can use invalidation mechanisms.

JPA level 2 (L2) cache plug-in

WebSphere eXtreme Scale includes level 2 (L2) cache plug-ins for both OpenJPA and Hibernate Java Persistence API (JPA) providers. When you use one of these plug-ins, your application uses the JPA API. A data grid is introduced between the application and the database, improving response times.

Using eXtreme Scale as an L2 cache provider increases performance when you are reading and querying data and reduces load to the database. WebSphere eXtreme Scale has advantages over built-in cache implementations because the cache is automatically replicated between all processes. When one client caches a value, all other clients are able to use the cached value that is locally in-memory.

You can configure the topology and properties for the L2 cache provider in the `persistence.xml` file. For more information about configuring these properties, see 7.1.1 *.

Tip: The JPA L2 cache plug-in requires an application that uses the JPA APIs. If you want to use WebSphere eXtreme Scale APIs to access a JPA data source, use the JPA loader. For more information, see JPA Loaders.

JPA L2 cache topology considerations

The following factors affect which type of topology to configure:

1. **How much data do you expect to be cached?**
   - If the data can fit into a single JVM heap, use the “Embedded topology” on page 403 or “Intra-domain topology” on page 402.
   - If the data cannot fit into a single JVM heap, use the “Embedded, partitioned topology” on page 404, or “Remote topology” on page 405
2. **What is the expected read-to-write ratio?**

   The read-to-write ratio affects the performance of the L2 cache. Each topology handles read and write operations differently.
   - “Embedded topology” on page 403: local read, remote write
   - “Intra-domain topology” on page 402: local read, local write
   - “Embedded, partitioned topology” on page 404: Partitioned: remote read, remote write
   - “Remote topology” on page 405: remote read, remote write.

   Applications that are mostly read-only should use embedded and intra-domain topologies when possible. Applications that do more writing should use intra-domain topologies.
3. **What is percentage of data is queried versus found by a key?**

   When enabled, query operations make use of the JPA query cache. Enable the JPA query cache for applications with high read to write ratios only, for example when you are approaching 99% read operations. If you use JPA query cache, you must use the “Embedded topology” on page 403 or “Intra-domain topology” on page 402.

   The find-by-key operation fetches a target entity if the target entity does not have any relationship. If the target entity has relationships with the EAGER fetch type, these relationships are fetched along with the target entity. In JPA data cache, fetching these relationships causes a few cache hits to get all the relationship data.
4. **What is the tolerated staleness level of the data?**

In a system with few JVMs, data replication latency exists for write operations. The goal of the cache is to maintain an ultimate synchronized data view across all JVMs. When you are using the intra-domain topology, a data replication delay exists for write operations. Applications using this topology must be able to tolerate stale reads and simultaneous writes that might overwrite data.

7.1.1+
Intra-domain topology

With an intra-domain topology, primary shards are placed on every container server in the topology. These primary shards contain the full set of data for the partition. Any of these primary shards can also complete cache write operations. This configuration eliminates the bottleneck in the embedded topology where all the cache write operations must go through a single primary shard.

In an intra-domain topology, no replica shards are created, even if you have defined replicas in your configuration files. Each redundant primary shard contains a full copy of the data, so each primary shard can also be considered as a replica shard. This configuration uses a single partition, similar to the embedded topology.

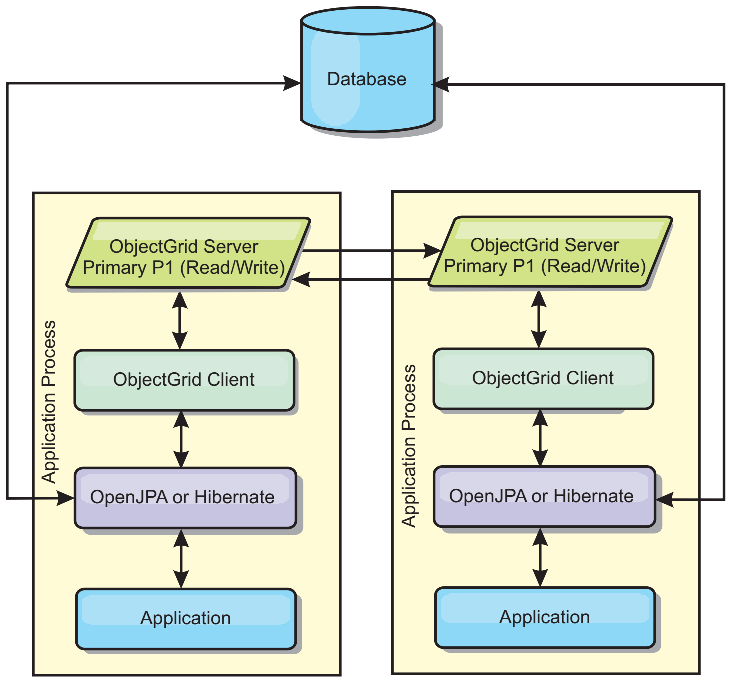


Figure 40. JPA intra-domain topology

Related JPA cache configuration properties for the intra-domain topology:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED,PlacementScope=CONTAINER_SCOPE,PlacementScopeTopology=HUB | RING
```

Advantages:
- Cache reads and updates are local.
- Simple to configure.

Limitations:
- This topology is best suited for when the container servers can contain the entire set of partition data.

- Replica shards, even if they are configured, are never placed because every container server hosts a primary shard. However, all the primary shards are replicating with the other primary shards, so these primary shards become replicas of each other.

Embedded topology

Tip: Consider using an intra-domain topology for the best performance.

An embedded topology creates a container server within the process space of each application. OpenJPA and Hibernate read the in-memory copy of the cache directly and write to all of the other copies. You can improve the write performance by using asynchronous replication. This default topology performs best when the amount of cached data is small enough to fit in a single process. With an embedded topology, create a single partition for the data.

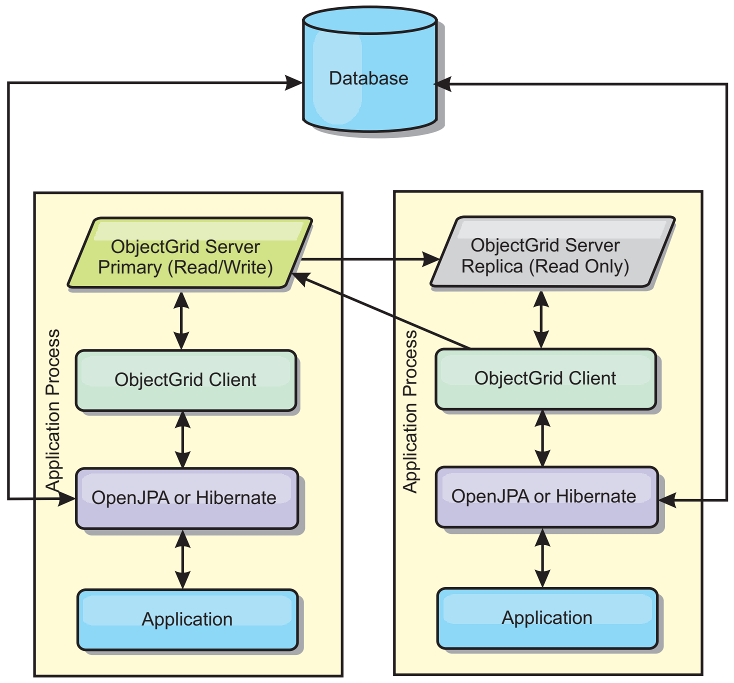


Figure 41. JPA embedded topology

Related JPA cache configuration properties for the embedded topology:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED,MaxNumberOfReplicas=num_replicas,ReplicaMode=SYNC | ASYNC | NONE
```

Advantages:

- All cache reads are fast, local accesses.
- Simple to configure.

Limitations:

- Amount of data is limited to the size of the process.
- All cache updates are sent through one primary shard, which creates a bottleneck.

Embedded, partitioned topology

Tip: Consider using an intra-domain topology for the best performance.

CAUTION:
Do not use the JPA query cache with an embedded partitioned topology. The query cache stores query results that are a collection of entity keys. The query cache fetches all entity data from the data cache. Because the data cache is divided up between multiple processes, these additional calls can negate the benefits of the L2 cache.

When the cached data is too large to fit in a single process, you can use the embedded, partitioned topology. This topology divides the data over multiple processes. The data is divided between the primary shards, so each primary shard contains a subset of the data. You can still use this option when database latency is high.

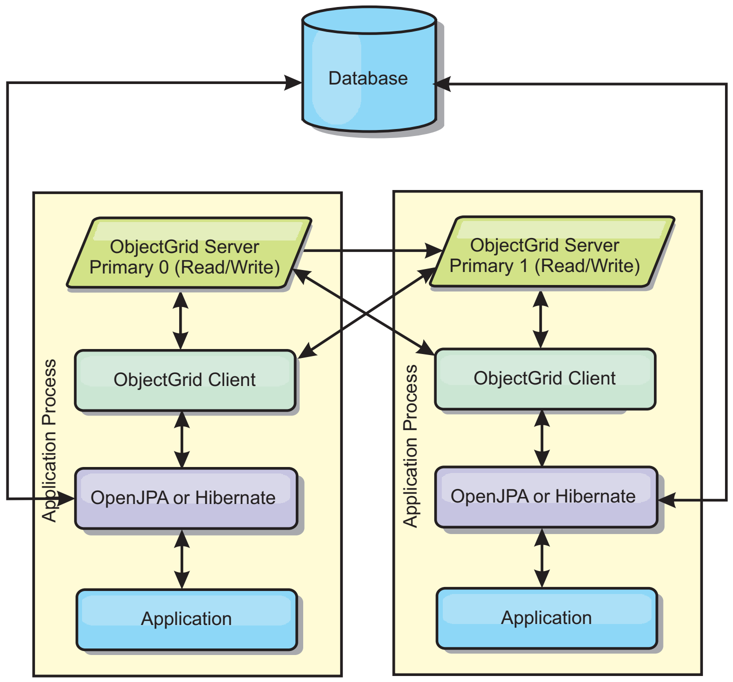


Figure 42. JPA embedded, partitioned topology

Related JPA cache configuration properties for the embedded, partitioned topology:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED_PARTITION,ReplicaMode=SYNC | ASYNC | NONE,
NumberOfPartitions=num_partitions,ReplicaReadEnabled=TRUE | FALSE
```

Advantages:
- Stores large amounts of data.
- Simple to configure
- Cache updates are spread over multiple processes.

Limitation:
- Most cache reads and updates are remote.

For example, to cache 10 GB of data with a maximum of 1 GB per JVM, 10 Java virtual machines are required. The number of partitions must therefore be set to 10 or more. Ideally, the number of partitions must be set to a prime number where each shard stores a reasonable amount of memory. Usually, the numberOfPartitions setting is equal to the number of Java virtual machines. With this setting, each JVM stores one partition. If you enable replication, you must increase the number of Java virtual machines in the system. Otherwise, each JVM also stores one replica partition, which consumes as much memory as a primary partition.

Read about sizing memory and partition count calculation in the *Administration Guide* to maximize the performance of your chosen configuration.

For example, in a system with four Java virtual machines, and the numberOfPartitions setting value of `4`, each JVM hosts a primary partition. A read operation has a 25 percent chance of fetching data from a locally available partition, which is much faster compared to getting data from a remote JVM. If a read operation, such as running a query, needs to fetch a collection of data that involves 4 partitions evenly, 75 percent of the calls are remote and 25 percent of the calls are local. If the ReplicaMode setting is set to either `SYNC` or `ASYNC` and the ReplicaReadEnabled setting is set to `true`, then four replica partitions are created and spread across four Java virtual machines. Each JVM hosts one primary partition and one replica partition. The chance that the read operation runs locally increases to 50 percent. The read operation that fetches a collection of data that involves four partitions evenly has 50 percent remote calls and 50 percent local calls. Local calls are much faster than remote calls. Whenever remote calls occur, the performance drops.

Remote topology

CAUTION:
Do not use the JPA query cache with a remote topology. The query cache stores query results that are a collection of entity keys. The query cache fetches all entity data from the data cache. Because the data cache is remote, these additional calls can negate the benefits of the L2 cache.

Tip: Consider using an intra-domain topology for the best performance.

A remote topology stores all of the cached data in one or more separate processes, reducing memory use of the application processes. You can take advantage of distributing your data over separate processes by deploying a partitioned, replicated eXtreme Scale data grid. As opposed to the embedded and embedded partitioned configurations described in the previous sections, if you want to manage the remote data grid, you must do so independent of the application and JPA provider.

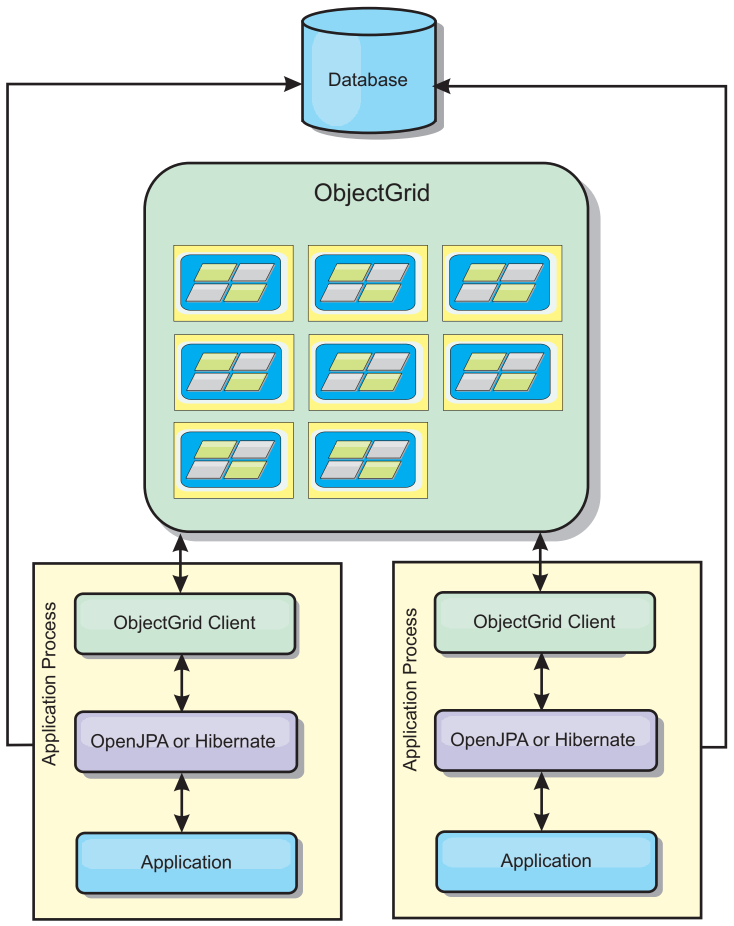


Figure 43. JPA remote topology

Related JPA cache configuration properties for the remote topology:

`ObjectGridName=`*`objectgrid_name`*`,ObjectGridType=REMOTE,AllowNearCache=TRUE`

Note: The AllowNearCache property is optional. If it is not included in the configuration, the default value is FALSE. This property is only used by a remote object grid type as long as the remote object grid server is also enabled for near caching as defined in the ObjectGrid descriptor XML file. To enable the L2 cache provider for near caching, set the property AllowNearCache is set to `TRUE`.

The REMOTE ObjectGrid type does not require any property settings because the ObjectGrid and deployment policy is defined separately from the JPA application. The JPA cache plug-in remotely connects to an existing remote ObjectGrid.

Because all interaction with the ObjectGrid is remote, this topology has the slowest performance among all ObjectGrid types.

Advantages:
- Stores large amounts of data.
- Application process is free of cached data.
- Cache updates are spread over multiple processes.
- Flexible configuration options.

Limitation:

- All cache reads and updates are remote.

Related tasks:

"Configuring the OpenJPA cache plug-in"
You can configure both DataCache and QueryCache implementations for OpenJPA.

"Troubleshooting multiple data center configurations" on page 632
Use this information to troubleshoot multiple data center configurations, including linking between catalog service domains.

"Configuring the Hibernate cache plug-in" on page 413
You can enable the cache to use the Hibernate cache plug-in by specifying properties files.

Related reference:

"Example: OpenJPA ObjectGrid XML files" on page 410
OpenJPA ObjectGrid XML files should be created based on the configuration of the persistence unit.

"Example: Hibernate ObjectGrid XML files" on page 417
Create Hibernate ObjectGrid XML files based on the configuration of a persistence unit.

Related information:

com.ibm.websphere.objectgrid.openJPA package

com.ibm.websphere.objectgrid.hibernate.cache package

Configuring the OpenJPA cache plug-in

You can configure both DataCache and QueryCache implementations for OpenJPA.

Before you begin

- You must determine the JPA cache plug-in topology that you want to use. See "JPA level 2 (L2) cache plug-in" on page 401 for more information about the different configurations and the properties to set for each topology.
- You must have an application that uses the JPA APIs. If you want to use WebSphere eXtreme Scale APIs to access data with JPA, use the JPA loader. For more information, see "Configuring JPA loaders" on page 419.

Procedure

1. Set properties in your `persistence.xml` file to configure the OpenJPA cache plug-in: You can set these properties on either the DataCache or Query cache implementation.

   DataCache and QueryCache configurations are independent of one another. You can enable either configuration. However, if both configurations are enabled, the QueryCache configuration uses the same configuration as the DataCache configuration, and its configuration properties are discarded.

   ```
   <property name="openjpa.DataCache"
            value="<object_grid_datacache_class(<property>=<value>,...)"/>
   ```

   or

   ```
   <property name="openjpa.QueryCache"
            value="<object_grid_querycache_class(<property>=<value>,...)"/>
   ```

   Note: You can enable the QueryCache configuration for embedded and embedded-intradomain topologies only.

You can specify the ObjectGridName property, the ObjectGridType property, and other simple deployment policy-related properties in the property list of the ObjectGrid cache class to customize cache configuration. An example follows:

```
<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"/>
```

See 7.1.1 * for a list of the properties that you can set.

2. In the `persistence.xml` file, you also must set the openjpa.RemoteCommitProvider property to `sjvm`.

   ```
   <property name="openjpa.RemoteCommitProvider" value="sjvm"/>
   ```

3. Optional: To further customize the data grid used by the cache, you can provide additional settings with XML files.

   For most scenarios, setting cache properties should be sufficient. To further customize the ObjectGrid used by the cache, you can provide OpenJPA ObjectGrid configuration XML files in the `META-INF` directory similarly to the `persistence.xml` file. During initialization, the cache tries to locate these XML files and process them if found.

   There are three types of OpenJPA ObjectGrid configuration XML files:

   - `openjpa-objectGrid.xml` (ObjectGrid configuration)

     File path: `META-INF/openjpa-objectGrid.xml`

     This file is used to customize ObjectGrid configuration for both the EMBEDDED and EMBEDDED_PARTITION type. With the REMOTE type, this file is ignored. By default, each entity class is mapped to its own BackingMap configuration named as an entity class name within the ObjectGrid configuration. For example, com.mycompany.Employee entity class is mapped to com.mycompany.Employee BackingMap. The default BackingMap configuration is `readOnly="false"`, `copyKey="false"`, `lockStrategy="NONE"`, and `copyMode="NO_COPY"`. You can customize some BackingMaps with your chosen configuration. You can use the `ALL_ENTITY_MAPS` reserved keyword to represent all maps excluding other customized maps listed in the `openjpa-objectGrid.xml` file. BackingMaps that are not listed in this `openjpa-objectGrid.xml` file use the default configuration. If customized BackingMaps do not specify the BackingMaps attribute or properties and these attributes are specified in the default configuration, the attribute values from the default configuration are applied. For example, if an entity class is annotated with `timeToLive=30`, the default BackingMap configuration for that entity has a `timeToLive=30`. If the custom `openjpa-objectGrid.xml` file also includes that BackingMap but does not specify timeToLive value, then the customize BackingMap has a `timeToLive=30` value by default. The `openjpa-objectGrid.xml` file intends to override or extend the default configuration.

   - `openjpa-objectGridDeployment.xml` (deployment policy)

     File path: `META-INF/openjpa-objectGridDeployment.xml`

     This file is used to customize deployment policy. When you are customizing deployment policy, if the `openjpa-objectGridDeployment.xml` file is provided, the default deployment policy is discarded. All deployment policy attribute values are from the provided `openjpa-objectGridDeployment.xml` file.

- `openjpa-objectGrid-client-override.xml` (client ObjectGrid override configuration)

  File path: `META-INF/openjpa-objectGrid-client-override.xml`

  This file is used to customize a client-side ObjectGrid. By default, the ObjectGrid cache applies a default client override ObjectGrid configuration that disables a near cache. You can enable the near cache by providing the `openjpa-objectGrid-client-override.xml` file that overrides this configuration. For more information about the settings to change in this file to enable near cache, see "Configuring the near cache" on page 353. The way that the `openjpa-objectGrid-client-override.xml` file works is similar to the `openjpa-objectGrid.xml` file. It overrides or extends the default client ObjectGrid override configuration.

Depending on the configured eXtreme Scale topology, you can provide any one of these three XML files to customize that topology.

For both the EMBEDDED and EMBEDDED_PARTITION types, you can provide any one of the three XML files to customize the ObjectGrid, deployment policy, and client ObjectGrid override configuration.

For a REMOTE ObjectGrid, the ObjectGrid cache does not create a dynamic ObjectGrid. Instead, the cache only obtains a client-side ObjectGrid from the catalog service. You can only provide the `openjpa-objectGrid-client-override.xml` file to customize the client ObjectGrid override configuration.

4. Optional: (Remote configurations only) Set up an external eXtreme Scale system if you want to configure a cache with a REMOTE ObjectGrid type.

   You must set up an external eXtreme Scale system if you want to configure a cache with a REMOTE ObjectGrid type. You need both ObjectGrid and ObjectGridDeployment configuration XML files that are based on the `persistence.xml` file to set up an external system. For examples of these configuration files, see "Example: OpenJPA ObjectGrid XML files" on page 410.

Results

EMBEDDED, EMBEDDED_PARTITION **7.1.1+** **, or intra-domain configuration**:

When an application starts, the plug-in automatically detects or starts a catalog service, starts a container server, and connect the container servers to the catalog service. The plug-in then communicates with the ObjectGrid container and its peers that are running in other application server processes using the client connection.

REMOTE configuration:

The deployment policy is specified separately from the JPA application. An external ObjectGrid system has both catalog service and container server processes. You must start a catalog service before starting container servers. See "Starting stand-alone servers" on page 460 and "Starting container servers" on page 463 for more information.

What to do next

- Develop an OpenJPA application that uses the configuration. For more information, see Example: Using the Hibernate plug-in to preload data into the ObjectGrid cache.

- In a production environment, create catalog service domains for your automatically created processes for your EMBEDDED or EMBEDDED_PARTITION configuration.
  - Stand-alone environment:

    If you are not running your servers inside a WebSphere Application Server process, the catalog service domain hosts and ports are specified using properties file named `objectGridServer.properties`. This file must be stored in the class path of the application and have the **catalogServiceEndPoints** property defined. The catalog service domain is started independently from the application processes and must be started before the application processes are started.

    The format of the `objectGridServer.properties` file follows:

    ```
    catalogServiceEndPoints=<hostname1>:<port1>,<hostname2>:<port2>
    ```
  - WebSphere Application Server environment:

    If you are running inside a WebSphere Application Server process, the JPA cache plug-in automatically connects to the catalog service or catalog service domain that is defined for the WebSphere Application Server cell.
- When you are using the `EMBEDDED` or `EMBEDDED_PARTITION` ObjectGridType value in a Java SE environment, use the System.exit(0) method at the end of the program to stop the embedded eXtreme Scale server. Otherwise, the program can become unresponsive.

Related concepts:

"JPA level 2 (L2) cache plug-in" on page 401
WebSphere eXtreme Scale includes level 2 (L2) cache plug-ins for both OpenJPA and Hibernate Java Persistence API (JPA) providers. When you use one of these plug-ins, your application uses the JPA API. A data grid is introduced between the application and the database, improving response times.

Related reference:

"Example: OpenJPA ObjectGrid XML files"
OpenJPA ObjectGrid XML files should be created based on the configuration of the persistence unit.

Related information:

com.ibm.websphere.objectgrid.openJPA package

Example: OpenJPA ObjectGrid XML files:

OpenJPA ObjectGrid XML files should be created based on the configuration of the persistence unit.

`persistence.xml` file

A `persistence.xml` file that is an example that represents the configuration of a persistence unit follows:

```
<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 file

The openjpa-objectGrid.xml file is used to customize ObjectGrid configuration for both the EMBEDDED and EMBEDDED_PARTITION type. The openjpa-objectGrid.xml file that matches the persistence.xml file follows:

```
<?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>
```

Important:

1. Each entity is mapped to a BackingMap named as the fully qualified entity class name.

   By default, entities are part of the second level cache. In the Entity classes which needs to be excluded from caching, You can include the `@DataCache(enabled=false)` annotation on the entity class that you want to exclude from L2 cache:

   ```
   import org.apache.openjpa.persistence.DataCache;
   @Entity
   @DataCache(enabled=false)
   public class OpenJPACacheTest { ... }
   ```
2. If entity classes are in an inheritance hierarchy, child classes map to the parent BackingMap. The inheritance hierarchy shares a single BackingMap.
3. The ObjectGridQueryCache map is required to support QueryCache.
4. The backingMapPluginCollection for each entity map must have the ObjectTransformer using the com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer class.
5. The backingMapPluginCollection for ObjectGridQueryCache map must have the key index named as QueryCacheKeyIndex as shown in the sample.
6. The evictor is optional for each map.

openjpa-objectGridDeployment.xml file

The `openjpa-objectGridDeployment.xml` file is used to customize deployment policy. The `openjpa-objectGridDeployment.xml` file that matches the `persistence.xml` file follows:

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>
```

Note: The ObjectGridQueryCache map is required to support QueryCache.

Related concepts:

"JPA level 2 (L2) cache plug-in" on page 401
WebSphere eXtreme Scale includes level 2 (L2) cache plug-ins for both OpenJPA and Hibernate Java Persistence API (JPA) providers. When you use one of these plug-ins, your application uses the JPA API. A data grid is introduced between the application and the database, improving response times.

Related tasks:

"Configuring the OpenJPA cache plug-in" on page 407
You can configure both DataCache and QueryCache implementations for OpenJPA.

"Troubleshooting multiple data center configurations" on page 632
Use this information to troubleshoot multiple data center configurations, including linking between catalog service domains.

Related information:

com.ibm.websphere.objectgrid.openJPA package

Configuring the Hibernate cache plug-in

You can enable the cache to use the Hibernate cache plug-in by specifying properties files.

Before you begin

- You must determine the JPA cache plug-in topology that you want to use. See "JPA level 2 (L2) cache plug-in" on page 401 for more information about the different configurations.
- You must have an application that uses the JPA APIs. If you want to use WebSphere eXtreme Scale APIs to access data with JPA, use the JPA loader. For more information, see "Configuring JPA loaders" on page 419.

Procedure

1. If you are using WebSphere Application Server, place the Java Archive (JAR) files in the appropriate locations for your configuration. 7.1.1

   The Hibernate cache plug-in is packaged in the `oghibernate-cache.jar` file and is installed in the *was_root*/`optionalLibraries/ObjectGrid` directory. To use the Hibernate cache plug-in, you have to include the `oghibernate-cache.jar` file in the Hibernate library. For example, if you include the Hibernate library in your application, also must include the `oghibernate-cache.jar` file. If you define a shared library to include Hibernate library, you must add the `oghibernate-cache.jar` file into the shared library directory.

   eXtreme Scale does not install the `cglib.jar` file in the WebSphere Application Server environment. If you have existing applications or shared libraries, such as hibernate, which depend on the `cglib.jar`, locate the `cglib.jar` file and include it in the classpath. For example, if your application includes all hibernate library JAR files, but excludes the `cglib.jar` file available with hibernate, you must include the `cglib.jar` file that comes from Hibernate in your application.

2. Set properties in your `persistence.xml` file to configure the Hibernate cache plug-in

   The syntax for setting properties in the `persistence.xml` file follows:

```
<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`** : The value of the **`provider_class`** property is the com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider class.
   - **`hibernate.cache.use_query_cache`**: To enable query cache, set the value to `true` on the **`use_query_cache`** property.

     Note: You can enable the query cache for embedded and embedded-intradomain topologies only.
   - **`objectgrid.configuration`**: Use the objectgrid.configuration property to specify eXtreme Scale cache configuration properties, including the ObjectGridType attribute that specifies how to place the shards on the data grid.

     You must specify a unique ObjectGridName property value to avoid potential naming conflicts. The other eXtreme Scale cache configuration properties are optional.

     To enable write-behind caching, use the following write-behind attributes on the objectgrid.configuration property. When write-behind caching is enabled, updates are temporarily stored in a JVM scope data storage until either the writeBehindInterval or writeBehindMaxBatchSize conditions are met, when the data is flushed to the cache.

```
writeBehind=true, writeBehindInterval=5000, writeBehindPoolSize=10, writeBehindMaxBatchSize=1000
```

     Attention: Unless writeBehind is enabled, the other write behind configuration settings are disregarded.

     For more information about the values that you can set in the **`objectgrid.configuration`** property, see *.
   - **`objectgrid.hibernate.regionNames`**: The objectgrid.hibernate.regionNames property is optional and should be specified when the regionNames values are defined after the eXtreme Scale cache is initialized. Consider the example of an entity class that is mapped to a regionName with the entity class unspecified in the `persistence.xml` file or not included in the Hibernate mapping file. Further, say it does have Entity annotation. Then, the regionName for this entity class is resolved at class loading time when the eXtreme Scale cache is initialized. Another example is the Query.setCacheRegion(String regionName) method that runs after the eXtreme Scale cache initialization. In these situations, include all possible dynamic determined regionNames in the objectgrid.hibernate.regionNames property so that the eXtreme Scale cache can prepare BackingMaps for all regionNames.

3. Optional: To further customize the data grid used by the cache, you can provide additional settings with XML files.

   For most scenarios, setting cache properties should be sufficient. To further customize the ObjectGrid used by the cache, you can provide Hibernate

ObjectGrid configuration XML files in the `META-INF` directory similarly to the `persistence.xml` file. During initialization, the cache tries to locate these XML files and process them if found.

There are three types of Hibernate ObjectGrid configuration XML files:

- `hibernate-objectGrid.xml` (ObjectGrid configuration)

  File path: `META-INF/hibernate-objectGrid.xml`

  By default, each entity class has an associated regionName (default to entity class name) that is mapped to a BackingMap configuration named as regionName within the ObjectGrid configuration. For example, the com.mycompany.Employee entity class has an associated regionName default to com.mycompany.Employee BackingMap. The default BackingMap configuration is readOnly="false", copyKey="false", lockStrategy="NONE", and `copyMode="NO_COPY"`. You can customize some BackingMaps with a chosen configuration. The reserved key word "ALL_ENTITY_MAPS" can be used to represent all maps excluding other customized maps listed in the `hibernate-objectGrid.xml`file. BackingMaps that are not listed in this `hibernate-objectGrid.xml` file use the default configuration.

- `hibernate-objectGridDeployment.xml` (deployment policy)

  File path: `META-INF/hibernate-objectGridDeployment.xml`

  This file is used to customize deployment policy. When you are customizing deployment policy, if the `hibernate-objectGridDeployment.xml` is provided, the default deployment policy is discarded. All deployment policy attribute values will come from the provided `hibernate-objectGridDeployment.xml` file.

- `hibernate-objectGrid-client-override.xml` (client ObjectGrid override configuration)

  File path: `META-INF/hibernate-objectGrid-client-override.xml`

  This file is used to customize a client-side ObjectGrid. By default, the ObjectGrid cache applies a default client override configuration that disables the near cache. You can enable the near cache by providing the `hibernate-objectGrid-client-override.xml` file that overrides this configuration. For more information about the settings to change in this file to enable near cache, see "Configuring the near cache" on page 353. The way that the `hibernate-objectGrid-client-override.xml` file works is similar to the `hibernate-objectGrid.xml` file. It overrides or extends the default client ObjectGrid override configuration.

Depending on the configured eXtreme Scale topology, you can provide any one of these three XML files to customize that topology.

For both the EMBEDDED and EMBEDDED_PARTITION type, you can provide any one of the three XML files to customize the ObjectGrid, deployment policy, and client ObjectGrid override configuration.

For a REMOTE ObjectGrid, the cache does not create a dynamic ObjectGrid. The cache only obtains a client-side ObjectGrid from the catalog service. You can only provide a `hibernate-objectGrid-client-override.xml` file to customize client ObjectGrid override configuration.

4. Optional: (Remote configurations only) Set up an external eXtreme Scale system if you want to configure a cache with a REMOTE ObjectGrid type. You also need to specify the libraries and their dependencies in the classpath for the eXtreme Scale container servers.

   You must set up an external eXtreme Scale system if you want to configure a cache with a REMOTE ObjectGrid type. You need both ObjectGrid and ObjectGridDeployment configuration XML files that are based on the persistence.xml file to set up an external system. For examples of these configuration files, see"Example: Hibernate ObjectGrid XML files" on page 417.

Results

EMBEDDED or EMBEDDED_PARTITION configuration:

When an application starts, the plug-in automatically detects or starts a catalog service, starts a container server, and connect the container servers to the catalog service. The plug-in then communicates with the ObjectGrid container and its peers that are running in other application server processes using the client connection.

Each JPA entity has an independent backing map assigned using the class name of the entity. Each BackingMap has the following attributes.

- `readOnly="false"`
- `copyKey="false"`
- `lockStrategy="NONE"`
- `copyMode="NO_COPY"`

REMOTE configuration:

The deployment policy is specified separately from the JPA application. An external ObjectGrid system has both catalog service and container server processes. You must start a catalog service before starting container servers. See "Starting stand-alone servers" on page 460 and "Starting container servers" on page 463 for more information.

What to do next

- Develop a Hibernate application that uses the configuration. For more information, see Example: Using the Hibernate plug-in to preload data into the ObjectGrid cache.
- In a production environment, create catalog service domains for your automatically created processes for your EMBEDDED or EMBEDDED_PARTITION configuration.
  - Stand-alone environment:

    If you are not running your servers inside a WebSphere Application Server process, the catalog service domain hosts and ports are specified using properties file named `objectGridServer.properties`. This file must be stored in the class path of the application and have the **catalogServiceEndPoints** property defined. The catalog service domain is started independently from the application processes and must be started before the application processes are started.

    The format of the `objectGridServer.properties` file follows:

    ```
    catalogServiceEndPoints=<hostname1>:<port1>,<hostname2>:<port2>
    ```
  - WebSphere Application Server environment:

    If you are running inside a WebSphere Application Server process, the JPA cache plug-in automatically connects to the catalog service or catalog service domain that is defined for the WebSphere Application Server cell. However, using an `objectGridServer.properties` file with defined `catalogServiceEndPoints` will cause problems because it will try to establish a connection to that catalog server instead of the one in WebSphere Application Server.

- When you are using the `EMBEDDED` or `EMBEDDED_PARTITION` ObjectGridType value in a Java SE environment, use the System.exit(0) method at the end of the program to stop the embedded eXtreme Scale server. Otherwise, the program can become unresponsive.

Related concepts:

"JPA level 2 (L2) cache plug-in" on page 401
WebSphere eXtreme Scale includes level 2 (L2) cache plug-ins for both OpenJPA and Hibernate Java Persistence API (JPA) providers. When you use one of these plug-ins, your application uses the JPA API. A data grid is introduced between the application and the database, improving response times.

Related reference:

"Example: Hibernate ObjectGrid XML files"
Create Hibernate ObjectGrid XML files based on the configuration of a persistence unit.

"Example: Hibernate ObjectGrid XML files"
Create Hibernate ObjectGrid XML files based on the configuration of a persistence unit.

Related information:

com.ibm.websphere.objectgrid.hibernate.cache package

Example: Hibernate ObjectGrid XML files:

Create Hibernate ObjectGrid XML files based on the configuration of a persistence unit.

`persistence.xml` file

An example `persistence.xml` file that represents the configuration of a persistence unit follows:

```
<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 file

Use the hibernate-objectGridDeployment.xml file to optionally customize the deployment policy. If you provide this file in the META-INF/hibernate-objectGridDeployment.xml directory, the default deployment policy is overridden by the configuration in this file.

```
<?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 file

If you are not using Hibernate with the Java Persistence API (JPA), use the following example hibernate-objectGrid.xml to create your Hibernate configuration:

```
<?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>
```

Note: The org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache and defaultCacheMap maps are required.

Related concepts:

"JPA level 2 (L2) cache plug-in" on page 401
WebSphere eXtreme Scale includes level 2 (L2) cache plug-ins for both OpenJPA and Hibernate Java Persistence API (JPA) providers. When you use one of these plug-ins, your application uses the JPA API. A data grid is introduced between the application and the database, improving response times.

Related tasks:

"Configuring the Hibernate cache plug-in" on page 413
You can enable the cache to use the Hibernate cache plug-in by specifying properties files.

"Troubleshooting multiple data center configurations" on page 632
Use this information to troubleshoot multiple data center configurations, including linking between catalog service domains.

"Configuring the Hibernate cache plug-in" on page 413
You can enable the cache to use the Hibernate cache plug-in by specifying properties files.

Related information:

com.ibm.websphere.objectgrid.hibernate.cache package

Configuring database integration

You can use WebSphere eXtreme Scale to lower the load on databases. You can use a Java Persistence API (JPA) between WebSphere eXtreme Scale and the database to integrate changes as a loader.

Before you begin

For a summary of the various topologies that you can create with a database, see "Database integration: Write-behind, in-line, and side caching" on page 21.

Configuring JPA loaders

A Java Persistence API (JPA) Loader is a plug-in implementation that uses JPA to interact with the database.

Before you begin

- You must have a JPA implementation, such as Hibernate or OpenJPA.
- Your database can be any back end that is supported by the chosen JPA provider.
- Decide whether you are going to use the JPALoader plug-in or the JPAEntityLoader plug-in. Use the JPALoader plug-in when you are storing data using the ObjectMap API. Use the JPAEntityLoader plug-in when you are storing data using the EntityManager API.

  Note: If you are using the JPA APIs to access the JPA data source, use the JPA L2 cache plug-in. The cache plug-in introduces the data grid between your application and the JPA data source, while still using a JPA application. For more information, see "JPA level 2 (L2) cache plug-in" on page 401.

About this task

For more information about how the Java Persistence API (JPA) Loader works, see JPA Loaders.

Procedure

1. Configure the necessary parameters that JPA requires to interact with a database.

   The following parameters are required. These parameters are configured in the JPALoader or JPAEntityLoader bean, and JPATxCallback bean.

   - **`persistenceUnitName`**: Specifies the persistence unit name. This parameter is required for two purposes: for creating a JPA EntityManagerFactory, and for locating the JPA entity metadata in the `persistence.xml` file. This attribute is set on the JPATxCallback bean.
   - **`JPAPropertyFactory`**: Specifies the factory to create a persistence property map to override the default persistence properties. This attribute is set on the JPATxCallback bean. To set this attribute, Spring style configuration is required.
   - **`entityClassName`**: Specifies the entity class name that is required to use JPA methods, such as EntityManager.persist, EntityManager.find, and so on. The JPALoader plug-in requires this parameter, but the parameter is optional for JPAEntityLoader. For the JPAEntityLoader plug-in, if an **`entityClassName`** parameter is not configured, the entity class configured in the ObjectGrid entity map is used. You must use the same class for the eXtreme Scale EntityManager and for the JPA provider. This attribute is set on the JPALoader or JPAEntityLoader bean.
   - **`preloadPartition`**: Specifies the partition at which the map preload is started. If the preload partition is less than zero, or greater than the total number of partitions minus 1, the map preload is not started. The default value is -1, which means the preload does not start by default. This attribute is set on the JPALoader or JPAEntityLoader bean.

   Other than the four JPA parameters to be configured in eXtreme Scale, JPA metadata are used to retrieve the key from the JPA entities. The JPA metadata can be configured as annotation, or as an `orm.xml` file specified in the `persistence.xml` file. It is not part of the eXtreme Scale configuration.
2. Configure XML files for the JPA configuration.

   To configure a JPALoader or JPAEntityLoader, see Plug-ins for communicating with databases.

Configure a JPATxCallback transaction callback along with the loader configuration. The following example is an ObjectGrid XML descriptor file (`objectgrid.xml`), that has a JPAEntityLoader and JPATxCallback configured:

configuring a loader including callback - 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="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>
```

If you want to configure a JPAPropertyFactory, you have to use a Spring style configuration. The following is an XML configuration file sample,`JPAEM_spring.xml` which configures a Spring bean to be used for eXtreme Scale configurations.

configuring a loader including JPA property factory - XML example
```
<?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>
```

The `Objectgrid.xml` configuration XML file follows. Notice the ObjectGrid name is `JPAEM`, which matches the ObjectGrid name in the `JPAEM_spring.xml` Spring configuration file.

JPAEM loader configuration - 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="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>
```

An entity can be annotated with both the JPA annotations and eXtreme Scale entity manager annotations. Each annotation has an XML equivalent that can be used. Thus, eXtreme Scale added the Spring namespace. You can also configure these using the Spring namespace support. For more information, see Spring framework overview.

Related concepts:

Programming for JPA integration
The Java Persistence API (JPA) is a specification that allows mapping Java objects to relational databases. JPA contains a full object-relational mapping (ORM) specification using Java language metadata annotations, XML descriptors, or both to define the mapping between Java objects and a relational database. A number of open-source and commercial implementations are available.

“Configuring cache integration” on page 364
WebSphere eXtreme Scale can integrate with other caching-related products. You can also use the WebSphere eXtreme Scale dynamic cache provider to plug WebSphere eXtreme Scale into the dynamic cache component in WebSphere Application Server. Another extension to WebSphere Application Server is the WebSphere eXtreme Scale HTTP session manager, which can help to cache HTTP sessions.

Configuring a JPA time-based data updater

You can configure a time-based database update using XML for a local or distributed eXtreme Scale configuration. You can also configure a local configuration programmatically.

About this task

For more information about how the Java Persistence API (JPA) time-based data updater works, see JPA time-based data updater.

Procedure

Create a timeBasedDBUpdate configuration.

- **With an XML file:**

  The following example shows an `objectgrid.xml` file that contains a timeBasedDBUpdate configuration:

  JPA time-based updater - 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="changeOG"
            entityMetadataXMLFile="userEMD.xml">
            <backingMap name="user" >
                <timeBasedDBUpdate timestampField="rowChgTs"
                      persistenceUnitName="userderby"
                      entityClass="com.test.UserClass"
                      mode="INVALIDATE_ONLY"
               />
            </backingMap>
        </objectGrid>
    </objectGrids>
    <backingMapPluginCollections>
</objectGridConfig>
```

In this example, the map "user" is configured with time-based database update. The database update mode is INVALIDATE_ONLY, and the timestamp field is rowChgTs.

When the distributed ObjectGrid "changeOG" is started in the container server, a time-based database update thread is automatically started in partition 0.

- **Programmatically:**

  If you create a local ObjectGrid, you can also create a TimeBasedDBUpdateConfig object and set it on the BackingMap instance:

  ```
  public void setTimeBasedDBUpdateConfig(TimeBasedDBUpdateConfig dbUpdateConfig);
  ```

  For more information about setting an object on the BackingMap instance, see the information about the BackingMap interface in the API documentation.

  Alternatively, you can annotate the timestamp field in the entity class using the com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp annotation. By configuring the value in the class, you do not have to configure the timestampField in the XML configuration.

What to do next

Start the JPA time-based data updater. See Starting the JPA time-based updater for more information.

Configuring REST data services

You can use WebSphere eXtreme Scale REST data service with WebSphere Application Server version 7.0,WebSphere Application Server Community Edition and Apache Tomcat.

About this task

The included sample has source code and compiled binaries to run a partitioned data grid. This sample demonstrates how to create a simple data grid, model the data using entities and provides two command-line client applications that allow adding and querying entities using Java or C#.

The sample Java client uses the Java EntityManager API to persist and query data in the data grid. This client can be run in Eclipse or using a command-line script. Note that the sample Java client does not demonstrate the REST data service, but allows updating data in the grid, so a web browser or other clients can read the data.

The sample Microsoft WCF Data Services C# client communicates with the eXtreme Scale data grid through the REST data service using the .NET framework. The WCF Data Services client can be used to both update and query the data grid.

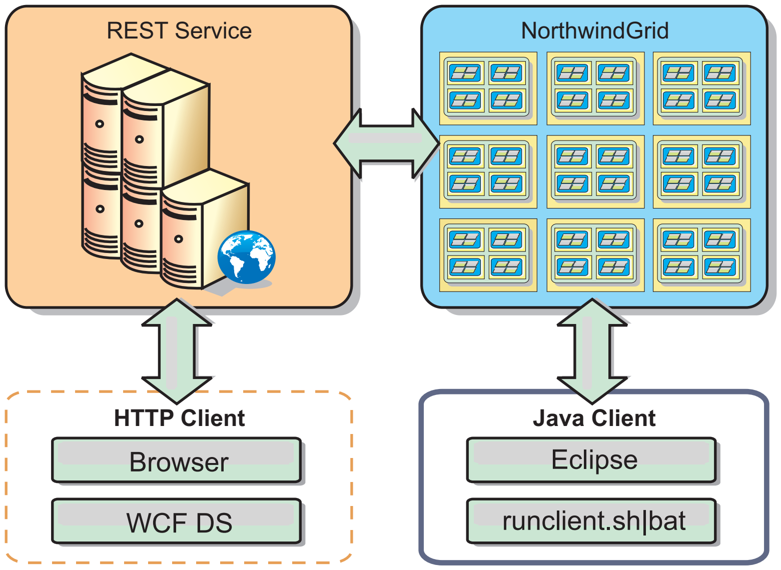


Figure 44. Getting started sample topology. HTTP clients using the REST data service and Java clients can access the same data grid.

Procedure

1. Configure and start the eXtreme Scale data grid. See "Enabling the REST data service."
2. Configure and start the REST data service in a web server. See "Configuring application servers for the REST data service" on page 432.
3. Run a client to interact with the REST data service. Two options are available:
   a. Run the sample Java client to populate the grid with data using the EntityManager API and query the data in the grid using a web browser and the eXtreme Scale REST data service. See "Using a Java client with REST data services" on page 448.
   b. Run the sample WCF Data Services C# client. See "Visual Studio 2008 WCF client with REST data service" on page 450.

Enabling the REST data service

The REST data service can represent WebSphere eXtreme Scale entity metadata to represent each entity as an EntitySet.

Starting a sample eXtreme Scale data grid

In general , before starting the REST data service, start the eXtreme Scale data grid. The following steps will start a single eXtreme Scale catalog service process and two container server processes.

WebSphere eXtreme Scale can be installed using three different methods:

- Trial install
- Stand-alone deployment
- WebSphere Application Server integrated deployment

Scalable data model in eXtreme Scale

The Microsoft Northwind sample uses the Order Detail table to establish a many-to-many association between Orders and Products.

Object to relational mapping specifications (ORMs) such as the ADO.NET Entity Framework and Java Persistence API (JPA) can map the tables and relationships using entities. However, this architecture does not scale. Everything must be located on the same machine, or an expensive cluster of machines to perform well.

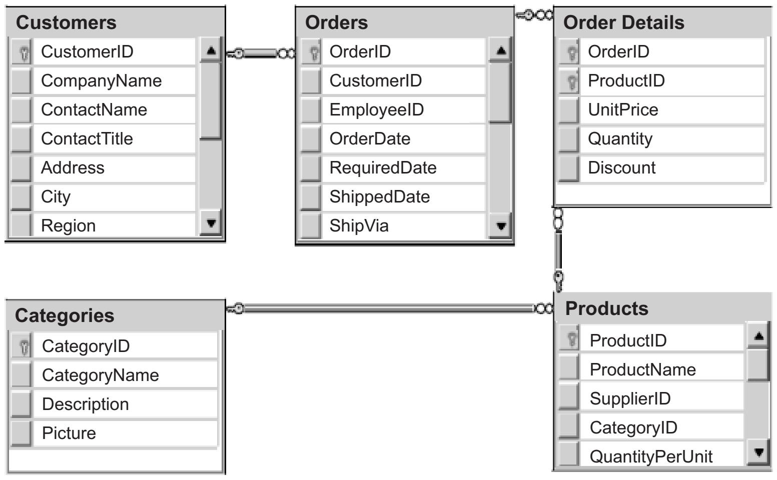


Figure 45. Microsoft SQL Server Northwind sample schema diagram

To create a scalable version of the sample, the entities must be modeled so each entity or group of related entities can be partitioned based off a single key. By creating partitions on a single key, requests can be spread out among multiple, independent servers. To achieve this configuration, the entities have been divided into two trees: the Customer and Order tree and the Product and Category tree. In this model, each tree can be partitioned independently and therefore can grow at different rates, increasing scalability.

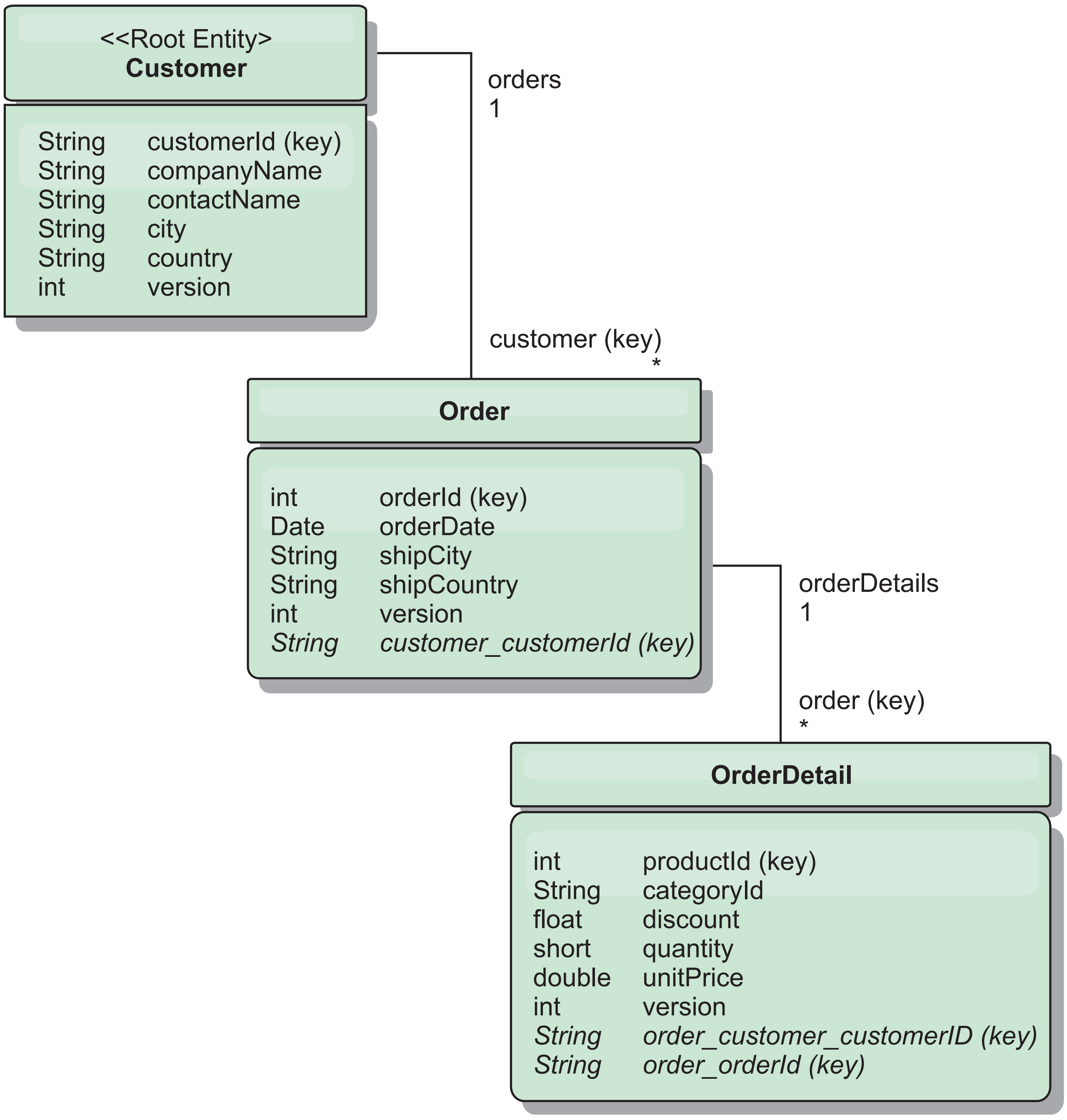


Figure 46. Customer and Order entity schema diagram

For example, both Order and Product have unique, separate integers as keys. In fact, the Order and Product tables are really independent of each other. For example, consider the effect of the size of a catalog, the number of products you sell, with the total number of orders. Intuitively, it might seem that having many products implies also having many orders, but this is not necessarily the case. If this were true, you could easily increase sales by just adding more products to your catalog. Orders and products have their own independent tables. You can further extend this concept so that orders and products each have their own separate, data grids. With independent data grids, you can control the number of partitions and servers, in addition to the size of each data grid separately so that your application can scale. If you double the size of your catalog, you must double the products data grid, but the order grid might be unchanged. The converse is true for an order surge, or expected order surge.

In the schema, a Customer has zero or more Orders, and an Order has line items (OrderDetail), each with one specific product. A Product is identified by ID (the Product key) in each OrderDetail. A single data grid stores Customers, Orders, and OrderDetails with Customer as the root entity of the data grid. You can retrieve Customers by ID, but you must get Orders starting with the Customer ID. So customer ID is added to Order as part of its key. Likewise, the customer ID and order ID are part of the OrderDetail ID.

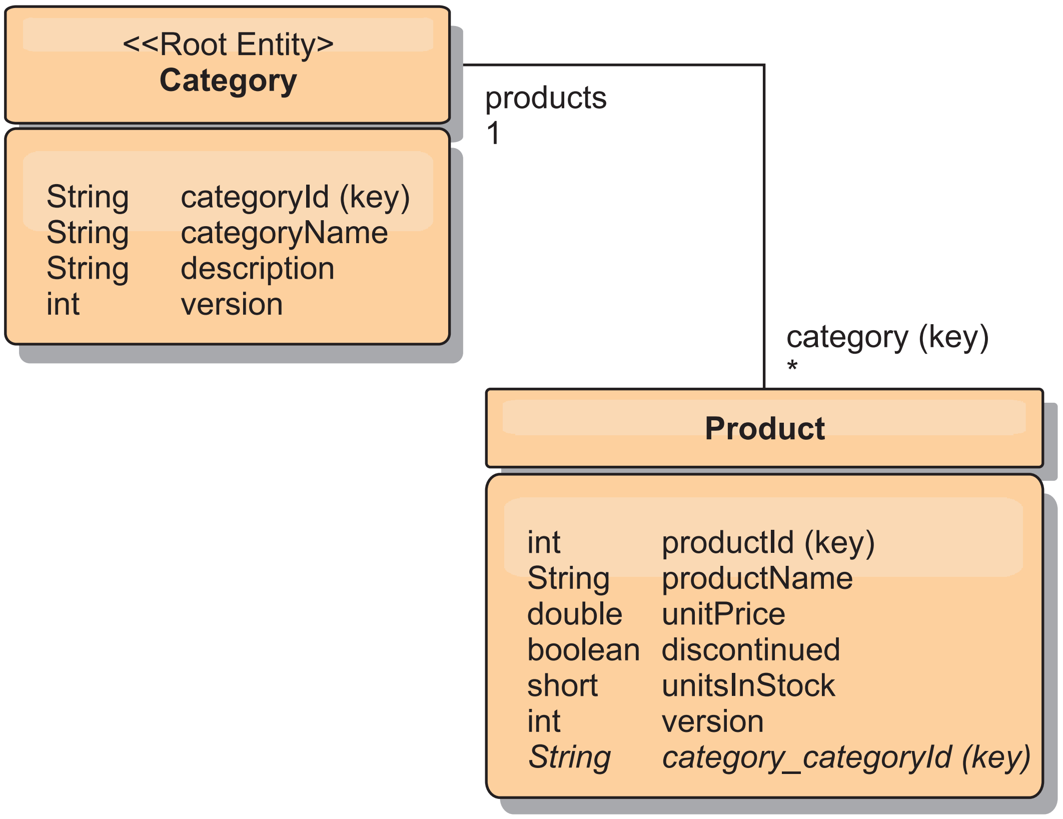


Figure 47. Category and Product entity schema diagram

In the Category and Product schema, the Category is the schema root. With this schema, customers can query products by category. See "Retrieving and updating data with REST" for additional details on key associations and their importance.

Retrieving and updating data with REST

The OData protocol requires that all entities can be addressed by their canonical form. This means that each entity must include the key of the partitioned, root entity, the schema root.

The following is an example of how to use the association from a root entity to address a child in :

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

In WCF Data Services, the child entity must be directly addressable, meaning that the key in the schema root must be a part of the key of the child: `/Order(customer_customerId='ACME', orderId=100)`. This is achieved by creating an association to the root entity where the one-to-one or many-to-one association to the root entity is also identified as a key. When entities are included as part of the key, the attributes of the parent entity are exposed as key properties.

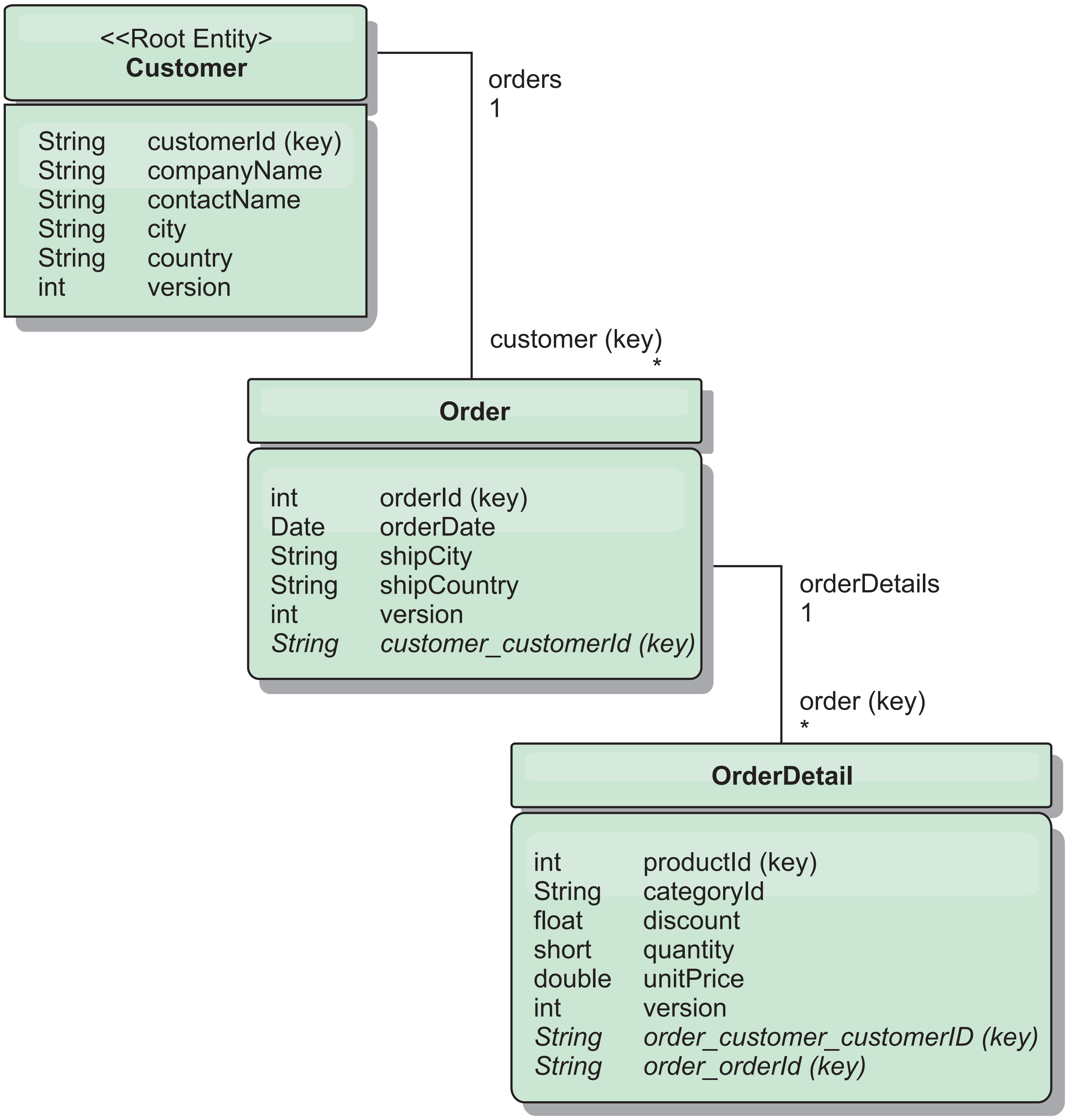


Figure 48. Customer and Order entity schema diagram

The Customer/Order entity schema diagram illustrates how each entity is partitioned using the Customer. The Order entity includes the Customer as part of its key and is therefore directly accessible. The REST data service exposes all key associations as individual properties: Order has customer_customerId and OrderDetail has order_customer_customerId and order_orderId.

Using the EntityManager API, you can find the Order using the Customer and order id:

