Optimizing Operations with WebSphere Extended Deployment V6.1

- Implement dynamic operations and resource sharing
- Build a high-performance infrastructure
- Monitor performance and health

Carla Sadtler
Nitin Gaur
Jeff Tennenbaum

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (September 2007)

This edition applies to WebSphere Extended Deployment V6.1 Operations Optimization.
# Contents

<table>
<thead>
<tr>
<th>Notices</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trademarks</td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preface</th>
<th>xi</th>
</tr>
</thead>
<tbody>
<tr>
<td>The team that wrote this IBM Redbooks publication</td>
<td>xi</td>
</tr>
<tr>
<td>Become a published author</td>
<td>xii</td>
</tr>
<tr>
<td>Comments welcome</td>
<td>xiii</td>
</tr>
</tbody>
</table>

## Chapter 1. Introduction
- 1.1 WebSphere Extended Development V6.1   | 2 |
- 1.2 Operations Optimization                | 2 |
- 1.3 Data Grid                               | 5 |
- 1.4 Compute Grid                            | 6 |

## Chapter 2. Operations Optimization
- 2.1 Middleware server support              | 8 |
  - 2.1.1 Lifecycle support                   | 8 |
  - 2.1.2 Terminology                         | 13|
- 2.2 Dynamic operations                     | 13|
  - 2.2.1 Resource sharing                    | 14|
  - 2.2.2 Policy-based workload management    | 15|
  - 2.2.3 Autonomic managers                  | 16|
  - 2.2.4 Traffic shaping                     | 18|
- 2.3 Monitoring runtime operations          | 21|
- 2.4 Health management                      | 23|
- 2.5 Application edition management         | 24|

## Chapter 3. Sample topology
- 3.1 The sample topology                     | 25|
- 3.2 Sample topology set up summary          | 26|
  - 3.2.1 Basic cell structure                | 28|
  - 3.2.2 On demand routers                   | 28|
  - 3.2.3 Web servers                         | 29|
  - 3.2.4 Assisted lifecycle servers          | 29|
  - 3.2.5 PHP servers                         | 30|
  - 3.2.6 WebSphere application servers       | 31|

## Chapter 4. Planning for Operations Optimization
- 4.1 Evaluating the current environment       | 33|

© Copyright IBM Corp. 2007. All rights reserved.
Chapter 6. Working with other middleware servers
6.1 Working with assisted lifecycle servers
6.1.1 Installing the middleware agent
6.1.2 Starting the middleware agent
6.1.3 Checking the status of the middleware agent
6.1.4 Defining the server to the WebSphere configuration
6.1.5 Managing the server
6.1.6 Apache Tomcat servers
6.1.7 JBOSS servers
6.1.8 WebSphere Application Server Community Edition
6.2 Working with PHP servers
6.2.1 Install the middleware agent
6.2.2 Verify the PHP install location in the middleware descriptor

Chapter 7. Working with ODRs and autonomic managers
7.1 Introduction
7.2 Creating an on demand router
7.3 Configuring an ODR
7.3.1 On Demand Router settings
7.4 Proxy plug-in generation
7.4.1 Disable automatic plug-in generation on the Web servers
7.4.2 Create a script to propagate the plug-in
7.4.3 Configure the ODR proxy plug-in configuration policy
7.5 Caching in the ODR
7.6 Configuration of custom error pages
7.7 Logging and tracing ODRs
7.8 Routing in a multi-cell environment
7.8.1 Configuring multi-cell routing cells
7.9 Autonomic request flow manager (ARFM)
7.9.1 Configuring the ARFM
7.10 Application placement controller (APC)
7.10.1 Configuring the APC
7.10.2 Logging and tracing the APC
7.11 Dynamic Workload Manager

Chapter 8. Working with dynamic clusters
8.1 Dynamic clusters versus static clusters
8.2 Creating dynamic clusters
8.2.1 Dynamic cluster membership policy
8.2.2 Dynamic cluster properties
8.2.3 Lazy application start
8.2.4 Vertical stacking
10.5.3 Work class request classification rules .................. 232
10.5.4 Logging and tracing of routing rules ..................... 236
10.6 Test the service policy ..................................... 237
10.7 Routing policies ........................................... 240
  10.7.1 Creating a work class and routing rule for a routing policy ... 241
10.8 Verifying service policies and routing policies ............... 244

Chapter 11. Monitoring operations ................................. 245
11.1 Introduction .................................................. 246
11.2 Runtime operations views .................................... 247
  11.2.1 Operation alerts ....................................... 248
  11.2.2 On Demand Routers view ............................. 249
  11.2.3 Core Groups view ..................................... 250
  11.2.4 Core components view ............................... 252
  11.2.5 Nodes view ........................................... 254
11.3 Reports ....................................................... 256
  11.3.1 Charts ................................................ 257
  11.3.2 Customized chart options ............................. 258
  11.3.3 Building charts and chart groups ...................... 260
11.4 Visualization data service .................................... 266

Chapter 12. Health management .................................... 269
12.1 Introduction to health management ............................ 270
  12.1.1 Health monitoring tips ................................ 271
12.2 Health policies .............................................. 272
  12.2.1 Health conditions .................................... 272
  12.2.2 Actions to take ....................................... 274
  12.2.3 Reaction modes ........................................ 276
12.3 The health controller ......................................... 276
  12.3.1 Locating the health controller ......................... 279
12.4 Defining health policies ...................................... 280
  12.4.1 Creating custom actions ............................... 281
  12.4.2 Defining custom health policies ...................... 284
  12.4.3 Defining health policies using predefined conditions .. 291
  12.4.4 Default health policies ............................... 305
12.5 Monitoring runtime tasks ..................................... 305
  12.5.1 Configuring e-mail notification for tasks ............. 307

Appendix A. Roadmap to implementation .......................... 309
Creating the basic environment .................................... 310
  Install the WebSphere software ................................ 310
  Create the deployment manager ................................ 310
  Web servers .................................................... 310
  Create the on demand routers ................................ 311
Integrating WebSphere application servers ......................... 311
Integrating PHP servers .............................................. 312
Integrating other middleware servers .............................. 313

Related publications .................................................... 315
IBM Redbooks .......................................................... 315
Online resources ......................................................... 315
How to get IBM Redbooks ............................................. 316
Help from IBM .......................................................... 316

Index ................................................................. 317
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- AIX®
- CICS®
- DataPower®
- HACMP™
- i5/OS®
- IBM®
- Redbooks®
- Tivoli®
- WebSphere®
- z/OS®

The following terms are trademarks of other companies:

Enterprise JavaBeans, EJB, Java, JavaBeans, JDBC, JDK, JMX, JSP, JVM, J2EE, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Internet Explorer, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.
Preface

Operations Optimization enables application services to share a common pool of heterogeneous resources, resulting in the following:

- Better resource utilization
- Predictable and consistent attainment of application service levels
- Increased flexibility to respond to changing application demands
- Improved ability to manage complex and heterogeneous environments

This IBM® Redbooks® publication will help you understand how WebSphere® Extended Deployment Operations Optimization can improve the performance and reliability of your systems. It takes you through the steps required to implement Operations Optimization.

The team that wrote this IBM Redbooks publication

This Redbooks publication was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Carla Sadtler is a Consulting IT Specialist at the ITSO, Raleigh Center. She writes extensively about WebSphere and Patterns for e-business areas. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative. She holds a degree in mathematics from the University of North Carolina at Greensboro.

Nitin Gaur is Senior WebSphere IT Specialist with the IBM TechWorks Organization. Prior to teaming with TechWorks, Nitin spent several years with the WebSphere OEM team and AIX® support. In his seven years with IBM he has achieved an array of industry recognized certifications. He is also an active member of Austin Technical Vitality Council (TVC), an IBM Academy affiliate since 2002. As a technical leader he has been involved in writing various technical papers and presentations at industry conferences. The range of the topics span from software architectures to improvement of management processes. He has written extensively on enterprise problem determination methodology, WebSphere Application Server, WebSphere Extended Deployment best practices, and stack product integration with WebSphere Extended Deployment.
Prior to joining IBM, he was a graduate student at the University of Maryland. He holds a Masters degree in Management Information Systems and an MBA.

**Jeff Tennenbaum** is a WebSphere IT Specialist and a member of the Federal Strategic Integrated Technology Team. Before joining his current team, Jeff worked as a WebSphere IT Specialist for the Mid-Atlantic Technical Sales Team. Prior to becoming a WebSphere IT Specialist, he spent eight years as the level 2 support lead for CICS® on Distributed Platforms. Jeff has been with IBM since 1989.

Thanks to the following people for their contributions to this project:

Keith B Smith  
IBM US

Brian K Martin  
IBM US

Ann Black  
IBM US

Gabe Montero  
IBM US

Aditya A Desai  
IBM US

Jacquelle D Leggett  
IBM US

Lan Vuong  
IBM US

**Become a published author**

Join us for a two-to-six week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs and increase your productivity and marketability.
Find out more about the residency program, browse the residency index, and apply online at the following Web site:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online Contact us review Redbooks form found at:
  ibm.com/redbooks

- Send your comments in an email to:
  redbooks@us.ibm.com

- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400
Introduction

WebSphere Extended Deployment provides an IT infrastructure that dynamically and reliably adapts to changing business demands. WebSphere Extended Deployment extends the capabilities of WebSphere Application Server Network Deployment and other middleware to help you optimize the utilization and management of your deployments and enhance the quality of service of your business-critical applications.

This chapter takes a high-level look at the features of WebSphere Extended Deployment V6.1.

Topics included in this chapter are as follows:

- WebSphere Extended Development V6.1
- Operations Optimization
- Data Grid
- Compute Grid
1.1 WebSphere Extended Development V6.1

Prior to V6.1, WebSphere Extended Deployment was packaged as one comprehensive product offering customers dynamic operations, object grid, and business grid capabilities.

**New in V6.1:** In V6.1, you will find that previous capabilities still exist but are enhanced and available selectively. You can get all capabilities, or select a subset from the following packaging options:

- WebSphere Extended Deployment Operations Optimization
- WebSphere Extended Deployment Data Grid
- WebSphere Extended Deployment Compute Grid

1.2 Operations Optimization

The V6.0 enhancements of Operations Optimization provide dynamic operations, runtime operations monitoring, and extended management features, as well as new support for non-WebSphere application servers. The features of Operations Optimization include the following:

- Policy-based request prioritization and flow control for HTTP, SOAP, IIOP, and JMS traffic
- Dynamic feedback-based workload management
- Visualization features to help you manage complex environments
- Virtualization and resource sharing
- Application placement for optimal service goal achievement
- Health management of application servers
- Application Edition Management

A traditional WebSphere Application Server environment is static in nature. It is comprised of a set number of servers and clusters that serve specific applications. Resources are dedicated to applications to ensure that they operate at capacity during peak loads. Because different applications often have varying needs for resources (high at times, low at others), this resource dedication often leads to an excess of physical capacity in terms of CPU and memory during off-peak periods.

The characteristic of these static environments is that they are not making best use of the overall capacity and the configuration in terms of numbers of servers. Additionally these environments cannot quickly respond to unexpected changes
in workload. For example if an application has a dramatic increase in load, there may be insufficient capacity in the servers set aside for that application to meet the demand. However, there may be sufficient capacity in other servers running other applications that cannot be used.

With the dynamic operations features of Operations Optimization you can change the way a typical WebSphere environment is configured today, to one that has the following features:

- Improves the utilization of available resources such as CPU and memory
- Classifies and monitors the workload
- Provides a business-centric view of the workload and how it is performing
- Can respond in real time to changes in the workload mix (without human intervention if so desired), using business guidelines that the organization specified

WebSphere Extended Deployment implements a virtualized environment by creating pools of resources that can be shared among applications, thereby optimizing utilization and simplifying overall deployment. As resources are needed for expected (or unexpected) spikes in workload demand, resources can be allocated where they are needed most.

User-defined policies based on business requirements specify performance goals for each application. WebSphere Extended Deployment dynamically allocates resources to each application aiming to meet these performance goals.

Optimization of the computing resources that you already own might allow you to run more applications on the machines that you already have in place.

Following are the key elements and functions of Operations Optimization V6.1:

- **On Demand Router (ODR)**
  The ODR is an intelligent proxy that acts as the entry point for traffic coming into an Extended Development cell, performing request prioritization, flow control, and dynamic workload management for HTTP requests and SOAP over HTTP requests.

- **Dynamic application placement**
  The dynamic application placement feature uses dynamic clusters to virtualize the application deployment targets, enabling provisioning of resources to help meet your stated performance goals.

  Each node within a dynamic cluster has an instance of an application server running that cluster’s applications that can be started dynamically as traffic for that application increases.
**New in V6.1:** New features for dynamic clusters include expression-based
dynamic cluster membership. Dynamic clusters are no longer dependent on
node groups. You can now specify isolation policies for dynamic clusters,
preventing applications from two dynamic clusters from executing on the
same node. Servers as well as nodes can now be placed in maintenance
mode.

- **Autonomic managers**
  Autonomic managers make decisions for the environment, including health
  management, traffic shaping, and application placement.

- **Traffic shaping features**
  Traffic shaping features classify requests and manage the flow of requests
  into the application servers. HTTP, SOAP, and SIP requests are classified
  and controlled in the ODR. JMS and IIOP traffic is classified and controlled at
  the application server level.

- **Health management**
  The health monitoring and management subsystem continuously monitors
  the operation of servers against user-defined health policies to detect
  functional degradation that is related to user application malfunctions.

  **New in V6.1:** Health management was enhanced by allowing you to define
  custom health conditions. New predefined health actions are available and
  custom actions can be created.

- **Runtime operation monitoring**
  With the new complexities of dynamic operations, the need arises for tools
  that extend monitoring and manageability capabilities. The visualization
  components of WebSphere Extended Deployment enhance the
  administrative console to provide live data on the performance and health
  characteristics of the entire cell.

  **New in V6.1:** Visualization changed significantly in V6.1, particularly in
  custom charting.

- **Application edition management**
  Loss of service to users means loss of business to you. The application
  edition management feature helps you ensure that the users of your
  application experience no loss of service when you install an application
  update in your environment.
1.3 Data Grid

The Data Grid option provides the high-end caching and transaction partitioning capabilities, known better as ObjectGrid, and the partitioning facility respectively of V6.0. There are also enhancements designed to improve performance across a wide range of application scenarios.

The data-intensive workload extenders of Data Grid improve the interaction between application services and underlying data sources, resulting in the following:

- Dramatically increased application performance
- Improved application scalability
- Increased developer efficiency

New in V6.1: ObjectGrid is enhanced to provide performance improvements across a wide range of application scenarios. The number of Java™ Virtual Machines (JVM™) and the size of data sets supported are significantly increased. Query capabilities were added allowing for parallel operations across the ObjectGrid configurations. Applications using different schemas for the underlying ObjectGrid information can execute concurrently, improving application availability.
1.4 Compute Grid

Compute Grid (previously known as Business Grid) features provide flexible support for mixed application types. Features of Compute Grid include the following:

- Batch workload services
- Compute-intensive workload services
- Long-running workload scheduler

Long running workload extenders support the scheduling and execution of long-running workloads in a standards-based WebSphere environment, resulting in improved application consistency and maintainability. It provides common development, administration, and management models for multiple workload types.

New in V6.1:

- Transactional batch programs are implemented as simple Plain Old Java Objects (POJO)s and packaged into EAR files for deployment.
- Compute Grid can be composed of both WebSphere Application Servers and native execution endpoints.
- Integration with external workload schedulers, such as Tivoli® Workload Scheduler (TWS). A Compute Grid integration layer enables TWS, and similar products, to dispatch and monitor Compute Grid activities. Third party workload scheduler products can also use this integration layer.
- Job management console WebSphere Extended Deployment Version 6.1 provides a new Web-based application for managing jobs. It is called the job management console.
- New administrator scripting interfaces to manage WebSphere Extended Deployment outside of the administrative console.
Operations Optimization

The focus of this book is the WebSphere Extended Deployment Operations Optimization package. This chapter provides an overview of Operations Optimization.

In this chapter, we discuss the following topics:

- Middleware server support
- Dynamic operations
- Monitoring runtime operations
- Health management
- Application edition management
2.1 Middleware server support

WebSphere Extended Deployment V6.1 provides support for a range of middleware servers. This support includes the following servers:

- WebSphere Application Server
- PHP server
- Apache HTTP Server
- Apache Tomcat server
- Apache Geronimo server
- JBoss server
- BEA WebLogic Server
- WebSphere Application Server Community Edition
- Custom HTTP servers

Prior to V6.1, middleware servers other than WebSphere application servers were supported and defined as HTTP endpoints, represented by port and host name. No administration for the servers was provided, there was minimal performance monitoring, and no health management was available.

With WebSphere Extended Deployment V6.1, a new middleware agent is provided that runs on these servers to provide enhanced support.

2.1.1 Lifecycle support

Middleware servers can be categorized along the lines of the lifecycle support provided in WebSphere Extended Development.

- Full lifecycle
- Assisted lifecycle
- Generic lifecycle

The type of server and its support is shown in Table 2-1.

<table>
<thead>
<tr>
<th>Supported Versions / Releases</th>
<th>Lifecycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere application server</td>
<td>Version 6.0.2, Version 6.1</td>
</tr>
<tr>
<td>PHP</td>
<td>Version 4.x, Version 5.x</td>
</tr>
</tbody>
</table>
For the current support information, see the following Web site:

- Supported middleware server types
  

### Full lifecycle support for servers

Full lifecycle support includes control of all aspects of an application server lifecycle. This includes creating and deleting server instances, managing and deploying applications, and exploiting all Extended Deployment capabilities with regard to application deployment environment.

With full lifecycle support you can do the following:

- Create and remove server instances.
- Use dynamic clusters. You can set policies that allow the dynamic cluster to expand automatically when new nodes are added to the cell.
- Use administrative utilities to manage the configuration and view runtime logs.
- Use provided operational control features to manage servers.

<table>
<thead>
<tr>
<th>Server Type</th>
<th>Supported Versions / Releases</th>
<th>Lifecycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere application server</td>
<td>V5.x</td>
<td>Assisted</td>
</tr>
</tbody>
</table>
| Apache Tomcat | Version 4.1.x  
Version 5.0x  
Version 5.5.x | Assisted |
| BEA WebLogic Server | Version 8.x  
Version 9.x | Assisted |
| WebSphere Application Server Community Edition | Version 1.x | Assisted |
| JBoss server | Version 4.0.x | Assisted |
| Apache Geronimo | Version 1.0  
Version 1.1 | Assisted |
| Apache HTTP Server | Version 1.3  
Version 2.0  
Version 2.2 | Assisted |
| Custom HTTP servers | | Generic |
► Deploy applications.
► Application Edition Management for interruption-free rollout.
► Prioritize and route traffic to the servers.
► Monitor and view health and performance data.

Following is a typical installation scenario using an established deployment manager and the administrative console:

1. Install the WebSphere or PHP software.
2. Create the node.
   – In the case of a WebSphere installation, create a custom profile and federate it to the cell.
   – In the case of a PHP installation, use the WebSphere centralized installation manager to install the middleware agent. This process also creates the node and federates it to the cell.
3. Define a dynamic cluster (which creates the servers automatically).
4. Deploy the applications and map them to their dynamic cluster.
5. Define service policies and associate them with the applications.

**Assisted lifecycle support**
Assisted lifecycle support provides the means to manage and perform traffic shaping for middleware servers. Application servers are created and applications are deployed to them external to Extended Deployment. Representations of the servers and applications are manually defined to the Extended Deployment cell.

Out-of-the-box templates are provided for use in creating representations of the servers in the cell configuration. The default template for each server type contains information specific to the product, such as default start and stop commands, port information, core configuration, log file location, and so on. Additionally, any administrator can define custom operations to execute on those servers. Extended Deployment also provides the capability to view remote log files and to view and edit remote configuration files.

Creating representations of the servers in Extended Deployment enables you to manage these servers. You can create dynamic clusters with these servers as members. Creating representations of the applications allows you to define service policies for use by the ODR and by the autonomic managers for dynamic application placement.
With assisted lifecycle support you can do the following:

- Create representations of existing servers using server-specific templates.
- Use autonomically operated dynamic clusters. Servers are added to dynamic clusters manually.
- Use administrative utilities to manage the external configuration and view runtime logs.
- Configure operations (start / stop) that can be performed to manage servers.
- Create representations of existing applications.
- Prioritize and route traffic to the servers.
- Monitor and view health and performance data.

Following is a typical installation scenario:

1. Install the middleware server software (for example, WebSphere Application Server Community Edition) on the middleware server host, and create the new server configuration. This is done independent of WebSphere Extended Deployment.
2. Deploy applications to the new server using the mechanisms provided by the middleware server.
3. Use the WebSphere administrative console to define the middleware host as a deployment target to the WebSphere cell.
4. Use the centralized installation manager to install the middleware agent on the middleware system. This process installs the WebSphere Extended Deployment agent software, creates a node, and synchronizes the node to the cell.
5. Use the administrative console to define the existing server to the cell using the default template.
6. Make any adjustments to the server definition (for example, if the default template had a different installation path than what was actually used).
7. Define a dynamic cluster.
8. Create representations of deployed applications and map them to the appropriate dynamic cluster.
9. Define and associate service policies to the applications.

**Generic lifecycle support**

Generic lifecycle support includes all the other servers. This type of support is for HTTP end points and involves manual configuration to add any administrative capability on the Extended Deployment console. However, if the administrator provides information about how to start and stop the servers as well as port
connector information, Extended Deployment can take operational action on the servers and perform traffic shaping. If lists of configuration file paths and log file paths are provided, then an administrator can edit or view those files from a central point.

Generic lifecycle support allows you to do the following:
- Create representations of existing servers using a generic template.
- Manually add nodes to dynamic clusters.
- Manually define server operations and external logs that can be managed from WebSphere.
- Create representations of existing applications.
- Prioritize and route traffic to the servers.
- Monitor and view health and performance data.

Following is a typical installation scenario for an unmanaged server using the deployment manager administrative console:
1. Install the middleware server software on the host, and create the new server configuration.
2. Deploy applications to the new server using the mechanisms provided by the middleware server.
3. Use the WebSphere administrative console to define the middleware host as a deployment target to the WebSphere cell.
4. Use the centralized installation manager to install the middleware agent on the middleware system. This process installs the software, creates a node, and federates the node to the cell.
5. Use the administrative console to define the existing server to the cell using the custom HTTP server template.
6. Define a dynamic cluster.
7. Create representations of deployed applications and map them to the appropriate dynamic cluster.
8. Define and associate service policies to the applications.
2.1.2 Terminology

We use the following terminology in this book to distinguish between the server types:

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere application server</td>
<td>A WebSphere application server can be hosted on a Network Deployment node or a WebSphere Extended Deployment node.</td>
</tr>
<tr>
<td>All application servers</td>
<td>All servers defined to the WebSphere cell (full, assisted, and generic lifecycle).</td>
</tr>
<tr>
<td>Full lifecycle server</td>
<td>WebSphere application servers and PHP servers.</td>
</tr>
<tr>
<td>Generic lifecycle server</td>
<td>Custom HTTP.</td>
</tr>
</tbody>
</table>

2.2 Dynamic operations

WebSphere Extended Deployment implements dynamic operations for middleware servers with the goal of providing a consistent quality of service. It accomplishes this with the following key capabilities:

- Resource sharing
  Hardware resources in a traditional WebSphere Application Server environment are planned to handle peak load. This often results in the under-utilization of resources in off-peak times.

WebSphere Extended Deployment allows you to pool resources. As resources are needed for expected (or unexpected) spikes in workload demand, application resources can be allocated to where they are needed most.

The key component of resource sharing is the dynamic cluster.
Policy-based workload management

Service policies can be defined and associated with applications to reflect operational goals based on business needs. Policies define service levels for the application requests. As requests arrive, they are classified based on these policies and treated accordingly. High priority applications receive preferential treatment, receiving the resources they need over lower priority application requests.

Traffic shaping

Traffic shaping is the process of classifying incoming requests based on policies and managing the distribution of requests among application servers. The On Demand Router (ODR) is the entry point to the WebSphere Extended Deployment environment for HTTP, SOAP, and SIP requests. It receives user requests, classifies them, and queues them for dispatch according to the service policies.

Traffic shaping also occurs for JMS and IIOP requests. These requests are classified and managed at the application server level.