```
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();
```

When using the REST data service, the Order can be retrieved with either of the URLs:
- `/Order(orderId=100, customer_customerId='ACME')`
- `/Customer('ACME')/orders?$filter=orderId eq 100`

The customer key is addressed using the attribute name of the Customer entity, an underscore character and the attribute name of the customer id: customer_customerId.

An entity can also include a non-root entity as part of its key if all of the ancestors to the non-root entity have key associations to the root. In this example, OrderDetail has a key-association to Order and Order has a key-association to the root Customer entity. Using the EntityManager API:

```
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);
...
```

The REST data service allows addressing the OrderDetail directly:

```
/OrderDetail(productId=500, order_customer_customerId='ACME', order_orderId =100)
```

The association from the OrderDetail entity to the Product entity has been broken to allow partitioning the Orders and Product inventory independently. The OrderDetail entity stores the category and product id instead of a hard relationship. By decoupling the two entity schemas, only one partition is accessed at a time.

The Category and Product schema illustrated in the diagram shows that the root entity is Category and each Product has an association to a Category entity. The Category entity is included in the Product identity. The REST data service exposes a key property: category_categoryId which allows directly addressing the Product.

Because Category is the root entity, in a partitioned environment, the Category must be known in order to find the Product. Using the EntityManager API, the transaction must be pinned to the Category entity prior to finding the Product.

Using the EntityManager API:

```
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));
...
```

The REST data service allows addressing the Product directly:

```
/Product(productId=500, category_categoryId='COMP')
```

Starting a stand-alone data grid for REST data services

Follow these steps to start the WebSphere eXtreme Scale REST service sample data grid for a stand-alone eXtreme Scale deployment.

Before you begin

Install the WebSphere eXtreme Scale Trial or full product:

- Install the stand-alone version of the product and apply any subsequent fixes.
- Download and extract the WebSphere eXtreme Scale Version 7.1 or later trial, which includes the WebSphere eXtreme Scale REST data service.

About this task

Start the WebSphere eXtreme Scale sample data grid.

Procedure

1. Start the catalog service process. Open a command-line or terminal window and set the JAVA_HOME environment variable:
   - Linux UNIX `export JAVA_HOME=`*`java_home`*
   - Windows `set JAVA_HOME=`*`java_home`*
2. `cd restservice_home/gettingstarted`
3. Start the catalog service process. To start the service *without* eXtreme Scale security, use the following commands.
   - Linux UNIX `./runcat.sh`
   - Windows `runcat.bat`

   To start the service witheXtreme Scale security, use the following commands.
   - Linux UNIX `./runcat_secure.sh`
   - Windows `runcat_secure.bat`
4. Start two container server processes. Open another command-line or terminal window and set the JAVA_HOME environment variable:
   - Linux UNIX `export JAVA_HOME=`*`java_home`*
   - Windows `set JAVA_HOME=`*`java_home`*
5. `cd restservice_home/gettingstarted`
6. Start a container server process:

   To start the server without eXtreme Scale security, use the following commands:
   - Linux UNIX `./runcontainer.sh container0`
   - Windows `runcontainer.bat container0`

   To start the server witheXtreme Scale security, use the following commands.
   - Linux UNIX `./runcontainer_secure.sh container0`

- Windows `runcontainer_secure.bat container0`

7. Open another command-line or terminal window and set the JAVA_HOME environment variable:
   - Linux UNIX `export JAVA_HOME=`*`java_home`*
   - Windows `set JAVA_HOME=`*`java_home`*
8. `cd restservice_home/gettingstarted`
9. Start a second container server process.

   To start the server without eXtreme Scale security, use the following commands.
   - Linux UNIX `./runcontainer.sh container1`
   - Windows `runcontainer.bat container1`

   To start the server with eXtreme Scale security, use the following commands.
   - Linux UNIX `./runcontainer_secure.sh container1`
   - Windows `runcontainer_secure.bat container1`

Results

Wait until the eXtreme Scale containers are ready before proceeding with the next steps. The container servers are ready when the following message is displayed in the terminal window:

```
CWOBJ1001I: ObjectGrid Server container_name is ready to process requests.
```

Where *container_name* is the name of the container that was started.

Starting a data grid for REST data services in WebSphere Application Server

Follow these steps to start a stand-alone WebSphere eXtreme Scale REST service sample data grid for a WebSphere eXtreme Scale deployment that is integrated with WebSphere Application Server. Although WebSphere eXtreme Scale is integrated with WebSphere Application Server, these steps start a stand-alone WebSphere eXtreme Scale catalog service process and container.

Before you begin

Install the product into a WebSphere Application Server Version 7.0.0.5 or later installation directory with security disabled. Augment at least one application server profile.

About this task

Start the WebSphere eXtreme Scale sample data grid.

Procedure

1. Start the catalog service process. Open a command-line or terminal window and set the JAVA_HOME environment variable:
   - Linux UNIX `export JAVA_HOME=`*`java_home`*
   - Windows `set JAVA_HOME=`*`java_home`*

   `cd restservice_home/gettingstarted`
2. Start the catalog service process.

To start the server without eXtreme Scale security, use the following commands.

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

To start the server witheXtreme Scale security, use the following commands.

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

3. Start two container server processes. Open another command-line or terminal window and set the JAVA_HOME environment variable:
   - Linux UNIX `export JAVA_HOME=`*`java_home`*
   - Windows `set JAVA_HOME=`*`java_home`*
4. Start a container server process.

   To start the server without eXtreme Scale security, use the following commands.
   a. Open a command-line window.
   b. `cd restservice_home/gettingstarted`
   c. To start the server *without* eXtreme Scale security, use the following commands.
      - Linux UNIX `./runcontainer.sh container0`
      - Windows `runcontainer.bat container0`
   d. To start the server with eXtreme Scale security, use the following commands.
      - Linux UNIX `./runcontainer_secure.sh container0`
      - Windows `runcontainer_secure.bat container0`
5. Start a second container server process.
   a. Open a command-line window.
   b. `cd restservice_home/gettingstarted`
   c. To start the server *without* eXtreme Scale security, use the following commands.
      - Linux UNIX `./runcontainer.sh container1`
      - Windows `runcontainer.bat container1`
   d. To start the server *with* eXtreme Scale security, use the following commands.
      - Linux UNIX `./runcontainer_secure.sh container1`
      - Windows `runcontainer_secure.bat container1`

Results

Wait until the container servers are ready before proceeding with the next steps. The container servers are ready when the following message is displayed:

```
CWOBJ1001I: ObjectGrid Server container_name is ready to process requests.
```

Where *container_name* is the name of the container that was started in the previous step.

Configuring application servers for the REST data service

You can configure various application servers to use the REST data service.

Deploying the REST data service on WebSphere Application Server

This topic describes how to configure the WebSphere eXtreme Scale REST data service on WebSphere Application Server or WebSphere Application Server Network Deployment Version 6.1.0.25 or or later. These instructions also apply to deployments whereWebSphere eXtreme Scale is integrated with the WebSphere Application Server deployment.

Before you begin

You must have one of the following environments on your system to configure and deploy the REST data service for WebSphere eXtreme Scale.

- WebSphere Application Server with the stand-alone WebSphere eXtreme Scale client:
  - The WebSphere eXtreme Scale Trial Version 7.1 with the REST data service is downloaded and extracted or theWebSphere eXtreme Scale Version 7.1.0.0 with cumulative fix 2 product is installed into a stand-alone directory.
  - WebSphere Application Server Version 6.1.0.25 or 7.0.0.5 or later is installed and running.
- WebSphere Application Server integrated with WebSphere eXtreme Scale:

  WebSphere eXtreme Scale Version 7.1.0.0 with cumulative fix 2 or later is installed on top of WebSphere Application Server Version 6.1.0.25 or 7.0 or later.

  Tip: The WebSphere eXtreme Scale REST data service only requires that the WebSphere eXtreme Scale client option be installed. The profile does not need to be augmented.

  Read about how to enable Java 2 security in the WebSphere Application Server information center.

Procedure

1. Configure and start a data grid.
   a. For details on configuring a data grid for use with the REST data service, see “Starting a data grid for REST data services in WebSphere Application Server” on page 431.
   b. Verify that a client can connect to and access entities in the data grid. For an example, see “Tutorial: Getting started with WebSphere eXtreme Scale” on page 1.
2. Build the eXtreme Scale REST service configuration JAR or directory. See the information about packaging and deploying the REST service in “Installing the REST data service” on page 220.
3. Add the REST data service configuration JAR or directory to the application server classpath:
   a. Open the WebSphere Application Server administrative console
   b. Navigate to **Environment** > **Shared libraries**
   c. Click **New**
   d. Add the following entries into the appropriate fields:
      - Name: `extremescale_rest_configuration`
      - Classpath: <REST service configuration jar or directory>
   e. Click **OK**
   f. Save the changes to the master configuration

4. Add the WebSphere eXtreme Scale client runtime JAR, `wsogclient.jar`, and the REST data service configuration JAR or directory to the application server classpath. This step is not necessary if WebSphere eXtreme Scale is integrated with the WebSphere Application Server installation.
   a. Open theWebSphere Application Server administrative console.
   b. Navigate to **Environment** > **Shared libraries**.
   c. Click **New**.
   d. Add the following entries into the fields:
      - Name: `extremescale_client_v71`
      - Classpath: `wxs_home/lib/wsogclient.jar`

      Remember: Add each path on a separate line.
   e. Click **OK**.
   f. Save the changes to the master configuration.
5. Install the REST data service EAR file, `wxsrestservice.ear`, to the WebSphere Application Server using the administrative console:
   a. Open the WebSphere Application Server administrative console.
   b. Click **Applications** > **New application**.
   c. Browse to the `/lib/wxsrestservice.ear` file on the file system and select it and click **Next**.
      - If using WebSphere Application Server Version 7.0, click Next.
      - If using WebSphere Application Server Version 6.1, enter a Context Root value with the name: `/wxsrestservice` and continue to the next step.
   d. Choose the detailed installation option, and click **Next**.
   e. On the application security warnings screen, click **Continue**.
   f. Choose the default installation options, and click **Next**.
   g. Choose a server to map the application to, and click **Next**.
   h. On the JSP reloading page, use the defaults, and click **Next**.
   i. On the shared libraries page, map the `wxsrestservice.war` module to the shared libraries that you defined:
      - `extremescale_rest_configuration`
      - `extremescale_client_v71`

      Tip: This shared library is required only if WebSphere eXtreme Scale is not integrated with WebSphere Application Server.
   j. On the map shared library relationship page, use the defaults, and click **Next**.
   k. On the map virtual hosts page, use the defaults, and click **Next**.
   l. On the map context roots page, set the context root to: wxsrestservice
   m. On the Summary screen, click **Finish** to complete the installation.
   n. Save the changes to the master configuration.
6. Start the `wxsrestservice` REST data service application:
   a. Go to the application in the administrative console.
      - WebSphere Application Server Version 7.0: In the administrative console, click **Applications** > **Application Types** > **WebSphere Applications**.
      - 7.1.1 WebSphere Application Server Version 6.1: In the administrative console, click **Applications** > **Enterprise Applications**.

b. Check the check box next to the `wxsrestservice` application, and click **Start**.

c. Review the `SystemOut.log` file for the application server profile. When the REST data service has started successfully, the following message is displayed in the `SystemOut.log` file for the server profile:

```
CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been
started.
```

7. Verify the REST data service is working: The port number can be found in the `SystemOut.log` file within the application server profile logs directory by looking at the first port displayed for message identifier: SRVE0250I. The default port is 9080.

   For example:`http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/` Result: The AtomPub service document is displayed.

   For example: `http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/$metadata`. The Entity Model Data Extensions (EDMX) document is displayed.
8. To stop the data grid processes, use `CTRL+C` in the respective command window.

Starting REST data services with WebSphere eXtreme Scale integrated in WebSphere Application Server 7.0:

This topic describes how to configure and start the eXtreme Scale REST data service using WebSphere Application Server version 7.0 that has been integrated and augmented with WebSphere eXtreme Scale.

Before you begin

Verify that the sample stand-alone eXtreme Scale data grid is started. See "Enabling the REST data service" on page 424 for details on how to start the data grid.

About this task

To get started with the WebSphere eXtreme Scale REST data service using WebSphere Application Server, follow these steps:

Procedure

1. Add the WebSphere eXtreme Scale REST data service sample configuration JAR to the classpath:
   a. Open the WebSphere Administration Console
   b. Navigate to Environment -> Shared libraries
   c. Click New
   d. Add the following entries into the appropriate fields:
      1) Name: extremescale_gettingstarted_config
      2) Classpath
         - `restservice_home/gettingstarted/restclient/bin`
         - `restservice_home/gettingstarted/common/bin`

         Remember: Each path must appear on a different line.
   e. Click **OK**
   f. Save the changes to the master configuration
2. Install the REST data service EAR file, wxsrestservice.ear, to the WebSphere Application Server using the WebSphere administration console:

a. Open the WebSphere administration console
b. Navigate to Applications -> New Application
c. Browse to `restservice_home/lib/wxsrestservice.ear` file on the file system. Select the file and click **Next**.
d. Choose the detailed installation options, and click **Next**.
e. On the application security warnings screen, click **Continue**.
f. Choose the default installation options, and click **Next**.
g. Choose a server to map the wxsrestservice.war module to, and click **Next**.
h. On the JSP reloading page, use the defaults, and click **Next**.
i. On the shared libraries page, map the "wxsrestservice.war" module to the following shared libraries that were defined during step 1: `extremescale_ gettingstarted _config`
j. On the map shared library relationship page, use the defaults, and click **Next**.
k. On the map virtual hosts page, use the defaults, and click **Next**.
l. On the map context roots page, set the context root to: wxsrestservice.
m. On the Summary screen, click **Finish** to complete the installation.
n. Save the changes to the master configuration.

3. If the eXtreme Scale data grid was started with eXtreme Scale security enabled, set the following property in the `restservice_home/gettingstarted/restclient/bin/wxsRestService.properties` file.

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties
```

4. Start the application server and the "wxsrestservice " eXtreme Scale REST data service application.

   After the application is started review the SystemOut.log for the application server and verify that the following message appears: `CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been started.`
5. Verify that the REST data service is working:
   a. Open a browser and navigate to:

      http://localhost:9080/wxsrestservice/restservice/NorthwindGrid

      The service document for the NorthwindGrid is displayed.
   b. Navigate to:

      http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/$metadata

      The Entity Model Data Extensions (EDMX) document is displayed
6. To stop the data grid processes, use CTRL+C in the respective command window to stop the process.

Deploying the REST data service on WebSphere Application Server Community Edition

You can configure the eXtreme Scale REST data service on WebSphere Application Server Community Edition Version 2.1.1.3 or later.

Before you begin

- An IBM (recommended) or Oracle JRE or JDK, Version 5 or later is installed and the JAVA_HOME environment variable is set.

- Download and install WebSphere Application Server Community Edition Version 2.1.1.3 or later to the `wasce_root` directory, for example the `/opt/IBM/wasce` directory. Read the installation instructions for information on version 2.1.1 or other versions.

Procedure

1. Configure and start a data grid.
   a. For details on configuring an eXtreme Scale data grid for use with the REST data service, read about "Starting a stand-alone data grid for REST data services" on page 430.
   b. Verify that an eXtreme Scale client can connect to and access entities in the data grid. For an example, see "Tutorial: Getting started with WebSphere eXtreme Scale" on page 1.
2. Build the eXtreme Scale REST service configuration JAR or directory. See the packaging and deployment information in the "Installing the REST data service" on page 220 topic for details.
3. Start the WebSphere Application Server Community Edition server:
   a. To start the server without Java SE security enabled, run the following command:

      UNIX Linux `wasce_root/bin/startup.sh`

      Windows `wasce_root/bin/startup.bat`
   b. To start the server with Java SE security enabled, follow these steps: UNIX Linux
      1) Open a command-line or terminal window and run the following copy command (or copy the contents of the specified policy file into your existing policy): `cp restservice_home/gettingstarted/wasce/ geronimo.policy wasce_root/bin`
      2) Edit the `wasce_root/bin/setenv.sh` file
      3) After the line that contains `"WASCE_JAVA_HOME="`, add the following: `export JAVA_OPTS="-Djava.security.manager -Djava.security.policy=geronimo.policy"`

      Windows
      1) Open a command-line window and run the following copy command or copy the contents of the specified policy file into your existing policy:

         `copy restservice_home\gettingstarted\wasce\geronimo.policy\bin`
      2) Edit the `wasce_root\bin\setenv.bat` file
      3) After the line that contains "set WASCE_JAVA_HOME=", add the following:

         ```
         set JAVA_OPTS="-Djava.security.manager
         -Djava.security.policy=geronimo.policy"
         ```
4. Add the ObjectGrid client runtime JAR to the WebSphere Application Server Community Edition repository:
   a. Open the WebSphere Application Server Community Edition administration console and log in. The default URL is: `http://localhost:8080/console` and the default userid is `system` and password is `manager`.
   b. Click the **Repository** link on the left side of the console window, in the **Services** folder.
   c. In the **Add Archive to Repository** section, fill in the following into the input text boxes:

Table 26. Add Archive to Repository

Text box	**Value**
File	`wxs_home/lib/ogclient.jar`
Group	com.ibm.websphere.xs
Artifact	ogclient
Version	7.1
Type	JAR

   d. Click the Install button

   See the following tech note for details on different ways class and library dependencies can be configured: Specifying external dependencies to applications running on WebSphere Application Server Community Edition.

5. Deploy and start the REST data service module, the `wxsrestservice.war` file, to the WebSphere Application Server Community Edition server.
   a. Copy and edit the sample deployment plan XML file: `restservice_home/gettingstarted/wasce/geronimo-web.xml` to include path dependencies to your REST data service configuration JAR or directory. See section for an example on setting the classpath to include your `wxsRestService.properties` file and other configuration files and metadata classes.
   b. Open the WebSphere Application Server Community Edition administration console and log in.

      Tip: The default URL is: `http://localhost:8080/console`. The default userid is `system` and password is `manager`.
   c. Click on the **Deploy New**link on the left side of the console window.
   d. On the **Install New Applications** page, enter the following values into the text boxes:

Table 27. Install New Applications

Text box	**Value**
Archive	`restservice_home/lib/wxsrestservice.war`
Plan	`restservice_home/gettingstarted/wasce/geronimo-web.xml`

      Tip: Use the path to the `geronimo-web.xml` file that you copied and edited in step 3.
   e. Click on the Install button. The console page then indicates that the application was successfully installed and started.
   f. Examine the WebSphere Application Server Community Edition system output log or console to verify that the REST data service has started successfully. The following message must appear:
      ```
      CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been
      started.
      ```
6. Start the WebSphere Application Server Community Edition server by running the following command:
   - UNIX Linux `wasce_root/bin/startup.sh`
   - Windows `wasce_root/bin/startup.bat`
7. Install the eXtreme Scale REST data service and the provided sample into the WebSphere Application Server Community Edition server:

a. Add the ObjectGrid client runtime JAR to the WebSphere Application Server Community Edition repository:
   1) Open the WebSphere Application Server Community Edition administration console and log in. The default URL is: `http://localhost:8080/console`. The default userid is `system` and password is `manager`.
   2) Click the **Repository** link on the left side of the console window, in the Services folder.
   3) In the **Add Archive to Repository** section, fill in the following into the input text boxes:

Table 28. Add Archive to Repository

Text box	Value
File	wxs_home/lib/ogclient.jar
Group	com.ibm.websphere.xs
Artifact	ogclient
Version	7.1
Type	JAR

   4) Click the install button.

      Tip: See the following technote for details on different ways class and library dependencies can be configured: Specifying external dependencies to applications running on WebSphere Application Server Community Edition

b. Deploy the REST data service module: `wxsrestservice.war` to the WebSphere Application Server Community Edition server.
   1) Edit the sample `restservice_home/gettingstarted/wasce/geronimo-web.xml` deployment XML file to include path dependencies to the getting started sample classpath directories:
      - Change the "classesDirs" for the two getting started client GBeans:

      The "classesDirs" path for the GettingStarted_Client_SharedLib GBean should be set to: `restservice_home/gettingstarted/restclient/bin`

      The "classesDirs" path for the GettingStarted_Common_SharedLib GBean should be set to: `restservice_home/gettingstarted/common/bin`
   2) Open the WebSphere Application Server Community Edition administration console and log in.
   3) Click on the **Deploy New** link on the left side of the console window.
   4) On the **Install New Applications** page, enter the following values into the text boxes:

Table 29. Install New Applications

Text box	Value
Archive	*restservice_home*/lib/wxsrestservice.war
Plan	*restservice_home*/gettingstarted/wasce/geronimo-web.xml

   5) Click the **Install** button.

      The console page then indicates that the application has successfully installed and started.

6) Examine the WebSphere Application Server Community Edition system output log to verify that the REST data service has started successfully by verifying that the following message is present:

   `CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been started.`

8. Verify that the REST data service is working:

   Open a Web browser and navigate to the following URL: `http://<host>:<port>/<context root>/restservice/<Grid Name>`

   The default port for WebSphere Application Server Community Edition is 8080 and is defined using the "HTTPPort" property in the `/var/config/config-substitutions.properties` file.

   For example: http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/

Results

The AtomPub service document is displayed.

Starting the REST data service in WebSphere Application Server Community Edition:

This topic describes how to configure and start the eXtreme Scale REST data service using WebSphere Application Server Community Edition.

Before you begin

Verify that the sample data grid is started. See "Enabling the REST data service" on page 424 for details on how to start the grid.

Procedure

1. Download and install WebSphere Application Server Community Edition Version 2.1.1.3 or later to `wasce_root`, such as: `/opt/IBM/wasce`
2. Start the WebSphere Application Server Community Edition server by running the following command:
   - Linux UNIX `wasce_root/bin/startup.sh`
   - Windows `wasce_root/bin/startup.bat`
3. If the eXtreme Scale grid was started with eXtreme Scale security enabled, set the following properties in the `restservice_home/gettingstarted/restclient/bin/wxsRestService.properties` file.

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties
loginType=none
```

4. Install the eXtreme Scale REST data service and the provided sample into the WebSphere Application Server Community Edition server:
   a. Add the ObjectGrid client runtime JAR to the WebSphere Application Server Community Edition repository:
      1) Open the WebSphere Application Server Community Edition administration console and log in.

         Tip: The default URL is: `http://localhost:8080/console`. The default user ID is `system` and password is `manager`.
      2) Click the **Repository**, in the Services folder.

3) In the **Add Archive to Repository** section, fill in the following into the input text boxes:

Table 30. Archive to repository

Text box	**Value**
File	wxs_home/lib/ogclient.jar
Group	com.ibm.websphere.xs
Artifact	ogclient
Version	7.0
Type	jar

4) Click the Install button.

Tip: See the following tech note for details on different methods of configuration class and library dependencies: Specifying external dependencies to applications running on WebSphere Application Server Community Edition.

b. Deploy the REST data service module, which is the `wxsrestservice.war` file, to the WebSphere Application Server Community Edition server.

1) Edit the sample `restservice_home/gettingstarted/wasce/geronimo-web.xml` deployment XML file to include path dependencies to the getting started sample classpath directories:

Change the classesDirs paths for the two getting started client GBeans:

- The "classesDirs" path for the GettingStarted_Client_SharedLib GBean should be set to: `restservice_home/gettingstarted/restclient/bin`
- The "classesDirs" path for the GettingStarted_Common_SharedLib GBean should be set to: `restservice_home/gettingstarted/common/bin`

2) Open the WebSphere Application Server Community Edition administrative console and log in.

Tip: The default URL is: `http://localhost:8080/console`. The default user ID is `system` and password is `manager`.

3) Click **Deploy New**.

4) On the **Install New Applications** page, enter the following values into the text boxes:

Table 31. Installation values

Text box	**Value**
Archive	`restservice_home/lib/wxsrestservice.war`
Plan	`restservice_home/gettingstarted/wasce/geronimo-web.xml`

5) Click on the Install button.

The console page should indicate that the application was successfully installed and started.

6) Examine the WebSphere Application Server Community Edition system output log or console to verify that the REST data service has started successfully by verify that the following message is present:

```
CWOBJ4000I: The WebSphere eXtreme Scale REST data service has
been started.
```

5. Verify that the REST data service is working:

   a. Open the following link in a browser window: `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid`. The service document for the NorthwindGrid grid is displayed.
   b. Open the following link in a browser window: `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata`. The Entity Model Data Extensions (EDMX) document is displayed.
6. To stop the grid processes, use `CTRL+C` in the respective command window to stop the process.
7. To stop WebSphere Application Server Community Edition, use the following command:
   - UNIX Linux `wasce_root/bin/shutdown.sh`
   - Windows `wasce_root\bin\shutdown.bat`

   Tip: The default user ID is `system` and password is `manager`. If you are using a custom port, use the `-port` option.

Deploying the REST data service on Apache Tomcat

This topic describes how to configure theWebSphere eXtreme Scale REST data service on Apache Tomcat Version 5.5 or later.

About this task

- An IBM or Oracle JRE or JDK, Version 5 or later installed and a specified JAVA_HOME environment variable.
- Apache Tomcat Version 5.5 or later is installed. See Apache Tomcat for details on how to install Tomcat.
- A stand-alone installation of WebSphere eXtreme Scale.

Procedure

1. If using an Oracle JRE or JDK, install the IBM ORB into Tomcat:
   a. Tomcat version 5.5:

      Copy all of the JAR files from:

      the *`wxs_home`*`/lib/endorsed` directory

      to:

      the *`tomcat_root`*`/common/endorsed` directory
   b. Tomcat version 6.0:

      Create an "endorsed" directory:

      UNIX Linux `mkdir tomcat_root/endorsed`

      Windows `md tomcat_root/endorsed`

      Copy all of the JAR files from:

      `wxs_home/lib/endorsed`

      to:

      `tomcat_root/common/endorsed`
2. Configure and start a data grid.
   a. For details on configuring a data grid for use with the REST data service, see Chapter 6, “Configuring,” on page 257.
   b. Verify that an eXtreme Scale client can connect to and access entities in the grid. For an example, see “Configuring REST data services” on page 423.

3. Build the eXtreme Scale REST service configuration JAR or directory. See the packaging and deployment information in “Installing the REST data service” on page 220 for details.
4. Deploy the REST data service module: wxsrestservice.war to the Tomcat server.

   Copy the `wxsrestservice.war` file from:

   restservice_home/`lib`

   to:

   tomcat_root/`webapps`
5. Add the ObjectGrid client runtime JAR and the application JAR to the shared classpath in Tomcat:
   a. Edit the *tomcat_root*/`conf/catalina.properties` file
   b. Append the following path names to the end of the shared.loader property, separating each path name with a comma:
      - *wxs_home*/`lib/ogclient.jar`
      - *restservice_home*/`gettingstarted/restclient/bin`
      - *restservice_home*/`gettingstarted/common/bin`
6. If you are using Java 2 security, add security permissions to the tomcat policy file:
   - If using Tomcat version 5.5:

     Merge the contents of the sample 5.5 `catalina policy` file found in *restservice_home*/`gettingstarted/tomcat/catalina-5_5.policy` with the *tomcat_root*/`conf/catalina.policy` file.
   - If using Tomcat version 6.0:

     Merge the contents of the sample 6.0 `catalina policy` file found in *restservice_home*/`gettingstarted/tomcat/catalina-6_0.policy` with the *tomcat_root*/`conf/catalina.policy` file.
7. Start the Tomcat server:
   - **If using Tomcat 5.5 on UNIX or Windows, or the Tomcat 6.0 ZIP distribution:**
     a. `cd` *tomcat_root*`/bin`
     b. Start the server:
        – Without Java 2 security enabled:

          UNIX Linux `./catalina.sh run`

          Windows `catalina.bat run`

        – With Java 2 security enabled:

          UNIX Linux `./catalina.sh run -security`

          Windows `catalina.bat run -security`
     c. The Apache Tomcat logs are displayed to the console. When the REST data service has started successfully, the following message is displayed in the administrative console:

        ```
        CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been
        started.
        ```
   - **If using Tomcat 6.0 on Windows using the Windows installer distribution:**
     a. `cd /bin`
     b. Start the Apache Tomcat 6 configuration tool:

        `tomcat6w.exe`

c. To enable Java 2 security (optional):

   Add the following entries to the Java Options in the Java tab in the Apache Tomcat 6 properties window:

   `-Djava.security.manager`

   `-Djava.security.policy=\conf\catalina.policy`

d. Click on the Start button on the Apache Tomcat 6 properties window to start the Tomcat server.

e. Review the following logs to verify that the Tomcat server has started successfully:

   – *tomcat_root*/`bin/catalina.log`

     Displays the status of the Tomcat server engine

   – *tomcat_root*/`bin/stdout.log`

     Displays the system output log

f. When the REST data service has started successfully, the following message is displayed in the system output log:

   ```
   CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been
   started.
   ```

8. Verify the REST data service is working.

   Open a Web browser and navigate to the following URL:

   http://*host:port*/*context_root*/restservice/*grid_name*

   The default port for Tomcat is 8080 and is configured in the *tomcat_root*/`conf/server.xml` file in the <Connector> element.

   For example:

   `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/`

Results

The AtomPub service document is displayed.

Starting REST data services in Apache Tomcat:

This topic describes how to configure and start the eXtreme Scale REST data service using Apache Tomcat, version 5.5 or later.

Before you begin

Verify that the sample eXtreme Scale data grid is started. See "Enabling the REST data service" on page 424 for details on how to start the data grid.

Procedure

1. Download and install Apache Tomcat Version 5.5 or later to tomcat_root. For example: `/opt/tomcat`
2. Install the eXtreme Scale REST data service and the provided sample into the Tomcat server as follows:
   a. If you are using an Oracle JRE or JDK, you must install the IBM ORB into Tomcat:
      - For Tomcat version 5.5

        Copy all of the JAR files from:

        `wxs_home/lib/endorsed`

        to

`tomcat_root/common/endorsed`

- For Tomcat version 6.0
  1) Create an "endorsed" directory
     - UNIX Linux `mkdir tomcat_root/endorsed`
     - Windows `md tomcat_root/endorsed`
  2) Copy all of the JAR files from:

     `wxs_home/lib/endorsed`

     to

     `tomcat_root/endorsed`

b. Deploy the REST data service module: wxsrestservice.war to the Tomcat server.

   Copy the wxsrestservice.war file from:

   `restservice_home/lib`

   to:

   `tomcat_root/webapps`

c. Add the ObjectGrid client runtime JAR and the application JAR to the shared classpath in Tomcat:
   1) Edit the `tomcat_root/conf/catalina.properties` file
   2) Append the following path names to the end of the shared.loader property in the form of a comma-delimited list:
      - wxs_home/lib/ogclient.jar
      - restservice_home/gettingstarted/restclient/bin
      - restservice_home/gettingstarted/common/bin

      Important: The path separator must be a **forward** slash.

3. If the eXtreme Scale data grid was started with eXtreme Scale security enabled, set the following properties in the `restservice_home/gettingstarted/restclient/bin/wxsRestService.properties` file.

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties
loginType=none
```

4. Start the Tomcat server with the REST data service:
   - If using Tomcat 5.5 on UNIX or Windows, or Tomcat 6.0 on UNIX:
     a. `cd tomcat_root/bin`
     b. Start the server:
        - UNIX Linux `./catalina.sh run`
        - Windows `catalina.bat run`
     c. The console then displays the Apache Tomcat logs. When the REST data service has started successfully, the following message is displayed in the administration console:

        ```
        CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been
        started.
        ```
   - If using Tomcat 6.0 on Windows:
     a. `cd tomcat_root/bin`
     b. Start the Apache Tomcat 6 configuration tool with the following command: `tomcat6w.exe`
     c. Click on the Start button on the Apache Tomcat 6 properties window to start the Tomcat server.

d. Review the following logs to verify that the Tomcat server has started successfully:
   - `tomcat_root/bin/catalina.log`

     Displays the status of the Tomcat server engine
   - `tomcat_root/bin/stdout.log`

     Displays the system output log.

e. When the REST data service has started successfully, the following message is displayed in the system output log: `CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been started.`

5. Verify that the REST data service is working:
   a. Open a browser and navigate to:

      http://localhost:8080/wxsrestservice/restservice/NorthwindGrid

      The service document for the NorthwindGrid is displayed.
   b. Navigate to:

      http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata

      The Entity Model Data Extensions (EDMX) document is displayed.
6. To stop the data grid processes, use CTRL+C in the respective command window.
7. To stop Tomcat, use CTRL +C in the window in which you started it.

Configuring Web browsers to access REST data service ATOM feeds

The eXtreme Scale REST data service creates ATOM feeds by default when using a web browser. The ATOM feed format may not be compatible with older browsers or may be interpreted such that the data cannot be viewed as XML. You can configure Internet Explorer Version 8 and Firefox Version 3 to display the ATOM feeds and XML within the browser.

About this task

The eXtreme Scale REST data service creates ATOM feeds by default when using a web browser. The ATOM feed format may not be compatible with older browsers or may be interpreted such that the data cannot be viewed as XML. For older browsers, you will be prompted to save the files to disk. Once the files are downloaded, use your favorite XML reader to look at the files. The generated XML is not formatted to be displayed, so everything will be printed on one line. Most XML reading programs, such as Eclipse, support reformatting the XML into a readable format.

For modern browsers, such as Microsoft Internet Explorer Version 8 and Firefox Version 3, the ATOM XML files can be displayed natively in the browser. The following topics provide details on how to configure Internet Explorer Version 8 and Firefox Version 3 to display the ATOM feeds and XML within the browser.

Procedure

Configure Internet Explorer Version 8

- To enable Internet Explorer to read the ATOM feeds that the REST data service generates use the following steps:
  1. Click **Tools** > **Internet Options**

2. Select the **Content** tab
3. Click the **Settings** button in the **Feeds and Web Slices** section
4. Uncheck the box: "Turn on feed reading view"
5. Click **OK** to return to the browser.
6. Restart Internet Explorer.

Configure Firefox Version 3

- Firefox does not automatically display pages with content type: application/atom+xml. The first time a page is displayed, Firefox prompts you to save the file. To display the page, open the file itself with Firefox as follows:
  1. From the application chooser dialog box, select the "Open with" radio button and click the **Browse** button.
  2. Navigate to your Firefox installation directory. For example: `C:\Program Files\Mozilla Firefox`
  3. Select `firefox.exe` and hit the **OK** button.
  4. Check the "Do this automatically for files like this..." check box.
  5. Click the **OK** button.
  6. Next, Firefox displays the ATOM XML page in a new browser window or tab
- Firefox automatically renders ATOM feeds in readable format. However, the feeds that the REST data service creates include XML. Firefox cannot display the XML unless you disable the feed renderer. Unlike Internet Explorer, in Firefox, the ATOM feed rendering plug-in must be explicitly edited. To configure Firefox to read ATOM feeds as XML files, follow these steps:
  1. Open the following file in a text editor: `<firefoxInstallRoot>\components\FeedConverter.js`. In the path, `<firefoxInstallRoot>` is the root directory where Firefox is installed.

     For Windows operating systems, the default directory is: `C:\Program Files\Mozilla Firefox`.
  2. Search for the snippet that looks as follows:
     ```
     // 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. Comment out the two lines that begin with `if` and `result` by placing `//` (two forward slashes) in front of them.
  4. Append the following statement to the snippet: `if(0) {`.
  5. The resulting text should look as follows:
     ```
     // 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. Save the file.
  7. Restart Firefox
  8. Now Firefox can automatically display all feeds in the browser.
- Test your setup by trying some URLs.

Example

This section describes some example URLs that can be used to view the data that was added by the getting started sample provided with the REST data service.

Before using the following URLs, add the default data set to the eXtreme Scale sample data grid using either the sample Java client or the sample Visual Studio WCF Data Services client.

The following examples assume the port is 8080 which can vary. See section for details on how to configure the REST data service on different application servers.

- View a single customer with the id of "ACME":

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')
```

- View all of the orders for customer "ACME":

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')/orders
```

- View the customer "ACME" and the orders:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')?$expand=orders
```

- View order 1000 for customer "ACME":

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Order(orderId=1000,customer_customerId='ACME')
```

- View order 1000 for customer "ACME" and its associated Customer:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
Order(orderId=1000,customer_customerId='ACME')?$expand=customer
```

- View order 1000 for customer "ACME" and its associated Customer and OrderDetails:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
Order(orderId=1000,customer_customerId='ACME')?$expand=customer,orderDetails
```

- View all orders for customer "ACME" for the month of October, 2009 (GMT):

```
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'
```

- View all the first 3 orders and orderDetails for customer "ACME" for the month of October, 2009 (GMT):

```
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
```

Using a Java client with REST data services

The Java client application uses the eXtreme Scale EntityManager API to insert data into the grid.

About this task

The previous sections described how to create an eXtreme Scale data grid and configure and start the eXtreme Scale REST data service. The Java client application uses the eXtreme Scale EntityManager API to insert data into the grid. It does not demonstrate how to use the REST interfaces. The purpose of this client is to demonstrate how the EntityManager API is used to interact with the eXtreme Scale data grid, and allow modifying data in the grid. To view data in the grid using the REST data service, use a web browser or use the Visual Studio 2008 client application.

Procedure

To quickly add content to the eXtreme Scale data grid, run the following command:

1. Open a command-line or terminal window and set the JAVA_HOME environment variable:

- Linux UNIX `export JAVA_HOME=`*`java_home`*
- Windows `set JAVA_HOME=`*`java_home`*

2. `cd restservice_home/gettingstarted`
3. Insert some data into the grid. The data that is inserted will be retrieved later using a Web browser and the REST data service.

   If the data grid was started *without*eXtreme Scale security, use the following commands.

   - UNIX Linux `./runclient.sh load default`
   - Windows `runclient.bat load default`

   If the data grid was started *with*eXtreme Scale security, use the following commands.

   - UNIX Linux `./runclient_secure.sh load default`
   - Windows `runclient_secure.bat load default`

   For a Java client, use the following command syntax:

   - UNIX Linux `runclient.sh` *`command`*
   - Windows `runclient.bat` *`command`*

   The following commands are available:

   - `load default`

     Loads a predefined set of Customer, Category and Product entities into the data grid and creates a random set of Orders for each customer.
   - `load category` *`categoryId categoryName firstProductId num_products`*

     Creates a product Category and a fixed number of Product entities in the data grid. The firstProductId parameter identifies the id number of the the first product and each subsequent product is assigned the next id until the specified number of products is created.
   - `load customer` *`companyCode contactNamecompanyName numOrders firstOrderIdshipCity maxItems discountPct`*

     Loads a new Customer into the data grid and creates a fixed set of Order entities for any random product currently loaded in the grid. The number of Orders is determined by setting the <numOrders> parameter. Each Order will have a random number of OrderDetail entities up to <maxItems>
   - `display customer` *`companyCode`*

     Display a Customer entity and the associated Order and OrderDetail entities.
   - `display category` *`categoryId`*

     Display a product Category entity and the associated Product entities.

Results

- `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`

Running and building the sample data grid and Java client with Eclipse

The REST data service getting started sample can be updated and enhanced using Eclipse. For details on how to setup your Eclipse environment see the text document: `restservice_home/gettingstarted/ECLIPSE_README.txt`.

After the WXSRestGettingStarted project is imported into Eclipse and is building successfully, the sample will automatically re-compile and the script files used to start the container server and client will automatically pick up the class files and XML files. The REST data service will also automatically detect any changes since the Web server is configured to read the Eclipse build directories automatically.

Important: When changing source or configuration files, both the eXtreme Scale container server and the REST data service application must be restarted. The eXtreme Scale container server must be started before the REST data service Web application.

Visual Studio 2008 WCF client with REST data service

The eXtreme Scale REST data service getting started sample includes a WCF Data Services client that can interact with the eXtreme Scale REST data service. The sample is written as a command-line application in C#.

Software requirements

The WCF Data Services C# sample client requires the following:
- Operating system
  - Microsoft Windows XP
  - Microsoft Windows Server 2003
  - Microsoft Windows Server 2008
  - Microsoft Windows Vista
- Microsoft Visual Studio 2008 with Service Pack 1

  Tip: See the previous link for additional hardware and software requirements.
- Microsoft .NET Framework 3.5 Service Pack 1
- Microsoft Support: An update for the .NET Framework 3.5 Service Pack 1 is available

Building and running the getting started client

The WCF Data Services sample client includes a Visual Studio 2008 project and solution and the source code for running the sample. The sample must be loaded into Visual Studio 2008 and compiled into a Windows runnable program before it can be run. To build and run the sample, see the text document: `restservice_home/gettingstarted/VS2008_README.txt`.

WCF Data Services C# client command syntax

Windows
```
WXSRESTGettingStarted.exe <service URL> <command>
```

The <service URL> is the URL of the eXtreme Scale REST data service configured in section .

The following commands are available:

- `load default`

  Loads a predefined set of Customer, Category and Product entities into the data grid and creates a random set of Orders for each customer.
- `load category <categoryId> <categoryName> <firstProductId> <numProducts>`

  Creates a product Category and a fixed number of Product entities in the data grid. The firstProductId parameter identifies the id number of the the first product and each subsequent product is assigned the next id until the specified number of products is created.
- `load customer <companyCode> <contactName> <companyName> <numOrders> <firstOrderId> <shipCity> <maxItems> <discountPct>`

  Loads a new Customer into the data grid and creates a fixed set of Order entities for any random product currently loaded in the data grid. The number of Orders is determined by setting the <numOrders> parameter. Each Order will have a random number of OrderDetail entities up to <maxItems>
- `display customer <companyCode>`

  Display a Customer entity and the associated Order and OrderDetail entities.
- `display category <categoryId>`

  Display a product Category entity and the associated Product entities.
- `unload`

  Remove all entities that were loaded using the "default load" command.

The following examples illustrate various commands.

- `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`

Configuring servers for OSGi

WebSphere eXtreme Scale includes a server OSGi bundle, allowing starting and configuring servers and containers within an OSGi framework. The configuration topics describe how to use the eXtreme Scale server bundle, OSGi Blueprint service and eXtreme Scale configuration to run eXtreme Scale servers in an Eclipse Equinox OSGi framework.

About this task

the following tasks are required to start an eXtreme Scale server within Eclipse Equinox:

Procedure

1. Create an OSGi bundle that will store the eXtreme Scale plug-ins, exposing them as services and update the ObjectGrid descriptor XML file to reference the services.
2. Configure OSGi to start an eXtreme Scale container server.
3. Install and start the eXtreme Scale server bundle in the OSGi framework.
4. Install and start the OSGi bundle that contains the eXtreme Scale plug-ins.

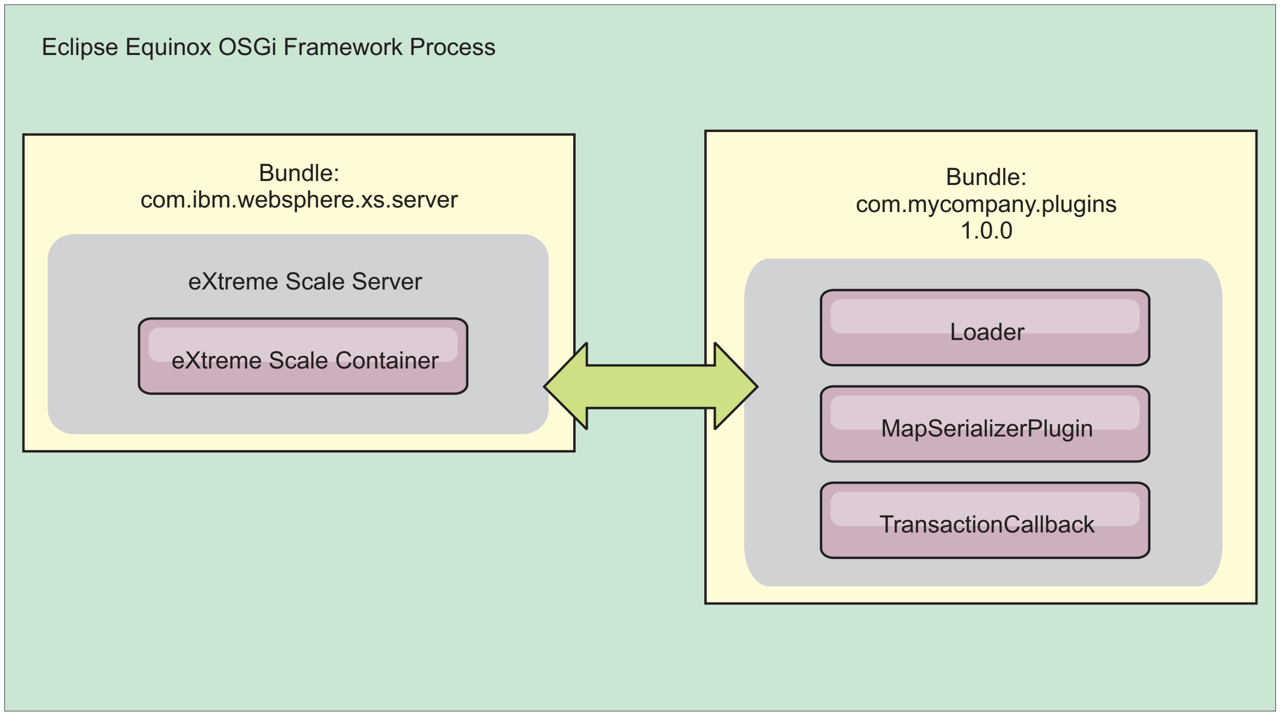


Figure 49. Eclipse Equinox process for installing and starting OSGi bundles with eXtreme Scale plug-ins

Configuring eXtreme Scale plug-ins with OSGi Blueprint

All eXtreme Scale ObjectGrid and BackingMap plug-ins can be defined as OSGi beans and services using the OSGi Blueprint Service available with Eclipse Gemini or Apache Aries.

Before you begin

Before you can configure your plug-ins as OSGi services, you must first package your plug-ins in an OSGi bundle, and understand the fundamental principles of the required plug-ins. The bundle must import the WebSphere eXtreme Scale server or client packages and other dependent packages required by the plug-ins, or create a bundle dependency on the eXtreme Scale server or client bundles This topic describes how to configure the Blueprint XML to create plug-in beans and expose them as OSGi services for eXtreme Scale to use.

About this task

Beans and services are defined in a Blueprint XML file, and the Blueprint container discovers, creates, and wires the beans together and exposes them as services. The process makes the beans available to other OSGi bundles, including the eXtreme Scale server and client bundles.

When creating custom plug-in services for use with eXtreme Scale, the bundle that is to host the plug-ins, must be configured to use Blueprint. In addition, a Blueprint XML file must be created and stored within the bundle. Read about building OSGi applications with the Blueprint Container specification for a general understanding of the specification.

Procedure

1. Create a Blueprint XML file. You can name the file anything. However, you must include the blueprint namespace:
```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
...
</blueprint>
```
2. Create bean definitions in the Blueprint XML file for each eXtreme Scale plug-in.

   Beans are defined using the `<bean>` element and can be wired to other bean references and can include initialization parameters.

   Important: When defining a bean, you must use the correct scope. Blueprint supports the singleton and prototype scopes. eXtreme Scale also supports a custom shard scope.

   Define most eXtreme Scale plug-ins as prototype or shard-scoped beans, since all of the beans must be unique for each ObjectGrid shard or BackingMap instance it is associated with. Shard-scoped beans can be useful when using the beans in other contexts to allow retrieving the correct instance.

   To define a prototype-scoped bean, use the `scope="prototype"` attribute on the bean:
```
<bean id="myPluginBean" class="com.mycompany.MyBean" scope="prototype">
...
</bean>
```
   To define a shard-scoped bean, you must add the `objectgrid` namespace to the XML schema, and use the `scope="objectgrid:shard"` attribute on the bean:
```
<?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. Create PluginServiceFactory bean definitions for each plug-in bean. All eXtreme Scale beans must have a PluginServiceFactory bean defined so that the correct bean scope can be applied. eXtreme Scale includes a BlueprintServiceFactory that you can use. It includes two properties that must be set. You must set the blueprintContainer property to the `blueprintContainer` reference, and the beanId property must be set to the bean identifier name. When eXtreme Scale looks up the service to instantiate the appropriate beans, the server looks up the bean component instance using the Blueprint container.

```
bean id="myPluginBeanFactory"
    class="com.ibm.websphere.objectgrid.plugins.osgi.BluePrintServiceFactory">
    <property name="blueprintContainer" ref="blueprintContainer" />
    <property name="beanId" value="myPluginBean" />
</bean>
```

4. Create a service manager for each PluginServiceFactory bean. Each service manager exposes the PluginServiceFactory bean, using the `<service>` element. The service element identifies the name to expose to OSGi, the reference to the PluginServiceFactory bean, the interface to expose, and the ranking of the service. eXtreme Scale uses the service manager ranking to perform service upgrades when the eXtreme Scale grid is active. If the ranking is not specified, the OSGi framework assumes a ranking of 0. Read about updating service rankings for more information.

   Blueprint includes several options for configuring service managers. To define a simple service manager for a PluginServiceFactory bean, create a `<service>` element for each PluginServiceFactory bean:

   ```
   <service ref="myPluginBeanFactory"
       interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory"
       ranking="1">
   </service>
   ```

5. Store the Blueprint XML file in the plug-ins bundle. The Blueprint XML file must be stored in the `OSGI-INF/blueprint` directory for the Blueprint container to be discovered.

   To store the Blueprint XML file in a different directory, you must specify the following Bundle-Blueprint manifest header:

   ```
   Bundle-Blueprint: OSGI-INF/blueprint.xml
   ```

Results

The eXtreme Scale plug-ins are now configured to be exposed in an OSGi Blueprint container, In addition, the ObjectGrid descriptor XML file is configured to reference the plug-ins using the OSGi Blueprint service.

Related concepts:

Samples

System APIs and plug-ins
A plug-in is a component that provides a function to the pluggable components, which include ObjectGrid and BackingMap. To most effectively use eXtreme Scale as an in-memory data grid or database processing space, you should carefully determine how best you can maximize performance with available plug-ins.

Related information:

Building OSGi applications with the Blueprint Container specification

OSGi Bundle Activator API documentation

Spring namespace schema

Configuring servers with OSGi Blueprint

You can configure WebSphere eXtreme Scale container servers using an OSGi blueprint XML file, allowing simplified packaging and development of self-contained server bundles.

Before you begin

This topic assumes that the following tasks have been completed:

- The Eclipse Equinox OSGi framework has been installed and started with either the Eclipse Gemini or Apache Aries blueprint container.
- The eXtreme Scale server bundle has been installed and started.
- The eXtreme Scale dynamic plug-ins bundle has been created.
- The eXtreme Scale ObjectGrid descriptor XML file and deployment policy XML file have been created.

About this task

This task describes how to configure an eXtreme Scale server with a container using a blueprint XML file. The result of the procedure is a container bundle. When the container bundle is started, the eXtreme Scale server bundle will track the bundle, parse the server XML and start a server and container.

A container bundle can optionally be combined with the application and eXtreme Scale plug-ins when dynamic plug-in updates are not required or the plug-ins do not support dynamic updating.

Procedure

1. Create a Blueprint XML file with the `objectgrid` namespace included. You can name the file anything. However, it must include the blueprint namespace:

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

   <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
             xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
             xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
                  http://www.ibm.com/schema/objectgrid/objectgrid.xsd">
   ...
   </blueprint>
   ```

2. Add the XML definition for the eXtreme Scale server with the appropriate server properties. See the Spring descriptor XML file for details on all available configuration properties. See the following example of the XML definition:

   ```
   <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. Add the XML definition for the eXtreme Scale container with the reference to the server definition and the ObjectGrid descriptor XML and ObjectGrid deployment XML files embedded in the bundle; for example:

   ```
   <objectgrid:container id="container"
       objectgridxml="/META-INF/objectGrid.xml"
       deploymentxml="/META-INF/objectGridDeployment.xml"
       server="xsServer" />
   ```

4. Store the Blueprint XML file in the container bundle. The Blueprint XML must be stored in the `OSGI-INF/blueprint` directory for the Blueprint container to be found.

   To store the Blueprint XML in a different directory, you must specify the Bundle-Blueprint manifest header; for example:

   ```
   Bundle-Blueprint: OSGI-INF/blueprint.xml
   ```

5. Package the files into a single bundle JAR file. See the following example of a bundle directory hierarchy:

```
MyBundle.jar
    /META-INF/manifest.mf
    /META-INF/objectGrid.xml
    /META-INF/objectGridDeployment.xml
    /OSGI-INF/blueprint/blueprint.xml
```

Results

An eXtreme Scale container bundle is now created and can be installed in Eclipse Equinox. When the container bundle is started, the eXtreme Scale server runtime environment in the eXtreme Scale server bundle, will automatically start the singleton eXtreme Scale server using the parameters defined in the bundle, and starts a container server. The bundle can be stopped and started, which results in the container stopping and starting. The server is a singleton and does not stop when the bundle is started the first time.

Configuring servers with OSGI config admin

You can use the OSGi configuration administration (config admin) service to configure WebSphere eXtreme Scale container servers.

About this task

To configure a server, the ManagedService persistent identifier (PID), com.ibm.websphere.xs.server, is set to reference the ObjectGrid server properties file on the file system. To configure a container, the ManagedServiceFactory PID, com.ibm.websphere.xs.container, is set to reference the ObjectGrid deployment XML file and ObjectGrid deployment policy XML file on the file system.

When the two PIDs are set in the config admin service, the eXtreme Scale server service automatically initializes the server and start the container with the specified configuration files. Config admin PIDs are persisted to the OSGi configuration directory. If the configuration is not cleared, the settings are retained between framework restarts.

Several third-party utilities exist for setting config admin properties. The following utilities are examples of tools that the product supports:
- The Luminis OSGi Configuration Admin command line client allows command line configuration.
- Apache Felix File Install allows specifying config admin PID settings in standard property files.

To configure eXtreme Scale container servers with the OSGi Configuration Administration command-line client for Luminis, complete the following steps

Procedure

1. Create a managed service PID for the ObjectGrid server properties file in the OSGi console, by running the following commands:
   ```
   osgi> cm create com.ibm.websphere.xs.server
   osgi> cm put com.ibm.websphere.xs.server objectgrid.server.props /mypath/server.properties
   ```
2. Create a managed service factory persistence identifier PID for the ObjectGrid container in the OSGi console by running the following commands.

   Attention: Use the PID that is created with the **`createf`** config admin command. The PID that is used in the following code snippet is only an example.