2.2.1 Resource sharing

In traditional WebSphere Application Server environments, applications are deployed directly to servers or a static cluster of servers running on specific hardware systems (nodes). When the server starts, so does the application. A peak load on one system cannot take advantage of resources sitting idle on another system.

With WebSphere Extended Deployment, applications are mapped to dynamic clusters spread across hardware pools. Each node in the dynamic cluster has an instance of an application server running that cluster’s applications. The server can be started to accommodate the demand for that application. This is referred to as dynamic application placement.

Dynamic clusters

With WebSphere Extended Deployment, applications are deployed to dynamic clusters. A dynamic cluster is similar to the traditional static cluster defined in Network Deployment; however, dynamic clusters expand to respond to workload demand and user-defined service goals and policies.

Dynamic cluster membership

When you define a dynamic cluster, you define nodes that host application servers within that dynamic cluster. The member nodes can be specifically designated, or, in the case of dynamic clusters with application servers with full lifecycle support, can be defined by rules. When membership is rules-based, any
new nodes added to the cell that meet the rule criteria are automatically added to the dynamic cluster. Application servers are automatically defined on the membership nodes according to properties set in the dynamic cluster.

**Dynamic cluster isolation**

The dynamic cluster isolation settings allow you to specify whether server instances of multiple dynamic clusters can coexist on a node. You can use these settings to ensure that dynamic clusters with specific applications are restricted from running server instances on the same node at the same time. This isolation of applications is done for security, performance, or stability reasons.

**Lazy application start**

The lazy application start feature optimizes server resource allocation during times of inactivity. A server (dynamic cluster member) associated with a lazy application is not started until a request for that application is received by the ODR.

**Vertical stacking**

Although Extended Deployment favors a horizontal approach to scaling, vertical stacking is also supported. This allows multiple instances of an application to be started on the same physical node in order to enable more throughput on that application and better consumption of the resources of the node. Vertical stacking should only be used when a single application server instance cannot consume the full CPU resources of a node.

### 2.2.2 Policy-based workload management

With WebSphere Extended Deployment, you can differentiate application service levels according to your business requirements. User requests are classified, prioritized, queued, and routed to servers based on application operational policies that are tied to performance goals. Application performance is optimized according to these policies that reflect service level goals and relative importance to the organization. Simply put, you can state what applications are important to you, and these applications will get the highest priority access to your WebSphere resources at the right time. This can help you ensure that your business-critical transactions get the best quality of service.

**Service policies**

A service policy gives the capability to designate the performance goal and business importance of different request types. This allows an enterprise’s performance to degrade in a controlled manner in periods of over-utilization.
A service policy consists of two key items: goal and relative importance. The goal defines how incoming work is evaluated and managed in order to ensure and detect if work is meeting its assigned service policy levels. The importance is used in times of resource contention in order to identify the most important work in the system and give it the priority.

**Service goal types**

There are four performance goal types: discretionary, average response time, percentile response time, and queue wait time (for long running applications).

Many unfavorable circumstances, including over-utilization, constrained computing resources, and error conditions can impact your ability to meet performance goals. When response time increases, actions must be taken to correct the situation. WebSphere Extended Deployment operational policy facilities will often be sufficient to meet your performance objectives and if not, can act as an early warning mechanism.

**Routing policies and work classes**

Routing policies define how work should be routed to either an application edition, including the base edition, or a generic server cluster. A generic server cluster is a group of non-WebSphere HTTP end points that can access traffic from the ODR. Routing policies are defined at the ODR for the generic server clusters, or the application level for an application edition.

> **Note:** Generic server clusters are deprecated in V6.1 in favor of using middleware servers with the middleware agent for an increased level of support and better performance management.

Routing policies are comprised of work classes and routing rules.

Work classes define how work for an application or a generic server cluster is classified. Work classes allow you to define patterns (such as an HTTP pattern, a JMS pattern, or an IIOP pattern) and then based on that pattern associate a service policy or routing rule with that pattern.

### 2.2.3 Autonomic managers

Autonomic managers monitor performance and health statistics through a series of sensors and turn various internal control knobs to optimize system performance and perform traffic shaping. This section describes each of the autonomic managers in a WebSphere Extended Deployment topology.
**Autonomic Request Flow Manager**
The Autonomic Request Flow Manager (ARFM) classifies incoming requests and monitors the performance of service classes on a continual basis. It adjusts the queue dispatching weights in the ODR and modifies the weights to align flow control with performance goals. The following three components work together to prioritize incoming requests:

- **ARFM gateway**
  ARFM gateways run in the ODRs for HTTP traffic and in the back-end servers for Internet Inter-ORB Protocol (IIOP) and JMS traffic. The gateways classify and queue incoming requests, dynamically creating queues as service policies appear in the request flow.

- **ARFM controller**
  The ARFM controller monitors request flow, execution times, and application placements and sets dispatch weights accordingly. It provides direction to the gateways and the application placement controller.

  The controller sends information about the amount of traffic and how well service goals are met to the Application Placement Controller (APC).

- **Work profiler**
  The work profiler monitors the CPU utilization of each node and sends this information also to the APC. The work profiler continually estimates the computational requirements of the various kinds of requests, based on observations of the system in operation.

There is a single work profiler and ARFM controller per cell. Each runs on a node agent.

**Application Placement Controller (APC)**
The primary responsibility of the APC is the management of an application’s location. It controls the location and cardinality of active instances in a given dynamic cluster. As demand increases dynamic cluster members are started to meet defined service goals.

A single application placement controller exists in the cell. The APC is hosted in a node agent process and is managed by the high-availability manager. If the node agent hosting the APC fails, it is started on another node agent in the cell.

**On Demand Configuration manager (ODC)**
The ODC maintains cell topology information and keeps the ARFM aware of its environment. It tracks updates in cell topology and state, including the following:

- Applications installed and removed
Optimizing Operations with WebSphere Extended Deployment V6.1

- Servers started and stopped
- Nodes added and removed
- Classification updates

**Dynamic workload manager**

The dynamic workload manager (DWLM) handles load balancing of work across an enterprise back end by maintaining a table of servers to which it is delivering work. In this table, each server is dynamically assigned a weight corresponding to its relative capacity to perform work. In WebSphere Extended Deployment terminology, the dynamic workload manager maintains a list of active server instances for each dynamic cluster and assigns each a routing weight according to observed performance trends. Requests are then routed to candidate server instances to balance workloads on the nodes within a dynamic cluster based on a weighted least outstanding requests algorithm.

### 2.2.4 Traffic shaping

Traffic shaping is done at different entry points depending on the type of request. For HTTP, SOAP, and SIP requests, traffic shaping occurs in the ODR. For IIOP and JMS requests, traffic shaping occurs at the application server.

Autonomic managers play a key role in traffic shaping. They perform the classification and prioritization of requests and manage the environment to balance the workload.

**HTTP, SOAP, and SIP traffic shaping in the ODR**

The ODR represents the entry point into a WebSphere Extended Deployment topology and controls the flow of requests into the back-end servers. As requests enter the ODR they are queued for dispatch to back-end servers based on request concurrency, operational policy, service policy weights, and load balancing. The ODR manages the queue lengths and dispatch rates for consistent quality of service.

**Class-based queuing of work**

As soon as a request is mapped to a transaction class, it is linked to that transaction class’ service policy. Each service policy is assigned a separate queue. The request is then placed into the queue that corresponds to its service policy. An autonomic manager occasionally adjusts dispatching weights of each queue to achieve performance goals based on measurements of arrival rates and service times.
**Limiting concurrency**

To protect against the prospect of an overloaded enterprise, the ODR limits the number of requests being serviced concurrently by an application server. Concurrency control also optimizes throughput. For example, you might get a higher throughput by executing 25 requests concurrently than if you executed 50 requests concurrently. When these limits are reached, requests begin queuing up in the ODR until demand subsides. If queues reach a (configurable) maximum length, subsequent requests can result in a message returned to the client indicating that the server is busy. The ARFM provides configurable options to limit concurrency.

WebSphere Extended Deployment autonomically computes a concurrency limit that normally takes effect before you would reach the container configured limit. Concurrency limits are computed based on configured maximum thread pool sizes of the Web container, EJB™ container, and J2C container on the target servers. Thread pool values for the Web and EJB containers can by found in the administrative console by selecting **Servers** → **Application servers** → **server_name** → **Thread Pools**. For J2C, these are the maximum and minimum connection settings on the resources. The container limits should be high enough that Extended Deployment’s concurrency is the limit that matters.

**Classification of requests**

HTTP requests can be classified by parameters that include the following:

- Client host name or IP address
- Cookie name
- Group ID of the request sender
- HTTP method
- Header name and value
- MIME type
- Listening port
- Communication protocol (HTTP, HTTPS, SOAP, and SOAPS)
- Server host name or IP address
- Date and time that a request must be honored
- User ID of request sender
- Virtual host

SOAP requests can be classified by the same criteria as HTTP requests, and in addition, the following parameters can be used:

- Web service
- Web service operation
- XPath expressions
For a full list of classification parameters, see the following:

- **Routing and service policies for work classes**
  

**IIOP traffic shaping**

Incoming IIOP traffic bypasses the ODR. It is intercepted and queued by ARFM gateways that run on the individual servers hosting the IIOP components. The gateways classify the requests and perform request flow prioritization to achieve balanced performance results according to service policies and load.

Only IIOP requests from standalone Enterprise JavaBeans™ (EJB) clients are handled by Extended Deployment IIOP request flow prioritization. EJB calls that originate from servlets are not re-prioritized, but are executed in the context of a flow that has already had flow control applied.

IIOP requests can be classified based on several attributes, including application name, server name, EJB name, and method name. To view a full list of the attributes that all requests can be classified, review Table 2 in the following WebSphere Extended Deployment Information Center article:

- **Routing and service policies for work classes**
  

**JMS traffic shaping**

WebSphere Extended Deployment provides added value for JMS applications that use the WebSphere Application Server default messaging provider. JMS traffic bypasses the ODR and is managed by the ARFM gateway running in the application server that hosts the JMS application.

Work classes for JMS are created in the same manner as work classes for HTTP requests. You can filter and classify a request based on the message-driven bean, the bus name, and the available destination (queue or topic).

The ARFM component runs on the application server hosting the JMS application and controls the flow to the application. In order to meet defined service policies, WebSphere Extended Deployment can temporarily suspend delivery of a JMS message to a message-driven bean or an application waiting to receive a message, while a destination with a higher priority may continue to handle JMS requests.
2.3 Monitoring runtime operations

WebSphere Extended Deployment provides you with the ability to monitor your runtime operations from the administrative console.

Real-time reporting shows you alerts indicating anomalies with the runtime environment. You can view the status of the cell based on ODRs, core groups, core components (autonomic managers), and nodes.

Figure 2-1 shows an example of the runtime operation view for nodes. You can view all the nodes in the cell or use the filtering function to narrow down the scope. In one view, you can see the status of each node: if it is in maintenance mode, the type of node, and the WebSphere packages installed on the node.

![Figure 2-1 Runtime operations view of nodes](image)

Chapter 2. Operations Optimization 21
You also have the ability to build and save customized reports for dynamic viewing (Figure 2-2). To build these reports, you select the type of component you want to monitor, for example, ODRs, application servers, nodes, dynamic clusters, or service policies. Next you select a specific instance of the component type to monitor. Then you select the data metric to use from a wide selection, including metrics such as average response time, throughput, or CPU utilization. These reports can help you ensure that you are meeting service level goals and can help you identify potential problems in the early stages.

Figure 2-2  Dynamic charting
The visualization data service can be configured to save operational data to text files that you can reuse with other charting programs.

Visualization data can also be used for capacity planning and charge back purposes. This is described in Best Practices for Implementing WebSphere Extended Deployment, SG24-7343, Chapter 2. Application hosting and chargeback.

2.4 Health management

WebSphere Extended Deployment provides a health monitoring and management subsystem that continuously monitors the operation of servers to detect functional degradation that is related to user application malfunctions. Health management provides a policy-driven approach to monitoring the application server environment and taking action when certain criteria are discovered.

With health management, you define health policies designed to identify potential problems, the action to take when the event occurs, and how the actions will be taken. You can define a health policy to monitor your system at the cell, dynamic cluster, static cluster, or application server/node level.