```
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
```

Results

eXtreme Scale container servers are now configured to start in an Eclipse Equinox OSGi framework.

What to do next

Container servers can also be programmatically created using the ServerFactory API and OSGi bundle activators. For details on using the ServerFactory API, see the API documentation.

Chapter 7. Administering

Administering and operating the product environment includes starting and stopping servers, managing the availability of the data grid, and recovering from data center failure scenarios. After you configure your catalog servers and container servers, you can start and stop the servers using various methods. The method that you use to start and stop servers depends on if you are using an embedded topology, a stand-alone topology, or a topology that is running within WebSphere Application Server.

Related tasks:

"Troubleshooting administration" on page 631
Use the following information to troubleshoot administration, including starting and stopping servers, using the **xscmd** utility, and so on.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

"Starting and stopping stand-alone servers"
You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

"Starting a stand-alone catalog service" on page 461
You must start the catalog service manually when you are using a distributed WebSphere eXtreme Scale environment that is not running in WebSphere Application Server.

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

Starting and stopping stand-alone servers

You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

Before you begin

If you are starting or stopping servers in a stand-alone environment that is using an external client security provider, you must set the *CLIENT_AUTH_LIB* environment variable before you run the start and stop scripts. For more information about setting this environment variable, see "Starting secure servers in a stand-alone environment" on page 606.

Related concepts:

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that

include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Chapter 7, "Administering," on page 459

Related reference:

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Related information:

"Getting started tutorial lesson 3.1: Starting catalog and container servers" on page 5
To run the sample client application, you must start a catalog server and a container server.

Starting stand-alone servers

When you are running a stand-alone configuration, the environment is comprised of catalog servers, container servers, and client processes. WebSphere eXtreme Scale servers can also be embedded within existing Java applications by using the embedded server API. You must manually configure and start these processes.

Before you begin

You can start WebSphere eXtreme Scale servers in an environment that does not have WebSphere Application Server installed. If you are using WebSphere Application Server, see "Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297.

Deprecated: The **startOgServer** and **stopOgServer** commands start servers that use the Object Request Broker (ORB) transport mechanism. The ORB is deprecated, but you can continue using these scripts if you were using the ORB in a previous release. The IBM eXtremeIO (XIO) transport mechanism replaces the ORB. Use the **startXsServer** and **stopXsServer** scripts to start and stop servers that use the XIO transport.

Related reference:

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

"Server trace options" on page 621
You can enable trace to provide information about your environment to IBM support.

Messages
When you encounter a message in a log or other parts of the product interface, you can look up the message by its component prefix to find out more information.

Related information:

Interface PlacementServiceMBean

Starting a stand-alone catalog service

You must start the catalog service manually when you are using a distributed WebSphere eXtreme Scale environment that is not running in WebSphere Application Server.

Before you begin

Deprecated: The **startOgServer** and **stopOgServer** commands start servers that use the Object Request Broker (ORB) transport mechanism. The ORB is deprecated, but you can continue using these scripts if you were using the ORB in a previous release. The IBM eXtremeIO (XIO) transport mechanism replaces the ORB. Use the **startXsServer** and **stopXsServer** scripts to start and stop servers that use the XIO transport.

- If you are using WebSphere Application Server, the catalog service automatically starts within the existing processes. For more information, see "Starting and stopping servers in a WebSphere Application Server environment" on page 475.

About this task

Start the catalog service with the **startOgServer** script. When you call the start command, use the **startOgServer.sh** script on UNIX platforms or **startOgServer.bat** on Windows.

The catalog service can run in a single process or can include multiple catalog servers to form a catalog service domain. A catalog service domain is required in a production environment for high availability. For more information, see High availability catalog service. You can also specify additional parameters to the script to bind the Object Request Broker (ORB) to a specific host and port, specify the domain, or enable security.

Procedure

- **Start a single catalog server process.**

To start a single catalog server, type the following commands from the command line:

1. Navigate to the `bin` directory.

   ```
   cd objectgridRoot/bin
   ```

2. Run the **`startOgServer`** command.

   ```
   startOgServer.bat|sh catalogServer
   ```

For a list of all of the available command-line parameters, see "**`startOgServer`** script" on page 466. Do not use a single Java virtual machine (JVM) to run the catalog service in a production environment. If the catalog service fails, no new clients are able to route to the deployed eXtreme Scale, and no new ObjectGrid instances can be added to the domain. For these reasons, you should start a set of Java virtual machines to run a catalog service domain.

- **Start a catalog service domain that consists of multiple endpoints.**

  To start a set of servers to run a catalog service, you must use the **`-catalogServiceEndPoints`** option on the `startOgServer` script. This argument accepts a list of catalog service endpoints in the format of *`serverName`*`:`*`hostName`*`:`*`clientPort`*`:`*`peerPort`*.The following example shows how to start the first of three Java virtual machines to host a catalog service:

  1. Navigate to the bin directory.

     ```
     cd wxs_install_root/bin
     ```

  2. Run the **`startOgServer`** command.

```
startOgServer.bat|sh cs1 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,cs2:MyServer2.company.com:6601:6602,cs3:MyServer3.company.com:6601:6602
```

     In this example, the `cs1` server on the `MyServer1.company.com` host is started. This server name is the first argument that is passed to the script. During initialization of the `cs1` server, the **`-catalogServiceEndpoints`** parameters are examined to determine which ports are allocated for this process. The list is also used to allow the `cs1` server to accept connections from other servers: `cs2` and `cs3`.

  3. To start the remaining catalog servers in the list, pass the following arguments to the **`startOgServer`** script. Starting the `cs2` server on the `MyServer2.company.com` host.

```
startOgServer.bat|sh cs2 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,cs2:MyServer2.company.com:6601:6602,cs3:MyServer3.company.com:6601:6602
```

     Starting `cs3` on `MyServer3.company.com`:

```
startOgServer.bat|sh cs3 -catalogServiceEndPoints
cs3:MyServer3.company.com:6601:6602,cs1:MyServer1.company.com:6601:6602,cs2:MyServer2.company.com:6601:6602
```

     The order of the list for the **`-catalogServiceEndpoints`** parameter can be different for the various catalog servers, but the servers contained in the list must be the same. Do not put any spaces in the list.

     Important: Start at least two catalog servers in parallel.

     You must start catalog servers that are in a data grid in parallel, because each server pauses to wait for the other catalog servers to join the core group. A catalog server that is configured for a data grid does not start until it identifies other members in the group. The catalog server eventually times out if no other servers become available.

- **Bind the ORB to a specific host and port.**

  Aside from ports defined in the **`catalogServiceEndpoints`** argument, each catalog service also uses an Object Request Broker (ORB) to accept connections from clients and containers. By default, the ORB listens on port `2809` of the localhost. If you want to bind the ORB to a specific host and port on a catalog service JVM, use the **`-listenerHost`** and **`-listenerPort`** arguments. The following example shows how to start a single JVM catalog server with its ORB bound to port `7000` on `MyServer1.company.com`:

```
startOgServer.sh catalogServer -listenerHost MyServer1.company.com
-listenerPort 7000
```

Each eXtreme Scale container and client must be provided with catalog service ORB endpoint data. Clients only need a subset of this data, but you should use at least two endpoints for high availability.

- Optional: **Name the catalog service domain**

  A catalog service domain name is not required when starting a catalog service. However, if you are using multi-master replication or are using multiple catalog service domains within the same set of processes, then you need to define a unique catalog service domain name. The default domain name is `DefaultDomain`. To give your domain a name, use the **`-domain`** option. The following example demonstrates how to start a single catalog service JVM with the domain name `myDomain`.

  ```
   startOgServer.sh catalogServer -domain myDomain
  ```

  For more information about configuring multi-master replication, see "Configuring multiple data center topologies" on page 329.
- **Start a secure catalog service.** For more information, see "Starting secure servers in a stand-alone environment" on page 606.
- **Start the catalog service programmatically.**

  Any JVM setting that is flagged by the CatalogServerProperties.setCatalogServer method can host the catalog service for eXtreme Scale. This method indicates to the eXtreme Scale server run time to instantiate the catalog service when the server is started. The following code shows how to instantiate the eXtreme Scale catalog server:

  ```
  CatalogServerProperties catalogServerProperties =
   ServerFactory.getCatalogProperties();
  catalogServerProperties.setCatalogServer(true);

  //The getInstance() method will start the catalog service.
  Server server = ServerFactory.getInstance();
  ```

  For more information about starting servers programmatically, see "Using the embedded server API to start and stop servers" on page 476.

Related concepts:

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Chapter 7, "Administering," on page 459

Starting container servers

You can start container servers from the command line using a deployment topology or using a `server.properties` file.

About this task

To start a container process, you need an ObjectGrid XML file. The ObjectGrid XML file specifies which eXtreme Scale servers the container hosts. Ensure that your container is equipped to host each ObjectGrid in the XML that you pass to it. All of the classes that these ObjectGrids require must be in the classpath for the container. For more information about the ObjectGrid XML file, see `objectGrid.xsd` file.

Procedure

- **Start the container server from the command line.**
  1. From the command line, navigate to the bin directory:
     ```
     cd wxs_install_root/bin
     ```
  2. Run the following command:
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Important: On the container server, the **`-catalogServiceEndPoints`** option is used to reference the Object Request Broker (ORB) host and port on the catalog service. The catalog service uses the **`-listenerHost`** and **`-listenerPort`** options to specify the host and port for ORB binding or accepts the default binding. When you are starting a container, use the **`-catalogServiceEndPoints`** option to reference the values that are passed to the **`-listenerHost`** and **`-listenerPort`** options on the catalog service. If **`-listenerHost`** and **`-listenerPort`** options are not used when the catalog service is started, the ORB binds to port 2809 on the localhost for the catalog service. Do not use the **`-catalogServiceEndPoints`** option to reference the hosts and ports that were passed to the **`-catalogServiceEndPoints`** option on the catalog service. On the catalog service, the **`-catalogServiceEndPoints`** option is used to specify ports necessary for static server configuration.
  This process is identified by c0, the first argument passed to the script. Use the `companyGrid.xml` to start the container. If your catalog server ORB is running on a different host than your container or it is using a non-default port, you must use the **`-catalogServiceEndPoints`** argument to connect to the ORB. For this example, assume that a single catalog service is running on port `2809` on `MyServer1.company.com`
- **Start the container using a deployment policy.**

  Although not required, a deployment policy is recommended during container start up. The deployment policy is used to set up partitioning and replication for eXtreme Scale. The deployment policy can also be used to influence placement behavior. Because the previous example did not provide a deployment policy file, the example receives all default values with regard to replication, partitioning, and placement. So, the maps in the CompanyGrid are in one mapSet. The mapSet is not partitioned or replicated. For more information about deployment policy files, see Deployment policy descriptor XML file. The following example uses the `companyGridDpReplication.xml` file to start a container JVM, the c0 JVM:
  1. From the command line, navigate to the bin directory:
     ```
     cd wxs_install_root/bin
     ```
  2. Run the following command:
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

Note: If you have Java classes stored in a specific directory, or you are using a loader or agent, instead of altering the StartOgServer script, you can launch the server with arguments as follows: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`
. In the `companyGridDpReplication.xml` file, a single map set contains all of the maps. This mapSet is divided into 10 partitions. Each partition has one synchronous replica and no asynchronous replicas. Any container that uses the `companyGridDpReplication.xml` deployment policy paired with the `companyGrid.xml` ObjectGrid XML file is also able to host CompanyGrid shards. Start another container JVM, the c1 JVM:

1. From the command line, navigate to the bin directory:
   ```
   cd wxs_install_root/bin
   ```
2. Run the following command:
   ```
   startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplication.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

Each deployment policy contains one or more objectgridDeployment elements. When a container is started, it publishes its deployment policy to the catalog service. The catalog service examines each objectgridDeployment element. If the objectgridName attribute matches the objectgridName attribute of a previously received objectgridDeployment element, the latest objectgridDeployment element is ignored. The first objectgridDeployment element received for a specific objectgridName attribute is used as the master. For example, assume that the c2 JVM uses a deployment policy that divides the mapSet into a different number of partitions:

`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>
```

Now, you can start a third JVM, the c2 JVM:

1. From the command line, navigate to the bin directory:
   ```
   cd wxs_install_root/bin
   ```
2. Run the following command:
   ```
   startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

The container on the c2 JVM is started with a deployment policy that specifies 5 partitions for mapSet1. However, the catalog service already holds the master copy of the objectgridDeployment for the CompanyGrid. When the c0 JVM was started it specified that 10 partitions exist for this mapSet. Because it was the first container to start and publish its deployment policy, its deployment policy became the master. Therefore, any objectgridDeployment attribute value that is equal to `CompanyGrid` in a subsequent deployment policy is ignored.

- **Start a container using a server properties file.**

  You can use a server properties file to set up trace and configure security on a container. Run the following commands to start container c3 with a server properties file:

  1. From the command line, navigate to the bin directory:
     ```
     cd wxs_install_root/bin
     ```
  2. Run the following command:
     ```
     startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     -serverProps ../serverProps/server.properties
     ```

  An example `server.properties` file follows:

  ```
  server.properties
  workingDirectory=
  traceSpec=*=all=disabled
  systemStreamToFileEnabled=true
  enableMBeans=true
  memoryThresholdPercentage=50
  ```

  This is a basic server properties file that does not have security enabled. For more information about the `server.properties` file, see Server properties file.

- **Start a container server programmatically.**

  For more information about starting container servers programmatically, see "Using the embedded server API to start and stop servers" on page 476.

Related reference:

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

"Configuring distributed deployments" on page 278
Use the deployment policy descriptor XML file and the ObjectGrid descriptor XML file to manage your topology.

`startOgServer` script

The **`startOgServer`** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Purpose

You can use the **`startOgServer`** script to start servers.

Deprecated: The **`startOgServer`** and **`stopOgServer`** commands start servers that use the Object Request Broker (ORB) transport mechanism. The ORB is deprecated, but you can continue using these scripts if you were using the ORB in a previous release. The IBM eXtremeIO (XIO) transport mechanism replaces the ORB. Use the **`startXsServer`** and **`stopXsServer`** scripts to start and stop servers that use the XIO transport.

Location

The **`startOgServer`** script is in the `bin` directory of the root directory, for example:

```
cd wxs_install_root/bin
```

Note: If you have Java classes stored in a specific directory, or you are using a loader or agent, instead of altering the **startOgServer** script, you can launch the server with arguments as follows: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`

.

Usage for catalog servers

To start a catalog server:

Windows

```
startOgServer.bat <server> [options]
```

UNIX

```
startOgServer.sh <server>[options]
```

To start a default configured catalog server, use the following commands:

Windows

```
startOgServer.bat catalogServer
```

UNIX

```
startOgServer.sh catalogServer
```

Options for starting catalog servers

The following parameters are all optional.

Parameters for starting a catalog server:

`-catalogServiceEndPoints <serverName:hostName:clientPort:peerPort>`
Specifies a list of catalog servers to link together into a catalog service domain. Each attribute is defined as follows:

serverName
Specifies the name of the catalog server.

hostName
Specifies the host name for the computer where the server is launched.

clientPort
Specifies the port that is used for peer catalog service communication.

peerPort
This value is the same as the haManagerPort. Specifies the port that is used for peer catalog service communication.

The following example starts the cs1 catalog server, which is in the same catalog service domain as the cs2 and cs3 servers:

```
startOgServer.bat|sh cs1 -catalogServiceEndPoints
cs1:MyServer1.company.com:6601:6602,cs2:MyServer2.company.com:6601:6602,cs3:MyServer3.company.com:6601:6602
```

If you start additional catalog servers, they must include the same servers in the **`-catalogServiceEndPoints`** argument. The order of the list can be different, but the servers contained in the list must be the same for each catalog server. Do not put any spaces in the list.

-clusterSecurityFile <cluster security xml file>

Specifies the `objectGridSecurity.xml` file on the hard disk, which describes the security properties that are common to all servers (including catalog servers and container servers). One of the property example is the authenticator configuration which represents the user registry and authentication mechanism.

Example:`/opt/xs/ogsecurity.xml`

Important: Windows If you are using Windows, the directory path does not support backslashes. If you have used backslashes, you must escape any backslash (\ ) characters in the path. For example, if you want to use the path C:\opt\ibm, enter C:\\opt\\ibm in the properties file. Windows directories with spaces are not supported.

-clusterSecurityUrl <cluster security xml URL>

Specifies the `objectGridSecurity.xml` file as a URL to the file on the hard disk or on the network, which describes the security properties that are common to all servers (including catalog servers and container servers). One of the property example is the authenticator configuration which represents the user registry and authentication mechanism.

Example:`file:///opt/xs/ogsecurity.xml`

-domain <domain name>

Specifies the name of the catalog service domain for this catalog server. The catalog service domain creates a group of highly available catalog servers. Each catalog server for a single domain should specify the same value for the **-domain** parameter.

-haManagerPort <port>

Specifies the port that is used by the high availability (HA) manager for heartbeat communication between peer container servers. The **haManagerPort** port is only used for peer-to-peer communication between container servers that are in same domain. If the haManagerPort property is not defined, then an ephemeral port is used. In WebSphere Application Server, this setting is inherited by the high availability manager port configuration.

-heartbeat 0|1|-1

Specifies how often a server failover is detected. An aggressive heartbeat interval can be useful when the processes and network are stable. If the network or processes are not optimally configured, heartbeats might be missed, which can result in a false failure detection. The heartbeat frequency level is a trade-off between use of resources and failure discovery time. The more frequent a heartbeat occurs, then more resources are used. However, failures are discovered more quickly. This property applies to the catalog service only.

Table 32. Valid heartbeat values

Value	Action	Description
-1	Aggressive	Specifies an aggressive heartbeat level. With this value, failures are detected more quickly, but more processor and network resources are used. This level is more sensitive to missing heartbeats when the server is busy. Failovers are typically detected within 5 seconds.
0	Typical (default)	Specifies a heartbeat level at a typical rate. With this value, failover detection occurs at a reasonable rate without overusing resources. Failovers are typically detected within 30 seconds.
1	Relaxed	Specifies a relaxed heartbeat level. With this value, a decreased heartbeat frequency increases the time to detect failures, but also decreases processor and network use. Failovers are typically detected within 180 seconds.

-JMXConnectorPort <port>
Defines the Secure Sockets Layer (SSL) port to which the Java Management Extensions (JMX) service binds. Only required if an SSL transport protocol is needed for JMX data.

-JMXServicePort <port>
Required only for WebSphere eXtreme Scale in a stand-alone environment. Specifies the port number on which the MBean server listens for communication with Java Management Extensions (JMX).

Default: `1099` for catalog servers

-jvmArgs <JVM arguments>
Specifies a set of JVM arguments. Every option after the **-jvmArgs** option is used to start the server Java virtual machine (JVM). When the **-jvmArgs** parameter is used, ensure that it is the last optional script argument specified.

Example:-jvmArgs `-Xms256M -Xmx1G`

-listenerHost <host name>

Specifies the host name to which the Object Request Broker (ORB) transport protocol binds for communication. The value must be a fully qualified domain name or IP address. If your configuration involves multiple network cards, set the listener host and port to the IP address for which to bind. By setting the listener and host port, it allows the transport mechanism in the JVM know which IP address to use. If you do not specify which IP address to use, symptoms such as connection timeouts, unusual API failures, and clients that seem to hang can occur.**Default:** `localhost`

-listenerPort <port>
Specifies the port number to which the ORB transport protocol binds for communication. **Default:** `2809`

-quorum true|false

Enables quorum for the catalog service. Quorum is used to ensure that most of the catalog service domain is available before partitions are moved to the available container servers. To enable quorum, set the value to `true` or `enabled`. The default value is `disabled`. This property applies to the catalog service only. For more information, see Catalog server quorums.

-script <script file>
Specifies the location of a custom script for commands you specify to start catalog servers or containers and then parameterize or edit as you require.

-serverProps <server properties file>
Specifies the server property file that contains the server-specific security properties. The file name specified for this property is just in plain file path format, such as `c:/tmp/og/catalogserver.props`.

-traceSpec <trace specification>

Enables trace and the trace specification string for the container server. Trace is disabled by default. This property applies to both the container server and the catalog service. Examples:
- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <trace file>

Specifies a file name to write trace information. This property applies to both the container server and the catalog service.

`-timeout <seconds>`
Specifies a number of seconds before the server start times out.

Usage for container servers 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]
```

Options for container servers

`-catalogServiceEndPoints <hostName:port,hostName:port>`
Specifies the Object Request Broker (ORB) host and port on the catalog service.

Default: localhost:2809

`-deploymentPolicyFile <deployment policy xml file>`
Specifies the path to the deployment policy file on the hard disk. The deployment policy is used to set up partitioning and replication. The deployment policy can also be used to influence placement behavior.

Example: `../xml/SimpleDP.xml`

`-deploymentPolicyUrl <deployment policy url>`
Specifies the URL for the deployment policy file on the hard disk or on the network. The deployment policy is used to set up partitioning and replication. The deployment policy can also be used to influence placement behavior.

Example: `file://xml/SimpleDP.xml`

`-JMXConnectorPort <port>`
Defines the Secure Sockets Layer (SSL) port to which the Java Management Extensions (JMX) service binds. Only required if an SSL transport protocol is needed for JMX data.

`-JMXServicePort <port>`

Required only for WebSphere eXtreme Scale in a stand-alone environment. Specifies the port number on which the MBean server listens for communication with Java Management Extensions (JMX). **Default:** `1099`

`-jvmArgs <JVM arguments>`
Specifies a set of JVM arguments. Every option after the **`-jvmArgs`** option is used to start the server Java virtual machine (JVM). When the **`-jvmArgs`** parameter is used, ensure that it is the last optional script argument specified.

Example:`-jvmArgs` `-Xms256M -Xmx1G`

`-listenerHost <host name>`

Specifies the host name to which the Object Request Broker (ORB) transport protocol binds for communication. The value must be a fully qualified domain name or IP address. If your configuration involves multiple network cards, set the listener host and port to the IP address for which to bind. By setting the listener and host port, it allows the transport mechanism in the JVM know which IP address to use. If you do not specify which IP address to use, symptoms such as connection timeouts, unusual API failures, and clients that seem to hang can occur.**Default:** `localhost`

`-listenerPort <port>`
Specifies the port number to which the ORB transport protocol binds for communication. **Default:** `2809`

`-objectgridFile <ObjectGrid descriptor xml file>`
Specifies the path to the ObjectGrid descriptor file. The ObjectGrid XML file specifies which eXtreme Scale servers the container hosts.

`-objectgridUrl <ObjectGrid descriptor url>`
Specifies a URL for the ObjectGrid descriptor file. The ObjectGrid XML file specifies which eXtreme Scale servers the container hosts.

`-script <script file>`
Specifies the location of a custom script for commands you specify to start catalog servers or containers and then parameterize or edit as you require.

`-serverProps <server properties file>`
Specifies the path to the server property file.

Example:`../security/server.props`

`-timeout <seconds>`
Specifies a number of seconds before the server start times out.

`-traceFile <trace file>`

Specifies a file name to write trace information. This property applies to both the container server and the catalog service.

`-traceSpec <trace specification>`

Enables trace and the trace specification string for the container server. Trace is disabled by default. This property applies to both the container server and the catalog service. Examples:
- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

`-zone <zone name>`
Specifies the zone to use for all of the containers within the server. See “Zone-preferred routing” on page 283 the information about zones in the *Product Overview* for more information about configuring zones.

Related concepts:

“Statistics modules” on page 530
WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics.

“Tuning Java virtual machines” on page 571
You must take into account several specific aspects of Java virtual machine (JVM) tuning for WebSphere eXtreme Scale best performance. In most cases, few or no special JVM settings are required. If many objects are being stored in the data grid, adjust the heap size to an appropriate level to avoid running out of memory.

“Java SE considerations” on page 60
WebSphere eXtreme Scale requires Java SE 5, Java SE 6, or Java SE 7. In general,

newer versions of Java SE have better functionality and performance.

"Java EE considerations" on page 62
As you prepare to integrate WebSphere eXtreme Scale in a Java Platform, Enterprise Edition environment, consider certain items, such as versions, configuration options, requirements and limitations, and application deployment and management.

"Tuning garbage collection with WebSphere Real Time" on page 577
Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary. WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

Related tasks:

"Starting and stopping stand-alone servers" on page 459
You can start and stop stand-alone catalog and container servers with scripts or the embedded server API.

7.1.1+ "Monitoring with CSV files" on page 525
You can enable monitoring data collected for a container server to be written to comma-separated values (CSV) files. These CSV files can contain information about the Java virtual machine (JVM), map, or ObjectGrid instance.

"Enabling statistics" on page 529
WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics. You can use several methods to retrieve the information from the statistics modules.

"Monitoring with the statistics API" on page 532
The Statistics API is the direct interface to the internal statistics tree. Statistics are disabled by default, but can be enabled by setting a StatsSpec interface. A StatsSpec interface defines how WebSphere eXtreme Scale should monitor statistics.

"Administering with the **`xscmd`** utility" on page 482
With the **`xscmd`** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related information:

"Getting started tutorial lesson 3.1: Starting catalog and container servers" on page 5
To run the sample client application, you must start a catalog server and a container server.

StatsSpec class

Tuning the IBM virtual machine for Java

Stopping stand-alone servers

You can use the `stopOgServer` script to stop eXtreme Scale server processes.

About this task

Run the **stopOgServer** script by navigating to the `bin` directory:

```
cd wxs_install_root/bin
```

Procedure

- **Stop a single container server.**

  Run the **stopOgServer** script to stop the container server. Use this command only when you are stopping a single container server. If you run the single catalog server stop command on several container servers in succession, you might see performance and churn issues for shard placement.

  ```
  stopOgServer containerServer -catalogServiceEndPoints MyServer1.company.com:2809
  ```

  Attention: The **-catalogServiceEndPoints** option should match the value of the **-catalogServiceEndPoints** option that was used to start the container. If a **-catalogServiceEndPoints** was not used to start the container, the default values are likely localhost or the hostname and 2809 for the ORB port to connect to the catalog service. Otherwise, use the values that are passed to **-listenerHost** and **-listenerPort** on the catalog service. If the **-listenerHost** and **-listenerPort** options are not used when starting the catalog service, the ORB binds to port 2809 on the localhost for the catalog service.
- **Stop multiple container servers.**

  To prevent churn and performance issues for shard placement when you want to stop multiple container servers at the same time, use the following command format. Separate a list of container servers with commas:

  ```
  stopOgServer containerServer0,containerServer1,containerServer2
  -catalogServiceEndPoints MyServer1.company.com:2809
  ```

  If you want to stop all of the containers on a specific zone or host, you can use the **-teardown** parameter. See "Stopping servers gracefully with the **xscmd** utility" on page 474 for more information.
- **Stop catalog servers.**

  Run the **stopOgServer** script to stop the catalog server.

  ```
  stopOgServer.sh catalogServer -catalogServiceEndPoints MyServer1.company.com:2809
  ```

  Attention: When you are stopping a catalog service, use the **-catalogServiceEndPoints** option to reference the Object Request Broker (ORB) host and port on the catalog service. The catalog service uses **-listenerHost** and **-listenerPort** options to specify the host and port for ORB binding or accepts the default binding. If the **-listenerHost** and **-listenerPort** options are not used when starting the catalog service, the ORB binds to port 2809 on the localhost for the catalog service. The **-catalogServiceEndPoints** option is different when stopping a catalog service than when you started the catalog service.

  Starting a catalog service requires peer access ports and client access ports, if the default ports were not used. Stopping a catalog service requires only the ORB port.
- **Stop the web console server.** To stop the web console server, run the **stopConsoleServer.bat|sh** script. This script is in the *wxs_install_root*/`ObjectGrid/bin` directory of your installation. For more information, see "Starting and logging on to the web console" on page 514.
- **Enable trace for the server stop process.**

If a container fails to stop, you can enable trace to help with debugging the problem. To enable trace during the stop of a server, add the **-traceSpec** and **-traceFile** parameters to the stop commands. The **-traceSpec** parameter specifies the type of trace to enable and the **-traceFile** parameter specifies path and name of the file to create and use for the trace data.

1. From the command line, navigate to the bin directory.

   ```
   cd wxs_install_root/bin
   ```

2. Run the **stopOgServer** script with trace enabled.

   ```
   stopOgServer.sh c4 -catalogServiceEndPoints MyServer1.company.com:2809
    -traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
   ```

   After the trace is obtained, look for errors related to port conflicts, missing classes, missing or incorrect XML files or any stack traces. Suggested startup trace specifications are:

   – `ObjectGrid=all=enabled`

   – `ObjectGrid*=all=enabled`

   For all of the trace specification options, see "Server trace options" on page 621.

- **Stop embedded servers programmatically.**

  For more information about stopping embedded servers programmatically, see "Using the embedded server API to start and stop servers" on page 476.

Related concepts:

"Hardware and software requirements" on page 59
Browse an overview of hardware and operating system requirements. Although you are not required to use a specific level of hardware or operating system for WebSphere eXtreme Scale, formally supported hardware and software options are available on the Systems Requirements page of the product support site. If a conflict exists between the information center and the System Requirements page, the information at the website takes precedence. Prerequisite information in the information center is provided as a convenience only.

Stopping servers gracefully with the `xscmd` utility

You can use the **xscmd** utility with the **-c teardown** command to stop a list or group of catalog and container servers. This command simplifies shutting down all or portions of a data grid. It also prevents unnecessary placement and recovery catalog service actions that normally occur when processes are stopped ungracefully.

Procedure

- Stop a specific list of servers.

  Provide a list of servers after the **-teardown** parameter:

  ```
  xscmd –c teardown -sl catalogServer1,catalogServer2,containerServer1
  ```

- Stop all the servers in a specific zone.

  Use the **-z** parameter and provide the name of the zone. The catalog server determines the servers that are running in the zone. The **xscmd** utility also prompts you with a list of the servers in the selected zone before the servers are shut down.

  ```
  xscmd –c teardown –z zone_name
  ```

- Stop all the servers on a specific host. For example, to shut down all the servers on `myhost.mycompany.com`, enter `-hf myhost.mycompany.com`.

Use the **-hf** parameter and provide the name of the host. The catalog server determines the servers that are running on the host. The **xscmd** utility prompts you with a list of the servers in the selected host before the servers are shut down.

```
xscmd –teardown –hf <host_name>
```

Attention: By default, the JVM continues to run when each eXtreme Scale server in an OSGi framework is stopped in the **xscmd** utility with the **-c teardown** command. If you want eXtreme Scale to exit the JVM, then this type of implementation must be planned for. You must set the server property **exitJVMOnTeardown** to `true` before the server is started. For more information, see Server properties file.

Related reference:

xscmd utility reference
You can use the following list of commands as a reference when you are using the **xscmd** utility.

xsadmin utility reference
You can pass arguments to the **xsadmin** utility with two different methods: with a command-line argument, or with a properties file.

Starting and stopping servers in a WebSphere Application Server environment

Catalog and container servers can automatically start in a WebSphere Application Server or WebSphere Application Server Network Deployment environment.

Before you begin

Configure catalog servers and container servers to run on WebSphere Application Server:

- "Configuring the catalog service in WebSphere Application Server" on page 298
- "Configuring container servers in WebSphere Application Server" on page 325

About this task

The life cycle of catalog and container servers in WebSphere Application Server is linked to the processes on which these servers run.

Procedure

- **Start catalog services in WebSphere Application Server**:

  The life cycle a catalog server is tied to the WebSphere Application Server process. After you configure the catalog service domain in WebSphere Application Server, restart each server that you defined as a part of the catalog service domain. The catalog service starts automatically on the servers that you associated with the catalog service domain. The catalog service can also start automatically in the following scenarios, depending on the edition of WebSphere Application Server:

  – **Base WebSphere Application Server**: You can configure your application to automatically start a container server and catalog service. This feature simplifies unit testing in development environments such as Rational Application Developer because you do not need to explicitly start a catalog service. See "Configuring WebSphere Application Server applications to automatically start container servers" on page 327 for more information.

  - **WebSphere Application Server Network Deployment**: The catalog service automatically starts in the deployment manager process if the deployment manager node has WebSphere eXtreme Scale installed and the deployment manager profile is augmented. See "Configuring the catalog service in WebSphere Application Server" on page 298 for more information.
- **Start container servers in WebSphere Application Server**:

  The life cycle of a container server is tied to the WebSphere Application Server application. When you start the configured application, the container servers also start.
- **Stop an entire data grid of servers**:

  You can stop catalog and container servers by stopping the applications and associated application servers. However, you can also stop an entire data grid with the **`xscmd`** utility or MBeans:
  - **In the `xscmd` utility:**

    See "Stopping servers gracefully with the **`xscmd`** utility" on page 474 for more information about stopping an entire data grid.
  - **With Mbeans:**

    Use the tearDownServers operation on the PlacementServiceMBean Mbean.

Related concepts:

"Configuration considerations for multi-master topologies" on page 41
Consider the following issues when you are deciding whether and how to use multi-master replication topologies.

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Related reference:

"Catalog service domain administrative tasks" on page 301
You can use the Jacl or Jython scripting languages to manage catalog service domains in your WebSphere Application Server configuration.

Using the embedded server API to start and stop servers

With WebSphere eXtreme Scale, you can use a programmatic API for managing the life cycle of embedded servers and containers. You can programmatically configure the server with any of the options that you can also configure with the command line options or file-based server properties. You can configure the embedded server to be a container server, a catalog service, or both.

Before you begin

- You must have a method for running code from within an already existing Java virtual machine. The eXtreme Scale classes must be available through the class loader tree.
- If your container servers are using IBM eXtremeMemory, you must first configure the native libraries. For more information, see "Configuring IBM eXtremeMemory" on page 346.

About this task

You can run many administration tasks with the Administration API. One common use of the API is as an internal server for storing Web application state. The Web server can start an embedded WebSphere eXtreme Scale server, report the container server to the catalog service, and the server is then added as a member of a larger distributed grid. This usage can provide scalability and high availability to an otherwise volatile data store.

You can programmatically control the complete life cycle of an embedded eXtreme Scale server. The examples are as generic as possible and only show direct code examples for the outlined steps.

Procedure

1. Obtain the ServerProperties object from the ServerFactory class and configure any necessary options.

   Every eXtreme Scale server has a set of configurable properties. When a server starts from the command line, those properties are set to defaults, but you can override several properties by providing an external source or file. In the embedded scope, you can directly set the properties with a ServerProperties object. You must set these properties before you obtain a server instance from the ServerFactory class. The following example snippet obtains a ServerProperties object, sets the CatalogServiceBootStrap field, and initializes several optional server settings. See the API documentation for a list of the configurable settings.

   ```
   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. If you want the server to be a catalog service, obtain the CatalogServerProperties object.

   Every embedded server can be a catalog service, a container server, or both a container server and a catalog service. The following example obtains the CatalogServerProperties object, enables the catalog service option, and configures various catalog service settings.

   ```
   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. Obtain a Server instance from the ServerFactory class. The Server instance is a process-scoped singleton that is responsible for managing the membership in the grid. After this instance has been instantiated, this process is connected and is highly available with the other servers in the grid. The following example shows how to create the Server instance:

   ```
    Server server = ServerFactory.getInstance();
   ```

   Reviewing the previous example, the ServerFactory class provides a static method that returns a Server instance. The ServerFactory class is intended to be the only interface for obtaining a Server instance. Therefore, the class ensures that the instance is a singleton, or one instance for each JVM or isolated classloader. The getInstance method initializes the Server instance. You must configure all the server properties before you initialize the instance. The Server class is responsible for creating new Container instances. You can use both the ServerFactory and Server classes for managing the life cycle of the embedded Server instance.

4. Start a Container instance using the Server instance.

   Before shards can be placed on an embedded server, you must create a container on the server. The Server interface has a createContainer method that takes a DeploymentPolicy argument. The following example uses the server instance that you obtained to create a container using a created DeploymentPolicy file. Note that Containers require a classloader that has the application binaries available to it for serialization. You can make these binaries available by calling the createContainer method with the Thread context classloader set to the classloader that you want to use.

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

5. Remove and clean up a container server.

   You can remove and clean up a container server by using the running the teardown method on the obtained Container instance. Running the teardown method on a container properly cleans up the container and removes the container from the embedded server.

   The process of cleaning up the container includes the movement and tearing down of all the shards that are placed within that container. Each server can contain many containers and shards. Cleaning up a container does not affect the life cycle of the parent Server instance. The following example demonstrates how to run the teardown method on a server. The teardown method is made available through the ContainerMBean interface. By using the ContainerMBean interface, if you no longer have programmatic access to this container, you can still remove and clean up the container with its MBean. A terminate method also exists on the Container interface, do not use this method unless it is absolutely needed. This method is more forceful and does not coordinate appropriate shard movement and clean up.

   ```
   container.teardown();
   ```

6. Stop the embedded server.

   When you stop an embedded server, you also stop any containers and shards that are running on the server. When you stop an embedded server, you must clean up all open connections and move or tear down all the shards. The following example demonstrates how to stop a server and using the waitFor method on the Server interface to ensure that the Server instance shuts down completely. Similarly to the container example, the stopServer method is made available through the ServerMBean interface. With this interface, you can stop a server with the corresponding Managed Bean (MBean).

   ```
   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
   ```

   Full code example:

   ```
    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)
        }

    }

}
```

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Embedded server API

WebSphere eXtreme Scale includes application programming interfaces (APIs) and system programming interfaces for embedding eXtreme Scale servers and clients within your existing Java applications.

Instantiate the eXtreme Scale server

You can use several properties to configure the eXtreme Scale server instance, which you can retrieve from the ServerFactory.getServerProperties method. The ServerProperties object is a singleton, so each call to the getServerProperties method retrieves the same instance.

You can create a server with the following code.

```
Server server = ServerFactory.getInstance();
```

All properties set before the first invocation of the getInstance method are used to initialize the server.

Set server properties

You can set the server properties until the ServerFactory.getInstance method is called for the first time. The first call of the getInstance method instantiates the eXtreme Scale server, and reads all the configured properties. Setting the properties after creation has no effect. The following example shows how to set properties before instantiating a Server instance.

```
// 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 listener host name to use to bind to.
serverProperties.setListenerHost("host.local.domain");

// Set the listener port to use to bind to.
serverProperties.setListenerPort(9010);

// Turn off all MBeans for this process.
serverProperties.setMBeansEnabled(false);

Server server = ServerFactory.getInstance();
```

Embed the catalog service

Any JVM setting that is flagged by the CatalogServerProperties.setCatalogServer method can host the catalog service for eXtreme Scale. This method indicates to the eXtreme Scale server run time to instantiate the catalog service when the server is started. The following code shows how to instantiate the eXtreme Scale catalog server:

```
CatalogServerProperties catalogServerProperties =
 ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();
```

Embed a container server

Run the Server.createContainer method for any JVM to host multiple eXtreme Scale container servers. The following code shows how to instantiate a container server:

```
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);
```

Self-contained server process

You can start all the services together, which is useful for development and also practical in production. By starting the services together, a single process does all of the following actions: starts the catalog service, starts a set of containers, and runs the client connection logic. Starting the services in this way sorts out programming issues before deploying in a distributed environment. The following code shows how to instantiate a self-contained eXtreme Scale server:

```
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);
```

Embed eXtreme Scale in WebSphere Application Server

The configuration for eXtreme Scale is set up automatically when you install eXtreme Scale in a WebSphere Application Server environment. You are not required to set any properties before you access the server to create a container. The following code shows how to instantiate an eXtreme Scale server inWebSphere Application Server:

```
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);
```

For a step by step example on how to start an embedded catalog service and container programmatically, see “Using the embedded server API to start and stop servers” on page 476.

Related tasks:

“Configuration methods” on page 257
You can configure most aspects of the product with XML files and property files. You can also use programmatic methods, including application and system programming interfaces, plug-ins, and managed beans.

“Configuring data grids” on page 261
Use an ObjectGrid descriptor XML file to configure data grids, backing maps, plug-ins, and so on. To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API. For a distributed topology, you need an ObjectGrid descriptor XML file and a deployment policy XML file.

Connecting to distributed ObjectGrid instances programmatically
You can connect to a distributed ObjectGrid with the connection end points for the catalog service domain. You must have the host name and listener port of each catalog server in the catalog service domain to which you want to connect.

"Configuring deployment policies" on page 277
Use the deployment policy descriptor XML file and the objectgrid descriptor XML file to manage a distributed topology. The deployment policy is encoded as an XML file that is provided to the container server. The deployment policy provides information about maps, map sets, partitions, replicas, and so on. It also controls shard placement behaviors.

"Installing eXtreme Scale bundles" on page 225
WebSphere eXtreme Scale includes bundles that can be installed into an Eclipse Equinox OSGi framework. These bundles are required to start eXtreme Scale servers or use eXtreme Scale clients in OSGi. You can install the eXtreme Scale bundles using the Equinox console or using the config.ini configuration file.

Installing the Liberty profile developer tools for WebSphere eXtreme Scale

Related reference:

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Related information:

ObjectGridManager interface

ClientClusterContext interface

DeploymentPolicy interface

Administering with the `xscmd` utility

With the **`xscmd`** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Before you begin

- Your catalog servers and container servers must be started. If your catalog servers are in a catalog service domain, at least two catalog servers must be started.
- Verify that the *JAVA_HOME* environment variable is set to use the runtime environment that installed with the product. If you are using the trial version of the product, you must set the *JAVA_HOME* environment variable.

About this task

The **`xscmd`** utility replaces the **`xsadmin`** sample utility as a fully supported monitoring and administration tool. You could complete similar operations with the **`xsadmin`** tool, but this tool is not supported. The **`xsadmin`** sample provides a method for parsing and discovering current deployment data, and can be used as a foundation for writing custom utilities. If you were previously using the **`xsadmin`** tool for monitoring and administration, consider updating your scripts to use the

xscmd utility. For information about mapping **xsadmin** commands to the new **xscmd** commands, see "**xsadmin** tool to **xscmd** tool migration" on page 249.

Procedure

1. Optional: If client authentication is enabled: Open a command-line window. On the command line, set appropriate environment variables.
2. Go to the *wxs_home*/`bin` directory.

   `cd` *`wxs_home`*`/bin`
3. Display help for the various **xscmd** options.
   - To display the general help, run the following command:
     - UNIX `./xscmd.sh -h`
     - Windows `xscmd.bat -h`
   - To display a list of all of the commands, run the following command:
     - UNIX `./xscmd.sh -lc`
     - Windows `xscmd.bat -lc`
   - To display the help for a specific command, run the following command:
     - UNIX `./xscmd.sh -h` *`command_name`*
     - Windows `xscmd.bat -h` *`command_name`*
   - To display a list of the command groups, run the following command:
     - UNIX `./xscmd.sh -lcg`
     - Windows `xscmd.bat -lcg`
   - To display a list of the commands within a command group, run the following command:
     - UNIX `./xscmd.sh -lc` *`command_group_name`*
     - Windows `xscmd.bat -lc` *`command_group_name`*
4. Run commands that connect to specific catalog servers. By default, **xscmd** connects to the catalog server on the local host, using the host name and port of `localhost:2809`. You can also provide a list of host names and ports to the command so that you can connect to catalog servers on other hosts. From the list, the **xscmd** utility connects to a random host. The list of hosts that you provide must be within the same catalog service domain.
   - Provide a list of stand-alone catalog servers to connect:
     - UNIX `./xscmd.sh -c <`*`command_name`*`> -cep` *`hostname:port`*`(,`*`hostname:port`*`)`
     - Windows `xscmd.bat -c <`*`command_name`*`> -cep` *`hostname:port`*`(,`*`hostname:port`*`)`

     In the previous commands, *command_name* is the name of the command that you are running. The *hostname:port* value is the catalog server host name and listener port. The listener port value on a stand-alone catalog server is specified when you run the **startOgServer** command.
   - Provide a list of WebSphere Application Server catalog servers to connect. You cannot connect to catalog servers that are running on WebSphere Application Server with the default localhost value:
     - UNIX `./xscmd.sh -c <`*`command_name`*`> -cep` *`was_hostname:port`*`(,`*`hostname:port`*`)`
     - Windows `xscmd.bat -c <`*`command_name`*`> -cep` *`was_hostname:port`*`(,`*`hostname:port`*`)`

In the previous commands, *command_name* is the name of the command that you are running. The *was_hostname* value is the host name of the catalog server in the WebSphere Application Server cell. The *port* value is the listener port. The listener port value in WebSphere Application Server is inherited by the BOOTSTRAP_ADDRESS port configuration. The default value is `9809` if the catalog server is running on the deployment manager. If you are running the catalog server on an application server, check the BOOTSTRAP_ADDRESS port configuration of the application server to determine the port number.

Important: If your container servers are running in a secured WebSphere Application Server environment, run the **`xscmd`** utility from the WebSphere eXtreme Scale Client installation in the WebSphere Application Server environment. For example, from the `/opt/IBM/WebSphere/AppServer/bin` directory.

Related concepts:

IBM eXtremeMemory
IBM eXtremeMemory enables objects to be stored in native memory instead of the Java heap. By moving objects off the Java heap, you can avoid garbage collection pauses, leading to more constant performance and predicable response times.

Zones
Zones give you control over shard placement. Zones are user-defined logical groupings of physical servers. The following are examples of different types of zones: different blade servers, chassis of blade servers, floors of a building, buildings, or different geographical locations in a multiple data center environment. Another use case is in a virtualized environment where many server instances, each with a unique IP address, run on the same physical server.

"Zone-preferred routing" on page 283
With zone-preferred routing, you can define how WebSphere eXtreme Scale directs transactions to zones.

Catalog server quorums
When the quorum mechanism is enabled, all the catalog servers in the quorum must be available for placement operations to occur in the data grid.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

"Planning for network ports" on page 51
WebSphere eXtreme Scale servers require several ports to operate.

"Statistics modules" on page 530
WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics.

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Chapter 7, "Administering," on page 459

Related reference:

"Web console statistics" on page 519
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **stopOgServer** script stops catalog and container servers.

"Example: Zone definitions in the deployment policy descriptor XML file" on page 288
You can specify zones and zone rules with the deployment policy descriptor XML file.

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

xscmd utility reference
You can use the following list of commands as a reference when you are using the **xscmd** utility.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

"**xsadmin** tool to **xscmd** tool migration" on page 249
In previous releases, the **xsadmin** tool was a sample command-line utility to monitor the state of the environment. The **xscmd** tool has been introduced as an officially supported administrative and monitoring command-line tool. If you were previously using the **xsadmin** tool, consider migrating your commands to the new **xscmd** tool.

"Server trace options" on page 621
You can enable trace to provide information about your environment to IBM support.

Messages
When you encounter a message in a log or other parts of the product interface, you can look up the message by its component prefix to find out more information.

Related information:

"Getting started tutorial lesson 4: Monitor your environment" on page 7
You can use the **xscmd** utility and web console tools to monitor your data grid environment.

"Module 5: Use the **xscmd** utility to monitor data grids and maps" on page 151
You can use the **xscmd** utility to show the primary data grids and map sizes of the `Grid` data grid. The **xscmd** tool uses the MBean to query all of the data grid artifacts, such as primary shards, replica shards, container servers, map sizes, and other data.

"Module 5: Use the **xscmd** tool to monitor data grids and maps" on page 127
You can use the **xscmd** tool to show the primary data grids and map sizes of the `Grid` data grid. The **xscmd** tool uses the MBean to query all of the data grid artifacts, such as primary shards, replica shards, container servers, map sizes, and so on.

Interface PlacementServiceMBean

Eclipse runtime options

Port number settings in WebSphere Application Server versions

StatsSpec class

Controlling placement

You can use several different options to control when shards are placed on various container servers in the configuration. During startup, you might choose to delay the placement of shards. When you are running all of your container servers, you might need to suspend, resume, or change placement while you maintain servers.

Procedure

Controlling placement during startup
You can control when shards begin to be placed while your environment is starting. Some control is in place by default. If you do not take any actions to control placement, shards begin to be placed immediately. When shards are placed immediately, the shards might not be placed evenly as subsequent container servers start, and further placement operations run to balance the distribution.

- Temporarily suspend the balancing of shards to prevent immediate shard placement when your container servers are starting.

  Suspending the balancing of shards prevents uneven shard placement. Before you start your container servers, use the **`xscmd -c suspendBalancing`** command to stop the balancing of shards for a specific data grid and map set. After the container servers are started, you can use the **`xscmd -c resumeBalancing`** command to begin the placement of shards on the container servers.
- 7.1.1+ Configure the **`placementDeferralInterval`** property to minimize the number of shard placement cycles on the container servers. Shard placement is triggered at the defined time interval.

  7.1.1+ **placementDeferralInterval**

  Specifies the interval in milliseconds for deferring the balancing and placement of shards on the container servers. Placement does not start until after the time specified in the property has passed. Increasing the deferral interval lowers processor utilization, but the placement of work items is completed over time. A decrease in the deferral interval increases short-term processor usage, but the placement of work items is more immediate and expedited.

  If multiple container servers are starting in succession, the deferral interval timer is reset if a new container server starts within the given interval. For example, if a second container server starts 10 seconds after the first container server, placement does not start until 15 seconds after the second container server started. However, if a third container server starts 20 seconds after the second container server, placement has already begun on the first two container servers.

  When container servers become unavailable, placement is triggered as soon as the catalog server learns of the event so that recovery can occur as quickly as possible.

  Default: 15000 ms (15 seconds)

  You can use the following tips to help determine if your placement deferral value is set to the right amount of time:

  - As you concurrently start the container servers, look at the CWOBJ1001 messages in the `SystemOut.log` file for each container server. The timestamp of these messages in each container server log file indicates the actual container server start time. You might consider adjusting the **placementDeferralInterval** property to include more container server starts. For example, if the first container server starts 90 seconds before the last container server, you might set the property to 90 seconds.
  - Note how long the CWOBJ1511 messages occur after the CWOBJ1001 messages. This amount of time can indicate if the deferral has occurred successfully.
  - If you are using a development environment, consider the length of the interval when you are testing your application.
- Configure the **numInitialContainers** attribute.

  If you previously used the **numInitialContainers** attribute, you can continue using the attribute. However, the use of the **xscmd -c suspendBalancing** and **xscmd -c resumeBalancing** commands followed by the **placementDeferralInterval** are suggested over the **numInitialContainers** attribute to control placement. The **numInitialContainers** attribute specifies the number of container servers that are required before initial placement occurs for the shards in this mapSet element. The **numInitialContainers** attribute is in the deployment policy descriptor XML file. If you have both **numInitialContainers** and **placementDeferralInterval** set, note that until the **numInitialContainers** value has been met, no placement occurs, regardless of the value of the **placementDeferralInterval** property.

Controlling placement after initial startup

- Force placement to occur.

  You can use the **xscmd -c triggerPlacement -g my_OG -ms my_Map_Set** command, where *my_OG* and *my_Map_Set* are set to values for your data grid and map set, to force placement to occur during a point in time at which placement might not occur otherwise. For example, you might run this command when the amount of time specified by the **placementDeferralInterval** property has not yet passed or when balancing is suspended.
- Reassign a primary shard.

  Use the **xscmd -c swapShardWithPrimary** command to assign a replica shard to be the new primary shard. The previous primary shard becomes a replica.
- Rebalance the primary and replica shards.

  Use the **xscmd -c balanceShardTypes** command to adjust the ratio of primary and replica shards to be equitable among the running container servers in the configuration. The ratio is consistent within one shard on each container server.
- Suspend or resume placement.

  Use the **xscmd -c suspendBalancing** command or the **xscmd -c resumeBalancing** command to stop and start the balancing of shards for a specific data grid and map set. When balancing has been suspended, the following placement actions can still run:
  - Shard promotion can occur when container servers fail.
  - Shard role swapping with the **xscmd -c swapShardWithPrimary** command.
  - Shard placement triggered balancing with the **xscmd -c triggerPlacement -g myOG -ms myMapSet** command.

What to do next

You can monitor the placement in the environment with the **`xscmd -c placementServiceStatus`** command.

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Container servers, partitions, and shards
The container server stores application data for the data grid. This data is generally broken into parts, which are called partitions. Partitions are hosted across multiple shard containers. Each container server in turn hosts a subset of the complete data. A JVM might host one or more shard containers and each shard container can host multiple shards.

Related reference:

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

"Configuring distributed deployments" on page 278
Use the deployment policy descriptor XML file and the ObjectGrid descriptor XML file to manage your topology.

ObjectGrid descriptor XML file
To configure WebSphere eXtreme Scale, use an ObjectGrid descriptor XML file and the ObjectGrid API.

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

Related information:

Interface PlacementServiceMBean

Managing ObjectGrid availability

The availability state of an ObjectGrid instance determines which requests can be processed at any particular time. You can use the StateManager interface to set and retrieve the state of an ObjectGrid instance.

About this task

Four availability states exist for a given ObjectGrid instance.

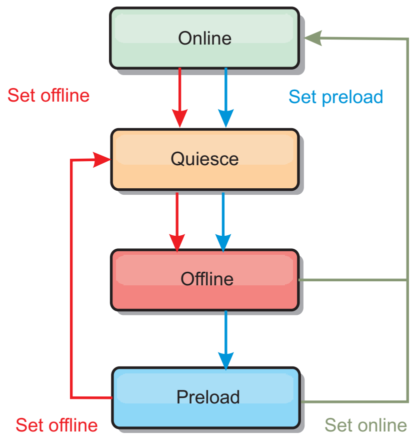


Figure 50. Availability states of an ObjectGrid instance

ONLINE
: The ONLINE state is the default availability state for an ObjectGrid. An ONLINE ObjectGrid is able to process any requests from a typical eXtreme Scale client. However, requests from a preload client are rejected while the ObjectGrid is ONLINE.

QUIESCE
: The QUIESCE state is transitional. An ObjectGrid that is in QUIESCE is soon moved to the OFFLINE state. While in the QUIESCE state, an ObjectGrid is allowed to process outstanding transactions. However, any new transactions are rejected. An ObjectGrid can remain in QUIESCE for up to 30 seconds. After this time, the availability state is moved to OFFLINE.

OFFLINE
: The OFFLINE state results in the rejection of all transactions that are sent to the ObjectGrid.

PRELOAD
: The PRELOAD state can be used to load data into an ObjectGrid from a preload client. While the ObjectGrid is in the PRELOAD state, only a preload client can commit transactions against the ObjectGrid. All other transactions are rejected.

A request is rejected if an ObjectGrid is not in the appropriate availability state to support that request. An AvailabilityException exception results whenever a request is rejected.

Procedure

1. Set the initial state of an ObjectGrid with the ObjectGrid configuration XML file.

   You can use the **`initialState`** attribute on an ObjectGrid to indicate its startup state. Normally, when an ObjectGrid completes initialization, it is available for routing. The state can later be changed to prevent traffic from routing to an ObjectGrid. If the ObjectGrid needs to be initialized, but not immediately available, you can use the **`initialState`** attribute.

   The initialState attribute is set on the ObjectGrid configuration XML file. The default state is ONLINE. Valid values include:

   - ONLINE (default)
   - PRELOAD

- OFFLINE

See ObjectGrid descriptor XML file for more information about the **initialState** attribute.

If the initialState attribute is set on an ObjectGrid, the state must be explicitly set back to online or the ObjectGrid will remain unavailable. An AvailabilityException exception results if the ObjectGrid is not in the ONLINE state.

See AvailabilityState API documentation for more information.

Using the initialState attribute for preloading

If the ObjectGrid is preloaded with data, there can be a period of time between when the ObjectGrid is available and switching to a preload state to block client traffic. To avoid this time period, the initial state on an ObjectGrid can be set to PRELOAD. The ObjectGrid still completes all the necessary initialization, but it blocks traffic until the state has changed and allows the preload to occur.

The PRELOAD and OFFLINE states both block traffic, but you must use the PRELOAD state if you want to initiate a preload.

Failover and balancing behavior

If a replica data grid is promoted to be a primary data grid, the replica does not use the **initialState** setting. If the primary data grid is moved for a rebalance, the **initialState** setting is not used because the data is copied to the new primary location before the move is completed. If replication is not configured, then the primary goes into the **initialState** setting if failover occurs, and a new primary must be placed.

2. Change the availability state with the StateManager interface.

   Use the StateManager interface to set the availability state of an ObjectGrid. To set the availability state of an ObjectGrid running on the servers, pass a corresponding ObjectGrid client to the StateManager interface. The following code demonstrates how to change the availability state of an ObjectGrid.

```
ClientClusterContext client = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

   Each shard of the ObjectGrid transitions to the desired state when the setObjectGridState method is called on the StateManager interface. When the method returns, all shards within the ObjectGrid should be in the proper state.