WebSphere Extended Deployment comes with predefined conditions, such as excessive memory usage or request / response times for use in building a health policy. In V6.1, you can build a custom policy using a custom expression to define the condition. Custom conditions are built based on metrics gathered at the ODR or server.

Actions to be taken when a monitored condition is detected are designed to bypass the problem and help in diagnosis. Pre-defined actions include restarting a server, putting a server into maintenance mode, and taking dumps for use in diagnosing the problem. You can also define a custom action to be taken. Actions can be taken automatically, or you can have them occur in supervised mode. Supervised mode requires an operator to allow the action.

Health monitoring can help you with unexpected issues and is particularly useful when you anticipate problems in your environment. It can help you bypass problems that could potentially disrupt operations and affect performance.
2.5 Application edition management

The application edition manager of WebSphere Extended Deployment helps you manage interruption-free production application deployments. Loss of service to users means loss of business to you. Using the application edition manager, you can ensure that the users of your application experience no loss of service when you install an application update in your environment.

The application edition manager also provides an application versioning model that supports multiple deployments of the same application in a WebSphere Extended Deployment cell. You can choose which edition to activate on a cluster, enabling you to either roll out an application update or revert to a previous level.

The application edition manager interacts with the ODR, dynamic workload manager, and application placement manager. This integration assures predictable application behavior when you apply application updates, ensuring a smooth transition from one application edition to another while the system continues to manage your application performance goals. The application edition manager’s edition control center provides control over the application update and rollout process, including edition activation across the application servers to which your application is deployed. Scripting APIs enable the integration of edition management functions with automated application deployment.
Sample topology

For the purpose of this IBM Redbooks publication, we set up a sample topology that allows us to demonstrate the various functions and features of IBM WebSphere Extended Deployment V6.1.

This chapter describes the sample topology used throughout this book.
3.1 The sample topology

Figure 3-1 illustrates the layout of the topology used in this publication. Each node is labeled with its node or dynamic cluster name in the cell.

Note: The Load Balancer was not implemented as part of this project. You can use WebSphere Network Deployment Edge Component’s Load Balancer to distribute workload between the two Web servers. In a real-world environment, you must make the Load Balancer highly available because this is the entry point into your environment! Refer to chapter 5 of *IBM WebSphere V6 Scalability and Performance Handbook*, SG24-6392 for more information about how to configure a highly available Load Balancer.
The sample topology consists of the following:

- Two Web server systems running IBM HTTP Server V6.1 and the Web server plug-in.
  
The Web server systems are configured as unmanaged nodes in our WebSphere cell, which is called XDcell1.

- Two On Demand Routers that receive requests from the Web servers and distribute them to the application servers based on routing and service policies and dynamic weights.

- Three systems running WebSphere Extended Deployment V6.1.
  
  Each system has a node agent and one or more application servers. The application servers belong to one dynamic cluster, named XDCuster1.

- Two systems running JBoss 4.0.5.
  
  Each system has a middleware agent and one or more JBoss servers. The servers belong to one dynamic cluster named JBoss_DC.

- Two systems running Tomcat 5.5.23.
  
  Each system has a middleware agent and one or more Tomcat servers. The servers belong to one dynamic cluster, named Tomcat_DC.

- Two systems running WebSphere Application Server Community Edition 1.1.01.
  
  Each system has a middleware agent and one or more Community Edition servers. The servers belong to one dynamic cluster, named WASCE_DC.

- Two systems running PHP 5.2.2.
  
  Each system has a middleware agent and one or more PHP servers. The servers belong to one dynamic cluster, named PHP_DC.

- A highly available deployment manager
  
  Two systems host the deployment manager, one of them is active and the other one is a backup or standby deployment manager. The deployment manager profile is located on a shared file system such as the IBM Storage Area Network File System (SAN FS) or Network File System Version 4.

**Note:** In a real-world environment, your database server should have some kind of backup to provide a highly available database. This can, for example, be achieved by using clustering software such as IBM HACMP™ or IBM Tivoli System Automation.
3.2 Sample topology set up summary

The following steps were executed to set up the sample topology.

3.2.1 Basic cell structure

The following steps created the basic cell structure.

1. WebSphere Application Server Network Deployment V6.1 and the required fixpacks and interim fixes were installed on the deployment manager system, XDdmgr1node.

A list of the required fixpacks and interim fixes can be found at the following Web address:

http://www-1.ibm.com/support/docview.wss?rs=3023&uid=swg27009516

Selected the appropriate operating system.

2. WebSphere Extended Deployment Operations Optimization V6.1 was installed on XDdmgr1node.

3. The deployment manager profile for XDcell1 was created on XDdmgr1node. The node name for the new profile was XDdmgr1node. After the profile was created, the deployment manager was started.

The steps required to create the deployment manager are discussed in Chapter 5, “Creating the basic environment” on page 63.

As part of this project we did not implement a second deployment manager for high availability, though in a production environment it is recommended that you do so. The use of two deployment managers in a high availability configuration requires a suitable shared file system to be in place. You can obtain more about using a high-availability deployment manager setup in 5.2, “High availability deployment manager” on page 69.

3.2.2 On demand routers

The following steps created the two on demand routers. These steps were performed on the XDodr1 and XDodr2 nodes.

1. WebSphere Application Server Network Deployment V6.1 and the required fixpacks and interim fixes were installed.

2. WebSphere Extended Deployment Operations Optimization V6.1 was installed.

3. A custom profile was created on each system and was federated to the XDcell1 cell as part of the profile creation.
4. The node agents on each system were started.
5. An On Demand Router server was created on each node using the administrative console.
6. The ODR proxy plug-in generation parameters were configured so the Web server plug-in generated would send traffic from the plug-in to the ODRs instead of to the application servers. A script was defined to automatically propagate the plug-in after each generation.

The steps required to create an ODR are discussed in Chapter 7, “Working with ODRs and autonomic managers” on page 115.

### 3.2.3 Web servers

The two Web servers on nodes XDIHS1 and XDIHS2 were created using the following steps. These steps were performed for each system.
1. IBM HTTP Server V6.1 was installed.
2. The Web server plug-in for the IBM HTTP Server was installed. The Web server plug-ins are included with WebSphere Application Server Network Deployment.
3. The configuration script created by the plug-in installation was copied to the XDdmg1node system and executed to create the unmanaged node and Web server definitions for the Web server.
4. Automatic generation and propagation was disabled for the default Web server plug-in on the Web server nodes.

### 3.2.4 Assisted lifecycle servers

The sample topology used several server types to illustrate the use of servers with an assisted lifecycle. The process was the same for each server type.
1. The software for each server type was installed and configured on the appropriate system. This included the WebSphere Application Server Community Edition, JBoss, and Apache Tomcat.
2. Each system was defined as a deployment target to WebSphere using the administrative console.
3. The middleware agent was installed on each system and the new nodes federated to the cell. The installation was done from the administrative console for the WebSphere cell using the centralized installation manager.
4. Middleware server definitions were created to represent the servers in the WebSphere administrative console.
5. Sample applications were installed on each server, using deployment techniques native to each server type.

6. Dynamic clusters were created. Each dynamic cluster contained all the servers of a specific type.

7. Middleware application definitions were created to represent these applications in the WebSphere administrative console. The dynamic cluster is the deployment target for these applications.

Table 3-1 shows the resulting configuration for the middleware servers.

<table>
<thead>
<tr>
<th>Dynamic cluster</th>
<th>Nodes</th>
<th>Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tomcat_DC</td>
<td>XDTomcat1</td>
<td>XDTomcat1</td>
</tr>
<tr>
<td></td>
<td>XDTomcat2</td>
<td>XDTomcat2</td>
</tr>
<tr>
<td>JBoss_DC</td>
<td>XDJBOSS1</td>
<td>default</td>
</tr>
<tr>
<td></td>
<td>XDJBOSS2</td>
<td>default</td>
</tr>
<tr>
<td>WASCE_DC</td>
<td>XDWASCE1</td>
<td>WASCE1</td>
</tr>
<tr>
<td></td>
<td>XDWASCE2</td>
<td>WASCE2</td>
</tr>
</tbody>
</table>

### 3.2.5 PHP servers

Two PHP servers were used in the topology.

1. The PHP software was installed and configured on the appropriate system.

2. Each system was defined as a deployment target to WebSphere using the administrative console.

3. The middleware agent was installed on each system and the new nodes federated to the cell. The installation was done from the administrative console for the WebSphere cell using the centralized installation manager.

4. A dynamic cluster was created using membership rules that included the PHP nodes as members of the cluster. Application servers were defined automatically on the servers for the cluster.

5. Sample applications were deployed to the dynamic cluster.

Table 3-2 on page 31 shows the resulting configuration for PHP servers.
Table 3-2  PHP servers and dynamic clusters

<table>
<thead>
<tr>
<th>Dynamic cluster</th>
<th>Nodes</th>
<th>Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP_DC</td>
<td>PHP_DC_XDPHP1_1</td>
<td>XDPHP1_1</td>
</tr>
<tr>
<td></td>
<td>PHP_DC_XDPHP2</td>
<td>XDPHP2</td>
</tr>
</tbody>
</table>

3.2.6 WebSphere application servers

The following steps were taken to create application servers on the Extended Deployment nodes, XDnode1, XDnode2, and XDnode3.

1. WebSphere Application Server Network Deployment V6.1 and the required fixpacks and interim fixes were installed on each system.

2. WebSphere Extended Deployment Operations Optimization V6.1 was installed on each system.

3. A custom profile was created on each system and was federated to the XDcell1 cell as part of the profile creation.

4. The node agents on each system were started.

5. A dynamic cluster was created. Rules were created for the dynamic cluster that specified any node added to the cell whose node name fit the pattern XD* would be able to host cluster members.

6. Sample applications were deployed to the dynamic cluster.

Table 3-3  WebSphere Extended Deployment nodes and dynamic clusters

<table>
<thead>
<tr>
<th>Dynamic cluster</th>
<th>Nodes</th>
<th>Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDCluster1</td>
<td>XDnode1</td>
<td>XDCluster1_XDnode1</td>
</tr>
<tr>
<td></td>
<td>XDnode2</td>
<td>XDCluster1_XDnode2</td>
</tr>
<tr>
<td></td>
<td>XDnode3</td>
<td>XDCluster1_XDnode3</td>
</tr>
</tbody>
</table>

This process completed the initial set up. The details of how these configuration steps were done is discussed throughout the rest of the book.
Planning for Operations Optimization

Preparing for a WebSphere Extended Deployment involves careful planning that is highly dependent on your current environment and expectations. This chapter discusses items for you to consider during your planning process.

We discuss the following topics in this chapter:
- Evaluating the current environment
- Managing expectations
- Topologies
- ODR planning considerations
- Strategies for dynamic clustering
- Installation and maintenance strategy
- Extended repository service
- Migration considerations
- Testing tools and methodologies
4.1 Evaluating the current environment

Evaluation of the existing environment is central to good planning and design considerations. A primary objective of WebSphere Extended Deployment V6.1 is to consolidate application runtime resources and deliver significant cost savings. Hence, taking an inventory of enterprise applications and the application deployment infrastructure, ranging from applied hardware resources to a myriad of software, becomes a vital consideration. Taking inventory of all the hardware and software assets can be instrumental in designing the Extended Deployment topology. This section discusses a few considerations in evaluating an enterprise application infrastructure.

4.1.1 Hardware platforms

This step involves listing all available hardware platforms currently deployed and planned for deployment. Such an inventory has several advantages. First, it gives you a broad picture of the most widely used hardware, which can be a basis for server consolidation. Second, it helps you understand software requirements and compatibility issues.

4.1.2 Application servers

Many environments today are heterogeneous and consist of multiple application server types from multiple vendors. Some well established vendors like IBM may have a larger share of application servers, but a mixed environment is often the reality. Business activities such as mergers and acquisitions also add to the mix of application servers.

Listing all available application servers has the following advantages.

- It is needed to design the Extended Deployment topology and apply various Extended Deployment deployment constructs such as dynamic clusters.
- It facilitates the analysis of various types of lifecycle support offered by Extended Deployment.

Understanding the various levels of lifecycle support is instrumental in managing a heterogenous Extended Deployment environment.