   Use an ObjectGridEventListener plug-in to change the availability state of a server side ObjectGrid. Only change the availability state of a server-side ObjectGrid when the ObjectGrid has a single partition. If the ObjectGrid has multiple partitions, the shardActivated method is called on each primary, which results in superfluous calls to change the state of the ObjectGrid

```
public class OGListener implements ObjectGridEventListener,
   ObjectGridEventGroup.ShardEvents {
    public void shardActivated(ObjectGrid grid) {
            StateManager stateManager = StateManagerFactory.getStateManager();
            stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

   Because QUIESCE is a transitional state, you cannot use the StateManager interface to put an ObjectGrid into the QUIESCE state. An ObjectGrid passes through this state on its way to the OFFLINE state.

3. Retrieve the availability state.

Use the getObjectGridState method of the StateManager interface to retrieve the availability state of a particular ObjectGrid.

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

The getObjectGridState method chooses a random primary within the ObjectGrid and returns its AvailabilityState. Because all shards of an ObjectGrid should be in the same availability state or transitioning to the same availability state, this method provides an acceptable result for the current availability state of the ObjectGrid.

Related reference:

Plug-ins for providing event listeners
You can use the ObjectGridEventListener, MapEventListener, ObjectGridLifecycleListener and BackingMapLifecycleListener plug-ins to configure notifications for various events in the eXtreme Scale cache. Listener plug-ins are registered with an ObjectGrid or BackingMap instance like other eXtreme Scale plug-ins and add integration and customization points for applications and cache providers.

Managing data center failures

About this task

Related concepts:

Catalog server quorums
When the quorum mechanism is enabled, all the catalog servers in the quorum must be available for placement operations to occur in the data grid.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Managing data center failures when quorum is enabled

When the data center enters a failure scenario, consider overriding quorum so that container server events are not ignored. You can use the **xscmd** utility to query about and run quorum tasks, such as the quorum status and overriding quorum.

Before you begin

- Configure the quorum mechanism to be the same setting in all of your catalog servers. See “Configuring the quorum mechanism” on page 319 for more information.
- Quorum is the minimum number of catalog servers that are necessary to conduct placement operations for the data grid and is the full set of catalog servers, unless you configure a lower number. WebSphere eXtreme Scale expects to lose quorum for the following reasons:
  - Catalog service JVM member fails
  - Network brown out
  - Data center loss

  The following message indicates that quorum has been lost. Look for this message in your catalog service logs.

  ```
  CWOBJ1254W: The catalog service is waiting for quorum.
  ```

About this task

Override quorum in a data center failure scenario only. When you override quorum, any surviving catalog server instance can be used. All survivors are notified when one is told to override quorum.

Procedure

- Query quorum status with the **xscmd** utility.
  ```
  xscmd -c showQuorumStatus -cep cathost:2809
  ```
  Use this option to display the quorum status of a catalog service instance. One of the following outcomes is displayed:
  - `Quorum is disabled`: The catalog servers are running in a quorum-disabled mode. Quroum disabled mode is a development or single data center mode. Do not use quorum disabled mode for multiple data center configurations.
  - `Quorum is enabled and the catalog server has quorum`: Quorum is enabled and the system is working normally.
  - `Quorum is enabled but the catalog server is waiting for quorum`: Quorum is enabled and quorum has been lost.
  - `Quorum is enabled and the quorum is overridden`: Quorum is enabled and quorum has been overridden.
  - `Quorum status is outlawed`: When a brown out occurs, splitting the catalog service into two partitions, A and B. The catalog server A has overridden quorum. The network partition resolves and the server in the B partition is outlawed, requiring a JVM restart. It also occurs if the catalog JVM in B restarts during the brown out and then the brown out clears.
- Override quorum with the **xscmd** utility.
  ```
  xscmd -c overrideQuorum -cep cathost:2809
  ```
  Running this command forces the surviving catalog servers to re-establish a quorum.
- Diagnose quorum with the **xscmd** utility.
  - **Display a list of the core groups:**

    Use the **-c listCoreGroups** option to display a list of all the core groups for the catalog server.
    ```
    xscmd -c listCoreGroups -cep cathost:2809
    ```
  - **Teardown servers:**

    Use the **–c teardown** option to remove a server manually from the data grid. Removing a server from the grid is usually not necessary. Servers are automatically removed when they are detected as failed, but the command is provided for use under the guidance of IBM support. See "Stopping servers gracefully with the **xscmd** utility" on page 474 for more information about using this command.
    ```
    xscmd –c teardown server1,server2,server3 -cep cathost:2809 –g Grid
    ```
  - **Display the route table:**

    Use the **-c routetable** option to display the current route table by simulating a new client connection to the data grid. It also validates the route table by confirming that all container servers are recognizing their role in the route table, such as which type of shard for which partition.
    ```
    xscmd -c routetable -cep cathost:2809 –g myGrid
    ```
  - **Check the map sizes:**

Use the **-c showMapSizes** option to verify that key distribution is uniform over the shards in the key. If some container servers have more keys than others, then it is likely the hash function on the key objects has a poor distribution.

```
xscmd -c showMapSizes -cep cathost:2809 -g myGrid -ms myMapSet
```

– **Set trace strings:**

Use the **-c setTraceSpec** option to set the trace settings for all JVMs that match the filter specified for the **xscmd** command. This setting changes the trace settings only, until another command is used or the JVMs modified fail or stop.

```
xscmd -c setTraceSpec -spec ObjectGrid*=event=enabled -cep cathost:1099
–g myGrid –hf host1
```

This string enables trace for all JVMs on the server with the specified host name, in this case `host1`.

– **Display unassigned shards:**

Use the **-c showPlacement -sf U** option to display the list of shards that cannot be placed on the data grid. Shards cannot be placed when the placement service has a constraint that is preventing placement. For example, if you start JVMs on a single physical server while in production mode, then only primary shards can be placed. Replicas are not assigned until JVMs start on a second physical server. The placement service places replicas only on JVMs with different IP addresses than the JVMs that are hosting the primary shards. Having no JVMs in a zone can also cause shards to be unassigned.

```
xscmd -c showPlacement -sf U  -cep cathost:2809 –g myGrid
```

Related concepts:

Catalog server quorums
When the quorum mechanism is enabled, all the catalog servers in the quorum must be available for placement operations to occur in the data grid.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Starting eXtreme Scale servers using the Eclipse Equinox OSGi framework

WebSphere eXtreme Scale container servers can be started in an Eclipse Equinox OSGi framework using several methods.

Before you begin

Before you can start an eXtreme Scale container, you must have completed the following tasks:

1. The WebSphere eXtreme Scale server bundle must be installed into Eclipse Equinox.
2. Your application must be packaged as an OSGi bundle.
3. Your WebSphere eXtreme Scale plug-ins (if any) must be packaged as an OSGi bundle. They can be bundled in the same bundle as your application or as separate bundles.

4. If your container servers are using IBM eXtremeMemory, you must first configure the native libraries. For more information, see "Configuring IBM eXtremeMemory" on page 346.

About this task

This task describes how to start an eXtreme Scale container server in an Eclipse Equinox OSGi framework. You can use any of the following methods to start container servers using the Eclipse Equinox implementation:

- OSGi Blueprint service

  You can include all configuration and metadata in an OSGi bundle. See the following image to understand the Eclipse Equinox process for this method:

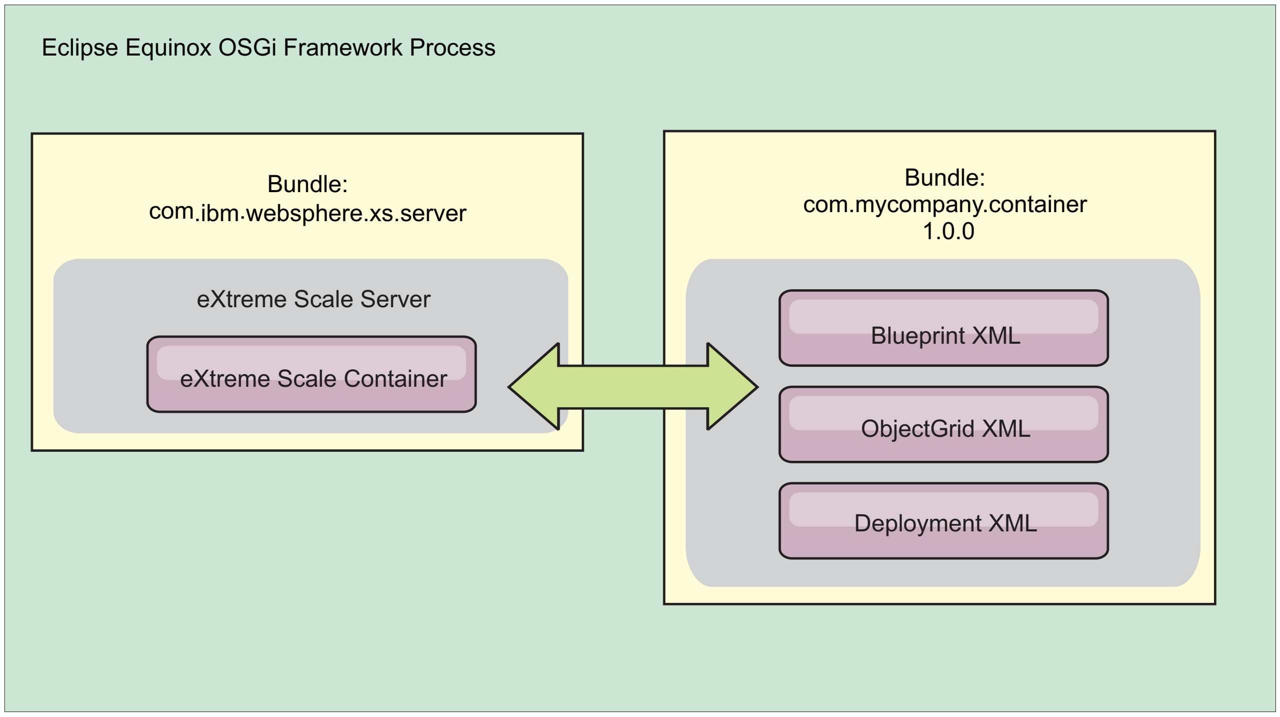


Figure 51. Eclipse Equinox process for including all configuration and metadata in an OSGi bundle

- OSGi Configuration Admin service

  You can specify configuration and metadata outside of an OSGi bundle. See the following image to understand the Eclipse Equinox process for this method:

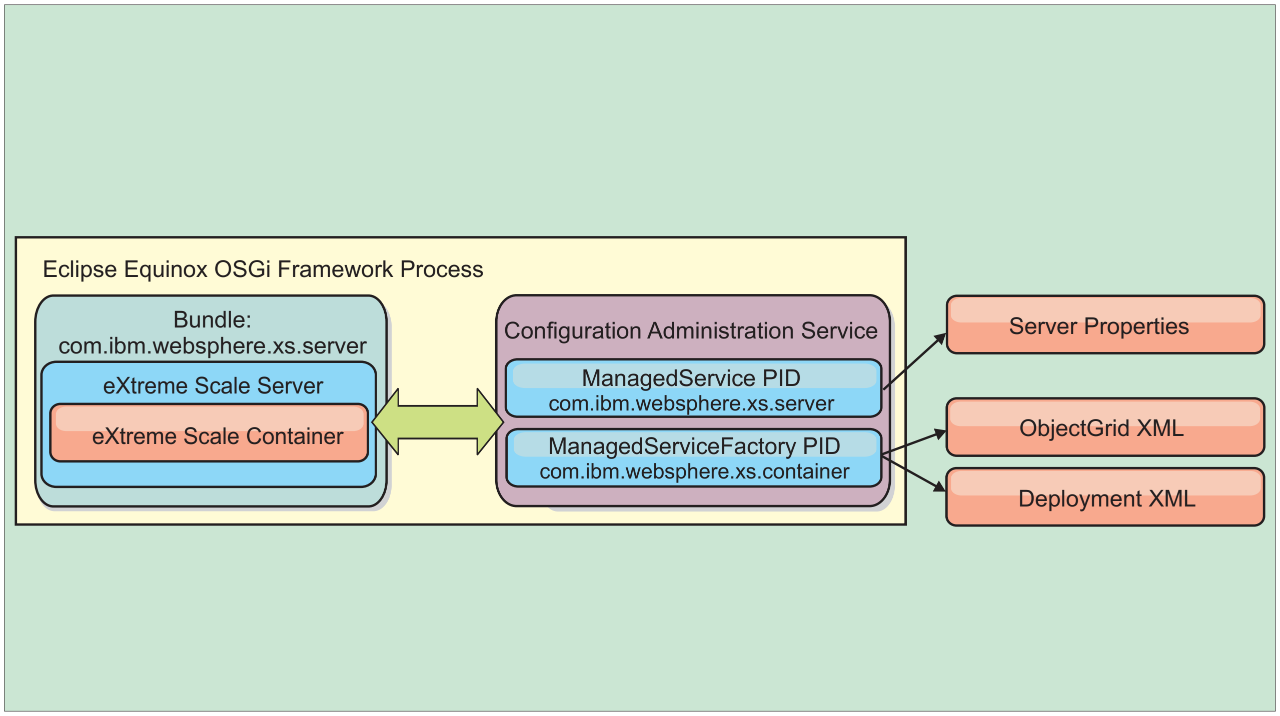

Eclipse Equinox OSGi Framework Process
Bundle: com.ibm.websphere.xs.server
eXtreme Scale Server
eXtreme Scale Container
Configuration Administration Service
ManagedService PID com.ibm.websphere.xs.server
ManagedServiceFactory PID com.ibm.websphere.xs.container
Server Properties
ObjectGrid XML
Deployment XML

Figure 52. Eclipse Equinox process for specify configuration and metadata outside of an OSGi bundle

- Programmatically

  Supports customized configuration solutions.

In each case, an eXtreme Scale server singleton is configured and one or more containers are configured.

The eXtreme Scale server bundle, `objectgrid.jar`, includes all of the required libraries to start and run an eXtreme Scale grid container in an OSGi framework. The server runtime environment communicates with user-supplied plug-ins and data objects using the OSGi service manager.

Important: After an eXtreme Scale server bundle is started and the eXtreme Scale server is initialized, it cannot be restarted . The Eclipse Equinox process must be restarted to restart an eXtreme Scale server.

You can use eXtreme Scale support for Spring namespace to configure eXtreme Scale container servers in a Blueprint XML file. When the server and container XML elements are added to the Blueprint XML file, the eXtreme Scale namespace handler automatically starts a container server using the parameters that are defined in the Blueprint XML file when the bundle is started. The handle stops the container when the bundle is stopped.

To configure eXtreme Scale container servers with Blueprint XML, complete the following steps:

Procedure

- Start an eXtreme Scale container server using OSGi blueprint.
  1. Create a container bundle.
  2. Install the container bundle into the Eclipse Equinox OSGi framework. See “Installing and starting OSGi-enabled plug-ins” on page 496.

3. Start the container bundle.

- Start an eXtreme Scale container server using OSGi configuration admin.
  1. Configure the server and container using config admin.
  2. When the eXtreme Scale server bundle is started, or the persistent identifiers are created with config admin, the server and container automatically start.
- Start an eXtreme Scale container server using the ServerFactory API. See the server API documentation.
  1. Create an OSGi bundle activator class, and use the eXtreme Scale ServerFactory API to start a server.

Installing and starting OSGi-enabled plug-ins

In this task, you install the dynamic plug-in bundle into the OSGi framework. Then, you start the plug-in.

Before you begin

Complete the following tasks before the OSGi-enabled plug-ins are installed and started.

- The eXtreme Scale server or client bundle is installed into the Eclipse Equinox OSGi framework. See “Installing eXtreme Scale bundles” on page 225.
- One or more dynamic BackingMap or ObjectGrid plug-ins are implemented. See Building eXtreme Scale dynamic plug-ins.
- The dynamic plug-ins are packaged as OSGi services in OSGi bundles.

About this task

Install the bundle with the Eclipse Equinox console. There are several different methods to install the bundle, including a modification of the `config.ini` configuration file. Products that embed Eclipse Equinox include alternative methods for adding bundles in the `config.ini` file. For more information, seeEclipse runtime options.

OSGi allows bundles to be started that have duplicate services. WebSphere eXtreme Scale uses the latest service ranking. When multiple OSGi frameworks are started in an eXtreme Scale data grid, you must make sure that the correct service rankings are started on each server. Failure to do so causes the grid to be started with a mixture of different versions.

To see which versions are in-use by the data grid, use the **`xscmd`** utility to check the current and available rankings. For more information, see “Updating OSGi services for eXtreme Scale plug-ins with **`xscmd`**” on page 500.

Procedure

Install the plug-in bundle into the Eclipse Equinox OSGi framework with the OSGi console.

1. Start the Eclipse Equinox framework with the console enabled.
   ```
   <java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
2. Install the plug-in bundle in the Equinox console.
   ```
   osgi> install file:///<path to bundle>
   ```

   Equinox lists the bundle ID for the newly installed bundle:

```
Bundle id is 17
```

3. Enter the following line to start the bundle in the Equinox console, where `<id>` is the bundle ID assigned when the bundle was installed:
   ```
   osgi>  start <id>
   ```
4. Retrieve the service status in the Equinox console to verify that the bundle started:
   ```
   osgi> ss
   ```
   When the bundle starts, the bundle lists the ACTIVE state, for example:
   ```
   17      ACTIVE      com.mycompany.plugin.bundle_VRM
   ```

Install the plug-in bundle into the Eclipse Equinox OSGi framework with the `config.ini` file.

5. Copy the plug-in bundle into the Eclipse Equinox plug-ins directory: For example:
   ```
   <equinox_root>/plugins
   ```
6. Edit the Eclipse Equinox `config.ini` configuration file, and add the bundle to the osgi.bundles property. For example:
   ```
   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
   ```
   Important: Verify that there is a blank line after the last bundle name. Each bundle is separated by a comma.
7. Start the Eclipse Equinox framework with the console enabled. For example:
   ```
   <java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
8. Retrieve the service status in the Equinox console to verify that the bundle started. For example:
   ```
   osgi> ss
   ```
   When the bundle starts, the bundle lists the ACTIVE state; for example:
   ```
   17      ACTIVE      com.mycompany.plugin.bundle_VRM
   ```

Results

The plug-in bundle is now installed and started. The eXtreme Scale container or client can now be started. For more information on developing eXtreme Scale plug-ins, see the System APIs and Plug-ins topic.

Administering OSGi-enabled services using the `xscmd` utility

You can use the **`xscmd`** utility to complete administrator tasks, such as viewing services and their rankings that are being used by each container, and updating the runtime environment to use new versions of the bundles.

About this task

With the Eclipse Equinox OSGi framework, you can install multiple versions of the same bundle, and you can update those bundles during run time. WebSphere eXtreme Scale is a distributed environment that runs the container servers in many OSGi framework instances.

Administrators are responsible for manually copying, installing, and starting bundles into the OSGi framework. eXtreme Scale includes an OSGi ServiceTrackerCustomizer to track any services that have been identified as eXtreme Scale plug-ins in the ObjectGrid descriptor XML file. Use the **xscmd** utility to validate which version of the plug-in is used, which versions are available to be used, and to perform bundle upgrades.

eXtreme Scale uses the service ranking number to identify the version of each service. When two or more services are loaded with the same reference, eXtreme Scale automatically uses the service with the highest ranking.

Procedure

- Run the **osgiCurrent** command, and verify that each eXtreme Scale server is using the correct plug-in service ranking.

  Since eXtreme Scale automatically chooses the service reference with the highest ranking, it is possible that the data grid may start with multiple rankings of a plug-in service.

  If the command detects a mismatch of rankings or if it is unable to find a service, a non-zero error level is set. If the command completed successfully then the error level is set to 0.

  The following example shows the output of the **osgiCurrent** command when two plug-ins are installed in the same grid on four servers. The loaderPlugin plug-in is using ranking 1, and the txCallbackPlugin is using ranking 2.

```
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
```

  The following example shows the output of the **osgiCurrent** command when server2 was started with a newer ranking of the loaderPlugin:

```
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
```

- Run the **osgiAll** command to verify that the plug-in services have been correctly started on each eXtreme Scale container server.

  When bundles start that contain services that an ObjectGrid configuration is referencing, the eXtreme Scale runtime environment automatically tracks the plug-in, but does not immediately use it. The **osgiAll** command shows which plug-ins are available for each server.

  When run without any parameters, all services are shown for all grids and servers. Additional filters, including the **-serviceName** `<service_name>` filter can be specified to limit the output to a single service or a subset of the data grid.

  The following example shows the output of the **osgiAll** command when two plug-ins are started on two servers. The loaderPlugin has both rankings 1 and 2 started and the txCallbackPlugin has ranking 1 started. The summary message at the end of the output confirms that both servers see the same service rankings:

```
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.
```

The following example shows the output of the **osgiAll** command when the bundle that includes the loaderPlugin with ranking 1 is stopped on server1. The summary message at the bottom of the output confirms that server1 is now missing the loaderPlugin with ranking 1:

```
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
```

The following example shows the output if the service name is specified with the **-sn** argument, but the service does not exist:

```
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.
```

- Run the **osgiCheck** command to check sets of plug-in services and rankings to see if they are available.

  The **osgiCheck** command accepts one or more sets of service rankings in the form: `-serviceRankings <service name>;<ranking>[,<serviceName>;<ranking>]`

  When the rankings are all available, the method returns with an error level of 0. If one or more rankings are not available, a non-zero error level is set. A table of all of the servers that do not include the specified service rankings is displayed. Additional filters can be used to limit the service check to a subset of the available servers in the eXtreme Scale domain.

  For example, if the specified ranking or service is absent, the following message is displayed:

```
Server  OSGi Service Unavailable Rankings
------  ------------ -------------------
server1 loaderPlugin 3
server2 loaderPlugin 3
```

- Run the **osgiUpdate** command to update the ranking of one or more plug-ins for all servers in a single ObjectGrid and MapSet in a single operation.

  The command accepts one or more sets of service rankings in the form: `-serviceRankings <service name>;<ranking>[,<serviceName>;<ranking>] -g <grid name> -ms <mapset name>`

  With this command, you can complete the following operations:
  - Verify that the specified services are available for updating on each of the servers.
  - Change the state of the grid to offline using the StateManager interface. See "Managing ObjectGrid availability" on page 488 for more information. This process quiesces the grid and waits until any running transactions have completed and prevents any new transactions from starting. This process also signals any ObjectGridLifecycleListener and BackingMapLifecycleListener plug-ins to discontinue any transactional activity. See Plug-ins for providing event listeners for information about event listener plug-ins.
  - Update each eXtreme Scale container running in an OSGi framework to use the new service versions.
  - Changes the state of the grid to online, allowing transactions to continue.

  The update process is idempotent so that if a client fails to complete any one task, it results in the operation being rolled back. If a client is unable to perform the rollback or is interrupted during the update process, the same command can be issued again, and it continues at the appropriate step.

  If the client is unable to continue, and the process is restarted from another client, use the `-force` option to allow the client to perform the update. The **osgiUpdate** command prevents multiple clients from updating the same map set concurrently. For more details about the **osgiUpdate** command, see "Updating OSGi services for eXtreme Scale plug-ins with **xscmd**."

Related reference:

Plug-ins for providing event listeners
You can use the ObjectGridEventListener, MapEventListener, ObjectGridLifecycleListener and BackingMapLifecycleListener plug-ins to configure notifications for various events in the eXtreme Scale cache. Listener plug-ins are registered with an ObjectGrid or BackingMap instance like other eXtreme Scale plug-ins and add integration and customization points for applications and cache providers.

Related information:

Eclipse runtime options

Updating OSGi services for eXtreme Scale plug-ins with `xscmd`

WebSphere eXtreme Scale supports upgrading container server plug-in bundles while the grid is active. This support allows administrators to complete application updates and additions without needing to restart grid processes.

Before you begin

Complete the following steps before you update eXtreme Scale OSGi bundles to a new version:

1. Start eXtreme Scale servers in a supported OSGi framework.

2. Separate all eXtreme Scale plug-ins into bundles, and they must use service rankings to identify each version of the plug-ins.
3. Specify cache objects as either Java primitive types such as `byte[]`, Integer or String, or they must be stored using a MapSerializerPlugin plug-in The data objects are stored in the eXtreme Scale bundle and are not upgraded. Only the plug-ins that interact with the data are updated.
4. Design cache object data to be version compatible. New plug-ins must be able to interact with data created by older plug-ins.
5. Design plug-ins to listen for ObjectGridLifecycle and BackingMapLifecycle events to refresh any references to other plug-ins or the metadata that the plug-ins might have so that they can be refreshed when it is updated.
6. The eXtreme Scale OSGi update process only affects servers. You must independently update any clients that are using plug-ins.

About this task

Without OSGi enablement, if an administrator needs to update the application plug-ins or cache objects, each grid node must be upgraded one-by-one, causing stress on the network, memory and cpu utilization. This is required since plug-ins and cache Java objects are directly stored in the grid. When classes are updated without restarting the processes, the grid plug-ins have conflicts because each class has a different ClassLoader.

The eXtreme Scale product includes the xscmd utility and MBeans that allows administrators to view all the plug-in bundles installed in each grid container's hosting OSGi framework and choose which revision to use. When the xscmd is used to update the plug-ins to a new ranking, the grid is quiesced and all transactions are drained, the plug-ins are updated, and the grid is activated again. If an error occurs during the update process, the process is rolled-back and the old ranking is restored.

Procedure

1. Create a version of the bundle, increasing the version number in the bundle manifest, and increasing the ranking for each eXtreme Scale plug-in service. If the original bundle version is `Bundle-Version: 1.0.0`, then the next version can be defined as `Bundle-Version: 1.1.0`.

   If the original service ranking is `ranking="1"`, then the next ranking can be defined as `ranking="2"`.

   Important: OSGi service rankings must be integers.
2. Copy the new bundle to each OSGi framework node that is hosting an eXtreme Scale container server.
3. Install the new bundle into the OSGi framework. The bundle is assigned a bundle identifier; for example:
   ```
   osgi> install <URL to bundle>
   ```
4. Start the new bundle using the assigned bundle identifier; for example:
   ```
   osgi> start <id>
   ```
   After the new bundle is started, the eXtreme Scale OSGi service tracker detects the bundle and makes it available for updating.
5. Use the **`xscmd -c osgiAll`** command to verify that each container server sees the new bundle. The **`osgiAll`** command queries all containers in the grid for all services that are referenced in the ObjectGrid descriptor XML file and displays all rankings that are available; for example:

```
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. Use the **xscmd -c osgiCheck** command to verify that one or more service rankings are valid update targets; for example:

```
xscmd -c osgiCheck -sr
mySerializerServiceFactory;2,myLoaderServiceFactory;2

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

7. If the **osgiCheck** command did not find any resulting errors, suspend the balancer of the placement service to avoid shard movements, in case of a failure during the update process. To suspend placement, use the **xscmd -c suspendBalancing** command for each object grid and map set that are affected by the update; for example:

```
xscmd -c suspendBalancing -g MyGrid -ms MyMapSet
```

8. After balancing has been suspended for each object grid and map set, use the **xscmd -c osgiCheck** command again to verify that one or more service rankings are valid update targets; for example:

```
xscmd -c osgiCheck -sr
mySerializerServiceFactory;2,myLoaderServiceFactory;2

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

9. After balancing has been suspended for the object grid and map set, use the **osgiUpdate** command to update the service on all of the servers for an object grid and map set; for example:

```
xscmd -c osgiUpdate  -sr
mySerializerServiceFactory;2,myLoaderServiceFactory;2 -g MyGrid -ms MyMapSet
```

10. Verify that the upgrade succeeded; for example:

```
Update succeeded for the following service rankings:
Service                     Ranking
-------                     -------
mySerializerServiceFactory  2
myLoaderServiceFactory      2
```

11. After you verify that the ranking has been updated successfully, enable balancing again, using the **xscmd -c resumeBalancing** command; for example:

```
xscmd -c resumeBalancing -g MyGrid -ms MyMapSet
```

12. Stop and uninstall the old bundle in each OSGi framework that is hosting the eXtreme Scale container. For example, enter the following code in the Eclipse Equinox console:

```
osgi> stop <id>
osgi> uninstall <id>
```

Results

The eXtreme Scale bundle has been updated to a new version.

Related reference:

Plug-ins for providing event listeners
You can use the ObjectGridEventListener, MapEventListener, ObjectGridLifecycleListener and BackingMapLifecycleListener plug-ins to configure notifications for various events in the eXtreme Scale cache. Listener plug-ins are registered with an ObjectGrid or BackingMap instance like other eXtreme Scale plug-ins and add integration and customization points for applications and cache providers.

Related information:

Eclipse runtime options

Administering with Managed Beans (MBeans)

You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

JMX MBean interfaces and WebSphere eXtreme Scale

Each MBean has get methods that represent attribute values. These get methods cannot be called directly from your program. The JMX specification treats attributes differently from operations. You can view attributes with a vendor JMX console, and you can perform operations in your program or with a vendor JMX console.

Package com.ibm.websphere.objectgrid.management

See the API documentation for an overview and detailed programming specifications for all of the available MBeans:Package com.ibm.websphere.objectgrid.management .

Related concepts:

"Statistics overview" on page 511
Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

Related tasks:

"Accessing Managed Beans (MBeans) using the wsadmin tool" on page 504
You can use the wsadmin utility provided in WebSphere Application Server to access managed bean (MBean) information.

"Accessing Managed Beans (MBeans) programmatically" on page 504
You can connect to MBeans with Java applications. These applications use the interfaces in the com.ibm.websphere.objectgrid.management package.

"Monitoring server statistics with managed beans (MBeans)" on page 549
You can used managed beans (MBeans) to track statistics in your environment.

"Monitoring with the `xscmd` utility" on page 535
The `xscmd` utility replaces the `xsadmin` sample utility as a fully supported monitoring and administration tool. With the `xscmd` utility, you can display textual information about your WebSphere eXtreme Scale topology.

Related information:

API documentation: Package com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

Accessing Managed Beans (MBeans) using the wsadmin tool

You can use the wsadmin utility provided in WebSphere Application Server to access managed bean (MBean) information.

Procedure

Run the wsadmin tool from the `bin` directory in your WebSphere Application Server installation. The following example retrieves a view of the current shard placement in a dynamic eXtreme Scale. You can run the wsadmin tool from any installation where eXtreme Scale is running. You do not have to run the wsadmin tool on the catalog service.

```
$ 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>
```

Related concepts:

"Statistics overview" on page 511
Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

Related reference:

"Administering with Managed Beans (MBeans)" on page 503
You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

Related information:

API documentation: Package com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

Accessing Managed Beans (MBeans) programmatically

You can connect to MBeans with Java applications. These applications use the interfaces in the com.ibm.websphere.objectgrid.management package.

About this task

Programmatic methods for accessing MBeans vary depending on the type of server to which you are connecting.

- Connect to a stand-alone catalog service MBean server
- Connect to a container MBean server
- Connect to a catalog service MBean server that is hosted in WebSphere Application Server
- Connect to a catalog service MBean server with security enabled

Procedure

- **Connect to a stand-alone catalog service MBean server:**

  The following example program connects to a stand-alone catalog service MBean server and returns an XML formatted string that lists each container server along with its allocated shards for a given ObjectGrid and MapSet.

```
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();
            }
        }
    }

}
```

Figure 53. CollectPlacementPlan.java

A few notes regarding the sample program:

– The **JMXServiceURL** value for the catalog service is always of the following form: `service:jmx:rmi:///jndi/rmi://<host>:<port>/objectgrid/MBeanServer`, where `<host>` is the host on which the catalog service is running and `<port>` is the JMX service port that is provided with the **-JMXServicePort** option when starting the catalog service. If no port is specified, the default is `1099`.

- For the ObjectGrid or map statistics to be enabled, you must specify the following property in the server properties file when you are starting an ObjectGrid container: `statsSpec=all=enabled`
- To disable the MBeans that are running in the container servers, specify the following property in the server properties file: `enableMBeans=false`.

An example of the output follows. This output indicates that two container servers are active. The `Container-0` container server hosts four primary shards. The `Container-1` container server hosts a synchronous replica for each of the primary shards on the `Container-0` container server. In this configuration, two synchronous replicas and one asynchronous replica are configured. As a result, the `Unassigned` container server is left with the remaining shards. If two more container servers are started, the `Unassigned` container server is not displayed.

```
<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>
```

- **Connect to a container MBean server:**

  Container servers host MBeans to query information about the individual maps and ObjectGrid instances that are running within the container server. The following example program prints the status of each container server that is hosted by the catalog server with the JMX address of `localhost:1099`:

```
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();
            }
        }
    }
}
```

Figure 54. CollectContainerStatus.java

The example program prints out the container server status for each container. An example of the output follows:

```
<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>
```

- **Connect to a catalog service MBean server that is hosted in WebSphere Application Server:**

  The method for programmatically accessing MBeans in WebSphere Application Server is slightly different from accessing MBeans in a stand-alone configuration.

  1. Create and compile a Java program to connect to the MBean server. An example program follows:

```
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();
            }
        }
    }

}
```

Figure 55. CollectPlacementPlan.java

2. Run the following command.

```
"$JAVA_HOME/bin/java" "$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>
```

This command assumes that the *was_root*/`bin/setupCmdLine.sh` script has been run to set the variables properly. An example of the format of the java.naming.provider.url property value is `corbaloc:iiop:1.0@<host>:<port>/NameService`.

- **Connect to a catalog service MBean server with security enabled:**

For more information about connecting to the catalog service MBean with security enabled, see “Java Management Extensions (JMX) security” on page 595.

What to do next

For more examples on how to display statistics and perform administrative operations with MBeans, see the **`xsadmin`** sample application. You can look at the source code of the xsadmin sample application in the *wxs_home*`/samples/xsadmin.jar` file in a stand-alone installation, or in the *wxs_home*`/xsadmin.jar` file in a WebSphere Application Server installation. See Sample: **`xsadmin`** utility for more information about the operations you can complete with the **`xsAdmin`** sample application.

You can also find more information about MBeans in the com.ibm.websphere.objectgrid.management package.

Related concepts:

“Statistics overview” on page 511
Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

Related reference:

“Administering with Managed Beans (MBeans)” on page 503
You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

Related information:

API documentation: Package com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

Chapter 8. Monitoring

You can use the included monitoring console, APIs, MBeans, logs, and utilities to monitor the performance of your application environment.

Statistics overview

Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

The following figure shows the general setup of statistics for WebSphere eXtreme Scale.

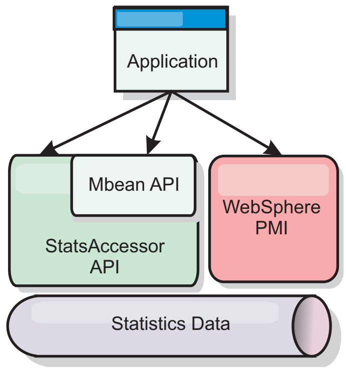


Figure 56. Statistics overview

Each of these APIs offer a view into the statistics tree, but are used for different reasons:

- **Statistics API**: The Statistics API allows developers direct access to statistics for flexible and customizable statistics integration solutions, such as custom MBeans or logging.
- **MBean API**: The MBean API is a specification-based mechanism for monitoring. The MBean API uses the Statistics API and runs local to the server Java Virtual Machine (JVM). The API and MBean structures are designed to readily integrate with other vendor utilities. Use the MBean API when you are running a distributed object grid.
- **WebSphere Application Server Performance Monitoring Infrastructure (PMI) modules**: Use PMI if you are running WebSphere eXtreme Scale within WebSphere Application Server. These modules provide a view of the internal statistics tree.

Statistics API

Much like a tree map, there is a corresponding path and key used to retrieve a specific module, or in this case granularity or aggregation level. For example, assume there is always an arbitrary root node in the tree and that statistics are being gathered for a map named "payroll," belonging to an ObjectGrid named "accounting." For example, to access the module for a map's aggregation level or granularity, you could pass in a String[] of the paths. In this case that would equate to String[] {root, "accounting", "payroll"}, as each String would represent the node's path. The advantage of this structure is that a user can specify the array to any node in the path and get the aggregation level for that node. So passing in String[] {root, "accounting"} would give you map statistics, but for the entire grid of "accounting." This leaves the user with both the ability to specify types of statistics to monitor, and at whatever level of aggregation is required for the application.

WebSphere Application Server PMI modules

WebSphere eXtreme Scale includes statistics modules for use with the WebSphere Application Server PMI. When a WebSphere Application Server profile is augmented with WebSphere eXtreme Scale, the augment scripts automatically integrate the WebSphere eXtreme Scale modules into the WebSphere Application Server configuration files. With PMI, you can enable and disable statistics modules, automatically aggregate statistics at various granularity, and even graph the data using the built-in Tivoli Performance Viewer. See “Monitoring with WebSphere Application Server PMI” on page 537 for more information.

Vendor product integration with Managed Beans (MBean)

The eXtreme Scale APIs and Managed Beans are designed to allow for easy integration with third party monitoring applications. JConsole or MC4J are some examples of lightweight Java Management Extensions (JMX) consoles that can be used to analyze information about an eXtreme Scale topology. You can also use the programmatic APIs to write adapter implementations to snapshot or track eXtreme Scale performance. WebSphere eXtreme Scale includes a sample monitoring application that allows out-of-the box monitoring capabilities, and can be used as a template for writing more advanced custom monitoring utilities.

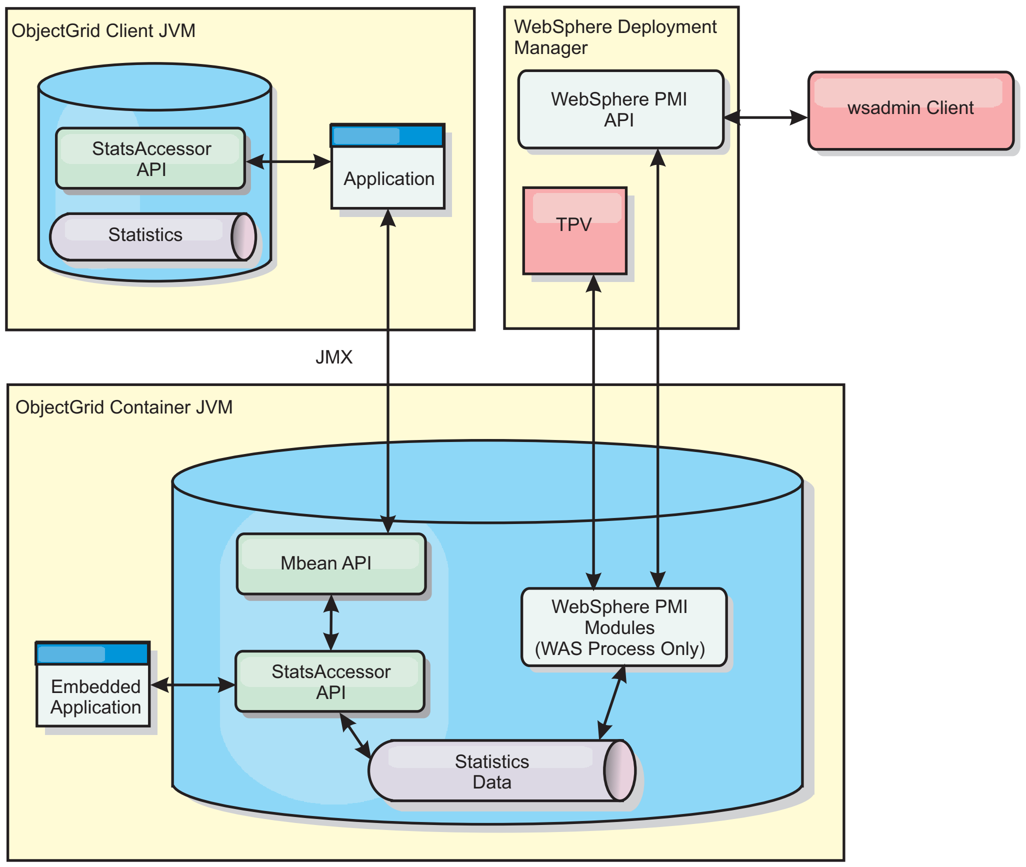


Figure 57. MBean overview

See Sample: **xsadmin** utility for more information. For more information about integrating with specific vendor applications, see the following topics:

- Monitoring eXtreme Scale with IBM Tivoli Monitoring agent
- "Monitoring eXtreme Scale with Hyperic HQ" on page 560
- "Monitoring eXtreme Scale applications with CA Wily Introscope" on page 557

Related tasks:

"Accessing Managed Beans (MBeans) using the wsadmin tool" on page 504
You can use the wsadmin utility provided in WebSphere Application Server to access managed bean (MBean) information.

"Accessing Managed Beans (MBeans) programmatically" on page 504
You can connect to MBeans with Java applications. These applications use the interfaces in the com.ibm.websphere.objectgrid.management package.

"Monitoring server statistics with managed beans (MBeans)" on page 549
You can used managed beans (MBeans) to track statistics in your environment.

"Monitoring with the **xscmd** utility" on page 535
The **xscmd** utility replaces the **xsadmin** sample utility as a fully supported monitoring and administration tool. With the **xscmd** utility, you can display textual information about your WebSphere eXtreme Scale topology.

Related reference:

"Administering with Managed Beans (MBeans)" on page 503
You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

Related information:

API documentation: Package com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

Monitoring with the web console

With the web console, you can chart current and historical statistics. This console provides some preconfigured charts for high-level overviews, and has a custom reports page that you can use to build charts from the available statistics. You can use the charting capabilities in the monitoring console of WebSphere eXtreme Scale to view the overall performance of the data grids in your environment.

Related reference:

"Web console statistics" on page 519
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **`stopOgServer`** script stops catalog and container servers.

Related information:

"Getting started tutorial lesson 4: Monitor your environment" on page 7
You can use the **`xscmd`** utility and web console tools to monitor your data grid environment.

Starting and logging on to the web console

Start the console server by running the **`startConsoleServer`** command and logging on to the server with the default user ID and password.

Before you begin

- 7.1.1 **System requirements**
  - Use a AIX, Linux, or Windows system to run the web console. If you are using a 64-bit operating system, you must use a 32-bit Java virtual machine (JVM) to host the web console.
  - Install a stand-alone WebSphere eXtreme Scale server on the system that is hosting the console server. See "Installing WebSphere eXtreme Scale or WebSphere eXtreme Scale Client in a stand-alone environment" on page 184 for more information.
  - The console server system must be able to connect to your catalog service. The catalog service also must be able to connect back to the web console server.
- **Web browser requirements**

  Use one of the following browsers with the web console:
  - Mozilla Firefox, version 3.5.x and later
  - Mozilla Firefox, version 3.6.x and later
  - Microsoft Internet Explorer, version 7 or 8

About this task

If your catalog servers are running in a stand-alone environment, with a Java security manager, then the user ID that you use to log in must correspond to a principal that has the following permissions in the Java security policy file:

```
grant principal javax.security.auth.x500.X500Principal "CN=manager,O=acme,OU=OGSample"
      {
         permission javax.management.MBeanPermission "*", "getAttribute,setAttribute,invoke,queryNames,addNotificationListener,removeNotificationListener";
      };
```

Procedure

1. Optional: If you want to run your console server on a port other than the default port, edit the *wxs_install_root*/`ObjectGrid/console/config/zero.config` file. The default port for the console server is `7080` for HTTP and `7443` for HTTPS. You can edit the following properties to change the default values:
   ```
   /config/http/port = 7080
   /config/https/port = 7443
   ```

   If you edit these values after the console server is already started, restart the server to use the new port numbers.
2. Start the console server. The **`startConsoleServer.bat|sh`** script for starting the console server is in the *wxs_install_root*/`ObjectGrid/bin` directory of your installation.
3. Log on to the console.
   a. From your web browser, go to `https://your.console.host:7443`, replacing `your.console.host` with the host name of the server onto which you installed the console.
   b. Log on to the console.
      - **User ID:** `admin`
      - **Password:** `admin`

      The console welcome page is displayed.
4. Edit the console configuration. Click **Settings** > **Configuration** to review the console configuration. The console configuration includes information such as:
   - Trace string for the WebSphere eXtreme Scale client, such as `*=all=disabled`
   - The Administrator name and password
   - The Administrator e-mail address

What to do next

- Connect your catalog servers to the web console to start tracking statistics. See “Connecting the web console to catalog servers” on page 516 for more information.
- If you need to stop the web console server, run the **`stopConsoleServer.bat|sh`** script. This script is in the *wxs_install_root*/`ObjectGrid/bin` directory of your installation.

Related reference:

“Web console statistics” on page 519
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **`stopOgServer`** script stops catalog and container servers.

Related information:

“Getting started tutorial lesson 4: Monitor your environment” on page 7
You can use the **`xscmd`** utility and web console tools to monitor your data grid environment.

Connecting the web console to catalog servers

To start viewing statistics in the web console, you must first connect to catalog servers that you want to monitor. Additional steps are required if your catalog servers have security enabled.

Before you begin

- The web console server must be running. See "Starting and logging on to the web console" on page 514 for more information.
- You must have at least one catalog server running to which you want to connect. See "Starting a stand-alone catalog service" on page 461 for more information.

Procedure

1. If your catalog servers have Secure Sockets Layer (SSL) enabled, you must configure a keystore and a truststore in the client properties file. You can enable SSL for a catalog server by setting the transportType attribute to `SSL-Required` or `SSL-Supported` in the server properties file. For more information about the SSL properties you can set on the server, see the Server properties file.
   a. Configure a keystore and truststore, and then exchange, or cross-import the public certificates. For example, you might copy the truststore and keystore to a location on the server that is running the web console.
   b. Edit the client properties file on the web console server to include the properties for SSL configuration. For more information about the SSL properties you can set for the client, see the Client properties file. For example, you can make a copy of the client properties file located in the*wxs_install_root*`/ObjectGridProperties/sampleclient.properties` and edit that file. The following properties are required for outbound SSL connections from the web console:

```
#------------------------------------------------------------------------------
# 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
```

   Important: Windows If you are using Windows, you must escape any backslash (\ ) characters in the path. For example, if you want to use the path `C:\opt\ibm`, enter `C:\\opt\\ibm` in the properties file. Windows directories with spaces are not supported.
2. Establish and maintain connections to catalog servers that you want to monitor. Repeat the following steps to add each catalog server to the configuration.
   a. Click **Settings** > **eXtreme Scale Catalog Servers**.
   b. Add a new catalog server.

1) Click the add icon (➕) to register an existing catalog server.
2) Provide information, such as the host name and listener port. See "Planning for network ports" on page 51 for more information about port configuration and defaults.
3) Click **OK**.
4) Verify that the catalog server has been added to the navigation tree.

3. Group the catalog servers that you created into a catalog service domain. You must create a catalog service domain when security is enabled in your catalog servers because security settings are configured in the catalog service domain.
   a. Click **Settings** > **eXtreme Scale Domains** page.
   b. Add a new catalog service domain.

1) Click the add icon (➕) to register a catalog service domain. Enter a name for the catalog service domain.
2) After you create the catalog service domain, you can edit the properties. The catalog service domain properties follow:

Name Indicates the host name of the domain, as assigned by the administrator.

Catalog servers
Lists one or more catalog servers that belong to the selected domain. You can add the catalog servers that you created in the previous step.

Generator class
Specifies the name of the class that implements the CredentialGenerator interface. This class is used to get credentials for clients. If you specify a value in this field, the value overrides the **`crendentialGeneratorClass`** property in the `client.properties` file.

Generator properties
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. If you specify a value in this field, the value overrides the **`credentialGeneratorProps`** property in the `client.properties` file.

eXtreme Scale client properties path
Specifies the path to the client properties file that you edited to include security properties in a previous step. For example, you might indicate the `c:\ObjectGridProperties\sampleclient.properties` file. If you want to stop the console from trying to use secure connections, you can delete the value in this field. After you set the path, the console uses an unsecured connection.

3) Click **OK**.
4) Verify that the domain has been added to the navigation tree.

To view information about an existing catalog service domain, click the name of the catalog service domain in the navigation tree on the **Settings** > **eXtreme Scale Domains** page.

4. View the connection status. The **Current domain** field indicates the name of the catalog service domain that is currently being used to display information in the web console. The connection status displays next to the name of the catalog service domain.

Related reference:

"Web console statistics" on page 519
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **`stopOgServer`** script stops catalog and container servers.

Related information:

"Getting started tutorial lesson 4: Monitor your environment" on page 7
You can use the **`xscmd`** utility and web console tools to monitor your data grid environment.

Viewing statistics with the web console

You can monitor statistics and other performance information with the web console.

Before you begin

Before you can view statistics with the web console, you must complete the following tasks:

1. Start the web console server. See "Starting and logging on to the web console" on page 514 for more information.
2. Connect your catalog servers to the web console server. See "Connecting the web console to catalog servers" on page 516 for more information.
3. Run active data grids and applications within the servers that are managed by your catalog service domain.

About this task

After you create your data grids and configure your applications to use the data grids, allow some time to pass for the statistics to become available. For example, with a dynamic cache data grid, statistics are not available until a WebSphere Application Server that is running a dynamic cache connects to the dynamic cache. In general, wait up to one minute after a major configuration change to see the changes in your statistics.

Tip: To view more specific information about any data point in a chart, you can move the mouse pointer over the data point.

Procedure

- To view the current server statistics, click **Monitor** > **Server Overview**.
- To view the performance of all of your data grids, click **Monitor** > **Data grid domain overview**.

- To view individual data grids, click **Monitor** > **Data grid overview** > ***data_grid_name***. This page shows a summary that includes the number of cache entries, the average transaction time, and average throughput.
- To view further details about a specific data grid, click **Monitor** > **Data grid details**. A tree displays with all of the data grids in your configuration. You can drill down into a specific data grid to view the maps that are a part of that data grid. You can either click a data grid name or a map for further information.
- To choose which statistics you would like your custom report to contain, click **Monitor** > **Custom reports**.

  Use this view to construct detailed data charts of the various statistics. Use the tree to explore the available data grids and servers and their associated statistics. A menu opens when you click or press enter on a node that references data that can be charted. Create a new chart containing the statistics, or add the statistics into an existing chart with compatible statistics. See "Monitoring with custom reports" on page 525 for more information.

Related reference:

"Web console statistics"
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **stopOgServer** script stops catalog and container servers.

Related information:

"Getting started tutorial lesson 4: Monitor your environment" on page 7
You can use the **xscmd** utility and web console tools to monitor your data grid environment.

Web console statistics

Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

- "Data grid domain overview"
- "Data grid overview" on page 520
- "Data grid details" on page 520
- "Server overview" on page 521
- "Custom reports: Catalog service domain statistics" on page 521
  - "Custom reports: Container server statistics" on page 521
  - "Custom reports: Data grid statistics" on page 523
  - "Custom reports: Map statistics" on page 523

Data grid domain overview

Data grid domain overview statistics are displayed on the **Monitor** > **Data Grid Domain Overview** page. Click one of the following tabs for more information about the data grid domain:

Used Capacity tab
: In the **Current Data Grid Used Capacity Distribution** chart, a picture of the **Total Pool**, and the **Largest Used Capacity Consumers** are displayed. Only the top 25 data grids are displayed. In the **Used Capacity Over Time** chart, the number of bytes that are consumed by the data grid is displayed.

Average Throughput tab
: The **5 Most Active Data Grids by Average Transaction Time in Milliseconds** chart contains a list of the top five data caches, organized by the average transaction time. The **Average Throughput Over time** chart displays the average, maximum, and minimum throughput within the last hour, day, and week.

Average Transaction Time tab
: The **5 Slowest Data Grids** chart displays data about the slowest data grids. The **Average Transaction Time Over Time** chart displays the average , maximum, and minimum transaction time within the last hour, day, and week.

Data grid overview

To view statistics for an individual data grid, click **Monitor** > **Data Grid Overview** > *data_grid_name*.

Current summary over last 30 seconds
: Displays the current number of cache entries, average transaction time, average throughput, and cache hit rate for the selected data grid.

Used Capacity tab
: The **Current summary over last 30 seconds** chart displays the number of cache entries and used capacity in bytes over a specified time range.

Cache Usage tab
: The **Cache Usage** chart helps to visualize the number of successful queries to the cache, and displays cache attempts, cache hits, and the cache hit rate over a specified time range.

Average Throughput tab
: The **Average Throughput vs. Average Transaction Time** chart displays the transaction time and throughput over a specified time range.

Data grid details

Data grid statistics are displayed on the **Monitor** > **Data Grid Details** page. You can look at data for a selected grid and the maps that are within that grid.

Current summary over last 30 seconds
: Displays the current used capacity, number of cache entries, average throughput, and average transaction time for the selected data grid.

Current eXtreme Scale Object Grid Map Used Capacity Distribution
: View a total pool, which includes the capacity by zone and the total capacity in each zone. Only the top 25 ObjectGrid maps are displayed. You can also view the largest used capacity consumers by each map.

Current Zone Used Capacity Distribution
: View a total pool, which includes the total pool and the top used capacity consumers in the zone of the selected data grid. You can also view the largest used capacity consumers by each zone.

Map statistics:

Current summary over last 30 seconds
: Displays the current used capacity, number of cache entries, average throughput, and average transaction time for the selected map.

Current Partition Used Capacity Distribution
View a partition, which includes the total pool and the top used capacity consumers. Only the top 25 partitions are displayed. You can also view the largest used capacity consumers by each partition.

Server overview

Server statistics are displayed on the **Monitor** > **Server Overview** page.

Current Server Used Memory Distribution
This chart is composed of two views. **Total Pool** displays the current amount of used (real) memory in the server run time. **Largest Used memory Consumers** breaks down the used memory by server; however only the top 25 servers that are using the most memory are displayed.

Total Memory Over Time
Displays the real memory usage in the server run time.

Used Memory Over Time
Displays the amount of used memory in the server run time.

Custom reports: Catalog service domain statistics

You can view catalog service domain statistics by creating a custom report. Click **Monitor** > **Custom Reports**.

Average Transaction Time (ms)
Displays the average time required to complete a transaction in this domain.

Average Transaction Throughput (trans/sec)
Displays the average number of transactions per second in this domain.

Maximum Transaction Time (ms)
Displays the time spent by the *most* time-consuming transaction in this domain.

Minimum Transaction Time (ms)
Displays the time spent by the *least* time-consuming transaction in this domain.

Total Transaction Time (ms)
Displays total time spent on transactions in this domain, since the time the domain was initialized.

Custom reports: Container server statistics

You can view container server statistics by creating a custom report. Click **Monitor** > **Custom Reports**.

Average Transaction Time (ms)
Displays the average time required to complete a transaction for this catalog server.

Average Transaction Throughput (trans/sec)
Displays the average number of transactions per second for this catalog server.

Maximum Transaction Time (ms)
Displays the time spent by the *most* time-consuming transaction for this catalog server.

Minimum Transaction Time (ms)
: Displays the time spent by the *least* time-consuming transaction for this catalog server.

Total Transaction Time (ms)
: Displays total time spent on transactions for this catalog server, since the time for this catalog server was initialized.

Total Entries in Cache
: Displays the current number of objects cached in the grids overseen by this catalog server.

Hit rate (percentage)
: Displays the hit rate (hit ratio) for the selected data grid. A high hit rate is desirable. The hit rate indicates how well the grid is helping to avoid accessing the persistent store.

Used Bytes
: Displays memory consumption by this map. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Minimum Used Bytes
: Displays the low point in memory consumption by this catalog service and its maps. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Maximum Used Bytes
: Displays the high point in memory consumption by this catalog service and its maps. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Total Number of Hits
: Displays the total number of times the requested data was found in the map, avoiding the need to access persistent store.

Total Number of Gets
: Displays the total number of times the map had to access the persistent store to obtain data.

Free Heap (MB)
: Displays the actual amount of heap available to the JVM being used by the catalog server.

Total Heap
: Displays the amount of heap available to the JVM being used by this catalog server.

Number of Available Processors
: Displays the number of processors that are available to this catalog service and its maps. For the highest stability, run your servers at 60% processor loading and JVM heaps at 60% heap loading. Spikes can then drive the processor usage to 80–90%, but do not regularly run your servers higher than these levels

Maximum Heap Size (MB)
: Displays the maximum amount of heap available to the JVM being used by this catalog server.

Used Memory
: Displays the used memory in the JVM being used by this catalog server.

Custom reports: Data grid statistics

You can view data grid statistics by creating a custom report. Click **Monitor** > **Custom Reports**.

Average Transaction Time (ms)
Displays the average time required to complete transactions involving this grid.

Average Transaction Throughput (trans/sec)
Displays the average number of transactions per second completed by this grid.

Maximum Transaction Time (ms)
Displays the time spent by the *most* time-consuming transaction completed by this grid.

Minimum Transaction time (ms)
Displays the time spent by the *least* time-consuming transaction completed by this grid.

Total Transaction Time (ms)
Displays the total amount of transaction processing time for this grid.

Custom reports: Map statistics

You can view map statistics by creating a custom report. Click **Monitor** > **Custom Reports**.

Total Entries in Cache
Displays the current number of objects cached in this map.

Hit Rate (percentage)
Displays the hit rate (hit ratio) for the selected map. A high hit rate is desirable. The hit rate indicates how well the map is helping to avoid accessing the persistent store.

Used Bytes
Displays memory consumption by this map. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Minimum Used Bytes
Displays the minimum consumption (in Bytes) for this map. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Maximum Used Bytes
Displays the maximum consumption (in Bytes) for this map. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Total Number of Hits
Displays the total number of times the requested data was found in the map, avoiding the need to access persistent store.

Total Number of Gets
Displays the total number of times the map had to access the persistent store to obtain data.

Free Heap (MB)
: Displays the current amount of heap available to this map, in the JVM being used by the catalog server.

Total Heap (MB)
: Displays the total amount of heap available to this map, in the JVM being used by the catalog server. For the highest stability, run your servers at 60% processor loading and JVM heaps at 60% heap loading. Spikes can then drive the processor usage to 80–90%, but do not regularly run your servers higher than these levels

Number of Available Processors
: Displays the number of processors available to this map. For the highest stability, run your servers at 60% processor loading and JVM heaps at 60% heap loading. Spikes can then drive the processor usage to 80–90%, but do not regularly run your servers higher than these levels

Maximum Heap Size (MB)
: Displays the maximum amount of heap available to this map, in the JVM being used by the catalog server.

Used Memory (MB)
: Displays the used amount of memory in this map.

Related tasks:

"Viewing statistics with the web console" on page 518
You can monitor statistics and other performance information with the web console.

"Monitoring with the web console" on page 514
With the web console, you can chart current and historical statistics. This console provides some preconfigured charts for high-level overviews, and has a custom reports page that you can use to build charts from the available statistics. You can use the charting capabilities in the monitoring console of WebSphere eXtreme Scale to view the overall performance of the data grids in your environment.