4.1.3 Other middleware components

It is also a recommended practice to list all the middleware components that interact or interface with the application server in some fashion. These components are not necessarily application servers (for example Apache Web
server). Such middleware components may include, but are not limited to, messaging infrastructure, caching module (fabric), security component, or any defined business component. Awareness of any such component is helpful in the planning process for architecture and performance considerations.

### 4.1.4 Applications

Inventory of existing applications is one of the most important activities in the evaluation process. The objective is two-fold.

- First, you need to identify which applications can exploit some or all of the features provided by Extended Deployment.

  For instance, if an application is not truly clusterable then there is very little Extended Deployment can do to provide the application with its QoS capabilities. However, note that these applications can coexist with Extended Deployment-enabled applications, and you can take advantage of certain health monitoring and visualization features, excluding the use of the autonomic actions.

- Second, applications need to be classified based on characteristics that influence how you implement service policies and resource sharing.

  These characteristics include things such as the application’s importance to the business, service level agreement requirements, and organizational requirements affecting applications. For example a set of applications from a specific LOB may be grouped together and deployed on a single server or a set of dynamic clusters.

It is also important to assess any application performance issues or shortcomings at this stage. If an application is already performing poorly in a cluster, Extended Deployment will not address any performance issues; instead, it only tries to provision more resources to meet the service objectives. As a best practice, you should subject each application to regression and stress tests to benchmark the performance results.

### 4.1.5 Skills

An inventory of existing skills and a list of the new skills required is very important to the ongoing success of an Extended Deployment environment. A plan should be developed for building the practitioner level skills of all involved in rolling out the project and managing the environment.

One approach is to upgrade the skills of the existing WebSphere administrators with Extended Deployment skills. This would greatly reduce the learning curve and require a shorter time for skills transfer.
In a heterogeneous environment, you also need to update the skills of non-WebSphere administrators to understand the role of middleware agents in the Extended Deployment environment.

The initial rollout of Extended Deployment may result in various cultural and training hurdles. It is therefore recommended that you involve all stakeholders in the initial education and ongoing education as various features of Extended Deployment are adapted.

4.2 Managing expectations

Managing the expectations of the stakeholders is an important task in any project. Understanding the capabilities of the Extended Deployment offerings is key. For example, Operations Optimization appeals more to the operations side of business and applies more to applications that are transactional in nature. Whereas Data Grid appeals to high performance computing aspects of application development, providing a rich infrastructure for extensible caching fabric and partitioning application components. Finding the correct fit to the array of features provided by Extended Deployment is the key to setting the right expectations.

The primary driver for implementing Extended Deployment is to realize significant cost savings through resource consolidation and autonomic administration driven by policies. This value proposition may lead to certain unrealistic expectations from stakeholders. For instance, the development and testing community may view Extended Deployment as a tool for performance improvement. The operations team may see Operations Optimization as a solution to all their operational issues, such as resource provisioning, problem management and mitigation, and policy driven request response. Finally, management may see Operations Optimization as an infrastructure asset that not only saves cost but also adapts to changing needs of the on demand business.

It is important to stress to the development community that Extended Deployment does not make an application perform any faster than it is already capable of. The role of Extended Deployment is to manage and provision the available resources to best meet the service objectives that support the service level agreement set by the business community or LOB. It is imperative that you performance test each application and establish a baseline benchmarking. Upon achieving acceptable results, a realistic service policy that scales well should be defined.

It is also important to stress to the operations community that while Extended Deployment can reduce the impact of faulty applications on the user community
and it can provide useful diagnostic information, it will not resolve the actual problems.

### 4.3 Topologies

A topology is typically a logical layout of nodes in a cell or cells that provides an end-to-end runtime environment for enterprise applications. In order to design a topology for your enterprise, it is necessary to understand the components of a topology design and the impact of various layers on the overall performance and complexity of the environment.

This section gives a high-level view of the types of nodes and servers that can exist in an Extended Deployment topology and the functional layout of these nodes. It provides a sample layout of common topologies, but remember that every enterprise environment is unique and may differ slightly or completely depending on need and other interacting components.

#### 4.3.1 Simple topology

Figure 4-1 on page 38 represents a supported topology of the product. In this topology, the On Demand Router (ODR) is between the HTTP server tier and the application server tier. The ODR, with the assistance of the autonomic managers, categorizes and prioritizes the work before routing work to the appropriate nodes. The potential nodes are marked with a circle. The deployment manager is the centralized management communication vehicle, as indicated by the dotted lines.

When planning a new WebSphere Extended Deployment installation, a good approach is to start with this simple topology in a test environment. Note that in a production environment there is no requirement for the Web server to sit in front of the ODR.
Install the deployment manager on a separate workstation than the application server nodes to prevent unmanaged work from occurring on the nodes. The solid lines represent the request processing links.

When you are familiar with the simple topology, you can add layers of complexity to your test environment. Include a firewall to secure your environment, and stress test to see the product features in action.

Typically, when the workload increases, you must ensure that you have enough resources to manage the workload and that the resources configure and function together. You can also include an Internet Protocol (IP) sprayer in this configuration to eliminate any single point of failure and make this topology highly available. An IP sprayer distributes requests from one IP address to multiple IP addresses.
4.3.2 Node types

A topology that consists of WebSphere and non-WebSphere servers can include node types:

**WebSphere Extended Deployment nodes** host a WebSphere Extended Deployment node agent and one or more WebSphere application servers. In addition, it may host a PHP server. All the profiles on the nodes are augmented by WebSphere Extended Deployment.

In a migration scenario, nodes still at WebSphere Extended Deployment V6.0.2 are supported with support for most dynamic operations features.

**WebSphere Application Server Network Deployment nodes** host a WebSphere Application Server V5.1 or higher node agent and application servers.

You can augment the node agent on the Network Deployment node, allowing you to monitor the node with Extended Deployment. The centralized installation manager allows you to select servers within a profile to augment, so it is possible to augment the node agent without augmenting the servers.

A Network Deployment V6.0.2 application server is supported as a normal application server (no Extended Deployment feature support). The ODR can route and prioritize traffic, but application editioning and application placement are not available.

A Network Deployment V5.x node can be supported as a Custom HTTP server. A middleware agent on the node provides this support.

**Other middleware nodes** host other servers. They support external monitoring with the middleware agent.

**Naming nodes:** Dynamic clusters for WebSphere and PHP applications are built based on membership rules. These rules define the nodes that participate in the cluster. The rules can be based on any of the following:

- Node group
- Node name
- Node host name
- Node property values

Take your strategy for implementing dynamic clusters into account when defining your nodes.
4.3.3 Server types

The term middleware server refers to a server on any middleware platform. Middleware servers include the following types:

- WebSphere Application Server
- PHP
- Apache HTTP Server
- Apache Tomcat
- Apache Geronimo
- JBoss
- BEA WebLogic Server
- WebSphere Application Server Community Edition
- Custom HTTP servers

Support for these servers is varied. Depending on the server type, you have full lifecycle support, assisted lifecycle support, or generic lifecycle support.

WebSphere Extended Deployment Version 6.1 introduces enhanced support for environments outside of the product domain in the form of assisted lifecycle support. Application servers that run on these platforms are represented more thoroughly in the product administrative domain through the use of a middleware agent installed on these machines. Support for these servers includes dynamic clustering and some health management.

For more information about lifecycle support, see section 2.1.1, “Lifecycle support” on page 8.

4.3.4 Managing non-WebSphere nodes

This use of non-WebSphere middleware nodes places a new challenge on understanding topologies, particularly for IBM WebSphere professionals who are already familiar with existing WebSphere topology concepts.

To keep things simple, the concept of cell, nodes, and servers—the traditional WebSphere topological constructs remain, and in a WebSphere ONLY environment there is absolutely no change. However, while we can manage middleware nodes with assisted lifecycle support from the WebSphere administrative console, they are not a part of WebSphere Cell.

In an environment where there are ONLY non-WebSphere nodes, and if these middleware nodes are to exploit the WebSphere Extended Deployment features, then the concept of cell has to be introduced in the environment to house the deployment manager and the ODRs. In this particular case the deployment manager and ODRs are part of traditional WebSphere cell, and the non-WebSphere nodes are managed from the administrative console.
4.3.5 Web server versus direct access to ODR

There have been arguments by industry practitioners on replacing the Web server (HTTP Server) with ODR, thereby simplifying the topology by eliminating a layer (Web server and plug-in) from the topology. Such a notion would imply that all the traffic destined for applications is sent directly from the IP sprayers behind a domain level firewall. The ODR, in turn routes the traffic based on URIs and other intelligent routing mechanisms enabled by Extended Deployment.

While it is conceivable to operate under such a topology, it is not recommended. The performance trade-off achieved by eliminating the HTTP server as a layer in the topology is negligible. The ODR is a fully functional application server and as such, could introduce a security risk if placed in the DMZ.

In addition, introducing the ODR into its own tier in an existing environment allows you to retain, unaltered, any static caching mechanisms managed by the Web servers.

4.3.6 Stack product integration

The term stack product refers to software that uses WebSphere Application Server as an underlying runtime. A stack product may or may not exploit all the features provided by WebSphere Application Server. Examples of IBM stack products include WebSphere Portal Server, WebSphere Process Server, and WebSphere Commerce. Stack products can also be non-IBM products.

A stack product adds an additional layer of processing and thus complexity to the topology. Many of these products may have a requirement of a dedicated database server or messaging infrastructure, which adds additional points of processing and management to the overall topology.

It is a recommended practice to fully understand the implications of introducing stack products to an existing environment. For example, it is important to evaluate if the stack product will support dynamic clustering and operate within the confines of Extended Deployment constructs.

It is also recommended to have some sort of isolation either through dynamic cluster policies or nodes. This is for manageability and possibly for licensing purposes.

4.4 ODR planning considerations

The ODR is a key element of an Extended Deployment environment. Its primary functions include request routing, intelligent routing based on sense and
response mechanism from back-end servers, and classification of incoming requests based on rules defined by the business owner.

When planning an Extended Deployment topology, many of the planning considerations are focused on the ODR.

### 4.4.1 Choosing the topology

One of the most important decisions to make for your WebSphere Extended Deployment operational environment is the topology you will use.

There are two topology considerations: classic and reformed. Both have their advantages and disadvantages, which we discuss in the following sections.

**Classic topology**

The classic topology shown in Figure 4-2 is similar to accepted WebSphere Network Deployment topologies. One or more Web servers are placed in the DMZ. The Web server plug-in is used to forward incoming requests to the ODRs.

Using the classic topology allows you to leverage existing investments, experiences, and processes. You also avoid the additional expense of a proxy server in the DMZ. The disadvantages are as follows:

- An additional network hop is introduced for the ODRs.
Does not support scale-out features (prioritization, intelligent routing) to other applications running on your HTTP servers in the DMZ.

**Reformed topology**

The advantages of using a reformed topology, shown in Figure 4-3, include full support of scale-out features and fewer network hops. It eliminates the necessity of configuring or generating the Web server plug-in.

![Reformed topology diagram](image)

*Figure 4-3  Reformed topology*

The disadvantages of this topology are that it might not work if you are using a third-party security proxy and that it requires a proxy in the DMZ.

The proxy server shown in Figure 4-3 is necessary to maintain the integrity of the DMZ security. There are three fundamental principles of a DMZ that are applied here:

- Inbound network traffic from outside must be terminated in the DMZ. A network transparent load balancer, such as Network Dispatcher, does not meet that requirement alone.
- The type of traffic and the number of open ports from the DMZ to the intranet must be limited.
- Components running in the DMZ must be hardened and must follow the principle of least function and low complexity.
Read the article *Exploring new network topologies made possible by WebSphere XD and the On Demand Router* by Kyle Brown, Keys Botzum, and Bill Hines. For more information about this topic, access the following Web site:


**Important:** The ODR should not be placed in the DMZ. The ODR is a complex Java application that frequently communicates with all application servers and nodes. This communication includes, for example, exchange of utilization and health status information.

Among other tasks, the ODR calculates the load balancing weights for each application based on this information. If the ODR was within the DMZ, the firewall would need a number of openings, which conflicts with the principles of a DMZ.

### 4.4.2 Using multiple ODRs

It is important to ensure that the ODRs are scalable and highly available. The decision, as to number of ODRs to place in an environment, depends on the enterprise and infrastructure, but it is generally recommended to have at least two ODRs to provide high availability. Various factors will come into play when determining whether to use additional ODRs, for example, the number of clients served, the number of applications, types and size of sessions, and security factors.