"Starting and logging on to the web console" on page 514
Start the console server by running the **`startConsoleServer`** command and logging on to the server with the default user ID and password.

"Connecting the web console to catalog servers" on page 516
To start viewing statistics in the web console, you must first connect to catalog servers that you want to monitor. Additional steps are required if your catalog servers have security enabled.

"Monitoring with the **`xscmd`** utility" on page 535
The **`xscmd`** utility replaces the **`xsadmin`** sample utility as a fully supported monitoring and administration tool. With the **`xscmd`** utility, you can display textual information about your WebSphere eXtreme Scale topology.

"Administering with the **`xscmd`** utility" on page 482
With the **`xscmd`** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related information:

"Getting started tutorial lesson 4: Monitor your environment" on page 7
You can use the **`xscmd`** utility and web console tools to monitor your data grid environment.

Monitoring with custom reports

You can build custom reports to save various charts that contain statistics about the catalog service domains, data grids, and container servers in your environment. You can save the custom reports and load them to view again later.

Before you begin

Before you can view statistics with the web console, you must complete the following tasks:

1. Start the web console server. See “Starting and logging on to the web console” on page 514 for more information.
2. Connect your catalog servers to the web console server. See “Connecting the web console to catalog servers” on page 516 for more information.
3. Run active data grids and applications within the servers that are managed by your catalog service domain.

Procedure

- Create a custom report.
  1. Click **Monitor** > **Custom Reports**. A list of the eXtreme Scale domains that you have defined are listed in a tree format. You can expand each of these domains to display the available statistics that you can add to the custom report.
  2. Add charts with the statistics you want to track. Available statistics are indicated by the chart icon (📈). Click one of the statistics that you want to track. Choose **Add to new chart** or **Add to existing chart**. Depending on your selection, the selected statistic either displays in a new chart tab or in the selected existing chart. You can only add a metric to an existing chart if the metrics already on the chart and the new metric use the same units.
- Save a custom report. Saving the custom report saves the statistics in all of the tabs you have created. To save the report, click **Save**.
- Load a custom report. Click **Load** and choose the saved custom report that you want to view.

Monitoring with CSV files

You can enable monitoring data collected for a container server to be written to comma-separated values (CSV) files. These CSV files can contain information about the Java virtual machine (JVM), map, or ObjectGrid instance.

About this task

By enabling monitoring data to be written to CSV files, you can download and analyze historical data for individual container servers. Data begins being collected when you start the server with the server properties that enable the CSV files. You can then download the CSV files at any time and use the files as you choose.

Procedure

1. Update the server properties file with the following properties that are related to enabling the CSV files.

```
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
```

For more information about these properties, see Server properties file.

2. Restart the server to pick up the changes to the server properties file.
3. Download the CSV file. The CSV file is written to the *`server_name`*`/logs` directory.
4. Import the CSV file into the program that you are using to process the data, such as a spreadsheet.

What to do next

For more information about the data that is contained in the CSV files, see "CSV file statistics definitions."

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

CSV file statistics definitions

The CSV files that you can download for a server include statistics that you can use to build historical charts or other information.

Java virtual machine (JVM) statistics log

TimeStamp (column 1)
Specifies the date and time of the statistics snapshot that was taken for the Java virtual machine (JVM).

ServerName (column 2)
Specifies the server name of the JVM.

Hostname (column 3)
Specifies the host name of the JVM.

FreeMemory (column 4)
Specifies the number of available bytes for the JVM.

MaxMemory (column 5)
Specifies the maximum number of bytes that can be allocated for the JVM.

TotalMemory (column 6)
Displays the real memory usage in the server run time.

AvailProcs (column 7)
Displays the number of processors that are available to this catalog service and its maps. For the highest stability, run your servers at 60% processor loading and JVM heaps at 60% heap loading. Spikes can then drive the processor usage to 80–90%, but do not regularly run your servers higher than these levels

Map statistics log

TimeStamp (column 1)
Specifies the date and time of the statistics snapshot that was taken for the map.

MapName (column 2)
Specifies the name of the map.

OgName (column 3)
Specifies the name of the data grid to which this map belongs.

PartitionId (column 4)
Specifies the ID of the partition.

MapSetName (column 5)
Specifies the map set to which this map belongs.

HitRate (column 6)
Displays the hit rate (hit ratio) for the selected map. A high hit rate is desirable. The hit rate indicates how well the data grid is helping to avoid accessing the persistent store.

Count (column 7)
Indicates the number of entries in the data grid since the server started. For example, a value of 100 indicates that the entry is the 100th sample entry that has been gathered since the server started.

TotalGetCount (column 8)
Displays the total number of times the map had to access the persistent store to obtain data.

TotalHitCount (column 9)
Displays the total number of times the requested data was found in the map, avoiding the need to access persistent store.

StartTime (column 10)
Specifies the time that the counters began from last reset call. The resets occur when the server starts or restarts.

LastCount (column 11)
Specifies the amount of time since the last data sample was taken.

LastTotalGetCount (column 12)
Indicates the current total number of get operations from the cache minus the number of get operations in the previous time period.

LastTotalHitCount (column 13)
Indicates the current total number of hits from the cache minus the number of hits in the previous time period.

UsedBytes (column 14)
Displays memory consumption by this map. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

MinUsedBytes (column 15)
Displays the low point in memory consumption by this catalog service and its maps. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

MaxUsedBytes (column 16)
Displays the high point in memory consumption by this catalog service and its maps. The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

LastUsedBytes (column 17)
Indicates the current UsedBytes value minus the UsedBytes value from the previous statistics collection period.

SampleLen (column 18)
Indicates the length, in milliseconds, of the time period during with the data was sampled.

ObjectGrid statistics log

TimeStamp (column 1)
Specifies the date and time of the statistics snapshot that was taken for the data grid.

OgName (column 2)
Specifies the name of the data grid.

PartitionId (column 3)
Specifies the partition ID.

Count (column 4)
Indicates a count of the entries in the data grid that have been gathered since the server started. For example, a value of 100 indicates that the entry is the 100th sample entry that has been gathered since the server started.

Hostname (column 5)
Specifies the host name.

DomainName (column 6)
Specifies the catalog service domain to which this data grid belongs.

MaxTime (column 7)
Displays the time spent by the *most* time-consuming transaction for this server.

MinTime (column 8)
Displays the time spent by the *least* time-consuming transaction for this server.

MeanTime (column 9)
Specifies the average time spent on a transaction.

TotalTime (column 10)
Displays total time spent on transactions for this server, since the time for this server was initialized.

AvgTransTime (column 11)
Displays the average time required to complete a transaction for this server.

AvgThroughPut (column 12)
Displays the average number of transactions per second for this server.

SumOfSquares (column 13)
Specifies the sum of squares value for the transaction time. This value measures the deviation from the mean at the given point in time.

SampleLen (column 14)
Indicates the length, in milliseconds, of the time period during with the data was sampled.

LastDataSample (column 15)
Specifies the time since the last sample was taken.

LastTotalTime (column 16)
Specifies the current total time minus the previous total time for the data sample.

StartTime (column 17)
Indicates the time that the statistics began to be collected since the last reset of the data. The data is reset when the server restarts.

Enabling statistics

WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics. You can use several methods to retrieve the information from the statistics modules.

About this task

For a list of all of the modules on which you can enable statistics, see StatsSpec class.

Procedure

- Enable statistics with the server properties file. You can use the **`statsSpec`** property in the server properties file for the container server to set the statistics specification when you start the server. For more information, see Server properties file.
- Enable statistics with the **`xscmd`** utility. You can use the **`-c setStatsSpec`** command to set the statistics specification at run time. For more information, see “Administering with the **`xscmd`** utility” on page 482.
- Enable statistics programmatically with the StatsSpec interface. For more information, see “Monitoring with the statistics API” on page 532.
- Enable statistics with JMX with the setStatsSpec operation on the DynamicServerMBean. For more information, see Interface DynamicServerMBean.

Example

Some examples of statsSpec strings that you might specify using the properties file, **`xscmd`** utility, or StatsSpec interface follow:

Enable all statistics for all modules:

```
all=enabled
```

Disable all statistics for all modules:

```
all=disabled
```

Enable statistics for all statistics in the OGStatsModule:

```
og.all=enabled
```

Enable statistics for all statistics in the OGStatsModule and MapStatsModule:

```
og.all=enabled;map.all=enabled
```

Enable statistics for only Map Used bytes statistic, and disable everything else:

```
all=disabled;map.usedbytes=enabled
```

Related concepts:

"Statistics modules"
WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

StatsSpec class

Statistics modules

WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics.

Overview

Statistics in WebSphere eXtreme Scale are tracked and contained within StatsModules components. Within the statistics model, several types of statistics modules exist:

OGStatsModule
Provides statistics for an ObjectGrid instance, including transaction response times.

MapStatsModule
Provides statistics for a single map, including the number of entries and hit rate.

QueryStatsModule
Provides statistics for queries, including plan creation and run times.

AgentStatsModule
Provides statistics for DataGrid API agents, including serialization times and run times.

HashIndexStatsModule
Provides statistics for HashIndex query and maintenance run times.

SessionStatsModule
Provides statistics for the HTTP session manager plug-in.

For details about the statistics modules, see the com.ibm.websphere.objectgrid.stats package in the API documentation.

Statistics in a local environment

The model is organized like an n-ary tree (a tree structure with the same degree for all nodes) comprised of all of the StatsModule types mentioned in the previous list. Because of this organization structure, every node in the tree is represented by the StatsFact interface. The StatsFact interface can represent an individual module or a group of modules for aggregation purposes. For example, if several leaf nodes in the tree represent particular MapStatsModule objects, the parent StatsFact node to these nodes contains aggregated statistics for all of the children modules. After you fetch a StatsFact object, you can then use interface to retrieve the corresponding StatsModule.

Much like a tree map, you use a corresponding path or key to retrieve a specific StatsFact. The path is a String[] value that consists of every node that is along the path to the requested fact. For example, you created an ObjectGrid called ObjectGridA, which contains two Maps: MapA and MapB. The path to the StatsModule for MapA would look like `[ObjectGridA, MapA]`. The path to the aggregated statistics for both maps would be: `[ObjectGridA]`.

Statistics in a distributed environment

In a distributed environment, the statistics modules are retrieved using a different path. Because a server can contain multiple partitions, the statistics tree needs to track the partition to which each module belongs. As a result, the path to look up a particular StatsFact object is different. Using the previous example, but adding in that the maps exist within partition 1, the path is `[1, ObjectGridA, MapA]` for retrieving that StatsFact object for MapA.

Related tasks:

"Enabling statistics" on page 529
WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics. You can use several methods to retrieve the information from the statistics modules.

"Monitoring with the statistics API" on page 532
The Statistics API is the direct interface to the internal statistics tree. Statistics are disabled by default, but can be enabled by setting a StatsSpec interface. A StatsSpec interface defines how WebSphere eXtreme Scale should monitor statistics.

"Administering with the **`xscmd`** utility" on page 482
With the **`xscmd`** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

StatsSpec class

Monitoring with the statistics API

The Statistics API is the direct interface to the internal statistics tree. Statistics are disabled by default, but can be enabled by setting a StatsSpec interface. A StatsSpec interface defines how WebSphere eXtreme Scale should monitor statistics.

About this task

You can use the local StatsAccessor API to query data and access statistics on any ObjectGrid instance that is in the same Java virtual machine (JVM) as the running code. For more information about the specific interfaces, see the API documentation. Use the following steps to enable monitoring of the internal statistics tree.

Procedure

1. Retrieve the StatsAccessor object. The StatsAccessor interface follows the singleton pattern. So, apart from problems related to the classloader, one StatsAccessor instance should exist for each JVM. This class serves as the main interface for all local statistics operations. The following code is an example of how to retrieve the accessor class. Call this operation before any other ObjectGrid calls.

```
public class LocalClient
{

   public static void main(String[] args) {

      // retrieve a handle to the StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

   }

}
```

2. Set the data grid StatsSpec interface. Set this JVM to collect all statistics at the ObjectGrid level only. You must ensure that an application enables all statistics that might be needed before you begin any transactions. The following example sets the StatsSpec interface using both a static constant field and using a spec String. Using a static constant field is simpler because the field has already defined the specification. However, by using a spec String, you can enable any combination of statistics that are required.

```
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. Send transactions to the grid to force data to be collected for monitoring. To collect useful data for statistics, you must send transactions to the data grid. The following code excerpt inserts a record into MapA, which is in ObjectGridA. Because the statistics are at the ObjectGrid level, any map within the ObjectGrid yields the same results.

```
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. Query a StatsFact by using the StatsAccessor API. Every statistics path is associated with a StatsFact interface. The StatsFact interface is a generic placeholder that is used to organize and contain a StatsModule object. Before you can access the actual statistics module, the StatsFact object must be retrieved.

```
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. Interact with the StatsModule object. The StatsModule object is contained within the StatsFact interface. You can obtain a reference to the module by using the StatsFact interface. Since the StatsFact interface is a generic interface, you must cast the returned module to the expected StatsModule type. Because this task collects eXtreme Scale statistics, the returned StatsModule object is cast to an OGStatsModule type. After the module is cast, you have access to all of the available statistics.

```
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();

}
```

Related concepts:

"Statistics modules" on page 530
WebSphere eXtreme Scale uses an internal statistics model to track and filter data, which is the underlying structure that all data views use to gather snapshots of statistics.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

StatsSpec class

Monitoring with the `xscmd` utility

The **xscmd** utility replaces the **xsadmin** sample utility as a fully supported monitoring and administration tool. With the **xscmd** utility, you can display textual information about your WebSphere eXtreme Scale topology.

Before you begin

- For the **xscmd** utility to display results, you must have created your data grid topology. Your catalog servers and container servers must be started. See "Starting and stopping stand-alone servers" on page 459 for more information.
- See "Administering with the **xscmd** utility" on page 482 for more information about starting the **xscmd** utility.

About this task

You can use the **xscmd** utility to view the current layout and specific state of the data grid, such as map content. In this example, the layout of the data grid in this task consists of a single *ObjectGridA* data grid with one *MapA* map that belongs to the *MapSetA* map set. This example demonstrates how you can display all active containers within a data grid and print out filtered metrics regarding the map size of the *MapA* map. To see all possible command options, run the **xscmd** utility without any arguments or with the **-help** option.

Procedure

Monitor the environment with the **xscmd** utility.

- To enable statistics for all of the servers, run the following command:
  - UNIX `./xscmd.sh -c setStatsSpec -spec ALL=enabled -g ObjectGridA`
  - Windows `xscmd.bat -c setStatsSpec -spec ALL=enabled -g ObjectGridA`
- To display all online container servers for a data grid, run the following command:
  - UNIX `./xscmd.sh -c showPlacement -g ObjectGridA -ms MapSetA`
  - Windows `xscmd.bat -c showPlacement -g ObjectGridA -ms MapSetA`

  All container information is displayed.

  Attention: To obtain this information when Transport Layer Security/Secure Sockets Layer (TLS/SSL) is enabled, you must start the catalog and container servers with the JMX service port set. To set the JMX service port, you can either use the **-JMXServicePort** option on the **startOgServer** script or you can call the setJMXServicePort method on the ServerProperties interface.
- To display information about the maps for the ObjectGridA data grid, run the following command:
  - UNIX `./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA`
  - Windows `xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA`
- To connect to the catalog service and display information about the MapA map for the entire catalog service domain, run the following command:
  - 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`

  The **xscmd** utility connects to the MBean server that is running on a catalog server. By connecting to a single catalog server, you can retrieve information about the entire catalog service domain. A catalog server can run as a stand-alone process, WebSphere Application Server process, or embedded within a custom application process. Use the **-cep** option to specify the catalog service host name and port. If you include a list of catalog servers for the **-cep** option, the catalog servers must be within the same catalog service domain. You can retrieve statistics for one catalog service domain at a time.

- To display the configured and runtime placement of your configuration, run one of the following commands:
  - `xscmd -c placementServiceStatus`
  - `xscmd -c placementServiceStatus -g ObjectGridA -ms MapSetA`
  - `xscmd -c placementServiceStatus -ms MapSetA`
  - `xscmd -c placementServiceStatus -g ObjectGridA`

  You can scope the command to display placement information for the entire configuration, a single data grid, a single map set, or a combination of a data grid and map set.

Related concepts:

"Statistics overview" on page 511
Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

Related reference:

"Web console statistics" on page 519
Depending on the view you are using in the web console, you can view different statistics about your configuration. These statistics include the used memory, the top used data grids, and the number of cache entries.

stopOgServer script
The **stopOgServer** script stops catalog and container servers.

"Administering with Managed Beans (MBeans)" on page 503
You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

"**xsadmin** tool to **xscmd** tool migration" on page 249
In previous releases, the **xsadmin** tool was a sample command-line utility to monitor the state of the environment. The **xscmd** tool has been introduced as an officially supported administrative and monitoring command-line tool. If you were previously using the **xsadmin** tool, consider migrating your commands to the new **xscmd** tool.

Related information:

"Getting started tutorial lesson 4: Monitor your environment" on page 7
You can use the **xscmd** utility and web console tools to monitor your data grid environment.

"Module 5: Use the **xscmd** utility to monitor data grids and maps" on page 151
You can use the **xscmd** utility to show the primary data grids and map sizes of the `Grid` data grid. The **xscmd** tool uses the MBean to query all of the data grid artifacts, such as primary shards, replica shards, container servers, map sizes, and other data.

"Module 5: Use the **xscmd** tool to monitor data grids and maps" on page 127
You can use the **xscmd** tool to show the primary data grids and map sizes of the `Grid` data grid. The **xscmd** tool uses the MBean to query all of the data grid artifacts, such as primary shards, replica shards, container servers, map sizes, and so on.

API documentation: Package com.ibm.websphere.objectgrid.management Interface PlacementServiceMBean

Monitoring with WebSphere Application Server PMI

WebSphere eXtreme Scale supports Performance Monitoring Infrastructure (PMI) when running in a WebSphere Application Server or WebSphere Extended Deployment application server. PMI collects performance data on runtime applications and provides interfaces that support external applications to monitor performance data. You can use the administrative console or the wsadmin tool to access monitoring data.

Before you begin

You can use PMI to monitor your environment when you are using WebSphere eXtreme Scale combined with WebSphere Application Server.

About this task

WebSphere eXtreme Scale uses the custom PMI feature of WebSphere Application Server to add its own PMI instrumentation. With this approach, you can enable and disable WebSphere eXtreme Scale PMI with the administrative console or with Java Management Extensions (JMX) interfaces in the wsadmin tool. In addition, you can access WebSphere eXtreme Scale statistics with the standard PMI and JMX interfaces that are used by monitoring tools, including the Tivoli Performance Viewer.

Procedure

1. Enable eXtreme Scale PMI. You must enable PMI to view the PMI statistics. See “Enabling PMI” for more information.
2. Retrieve eXtreme Scale PMI statistics. View the performance of your eXtreme Scale applications with the Tivoli Performance Viewer. See “Retrieving PMI statistics” on page 539 for more information.

What to do next

For more information about the wsadmin tool, see “Accessing Managed Beans (MBeans) using the wsadmin tool” on page 504.

Enabling PMI

You can use WebSphere Application Server Performance Monitoring Infrastructure (PMI) to enable or disable statistics at any level. For example, you can choose to enable the map hit rate statistic for a particular map, but not the number of entry statistic or the loader batch update time statistic. You can enable PMI in the administrative console or with scripting.

Before you begin

- Your application server must be started and have an eXtreme Scale-enabled application installed.
- To enable PMI with wsadmin scripting, you also must be able to log in and use the wsadmin tool. For more information about the wsadmin tool, see WebSphere Application Server Information Center: Scripting the application serving environment (wsadmin).

About this task

Use WebSphere Application Server PMI to provide a granular mechanism with which you can enable or disable statistics at any level. For example, you can choose to enable the map hit rate statistic for a particular map, but not the number of entry or the loader batch update time statistics. This section shows how to use the administrative console and wsadmin scripts to enable ObjectGrid PMI.

Procedure

- **Enable PMI in the administrative console.**
  1. In the administrative console, click **Monitoring and Tuning** > **Performance Monitoring Infrastructure** > *server_name*.
  2. Verify that Enable Performance Monitoring Infrastructure (PMI) is selected. This setting is enabled by default. If the setting is not enabled, select the check box, then restart the server.
  3. Click **Custom**. In the configuration tree, select the ObjectGrid and ObjectGrid Maps module. Enable the statistics for each module.

  The transaction type category for ObjectGrid statistics is created at runtime. You can see only the subcategories of the ObjectGrid and Map statistics on the **Runtime** tab.
- **Enable PMI with scripting.**
  1. Open a command line prompt. Navigate to the *was_root*/`bin` directory. Type **`wsadmin`** to start the wsadmin command line tool.
  2. Modify the eXtreme Scale PMI runtime configuration. Verify that PMI is enabled for the server using the following commands:
     ```
     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 $pmi.
     ```
     If PMI is not enabled, run the following commands to enable PMI:
     ```
     wsadmin>$AdminConfig modify $pmi {{enable true}}
     wsadmin>$AdminConfig save
     ```
     If you need to enable PMI, restart the server.
  3. Set variables for changing the statistic set to a custom set using the following commands:
     ```
     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>$sigs set 0 java.lang.String
     ```
  4. Set statistic set to custom using the following command:
     ```
     wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
     ```
  5. Set variables to enable the objectGridModule PMI statistic using the following commands:
     ```
     wsadmin>set params [java::new {java.lang.Object[]} 2]
     wsadmin>$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. Set the statistics string using the following command:
   ```
   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. Set the statistics string using the following command:
   ```
   wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
   ```

These steps enable eXtreme Scale runtime PMI, but do not modify the PMI configuration. If you restart the application server, the PMI settings are lost except for the main PMI enablement.

Example

You can perform the following steps to enable PMI statistics for the sample application:

1. Launch the application using the `http://host:port/ObjectGridSample` Web address, where host and port are the host name and HTTP port number of the server where the sample is installed.
2. In the sample application, click ObjectGridCreationServlet, and then click action buttons 1, 2, 3, 4, and 5 to generate actions to the ObjectGrid and maps. Do not close this servlet page right now.
3. In the administrative console, click **Monitoring and Tuning** > **Performance Monitoring Infrastructure** > *server_name* Click the **Runtime** tab.
4. Click the **Custom** radio button.
5. Expand the ObjectGrid Maps module in the runtime tree, then click the clusterObjectGrid link. Under the ObjectGrid Maps group, there is an ObjectGrid instance called clusterObjectGrid, and under the clusterObjectGrid group four maps exist: counters, employees, offices, and sites. In the ObjectGrids instance, there is the clusterObjectGrid instance, and under that instance is a transaction type called DEFAULT.
6. You can enable the statistics of interest to you. For example, you can enable number of map entries for employees map, and transaction response time for the DEFAULT transaction type.

What to do next

After PMI is enabled, you can view PMI statistics with the administrative console or through scripting.

Retrieving PMI statistics

By retrieving PMI statistics, you can see the performance of your eXtreme Scale applications.

Before you begin

- Enable PMI statistics tracking for your environment. See "Enabling PMI" on page 537 for more information.
- The paths in this task are assuming you are retrieving statistics for the sample application, but you can use these statistics for any other application with similar steps.

- If you are using the administrative console to retrieve statistics, you must be able to log in to the administrative console. If you are using scripting, you must be able to log in to wsadmin.

About this task

You can retrieve PMI statistics to view in Tivoli Performance Viewer by completing steps in the administrative console or with scripting.

- Administrative console steps
- Scripting steps

For more information about the statistics that can be retrieved, see "PMI modules" on page 541.

Procedure

- Retrieve PMI statistics in the administrative console.
  1. In the administrative console, click **Monitoring and tuning** > **Performance viewer** > **Current activity**
  2. Select the server that you want to monitor using Tivoli Performance Viewer, then enable the monitoring.
  3. Click the server to view the Performance viewer page.
  4. Expand the configuration tree. Click **ObjectGrid Maps** > **clusterObjectGrid** select **employees**. Expand **ObjectGrids** > **clusterObjectGrid** and select **DEFAULT**.
  5. In the ObjectGrid sample application, go to the ObjectGridCreationServlet servlet , click button 1, then populate maps. You can view the statistics in the viewer.
- Retrieve PMI statistics with scripting.
  1. On a command line prompt, navigate to the *was_root*/`bin` directory. Type `wsadmin` to start the wsadmin tool.
  2. Set variables for the environment using the following commands:
     ```
     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. Set variables to get mapModule statistics using the following commands:
     ```
     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. Get mapModule statistics using the following command:
     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
     ```
  5. Set variables to get objectGridModule statistics using the following commands:
     ```
     wsadmin>set params2 [java::new {java.lang.Object[]} 3]
     wsadmin>$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. Get objectGridModule statistics using the following command:

```
wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
```

Results

You can view statistics in the Tivoli Performance Viewer.

PMI modules

You can monitor the performance of your applications with the performance monitoring infrastructure (PMI) modules.

objectGridModule

The objectGridModule contains a time statistic: transaction response time. A transaction is defined as the duration between the Session.begin method call and the Session.commit method call. This duration is tracked as the transaction response time. The root element of the objectGridModule, "root", serves as the entry point to the WebSphere eXtreme Scale statistics. This root element has ObjectGrids as its child elements, which have transaction types as their child elements. The response time statistic is associated with each transaction type.

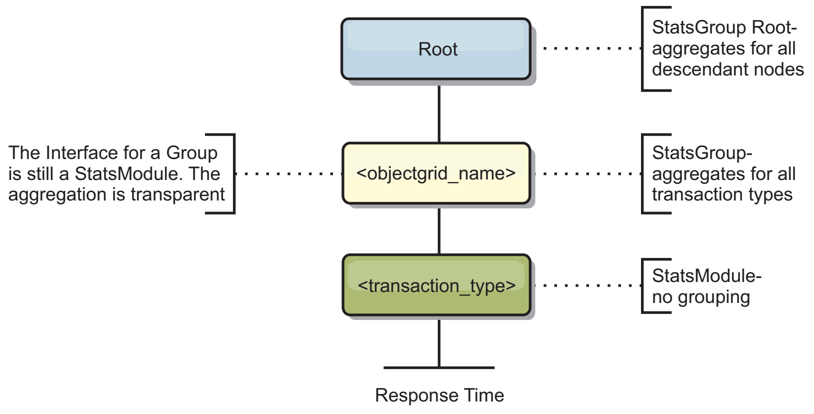

Root
StatsGroup Root-aggregates for all descendant nodes
The Interface for a Group is still a StatsModule. The aggregation is transparent
<objectgrid_name>
StatsGroup-aggregates for all transaction types
<transaction_type>
StatsModule-no grouping
Response Time

Figure 58. ObjectGridModule module structure

The following diagram shows an example of the ObjectGridModule structure. In this example, two ObjectGrid instances exist in the system: ObjectGrid A and ObjectGrid B. The ObjectGrid A instance has two types of transactions: A and default. The ObjectGrid B instance has only the default transaction type.

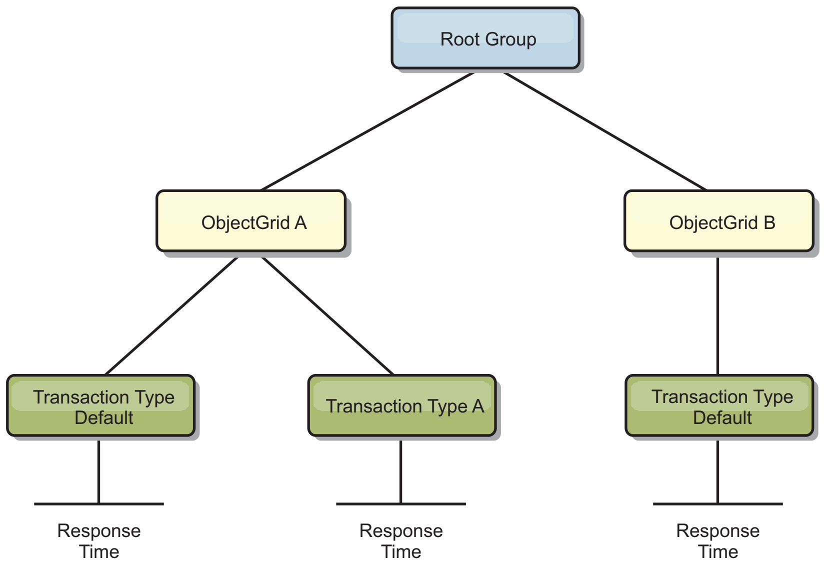


Figure 59. ObjectGridModule module structure example

Transaction types are defined by application developers because they know what types of transactions their applications use. The transaction type is set using the following Session.setTransactionType(String) method:

```
/**
* 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);
```

The following example sets transaction type to `updatePrice`:

```
// 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();
```

The first line indicates that the subsequent transaction type is updatePrice. An updatePrice statistic exists under the ObjectGrid instance that corresponds to the session in the example. Using Java Management Extensions (JMX) interfaces, you can get the transaction response time for updatePrice transactions. You can also get the aggregated statistic for all types of transactions on the specified ObjectGrid instance.

mapModule

The mapModule contains three statistics that are related to eXtreme Scale maps:

- **Map hit rate** - *BoundedRangeStatistic*: Tracks the hit rate of a map. Hit rate is a float value between 0 and 100 inclusively, which represents the percentage of map hits in relation to map get operations.
- **Number of entries**-*CountStatistic*: Tracks the number of entries in the map.
- **Loader batch update response time**-*TimeStatistic*: Tracks the response time that is used for the loader batch-update operation.

The root element of the mapModule, "root", serves as the entry point to the ObjectGrid Map statistics. This root element has ObjectGrids as its child elements, which have maps as their child elements. Every map instance has the three listed statistics. The mapModule structure is shown in the following diagram:

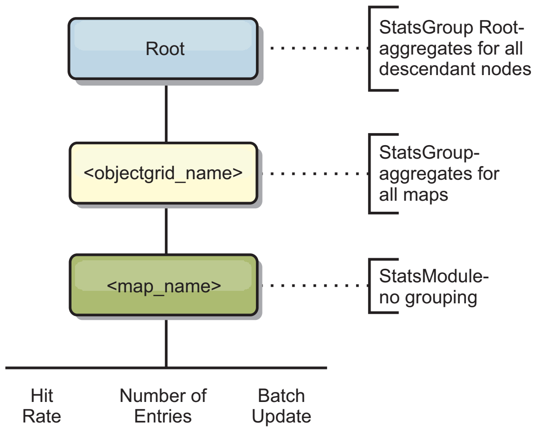
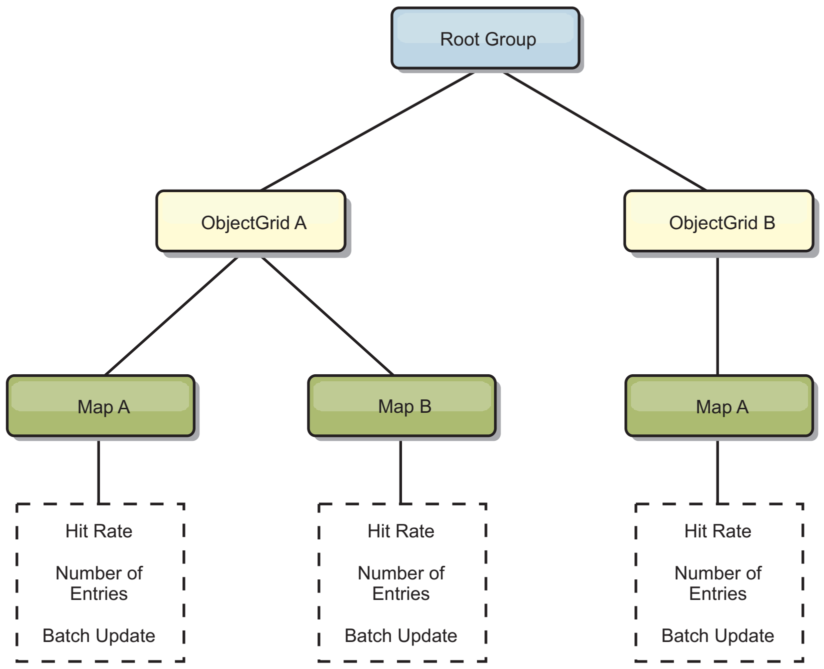


Figure 60. mapModule structure

The following diagram shows an example of the mapModule structure:

Figure 61. mapModule module structure example



hashIndexModule

The hashIndexModule contains the following statistics that are related to Map-level indexes:

- **Find Count**-*CountStatistic*: The number of invocations for the index find operation.
- **Collision Count**-*CountStatistic*: The number of collisions for the find operation.
- **Failure Count**-*CountStatistic*: The number of failures for the find operation.
- **Result Count**-*CountStatistic*: The number of keys returned from the find operation.
- **BatchUpdate Count**-*CountStatistic*: The number of batch updates against this index. When the corresponding map is changed in any manner, the index will have its doBatchUpdate() method called. This statistic will tell you how frequently your index is changing or being updated.
- **Find Operation Duration Time**-*TimeStatistic*: The amount of time the find operation takes to complete

The root element of the hashIndexModule, "root", serves as the entry point to the HashIndex statistics. This root element has ObjectGrids as its child elements, ObjectGrids have maps as their child elements, which finally have HashIndexes as their child elements and leaf nodes of the tree. Every HashIndex instance has the three listed statistics. The hashIndexModule structure is shown in the following diagram:

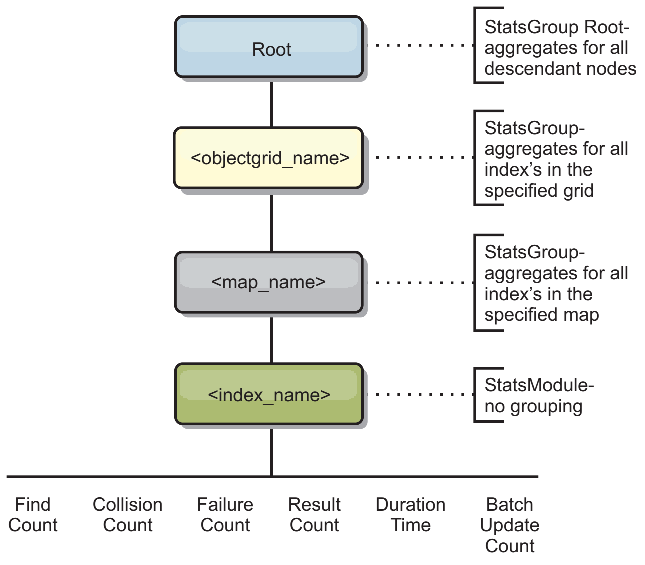

Root
StatsGroup Root-aggregates for all descendant nodes
<objectgrid_name>
StatsGroup-aggregates for all index's in the specified grid
<map_name>
StatsGroup-aggregates for all index's in the specified map
<index_name>
StatsModule-no grouping
Find Count
Collision Count
Failure Count
Result Count
Duration Time
Batch Update Count

Figure 62. hashIndexModule module structure

The following diagram shows an example of the hashIndexModule structure:

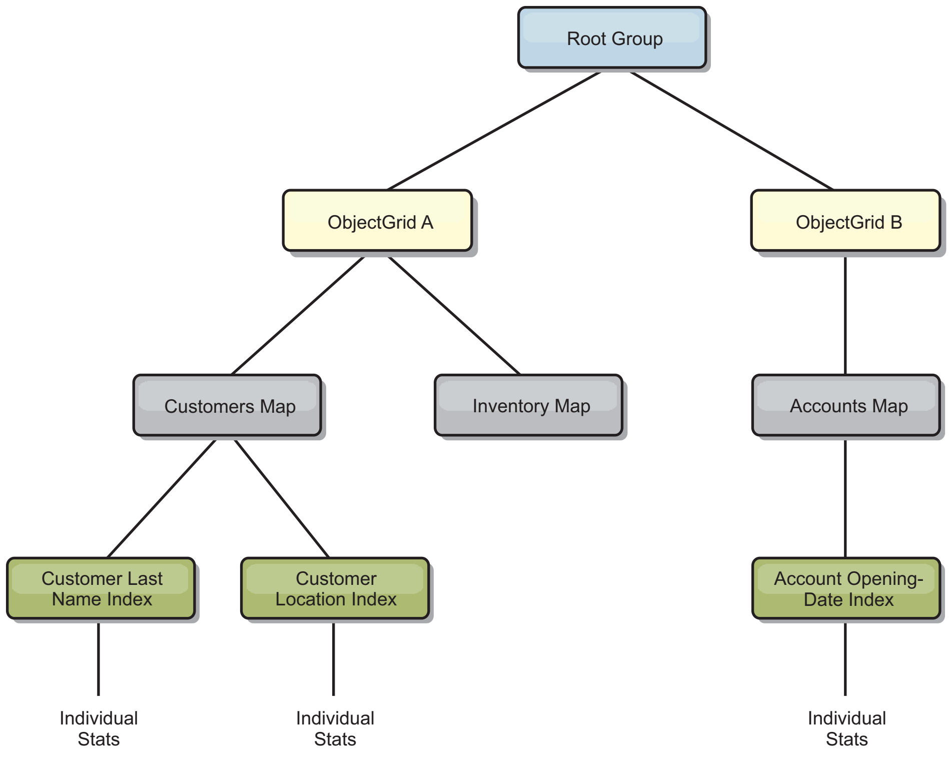


Figure 63. hashIndexModule module structure example

agentManagerModule

The agentManagerModule contains statistics that are related to map-level agents:

- **Reduce Time**: *TimeStatistic* - The amount of time for the agent to finish the reduce operation.
- **Total Duration Time**: *TimeStatistic* - The total amount of time for the agent to complete all operations.

- **Agent Serialization Time**: *TimeStatistic* - The amount of time to serialize the agent.
- **Agent Inflation Time**: *TimeStatistic* - The amount of time it takes to inflate the agent on the server.
- **Result Serialization Time**: *TimeStatistic* - The amount of time to serialize the results from the agent.
- **Result Inflation Time**: *TimeStatistic* - The amount of time to inflate the results from the agent.
- **Failure Count**: *CountStatistic* - The number of times that the agent failed.
- **Invocation Count**: *CountStatistic* - The number of times the AgentManager has been invoked.
- **Partition Count**: *CountStatistic* - The number of partitions to which the agent is sent.

The root element of the agentManagerModule, "root", serves as the entry point to the AgentManager statistics. This root element has ObjectGrids as its child elements, ObjectGrids have maps as their child elements, which finally have AgentManager instances as their child elements and leaf nodes of the tree. Every AgentManager instance has statistics.

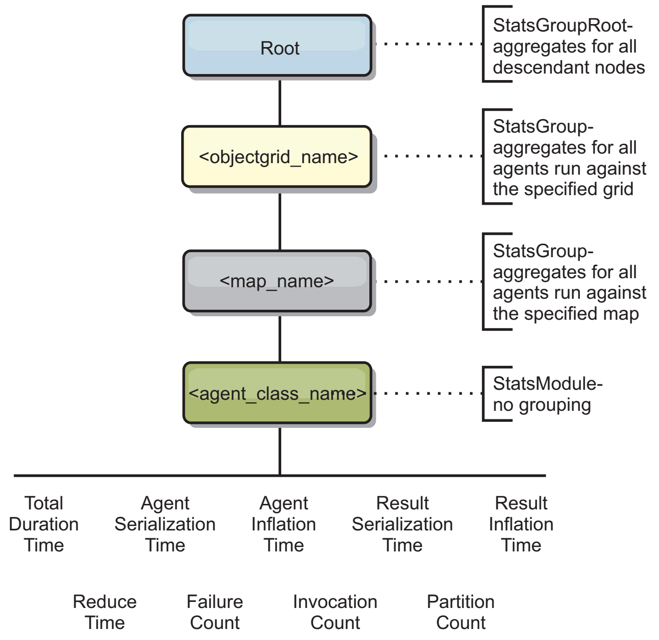


Figure 64. agentManagerModule structure

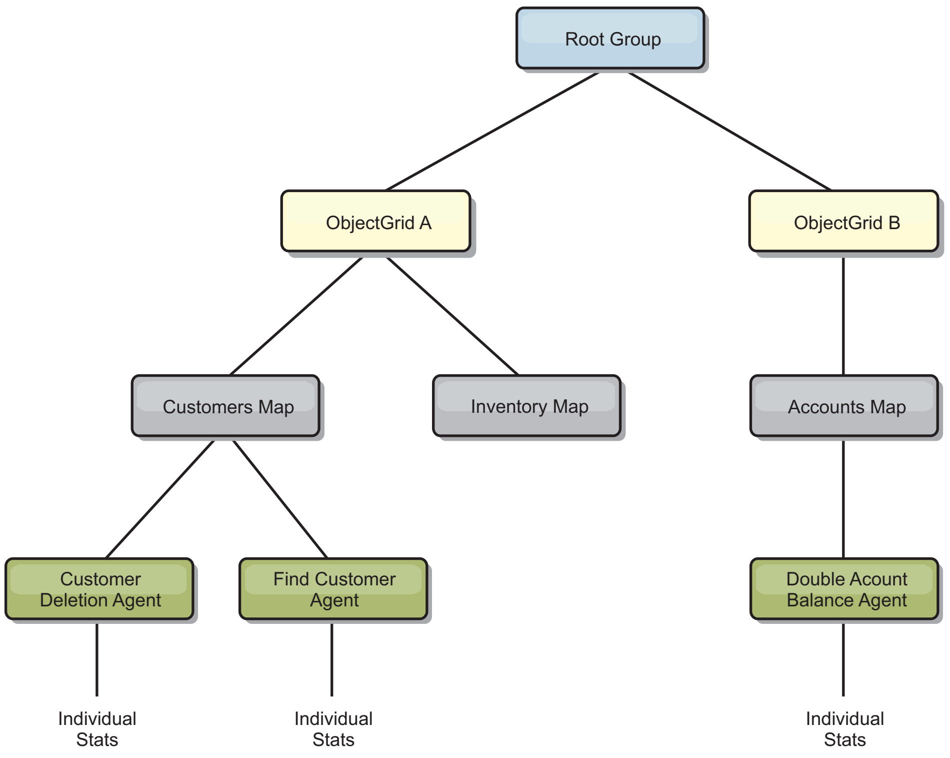


Figure 65. agentManagerModule structure example

queryModule

The queryModule contains statistics that are related to eXtreme Scale queries:

- **Plan Creation Time**: *TimeStatistic* - The amount of time to create the query plan.
- **Execution Time**: *TimeStatistic* - The amount of time to run the query.
- **Execution Count**: *CountStatistic* - The number of times the query has been run.
- **Result Count**: *CountStatistic* - The count for each the result set of each query run.
- **FailureCount**: *CountStatistic* - The number of times the query has failed.

The root element of the queryModule, "root", serves as the entry point to the Query Statistics. This root element has ObjectGrids as its child elements, which have Query objects as their child elements and leaf nodes of the tree. Every Query instance has the three listed statistics.

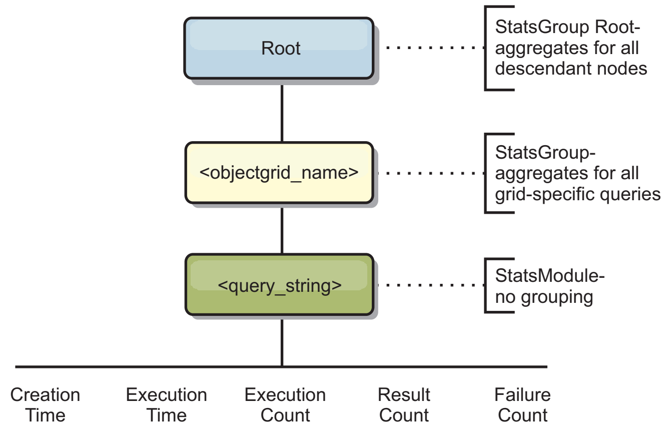


Figure 66. queryModule structure

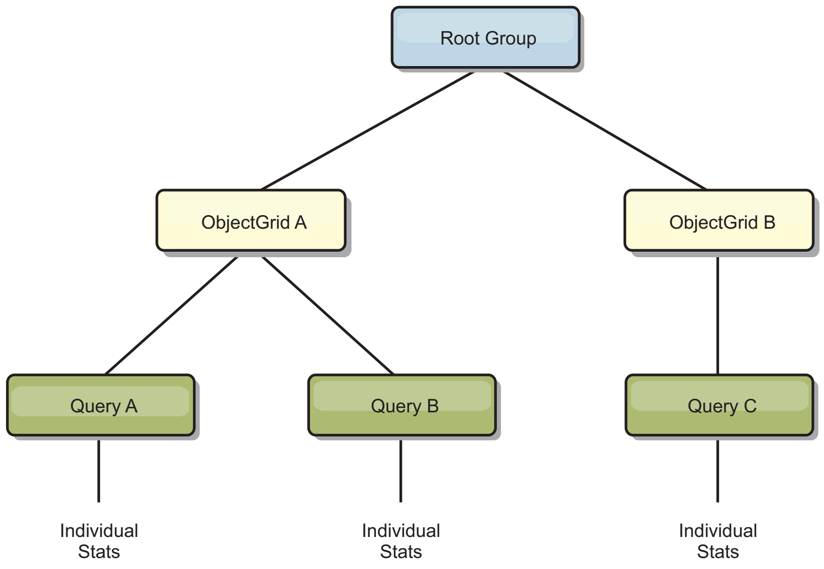


Figure 67. QueryStats.jpg queryModule structure example

Accessing Managed Beans (MBeans) using the wsadmin tool

You can use the wsadmin utility provided in WebSphere Application Server to access managed bean (MBean) information.

Procedure

Run the wsadmin tool from the `bin` directory in your WebSphere Application Server installation. The following example retrieves a view of the current shard placement in a dynamic eXtreme Scale. You can run the wsadmin tool from any installation where eXtreme Scale is running. You do not have to run the wsadmin tool on the catalog service.

```
$ 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>
```

Related concepts:

"Statistics overview" on page 511
Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

Related reference:

"Administering with Managed Beans (MBeans)" on page 503
You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

Related information:

API documentation: Package com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

Monitoring server statistics with managed beans (MBeans)

You can used managed beans (MBeans) to track statistics in your environment.

Before you begin

For the attributes to be recorded, you must enable statistics. You can enable statistics on the server, or enable HTTP session statistics to track attributes on your client application. For more information on how to enable HTTP session statistics, see xref.

You can enable statistics in one of the following ways:

- **With the server properties file:**

  You can enable statistics in the server properties file with a key-value entry of `statsSpec=<StatsSpec>`. Some examples of possible settings follow:
  - To enable all statistics, use `statsSpec=all=enabled`
  - To enable only ObjectGrid statistics, use `statsSpec=og.all=enabled`. To see a description of all possible statistics specifications, see the StatsSpec API in the API documentation.

  For more information about the server properties file, see Server properties file.
- **With a managed bean:**

  You can enable statistics using the StatsSpec attribute on the ObjectGrid MBean. For more information, see the StatsSpec API in the API documentation.
- **Programmatically:**

You can also enable statistics programmatically with the StatsAccessor interface, which is retrieved with the StatsAccessorFactory class. Use this interface in a client environment or when you need to monitor a data grid that is running in the current process.

Procedure

- **Access MBean statistics using the wsadmin tool.**

  For more information, see "Accessing Managed Beans (MBeans) using the wsadmin tool" on page 504.
- **Access MBean statistics programmatically.**

  For more information, see "Accessing Managed Beans (MBeans) programmatically" on page 504.

Example

For an example of how to use managed beans, see Sample: `xsadmin` utility.

Related concepts:

"Statistics overview" on page 511
Statistics in WebSphere eXtreme Scale are built on an internal statistics tree. The StatsAccessor API, Performance Monitoring Infrastructure (PMI) modules, and MBean API are built from the internal tree.

Related reference:

"Administering with Managed Beans (MBeans)" on page 503
You can use several different types of Java Management Extensions (JMX) MBeans to administer and monitor deployments. Each MBean refers to a specific entity, such as a map, data grid, server, or service.

Related information:

API documentation: Package com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

Monitoring with vendor tools

WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli Monitoring and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

Related tasks:

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Monitoring with the IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale

The IBM Tivoli Enterprise Monitoring Agent is a feature-rich monitoring solution that you can use to monitor databases, operating systems and servers in distributed and host environments. WebSphere eXtreme Scale includes a customized agent that you can use to introspect eXtreme Scale management beans. This solution works effectively for both stand-alone eXtreme Scale and WebSphere Application Server deployments.

Before you begin

- Install WebSphere eXtreme Scale Version 7.0.0 or later.

  Also, statistics must be enabled in order to collect statistical data from WebSphere eXtreme Scale servers. Various options for enabling statistics are described in "Monitoring server statistics with managed beans (MBeans)" on page 549 and Sample: `xsadmin` utility
- Install IBM Tivoli Monitoring Version 6.2.1 with fix pack 2 or later.
- Install the Tivoli OS agent on each server or host on which eXtreme Scale servers run.
- Install the WebSphere eXtreme Scale agent, which you can download for free from the IBM Open Process Automation Library (OPAL) site.

Complete the following steps to install and configure the Tivoli Monitoring Agent:

Procedure

1. Install the Tivoli Monitoring Agent for WebSphere eXtreme Scale.

   Download the Tivoli installation image and extract its files to a temporary directory.
2. Install eXtreme Scale application support files.

   Install eXtreme Scale application support on each of the following deployments.
   - Tivoli Enterprise Portal Server (TEPS)

- Enterprise Desktop client (TEPD)
- Tivoli Enterprise Monitoring Server (TEMS)

a. From the temporary directory that you created, start a new command window and run the appropriate executable file for your platform. The installation script automatically detects your Tivoli deployment type (TEMS, TEPD, or TEPS). You can install any type on a single host or on multiple hosts; and all of the three deployment types require the installation of the eXtreme Scale agent application support files.

b. In the **Installer** window, verify that the selections for the Tivoli Components deployed are correct. Click **Next**.

c. If you are prompted, submit your hostname and administrative credentials. Click **Next**.

d. Select the **Monitoring Agent for WebSphere eXtreme Scale**. Click **Next**.

e. You are notified of what installation actions are to be performed. Click **Next**, and you can see the progress of the installation until completion.

After completing the procedure, all application support files required by WebSphere eXtreme Scale agent are installed.

3. Install the agent on each of the eXtreme Scale nodes.

   You install a Tivoli OS agent on each of the computers. You do not need to configure or start this agent. Use the same installation image from the previous step to run the platform specific executable file.

   As a guideline, you need to install only one agent per host. Each agent is capable of supporting many instances of eXtreme Scale servers. For best performance, use one agent instance for monitoring about 50 eXtreme Scale servers.

   a. From the installation wizard welcome screen, click **Next** to open the screen to specify installation path information.

   b. For the **Tivoli Monitoring installation directory** field, enter or browse to `C:\IBM\ITM` (or `/opt/IBM/ITM`). Then for the **Location for installable media** field, verify that the displayed value is correct and click **Next**.

   c. Select the components you want to add, such as **Perform a local install of the solution** and click **Next**.

   d. Select the applications for which to add support for by selecting the application, such as **Monitoring Agent for WebSphere eXtreme Scale**, and click **Next**.

   e. You can see the progress until application support is added successfully.

   Note: Repeat these steps on each of the eXtreme Scale nodes. You can also use silent installation. See the IBM Tivoli Monitoring Information Center for more information about silent installation.

4. Configure the WebSphere eXtreme Scale agent.

   Each of the agents installed need to be configured to monitor any catalog server, eXtreme Scale server, or both.

   The steps to configure Windows and UNIX platforms are different. Configuration for the Windows platform is completed with the **Manage Tivoli Monitoring Services** user interface. Configuration for UNIX platforms is command-line based.

   Windows Use the following steps to initially configure the agent on Windows

a. From the **Manage Tivoli Enterprise Monitoring Services** window, click **Start** > **All Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
b. Right click on **Monitoring Agent for WebSphere eXtreme Scale** and select **Configure using default**, which opens a window to create a unique instance of the agent.
c. Choose a unique name: for example, `instance1`, and click **Next**.

- If you plan to monitor stand-alone eXtreme Scale servers, complete the following steps:
  a. Update the Java parameters, ensure that the **Java Home** value is correct. JVM arguments can be left empty. Click **Next**.
  b. Select the type of **MBean server connection type**, Use `JSR-160-Complaint Server` for stand-alone eXtreme Scale servers. Click **Next**.
  c. If security is enabled, update **User ID** and **Password** values. Leave the **JMX service URL** value as is. You override this value later. Leave the **JMX Class Path Information** field as it is. Click **Next**.

  To configure the servers for the agent on Windows, complete the following steps:
  a. Set up subnode instances of eXtreme Scale servers in the **WebSphere eXtreme Scale Grid Servers** pane. If no container servers exist on your computer, click **Next** to proceed to the catalog service pane.
  b. If multiple eXtreme Scale container servers exist on your computer, configure the agent to monitor each one server.
  c. You can add as many eXtreme Scale servers as you require, if their names and ports are unique, by clicking **New**. (When an eXtreme Scale server is started, a JMXPort value must be specified.)
  d. After you configure the container servers, click **Next**, which brings you to the **WebSphere eXtreme Scale Catalog Servers** pane.
  e. If you have no catalog servers, click **OK**. If you have catalog servers, add a new configuration for each server, as you did with the container servers. Again, choose a unique name, preferably the same name that is used when starting the catalog service. Click **OK** to finish.
- If you plan to monitor servers for the agent on eXtreme Scale servers that are embedded within a WebSphere Application Server process, complete the following steps:
  a. Update the Java parameters, ensure that the **Java Home** value is correct. JVM arguments can be left empty. Click **Next**.
  b. Select the **MBean server connection type**. Select the WebSphere Application Server version that is appropriate for your environment. Click **Next**.
  c. Ensure that the WebSphere Application Server information in the panel is correct. Click **Next**.
  d. Add only one subnode definition. Give the subnode definition a name, but do not update the port definition. Within WebSphere Application Server environment, data can be collected from all the application server that are managed by the node agent that is running on the computer. Click **Next**.
  e. If there no catalog servers exist in the environment, click **OK**. If you have catalog servers, add a new configuration for each catalog server, similarly

to the container servers. Choose a unique name for the catalog service, preferably the same name that you use when starting the catalog service. Click **OK** to finish.

Note: The container servers do not need to be collocated with the catalog service.

Now that the agent and servers are configured and ready, on the next window, right click on `instance1` to start the agent.

UNIX To configure the agent on the UNIX platform on the command line, complete the following steps:

An example follows for stand-alone servers that uses a JSR160 Compliant connection type. The example shows three eXtreme Scale containers on the single host (rhea00b02) and the JMX listener addresses are 15000,15001 and 15002 respectively. There are no catalog servers.

Output from the configuration utility displays in *`monospace italics`*, while the user response is in **`monospace bold`**. (If no user response was required, the default was selected by pressing the enter key.)

```
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...
```

The previous example creates an agent instance called "inst1", and updates the Java Home settings. The eXtreme Scale container servers are configured, but the catalog service is not configured.

Note: The previous procedure creates a text file of the following format in the directory: `<ITM_install>/config/<host>_xt_<instance name>.cfg`.

Example: `rhea00b02_xt_inst1.cfg`

It is best to edit this file with your choice of plain text editor. An example of the content of such the file follows:

```
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 }  ]]
```

An example that shows a configuration on a WebSphere Application Server deployment follows:

```
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 #
```

For WebSphere Application Server deployments, you do not need to create multiple sub nodes. The eXtreme Scale agent connects to the node agent to gather all the information from application servers for which it is responsible.

`SECTION=CAT` signifies a catalog service line whereas `SECTION=OGS` signifies an eXtreme Scale server configuration line.

5. Configure the JMX port for all eXtreme Scalecontainer servers.

   When eXtreme Scale container servers are started, without specifying the **`-JMXServicePort`** argument, an MBean server is assigned a dynamic port. The agent needs to know in advance with which JMX port to communicate. The agent does not work with dynamic ports.

   When you start the servers, you must specify the `-JMXServicePort <port_number>` argument when you start the eXtreme Scale server using the start server command. Running this command ensures that the JMX server within the process listens to a static pre-defined port.

   For the previous examples in a UNIX installation, two eXtreme Scale servers need to be started with ports set:

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

   a. Start the eXtreme Scale agent.

      Assuming the `inst1` instance was created, as in the previous example, issue the following commands.

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

   b. Stop the eXtreme Scale agent.

      Assuming “inst1” was the instance created, as in the previous example, issue the following commands.

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

6. Enable Statistics for all eXtreme Scale container servers.

   The agent uses the eXtreme Scale statistics MBeans to record statistics. The eXtreme Scale statistics specification must be enabled using one of the following methods.

   - Configure server properties to enable all statistics when the container servers are started: all=enabled.
   - Use the xsadmin sample utility to enable statistics for all active containers using the -setstatsspec all=enabled parameters.

Results

After all servers are configured and started, MBeans data is displayed on the IBM Tivoli Portal console. Predefined workspaces show graphs and data metrics at each node level.

The following workspaces are defined: **eXtreme Scale Grid Servers** node for all nodes monitored.

- eXtreme Scale Transactions View
- eXtreme Scale Primary Shard View
- eXtreme Scale Memory View
- eXtreme Scale ObjectMap View

You can also configure your own workspace. For more information, see the information about customizing workspaces in the IBM Tivoli Monitoring Information Center.

Monitoring eXtreme Scale applications with CA Wily Introscope

CA Wily Introscope is a third-party management product that you can use to detect and diagnose performance problems in enterprise application environments. eXtreme Scale includes details on configuring CA Wily Introscope to introspect select portions of the eXtreme Scale run time to quickly view and validate eXtreme Scale applications. CA Wily Introscope works effectively for both stand-alone and WebSphere Application Server deployments.