### 4.4.3 Configuring ODRs to route to multiple cells

Enable cross-cell communication when you have a WebSphere Extended Deployment administrative cell that contains servers that are enabled with on demand routers (ODR) that route to other WebSphere Application Server administrative cells.

The use of multiple cells can be useful in the following situations:
- An existing environment contains multiple cells, and you want to migrate to WebSphere Extended Deployment.
- You are implementing WebSphere Extended Deployment and merging multiple LOBs with different IT custodial requirements. Using multiple cells allows you to functionally integrate the various LOBs, while separating their management.
- You want to isolate stack products for manageability.
You want to provide a backup if your primary cluster fails, possibly across geographies.

You want to balance the load between multiple clusters.

You want to load balance and provide failover between two WebSphere Portal clusters spread across two cells. WebSphere Portal Server currently has a limitation of one cluster per cell.

ODR routing to multiple cells can extend the boundaries of an organization, while still maintaining administrative boundaries. In other words, each cell still has their administrative control, while the ODR, which resides in its own cell, is the routing engine for all the cells. While this option is attractive and promises to bridge the gaps between various isolated environments, there are a few caveats to consider.

First, you must ensure the flow of information between each cell and the ODR. For the ODR to make intelligent routing decisions, it has to know the server state information from the cells. While a core group bridge facilitates such a flow of information between cells and ODR, proper topological planning and manual configuration is required. It is recommended that you test this configuration thoroughly on a smaller scale first, as this configuration requires each cell to be restarted. After the functionality and behavior of cross cell communication is confirmed, the knowledge obtained from the test environment can be utilized to implement this in a production environment with minimal downtime.

You should implement cross-cell communication one cell at a time. Point the Web server to a cell-specific ODR while configuring the ODR cell and any other cell to minimize the downtime of a production cell while the other cells are being configured and restarted.

It is important that you understand the limitation of a core group and its fundamentals. For instance, a core group cannot contain members from multiple firewall protection domains. When there is a large number of processes in the cell, the core group protocols, such as the View Synchrony Protocol, consume correspondingly large amounts of resources such as CPU. These are significant considerations while defining core groups and core group coordinators. More information regarding core groups can be found in the following WebSphere Application Server Information Center article:

- Core groups (high availability domains)
  

Another very important thing to consider is network and network security. With a tremendous amount of information flowing between cells it is important to consider the network bandwidth requirement between cells. This is particularly
true since the ODR cell requires up-to-date information regarding server state information to make intelligent routing decisions. From a network security standpoint, if a firewall is being utilized between cells, some ports may have to be opened to facilitate exchange of information.

Verify that the firewall is open for the ports that are listed in the deployment_manager_profile/config/cells/cell_name/XD-CGB-EXPORT configuration file to support communication. This file defines the endpoints required for cross cell communication in a runtime environment.

**Note:** Do not bridge cells unless it is necessary. Bridging large cells significantly increases the amount of bulletin board data that each cell has to handle. More data increases the amount of time it takes for the bridge to settle during startup and recover during fail over. Bridging very large cells should not be a casual decision, but a well planned approach.

See the following Information Center article for more information:

- **Configuring WebSphere Extended Deployment for cross-cell communication**
  

Cross-cell communication is also discussed in section 7.8, “Routing in a multi-cell environment” on page 132.

**Multiple core groups in single cell**

In larger cells we recommend that you configure more than one core group, with a core group bridge between the core groups. The core group bridge is automatically configured.

**Best practices:**

- Restrict a core group to 50 server instances.
- Confine a dynamic cluster to one core group.

Core groups that are in the same cell can communicate and share WLM information. If core group bridges were not configured at process startup, every WebSphere Extended Deployment process automatically configures a core group access point for each configured core group. WebSphere Extended Deployment assumes that each of the node agents in each core group is a core group bridge. The configuration is modified to make each node agent a core group bridge. As long as two or more node agents per core group are running, the core groups can communicate.
Configuring multiple core groups is as simple as creating the second core group. WebSphere Extended Deployment creates the core group bridge for you.

4.4.4 Scope of plug-in generation

In Network Deployment environments, the Web server plug-in configuration file is generated based on the Web server definition in WebSphere and propagated to the Web server. When using an ODR, you must generate a special Web server plug-in configuration file that routes incoming traffic to the ODRs instead of directly to the application servers. This configuration file is generated by the ODR and propagated to the Web server.

The most likely topology is to have the Web server route all application traffic to all ODRs in the cell. If for some reason you choose to subset application traffic, having each ODR handle a unique set of URIs, you can have the Web server plug-in generated to do this. The setting that controls this is the scope you select when configuring the ODR plug-in generation. Which ODR you choose to generate the plug-in is also depends on the scope you choose.

The scope options are as follows:

- **All ODRs in the same cell as itself (Cell scope):**
  
  This is the simplest and most common approach. The plug-in contains information about all the ODRs. The ODR generates a plug-in configuration that includes all the URIs that are handled by all the ODRs in the cell.

- **All ODRs in the same node as itself (Node scope):**
  
  The node scope includes all the URIs configured for the node. This can be instrumental in scenarios where testing is required for a new ODR or when the ODRs are used to route traffic to different segments of the application network.

- **All ODRs on the same server as itself (Server scope):**
  
  At the server level, the ODR only generates a plug-in configuration file for the ODR that is currently configured. This scope can also be used when testing a new ODR, or if for some reason you decide to use one Web server per ODR.

If all Web server plug-ins route to all ODRs, choose any single ODR to generate a plugin-cfg.xml file. This is the simplest approach to plug-in generation, as the plug-in will contain info about all the ODRs.

If however, you want a single Web server to front each ODR, configure each ODR to generate a plugin-cfg.xml file. This can be instrumental in scenarios where testing is required for new ODRs that are added to the environment, or for security reasons the ODRs are utilized to route traffic to different segments of the
application network. The ODR uses the proxy plug-in configuration policy to determine how to generate a plugin-cfg.xml file.

Information on generating and propagating the plug-in configuration file can be found in section 7.4, “Proxy plug-in generation” on page 122.

4.4.5 ODR performance considerations

The ODR is basically a server instance and has limitations similar to any other application servers, such as a heap size, performance constraints, and so on.

Note: Due to the critical function of ODR, we do not recommend that ODR be involved in any application serving or caching role.

Consider the following items to ensure optimal ODR performance:

Hardware
This is an important consideration. Care must be exercised to ensure that the hardware is scalable to accommodate growth and has adequate resources such as real memory, CPU, and faster bus speeds. You must also ensure that the chosen hardware is in conformity to supported and recommended hardware for WebSphere Extended Deployment V6.1 and WebSphere Application Server Network Deployment V6.1.

Message size
You will see a linear trend in performance as message sizes grow. Meaning that message size has an adverse effect on ODR performance. This is indicative of more time spent in the network layer, since as message size increases so too does the time it takes to transfer these messages. ODR is scalable enough in its handling of larger message sizes to completely max out a network connection. It is generally recommended that the message size be kept small, but if this is unavoidable then perhaps adding ODRs to the environment may result in optimal performance.

Number of clients
As the number of clients increase, there is more overhead inherent in keeping track of all clients. We therefore recommend that you have a close estimation of clients accessing the environment, and evaluate the performance levels of the current set of ODRs. The decision to add additional ODRs should be considered if the client base increases over time or even exponentially for a short time, due to a business activity such as a promotional offer.
Number of routing rules
Generally the number of routing rules have no impact on ODR performance. Still, a best practice is to try to minimize the number of routing rules. This facilitates better management of rules and less conflicting rules.

SSL
Much of overhead in enabling SSL is the extra working set size (increased memory usage) from taking every encrypted buffer from the network and duplicating it into an unencrypted buffer (and vice versa). The overhead of SSL can also increase the message size. Increasing heap size can help reduce overhead.

The use of an SSL accelerator (and SSL proxy) can help alleviate performance issues, although some would argue against the use of these because the performance gains can be offset due to context switching. Products, such as WebSphere DataPower®, provide this functionality and can offload CPU load from the ODR.

In general, the less encryption, the better the performance, but since it is rare that performance considerations override security concerns, the SSL impact should be factored in ODR planning and provisioning.

JVM settings
In the event of ODR performance degradation, evaluating the heap size is the place to start. Consider the following:

- The ODR footprint is about 90MB.
- Tests show that the ODR generates approximately about 4-5 KB of temporary Obj per request.
- Initial heap calculation = 90MB + 0.05 MB/req * peak req./sec * Desired time between garbage collections (GC).

Many platforms provide additional tuning parameters to optimize garbage collection, for example Sun™, HP, and IBM's J9. For instance with generational GC, the permanent memory region should be set to approximately 100MB to factor in 90 MB of ODR footprint.

To get a better understanding of memory usage by the ODR or for any application server, you can set the verbosegc setting for the JVM. The ODR will then write heap information to the native_stderr.log. For more information about enabling verbosegc see the following article:

- Enabling verbosegc in WebSphere Application Server
  http://www-1.ibm.com/support/docview.wss?uid=swe2114927
**Connection keep-alive settings**
Minimize the time spent setting up and tearing down connections between the ODR and the application servers.

**Persistent connection settings**
The ODR reuses the connection to an application server for as long as the application server will allow. In the application server, you can adjust the HTTP transport to allow a higher number of connections.

The setting is called the **maximum persistent requests** setting. This setting specifies the maximum number of persistent (keep-alive) requests that are allowed on a single HTTP connection. The default setting is 100. It can be adjusted on the Web container HTTP transport channel.

To adjust the setting, do the following:
1. Select **Application servers** → **server_name** → **Web container** → **Web container transport chains**.
2. Select the normal inbound chain for serving requests (usually be named WCInboundDefault, on port 9080), for example **WCInboundDefault** → **HTTP inbound channel (HTTP_2)**.
3. Increase the setting for **maximum persistent requests** to a large number for the best performance.

**Maximum open connections**
The ODRs maximum open connections per server should be increased to allow for more connections to be created and pooled. Always keep in mind the peak connection load and the results from application performance tests.

You can use the Tivoli Performance Viewer and Advisor (which is bundled as part of the WebSphere Application Server) to monitor these settings while performance testing. More information regarding the Tivoli Performance Viewer can be found at the following Web site:


Use the following steps to adjust this setting:
1. Select **Servers** → **On Demand Routers** → **odr** → **Web container** → **Web container transport chains** → **WCInboundDefault** → **TCP inbound channel (TCP_1)**.
2. Increase the maximum open connections value.
Tuning the Web server keep-alive settings
Set the keep-value setting on the Web server high enough to avoid tear-down and re-creation of the connection to the ODR.

Even though the ODR client in the traditional sense is the Web server, the purpose of modifying the keep-alive settings is to take into account the characteristics of end users, which includes varying lag times (“think times”) between requests.

Having a long keep-alive setting prevents the session from timing out when a long pause takes place between requests.

ODR caching
Disable ODR caching if not in use. If enabled, the ODR must go through a process to check if a request should be cached and examine the cache repository. This can cause processing overheads.

You can enable or disable ODR caching in the ODR properties. Select Servers → On Demand Routers → odr_name → On Demand Router Properties → On Demand Router settings.
Access logging
The advice is simple, turn off access logging if it is not needed. Access logging can be enabled or disabled in the ODR properties. Select Servers → On Demand Routers → odr_name → On Demand Router Properties → On Demand Router settings.

![ODR access logging](image)

### 4.5 Strategies for dynamic clustering

Dynamic clusters are central to the operations optimization theme of Extended Deployment. The dynamic cluster extends its counterpart, the static cluster, in WebSphere Application Server Network Deployment. A dynamic cluster is an application deployment target that is automatically expanded and contracted as needed.

When planning for dynamic clustering, consider the following:

- A dynamic cluster consists of servers that host the applications in the cluster.
- A dynamic cluster can only contain one server type.
- Strategies for deploying applications to dynamic clusters are built based on enterprise and performance requirements. For example, applications can be grouped based on their function, resources used, or the LOB they support.

It is common to have one dynamic cluster per application. Consider that if applications share a dynamic cluster they therefore share application servers. If health management decides to recycle an application server because one application is failing, all other applications on that server will be impacted.

- A dynamic cluster for non-WebSphere servers consists of pre-existing servers that you manually specify.
- A dynamic cluster for PHP or WebSphere application servers consists of nodes identified using a membership policy. Application servers are generated for the cluster.
- A node can host servers for multiple dynamic clusters.
- In V6.0, node groups were used as the basis of dynamic cluster membership, this is no longer necessary in V6.1. While the concept of a node group still
exists, its primary use is as an implicit construct defined while configuring dynamic clusters.

The strategy for defining dynamic clusters that best fits an enterprise depends on the number of applications utilizing the shared resources, quality of service (QoS) desired for the applications, and resources available to support the service level goals. There may also be cultural and political reasons that may play into creation and use of dynamic clusters. An application inventory will help you understand the factors in play in your environment.

We recommend that the most important (based on any criteria such as a revenue or usage) and most resource intensive applications be placed in their own dedicated dynamic cluster.

Less frequently used applications or those using low levels of resources can be assigned to one dynamic cluster. When adding any new application, initially assign it to its own unique dynamic cluster to measure and analyze performance metrics and to evaluate its impact on the performance of existing applications.

4.5.1 Dynamic cluster isolation

An isolation preference indicates whether you want the dynamic cluster to run on the same nodes as other instances of dynamic clusters, or if you want the dynamic cluster to be the only dynamic cluster that is running on a node. This can impact your topology, as depending on your Isolation policy levels you may need additional dynamic clusters and additional nodes. Exercise caution in choosing the isolation levels, as an improper isolation policy in an already resource constrained environment can cause resource allocation issues.

Following are the options for isolation settings:

- Use *No isolation requirements* if cluster instances in this dynamic cluster can run on the same node as other instances from different dynamic clusters. This option is the default option.

- Use *Strict isolation* if cluster instances in this dynamic cluster can run only with other instances of the same dynamic cluster on the same node.

- Use *Associate with an isolation group* if the cluster instance can run with any other instance of a dynamic cluster that is in the same shared group on the same node.
4.6 Installation and maintenance strategy

IBM WebSphere Extended Deployment V6.1 introduces a new feature called the centralized installation manager. This feature enables installation of software packages and maintenance from the deployment manager to target hosts defined to the cell.

The centralized installation manager is installed and its repository created when you install Operations Optimization. After installed, you can also use centralized installation manager with Compute Grid V6.1 and Data Grid Version 6.1.

With the centralized installation manager you can perform the following installation types:

- Product install
- Refresh pack, fix pack, or maintenance tool
- Interim fix.

The centralized installation manager does not install maintenance on the deployment manager.

The centralized installation manager is accessed from the administrative console by navigating to System administration → Centralized installation Manager. Select Available installations to see the installation packages available. Initially you will see the Operations Optimization and middleware agent as product install options. You can download additional packages for the IBM support site.

When you are building a new topology that includes non-WebSphere servers, define the new hosts to the cell as installation targets and install the middleware agent to the host using the centralized installation manager. For more information about this, see section 6.1.1, “Installing the middleware agent” on page 86.

4.7 Extended repository service

With the repository checkpoint and restore function, you can back up copies of files from the master configuration repository. You can use the backups to restore the configuration to a previous state if future configuration changes cause operational problems. By using this function, you can reduce recovery time for problems that are caused by configuration changes. Studies of unplanned outages have shown that as much as 36% of unplanned outages are due to operator errors. A common source of operator error is a bad configuration change. The ability to quickly undo a bad configuration change is critical to minimizing the outage window.
Repository checkpoints represent saved images of the repository before configuration changes are made.

The following two checkpoint types are supported:

- **A full checkpoint** is a complete copy of the entire configuration repository. A full checkpoint is useful to take a snapshot of a known working configuration to establish a baseline. A full checkpoint includes applications and connectors, so it can be very large.

  Full checkpoints are created manually at administrative discretion.

- **A Delta checkpoint** is not a full copy of the configuration, but rather, it is a subset. The subset is comprised of a before-image snapshot of the individual configuration files modified by a discrete configuration change. A configuration save marks the end of a discrete configuration change.

  The delta checkpoint is formed by making a copy of the configuration documents affected by the configuration change before the changes are actually applied.

  Delta checkpoints are created automatically by the system each time a configuration change is made.

Checkpoints can be used to restore the configuration repository back to a prior state. Use a full checkpoint to restore the entire configuration repository back to the state it was in at the time the full checkpoint was made.

Use delta checkpoints to undo recent changes. Delta checkpoints can only be restored in the reverse order in which they were created. Each delta checkpoint has a sequence number. The highest sequence number represents the most recent delta checkpoint. Delta checkpoints can be restored in descending sequence number only. After the configuration repository is restored from a delta checkpoint, that checkpoint is destroyed.

The administrator has no control over the naming of the delta checkpoints. This may potentially be an issue to associate a delta checkpoint with change. The only way to counteract this issue is to manually drill down into the delta checkpoint repository, and examine the files that were changed. The file list is displayed on the administrative console and can be opened on the administrative console itself. At this point there is no mechanism to compare the differences between the files of two or more delta checkpoints. One way to spot the difference is to use a ‘diff’ like unix utility or similar tool to compare the two files.

As a best practice it is always advisable to take a full checkpoint at regular intervals, and for full checkpoints, unlike delta checkpoints the administrator has the capability to assign the names. This can be particularly helpful, as a full
checkpoint on regular intervals can be more descriptive. This makes restoring a lot easier, as delta checkpoints are built upon latest full checkpoint repository.

4.8 Migration considerations

Migrating from either an existing IBM WebSphere Extended Deployment environment or upgrading an existing IBM WebSphere Network deployment environment is usually a phased approach. It is prudent to plan the migration process and understand the implications of an environment that is in midst of a migration plan. This is especially true in the case of Extended Deployment, which has defined support and capability for various versions of WebSphere and non-WebSphere nodes.

The following steps are required for migration from an existing WebSphere Extended Deployment environment:

1. Install the new version of WebSphere Extended Deployment on the systems.
2. Augment the new and existing profiles in the cell with Version 6.1. This could mean either augmenting a new profile or migrating from Extended Deployment 6.0 to Extended Deployment 6.1.

More information about augmenting a profile can be found at the following Web site:


To avoid any downtime or impact on the business it is important that you research the process required for migration.

4.8.1 Upgrading to Extended Deployment V6.1 from a Network Deployment environment

When implementing a new Extended Deployment environment on an existing Network Deployment environment, one of the following approaches can be taken:

Suggested approach:

1. Install WebSphere Extended Deployment.
2. Create a new WebSphere Extended Deployment v6.1 environment, including the deployment manager, ODRs, nodes, and dynamic clusters.
3. Install the applications in this new cell one at a time.
4. Define the service policies for the new applications and test each thoroughly.
Alternate approach:

1. Install WebSphere Application and WebSphere Application Server Network Deployment V6.1.0.7.

2. Incrementally migrate each server, starting with the deployment manager to the V6.1 level, verifying that the applications are migrated successfully.

3. Install WebSphere Extended Deployment.

4. Augment the WebSphere profiles, starting with deployment manager with WebSphere Extended Deployment V6.1.

5. Define the service policies for the new applications, and test each thoroughly.

This approach is appropriate for environments that may have issues migrating applications to Network Deployment V6.1, for example, for installations at WebSphere Application Server V5.

4.8.2 Migrating from Extended Deployment 6.0.2

WebSphere Extended Deployment Version 6.1 can support migration from WebSphere Extended Deployment Version 6.0.2 when Version 6.0.2 is installed on Network Deployment Version 6.0.2 or Network Deployment Version 6.1. While both installations are supported, the steps to migrate from each type of environment are slightly different.

WebSphere Extended Deployment Version 6.1 introduces a new strategy for migrating and merging the configuration of separate products. With previous versions, you were required to migrate the deployment manager and all of the nodes at the same time. With Version 6.1, you first migrate the deployment manager, and then migrate each node one at a time, resulting in a mixed cell environment.

If you have an existing installation of WebSphere Extended Deployment Version 6.0.2 on WebSphere Application Server Network Deployment Version 6.1, you can migrate directly from the product Version 6.0.2 to Version 6.1 without having to install or migrate WebSphere Application Server Network Deployment V6.1.

If you plan to run a mixed cell during migration, keep the following in mind:

- The template for the dynamic cluster is new for V6.1 and is upgraded during migration. Pre-existing V6.0.2 dynamic clusters will have this new template. The template can be used as a mechanism to manage all cluster members, or each member can be managed individually. After the template is migrated up to V6.1, it cannot be used to manage any back-level members. When a change is made to the template, it is not propagated out to any cluster.
member at a lower version than itself. Those back-level members must be manually managed (like static cluster members).

- When creating a new dynamic cluster you cannot have mixed versions for members.
- The autonomic controllers will only run on the V6.1 node agents.
- For custom health expressions, the MBean and PMI metric must be available from the V6.0.2 server or a failure may occur.

**Extended Deployment V6.0.2 on Network Deployment V6.0.2**
The following is a summary of the steps required to migrate an Extended Deployment V6.0.2 deployment manager to Extended Deployment V6.1 when the underlying Network Deployment installation is V6.0.2:

1. Prepare a new deployment manager profile:
   b. Create a new deployment manager profile.
   c. Install WebSphere Extended Deployment Version 6.1.
   d. Augment the new deployment manager profile for WebSphere Extended Deployment 6.1 by using the xd_augment template.

2. Migrate the existing deployment manager to Network Deployment Version 6.1 by using WASPreUpgrade and WASPostUpgrade commands.
   The WASPreUpgrade command saves the configuration of the existing WebSphere Application Server installation into a migration-specific backup directory.
   The WASPostUpgrade command retrieves the saved configuration to migrate to WebSphere Application Server Version 6.1 and adds all migrated applications into the app_server_root\installedApps directory for the Version 6.1 installation.
   At the end of this step, your deployment manager profile is at Network Deployment V6.1 and Extended Deployment V6.0.2.

3. Migrate the existing deployment manager to the new deployment manager profile using the XDUpgrade command or the migration wizard.
   At the end of this step, the new deployment manager profile is at Network Deployment V6.1 and Extended Deployment V6.1.
The following is a summary of the steps required to migrate an Extended Deployment V6.0.2 node to Extended Deployment V6.1 when the underlying Network Deployment installation is V6.0.2:

1. Prepare a new custom profile for the node:
   b. Create a new custom profile.
   c. Install WebSphere Extended Deployment Version 6.1
   d. Augment the new custom profile for WebSphere Extended Deployment 6.1 by using the xd_augment template.

2. Migrate the custom profile for the existing node to Network Deployment Version 6.1 by using WASPreUpgrade and WASPostUpgrade commands.
   At the end of this step, your custom profile is at Network Deployment V6.1 and Extended Deployment V6.0.2.

3. Migrate the existing node to the new custom profile using the \texttt{XDUpgrade} command or migration wizard.
   At the end of this step, the new custom profile is at Network Deployment V6.1 and Extended Deployment V6.1.

\section*{Extended Deployment V6.0.2 on Network Deployment V6.1}

The following is a summary of the steps required to migrate an Extended Deployment V6.0.2 deployment manager to Extended Deployment V6.1 when the underlying Network Deployment installation is V6.1:

1. Stop the deployment manager.

2. Uninstall WebSphere Deployment Version 6.0.2.


4. Augment the WebSphere Application Server Version 6.1 deployment manager profile for WebSphere Extended Deployment 6.1 by using the xd_augment template.

5. Migrate the Version 6.0.2 deployment manager by using the migration wizard or \texttt{XDUpgrade} command.

The following is a summary of the steps required to migrate an Extended Deployment V6.0.2 node to Extended Deployment V6.1 when the underlying Network Deployment installation is V6.1:

1. Stop the application servers that are running on the node and the node agent.

2. Uninstall WebSphere Deployment Version 6.0.2.

4. Augment the WebSphere Application Server Version 6.1 deployment manager profile for WebSphere Extended Deployment 6.1 by using the xd_augment template from the profile_root/bin directory on the workstation hosting the node.

5. Migrate Version 6.0.2 node by using the migration wizard or the XDUpgrade command.

**For more information**

The articles referenced in the following links to the Information Center provide more information about migration from a WebSphere Extended Deployment V6.0.2 environment.