Overview

To monitor eXtreme Scale applications with CA Wily Introscope, you must put settings into the ProbeBuilderDirective (PBD) files that give you access to the monitoring information for eXtreme Scale.

Attention: The instrumentation points for Introscope might change with each fix pack or release. When you install a new fix pack or release, check the documentation for any changes in the instrumentation points.

You can configure CA Wily Introscope ProbeBuilderDirective (PBD) files to monitor your eXtreme Scale applications. CA Wily Introscope is an application management product with which you can proactively detect, triage and diagnose performance problems in your complex, composite and Web application environments.

PBD file settings for monitoring the catalog service

You can use one or more of the following settings in your PBD file to monitor the catalog service.

```
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}"
```

Classes for monitoring the catalog service

HAControllerImpl
The HAControllerImpl class handles core group life cycle and feedback events. You can monitor this class to get an indication of the core group structure and changes.

ServerAgent
The ServerAgent class is responsible for communicating core group events with the catalog service. You can monitor the various heartbeat calls to spot major events.

PlacementServiceImpl
The PlacementServiceImpl class coordinates the containers. You can use the methods on this class to monitor server join and placement events.

BalanceGridEventListener
The BalanceGridEventListener class controls the catalog leadership. You can monitor this class to get an indication of which catalog service is currently acting as the leader.

PBD file settings for monitoring the containers

You can use one or more of the following settings in your PBD file to monitor the containers.

```
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}"
```

Classes for monitoring the containers

ShardImpl
The ShardImpl class has the processMessage method. The processMessage method is the method for client requests. With this method, you can get server side response time and request counts. By watching the counts across all the servers and monitoring heap utilization, you can determine if the grid is balanced.

CheckpointIterator
: The CheckpointIterator class has the activateListener method call which puts primaries into peer mode. When the primaries are put into peer mode, the replica is up to date with the primary after the method completes. When a replica is regenerating from a full primary, this operation can take an extended period of time. The system is not fully recovered until this operation completes, so you can use this class to monitor the progress of the operation.

CommittedLogSequenceListenerProxy
: The CommittedLogSequenceListenerProxy class has two methods of interest. The applyCommitted method runs for every transaction and the sendApplyCommitted runs as the replica is pulling information. The ratio of how often these two methods run can give you some indication of how well the replica is able to keep up with the primary.

PBD file settings for monitoring the clients

You can use one or more of the following settings in your PBD file to monitor the clients.

```
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}"
```

Classes for monitoring the clients

ORBClientCoreMessageHandler
: The ORBClientCoreMessageHandler class is responsible for sending application requests to the containers. You can monitor the sendMessage method for client response time and number of requests.

ClusterStore
: The ClusterStore class holds the routing information on the client side.

BaseMap
: The BaseMap class has the evictMapEntries method that is called when the evictor wants to remove entries from the map.

SelectionServiceImpl
: The SelectionServiceImpl class makes the routing decisions. If the client is making failover decisions, you can use this class to see the actions that are completed from the decisions.

ObjectGridImpl
: The ObjectGridImpl class has the getSession method that you can monitor to see the number of requests to this method.

Monitoring eXtreme Scale with Hyperic HQ

Hyperic HQ is a third-party monitoring solution that is available freely as an open source solution or as an enterprise product. WebSphere eXtreme Scale includes a plug-in that allows Hyperic HQ agents to discover eXtreme Scale container servers and to report and aggregate statistics using eXtreme Scale management beans. You can use Hyperic HQ to monitor stand-alone eXtreme Scale deployments.

Before you begin

- This set of instructions is for Hyperic Version 4.0. If you have a newer version of Hyperic, see the Hyperic documentation for information such as the path names and how to start agents and servers.
- Download the Hyperic server and agent installations. One server installation must be running. To detect all of the eXtreme Scale servers, a Hyperic agent must be running on each machine on which an eXtreme Scale server is running. See the Hyperic website for download information and documentation support.
- You must have access to the `objectgrid-plugin.xml` and `hqplugin.jar` files. These files are in the *wxs_install_root*/`hyperic/etc` directory.

About this task

By integrating eXtreme Scale with Hyperic HQ monitoring software, you can graphically monitor and display metrics about the performance of your environment. You set up this integration by using a plug-in implementation on each agent.

Procedure

1. Start your eXtreme Scale servers. The Hyperic plug-in looks at the local processes to attach to the Java virtual machines that are running eXtreme Scale. To properly attach to the Java virtual machines, each server must be started with the **-jmxServicePort** option. For information about starting servers with the **-jmxServicePort** option, see “Starting and stopping stand-alone servers” on page 459.
2. Put the `extremescale-plugin.xml` file and the `wxshyperic.jar` file in the appropriate server and agent plug-in directories in your Hyperic configuration. To integrate with Hyperic, both the agent and server installations must have access to the plug-in and Java archive (JAR) files. Although the server can dynamically swap configurations, you should complete the integration before you start any of the agents.
   a. Place the `extremescale-plugin.xml` file in the server `plugin` directory, which is at the following location:

      hyperic_home`/server_home/hq-engine/server/default/deploy/hq.ear/hq-plugins`

   b. Place the `extremescale-plugin.xml` file in the agent `plugin` directory, which is at the following location:

      agent_home`/bundles/gent-4.0.2-939/pdk/plugins`

   c. Put the `wshyperic.jar` file in the agent `lib` directory, which is at the following location

      agent_home`/bundles/gent-4.0.2-939/pdk/lib`
3. Configure the agent. The `agent.properties` file serves as a configuration point for the agent runtime. This property is in the *agent_home*`/conf` directory. The following keys are optional, but of importance to the eXtreme Scale plug-in:
   -
     ```
     autoinventory.defaultScan.interval.millis=<time_in_milliseconds>
     ```

Sets the interval in milliseconds between Agent discoveries.

- `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

  : Enables verbose debug statements from the eXtreme Scale plug-in.
- `username=<`*`username`*`>`: Sets the Java Management Extensions (JMX) user name if security is enabled.
- `password=<`*`password`*`>`: Sets the JMX password if security is enabled.
- `sslEnabled=<`*`true|false`*`>`: Tells the plug-in whether or not to use Secure Sockets Layer (SSL). The value is `false` by default.
- `trustPath=<`*`path`*`>`: Sets the trust path for the SSL connection.
- `trustType=<`*`type`*`>`: Sets the trust type for the SSL connection.
- `trustPass=<`*`password`*`>`: Sets the trust password for the SSL connection.

4. Start the agent discovery. The Hyperic agents send discoveries and metrics information to the server. Use the server to customize data views and group logical inventory objects to generate useful information. After the server is available, you must run the launch script or start the Windows service for the agent:
   - Linux *`agent_home`*`/bin/hq-agent.sh start`
   - Windows Start the agent with the Windows service.

   After you start the agents, the servers are detected and groups are configured. You can log into the server console and choose which resources to add to the inventory database for the server. The server console is at the following URL by default: `http://<`*`server_host_name`*`>:7080/`
5. Statistics must be enabled for Hyperic to collect statistical data.

   Use the **SetStatsSpec** control action on the Hyperic console for eXtreme Scale. Navigate to the resource, then use the **Control Action** drop-down list on the **Control** tabbed page to specify a `SetStatsSpec` setting with `ALL=enabled` in the **Control Arguments** text box.

   Catalog servers are not detected by the filter set in the Hyperic console. See the information about the **`statsSpec`** property in Server properties file, which enable statistics as soon as the containers start. Various options for enabling statistics are described in "Monitoring server statistics with managed beans (MBeans)" on page 549 and Sample: **`xsadmin`** utility
6. Monitor servers with the Hyperic console. After the servers are added to the inventory model, their services are no longer needed.
   - **Dashboard view**: When you viewed the resource detection events, you logged into the main dashboard view. The dashboard view is a generic view that acts as a message center that you can customize. You can export graphs or inventory objects to this main dashboard.
   - **Resources view**: You can query and view the entire inventory model from this page. After the services have been added, you can see every eXtreme Scale server properly labeled and listed together under the servers section. You can click on the individual servers to see the basic metrics.
7. View the entire server inventory on the Resource View page. On this page, you can then select multiple ObjectGrid servers and group them together. After you group a set of resources, their common metrics can be graphed to show overlays and differences among group members. To display an overlay, select the metrics on the display of your Server Group. The metric then displays in the charting area. To display an overlay for all group members, click the

underlined metric name. You can export any of the charts, node views, and comparative overlays to the main dashboard with the **Tools** menu.

Monitoring eXtreme Scale information in DB2

When the JPALoader or JPAEntityLoader is used with DB2 as the back-end database, eXtreme Scale-specific information can be passed to DB2. You can view this information by a performance monitor tool such as DB2 Performance Expert to monitor the eXtreme Scale applications that are accessing the database.

Before you begin

See "Collecting trace" on page 619 for more information about the different methods for setting trace that you can use.

About this task

When the loader is configured to use DB2 as the back-end database, the following eXtreme Scale information can be passed to DB2 for monitoring purposes:

- **User:** Specifies the name of the user that authenticates to eXtreme Scale. When basic authentication is not used, the principals from the authentication are used.
- **Workstation Name:** Specifies the host name, IP of the eXtreme Scale container server.
- **Application Name:** Specifies the name of the ObjectGrid, Persistence Unit name (if set).
- **Accounting Information:** Specifies the thread ID, transaction type, transaction id, and the connection string.

Read about the DB2 Performance Expert to learn how to monitor database access.

Procedure

- To enable all eXtreme Scale client information, set the following trace strings:
  ```
  ObjectGridClientInfo*=event=enabled
  ```
- To enable all but user information, use one of the following settings:

  -
    ```
    ObjectGridClientInfo*=event=enabled,ObjectGridClientInfoUser=event=disabled
    ```
    or

  -
    ```
    ObjectGridClientInfo=event=enabled
    ```

Results

After you turn on the trace function, data displays in the performance monitor tool such as DB2 Performance Expert.

Example

In the following example, user `bob` is authenticated as an eXtreme Scale user. The application is accessing the `mygrid` data grid using the `DB2Hibernate` persistence unit. The container server is named `XS_Server1`. The resulting information follows:

- **User**=`bob`
- **Workstation Name**=`XS_Server1,192.168.1.101`
- **Application Name**=`mygrid,DB2Hibernate`

- **Accounting Information**=1, DEFAULT,FE7954BD-0126-4000-E000-2298094151DB,com.ibm.db2.jcc.t4.b@71787178

In the following example, user bob is authenticated using a WebSphere Application Server token. The application is accessing the mygrid data grid using the DB2OpenJPA persistence unit name. The container server is named XS_Server2. The resulting information follows:

- **User**

  ```
  =acme.principal.UserPrincipal[Bob],acme.principal.
  GroupPrincipal[admin]
  ```
- **Workstation Name**=XS_Server2,192.168.1.102
- **Application Name**=mygrid,DB2OpenJPA
- **Accounting Information**=188,DEFAULT,FE72BC63-0126-4000-E000-851C092A4E33,com.ibm.ws.rsadapter.jdbc.WSJccSQLJConnection@2b432b43

Related concepts:

"Loaders" on page 28
With a Loader plug-in, a data grid map can behave as a memory cache for data that is typically kept in a persistent store on either the same system or another system. Typically, a database or file system is used as the persistent store. A remote Java virtual machine (JVM) can also be used as the source of data, allowing hub-based caches to be built using eXtreme Scale. A loader has the logic for reading and writing data to and from a persistent store.

Plug-ins for communicating with databases
With a Loader plug-in, an ObjectGrid map can behave as a memory cache for data that is typically kept in a persistent store on either the same system or some other system. Typically, a database or file system is used as the persistent store. A remote Java virtual machine (JVM) can also be used as the source of data, allowing hub-based caches to be built using ObjectGrid. A loader has the logic for reading and writing data to and from a persistent store.

Chapter 9. Tuning performance

You can tune settings in your environment to increase the overall performance of your WebSphere eXtreme Scale environment.

Tuning operating systems and network settings

Network tuning can reduce Transmission Control Protocol (TCP) stack delay by changing connection settings and can improve throughput by changing TCP buffers.

Operating systems

The tuning settings might improve WebSphere eXtreme Scale performance. Tune according to your network and application load.

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
```

Jumbo frames

For Ethernet networks, enabling jumbo frames (frame size or Maximum Transmission Unit (MTU) of 9000 bytes) on all systems (hosts and switches) can provide a significant performance improvement, especially when the application uses large payload sizes. Check the operating instructions for the particular host and switches in your network for information about how to enable jumbo frames. The steps to configure jumbo frames are particular to each equipment type.

Note: Enabling jumbo frames on some hosts in the configuration and not others can cause the switch to become a bottleneck point. The switch must convert between frame sizes on different ports. Therefore, it is best to enable jumbo frames on all hosts in the configuration or none of the hosts in the configuration.

ORB properties

Object Request Broker (ORB) properties modify the transport behavior of the data grid. These properties can be set with an `orb.properties` file, as settings in the WebSphere Application Server administrative console, or as custom properties on the ORB in the WebSphere Application Server administrative console.

Deprecated: The Object Request Broker (ORB) is deprecated. If you were not using the ORB in a previous release, use IBM eXtremeIO (XIO) for your transport mechanism. If you are using the ORB, consider migrating your configuration to use XIO.

orb.properties

The `orb.properties` file is in the `java/jre/lib` directory. When you modify the `orb.properties` file in a WebSphere Application Server `java/jre/lib` directory, the ORB properties are updated on the node agent and any other Java virtual machines (JVM) that are using the Java runtime environment (JRE). If you do not want this behavior, use custom properties or the ORB settings WebSphere Application Server administrative console.

Default WebSphere Application Server settings

WebSphere Application Server has some properties defined on the ORB by default. These settings are on the application server container services and the deployment manger. These default settings override any settings that you create in the `orb.properties` file. For each described property, see the **Where to specify** section to determine the location to define the suggested value.

File descriptor settings

For UNIX and Linux systems, a limit exists for the number of open files that are allowed per process. The operating system specifies the number of open files permitted. If this value is set too low, a memory allocation error occurs on AIX, and too many files opened are logged.

In the UNIX system terminal window, set this value higher than the default system value. For large SMP machines with clones, set to unlimited.

For AIX configurations set this value to unlimited with the command: `ulimit -n unlimited`.

For Solaris configurations set this value to `16384` with the command: `ulimit -n 16384`.

To display the current value use the command: `ulimit –a`.

Baseline settings

The following settings are a good baseline but not necessarily the best settings for every environment. Understand the settings to help make a good decision on what values are appropriate in your environment.

```
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
com.ibm.CORBA.SocketWriteTimeout=30
```

Property descriptions

Timeout Settings

The following settings relate to the amount of time that the ORB waits before giving up on request operations. Use these settings to prevent excess threads from being created in an abnormal situation.

Request timeout

Property name: com.ibm.CORBA.RequestTimeout

Valid value: Integer value for number of seconds.

Suggested value: 30

Where to specify: WebSphere Application Server administrative console

Description: Indicates how many seconds any request waits for a response before giving up. This property influences the amount of time a client takes to fail over if a network outage failure occurs. If you set this property too low, requests might time out inadvertently. Carefully consider the value of this property to prevent inadvertent timeouts.

Connect timeout

Property name: com.ibm.CORBA.ConnectTimeout

Valid value: Integer value for number of seconds.

Suggested value: 10

Where to specify: `orb.properties` file

Description: Indicates how many seconds a socket connection attempt waits before giving up. This property, like the request timeout, can influence the time a client takes to fail over if a network outage failure occurs. In general, set this property to a smaller value than the request timeout value because the amount of time to establish connections is relatively constant.

Fragment timeout

Property name: com.ibm.CORBA.FragmentTimeout

Valid value: Integer value for number of seconds.

Suggested value: 30

Where to specify: `orb.properties` file

Description: Indicates how many seconds a fragment request waits before giving up. This property is similar to the request timeout property.

Socket write timeout

Property name: com.ibm.CORBA.SocketWriteTimeout

Valid value: Integer value for number of seconds.

Suggested value: 30

Where to specify: `orb.properties` file

Description: Indicates how many seconds a socket write waits before giving up. This property is similar to the request timeout property.

Thread Pool Settings

These properties constrain the thread pool size to a specific number of threads. The threads are used by the ORB to spin off the server requests after they are received on the socket. Setting these property values too low results in an increased socket queue depth and possibly timeouts.

Connection multiplicity

Property name: com.ibm.CORBA.ConnectionMultiplicity

Valid value: Integer value for the number of connections between the client and server. The default value is 1. Setting a larger value sets multiplexing across multiple connections.

Suggested value: 1

Where to specify: `orb.properties` file**Description:** Enables the ORB to use multiple connections to any server. In theory, setting this value promotes parallelism over the connections. In practice, performance does not benefit from setting the connection multiplicity. Do not set this parameter.

Open connections

Property names: com.ibm.CORBA.MinOpenConnections, com.ibm.CORBA.MaxOpenConnections

Valid value: An integer value for the number of connections.

Suggested value: 1024

Where to specify: WebSphere Application Server administrative console**Description:** Specifies a minimum and maximum number of open connections. The ORB keeps a cache of connections that have been established with clients. These connections are purged when this value is passed. Purging connections might cause poor behavior in the data grid.

Is Growable

Property name: com.ibm.CORBA.ThreadPool.IsGrowable

Valid value: Boolean; set to `true` or `false`.

Suggested value: `false`

Where to specify: `orb.properties` file**Description:** If set to `true`, the thread pool that the ORB uses for incoming requests can grow beyond what the pool supports. If the pool size is exceeded, new threads are created to handle the request but the threads are not pooled. Prevent thread pool growth by setting the value to `false`.

Server socket queue depth

Property name: com.ibm.CORBA.ServerSocketQueueDepth

Valid value: An integer value for the number of connections.

Suggested value: `1024`

Where to specify: `orb.properties` file**Description:** Specifies the length of the queue for incoming connections from clients. The ORB queues incoming connections from clients. If the queue is full, then connections are refused. Refusing connections might cause poor behavior in the data grid.

Fragment size

Property name: com.ibm.CORBA.FragmentSize

Valid value: An integer number that specifies the number of bytes. The default is 1024.

Suggested value: `0`

Where to specify: `orb.properties` file**Description:** Specifies the maximum packet size that the ORB uses when sending a request. If a request is larger than the fragment size limit, then that request is divided into request fragments that are each sent separately and reassembled on the server. Fragmenting requests is helpful on unreliable networks where packets might need to be resent. However, if the network is reliable, dividing the requests into fragments might cause unnecessary processing.

No local copies

Property name: com.ibm.CORBA.iiop.NoLocalCopies

Valid value: Boolean; set to `true` or `false`.

Suggested value: `true`

Where to specify: WebSphere Application Server administrative console, **Pass by reference** setting. **Description:** Specifies whether the ORB passes by reference. The ORB uses pass by value invocation by default. Pass by value invocation causes extra garbage and serialization costs to the path when an interface is started locally. By setting this value to true, the ORB uses a pass by reference method that is more efficient than pass by value invocation.

No Local Interceptors

Property name: com.ibm.CORBA.NoLocalInterceptors

Valid value: Boolean; set to `true` or `false`.

Suggested value: `true`

Where to specify: `orb.properties` file **Description:** Specifies whether the ORB starts request interceptors even when making local requests (intra-process). The interceptors that WebSphere eXtreme Scale uses are for security and route handling are not required if the request is handled within the process. Interceptors that go between processes are only required for Remote Procedure Call (RPC) operations. By setting the no local interceptors, you can avoid the extra processing that using local interceptors introduces.

Attention: If you are usingWebSphere eXtreme Scale security, set the com.ibm.CORBA.NoLocalInterceptors property value to `false`. The security infrastructure uses interceptors for authentication.

Related tasks:

"Configuring Object Request Brokers" on page 341
The Object Request Broker (ORB) is used by WebSphere eXtreme Scale to communicate over a TCP stack. Use the `orb.properties` file to pass the properties that are used by the ORB to modify the transport behavior of the data grid. No action is required to use the ORB provided by WebSphere eXtreme Scale or WebSphere Application Server for your WebSphere eXtreme Scale servers.

"Configuring the Object Request Broker with stand-alone WebSphere eXtreme Scale processes" on page 342
You can use WebSphere eXtreme Scale with applications that use the Object Request Broker (ORB) directly in environments that do not contain WebSphere Application Server or WebSphere Application Server Network Deployment.

"Configuring a custom Object Request Broker" on page 343
WebSphere eXtreme Scale uses the Object Request Broker (ORB) to enable communication among processes. No action is required to use the Object Request Broker (ORB) provided by WebSphere eXtreme Scale or WebSphere Application Server for your WebSphere eXtreme Scale servers. Little effort is required to use the same ORBs for your WebSphere eXtreme Scale clients. If instead you must use a custom ORB, the ORB supplied with the IBM SDK is a good choice, although you must configure the ORB. ORBs from other vendors can be used, also with configuration.

Tuning IBM eXtremeIO (XIO)

You can use XIO server properties to tune the behavior of the XIO transport in the data grid.

Server properties for tuning XIO

You can set the following properties in the server properties file:

maxXIONetworkThreads
Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Default:`50`

minXIONetworkThreads
Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Default:`50`

maxXIOWorkerThreads
Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Default:`128`

minXIOWorkerThreads
Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Default:`128`

xioChannel.xioContainerTCPNonSecure.Port

Deprecated: This property is deprecated. The value that is specified by the listenerPort property is used instead.
Specifies the non-secure listener port number of eXtremeIO on the server. If you do not set the value, an ephemeral port is used. This property is used only when the **`transportType`** property is set to `TCP/IP`.

xioChannel.xioContainerTCPSecure.Port

Deprecated: This property is deprecated. The value that is specified by the listenerPort property is used instead.
Specifies the SSL port number of eXtremeIO on the server. This property is used only when the **`transportType`** property is set to `SSL-Supported` or `SSL-Required`.

Tuning Java virtual machines

You must take into account several specific aspects of Java virtual machine (JVM) tuning for WebSphere eXtreme Scale best performance. In most cases, few or no special JVM settings are required. If many objects are being stored in the data grid, adjust the heap size to an appropriate level to avoid running out of memory.

IBM eXtremeMemory

7.1.1+ By configuring eXtremeMemory, you can store objects in native memory instead of on the Java heap. Configuring eXtremeMemory enables eXtremeIO, a new transport mechanism. By moving objects off the Java heap, you can avoid garbage collection pauses, leading to more constant performance and predicable response times. For more information, see "Configuring IBM eXtremeMemory" on page 346.

Tested platforms

Performance testing occurred primarily on AIX (32 way), Linux (four way), and Windows (eight way) computers. With high-end AIX computers, you can test heavily multi-threaded scenarios to identify and fix contention points.

Garbage collection

WebSphere eXtreme Scale creates temporary objects that are associated with each transaction, such as request and response, and log sequence. Because these objects affect garbage collection efficiency, tuning garbage collection is critical.

All modern JVMs use parallel garbage collection algorithms, which means that using more cores can reduce pauses in garbage collection. A physical server with eight cores has a faster garbage collection than a physical with four cores.

When the application must manage a large amount of data for each partition, then garbage collection might be a factor. A read mostly scenario performs even with large heaps (20 GB or more) if a generational collector is used. However, after the tenure heap fills, a pause proportional to the live heap size and the number of processors on the computer occurs. This pause can be large on smaller computers with large heaps.

IBM virtual machine for Java garbage collection

For the IBM virtual machine for Java, use the **optavgpause** collector for high update rate scenarios (100% of transactions modify entries). The **gencon** collector works much better than the **optavgpause** collector for scenarios where data is updated relatively infrequently (10% of the time or less). Experiment with both collectors to see what works best in your scenario. Run with verbose garbage collection turned on to check the percentage of the time that is being spent collecting garbage. Scenarios have occurred where 80% of the time is spent in garbage collection until tuning fixed the problem.

Use the **-Xgcpolicy** parameter to change the garbage collection mechanism. The value of the **-Xgcpolicy** parameter can be set to: **-Xgcpolicy:gencon** or **-Xgcpolicy:optavgpause**, depending on which garbage collector you want to use.

- In a WebSphere Application Server configuration, set the **-Xgcpolicy** parameter in the administrative console. Click **Servers** > **Application servers** > **server_name** > **Process definition** > **Java Virtual Machine**. Add the parameter in the **Generic JVM arguments** field.
- In a stand-alone configuration, pass the **-jvmArgs** parameter to the start server script to specify the garbage collector. The **-jvmArgs** parameter must be the last parameter that is passed to the script.

Other garbage collection options

Attention: If you are using an Oracle JVM, adjustments to the default garbage collection and tuning policy might be necessary.

WebSphere eXtreme Scale supports WebSphere Real Time Java. With WebSphere Real Time Java, the transaction processing response for WebSphere eXtreme Scale is more consistent and predictable. As a result, the impact of garbage collection and thread scheduling is greatly minimized. The impact is reduced to the degree that the standard deviation of response time is less than 10% of regular Java.

JVM performance

WebSphere eXtreme Scale can run on different versions of Java Platform, Standard Edition. WebSphere eXtreme Scale supports Java SE Version 5 or later . For improved developer productivity and performance, use Java SE Version 5 or later 7.1.1.1+, or Java SE Version 7 to take advantage of annotations and improved garbage collection. WebSphere eXtreme Scale works on 32-bit or 64-bit Java virtual machines.

WebSphere eXtreme Scale is tested with a subset of the available virtual machines, however, the supported list is not exclusive. You can run WebSphere eXtreme Scale on any vendor JVM at Edition 5 or later. However, if a problem occurs with a vendor JVM, you must contact the JVM vendor for support. If possible, use the JVM from the WebSphere run time on any platform that WebSphere Application Server supports.

In general, use the latest available version of Java Platform, Standard Edition for the best performance.

Heap size

The recommendation is 1 to 2 GB heaps with a JVM per four cores. The optimum heap size number depends on the following factors:
- Number of live objects in the heap.
- Complexity of live objects in the heap.
- Number of available cores for the JVM.

For example, an application that stores 10 K byte arrays can run a much larger heap than an application that uses complex graphs of POJOs.

Note:

When running on Solaris, you must choose between a 32-bit or a 64-bit environment. If you do not specify either version, then the JVM runs as a 32-bit environment. If you are running WebSphere eXtreme Scale on Solaris and you encounter a heap size limit, then the -d64 option should be used to force the JVM to run in 64-bit mode which will support heap sizes greater than 3.5 GB.

Thread count

The thread count depends on a few factors. A limit exists for how many threads a single shard can manage. A shard is an instance of a partition, and can be a primary or a replica. With more shards for each JVM, you have more threads with each additional shard providing more concurrent paths to the data. Each shard is as concurrent as possible although there is a limit to the concurrency.

Object Request Broker (ORB) requirements

Deprecated: The Object Request Broker (ORB) is deprecated. If you were not using the ORB in a previous release, use IBM eXtremeIO (XIO) for your transport mechanism. If you are using the ORB, consider migrating your configuration to use XIO.

The IBM SDK includes an IBM ORB implementation that has been tested with WebSphere Application Server and WebSphere eXtreme Scale. To ease the support process, use an IBM-provided JVM. Other JVM implementations use a different ORB. The IBM ORB is only supplied with IBM-provided Java virtual machines. WebSphere eXtreme Scale requires a working ORB to operate. You can use WebSphere eXtreme Scale with ORBs from other vendors. However, if you have a problem with a vendor ORB, you must contact the ORB vendor for support. The IBM ORB implementation is compatible with third partyJava virtual machines and can be substituted if needed.

orb.properties tuning

In the lab, the following file was used on data grids of up to 1500 JVMs. The `orb.properties` file is in the `lib` folder of the runtime environment.

```
# 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
```

Related reference:

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

Tuning the IBM virtual machine for Java

Tuning the heartbeat interval setting for failover detection

You can configure the amount of time between system checks for failed servers with the heartbeat interval setting. This setting applies to catalog servers only.

About this task

Configuring failover varies depending on the type of environment you are using. If you are using a stand-alone environment, you can configure failover with the command line. If you are using a WebSphere Application Server Network Deployment environment, you must configure failover in the WebSphere Application Server Network Deployment administrative console.

Procedure

- Configure failover for stand-alone environments.
  - With the **-heartbeat** parameter in the **startOgServer** script when you start the catalog server.
  - With the heartBeatFrequencyLevel property in the server properties file for the catalog server.

  Use one of the following values:

Table 33. Valid heartbeat values

Value	Action	Description
-1	Aggressive	Specifies an aggressive heartbeat level. With this value, failures are detected more quickly, but more processor and network resources are used. This level is more sensitive to missing heartbeats when the server is busy. Failovers are typically detected within 5 seconds.
0	Typical (default)	Specifies a heartbeat level at a typical rate. With this value, failover detection occurs at a reasonable rate without overusing resources. Failovers are typically detected within 30 seconds.
1	Relaxed	Specifies a relaxed heartbeat level. With this value, a decreased heartbeat frequency increases the time to detect failures, but also decreases processor and network use. Failovers are typically detected within 180 seconds.

An aggressive heartbeat interval can be useful when the processes and network are stable. If the network or processes are not optimally configured, heartbeats might be missed, which can result in a false failure detection.

- Configure failover for WebSphere Application Server environments.

  You can configure WebSphere Application Server Network Deployment

  7.1.1 Version 6.1 and later to allow WebSphere eXtreme Scale to fail over very quickly. The default failover time for hard failures is approximately 200 seconds. A hard failure is a physical computer or server crash, network cable disconnection or operating system error. Failures because of process crashes or soft failures typically fail over in less than one second. Failure detection for soft failures occurs when the network sockets from the dead process are closed automatically by the operating system for the server hosting the process.

  Core group heartbeat configuration

  WebSphere eXtreme Scale running in a WebSphere Application Server process inherits the failover characteristics from the core group settings of the application server. The following sections describe how to configure the core group heartbeat settings for different versions of WebSphere Application Server Network Deployment:

  – **Update the core group settings for WebSphere Application Server Network Deployment 7.1.1 Version 6.1 and 7.0:**

    Specify the heartbeat interval in seconds on WebSphere Application Server versions from Version 6.0 through Version 6.1.0.12 or in milliseconds starting with Version 6.1.0.13. You must also specify the number of missed heartbeats. This value indicates how many heartbeats can be missed before a peer Java virtual machine (JVM) is considered as failed. The hard failure detection time is approximately the product of the heartbeat interval and the number of missed heartbeats.

    These properties are specified using custom properties on the core group using the WebSphere administrative console. See Core group custom properties for configuration details. These properties must be specified for all core groups used by the application:

    - The heartbeat interval is specified using either the IBM_CS_FD_PERIOD_SEC custom property for seconds or the IBM_CS_FD_PERIOD_MILLIS custom property for milliseconds (requires Version 6.1.0.13 or later).
    - The number of missed heartbeats is specified using the IBM_CS_FD_CONSECUTIVE_MISSED custom property.

    The default value for the IBM_CS_FD_PERIOD_SEC property is 20 and for the IBM_CS_FD_CONSECUTIVE_MISSED property is 10. If the

IBM_CS_FD_PERIOD_MILLIS property is specified, then it overrides any of the set IBM_CS_FD_PERIOD_SEC custom properties. The values of these properties are positive integer values.

Use the following settings to achieve a 1500 ms failure detection time for WebSphere Application Server Network Deployment Version 6.x servers:

- Set `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment V6.1.0.13 and later)
- Set `IBM_CS_FD_CONSECUTIVE_MISSED = 2`

– **Update the core group settings for WebSphere Application Server Network Deployment Version 7.0**

WebSphere Application Server Network Deployment Version 7.0 provides two core group settings that can be adjusted to increase or decrease failover detection:

- **Heartbeat transmission period.** The default is 30000 milliseconds.
- **Heartbeat timeout period.** The default is 180000 milliseconds.

For more details on how change these settings, see the WebSphere Application Server Network Deployment Information center: Discovery and failure detection settings.

Use the following settings to achieve a 1500 ms failure detection time for WebSphere Application Server Network Deployment Version 7 servers:

- Set the heartbeat transmission period to 750 milliseconds.
- Set the heartbeat timeout period to 1500 milliseconds.

What to do next

When these settings are modified to provide short failover times, there are some system-tuning issues to be aware of. First, Java is not a real-time environment. It is possible for threads to be delayed if the JVM is experiencing long garbage collection times. Threads might also be delayed if the machine hosting the JVM is heavily loaded (due to the JVM itself or other processes running on the machine). If threads are delayed, heartbeats might not be sent on time. In the worst case, they might be delayed by the required failover time. If threads are delayed, false failure detections occur. The system must be tuned and sized to ensure that false failure detections do not happen in production. Adequate load testing is the best way to ensure this.

Note: The current version of eXtreme Scale supports WebSphere Real Time.

Related concepts:

"Installation topologies" on page 172
With WebSphere eXtreme Scale, you can create many installation topologies that include stand-alone servers, WebSphere Application Server, or both.

Catalog service
The catalog service controls placement of shards and discovers and monitors the health of container servers in the data grid. The catalog service hosts logic that should be idle and has little influence on scalability. It is built to service hundreds of container servers that become available simultaneously, and run services to manage the container servers.

Related reference:

Server properties file
The server properties file contains several properties that define different settings for your server, such as trace settings, logging, and security configuration. The server properties file is used by both catalog service and container servers in both

stand-alone servers and servers that are hosted in WebSphere Application Server.

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Tuning garbage collection with WebSphere Real Time

Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary. WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

Related tasks:

"Configuring the HTTP session manager for various application servers" on page 380
WebSphere eXtreme Scale is bundled with a session management implementation that overrides the default session manager for a web container. This implementation provides session replication, high availability, better scalability, and configuration options. You can enable the WebSphere eXtreme Scale session replication manager and generic embedded ObjectGrid container startup for various application servers such as Tomcat.

"Configuring HTTP session manager with WebSphere Portal" on page 377
You can persist HTTP sessions from WebSphere Portal into a data grid.

"Configuring the HTTP session manager with WebSphere Application Server" on page 365
While WebSphere Application Server provides session management function, the performance degrades as the number of requests increases. WebSphere eXtreme Scale comes bundled with a session management implementation that provides session replication, high availability, better scalability, and more robust configuration options.

"Configuring WebSphere eXtreme Scale with WebSphere Application Server" on page 297
You can run catalog service and container server processes in WebSphere Application Server. The process to configure these servers is different than a stand-alone configuration. The catalog service can automatically start in WebSphere Application Server servers or deployment managers. Container process start when an eXtreme Scale application is deployed and started in the WebSphere Application Server environment.

Related reference:

"**startOgServer** script" on page 466
The **startOgServer** script starts container and catalog servers . You can use a variety of parameters when you start your servers to enable trace, specify port numbers, and so on.

Related information:

Configure WebSphere Commerce to use WebSphere eXtreme Scale for dynamic cache to improve performance and scale

WebSphere Business Process Management and Connectivity integration

Tuning the IBM virtual machine for Java

WebSphere Real Time in a stand-alone environment

You can use WebSphere Real Time with WebSphere eXtreme Scale. By enabling WebSphere Real Time, you can get more predictable garbage collection along with a stable, consistent response time and throughput of transactions in a stand-alone eXtreme Scale environment.

Advantages of WebSphere Real Time

WebSphere eXtreme Scale creates many temporary objects that are associated with each transaction. These temporary objects deal with requests, responses, log sequences, and sessions. Without WebSphere Real Time, transaction response time can go up to hundreds of milliseconds. However, using WebSphere Real Time with WebSphere eXtreme Scale can increase the efficiency of garbage collection and reduce response time to 10% of the stand-alone configuration response time.

Enabling WebSphere Real Time

Install WebSphere Real Time and stand-alone WebSphere eXtreme Scale onto the computers on which you plan to run eXtreme Scale. Set the JAVA_HOME environment variable to point to a standard Java SE Runtime Environment (JRE).

Set the JAVA_HOME environment variable to point to the installed WebSphere Real Time. Then enable WebSphere Real Time as follows.

1. Edit the stand-alone installation `objectgridRoot/bin/setupCmdLine.sh | .bat` file by removing the comment from the following line.
   ```
   WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome
   -Xgc:targetUtilization=80"
   ```
2. Save the file.

Now you have enabled WebSphere Real Time. If you want to disable WebSphere Real Time, you can add the comment back to the same line.

Best practices

WebSphere Real Time allows eXtreme Scale transactions to have a more predictable response time. Results show that the deviation of an eXtreme Scale transaction's response time improves significantly with WebSphere Real Time compared to standard Java with its default garbage collector. Enabling WebSphere Real Time with eXtreme Scale is optimal if your application's stability and response time are essential.

The best practices described in this section explain how to make WebSphere eXtreme Scale more efficient through tuning and code practices depending on your expected load.

- Set right level of processor usage for your application and garbage collector.

  WebSphere Real Time provides capacity to control the processor usage so that garbage collection impact on your application is controlled and minimized. Use the `-Xgc:targetUtilization=NN` parameter to specify `NN` percentage of the processor that is used by your application in every 20 seconds. The default for WebSphere eXtreme Scale is 80%, but you can modify the script in `objectgridRoot/bin/setupCmdLine.sh` file to set different number such as `70`, which provides more processor capacity to the garbage collector. Deploy enough servers to maintain processor load under 80% for your applications.

- Set a larger size of heap memory.

  WebSphere Real Time uses more memory than regular Java, so plan your WebSphere eXtreme Scale with a large heap memory and set the heap size when you start catalog servers and containers with the `–jvmArgs –XmxNNNM` parameterin the **`ogStartServer`** command. For example, to you might use `–jvmArgs –Xmx500M` parameter to start catalog servers, and use appropriate memory size to start containers. You can set the memory size to 60-70% of your expected data size per JVM. If you do not set this value, a OutOfMemoryError error could result. Optionally, you also can use the `–jvmArgs –Xgc:noSynchronousGCOnOOM` parameterto prevent nondeterministic behavior when the JVM runs out of memory.
- Adjust threads for garbage collection.

  WebSphere eXtreme Scale creates a lot of temporary objects associated with each transaction and Remote Procedure Call (RPC) threads. Garbage collection has performance benefits if your computer has enough processor cycles. The default number of threads is 1. You can change the number of threads with the `–Xgcthreads n` argument. The suggested value of this argument is the number of cores that are available with consideration of the number of Java virtual machines per computer.
- Adjust the performance for short-running applications with WebSphere eXtreme Scale.

  WebSphere Real Time is tuned for long running applications. Usually you need to run WebSphere eXtreme Scale continuous transactions for two hours to get reliable performance data. You can use the `–Xquickstart` parameter to make your short-running applications perform better. This parameter tells just-in-time (JIT) compiler to use lower level of optimization.
- Minimize WebSphere eXtreme Scale client queue and WebSphere eXtreme Scale client relay.

  The main advantage of using WebSphere eXtreme Scale with WebSphere Real Time is to have highly reliable transaction response time, which usually has several times of order magnitude improvements on the deviation of transaction response time. Any queued client requests and client request relay through other software impacts the response time that is beyond the control of WebSphere Real Time and WebSphere eXtreme Scale. You should change your threads and sockets parameters to maintain steady and smooth load without any significant delay and decrease your queue depth.
- Write WebSphere eXtreme Scale applications to use WebSphere Real Time threading.

  Without modifying your application, you can get highly reliable WebSphere eXtreme Scale transaction response time with several order magnitude improvements on the deviation of response time. You can further exploit threading advantage of your transactional applications from regular Java thread to RealtimeThread which provides better control on thread priority and scheduling control.

  Your application currently includes the following code.

  ```
  public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
  ```

  You can optionally replace this code with the following.

  ```
  public class WXSCacheAppImpl extends RealtimeThread implements
  WXSCacheAppIF
  ```

WebSphere Real Time in WebSphere Application Server

You can use WebSphere® Real Time with eXtreme Scale in a WebSphere Application Server Network Deployment environment version 7.0. By enabling WebSphere Real Time, you can get more predictable garbage collection along with a stable, consistent response time and throughput of transactions.

Advantages

Using WebSphere eXtreme Scale with WebSphere Real Time increases consistency and predictability at a cost of performance throughput in comparison to the default garbage collection policy employed in the standard IBM Java™ SE Runtime Environment (JRE). The cost versus benefit proposition can vary based on several criteria. The following are some of the major criteria:

- Server capabilities - Available memory, CPU speed and size, network speed and use
- Server loads – Sustained CPU load, peak CPU load
- Java configuration – Heap sizes, target use, garbage-collection threads
- WebSphere eXtreme Scale copy mode configuration – byte array vs. POJO storage
- Application specifics – Thread usage, response requirements and tolerance, object size, and so on.

In addition to this metronome garbage collection policy available in WebSphere Real Time, there are optional garbage collection policies available in standard IBM Java™ SE Runtime Environment (JRE). These policies, optthruput (default), gencon, optavgpause and subpool are specifically designed to solve differing application requirements and environments. For more information on these policies, see "Tuning Java virtual machines" on page 571. Depending upon application and environment requirements, resources and restrictions, prototyping one or more of these garbage collection policies can ensure that you meet your requirements and determine an optimal policy.

Capabilities with WebSphere Application Server Network Deployment

1. The following are some supported versions.
   - WebSphere Application Server Network Deployment version 7.0.0.5 and above.
   - WebSphere Real Time V2 SR2 for Linux and above. See IBM WebSphere Real Time V2 for Linux for more information.
   - WebSphere eXtreme Scale version 7.0.0.0 and above.
   - Linux 32 and 64 bit operating systems.
2. WebSphere eXtreme Scale servers cannot be collocated with a WebSphere Application Server DMgr.
3. Real Time does not support DMgr.
4. Real Time does not support WebSphere Node Agents.

Enabling WebSphere Real Time

Install WebSphere Real Time and WebSphere eXtreme Scale onto the computers on which you plan to run eXtreme Scale. Update the WebSphere Real Time Java to SR2.

You can specify the JVM settings for each server through the WebSphere Application Server version 7.0 console as follows.

Choose **Servers** > **Server types** > **WebSphere application servers** > **<required installed server>**

On the resulting page, choose "Process definition."

On the next page, click Java Virtual Machine at the top of the column on the right. (Here you can set heap sizes, garbage collection and other flags for each server.)

Set the following flags in the "Generic JVM arguments" field:

```
-Xrealtime -Xgcpolicy:metronome  -Xnocompressedrefs -Xgc:targetUtilization=80
```

Apply and save changes.

To use Real Time in WebSphere Application Server 7.0 to with eXtreme Scale servers including the JVM flags above, you must create a JAVA_HOME environment variable.

Set JAVA_HOME as follows.
1. Expand "Environment".
2. Select "WebSphere variables".
3. Ensure that "All scopes" is checked under “Show scope".
4. Select the required server from the drop-down list. (Do not select DMgr or node agent servers.)
5. If the JAVA_HOME environment variable is not listed, select "New," and specify JAVA_HOME for the variable name. In the "Value" field, enter the fully qualified path name to Real Time.
6. Apply and then save your changes.

Best practices

For a set of best practices see the best practices section in “Tuning garbage collection with WebSphere Real Time” on page 577. There are some important modifications to note in this list of best practices for a stand-alone WebSphere eXtreme Scale environment when deploying into a WebSphere Application Server Network Deployment environment.

You must place any additional JVM command line parameters in the same location as the garbage collection policy parameters specified in the previous section.

An acceptable initial target for sustained processor loads is 50% with short duration peek loads hitting up to 75%. Beyond this, you must add additional capacity before you see measurable degradation in predictability and consistency. You can increase performance slightly if you can tolerate longer response times. Exceeding an 80% threshold often leads to significant degradation in consistency and predictability.

Tuning the dynamic cache provider

The WebSphere eXtreme Scale dynamic cache provider supports the following configuration parameters for performance tuning.

About this task

- **com.ibm.websphere.xs.dynacache.ignore_value_in_change_event**: When you register a change event listener with the dynamic cache provider and generate a ChangeEvent instance, there is overhead associated with deserializing the cache entry so the value can be put inside the ChangeEvent. Setting this optional parameter on the cache instance to `true` skips the deserialization of the cache entry when generating ChangeEvents. The value returned is either null for a remove operation or a byte array containing the serialized form of the object. InvalidationEvent instances carry a similar performance penalty, which you can avoid by setting com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent to `true`.
- **com.ibm.websphere.xs.dynacache.enable_compression**: By default, the eXtreme Scale dynamic cache provider compresses the cache entries in memory to increase cache density, which can save a significant amount of memory for applications like servlet caching. If you know that most of your cache data is not be compressible, consider setting this value to `false`.

Chapter 10. Security

WebSphere eXtreme Scale can secure data access, including allowing for integration with external security providers. Aspects of security include authentication, authorization, transport security, data grid security, local security, and JMX (MBean) security.

Data grid authentication

You can use the secure token manager plug-in to enable server-to-server authentication, which requires you to implement the SecureTokenManager interface.

The generateToken(Object) method takes an object protect, and then generates a token that cannot be understood by others. The verifyTokens(byte[]) method does the reverse process: it converts the token back to the original object.

A simple SecureTokenManager implementation uses a simple encoding algorithm, such as a XOR algorithm, to encode the object in serialized form and then use corresponding decoding algorithm to decode the token. This implementation is not secure and is easy to break.

WebSphere eXtreme Scale default implementation

WebSphere eXtreme Scale provides an immediately available implementation for this interface. This default implementation uses a key pair to sign and verify the signature, and uses a secret key to encrypt the content. Every server has a JCEKS type keystore to store the key pair, a private key and public key, and a secret key. The keystore has to be the JCEKS type to store secret keys. These keys are used to encrypt and sign or verify the secret string on the sending end. Also, the token is associated with an expiration time. On the receiving end, the data is verified, decrypted, and compared to the receiver secret string. Secure Sockets Layer (SSL) communication protocols are not required between a pair of servers for authentication because the private keys and public keys serve the same purpose. However, if server communication is not encrypted, the data can be stolen by looking at the communication. Because the token expires soon, the replay attack threat is minimized. This possibility is significantly decreased if all servers are deployed behind a firewall.

The disadvantage of this approach is that the WebSphere eXtreme Scale administrators have to generate keys and transport them to all servers, which can cause security breach during transportation.

Related tasks:

"Authenticating and authorizing clients" on page 586
You can enable security and credential authentication to authenticate clients. In addition, you can authorize administrative clients to access the data grid.

"Authenticating application clients" on page 586
Application client authentication consists of enabling client-server security and credential authentication, and configuring an authenticator and a system credential generator.

"Authorizing application clients" on page 589
Application client authorization consists of ObjectGrid permission classes, authorization mechanisms, a permission checking period, and access by creator only authorization.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Class ClientSecurityConfigurationFactory

Data grid security

Data grid security ensures that a joining server has the right credentials, so a malicious server cannot join the data grid. Data grid security uses a shared secret string mechanism.

All WebSphere eXtreme Scale servers, including catalog servers, agree on a shared secret string. When a server joins the data grid, it is challenged to present the secret string. If the secret string of the joining server matches the string in the president server or catalog server, the joining server is accepted. If the string does not match, the join request is rejected.

Sending a clear text secret is not secure. The WebSphere eXtreme Scale security infrastructure provides a secure token manager plug-in to allow the server to secure this secret before sending. You must decide how to implement the secure operation. WebSphere eXtreme Scale provides an out-of-the-box implementation, in which the secure operation is implemented to encrypt and sign the secret.

The secret string is set in the `server.properties` file. See Server properties file for more information about the authenticationSecret property.

SecureTokenManager plug-in

A secure token manager plug-in is represented by the com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager interface.

For more information about the SecureTokenManager plug-in, see SecureTokenManager API documentation.

The generateToken(Object) method takes an object, and then generates a token that cannot be understood by others. The verifyTokens(byte[]) method does the reverse process: the method converts the token back to the original object.

A simple SecureTokenManager implementation uses a simple encoding algorithm, such as an exclusive or (XOR) algorithm, to encode the object in serialized form and then use the corresponding decoding algorithm to decode the token. This implementation is not secure.

WebSphere eXtreme Scale provides an immediately available implementation for this interface.

The default implementation uses a key pair to sign and verify the signature, and uses a secret key to encrypt the content. Every server has a JCEKS type keystore to store the key pair, a private key and public key, and a secret key. The keystore has to be the JCEKS type to store secret keys.

These keys are used to encrypt and sign or verify the secret string on the sending end. Also, the token is associated with an expiration time. On the receiving end, the data is verified, decrypted, and compared to the receiver secret string. Secure Sockets Layer (SSL) communication protocols are not required between a pair of servers for authentication because the private keys and public keys serve the same purpose. However, if server communication is not encrypted, the data can be stolen by looking at the communication. Because the token expires soon, the replay attack threat is minimized. This possibility is significantly decreased if all servers are deployed behind a firewall.

The disadvantage of this approach is that the WebSphere eXtreme Scale administrators have to generate keys and transport them to all servers, which can cause security breach during transportation.

Sample scripts to create default secure token manager properties

As noted in the previous section, you can create a keystore that contains a key pair to sign and verify the signature and a secret key to encrypt the content.

For example, you can use the JDK 6 keytool command to create the keys as follows:

```
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
```

These two commands create a key pair "keypair1" and a secret key "seckey1". You can then configure the following in the server property file:

```
secureTokenKeyStore=key1.jck
secureTokenKeyStorePassword=key111
secureTokenKeyStoreType=JCEKS
secureTokenKeyPairAlias=keypair1
secureTokenKeyPairPassword=keypair1
secureTokenSecretKeyAlias=seckey1
secureTokenSecretKeyPassword=seckey1
secureTokenCipherAlgorithm=DES
secureTokenSignAlgorithm=RSA
```

Configuration

See Server properties for more information about the properties that you use to configure the secure token manager.

Related tasks:

"Authenticating and authorizing clients" on page 586
You can enable security and credential authentication to authenticate clients. In addition, you can authorize administrative clients to access the data grid.

"Authenticating application clients" on page 586
Application client authentication consists of enabling client-server security and credential authentication, and configuring an authenticator and a system credential generator.

"Authorizing application clients" on page 589
Application client authorization consists of ObjectGrid permission classes, authorization mechanisms, a permission checking period, and access by creator only authorization.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Class ClientSecurityConfigurationFactory

Authenticating and authorizing clients

You can enable security and credential authentication to authenticate clients. In addition, you can authorize administrative clients to access the data grid.

Related concepts:

"Data grid authentication" on page 583
You can use the secure token manager plug-in to enable server-to-server authentication, which requires you to implement the SecureTokenManager interface.

"Data grid security" on page 584
Data grid security ensures that a joining server has the right credentials, so a malicious server cannot join the data grid. Data grid security uses a shared secret string mechanism.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Class ClientSecurityConfigurationFactory

Authenticating application clients

Application client authentication consists of enabling client-server security and credential authentication, and configuring an authenticator and a system credential generator.

Procedure

- Enable client-server security

  You must enable security on both the client and server to successfully authenticate with the ObjectGrid.

  1. Enable client security.

     WebSphere eXtreme Scale provides a client property sample file, the `sampleClient.properties` file, in the *was_root*`/optionalLibraries/ObjectGrid/properties` directory for a WebSphere Application Server installation, or the `/ObjectGrid/properties` directory in a mixed-server installation. You can modify this template file with appropriate values. Set the **`securityEnabled`** property in the `objectgridClient.properties` file to `true`. The **`securityEnabled`** property indicates if security is enabled. When a client connects to a server, the value on the client and server side must be set both `true` or both `false`. For example, if the connected server security is enabled, the property value must be set to `true` on the client side for the client to connect to the server.

     When you set **`securityEnabled`** to true, then the following settings are available:

     `credentialAuthentication`
       Specifies the client credential authentication support. Use one of the following valid values:
       – `Never`: The client does not support credential authentication.

– `Supported`: The client supports credential authentication if the server also supports credential authentication. (Default)
– `Required`: The client requires credential authentication.

`authenticationRetryCount`
Specifies the number of times that authentication is tried if the credential is expired. If the value is set to 0, attempts to authenticate are not tried again.

Default: `3`

`credentialGeneratorClass`
Specifies the name of the class that implements the com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator interface. To specify this property, the credentialAuthentication property must be set to `Supported` or `Required`. This class is used to get credentials for clients.

Default: no value

`credentialGeneratorProps`
Specifies the properties for the CredentialGenerator implementation class. The properties are set to the object with the setProperties(String) method. To specify this property, the credentialAuthentication property must be set to `Supported` or `Required`. The credentialGeneratorprops value is used only if the value of the credentialGeneratorClass property is not null.

`transportType`
Specifies the client transport type. The possible values are:
– `TCP/IP`: Indicates that the client only supports TCP/IP connections.
– `SSL-Supported`: Indicates that the client supports both TCP/IP and Secure Sockets Layer (SSL) connections. (Default)
– `SSL-Required`: Indicates that the client requires SSL connections.

`protocol`
Indicates the type of security protocol to use for the client. Set this protocol value based on which security provider you use. If you indicate a value that is not valid, a security exception results that indicates that the protocol value is incorrect.

Valid values: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, and so on.

The com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration interface represents the `security.ogclient.props` file. You can use the com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory public API to create an instance of this interface with default values, or you can create an instance by passing the ObjectGrid client security property file. The `security.ogclient.props` file contains other properties. See the ClientSecurityConfiguration API Documentation and ClientSecurityConfigurationFactory API Documentation for more details.

2. Enable server security.

   To enable the security on the server side, you can set the **`securityEnabled`** property in the `security.xml` file to `true`. Use a security descriptor XML file to specify the data grid security configuration to isolate the grid-wide security configuration from the non-security configuration.

- Enable credential authentication.

  After the eXtreme Scale client retrieves the Credential object using the CredentialGenerator object, the Credential object is sent along with the client request to the eXtreme Scale server. The server authenticates the Credential object before processing the request. If the Credential object is authenticated successfully, a Subject object is returned to represent this Credential object. This Subject object is then used for authorizing the request.

Set the **credentialAuthentication** property on the client and server properties files to enable the credential authentication. For more information, see Client properties file and Server properties file.

The following table provides which authentication mechanism to use under different settings.

Table 34. Credential authentication under client and server settings

Client credential authentication	Server credential authentication	Result
No	Never	Disabled
No	Supported	Disabled
No	Required	Error case
Supported	Never	Disabled
Supported	Supported	Enabled
Supported	Required	Enabled
Required	Never	Error case
Required	Supported	Enabled
Required	Required	Enabled

- Configure an authenticator.

  The eXtreme Scale server uses the Authenticator plug-in to authenticate the Credential object. An implementation of the Authenticator interface gets the Credential object and then authenticates it to a user registry, for example, a Lightweight Directory Access Protocol (LDAP) server, and so on. eXtreme Scale does not provide a registry configuration. Connecting to a user registry and authenticating to it must be implemented in this plug-in.

  For example, one Authenticator implementation extracts the user ID and password from the credential, uses them to connect and validate to an LDAP server, and creates a Subject object as a result of the authentication. The implementation can use Java Authentication and Authorization Service (JAAS) login modules. A Subject object is returned as a result of authentication.

  You can configure the authenticator in the security descriptor XML file, as shown in the following example:

```
<?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>
```

  Use the **-clusterSecurityFile** option when starting a secure server to set the security XML file. See the Java SE security tutorial in the *Product Overview* for more information.

- Configure a system credential generator.

The system credential generator is used to represent a factory for the system credential. A system credential is similar to an administrator credential. You can configure the SystemCredentialGenerator element in the catalog security XML file, as shown in the following example:

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.
 builtins.UserPasswordCredentialGenerator">
         <property name="properties" type="java.lang.String" value="manager manager1"
      description="username password" />
 </systemCredentialGenerator>
```

For demonstration purposes, the user name and password are stored in clear text. Do not store the user name and password in clear text in a production environment.

WebSphere eXtreme Scale provides a default system credential generator, which uses the server credentials. If you do not explicitly specify the system credential generator, this default system credential generator is used.

Related concepts:

"Data grid authentication" on page 583
You can use the secure token manager plug-in to enable server-to-server authentication, which requires you to implement the SecureTokenManager interface.

"Data grid security" on page 584
Data grid security ensures that a joining server has the right credentials, so a malicious server cannot join the data grid. Data grid security uses a shared secret string mechanism.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Class ClientSecurityConfigurationFactory

Authorizing application clients

Application client authorization consists of ObjectGrid permission classes, authorization mechanisms, a permission checking period, and access by creator only authorization.

About this task

For eXtreme Scale, authorization is based on the Subject object and permissions. The product supports two kinds of authorization mechanisms: Java Authentication and Authorization Service (JAAS) and custom authorization.

There are four different types of permission classes as follows.

- The MapPermission class represents permissions to access the data in ObjectGrid maps.
- The ObjectGridPermission class represents permissions to access ObjectGrid.
- The ServerMapPermission class represents permissions to access ObjectGrid maps on the server side from a client.
- The AgentPermission class represents permissions to start an agent on the server side.

For more information on APIs and associated permissions, see the topic on client authorization programming in the *Programming Guide*.

Procedure

1. Set the permission checking period.

   eXtreme Scale supports caching the map permission checking results for performance reasons. Without this mechanism, when a method that is in the list of methods that for your particular permission class is called, the runtime calls the configured authorization mechanism to authorize access. With this permission checking period set, the authorization mechanism is called periodically based on the permission checking period. For a list of methods for each permission class, see the topic on client authorization programming in the *Programming Guide*.

   The permission authorization information is based on the Subject object. When a client tries to access the methods, the eXtreme Scale runtime looks up the cache based on the Subject object. If the object cannot be found in the cache, the runtime checks the permissions granted for this Subject object, and then stores the permissions in a cache.

   The permission checking period must be defined before the ObjectGrid is initialized. The permission checking period can be configured in two ways:

   You can use the ObjectGrid XML file to define an ObjectGrid and set the permission check period. In the following example, the permission check period is set to 45 seconds:

```
<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>
```

   If you want to create an ObjectGrid with APIs, call the following method to set the permission checking period. This method can be called only before the ObjectGrid instance is initialized. This method applies only to the local eXtreme Scale programming model when you instantiate the ObjectGrid instance directly.

```
/**
 * 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);
```

2. Configure access-by-creator-only authorization.

   Access by creator only authorization ensures that only the user (represented by the Principal objects associated with it) who inserts the entry into the ObjectGrid map can access (read, update, invalidate and remove) that entry.

   The existing ObjectGrid map authorization model is based on the access type but not data entries. In other words, a user has a particular type of access, such as read, write, insert, delete, or invalidate, to either all the data in the map or none of the data. However, eXtreme Scale does not authorize users for individual data entry. This feature offers a new way to authorize users to data entries.

In a scenario where different users access different sets of data, this model can be useful. When the user loads data from the persistent store into the ObjectGrid maps, the access can be authorized by the persistent store. In this case, there is no need to do another authorization in the ObjectGrid map layer. You need only ensure that the person who loads the data into the map can access it by enabling the access by creator only feature.

Creator only mode attribute values:

`disabled`
The access by creator only feature is disabled.

`complement`
The access by creator only feature is enabled to complement the map authorization. In other words, both map authorization and access by creator only feature takes effect. Therefore, you can further limit the operations to the data. For example, the creator cannot invalidate the data.

`supersede`
The access by creator only feature is enabled to supersede the map authorization. In other words, the access by creator only feature supersedes the map authorization; no map authorization occurs.

a. Configure the access-by-creator-only mode with an XML file.

   You can use the ObjectGrid XML file to define an ObjectGrid and set the access by creator only mode to either `disabled`, `complement`, or `supersede`, as shown in the following example:

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
         accessByCreatorOnlyMode="supersede"
        <bean id="TransactionCallback"
              classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
</objectGrids>
```

b. Configure the access-by-creator-only mode programmatically.

   If you want to create an ObjectGrid programmatically, you can call the following method to set the access by creator only mode. Calling this method applies only to the local eXtreme Scale programming model when you directly instantiate the ObjectGrid instance:

```
/**
 * 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);
```

   To further illustrate, consider a scenario in which an ObjectGrid map account is in a banking grid, and Manager1 and Employee1 are the two users. The eXtreme Scale authorization policy grants all access permissions to Manager1, but only read access permission to Employee1. The JAAS policy for the ObjectGrid map authorization is shown the following example:

```
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"
};
```

Remember: Consider how the access by creator only feature affects authorization:

- **`disabled`** If the access by creator only feature is disabled, the map authorization is no different. The user "Manager1" can access all the data in the "account" map. The user "Employee1" can read and insert all the data in the map but cannot update, invalidate, remove any data in the map.
- **`complement`** If the access by creator only feature is enabled with "complement" option, both the map authorization and access by creator only authorization will take effect. The user "Manager1" can access the data in the "account" map, but only if the user alone loaded them into the map. The user "Employee1" can read the data in the "account" map, but only if that user alone loaded them into the map. (However, this user cannot update, invalidate, or remove any data in the map.)
- **`supersede`** If the access by creator only feature is enabled with "supersede" option, the map authorization will not be enforced. The access by creator only authorization will be the only authorization policy. The user "Manager1" has the same privilege as in the "complement" mode: this user can access the data in the "account" map only if the same user loaded the data into the map. However, the user "Employee1" now has full access to the data in the "account" map if this user loaded them into the map. In other words, the authorization policy defined in the Java Authentication and Authorization Service (JAAS) policy will then not be enforced.

Related concepts:

"Data grid authentication" on page 583
You can use the secure token manager plug-in to enable server-to-server authentication, which requires you to implement the SecureTokenManager interface.

"Data grid security" on page 584
Data grid security ensures that a joining server has the right credentials, so a malicious server cannot join the data grid. Data grid security uses a shared secret string mechanism.

Related reference:

Client properties file
Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Class ClientSecurityConfigurationFactory

Configuring secure transport types

Transport layer security (TLS) provides secure communication between the client and server. The communication mechanism that is used depends on the value of the **`transportType`** parameter that is specified in the client and server configuration files.

About this task

When Secure Sockets Layer (SSL) is used, the SSL configuration parameters must be provided on both the client and server side. In a Java SE environment, the SSL configuration is configured in the client or server property files. If the client or server is in WebSphere Application Server, then you can use the existing WebSphere Application Server CSIV2 transport settings for your container servers and clients. See "Security integration with WebSphere Application Server" on page 601 for more information.

Table 35. Transport protocol to use under client transport and server transport settings.

If the transportType settings are different between the client and server, the resulting protocol can vary or result in an error.

Client transportType property	Server transportType property	Resulting protocol
TCP/IP	TCP/IP	TCP/IP
TCP/IP	SSL-supported	TCP/IP
TCP/IP	SSL-required	Error
SSL-supported	TCP/IP	TCP/IP
SSL-supported	SSL-supported	SSL (if SSL fails, then TCP/IP)
SSL-supported	SSL-required	SSL
SSL-required	TCP/IP	Error
SSL-required	SSL-supported	SSL
SSL-required	SSL-required	SSL

Procedure

1. To set the **`transportType`** property in the client security configuration, see Client properties file.
2. To set the **`transportType`** property in the container and catalog server security configuration, see Server properties file.

Transport layer security and secure sockets layer

WebSphere eXtreme Scale supports both TCP/IP and Transport Layer Security/Secure Sockets Layer (TLS/SSL) for secure communication between clients and servers.

TLS and SSL encryption in for clients and servers

TLS/SSL is sometimes enabled in one direction. For example, the server public certificate is imported in the client truststore, but the client public certificate is not imported into the server truststore. However, WebSphere eXtreme Scale extensively uses data grid agents. A characteristic of a data grid agent is when the server sends responses back to the client, it creates a new connection. The eXtreme Scale server then acts as a client. Therefore, you must import the client public certificate into the server truststore.

Transport layer security for Oracle JDK

WebSphere eXtreme Scale requires IBM Java Secure Sockets Extension (IBMJSSE) or the IBM Java Secure Sockets Extension 2 (IBMJSSE2). The IBMJSSE and IBMJSSE2 providers contain a reference implementation supporting SSL and Transport Layer Security (TLS) protocols and an application programming interface (API) framework.

The Oracle JDK does not ship the IBM JSSE and IBM JSSE2 providers, therefore transport security cannot be enabled with an Oracle JDK. In order to make this work, an Oracle JDK shipped with WebSphere Application Server is required. The WebSphere Application Server shipped Oracle JDK contains the IBM JSSE and IBM JSSE2 providers.

See "Configuring a custom Object Request Broker" on page 343 for information about using a non-IBM JDK for WebSphere eXtreme Scale. If `-Djava.endorsed.dirs` is configured, it points to both the `objectgridRoot/lib/endorsed` and the `JRE/lib/endorsed` directories. The directory `objectgridRoot/lib/endorsed` is required so the IBM ORB is used, and the directory `JRE/lib/endorsed` is required to load the IBM JSSE and IBM JSSE2 providers.

Work with step 4 of the security tutorial in the *Product Overview* for information about how to configure your required SSL properties, to create keystores and truststores, and to start secure servers in WebSphere eXtreme Scale.

Configuring Secure Sockets Layer (SSL) parameters for clients or servers

How you configure SSL parameters varies between clients and servers.

About this task

TLS/SSL is sometimes enabled in one direction. For example, the server public certificate is imported in the client truststore, but the client public certificate is not imported to the server truststore. However, WebSphere eXtreme Scale extensively uses data grid agents. A characteristic of a data grid agent is when the server sends responds back to the client, it creates a connection. The eXtreme Scale server then acts as a client. Therefore, you must import the client public certificate into the server truststore.

Procedure

- Configure client SSL parameters.

  Use one of the following options to configure SSL parameters on the client:
  - Create a com.ibm.websphere.objectgrid.security.config.SSLConfiguration object by using the com.ibm.websphere.objectgrid.security.config. ClientSecurityConfigurationFactory factory class.
  - Configure the parameters in the `client.properties` file. You can then either set the property file as a JVM client property or you can use the WebSphere eXtreme Scale APIs. Pass the properties file into the ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String) method for the client and use the returned object as a parameter to the ObjectGridManager.connect(String, ClientSecurityConfiguration, URL) method.
- Configure server SSL parameters.

  SSL parameters are configured for servers using the `server.properties` file. To start a container or catalog server with a specific property file, use the **`-serverProps`** parameter on the **`startOgServer`** script. For more information about the SSL parameters you can set for eXtreme Scale servers, see Security server properties.

Related reference:

Client properties file

Create a properties file based on your requirements for WebSphere eXtreme Scale client processes.

Java Management Extensions (JMX) security

You can secure managed beans (MBean) invocations in a distributed environment.

For more information about the MBeans that are available, see “Administering with Managed Beans (MBeans)” on page 503.

In the distributed deployment topology, MBeans are directly hosted in the catalog servers and container servers. In general, JMX security in a distributed topology follows the JMX security specification as specified in the Java Management Extensions (JMX) Specification. It consists of the following three parts:

1. Authentication: The remote client needs to be authenticated in the connector server.
2. Access control: MBean access control limits who can access the MBean information and who can perform the MBean operations.
3. Secure transport: The transport between the JMX client and server can be secured with TLS/SSL.

Authentication

JMX provides methods for the connector servers to authenticate the remote clients. For the RMI connector, authentication is completed by supplying an object that implements the JMXAuthenticator interface when the connector server is created. So eXtreme Scale implements this JMXAuthenticator interface to use the ObjectGrid Authenticator plug-in to authenticate the remote clients. See “Java SE security tutorial - Step 2” on page 92 for details on how eXtreme Scale authenticates a client.

The JMX client follows the JMX APIs to provide credentials to connect to the connector server. The JMX framework passes the credential to the connector server, and then calls the JMXAuthenticator implementation for authentication. As described previously, the JMXAuthenticator implementation then delegates the authentication to the ObjectGrid Authenticator implementation.

Review the following example that describes how to connect to a connector server with a credential:

```
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);
```

In the preceding example, a UserPasswordCredential object is provided with the user ID set to `admin` and the password set to `xxxxx`. This UserPasswordCredential object is set in the environment map, which is used in the JMXConnector.connect(Map) method. This UserPasswordCredential object is then passed to the server by the JMX framework, and finally passed to the ObjectGrid authentication framework for authentication.

The client programming model strictly follows the JMX specification.

Access control

A JMX MBean server might have access to sensitive information and might be able to perform sensitive operations. JMX provides necessary access control that identifies which clients can access that information and who can perform those operations. The access control is built on the standard Java security model by defining permissions that control access to the MBean server and its operations.

For JMX operation access control or authorization, eXtreme Scale relies on the JAAS support provided by the JMX implementation. At any point in the execution of a program, there is a current set of permissions that a thread of execution holds. When such a thread calls a JMX specification operation, these permissions are known as the held permissions. When a JMX operation is performed, a security check is done to check whether the needed permission is implied by the held permission.

The MBean policy definition follows the Java policy format. For example, the following policy grants all signers and all code bases with the right to retrieve the server JMX address for the PlacementServiceMBean. However, the signers and code bases are restricted to the com.ibm.websphere.objectgrid domain.

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
     [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

You can use the following policy example to complete authorization based on remote client identity. The policy grants the same MBean permission as shown in the preceding example, except only to users with X500Principal name as: `CN=Administrator,OU=software,O=IBM,L=Rochester,ST=MN,C=US`.

```
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 policies are checked only if the security manager is turned on. Start catalog servers and container servers with the -Djava.security.manager JVM argument to enforce the MBean operation access control.

Secure transport

The transport between the JMX client and server can be secured with TLS/SSL. If the transportType of catalog server or container server is set to SSL_Required or SSL_Supported, then you must use SSL to connect to the JMX server.

To use SSL, you need to configure the truststore, truststore type, and truststore password on the MBean client with -D system properties:

1. `-Djavax.net.ssl.trustStore=TRUST_STORE_LOCATION`
2. `-Djavax.net.ssl.trustStorePassword=TRUST_STORE_PASSWORD`
3. `-Djavax.net.ssl.trustStoreType=TRUST_STORE_TYPE`

If you use com.ibm.websphere.ssl.protocol.SSLSocketFactory as your SSL socket factory in your *java_home*`/jre/lib/security/java.security` file, then use the following properties:

1. `-Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION`

2. `-Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD`
3. `-Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE`

To obtain this information when Transport Layer Security/Secure Sockets Layer (TLS/SSL) is enabled in stand-alone configurations, you must start the catalog and container servers with the JMX service port set. Use one of the following methods to set the JMX service port:

- Use the **`-JMXServicePort`** option on the **`startOgServer`** script.
- If you are using an embedded server, call the setJMXServicePort method in the ServerProperties interface to set the JMX service port.

The default value for the JMX service port on catalog servers is `1099`. You must use a different port number for each JVM in your configuration. If you want to use JMX/RMI, explicitly specify the**`-JMXServicePort`** option and port number, even if you want to use the default port value.

Setting the JMX service port is required when you want to display container server information from the catalog server. For example, the port is required when you are using the **`xscmd -c showMapSizes`** command.

Set the JMX connector port to avoid ephemeral port creation. Use one of the following methods to set the JMX connector port.

- Use the **`-JMXConnectorPort`** option on the **`startOgServer`** script.
- If you are using an embedded server, call the setJMVConnectorPort method in the ServerProperties interface.

Security integration with external providers

To protect your data, the product can integrate with several security providers.

WebSphere eXtreme Scale can integrate with an external security implementation. This external implementation must provide authentication and authorization services for WebSphere eXtreme Scale. WebSphere eXtreme Scale has plug-in points to integrate with a security implementation.WebSphere eXtreme Scale has been successfully integrated with the following components:

- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- ObjectGrid security
- Tivoli Access Manager
- Java Authentication and Authorization Service (JAAS)

eXtreme Scale uses the security provider for the following tasks:

- Authenticating clients to servers.
- Authorizing clients to access certain eXtreme Scale artifacts or to specify what can be done with eXtreme Scale artifacts.

eXtreme Scale has the following types of authorizations:

Map authorization
Clients or groups can be authorized to perform insert, read, update, evict or delete operations on maps.

ObjectGrid authorization
: Clients or groups can be authorized to perform object or entity queries on objectGrids.

DataGrid agent authorization
: Clients or groups can be authorized to allow DataGrid agents to be deployed to an ObjectGrid.

Server-side map authorization
: Clients or groups can be authorized to replicate a server map to client side or create a dynamic index to the server map.

Administration authorization
: Clients or groups can be authorized to perform administration tasks.

Note: If you had security already enabled for your back end , remember that these security settings are no longer sufficient to protect your data. Security settings from your database or other datastore does not in any way transfer to your cache. You must separately protect the data that is now cached using the eXtreme Scale security mechanism, including authentication, authorization, and transport level security.

Important: 7.1.1+ Use a Development Kit or Runtime Environment at Version 1.6 and later to support SSL Transport security with WebSphere eXtreme Scale Version 7.1.1 and later.

Securing the REST data service

Secure multiple aspects of the REST data service. Access to the eXtreme Scale REST data service can be secured through authentication and authorization. Access can also be controlled by service-scoped configuration rules, known as access rules. Transport security is the third consideration.

About this task

Access to the eXtreme Scale REST data service can be secured through authentication and authorization. Authentication and authorization is accomplished by integrating with eXtreme Scale security.

Access can also be controlled by service-scoped configuration rules, known as access rules Two types of access rules exist, service operation rights which control the CRUD operations that are allowed by the service and entity access rights which control the CRUD operations that are allowed for a particular entity type.

Transport security is provided by the hosting container configuration for connections between the web client and the REST service. And transport security is provided by eXtreme Scale client configuration (for REST service to eXtreme Scale data grid connections).

Procedure

- Control authentication and authorization.

  Access to the eXtreme Scale REST data service can be secured through authentication and authorization. Authentication and authorization are accomplished by integrating with eXtreme Scale security.

  The eXtreme Scale REST data service uses eXtreme Scale security, for authentication and authorization, to control which users can access the service and the operations a user is allowed to perform through the service. The eXtreme Scale REST data service uses either a configured global credential, with user and password, or a credential derived from an HTTP BASIC challenge that is sent with each transaction to the eXtreme Scale data grid where authentication and authorization is performed.

1. Configure eXtreme Scale client authentication and authorization on the grid See “Security integration with external providers” on page 597 for details about how to configure eXtreme Scale client authentication and authorization.
2. Configure the eXtreme Scale client, which is used by the REST service, for security.

   The eXtreme Scale REST data service invokes the eXtreme Scale client library when communicating with the eXtreme Scale grid. Therefore, the eXtreme Scale client must be configured for eXtreme Scale security.

   eXtreme Scale client authentication is enabled via properties in objectgrid client properties file. At a minimum, the following attributes must be enabled when using client security with the REST service:

   ```
   securityEnabled=true
   credentialAuthentication=Supported [-or-] Required
   credentialGeneratorProps=user:pass [-or-] {xor encoded user:pass}
   ```

   Remember: The user and password specified in the credentialGeneratorProps property must map to an ID in the authentication registry and have sufficient ObjectGrid policy rights to connect to and create ObjectGrids.

   A sample objectgrid client policy file is located in *restservice_home*`/security/security.ogclient.properties`. See also Client properties file.
3. Configure the eXtreme Scale REST data service for security.

   The eXtreme Scale REST data service configuration properties file needs to contain the following entries to integrate with eXtreme Scale security:

   `ogClientPropertyFile=`*`file_name`*

   The ogClientPropertyFile is the location of the propery file that contains ObjectGrid client properties mentioned in the preceding step. The REST service uses this file to initialize the eXtreme Scale client to talk to the grid when security is enabled.

   `loginType=basic [-or-] none`

   The loginType property configures the REST service for the login type. If a value of `none` is specified, the “global” user id and password defined by the credentialGeneratorProps will be sent to the grid for each transaction. If a value of `basic` is specified, the REST service will present an HTTP BASIC challenge to the client asking for credentials that it will send in each transaction when communicating with the grid.

   For more information about the ogClientPropertyFile and loginType properties, refer to REST data service properties file.

- Apply access rules.

  Access can also be controlled by service scoped configuration rules, known as access rules Two types of access rules exist, service operation rights which control the CRUD operations that are allowed by the service and entity access rights which control the CRUD operations that are allowed for a particular entity type.

  The eXtreme Scale REST data service optionally allows access rules that can be configured to restrict access to the service and entities in the service. These access rules are specified in the REST service access rights property file. The name of this file is specified in the REST data service properties file by the wxsRestAccessRightsFile property. For more information about this property, see REST data service properties file. This file is a typical Java property file with key and value pairs. Two types of access rules exist, service operation rights which control the CRUD operations that are allowed by the service and entity access rights which control the CRUD operations that are allowed for a particular entity type.

1. Configure service operation rights.

   Service Operations rights specify access rights that apply to all the ObjectGrids exposed via the REST service or to all entities of an individual ObjectGrid as specified.

   Use the following syntax.

   ```
   serviceOperationRights=service_operation_right
   serviceOperationRights.grid_name -OR- *=service_operation_right
   ```

   where

   – serviceOperationRights can be one of the following [NONE, READSINGLE, READMULTIPLE, ALLREAD, ALL]

   – serviceOperationRights.*grid_name -OR-* * implies that the access right applies to all the ObjectGrids, else name of a specific ObjectGrid can be provided.

   For example:

   ```
   serviceOperationsRights=ALL
   serviceOperationsRights.*=NONE
   serviceOperationsRights.EMPLOYEEGRID=READSINGLE
   ```

   The first example specifies that all service operations are allowed for all the ObjectGrids exposed by this REST Service. The second example is similar to the first example as it also applies to all the ObjectGrids exposed by the REST service, however it specifies the access right as NONE, which means none of the service operations are allowed on the ObjectGrids. The last example specifies how to control the service operations for a specific grid, here only Reads which results in a single record are allowed for all entities of the EMPLOYEEGRID.

   The default assumed by the REST service is `serviceOperationsRights=ALL` which means that all operations are allowed for all the ObjectGrids exposed by this service. This is different from the Microsoft implementation, for which the default is `NONE`, so no operations are allowed on the REST Service.

   Important: The service operations rights are evaluated in the order they are specified in this file, so the last specified right will override the rights preceding it.

2. Configure entity access rights.

   Entity set rights specify access rights that apply to specific ObjectGrid entities exposed via the REST service. These rights provide a way to impose tighter and more finer-grained access control on individual ObjectGrid entities than compared to Service Operation rights.

   Use the following syntax.

   ```
   entitySetRights.grid_name.entity_name=entity_set_right
   ```

   where

   – *entity_set_right* can be one of the following rights.

Table 36. Entity access rights. Supported values.

Access right	Description
NONE	Denies all rights to access data
READSINGLE	Allows to read single data items
READMULTIPLE	Allows reading sets of data
ALLREAD	Allows reading single or multiple sets of data
WRITEAPPEND	Allows creating new data items in data sets
WRITEREPLACE	Allows replacing data
WRITEDELETE	Allows deleting data items from data sets
WRITEMERGE	Allows merging data
ALLWRITE	Allows to write (i.e. create, replace, merge or delete) data
ALL	Allows creating, reading, updating, and deleting data

   – *entity_name* is the name of a specific ObjectGrid within the REST service.
   – *grid_name* is the name of a specific entity within the specified ObjectGrid.

   Note: If both service operation rights and entity set rights are specified for a respective ObjectGrid and its entities, then the more restrictive of those rights will be enforced, as illustrated in the following examples. Note also that the entity set rights are evaluated in the order they are specified in the file. The last specified right will override the rights preceding it.

   Example 1: If serviceOperationsRights.NorthwindGrid=READSINGLE and entitySetRights.NorthwindGrid.Customer=ALL are specified. READSINGLE will be enforced for the Customer entity.

   Example 2: If serviceOperationsRights.NorthwindGrid=ALLREAD is specified and entitySetRights.NorthwindGrid.Customer=ALLWRITE is specified then only Reads will be allowed for all entities of NorthwindGrid. However for Customer its entity set rights will prevent any Reads (since it specified ALLWRITE) and hence effectively the Customer entity will have access right as NONE.

- Secure transports.

  Transport security is provided by the hosting container configuration for connections between the web client and REST service. Transport security is provided by the eXtreme Scale client configuration for connections between the REST service and the eXtreme Scale grid.

  1. Secure the connection from the client and REST service. Transport security for this connection is provided by the hosting container environment, not in eXtreme Scale.
  2. Secure the connection from the REST service and the eXtreme Scale grid. Transport security for this connection is configured in eXtreme Scale. See “Transport layer security and secure sockets layer” on page 593.

Security integration with WebSphere Application Server

When WebSphere eXtreme Scale is deployed in a WebSphere Application Server environment, you can simplify the authentication flow and transport layer security configuration from WebSphere Application Server.

Simplified authentication flow

When eXtreme Scale clients and servers are running in WebSphere Application Server and in the same security domain, you can use the WebSphere Application Server security infrastructure to propagate the client authentication credentials to the eXtreme Scale server. For example, if a servlet acts as an eXtreme Scale client to connect to an eXtreme Scale server in the same security domain, and the servlet is already authenticated, it is possible to propagate the authentication token from the client (servlet) to the server, and then use the WebSphere Application Server security infrastructure to convert the authentication token back to the client credentials.

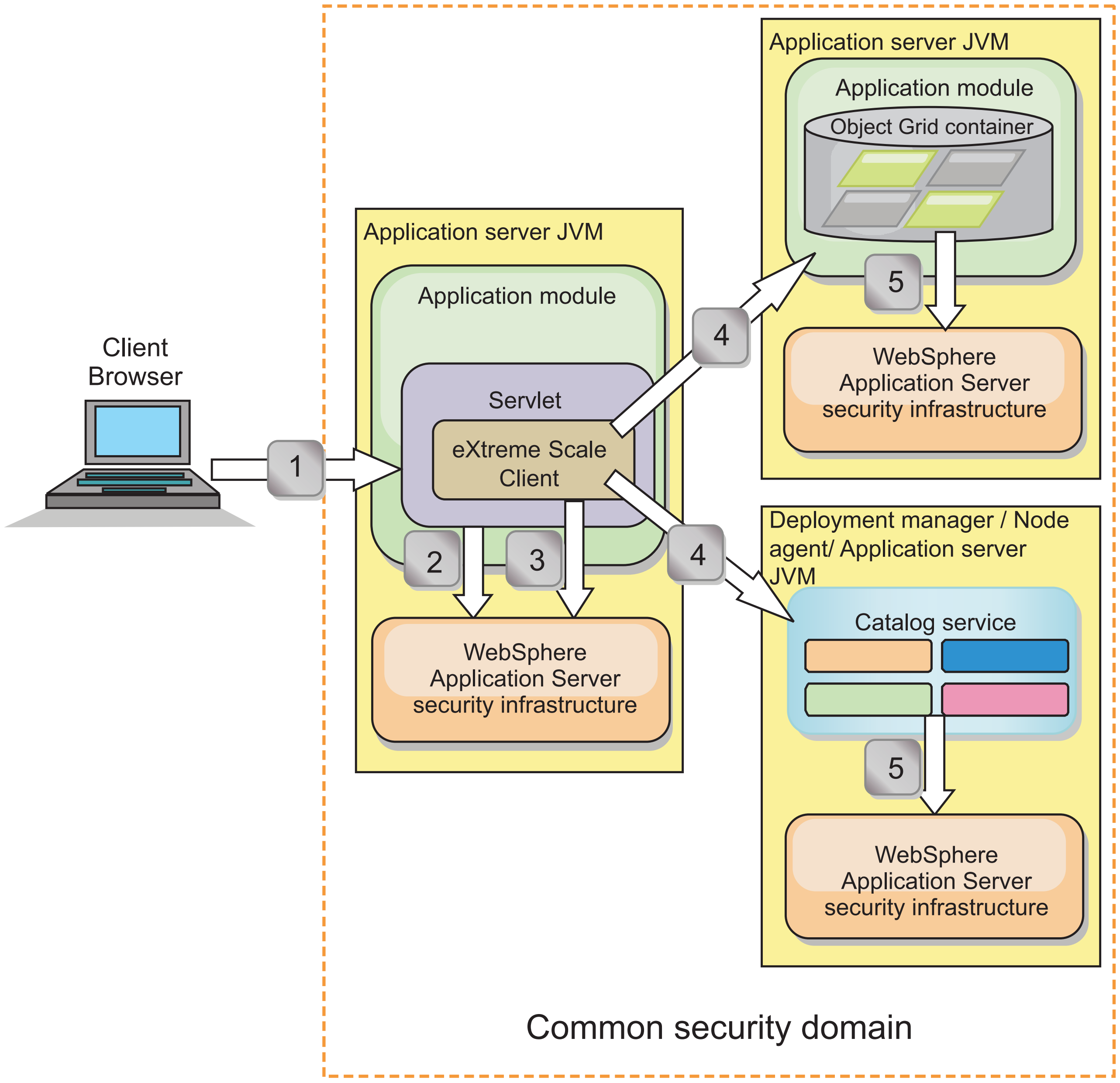

Client Browser
1
Application server JVM
Application module
Servlet
eXtreme Scale Client
2
3
WebSphere Application Server security infrastructure
4
4
Application server JVM
Application module
Object Grid container
5
WebSphere Application Server security infrastructure
Deployment manager / Node agent/ Application server JVM
Catalog service
5
WebSphere Application Server security infrastructure
Common security domain

Figure 68. Authentication flow for servers within the same security domain

In the previous diagram, the application servers are in the same security domain. One application server hosts the web application, which is also an eXtreme Scale client. The other application server hosts the container server. The deployment manager or node agent Java virtual machine (JVM) hosts the catalog service.

Note: Use this type of configuration in development environments. However, for production environments run the catalog servers in separate processes, and if possible, run catalog servers on a different system from where the container servers are running.
The arrows in the diagram indicate how the authentication process flows:

1. An enterprise application user uses a Web browser to log in to the first application server with a user name and password.
2. The first application server sends the client user name and password to the WebSphere Application Server security infrastructure to authenticate with the user registry. For example, this user registry might be an LDAP server. As a result, the security information is stored in the application server thread.
3. The JavaServer Pages (JSP) file acts as an eXtreme Scale client to retrieve the security information from the server thread. The JSP file calls the WebSphere Application Server security infrastructure to get the security tokens that represent the enterprise application user.
4. The eXtreme Scale client, or JSP file, sends the security tokens with the request to the container server and catalog service that is hosted in the other JVMs. The catalog server and container server use the WebSphere Application Server security tokens as an eXtreme Scale client credential.
5. The catalog and container servers send the security tokens to the WebSphere Application Server security infrastructure to convert the security tokens into user security information. This user security information is represented by a Subject object, which contains the principals, public credentials, and private credentials. This conversion can occur because the application servers that are hosting the eXtreme Scale client, catalog server, and container server are sharing the same WebSphere Application Server Lightweight Third-Party Authentication (LTPA) tokens.

Authentication integration

Distributed security integration with WebSphere Application Server:

For the distributed model, use the following classes:

- com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator
- com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenAuthenticator
- com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredential

For examples on how to use these classes, see “Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server” on page 106.

On the server side, use the WSTokenAuthentication authenticator to authenticate the WSTokenCredential object.

Local security integration with WebSphere Application Server:

For the local ObjectGrid model, use the following classes:

- com.ibm.websphere.objectgrid.security.plugins.builtins. WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins. WSSubjectValidationImpl

For more information about these classes, see Local security programming. You can configure the WSSubjectSourceImpl class as the SubjectSource plug-in, and the WSSubjectValidationImpl class as the SubjectValidation plug-in.

Transport layer security support in WebSphere Application Server

When an eXtreme Scale client, container server, or catalog server is running in a WebSphere Application Server process, eXtreme Scale transport security is managed by the WebSphere Application Server CSIV2 transport settings. For the eXtreme Scale client or container server, you should not use eXtreme Scale client or server properties to configure the SSL settings. All the SSL settings should be specified in the WebSphere Application Server configuration.

However, the catalog server is a little different. The catalog server has its own proprietary transport paths which cannot be managed by the WebSphere Application Server CSIV2 transport settings. Therefore, the SSL properties still need to be configured in the server properties file for the catalog server. See "Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server" on page 106 for more information.

Configuring client security on a catalog service domain

By configuring client security on a catalog service domain, you can define default client authentication configuration properties. These properties are used when a client properties file is not located in the Java virtual machine (JVM) that is hosting the client or when the client does not programmatically specify security properties. If a client properties file exists, the properties that you specify in the console override the values in the file. You can override these properties by specifying a `splicer.properties` file with the com.ibm.websphere.xs.sessionFilterProps custom property or by splicing the application EAR file.

Before you begin

- You must know the CredentialGenerator implementation that you are using to authenticate clients with the remote data grid. You can use one of the implementations that are provided by WebSphere eXtreme Scale: UserPasswordCredentialGenerator or WSTokenCredentialGenerator.

  You can also use a custom implementation of the CredentialGenerator interface. The custom implementation must be in the class path of the runtime client and the server. If you are configuring an HTTP session scenario with WebSphere Application Server, you must put the implementation in the class path of the deployment manager and the class path of the application server in which the client is running.
- You must have a catalog service domain defined. See "Creating catalog service domains in WebSphere Application Server" on page 299 for more information.

About this task

You must configure client security on the catalog service domain when you have enabled credential authentication on the server side, by configuring one of the following scenarios:

- The server-side security policy has the **`credentialAuthentication`** property set to `Required`.

- The server-side security policy has the **credentialAuthentication** property set to Supported AND an **authorizationMechanism** has been specified in the ObjectGrid XML file.

In these scenarios, a credential must be passed from the client. The credential that is passed from the client is retrieved from the getCredential method on a class that implements the CredentialGenerator interface. In an HTTP session configuration scenario, the run time must know the CredentialGenerator implementation to use to generate a credential that is passed to a remote data grid. If you do not specify the CredentialGenerator implementation class to use, the remote data grid would reject requests from the client because the client cannot be authenticated.

Procedure

Define client security properties. In the WebSphere Application Server administrative console, click **System administration** > **WebSphere eXtreme Scale** > **Catalog service domains** > ***catalog_service_domain_name*** > **Client security properties**. Specify client security properties on the page and save your changes. See “Client security properties” on page 317 for a list of the properties you can set.

Results

The client security properties that you configured on the catalog service domain are used as default values. The values you specify override any properties that are defined in the client.properties files.

What to do next

Configure your applications to use WebSphere eXtreme Scale for session management. See “Configuring WebSphere Application Server HTTP session persistence to a data grid” on page 368 for more information.

Enabling data grid authorization

WebSphere eXtreme Scale provides several security endpoints to integrate custom mechanisms. In the local programming model, the main security function is authorization, and has no authentication support. You must authenticate independently from the already existing WebSphere Application Server authentication. However, you can use the provided plug-ins to obtain and validate Subject objects.

About this task

You can enable local security with the ObjectGrid XML descriptor file or programmatically.

Procedure

Enable local security with the ObjectGrid XML descriptor XML file.
The secure-objectgrid-definition.xml file that is used in the ObjectGridSample enterprise application sample is shown in the following example. Set the securityEnabled attribute to true to enable security.

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
        authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
</objectGrids>
```

What to do next

Start the container and catalog servers with security enabled.

Related reference:

Deployment policy descriptor XML file
To configure a deployment policy, use a deployment policy descriptor XML file.

Starting and stopping secure servers

Security is enabled by specifying security-specific configurations when you start and stop servers.

Starting secure servers in a stand-alone environment

To start secure stand-alone servers, you pass the proper configuration files by specifying parameters on the **startOgServer** or command.

Procedure

- Start secure container servers.

  Starting a secure container server requires the following security configuration file:

  – **Server property file:** The server property file configures the security properties specific to the server. Refer to the Server properties file for more details.

  Specify the location of this configuration file by providing the following argument to the **startOgServer** script:

  -serverProps
  Specifies the location of the server property file, which contains the server-specific security properties. The file name specified for this property is in plain file path format, such as `../security/server.properties`.

  Enter the following lines when you run the **startOgServer** command command:

  UNIX Linux

```
startOgServer.sh <arguments> -jvmargs -Djava.security.auth.login.config=jaas.config
-Djava.security.manager -Djava.security.policy="auth.policy" -Dobjectgrid.home=$OBJECTGRID_HOME
```

  Windows

```
startOgServer.bat <arguments> -jvmargs -Djava.security.auth.login.config=jaas.config
-Djava.security.manager -Djava.security.policy="auth.policy" -Dobjectgrid.home=%OBJECTGRID_HOME%
```

- Start secure catalog servers.

  To start a secure catalog service, you must have the following configuration files:

  – **Security descriptor XML file:** The security descriptor XML file describes the security properties common to all servers, including catalog servers and container servers. One property example is the authenticator configuration which represents the user registry and authentication mechanism.

  – **Server property file:** The server property file configures the security properties that are specific to the server.

  Specify the location of these configuration files by providing the following arguments to the **startOgServer** script:

  -clusterSecurityFile and -clusterSecurityUrl
  These arguments specify the location of the Security descriptor XML file.

Use the **`-clusterSecurityFile`** parameter to specify a local file, or the **`-clusterSecurityUrl`** parameter to specify the URL of the `objectGridSecurity.xml` file.

-serverProps
Specifies the location of the server property file, which contains the server-specific security properties. The file name specified for this property is in plain file path format, such as `c:/tmp/og/catalogserver.props`.

Starting secure servers in WebSphere Application Server

To start secure servers in WebSphere Application Server, you must specify the security configuration files in the generic Java virtual machine (JVM) arguments.

Procedure

- Associate WebSphere eXtreme Scale catalog servers with WebSphere application servers using the administrative console. In the administrative console, click **System Administration** > **WebSphere eXtreme Scale** > **Catalog Service Domains**.
- Associate WebSphere eXtreme Scale container servers with particular WebSphere application servers by deploying an enterprise archive (EAR) file that contains the required XML descriptors for the data grid. For more information about this procedure, see “Tutorial: Integrate WebSphere eXtreme Scale security with WebSphere Application Server” on page 106.
- Specify Java virtual machine (JVM) arguments that point to configuration files to make the catalog and container servers secure. In addition, specify `securityEnabled="true"` in the `objectgrid xml` file for each data grid. After you specify the JVM arguments and enable security in your data grids, you can start the servers or clusters that act as eXtreme Scale catalog servers or container servers.
- Start catalog and containers servers with the WebSphere Application Server administrative console, or use the WebSphere Application Server command line.

What to do next

“Stopping secure servers”

Stopping secure servers

Stopping secure catalog servers or container servers requires one security configuration file.

Procedure

- Stop a secure catalog server or container server in stand-alone deployments. In stand-alone environments, stop WebSphere eXtreme Scale catalog and container servers using the teardown function of the **`xscmd`** command, or using the **`stopXsServer`** or **`stopOgServer`** commands.
- Use the WebSphere Application Server administrative console to stop eXtreme Scale server that run with WebSphere Application Server.

Configuring security profiles for the `xscmd` utility

By creating a security profile, you can use saved security parameters to use the **`xscmd`** utility with secure environments.

Before you begin

For more information about setting up the **xscmd** utility, see "Administering with the **xscmd** utility" on page 482.

About this task

You can use the **-ssp** *profile_name* or **--saveSecProfile** *profile_name* parameter with the rest of your **xscmd** command. to save a security profile. The profile can contain settings for user names and passwords, credential generators, keystores, truststores, and transport types.

The **ProfileManagement** command group in the **xscmd** utility contains commands for managing your security profiles.

Procedure

- Save a security profile.

  To save a security profile, use the **-ssp** *profile_name* or **--saveSecProfile** *profile_name* parameter with the rest of your command. Adding this parameter to your command saves the following parameters:

  ```
  -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>
  ```

  Security profiles are saved in the *user_home*`\.xscmd\profiles\security\<`*profile_name*`>.properties` directory.

  Important: Do not include the `.properties` file name extension on the *profile_name* parameter. This extension is automatically added to the file name.

- Use a saved security profile.

  To use a saved security profile, add the **-sp** *profile_name* or **--securityProfile** *profile_name* parameter to the command you are running. Command example:
  `xscmd -c listHosts -cep myhost.mycompany.com -sp myprofile`

- List the commands in the **ProfileManagement** command group.

  Run the following command: **xscmd -lc ProfileManagement**.

- List the existing security profiles.

  Run the following command: **xscmd -c listProfiles -v**.

- Display the settings that are saved in a security profile.

  Run the following command: **xscmd -c showProfile -pn** *profile_name*.

- Remove an existing security profile.

  Run the following command: **xscmd -c RemoveProfile -pn** *profile_name*.

Related reference:

"**xsadmin** tool to **xscmd** tool migration" on page 249
In previous releases, the **xsadmin** tool was a sample command-line utility to monitor the state of the environment. The **xscmd** tool has been introduced as an officially supported administrative and monitoring command-line tool. If you were previously using the **xsadmin** tool, consider migrating your commands to the new **xscmd** tool.

Securing J2C client connections

About this task

Applications reference the connection factory, which establishes the connection to the remote data grid. Each connection factory hosts a single eXtreme Scale client connection that is reused for all application components.

Important: Since the eXtreme Scale client connection might include a near cache, it is important that applications do not share a connection. A connection factory must exist for a single application instance to avoid problems sharing objects between applications.

You can set the credential generator with the API or in the client properties file. In the client properties file, the securityEnabled and credentialGenerator properties are used.

Attention: In the following example, some lines of code are continued on the next line for publication purposes.

```
securityEnabled=true
credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins.
    UserPasswordCredentialGenerator
credentialGeneratorProps=operator XXXXXX
```

The credential generator and credential in the client properties file are used for the eXtreme Scale connect operation and the default J2C credentials. Therefore, the credentials that are specified with the API are used at J2C connect time for the J2C connection. However, if no credentials are specified at J2C connect time, then the credential generator in the client properties file is used.

Procedure

1. Set up secure access where the J2C connection represents the eXtreme Scale client. Use the ClientPropertiesResource connection factory property or the ClientPropertiesURL connection factory property to configure client authentication.

   If you are using WebSphere eXtreme Scale with WebSphere Application Server, then specify the client properties on the catalog service domain configuration. When the connection factory references the domain, it automatically uses this configuration.
2. Configure the client security properties to use the connection factory that references the appropriate credential generator object for eXtreme Scale. These properties are also compatible with eXtreme Scale server security. For example, use the WSTokenCredentialGenerator credential generator for WebSphere credentials when eXtreme Scale is installed with WebSphere Application Server. Alternatively, use the UserPasswordCredentialGenerator credential generator when you run the eXtreme Scale in a stand-alone environment. In the following example, credentials are passed programmatically using the API call instead of using the configuration in the client properties:

```
XSConnectionSpec spec = new XSConnectionSpec();
spec.setCredentialGenerator(new UserPasswordCredentialGenerator("operator", "xxxxxx"));
Connection conn = connectionFactory.getConnection(spec);
```

3. (Optional) Disable the near cache, if required.

   All J2C connections from a single connection factory share a single near cache. Grid entry permissions and map permissions are validated on the server, but not on the near cache. When an application uses multiple credentials to create J2C connections, and the configuration uses specific permissions for grid entries and maps for those credentials, then disable the near cache. Disable the near cache using the connection factory property, ObjectGridResource or ObjectGridURL. For more information about disabling the near cache, see "Configuring the near cache" on page 353.

4. (Optional) Set security policy settings, if required.

   If the J2EE application contains the embedded eXtreme Scale resource adapter archive (RAR) file configuration, you might be required to set additional security policy settings in the security policy file for the application. For example, these policies are required:

   ```
   permission com.ibm.websphere.security.WebSphereRuntimePermission "accessRuntimeClasses";
   permission java.lang.RuntimePermission "accessDeclaredMembers";
   permission javax.management.MBeanTrustPermission "register";
   permission java.lang.RuntimePermission "getClassLoader";
   ```

   Additionally, any property or resource files used by connection factories require file or other permissions, such as `permission java.io.FilePermission "filePath";`. For WebSphere Application Server, the policy file is `META-INF/was.policy`, and it is located in the J2EE EAR file.

Results

The client security properties that you configured on the catalog service domain are used as default values. The values that you specify override any properties that are defined in the `client.properties` files.

What to do next

Use eXtreme Scale data access APIs to develop client components that you want to use transactions.

Chapter 11. Troubleshooting

In addition to the logs and trace, messages, and release notes discussed in this section, you can use monitoring tools to figure out issues such as the location of data in the environment, the availability of servers in the data grid, and so on. If you are running in a WebSphere Application Server environment, you can use Performance Monitoring Infrastructure (PMI). If you are running in a stand-alone environment, you can use a vendor monitoring tool, such as CA Wily Introscope or Hyperic HQ. You can also use and customize the **`xscmd`** utility to display textual information about your environment.

Troubleshooting and support for WebSphere eXtreme Scale

To isolate and resolve problems with your IBM products, you can use the troubleshooting and support information. This information contains instructions for using the problem-determination resources that are provided with your IBM products, including WebSphere eXtreme Scale .

Techniques for troubleshooting problems

Troubleshooting is a systematic approach to solving a problem. The goal of troubleshooting is to determine why something does not work as expected and how to resolve the problem. Certain common techniques can help with the task of troubleshooting.

The first step in the troubleshooting process is to describe the problem completely. Problem descriptions help you and the IBM technical-support representative know where to start to find the cause of the problem. This step includes asking yourself basic questions:

- What are the symptoms of the problem?
- Where does the problem occur?
- When does the problem occur?
- Under which conditions does the problem occur?
- Can the problem be reproduced?

The answers to these questions typically lead to a good description of the problem, which can then lead you to a problem resolution.

What are the symptoms of the problem?

When starting to describe a problem, the most obvious question is “What is the problem?” This question might seem straightforward; however, you can break it down into several more-focused questions that create a more descriptive picture of the problem. These questions can include:

- Who, or what, is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, crash, performance degradation, or incorrect result?

Where does the problem occur?

Determining where the problem originates is not always easy, but it is one of the most important steps in resolving a problem. Many layers of technology can exist between the reporting and failing components. Networks, the data grid, and servers are only a few of the components to consider when you are investigating problems.

The following questions help you to focus on where the problem occurs to isolate the problem layer:

- Is the problem specific to one platform or operating system, or is it common across multiple platforms or operating systems?
- Is the current environment and configuration supported?
- Do all users have the problem?
- (For multi-site installations.) Do all sites have the problem?

If one layer reports the problem, the problem does not necessarily originate in that layer. Part of identifying where a problem originates is understanding the environment in which it exists. Take some time to completely describe the problem environment, including the operating system and version, all corresponding software and versions, and hardware information. Confirm that you are running within an environment that is a supported configuration; many problems can be traced back to incompatible levels of software that are not intended to run together or have not been fully tested together.

When does the problem occur?

Develop a detailed timeline of events leading up to a failure, especially for those cases that are one-time occurrences. You can most easily develop a timeline by working backward: Start at the time an error was reported (as precisely as possible, even down to the millisecond), and work backward through the available logs and information. Typically, you need to look only as far as the first suspicious event that you find in a diagnostic log.

To develop a detailed timeline of events, answer these questions:

- Does the problem happen only at a certain time of day or night?
- How often does the problem happen?
- What sequence of events leads up to the time that the problem is reported?
- Does the problem happen after an environment change, such as upgrading or installing software or hardware?

Responding to these types of questions can give you a frame of reference in which to investigate the problem.

Under which conditions does the problem occur?

Knowing which systems and applications are running at the time that a problem occurs is an important part of troubleshooting. These questions about your environment can help you to identify the root cause of the problem:

- Does the problem always occur when the same task is being performed?
- Does a certain sequence of events need to happen for the problem to occur?
- Do any other applications fail at the same time?

Answering these types of questions can help you explain the environment in which the problem occurs and correlate any dependencies. Remember that just because multiple problems might have occurred around the same time, the problems are not necessarily related.

Can the problem be reproduced?

From a troubleshooting standpoint, the ideal problem is one that can be reproduced. Typically, when a problem can be reproduced you have a larger set of tools or procedures at your disposal to help you investigate. Consequently, problems that you can reproduce are often easier to debug and solve.

However, problems that you can reproduce can have a disadvantage: If the problem is of significant business impact, you do not want it to recur. If possible, re-create the problem in a test or development environment, which typically offers you more flexibility and control during your investigation.

- Can the problem be re-created on a test system?
- Are multiple users or applications encountering the same type of problem?
- Can the problem be recreated by running a single command, a set of commands, or a particular application?

Searching knowledge bases

You can often find solutions to problems by searching IBM knowledge bases. You can optimize your results by using available resources, support tools, and search methods.

About this task

You can find useful information by searching the information center for WebSphere eXtreme Scale . However, sometimes you need to look beyond the information center to answer your questions or resolve problems.

Procedure

To search knowledge bases for information that you need, use one or more of the following approaches:

- Search for content by using the IBM Support Assistant (ISA).

  ISA is a no-charge software serviceability workbench that helps you answer questions and resolve problems with IBM software products. You can find instructions for downloading and installing ISA on the ISA website.
- Find the content that you need by using the IBM Support Portal.

  The IBM Support Portal is a unified, centralized view of all technical support tools and information for all IBM systems, software, and services. The IBM Support Portal lets you access the IBM electronic support portfolio from one place. You can tailor the pages to focus on the information and resources that you need for problem prevention and faster problem resolution. Familiarize yourself with the IBM Support Portal by viewing the demo videos (https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos) about this tool. These videos introduce you to the IBM Support Portal, explore troubleshooting and other resources, and demonstrate how you can tailor the page by moving, adding, and deleting portlets.
- Search for content about WebSphere eXtreme Scale by using one of the following additional technical resources:

  - WebSphere eXtreme Scale release notes
  - WebSphere eXtreme Scale Support website
  - WebSphere eXtreme Scale forum
- Search for content by using the IBM masthead search. You can use the IBM masthead search by typing your search string into the Search field at the top of any ibm.com® page.
- Search for content by using any external search engine, such as Google, Yahoo, or Bing. If you use an external search engine, your results are more likely to include information that is outside the ibm.com domain. However, sometimes you can find useful problem-solving information about IBM products in newsgroups, forums, and blogs that are not on ibm.com.

  Tip: Include "IBM" and the name of the product in your search if you are looking for information about an IBM product.

Getting fixes

A product fix might be available to resolve your problem.

Procedure

To find and install fixes:

1. Obtain the tools required to get the fix. Use the IBM Update Installer to install and apply various types of maintenance packages for WebSphere eXtreme Scale or WebSphere eXtreme Scale Client. Because the Update Installer undergoes regular maintenance, you must use the most current version of the tool.
2. Determine which fix you need. See the Recommended fixes for WebSphere eXtreme Scale to select the latest fix. When you select a fix, the download document for that fix opens.
3. Download the fix. In the download document, click the link for the latest fix in the "Download package" section.
4. Apply the fix. Follow the instructions in the "Installation Instructions" section of the download document.
5. Subscribe to receive weekly e-mail notifications about fixes and other IBM Support information.

Getting fixes from Fix Central

You can use Fix Central to find the fixes that are recommended by IBM Support for a variety of products, including WebSphere eXtreme Scale . With Fix Central, you can search, select, order, and download fixes for your system with a choice of delivery options. A WebSphere eXtreme Scale product fix might be available to resolve your problem.

Procedure

To find and install fixes:

1. Obtain the tools that are required to get the fix. If it is not installed, obtain your product update installer. You can download the installer from Fix Central. This site provides download, installation, and configuration instructions for the update installer.
2. Select as the product, and select one or more check boxes that are relevant to the problem that you want to resolve.
3. Identify and select the fix that is required.

4. Download the fix.
   a. Open the download document and follow the link in the "Download Package" section.
   b. When downloading the file, ensure that the name of the maintenance file is not changed. This change might be intentional, or it might be an inadvertent change that is caused by certain web browsers or download utilities.
5. Apply the fix.
   a. Follow the instructions in the "Installation Instructions" section of the download document.
   b. For more information, see the "Installing fixes with the Update Installer" topic in the product documentation.
6. Optional: Subscribe to receive weekly e-mail notifications about fixes and other IBM Support updates.

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users resolve problems with the product.

Before you begin

After trying to find your answer or solution by using other self-help options, such as release notes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM maintenance contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the "*Software Support Handbook*".

Procedure

To contact IBM Support about a problem:
1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM support topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
   - With IBM Support Assistant (ISA). For more information, see "IBM Support Assistant for WebSphere eXtreme Scale" on page 638.
   - Online through the IBM Support Portal: You can open, update, and view all of your service requests from the Service Request portlet on the Service Request page.
   - By phone: For the phone number to call in your region, see the Directory of worldwide contacts web page.

Results

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

Exchanging information with IBM

To diagnose or identify a problem, you might need to provide IBM Support with data and information from your system. In other cases, IBM Support might provide you with tools or utilities to use for problem determination.

Sending information to IBM Support

To reduce the time that is required to resolve your problem, you can send trace and diagnostic information to IBM Support.

Procedure

To submit diagnostic information to IBM Support:

1. Open a problem management record (PMR).
2. Collect the diagnostic data that you need. Diagnostic data helps reduce the time that it takes to resolve your PMR. You can collect the diagnostic data manually or automatically:
   - Collect the data manually.
   - Collect the data automatically.
3. Compress the files by using the `.zip` or `.tar` file format.
4. Transfer the files to IBM. You can use one of the following methods to transfer the files to IBM:
   - IBM Support Assistant
   - The Service Request tool
   - Standard data upload methods: FTP, HTTP
   - Secure data upload methods: FTPS, SFTP, HTTPS
   - E-mail

   If you are using a z/OS product and you use ServiceLink / IBMLink to submit PMRs, you can send diagnostic data to IBM Support in an e-mail or by using FTP.

   All of these data exchange methods are explained on the IBM Support website.

Receiving information from IBM Support

Occasionally an IBM technical-support representative might ask you to download diagnostic tools or other files. You can use FTP to download these files.

Before you begin

Ensure that your IBM technical-support representative provided you with the preferred server to use for downloading the files and the exact directory and file names to access.

Procedure

To download files from IBM Support:

1. Use FTP to connect to the site that your IBM technical-support representative provided and log in as `anonymous`. Use your e-mail address as the password.
2. Change to the appropriate directory:
   a. Change to the `/fromibm` directory.

```
cd fromibm
```

   b. Change to the directory that your IBM technical-support representative provided.

```
cd nameofdirectory
```

3. Enable binary mode for your session.

```
binary
```

4. Use the **get** command to download the file that your IBM technical-support representative specified.

```
get filename.extension
```

5. End your FTP session.

```
quit
```

Subscribing to Support updates

To stay informed of important information about the IBM products that you use, you can subscribe to updates.

About this task

By subscribing to receive updates about the product, you can receive important technical information and updates for specific IBM Support tools and resources. You can subscribe to updates by using one of two approaches:

Social media subscriptions
The following RSS feed is available for the product:

- RSS feed for WebSphere eXtreme Scale forum

For general information about RSS, including steps for getting started and a list of RSS-enabled IBM web pages, visit the IBM Software Support RSS feeds site.

My Notifications
With My Notifications, you can subscribe to Support updates for any IBM product. My Notifications replaces My Support, which is a similar tool that you might have used in the past. With My Notifications, you can specify that you want to receive daily or weekly e-mail announcements. You can specify what type of information you want to receive, such as publications, hints and tips, product flashes (also known as alerts), downloads, and drivers. My Notifications enables you to customize and categorize the products about which you want to be informed and the delivery methods that best suit your needs.

Procedure

To subscribe to Support updates:

1. Subscribe to the RSS feed for the WebSphere eXtreme Scale forum .
   a. On the subscription page, click the RSS feed icon.
   b. Select the option that you want to use to subscribe to the feed.
   c. Click **Subscribe**.
2. Subscribe to My Notifications by going to the IBM Support Portal and click **My Notifications** in the **Notifications** portlet.
3. Sign in using your IBM ID and password, and click **Submit**.
4. Identify what and how you want to receive updates.
   a. Click the **Subscribe** tab.

b. Select the appropriate software brand or type of hardware.

c. Select one or more products by name and click **Continue**.

d. Select your preferences for how to receive updates, whether by e-mail, online in a designated folder, or as an RSS or Atom feed.

e. Select the types of documentation updates that you want to receive, for example, new information about product downloads and discussion group comments.

f. Click **Submit**.

Results

Until you modify your RSS feeds and My Notifications preferences, you receive notifications of updates that you have requested. You can modify your preferences when needed; for example, if you stop using one product and begin using another product.

Related information

IBM Software Support RSS feeds

Subscribe to My Notifications support content updates

My Notifications for IBM technical support

My Notifications for IBM technical support overview

Enabling logging

You can use logs to monitor and troubleshoot your environment.

About this task

Logs are saved different locations and formats depending on your configuration.

Procedure

- **Enable logs in a stand-alone environment.**

  With stand-alone catalog servers, the logs are in the location where you run the start server command. For container servers, you can use the default location or set a custom log location:

  – **Default log location:** The logs are in the directory where the start server command was run. If you start the servers in the *wxs_home*/`bin` directory, the logs and trace files are in the `logs/<`*`server_name`*`>` directories in the `bin` directory.

  – **Custom log location:** To specify an alternate location for container server logs, create a properties file, such as `server.properties`, with the following contents:

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

  The **`workingDirectory`** property is the root directory for the logs and optional trace file. WebSphere eXtreme Scale creates a directory with the name of the container server with a `SystemOut.log` file, a `SystemErr.log` file, and a trace file. To use a properties file during container startup, use the **`-serverProps`** option and provide the server properties file location.

- **Enable logs in WebSphere Application Server.**

See WebSphere Application Server: Enabling and disabling logging for more information.

- **Retrieve FFDC files.**

  FFDC files are for IBM support to aid in debug. These files might be requested by IBM support when a problem occurs. These files are in a directory labeled, `ffdc`, and contain files that resemble the following:

  ```
  server2_exception.log
  server2_20802080_07.03.05_10.52.18_0.txt
  ```

What to do next

View the log files in their specified locations. Common messages to look for in the `SystemOut.log` file are start confirmation messages, such as the following example:

```
CWOBJ1001I: ObjectGrid Server catalogServer01 is ready to process requests.
```

For more information about a specific message in the log files, see Messages.

Related reference:

"Server trace options" on page 621
You can enable trace to provide information about your environment to IBM support.

Messages
When you encounter a message in a log or other parts of the product interface, you can look up the message by its component prefix to find out more information.

Collecting trace

You can use trace to monitor and troubleshoot your environment. You must provide trace for a server when you work with IBM support.

About this task

Collecting trace can help you monitor and fix problems in your deployment of WebSphere eXtreme Scale. How you collect trace depends on your configuration. See "Server trace options" on page 621 for a list of the different trace specifications you can collect.

Procedure

- **Collect trace within a WebSphere Application Server environment.**

  If your catalog and container servers are in a WebSphere Application Server environment, see WebSphere Application Server: Working with trace for more information.

- **Collect trace with the stand-alone catalog or container server start command.**

  You can set trace on a catalog service or container server by using the **`-traceSpec`** and **`-traceFile`** parameters with the start server command. For example:

  ```
  startOgServer.sh catalogServer -traceSpec ObjectGridPlacement=all=enabled -traceFile  /home/user1/logs/trace.log
  ```

  The **`-traceFile`** parameter is optional. If you do not set a **`-traceFile`** location, the trace file goes to the same location as the system out log files.For more information about these parameters, see "**`startOgServer`** script" on page 466.

- **Collect trace on the stand-alone catalog or container server with a properties file.**

To collect trace from a properties file, create a file, such as a `server.properties` file, with the following contents:

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

The **`workingDirectory`** property is the root directory for the logs and optional trace file. If the **`workingDirectory`** value is not set, the default working directory is the location used to start the servers, such as *wxs_home*/`bin`. To use a properties file during server startup, use the **`-serverProps`** parameter with the **`startOgServer`** command and provide the server properties file location.For more information about the server properties file and how to use the file, see Server properties file.

- **Collect trace on a stand-alone client.**

  You can start trace collection on a stand-alone client by adding system properties to the startup script for the client application. In the following example, trace settings are specified for the `com.ibm.samples.MyClientProgram` application:

  ```
  java -DtraceSettingsFile=MyTraceSettings.properties
  -Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
  -Djava.util.logging.configureByServer=true com.ibm.samples.MyClientProgram
  ```

  Fore more information, see WebSphere Application Server: Enabling trace on client and stand-alone applications.

- **Collect trace with the ObjectGridManager interface.**

  You can also set trace during run time on an ObjectGridManager interface. Setting trace on an ObjectGridManager interface can be used to get trace on an eXtreme Scale client while it connects to an eXtreme Scale and commits transactions. To set trace on an ObjectGridManager interface, supply a trace specification and a trace log.

  ```
  ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
  ...
  manager.setTraceEnabled(true);
  manager.setTraceFileName("logs/myClient.log");
  manager.setTraceSpecification("ObjectGridReplication=all=enabled");
  ```

  For more information about the ObjectGridManager interface, see Interacting with an ObjectGrid using the ObjectGridManager interface.

- **Collect trace on container servers with the `xscmd` utility.**

  To collect trace with the **`xscmd`** utility, use the **`-c setTraceSpec`** command. Use the **`xscmd`** utility to collect trace on a stand-alone environment during run time instead of during startup. You can collect trace on all servers and catalog services or you can filter the servers based on the ObjectGrid name, and other properties. For example, to collect ObjectGridReplication trace with access to the catalog service server, run:

  ```
  xscmd -c setTraceSpec -spec "ObjectGridReplication=all=enabled"
  ```

  You can also disable trace by setting the trace specification to `*=all=disabled`.

Results

Trace files are written to the specified location.

Related reference:

"Server trace options" on page 621
You can enable trace to provide information about your environment to IBM support.

Messages
When you encounter a message in a log or other parts of the product interface, you can look up the message by its component prefix to find out more information.

Server trace options

You can enable trace to provide information about your environment to IBM support.

Trace option components

WebSphere eXtreme Scale trace is divided into several different components. You can specify the level of trace to use for a catalog server or container server. Common levels of trace include: all, debug, entryExit, and event.

An example trace string follows:

```
ObjectGridComponent=level=enabled
```

You can concatenate trace strings. Use the * (asterisk) symbol to specify a wildcard value, such as `ObjectGrid*=all=enabled`. If you need to provide a trace to IBM support, a specific trace string is requested. For example, if a problem with replication occurs, the `ObjectGridReplication=debug=enabled` trace string might be requested.

Trace specification

Table 37. Trace options. Trace options for WebSphere eXtreme Scale

Trace option	Description
ObjectGrid	General core cache engine.
ObjectGridCacheInvalidator	Near-cache invalidation
ObjectGridCatalogServer	General catalog service.
ObjectGridChannel	Static deployment topology communications.
ObjectGridClientInfo	DB2 client information.
ObjectGridClientInfoUser	DB2 user information.
ObjectgridCORBA	Dynamic deployment topology communications.
ObjectGridDataGrid	The AgentManager API.
ObjectGridDynaCache	The WebSphere eXtreme Scale dynamic cache provider.
ObjectGridEntityManager	The EntityManager API. Use with the Projector option.
ObjectGridEvictors	ObjectGrid built-in evictors.
ObjectGridJPA	Java Persistence API (JPA) loaders.
ObjectGridJPACache	JPA cache plug-ins.
ObjectGridLocking	ObjectGrid cache entry lock manager.
ObjectGridMBean	Management beans.
ObjectGridMonitor	Historical monitoring infrastructure.
7.1.1+ ObjectGridNative	WebSphere eXtreme Scale native code trace, including eXtremeMemory native code.

Table 37. Trace options (continued). Trace options for WebSphere eXtreme Scale

Trace option	**Description**
7.1.1+ ObjectGridOSGi	The WebSphere eXtreme Scale OSGi integration components.
ObjectGridPlacement	Catalog server shard placement service.
ObjectGridPubSub	Catalog and container server shard placement service.
ObjectGridQuery	ObjectGrid query.
ObjectGridReplication	Replication service.
ObjectGridRest	REST gateway.
ObjectGridRouting	Client/server routing details.
ObjectGridSecurity	Security trace.
7.1.1+ ObjectGridSerializer	The DataSerializer plug-in infrastructure.
ObjectGridStats	ObjectGrid statistics.
ObjectGridStreamQuery	The Stream Query API.
7.1.1+ ObjectGridTransactionManager	The WebSphere eXtreme Scale transaction manager.
ObjectGridWriteBehind	ObjectGrid write behind.
7.1.1+ ObjectGridXM	General IBM eXtremeMemory trace.
7.1.1+ ObjectGridXMEviction	eXtremeMemory eviction trace.
7.1.1+ ObjectGridXMTransport	eXtremeMemory general transport trace.
7.1.1+ ObjectGridXMTransportInbound	eXtremeMemory inbound specific transport trace.
ObjectGridXMTransportOutbound	eXtremeMemory outbound specific transport trace.
Projector	The engine within the EntityManager API.
QueryEngine	The query engine for the Object Query API and EntityManager Query API.
QueryEnginePlan	Query plan trace.
7.1.1+ TCPChannel	The IBM eXtremeIO TCP/IP channel.
7.1.1+ WXSRevision	Revision control for replication.
7.1.1+ XsByteBuffer	WebSphere eXtreme Scale byte buffer trace.

Related tasks:

"Enabling logging" on page 618
You can use logs to monitor and troubleshoot your environment.

"Collecting trace" on page 619
You can use trace to monitor and troubleshoot your environment. You must provide trace for a server when you work with IBM support.

"Starting stand-alone servers" on page 460
When you are running a stand-alone configuration, the environment is comprised of catalog servers, container servers, and client processes. WebSphere eXtreme Scale servers can also be embedded within existing Java applications by using the embedded server API. You must manually configure and start these processes.

"Administering with the **xscmd** utility" on page 482
With the **xscmd** utility, you can complete administrative tasks in the environment such as: establishing multi-master replication links, overriding quorum, and stopping groups of servers with the teardown command.

Analyzing log and trace data

You can use the log analysis tools to analyze how your runtime environment is performing and solve problems that occur in the environment.

About this task

You can generate reports from the existing log and trace files in the environment. These visual reports can be used for the following purposes:

- **To analyze runtime environment status and performance:**
  - Deployment environment consistency
  - Logging frequency
  - Running topology versus configured topology
  - Unplanned topology changes
  - Quorum status
  - Partition replication status
  - Statistics of memory, throughput, processor usage, and so on
- **To troubleshoot problems in the environment:**
  - Topology views at specific points in time
  - Statistics of memory, throughput, processor usage during client failures
  - Current fix pack levels, tuning settings
  - Quorum status

Log analysis overview

You can use the **xsLogAnalyzer** tool to help troubleshoot issues in the environment.

All failover messages

Displays the total number of failover messages as a chart over time. Also displays a list of the failover messages, including the servers that have been affected

All eXtreme Scale critical messages

Displays message IDs along with the associated explanations and user actions, which can save you the time from searching for messages.

All exceptions

Displays the top five exceptions, including the messages and how many times they occurred, and what servers were affected by the exception.

Topology summary

Displays a diagram of how your topology is configured according to the log files. You can use this summary to compare to your actual configuration, possibly identifying configuration errors.

Topology consistency: Object Request Broker (ORB) comparison table

Displays ORB settings in the environment. You can use this table to help determine if the settings are consistent across your environment.

Event timeline view

Displays a timeline diagram of different actions that have occurred on the data grid, including life cycle events, exceptions, critical messages, and first-failure data capture (FFDC) events.

Running log analysis

You can run the **xsLogAnalyzer** tool on a set of log and trace files from any computer.

Before you begin

- Enable logs and trace. See "Enabling logging" on page 618 and "Collecting trace" on page 619 for more information.
- Collect your log files. The log files can be in various locations depending on how you configured them. If you are using the default log settings, you can get the log files from the following locations:
  - In a stand-alone installation: *wxs_install_root*`/bin/logs/<`*`server_name`*`>`
  - In an installation that is integrated with WebSphere Application Server: *was_root*`/logs/<`*`server_name`*`>`
- Collect your trace files. The trace files can be in various locations depending on how you configured them. If you are using the default trace settings, you can get the trace files from the following locations:
  - In a stand-alone installation: If no specific trace value is set, the trace files are written to the same location as the system out log files.
  - In an installation that is integrated with WebSphere Application Server: *was_root*`/profiles/`*`server_name`*`/logs`.

  Copy the log and trace files to the computer from which you are planning to use the log analyzer tool.
- If you want to create custom scanners in your generated report, create a scanner specifications properties file and configuration file before you run the tool. For more information, see "Creating custom scanners for log analysis" on page 625.

Procedure

1. Run the **xsLogAnalyzer** tool.

   The script is in the following locations :
   - In a stand-alone installation: *wxs_install_root*`/ObjectGrid/bin`
   - In an installation that is integrated with WebSphere Application Server: *was_root*`/bin`

   Tip: If your log files are large, consider using the **-startTime**, **-endTime**, and **-maxRecords** parameters when you run the report to restrict the number of log entries that are scanned. Using these parameters when you run the report makes the reports easier to read and run more effectively. You can run multiple reports on the same set of log files.

   ```
   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
Specifies the absolute path to the log directory that you want to evaluate (required).

-outDir
Specifies an existing directory to write the report output. If you do not specify a value, the report is written to the root location of the **`xsLogAnalyzer`** tool.

-startTime
Specifies the start time to evaluate in the logs. The date is in the following format: *`year.month.day_hour.minute.second.millisecond`*

-endTime
Specifies the end time to evaluate in the logs. The date is in the following format: *`year.month.day_hour.minute.second.millisecond`*

-trace Specifies a trace string, such as `ObjectGrid*=all=enabled`.

-maxRecords
Specifies the maximum number of records to generate in the report. The default is `100`. If you specify the value as 50, the first 50 records are generated for the specified time period.

2. Open the generated files. If you did not define an output directory, the reports are generated in a folder called `report_`*`date_time`*. To open the main page of the reports, open the `index.html` file.
3. Use the reports to analyze the log data. Use the following tips to maximize the performance of the report displays:
   - To maximize the performance of queries on the log data, use as specific information as possible. For example, a query for `server` takes much longer to run and returns more results than `server_host_name`.
   - Some views have a limited number of data points that are displayed at one time. You can adjust the segment of time that is being viewed by changing the current data, such as start and end time, in the view.

What to do next

For more information about troubleshooting the **`xsLogAnalyzer`** tool and the generated reports, see "Troubleshooting log analysis" on page 626.

Creating custom scanners for log analysis

You can create custom scanners for log analysis. After you configure the scanner, the results are generated in the reports when you run the **`xsLogAnalyzer`** tool. The custom scanner scans the logs for event records based on the regular expressions that you specified.

Procedure

1. Create a scanner specifications properties file that specifies the general expression to run for the custom scanner.
   a. Create and save a properties file. The file must be in the *`loganalyzer_root`*`/config/custom` directory. You can name the file as: you like. The file is used by the new scanner, so naming the scanner in the properties file is useful, for example: *`my_new_server_scanner_spec`*`.properties`.
   b. Include the following properties in the *`my_new_server_scanner_spec`*`.properties` file:

```
include.regular_expression = REGULAR_EXPRESSION_TO_SCAN
```

The *REGULAR_EXPRESSION_TO_SCAN* variable is a regular expression on which to filter the log files.

Example: To scan for instances of lines that contain both the `"xception"` and `"rror"` strings regardless of the order, set the **`include.regular_expression`** property to the following value:

```
include.regular_expression = (xception.+rror)|(rror.+xception)
```

This regular expression causes events to be recorded if the string `"rror"` comes before or after the `"xception"` string.

Example:To scan through each line in the logs for instances of lines that contain either the phrase `"xception"` or the phrase `"rror"` strings regardless of the order, set the **`include.regular_expression`** property to the following value:

```
include.regular_expression = (xception)|(rror)
```

This regular expression causes events to be recorded if the either the `"rror"` string or the `"xception"` string exist.

2. Create a configuration file that the **`xsLogAnalyer`** tool uses to create the scanner.
   a. Create and save a configuration file. The file must be in the *`loganalyzer_root`*`/config/custom` directory. You can name the file as *`scanner_name`*`Scanner.config`, where *scanner_name* is a unique name for the new scanner. For example, you might name the file `serverScanner.config`
   b. Include the following properties in the *`scanner_name`*`Scanner.config` file:

      ```
      scannerSpecificationFiles = LOCATION_OF_SCANNER_SPECIFICATION_FILE
      ```

      The *LOCATION_OF_SCANNER_SPECIFICATION_FILE* variable is the path and location of the specification file that you created in the previous step. For example: *`loganalyzer_root`*`/config/custom/my_new_scanner_spec.properties`. You can also specify multiple scanner specification files by using a semi-colon separated list:

```
scannerSpecificationFiles = LOCATION_OF_SCANNER_SPECIFICATION_FILE1;LOCATION_OF_SCANNER_SPECIFICATION_FILE2
```

3. Run the **`xsLogAnalyzer`** tool. For more information, see “Running log analysis” on page 624.

Results

After you run the **`xsLogAnalyzer`** tool, the report contains new tabs in the report for the custom scanners that you configured. Each tab contains the following views:

Charts A plotted graph that illustrates recorded events. The events are displayed in the order in which the events were found.

Tables A tabular representation of the recorded events.

Summary reports

Troubleshooting log analysis

Use the following troubleshooting information to diagnose and fix problems with the **`xsLogAnalyzer`** tool and its generated reports.

Procedure

- **Problem:** Out of memory conditions occur when you are using the **`xsLogAnalyzer`** tool to generate reports. An example of an error that might occur follows: `java.lang.OutOfMemoryError: GC overhead limit exceeded`.

  Solution: The **`xsLogAnalyzer`** tool runs within a Java virtual machine (JVM). You can configure the JVM to increase the heap size before you run the **`xsLogAnalyzer`** tool by specifying some settings when you run the tool. Increasing the heap size enables more event records to be stored in JVM memory. Start with a setting of 2048M, assuming the operating system has enough main memory. On the same command-line instance in which you are planning to run the **`xsLogAnalyzer`** tool, set the maximum JVM heap size:

  ```
  java -XmxHEAP_SIZEm
  ```

  The *HEAP_SIZE* value can be any integer and represents the number of megabytes that are allocated to JVM heap. For example, you might run `java -Xmx2048m`. If the out of memory messages continue, or you do not have the resources to allocate 2048m or more of memory, limit the number of events that are being held in the heap. You can limit the number of events in the heap up by passing the **`-maxRecords`** parameter to the .**`xsLogAnalyzer`** command
- **Problem:** When you open a generated report from the **`xsLogAnalyzer`** tool, the browser hangs or does not load the page.

  Cause: The generated HTML files are too large and cannot be loaded by the browser. These files are large because the scope of the log files that you are analyzing is too broad.

  Solution: Consider using the **`-startTime`**, **`-endTime`**, and **`-maxRecords`** parameters when you run the **`xsLogAnalyzer`** tool to restrict the number of log entries that are scanned. Using these parameters when you run the report makes the reports easier to read and run more effectively. You can run multiple reports on the same set of log files.

Troubleshooting installation

Use this information to troubleshoot issues with your installation and updates.

Procedure

- **Problem:** When you run the installation command from a remote computer, such as `\\mymachine\downloads\`, the following message displays: `CMD.EXE was started with the above path as the current directory. UNC paths are not supported. Defaulting to Windows directory.` As a result, the installation does not complete correctly.

  Solution: Map the remote computer to a network drive. For example, in Windows, you can right-click **My computer** and choose **Map Network Drive** and include the uniform naming conventions (UNC) path to the remote computer. You can then run the installation script from the network drive successfully, for example: `y:\mymachine\downloads\WXS\install.bat`.
- **Problem:** The installation completes unsuccessfully.

  Solution: Check the log files to see where the installation failed. When the installation completes unsuccessfully, the logs are in the *wxs_install_root*`/logs/wxs` directory.
- **Problem:** A catastrophic failure occurs during the installation.

  Solution: Check the log files to see where the installation failed. When the installation fails when it is partially completed, the logs can generally be found in the *`user_root`*`/wxs_install_logs/` directory.

- Windows **Problem:** If you are installing the WebSphere eXtreme Scale Client on Windows, you might see the following text in the results of the installation:

  ```
  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.
  ```

  Solution: If you see a failure with the `iscdeploy.sh` file, you can ignore the error. This error does not cause any problems.
- Linux **Problem:**

  If you have a full installation and try to apply WebSphere eXtreme Scale Client only maintenance with the update installer, you see the following message:

  ```
  Prerequisite checking has failed. Click Back to select a different package,
  or click Cancel to exit.

  Failure messages are:

  Required feature wxs.client.primary is not found.
  ```

  If you have WebSphere eXtreme Scale Client installed and try to apply a full maintenance package with the update installer, you see the following message:

  ```
  Prerequisite checking has failed. Click Back to select a different package,
  or click Cancel to exit.

  Failure messages are:

  Required feature wxs.primary is not found.
  ```

  Solution: The maintenance package that you install must match the type of installation. Download and apply the maintenance package that applies to your installation type.
- Linux **Problem:** The installation hangs.

  Solution: Sometimes, when installing WebSphere eXtreme Scale on Linux as a non-root user, the installer can hang. This is likely because the maximum number of open files is set too low on your Linux operating system. You will need to raise the allowed limit in the `/etc/limits.conf` or `/etc/security/limits.conf` file (where the file is located depends on your specific Linux distribution) to at least 8192.

Troubleshooting cache integration

Use this information to troubleshoot issues with your cache integration configuration, including HTTP session and dynamic cache configurations.

Procedure

- 7.1.1+ **Problem:** HTTP session IDs are not being reused.

  Cause: You can reuse session IDs. If you create a data grid for session persistence in Version 7.1.1 or later, session ID reuse is automatically enabled. However, if you created prior configurations, this setting might already be set with the wrong value.

  Solution: Check the following settings to verify that you have HTTP session ID reuse enabled:

  – The `reuseSessionId` property in the `splicer.properties` file must be set to `true`.

– The `HttpSessionIdReuse` custom property value must be set to `true`. This custom property might be set on one of the following paths in the WebSphere Application Server administrative console:
  - **Servers** > *server_name* > **Session management** > **Custom properties**
  - **Dynamic clusters** > *dynamic_cluster_name* > **Server template** > **Session management** > **Custom properties**
  - **Servers** > **Server Types** > **WebSphere application servers** > *server_name*, and then, under Server Infrastructure, click **Java and process management** > **Process definition** > **Java virtual machine** > **Custom properties**
  - **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Web container settings** > **Web container**

  If you update any custom property values, reconfigure eXtreme Scale session management so the `splicer.properties` file becomes aware of the change.

- **Problem:** When you are using a data grid to store HTTP sessions and the transaction load is high, a CWOBJ0006W message displays in the `SystemOut.log` file.

  ```
  CWOBJ0006W: An exception occurred:
  com.ibm.websphere.objectgrid.ObjectGridRuntimeException:
  java.util.ConcurrentModificationException
  ```

  This message occurs only when the **`replicationInterval`** parameter in the `splicer.properties` file is set to a value greater than zero and the Web application modifies a List object that was set as an attribute on the HTTPSession.

  Solution: Clone the attribute that contains the modified List object and put the cloned attribute into the session object.

Related reference:

"XML files for HTTP session manager configuration" on page 382
When you start a container server that stores HTTP session data, you can either use the default XML files or you can specify customized XML files. These files create specific ObjectGrid names, number of replicas, and so on.

"Servlet context initialization parameters" on page 388
The following list of servlet context initialization parameters can be specified in the splicer properties file as required in the chosen splicing method.

"`splicer.properties` file" on page 392
The `splicer.properties` file contains all of the configuration options for configuring a servlet-filter-based session manager.

Troubleshooting the JPA cache plug-in

Use this information to troubleshoot issues with your JPA cache plug-in configuration. These problems can occur in both Hibernate and OpenJPA configurations.

Procedure

- **Problem:** The following exception displays: `CacheException: Failed to get ObjectGrid server`.

  With either an `EMBEDDED` or `EMBEDDED_PARTITION` **`ObjectGridType`** attribute value, the eXtreme Scale cache tries to obtain a server instance from the run time. In a Java Platform, Standard Edition environment, an eXtreme Scale server with embedded catalog service is started. The embedded catalog service tries to listen to port 2809. If that port is being used by another process, the error occurs.

Solution: If external catalog service endpoints are specified, for example, with the `objectGridServer.properties` file, this error occurs if the host name or port is specified incorrectly. Correct the port conflict.

- **Problem:** The following exception displays: `CacheException: Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]`

  This error occurs because the cache cannot get the ObjectGrid instance from the provided catalog service end points.

  Solution: This problem typically occurs because of an incorrect host name or port.
- **Problem:** The following exception displays: `CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType`

  This exception results if you have many persistence units configured and the eXtreme Scale caches of these units are configured with the same ObjectGrid name and `EMBEDDED` **`ObjectGridType`** attribute value. These persistence unit configurations could be in the same or different `persistence.xml` files.

  Solution: You must verify that the ObjectGrid name is unique for each persistence unit when the **`ObjectGridType`** attribute value is `EMBEDDED`.
- **Problem:** The following exception displays: `CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]`

  With a REMOTE ObjectGrid type, if the obtained client-side ObjectGrid does not have complete entity backing maps to support the persistence unit cache, this exception occurs. For example, five entity classes are listed in the persistence unit configuration, but the obtained ObjectGrid only has two BackingMaps. Even though the obtained ObjectGrid might have 10 BackingMaps, if any one of the five required entity BackingMaps are not found in the 10 backing maps, this exception still occurs.

  Solution: Make sure that your backing map configuration supports the persistence unit cache.

Troubleshooting IBM eXtremeMemory

Use the following information to troubleshoot eXtremeMemory.

Procedure

Problem: If the shared resource, libstdc++.so.5, is not installed, then when you start the container server, IBM eXtremeMemory native libraries do not load.

Linux **Symptom:** On a Linux 64-bit operating system, if you try to start a container server with the enableXM server property set to `true`, and the `libstdc++.so.5` shared resource is not installed, you get an error similar to the following example:

```
00000000 Initialization W CWOBJ0006W: An exception occurred: java.lang.reflect.InvocationTargetException
 at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
 at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:56)
 at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:39)
 at java.lang.reflect.Constructor.newInstance(Constructor.java:527)
 at com.ibm.websphere.objectgrid.server.ServerFactory.initialize(ServerFactory.java:350)
 at com.ibm.websphere.objectgrid.server.ServerFactory$2.run(ServerFactory.java:303)
 at java.security.AccessController.doPrivileged(AccessController.java:202)
 at com.ibm.websphere.objectgrid.server.ServerFactory.getInstance(ServerFactory.java:301)
 at com.ibm.ws.objectgrid.InitializationService.main(InitializationService.java:302)

Caused by: com.ibm.websphere.objectgrid.ObjectGridRuntimeException: java.lang.UnsatisfiedLinkError:
 OffheapMapdbg (Not found in java.library.path)
 at com.ibm.ws.objectgrid.ServerImpl.<init&gt;(ServerImpl.java:1033)
... 9 more Caused by: java.lang.UnsatisfiedLinkError: OffheapMapdbg (Not found in java.library.path)
```

```
 at java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1011)
 at java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:975)
 at java.lang.System.loadLibrary(System.java:469)
 at com.ibm.ws.objectgrid.io.offheap.ObjectGridHashTableOH.initializeNative(ObjectGridHashTableOH.java:112)
 at com.ibm.ws.objectgrid.io.offheap.ObjectGridHashTableOH.<clinit&gt;(ObjectGridHashTableOH.java:87)
 at java.lang.J9VMInternals.initializeImpl(Native Method)
 at java.lang.J9VMInternals.initialize(J9VMInternals.java:200)
 at com.ibm.ws.objectgrid.ServerImpl.<init&gt;(ServerImpl.java:1028)
... 9 more
```

Cause: The shared resource `libstdc++.so.5` has not been installed.
Diagnosing the problem: To verify that the resource `libstdc++.so.5` is installed, issue the following command from the ObjectGrid/native directory of your installation:

```
ldd libOffheapMap.so
```

If you do not have the shared library installed, you get the following error:

```
ldd libOffheapMap.so
libstdc++.so.5 => not found
```

Resolving the problem: Use the package installer of your 64-bit Linux distribution to install the required resource file. The package might be listed as `compat-libstdc++-33.x86_64` or `libstdc++5`. After installing the required resource, verify that the `libstdc++5`package is installed by issuing the following command from the ObjectGrid directory of your installation:

```
ldd libOffheapMap.so
```

Troubleshooting administration

Use the following information to troubleshoot administration, including starting and stopping servers, using the **xscmd** utility, and so on.

Procedure

- **Problem:** Administration scripts are missing from the *profile_root*/bin directory of a WebSphere Application Server installation.

  Cause: When you update the installation, new script files do not automatically get installed in the profiles.

  Solution: If you want to run a script from your *profile_root*/bin directory, unaugment and reaugment the profile with the latest release. For more information, see Unaugmenting a profile using the command prompt and "Creating and augmenting profiles for WebSphere eXtreme Scale" on page 203.
- **Problem:** When you are running a **xscmd** command, the following message is printed to the screen:

  ```
  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
  ```

  Cause: A connection problem occurred with the catalog server.

  Solution: Verify that your catalog servers are running and are available through the network. This message can also occur when you have a catalog service domain defined, but less than two catalog servers are running. The environment is not available until two catalog servers are started.

Related concepts:

"Example: Configuring catalog service domains" on page 295
When you are using the catalog service, a minimum of two catalog servers are required to avoid a single point of failure. Depending on the number of nodes in your environment, you can create different configurations to ensure that at least two catalog servers are always running.

Chapter 7, "Administering," on page 459

Troubleshooting multiple data center configurations

Use this information to troubleshoot multiple data center configurations, including linking between catalog service domains.

Before you begin

You must use the **xscmd** utility to troubleshoot your multiple data center configurations. For more information, see "Administering with the **xscmd** utility" on page 482.

Procedure

Problem: Data is missing in one or more catalog service domains. For example, you might run the **xscmd -c establishLink** command. When you look at the data for each linked catalog service domain, the data looks different, for example from the **xscmd -c showMapSizes** command.
Solution: You can troubleshoot this problem with the **xscmd -c showLinkedPrimaries** command. This command prints out each primary shard, and including which foreign primaries are linked.
In the described scenario, you might discover from running the **xscmd -c showLinkedPrimaries** command that the first catalog service domain primary shards are linked to the second catalog service domain primary shards, but the second catalog service domain does not have links to the first catalog service domain. You might consider rerunning the **xscmd -c establishLink** command from the second catalog service domain to the first catalog service domain.

Related concepts:

"Planning multiple data center topologies" on page 35
Using multi-master asynchronous replication, two or more data grids can become exact copies of each other. Each data grid is hosted in an independent catalog service domain, with its own catalog service, container servers, and a unique name. With multi-master asynchronous replication, you can use links to connect a collection of catalog service domains. The catalog service domains are then synchronized using replication over the links. You can construct almost any topology through the definition of links between the catalog service domains.

"JPA level 2 (L2) cache plug-in" on page 401
WebSphere eXtreme Scale includes level 2 (L2) cache plug-ins for both OpenJPA and Hibernate Java Persistence API (JPA) providers. When you use one of these plug-ins, your application uses the JPA API. A data grid is introduced between the application and the database, improving response times.

Related reference:

"Example: OpenJPA ObjectGrid XML files" on page 410
OpenJPA ObjectGrid XML files should be created based on the configuration of the persistence unit.

"Example: Hibernate ObjectGrid XML files" on page 417
Create Hibernate ObjectGrid XML files based on the configuration of a persistence

unit.

Related information:

Improve response time and data availability with WebSphere eXtreme Scale multi-master capability

com.ibm.websphere.objectgrid.openJPA package

com.ibm.websphere.objectgrid.hibernate.cache package

Troubleshooting loaders

Use this information to troubleshoot issues with your database loaders.

Procedure

- **Problem:** The loader is unable to communicate with the database. A LoaderNotAvailableException exception occurs.

  Explanation: The loader plug-in can fail when it is unable to communicate to the database back end. This failure can happen if the database server or the network connection is down. The write-behind loader queues the updates and tries to push the data changes to the loader periodically. The loader must notify the ObjectGrid run time that there is a database connectivity problem by throwing a LoaderNotAvailableException exception.

  Solution: The Loader implementation must be able to distinguish a data failure or a physical loader failure. Data failure should be thrown or rethrown as a LoaderException or an OptimisticCollisionException, but a physical loader failure must be thrown or rethrown as a LoaderNotAvailableException. ObjectGrid handles these two exceptions differently:

  – If a LoaderException is caught by the write-behind loader, the write-behind loader considers the exception a failure, such as duplicate key failure. The write-behind loader unbatches the update, and tries the update one record at one time to isolate the data failure. If A {{LoaderException}}is caught again during the one record update, a failed update record is created and logged in the failed update map.

  – If a LoaderNotAvailableException is caught by the write-behind loader, the write-behind loader considers it failed because it cannot connect to the database end, for example, the database back-end is down, a database connection is not available, or the network is down. The write-behind loader waits for 15 seconds and then try the batch update to the database again.

  The common mistake is to throw a LoaderException while a LoaderNotAvailableException must be thrown. All the records queued in the write-behind loader become failed update records, which defeats the purpose of back-end failure isolation.

- **Problem:** When you are using an OpenJPA loader with DB2 in WebSphere Application Server, a closed cursor exception occurs.

  The following exception is from DB2 in the org.apache.openjpa.persistence.PersistenceException log file:

  ```
  [jcc][t4][10120][10898][3.57.82] Invalid operation: result set is closed.
  ```

  Solution: By default, the application server configures the resultSetHoldability custom property with a value of 2 (CLOSE_CURSORS_AT_COMMIT). This property causes DB2 to close its resultSet/cursor at transaction boundaries. To remove the exception, change the value of the custom property to 1 (HOLD_CURSORS_OVER_COMMIT). Set the resultSetHoldability custom property on the following path in the WebSphere Application Server cell:

Resources > **JDBC provider** > **DB2 Universal JDBC Driver Provider** > **DataSources** > ***data_source_name*** > **Custom properties** > **New**.

- **Problem** DB2 displays an exception: `The current transaction has been rolled back because of a deadlock or timeout. Reason code "2".. SQLCODE=-911, SQLSTATE=40001, DRIVER=3.50.152`

  This exception occurs because of a lock contention problem when you are running with OpenJPA with DB2 in WebSphere Application Server. The default isolation level for WebSphere Application Server is Repeatable Read (RR), which obtains long-lived locks with DB2.**Solution:**

  Set the isolation level to Read Committed to reduce the lock contention. Set the webSphereDefaultIsolationLevel data source custom property to set the isolation level to 2(TRANSACTION_READ_COMMITTED) on the following path in the WebSphere Application Server cell: **Resources** > **JDBC provider** > ***JDBC_provider*** > **Data sources** > ***data_source_name*** > **Custom properties** > **New**. For more information about the webSphereDefaultIsolationLevel custom property and transaction isolation levels, see Requirements for setting data access isolation levels.
- **Problem:** When you are using the preload function of the JPALoader or JPAEntityLoader, the following CWOBJ1511 message does not display for the partition in a container server: `CWOBJ1511I: GRID_NAME:MAPSET_NAME:PARTITION_ID (primary) is open for business.`

  Instead, a TargetNotAvailableException exception occurs in the container server, which activates the partition that is specified by the preloadPartition property.

  Solution: Set the preloadMode attribute to `true` if you use a JPALoader or JPAEntityLoader to preload data into the map. If the preloadPartition property of the JPALoader and JPAEntityLoader is set to a value between `0` and `total_number_of_partitions - 1`, then the JPALoader and JPAEntityLoader try to preload the data from backend database into the map. The following snippet of code illustrates how the preloadMode attribute is set to enable asynchronous preload:

  ```
  BackingMap bm = og.defineMap( "map1" );
  bm.setPreloadMode( true );
  ```

  You can also set the preloadMode attribute by using an XML file as illustrated in the following example:

  ```
   <backingMap name="map1" preloadMode="true" pluginCollectionRef="map1"
    lockStrategy="OPTIMISTIC" />
  ```

Related concepts:

Programming for JPA integration
The Java Persistence API (JPA) is a specification that allows mapping Java objects to relational databases. JPA contains a full object-relational mapping (ORM) specification using Java language metadata annotations, XML descriptors, or both to define the mapping between Java objects and a relational database. A number of open-source and commercial implementations are available.

"Configuring cache integration" on page 364
WebSphere eXtreme Scale can integrate with other caching-related products. You can also use the WebSphere eXtreme Scale dynamic cache provider to plug WebSphere eXtreme Scale into the dynamic cache component in WebSphere Application Server. Another extension to WebSphere Application Server is the WebSphere eXtreme Scale HTTP session manager, which can help to cache HTTP sessions.

Troubleshooting XML configuration

When you configure eXtreme Scale, you can encounter unexpected behavior with your XML files. The following sections describe problems that can occur and solutions.

Procedure

- **Problem:** Your deployment policy and ObjectGrid XML files must match.

  The deployment policy and ObjectGrid XML files must match. If they do not have matching ObjectGrid names and map names, errors occur.

  If the backingMap list in an ObjectGrid XML file does not match the map references list in a deployment policy XML file, an error occurs on the catalog server.

  For example, the following ObjectGrid XML file and deployment policy XML file are used to start a container process. The deployment policy file has more map references than are listed in the ObjectGrid XML file.

  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>
```

  Messages: An error message occurs in the `SystemOut.log` file when the deployment policy is incompatible with the ObjectGrid XML file. For the preceding example, the following message occurs:

```
CWOBJ3179E: The map ledger reference in the mapSet mapSet1 of ObjectGrid accounting
deployment descriptor file does not reference a valid backing map from the ObjectGrid
XML.
```

  If the deployment policy is missing map references to backingMaps that are listed in the ObjectGrid XML file, an error message occurs in the `SystemOut.log` file. For example:

```
CWOBJ3178E: The map ledger in ObjectGrid accounting referenced in the ObjectGrid XML
was not found in the deployment descriptor file.
```

  Solution: Determine which file has the correct list and alter the relevant code accordingly.

- **Problem:** Incorrect ObjectGrid names between XML files also causes and error.

  The name of the ObjectGrid is referenced in both the ObjectGrid XML file and the deployment policy XML file.

  Message: An ObjectGridException occurs with a caused by exception of IncompatibleDeploymentPolicyException. An example follows.

  Caused by:
  com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: The

objectgridDeployment with objectGridName "accountin" does not have a corresponding objectGrid in the ObjectGrid XML.

The ObjectGrid XML file is the master list of ObjectGrid names. If a deployment policy has an ObjectGrid name that is not contained in the ObjectGrid XML file, an error occurs.

Solution: Verify details such as the spelling of the ObjectGrid name. Remove any extra names, or add missing ObjectGrid names, to the ObjectGrid XML or deployment policy XML files. In the example message, the objectGridName is misspelled as "accountin" instead of "accounting".

- **Problem:** Some of the attributes in the XML file can only be assigned certain values. These attributes have acceptable values enumerated by the schema. The following list provides some of the attributes:
  - authorizationMechanism attribute on the objectGrid element
  - copyMode attribute on the backingMap element
  - lockStrategy attribute on the backingMap element
  - ttlEvictorType attribute on the backingMap element
  - type attribute on the property element
  - initialState on the objectGrid element
  - evictionTriggers on the backingMap element

  If one of these attributes is assigned an invalid value, XML validation fails. In the following example XML file, an incorrect value of INVALID_COPY_MODE is used:

  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>
  ```

  The following message appears in the log.

  ```
  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.
  ```

- **Problem:** Missing or incorrect attributes or tags in an XML file causes errors, such as the following example in which the ObjectGrid XML file is missing the closing < /objectGrid > tag:

  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>
  ```

  Message:

  ```
  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.
  ```

  An ObjectGridException about the invalid XML file occurs with the name of the XML file.

  Solution: Ensure that the necessary tags and attributes appear in your XML files with correct format.

- **Problem:** If an XML file is formatted with incorrect or missing syntax, the CWOBJ2403E appears in the log. For example, the following message is displayed when a quotation is missing on one of the XML attributes

  ```
  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".
  ```

  An ObjectGridException about the invalid XML file also occurs.

  Solution: Various solutions can be used for a given XML syntax error. Consult relevant documentation about XML script writing.

- **Problem:** Referencing a nonexistent plug-in collection causes an XML file to be invalid. For example, when using XML to define BackingMap plug-ins, the pluginCollectionRef attribute of the backingMap element must reference a backingMapPluginCollection. The pluginCollectionRef attribute must match the backingMapPluginCollection elements.

  Message:

  If the pluginCollectionRef attribute does not match any ID attributes of any of the backingMapPluginConfiguration elements, the following message, or one that is similar, is displayed in the log.

  ```
  [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'.
  ```

  The following XML file is used to produce the error. Notice that the name of the BackingMap book has its pluginCollectionRef attribute set to bookPlugins, and the single backingMapPluginCollection has an ID of collection1.

  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>
  ```

  Solution:

  To fix the problem, ensure that the value of each pluginCollectionRef matches the ID of one of the backingMapPluginCollection elements. Simply change the name of pluginCollectionRef to collection1 to not receive this error. Alternatively, change the ID of the existing backingMapPluginCollection to match the pluginCollectionRef, or add an additional backingMapPluginCollection with an ID that matches the pluginCollectionRef to correct the error.

- **Problem:** The IBM Software Development Kit (SDK) Version 5 contains an implementation of some Java API for XML Processing (JAXP) function to use for XML validation against a schema. When using an SDK that does not contain this implementation, attempts to validate might fail.

  When you attempt to validate XML with an SDK that does not have the necessary implementation, the log contains the following error:

  ```
  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)...
```

The SDK that is used does not contain an implementation of JAXP function that is necessary to validate XML files against a schema.

Solution: If you want to validate XML by using an SDK that does not contain JAXP implementation, download Apache Xerces, and include its Java archive (JAR) files in the classpath. To avoid this problem, after you download Xerces and include the JAR files in the classpath, you can validate the XML file successfully.

Troubleshooting security

Use this information to troubleshoot issues with your security configuration.

Procedure

- **Problem:** The client end of the connection requires Secure Sockets Layer (SSL), with the transportType setting set to `SSL-Required`. However, the server end of the connection does not support SSL, and has the transportType setting set to `TCP/IP`. As a result, the following exception gets chained to another exception in the log files:

  ```
  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)
  ```

  The address in this exception could be a catalog server, container server, or client.

  Solution: See "Configuring secure transport types" on page 592 for a table with the valid security configurations between clients and servers.
- When agent is used, the client sends the agent call to the server, and server sends the response back to the client to acknowledge the agent call. When the agent finishes processing, the server initiates a connection to send the agent results. This makes the container server a client from connect point of view. Therefore, if TLS or SSL is configured, make sure the client public certificate is imported in the server truststore.

IBM Support Assistant for WebSphere eXtreme Scale

You can use the IBM Support Assistant to collect data, analyze symptoms, and access product information.

IBM Support Assistant Lite

IBM Support Assistant Lite for WebSphere eXtreme Scale provides automatic data collection and symptom analysis support for problem determination scenarios.

IBM Support Assistant Lite reduces the amount of time it takes to reproduce a problem with the proper Reliability, Availability, and Serviceability tracing levels set (trace levels are set automatically by the tool) to streamline problem determination. If you need further assistance, IBM Support Assistant Lite also reduces the effort required to send the appropriate log information to IBM Support.

IBM Support Assistant Lite is included in each installation of WebSphere eXtreme Scale Version 7.1.0

IBM Support Assistant

IBM® Support Assistant (ISA) provides quick access to product, education, and support resources that can help you answer questions and resolve problems with IBM software products on your own, without needing to contact IBM Support. Different product-specific plug-ins let you customize IBM Support Assistant for the particular products you have installed. IBM Support Assistant can also collect system data, log files, and other information to help IBM Support determine the cause of a particular problem.

IBM Support Assistant is a utility to be installed on your workstation, not directly onto the WebSphere eXtreme Scale server system itself. The memory and resource requirements for the Assistant could negatively affect the performance of the WebSphere eXtreme Scale server system. The included portable diagnostic components are designed for minimal impact to the normal operation of a server.

You can use IBM Support Assistant to help you in the following ways:
- To search through IBM and non-IBM knowledge and information sources across multiple IBM products to answer a question or solve a problem
- To find additional information through product-specific Web resources; including product and support home pages, customer news groups and forums, skills and training resources and information about troubleshooting and commonly asked questions
- To extend your ability to diagnose product-specific problems with targeted diagnostic tools available in the Support Assistant
- To simplify collection of diagnostic data to help you and IBM resolve your problems (collecting either general or product/symptom-specific data)
- To help in reporting of problem incidents to IBM Support through a customized online interface, including the ability to attach the diagnostic data referenced above or any other information to new or existing incidents

Finally, you can use the built-in Updater facility to obtain support for additional software products and capabilities as they become available. To set up IBM Support Assistant for use with WebSphere eXtreme Scale, first install IBM Support Assistant using the files provided in the downloaded image from the IBM Support Overview Web page at: http://www-947.ibm.com/support/entry/portal/Overview/Software/Other_Software/IBM_Support_Assistant. Next, use IBM Support Assistant to locate and install any product updates. You can also choose to install plug-ins available for other IBM software in your environment. More information and the latest version of the IBM Support Assistant are available from the IBM Support Assistant Web page at: http://www.ibm.com/software/support/isa/.

Notices

Trademarks

Index

A

administration
  overview 459
  troubleshooting 631
  WebSphere Application Server 297
AP 35
APIs
  administration 476
  AvailabilityState 488
  embedded server 480
  MBean 532
  StateManager 488
  statistics 532
architecture
  topologies 14
authentication
  integrating security
    in mixed environments 129
availability
  managing states 488
availability partition (AP) 35
AvailabilityState API 488

B

benefits
  write-behind caching 26
best practices
  real time
    stand-alone environment 578
build definition file
  CIP 189
  IIP 192

C

cache
  distributed 19
  embedded 17
  local 14
cache integration
  configuration 364
  troubleshooting 628
calculations
  partition count 65
  sizing memory 65
capacity planning 65
catalog servers
  configuration 293
catalog service
  best practices for 295
  catalog service domains 475
  cluster 295
  configuration 294
  high availability 295
  starting in an environment that is not running WebSphere Application Server 461
  starting in WebSphere Application Server 475
catalog service *(continued)*
  WebSphere Application Server 298
catalog service domain 295
catalog service domains
  administrative tasks 301
  WebSphere Application Server 299
client authorization
  creator only access 589
  custom 589
  JAAS 589
client-server security
  secure sockets layer (SSL) 593
  TCP/IP 593
  transport layer security (TLS) 593
clients
  invalidation 355
  near cache 353
  overrides 348
  overview 347
  XML configuration 349, 350
coherent cache 21
commands
  manageprofiles 206
  routetable 491
  startOgServer 459
  startXsServer 459
  stopOgServer 459
  stopXsServer 459
  teardown 474
complete cache 22
configuration 51
  data center topologies 330
  methods 257
  overview 257
configuration files
  deployment policy zones example 289
  Hibernate 417
  orb.properties file 566
  wxssetup.response.txt file 197
connection factories
  configuring 362
  configuring Eclipse environments 363
container server
  configuration 324
  placement 486
container servers
  configuration
    overview 293
  starting 464
  WebSphere Application Server
    automatically start 327
    configuration 325
CPU sizing
  for parallel transactions 67
  for transactions 67
CSV file
  statistics definitions 526
csv files 525
custom properties
  ORB properties 566
customization definitions
  generating 232
customized jobs
  running 234
  uploading 234
customizing 231

D

data centers
  configuration 330
  managing failures for 491
  topology configuration 330
data grid security
  JSSE 584
  token manager 584
data grids
  configuration 261
database
  data preloading 29
  data preparation 29
  database synchronization techniques 31
  read-through cache 23
  side cache 23
  sparse and complete cache 22
  synchronization 31
  write-behind cache 26
  write-through cache 23
database integration
  configuration 419
DB2 562
deployment policies
  configuration 277
deprecated APIs 253, 255
directory conventions 63, 181
distributed cache 19
distributed deployment
  configuration 278
distributing changes
  peer JVMs 267
dynamic cache
  tuning 582
dynamic cache provider
  capacity planning 68
  configuring 397

E

Eclipse Equinox
  environment setup 223
embedded cache 17
enableXm property 340, 346
entity manager 81, 83
  creating an entity class 81
  entity relationship 83
  querying 89
  tutorial 81, 83

entity manager *(continued)*
  updating entries 88, 89
  using an index to update and remove entries 89
entity managerEntityManager
  creating an order entity schema 85
event listener 271
event-based validation 32
evictors
  configuration with XML 263
extension files 231
eXtreme IO 340
eXtreme memory 340, 346
eXtreme Scale overview 13
eXtremeIO
  configuration 340, 346
eXtremeMemory
  configuration 340, 346

F

failover
  configuration 321, 574
First steps console 204
fixes
  getting 614

G

getting started
  overview 1
grid authorization 583

H

Hibernate
  configuration 413
  configuration with XML 417
HTTP session manager
  configuration 364
  configuration with XML 382
  parameters for configuring 388
  WebSphere Application Server 365
  with WebSphere Virtual Enterprise 380
HTTP sessions
  splicer.properties file 393
Hyperic HQ 560

I

IBM Installation Factory
  build definition file 188
IBM Support Assistant 638
IBM Tivoli Monitoring 551
IBM Update Installer for WebSphere
  uninstalling
    CIP 192
IBM Update Installer for WebSphere Software 248
in-line cache 23
indexes
  data quality 34
  performance 34
initial configuration 228
installation
  customized installation package 195
  IBM Installation Factory for CIP 188
  IBM Installation Factory for IIP 188
  overview 171
  planning 59, 172
  REST data service 220
  silent 195, 217, 219
  silent response file 197
  stand-alone 184
  topologies 171
  troubleshooting 229, 627
  types 171
  uninstallation 230
  verification 227
  WebSphere Application Server 186
  WebSphere Application Server Network Deployment 186
  wizard 183
Installation Factory
  CIP
    maintenance 191
Installation Factory plug-in
  build definition file
    modify 194
  installing
    CIP 190
    IIP 193
installing
  maintenance 248
integrating with other servers 49
Introscope 557
invalidation 271

J

Java EE
  considerations 62, 180
Java Message Service (JMS)
  event listener 271
  peer-to-peer replication 267
Java Persistence API (JPA)
  cache plug-in
    introduction 401
  cache topology
    embedded 401
    embedded partitioned 401
    remote 401
  configuration
    overview 420
  time-based data updater
    configuration 422
Java SE
  considerations 60, 179
Java virtual machine 571
JDK
  considerations 60, 179
JMS
  peer-to-peer replication 267
JMX security access control
  authentication 595
  JAAS support 595
  secure transport 595
jobs 231
JPA cache plug-in
  troubleshooting 629
JVM 571

L

listener
  Java Message Service (JMS) 271
loaders
  database 28
  JPA 420
  troubleshooting 633
local cache
  peer replication 15
local deployments 262
local security
  enabling 605
locking
  configuration with XML 264
  configuring programmatically 264
  no 264
  optimistic 264
  pessimistic 264
locks
  timeout 266
log analysis
  custom 625
  overview 623
  running 624
  troubleshooting 627
log data 623
log element 267
log sequence 267
logs 618

M

managed beans 549
manageprofiles command 203
maxXmSize property 340, 346
MBean
  wsadmin 504, 548
MBeans
  accessing with security enabled 595
  administering with 503
  overview 549
  programmatically 505
migrating 235
migration 238
monitoring
  agent 551
  CA Wily Introscope 557
  csv files 525
  DB2 562
  Hyperic HQ 560
  overview 511
  Peformance Monitoring Infrastructure (PMI) 537
  statistics API 532
  statistics modules 530
  vendor tool overview 550
  with Tivoli Enterprise Monitoring Agent 551
multi-master data grid replication
  planning 35
multi-master replication
  configuration planning 41
  design planning 44
  planning 35
  planning for loaders 42

multimaster replication
  topologies 36
multiple data center configurations 632

N

network 565
network cards
  configuration 338
network ports
  planning for firewall
    definitions 51

O

object query
  index 75
  map schema 73
  primary key 73
  tutorial 73, 75, 77
Object Request Broker (ORB)
  configuration 341
  custom configuration 343
  orb.properties file 566
  properties 566
  stand-alone eXtreme Scale 342
  WebSphere Application Server 341
offline 488
online 488
OpenJPA
  cache plug-in
    configuration 407
  ObjectGrid XML files
    example 410
operating systems
  tuning 565
operational checklist 259
ORB
  configuration 341
  custom 343
  WebSphere Application Server 341
OSGi
  administering services 497
  configuring servers 454
  Eclipse Equinox environment 223
  installing bundles 225
  installing plug-ins 496
  starting servers 493
  tutorials
    configuration files 157
    configuring containers 162
    configuring servers 161
    finding service rankings 169
    installing bundles 159
    installing protocol buffers 162
    overview 154
    preparing to install bundles 155
    querying bundles 166
    querying service rankings 167
    running bundles 153
    running clients 164
    sample bundles 155
    setting up Eclipse to run clients 165
    starting bundles 160, 163
    starting clients 165
OSGi *(continued)*
  tutorials *(continued)*
    updating service rankings 169
    upgrading bundles 166
OSGi container
  Apache Aries Blueprint configuration 452
OSGi plug-ins
  administering with 500
  configuration 451

P

parallel transactions 67
passwords
  web console 514
peer-to-peer replication 267
per partition 67
Performance Monitoring Infrastructure
  enablement 537
  modules 541
  statistics retrieval 539
Performance Monitoring Infrastructure (PMI)
  monitoring 537
performance tuning 565
placement 486
plan
  installation 59, 172
planning 13, 565
  application deployment 13
  network settings 565
  operating systems 565
  operational checklist 259
PMI
  monitoring 537
ports
  configuration 334
  planning for firewall
    router 51
  stand-alone configuration 334
  WebSphere Application Server 338
post-installation tasks 228
preload 488
product overview
  product integration
    with WebSphere Application Server 106
Profile Management Tool
  custom definitions 232
  toolbox 231
  z/OS customization 231
Profile Management Tool plug-in
  overview 203
  profile augmentation 205
  profile creation 204
profiles
  augmentation 203
  augmentation with command 206
  augmentation with UI 205
  creation 203
  creation with command 206
  creation with UI 204
  non-root user 212
properties
  Object Request Broker (ORB) 566

Q

quiesce 488
quorums
  configuring 319
  override 491

R

real time
  stand-alone environment 578
  tuning garbage collection 577
  WebSphere Application Server 580
replication
  configuration with JMS 267
  JMS event listener 271
requirements
  hardware 59, 177
  software 59, 177
response file 217
response time
  real time
    stand-alone environment 578
  tuning garbage collection
    real time 577
Rest data service
  securing 598
REST data service
  Apache Tomcat
    deployment 442
    starting 444
  application servers
    configuration 433
  ATOM feeds
    configuration 446
  configuration
    overview 423
  data grid
    starting 431
  data model
    overview 425
  enablement
    overview 424
  installation 220
  Java client
    configuration 448
  retrieve and update data
    overview 427
  stand-alone data grid
    starting 430
  Visual Studio 2008 WCF client
    configuration 450
  WebSphere Application Server
    deployment 433
    starting 435
  WebSphere Application Server Community Edition
    deployment 436
    starting 440
rolling back
  product files
    GUI 245
    silent 247
routetable command 491
runtime files
  stand-alone 215
  WebSphere Application Server 213

S

secure servers
  REST data service 598
  starting 606
  stopping 606, 607
  WebSphere Application Server 607
Secure Sockets Layer (SSL)
  catalog servers 516
security
  authentication 56, 586
  authorization 56
  client security 604
  configuration 604
  integration 597
  integration with WebSphere Application Server 602
  introduction 597
  J2C client connections 609
  local 605
  overview 583
  plug-ins 605
  secure transport 56
  single sign-on (SSO) 586
  transport types 593
  troubleshooting 638
security profile 608
server properties
  enableXm 340, 346
  maxXmSize 340, 346
  xIOContainerTCPNonSecurePort 340
session manager
  persistence to data grid 369
  WebSphere Application Server 365, 380
session manager interoperability
  with WebSphere products 49
side cache
  database integration 23
silent installation 197
SIP
  session 375
  session management 375
sparse cache 22
SSL parameters 594
stand-alone
  Object Request Broker (ORB) 343
  REST 433
stand-alone servers
  starting 460
starting
  catalog servers 466
  catalog service 466
  container servers 466
  programmatically 476
  servers 460
  servers for REST data service 433
startOgServer 459, 464
  options 466
startXsServer 459
statistics
  enabling 529
  overview 511
  statistics API 532
stopOgServer 459
stopping
  programmatically 476
stopping servers 473
stopXsServer 459
Support 638

T

teardown command 474, 491
time-based data updater 422
timeoutrequest retry 357
topologies
  installation 172
  plan 14
trace
  options for configuring 621
  troubleshooting 619
trace data 623
transport 340
transports
  configuration 339
  eXtremeIO 340
  ORB 341
troubleshoot
  cache integration 628
  HTTP session 628
troubleshooting 611
  administration 631
  identifying problems, techniques for 611
  installation 229, 627
  trace 619
  XML configurations 635
troubleshooting and support
  getting fixes 614
    Fix Central 614
  IBM Support 615
  overview 611
  search known problems 613
  subscribing to IBM Support 617
  troubleshooting techniques 611
tuning
  garbage collection
    real time 577
  Java virtual machines 571
  network ports 51
  network settings 565
  operating systems 565
tutorial
  configuring catalog server security 114
tutorials 73
  accessing tutorial files 108, 130
  adding SSL properties 121, 146
  authorization 98
  catalog server security
    configuration 116
  client authentication 94, 95
  client authenticator 90
  client authorization 90
  client server security
    configuration 113
  configuration files 157
  configuring authentication
    in mixed environments 135
  configuring authorization
    for groups 125
  configuring catalog server security 138
  configuring client security 137

tutorials *(continued)*
  configuring container server security 141
  configuring eXtreme Scale containers 162
  configuring eXtreme Scale servers 161
  configuring for WebSphere Application Server 112
  configuring transport security 119, 145
  configuring transports
    inbound 120, 146
    outbound 120, 146
  configuring WebSphere Application Server 111
  creating entity classes 81
  enabling authorization 123, 149
    for users 124, 149
  endpoints secure communication 102
  finding service rankings 169
  forming entity manager relationships 83
  installing bundles 159
  installing eXtreme Scale bundles 160
  installing Google Protocol Buffers 162
  installing samples 143
  integrating product security
    with WebSphere Application Server 106
  integrating security
    in mixed environments 128
  monitoring data grids and maps
    with xscmd 127, 151
  object query 73, 75, 77
  ordering entity schemas 85
  OSGi
    configuration files 157
    configuring containers 162
    configuring servers 161
    finding service rankings 169
    installing bundles 159
    installing protocol buffers 162
    overview 154
    preparing to install bundles 155
    querying bundles 166
    querying service rankings 167
    running clients 164
    sample bundles 155
    setting up Eclipse to run clients 165
    starting bundles 153, 160, 163
    starting clients 165
    updating service rankings 169
    upgrading bundles 166
  overview
    starting servers and containers 154
  planning for mixed environments 130
  preparing to install eXtreme Scale bundles 155
  querying bundles 166
  querying local data grids 73
  querying service rankings 167

tutorials *(continued)*
  running sample clients
    in OSGi 164
  running samples 117, 121, 143, 147
  sample installation 117
  sample OSGi bundles 155
  setting up Eclipse
    for OSGi 165
  start OSGi bundles 163
  starting bundles 153
  starting client applications
    in the OSGi framework 165
  storing information in entities 81
  topology overview 108, 130
  unsecured example 90
  unsecured sample 91, 92
  updating and removing entities
    using queries 89
  updating and removing entries
    using an index 89
  updating bundles 166
  updating entries 88
  updating service rankings 169
  using JAAS authorization 122, 148
  WebSphere Application Server 107
  WebSphere Application Server
   configuration 134

U

uninstallation 230
  updates 245
updating 235
  product files 239
    command line 241
    GUI 239
    silent 243

W

wasprofile command 203
web console
  catalog server connection 516
  custom reports 525
  overview 514
  starting 514
  statistic descriptions 519
  statistics 518
WebSphere Application Server 248
  configuring with WebSphere eXtreme
   Scale 297
WebSphere Customization Toolbox
  custom definitions 232
  installing 231
  overview 231
WebSphere eXtreme Scale
  configuring with WebSphere
   Application Server 297
WebSphere Portal
  configuration 377
Wily Introscope 557
write-behind
  database integration 26
wsadmin
  MBean 504, 548
wsadmin tool
  catalog service domain 301
  MBeans 504, 548

X

xIOContainerTCPNonSecurePort
 property 340
XML configurations
  troubleshooting 635
xsadmin
  migration to xscmd 249
xscmd
  migration 249
  security profile 608
xscmd utility
  administration 482
  monitoring with 535
xsloganalyzer 624, 625

Z

zones
  container servers 287
  data center 281
  deployment policy descriptor XML
   file 289
  monitor 292
  over WANs 281
  routing 283
  shard placement 280
  striping across 281
  zone examples 281