1. For an overview of the migration steps required, see *Where migration tasks are performed*:


2. For information about the migration process, see *Migrating to WebSphere Extended Deployment Version 6.1*:


3. For information about migrating a deployment manager, see *Migrating to a Version 6.1 deployment manager*:


4. For information about migrating a node, see *Migrating to a Version 6.1 node*:


### 4.9 Testing tools and methodologies

When implementing a new or migrated Extended Deployment environment, it is important to ensure that the desired features work and produce the results that you expect. There are several areas of testing, for instance the health policies and health actions, the service policies used to classify requests, and so on.

From a tooling standpoint, there are several tools available for generating requests and sending them to the ODR. For example, JMeter from the Apache jakarta project. This tool can be downloaded from the following Web site:

While tooling can provide a mechanism to load test, the methodologies provide for establishing test cases to test for the health policies and subsequent health actions, service policies, and other features of Extended Deployment.

A recommended practice is to test the access to all the applications through the ODR prior to moving into production. In order to simulate a real world environment it is best to have all the ODRs and a parallel HTTP server in the test environment. With this topology the production traffic can still be routed via the original HTTP server and plug-in. Test traffic can be routed via the HTTP server in the test environment to the ODRs and from the ODRs to the application servers. The production HTTP plug-in can be updated from the ODRs, upon successful completion of the tests.

We recommend that you adopt the Extended Deployment environment in piece-meal fashion and not all at once. Such a strategy mitigates deployment risks and allows for the operations team to build skills in Extended Deployment concepts.

Also, during the initial phases of Extended Deployment implementation it is advisable to configure all health actions in supervise mode to build a comfort level with health policy and resulting actions.
Creating the basic environment

This chapter discusses how to create a deployment manager profile and how to access the administrative console. It also discusses the use of a second deployment manager for high availability. Finally it discusses how to add new nodes to the cell for the WebSphere application servers.

The following topics are discussed in this chapter:

- Creating the deployment manager
- High availability deployment manager
- Adding WebSphere nodes to the cell
5.1 Creating the deployment manager

Installing the WebSphere software does not give you an operational environment. The runtime components are defined by profiles. You can create profiles as part of the installation or after installation.

The first profile that needs to be created is the deployment manager profile on one of the deployment manager machines—in our case XDDmgr1node. Creating the deployment manager, in effect, also creates the cell.

Note that if you are planning to install two deployment managers for use in a high availability configuration, the profile must be created on the shared file system.

z/OS® users: The sample topology used distributed systems. The instructions in this chapter for creating profiles applies only to distributed systems. If you are not familiar with the basics of using WebSphere Application Server Network Deployment (the underlying technology for WebSphere Extended Deployment), see WebSphere Application Server V6.1: System Management and Configuration, SG24-7304 for more information, including how to create profiles on z/OS.

5.1.1 Creating a profile using the profile management tool

Note: The Profile Management Tool can also be started from the launchpad at the end of the WebSphere Application Server installation. However, this is not recommended if you need to also install WebSphere refresh and fix packs.

1. Launch the profile management tool using the appropriate command for your platform.
   - On Windows® systems: `app_server_root\bin\ProfileManagement\pmt.bat`
   - On UNIX®: `app_server_root/bin/ProfileManagement/pmt.sh`

2. To create a new profile, select Create.

Profiles in Extended Deployment are actually Network Deployment profiles that were augmented for use with Extended Deployment. The Augment option is used for augmenting a profile that was created in a Network Deployment cell before you installed Extended Deployment.
3. Click **Next**.

4. The next panel is a Welcome window. Click **Next**.

5. Select WebSphere Extended Deployment Operations Optimization as the environment. Any other selection results in a profile that has not been augmented for Extended Deployment.

6. Click **Next**.

7. Select **Deployment manager** as the profile type, and click **Next**.
8. Select Advanced profile creation, and click Next.

9. Take the default that allows deployment of the administrative console, and click Next.

10. Type the names to use for the new node and cell. Enter the host name of the system, and click Next.
11. Select whether to have administrative security enabled or not.

If you enable the security during profile creation on distributed systems, a file-based user repository is created and populated with the administrator ID. This file-based system can be federated with other repository types to form an overall repository system. If you do not want to use the file-based repository, do not enable administrative security during profile creation.

*WebSphere for z/OS:* You can choose to use the file-based repository or use the z/OS system SAF-compliant security database. Whether you choose to enable administration security during profile creation or after, it is important that you do it before going into production.

**Important:** It seems pretty obvious, but make sure that you note the user ID and password specified here. You will need this in order to do anything with the cell you are creating.

12. The next panel allows you to specify port values to use. Ports must be unique on the system. The profile management tool selects a set of ports for this profile unique to the WebSphere installation.

**Note:** Write down the SOAP connector port number. You will need to know this later when you federate nodes to the cell.

Also note the Administrative console port. This is the port used to access the administrative console.

13. *Windows:* Select whether to run the deployment manager as a Windows service.

**Note:** Running the process as a Windows service allows you to change the service startup type to “Automatic”, so that the deployment manager will start when the system starts; otherwise, it must be started manually.

On UNIX and Linux®, the administrator can choose to include entries in inittab for WebSphere processes. Each such process will then be automatically restarted if it fails or the system restarts.

*WebSphere for z/OS* takes advantage of the z/OS Automatic Restart Management (ARM) to recover application servers.

14. Review the profile summary, and click **Create**.

The profile will be created. The last step will launch the First Steps menu.
5.1.2 Starting the deployment manager

The next step is to start the deployment manager using the First Steps menu, which you can launch from the Profile creation is complete panel or using the \texttt{startManager} command. The \texttt{startManager} command is profile-aware and must be issued from the profile’s bin directory. For example:

\texttt{app_server_root/profiles/XDcell1/bin/startManager}

5.1.3 Logging on to the administrative console

You can open the administrative console to further verify that the deployment manager was successfully started.

The default Web location for the console is:

\texttt{http://dmgr_node:port.ibm/console}

Where \texttt{port} is the administrative console port specified during profile creation.

If you enabled administrative security, you will have to enter the user ID and password you specified during profile creation. If you did not enable it, you can enter any value in the user ID field, or you can leave the field blank and simply click Login. The user ID tracks changes you make and only one administrator with the ID can login at a time.
5.2 High availability deployment manager

The high availability deployment manager function provides high availability for the deployment manager server and uses a hot-standby model for availability. With this support, two or more deployment manager's peers may be defined and started in the same cell. One is considered active, also known as primary, and hosts the cell's administrative function, while the others are backups in standby mode. In standby mode, the deployment manager cannot be used to perform administrative functions. If the active manager is stopped or fails, a standby takes over and is designated the new active deployment manager.
Using an HA deployment manager eliminates the deployment manager as a single point of failure for cell administration. This is important in environments that have significant reliance on automated operations, including application deployment and server monitoring.

Each deployment manager is installed and configured to run on a different physical or logical machine. The deployment managers need not be hosted on homogenous operating platforms, although like platforms are recommended. A command line utility is provided with WebSphere Extended Deployment to clone the original cell deployment manager into additional deployment managers.

Each deployment manager shares the same instance of the master configuration repository and workspace area, which are located on a shared file system. The file system must support fast lock recovery. The IBM Storage Area Network File System (SAN FS) is recommended, and the Network File System Version 4 (NFS) is also an option.

A typical HA deployment manager configuration is built with a pair of deployment managers sharing a single WebSphere install and master repository located on a SAN FS. It is also possible to install the WebSphere binaries locally and to put only the deployment manager profile onto the shared file system. You might have to use this configuration method if your deployment managers are hosted on different operating system platforms. A redundant ODR pair is configured for communication availability.

The HA deployment manager function supports only use of the JMX™ SOAP connector. The JMX RMI connector is not supported in this configuration.

We recommend that you configure the deployment managers into the same core group so that routing information exposed to the ODR is consistent across all deployment managers. If the deployment managers are placed into separate core groups, the core groups must be connected with a core group bridge.

Normal operation includes starting at least two deployment managers. A highly available deployment manager component runs in each deployment manager to control which deployment manager is elected as the active one. Any other deployment manager in the configuration is in standby mode.

The WebSphere Extended Deployment On Demand Router (ODR) is configured with the communication end-points for the administrative console, wsadmin, and scripting. The ODR recognizes which deployment manager instance is active and routes all administrative communication to that instance.
In other words, after the HA deployment manager is set up, you do not access the administrative console on the deployment manager system itself any more but through the ODRs. For example, use the URL:

http://odr1:9060/admin

instead of:

http://dmgr1:9060/admin

All administrative operations are performed through the elected active deployment manager. The standby deployment manager is fully initialized and ready to do work but cannot be used for administration. This is because the administrative function does not currently support multiple concurrent server processes writing to the same configuration. Therefore, the standby rejects any login and JMX requests. However, if the active deployment manager is stopped or fails, the HA deployment manager component recognizes the loss of the active deployment manager and dynamically switches the standby into active mode, so it can take over for the lost deployment manager.

The active and standbys share work spaces. When a deployment manager takeover occurs, work is not lost. When the deployment manager takeover occurs, the ODR automatically recognizes the election of the new active deployment manager and reroutes administrative requests to the new active deployment manager, as depicted in Figure 5-6 on page 72.
While the HA deployment manager component can detect deployment manager failures and initiate takeover, there are edge conditions where each deployment manager could temporarily believe it is the active deployment manager. To prevent this, the active deployment manager holds a file lock in the shared file system. Because of this, the takeover of the active deployment manager by the standby takes a brief period of time—approximately equal to the time it takes for the shared file system to detect the loss of the active deployment manager and release the lock. SAN FS and NFS both use a lock lease model and have configurable times for lock release for failed lock holders. This can be configured as low as 10 seconds for SAN FS.

This chapter gives you information about the following possible scenarios:

- The topology is configured with an active deployment manager, two ODRs and all nodes. Now you want to add the HA deployment manager. This scenario assumes that the existing deployment manager profile is located on the shared file system. See section 5.2.1, “Configuring the HA deployment manager” on page 73 for details.

- You create a new cell with an HA deployment manager. See section 5.2.2, “Creating a new cell with an HA deployment manager” on page 81. The step-by-step configuration instructions in section 5.2.1, “Configuring the HA deployment manager” on page 73 are valid for this scenario also.
You have an existing WebSphere Extended Deployment cell that does not use the HA deployment manager function yet. The deployment manager profile is not located on the shared file system. You must therefore move the existing configuration onto the shared file system in order to add additional deployment managers. See section 5.2.3, “Converting existing cell to HA deployment manager configuration” on page 81. Again, you can use the step-by-step instructions in section 5.2.1, “Configuring the HA deployment manager” on page 73 as a reference.

### 5.2.1 Configuring the HA deployment manager

This section shows you how to add a second deployment manager to an existing environment to take advantage of the HA deployment manager function in WebSphere Extended Deployment. This assumes that the profile for the existing deployment manager is located on a shared file system, making it fairly simple to add the second deployment manager to the environment.

**Important:** Be certain that the following two conditions are met.

- The primary and backup deployment manager must share the same master configuration repository and workspace area. Thus you need to make sure that the two systems have a suitable shared file system available, for example IBM SAN FS or NFS version 4.
- You can install the WebSphere binaries into the shared file system or individually on each system. The deployment manager profile *must* reside on the shared file system.

**Note:** If your deployment manager profile is not located on a shared file system, then you need to follow the steps explained in section 5.2.3, “Converting existing cell to HA deployment manager configuration” on page 81 to first move the profile to the shared file system.

**Prerequisites**

The following prerequisites are assumed to be met for this configuration:

- You have an appropriate shared file system and it is accessible by both deployment manager systems.
- You have WebSphere Application Server Network Deployment and WebSphere Extended Deployment installed on the shared file system or
locally on each deployment manager system. The existing deployment manager profile must be on the shared file system.

In this example, one of the deployment managers is on AIX while the other runs on Linux. Therefore the binaries are installed locally on each system, and only the deployment manager profile is located on the shared file system.

- The cell’s deployment manager is started.
- The On Demand Routers are installed, configured, and running.

Table 5-1 lists the system names and IP addresses that are used in the sample configuration:

<table>
<thead>
<tr>
<th>System name</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>odr1</td>
<td>10.2.2.1</td>
</tr>
<tr>
<td>odr2</td>
<td>10.2.2.2</td>
</tr>
<tr>
<td>dmgr1</td>
<td>10.2.3.1</td>
</tr>
<tr>
<td>dmgr2</td>
<td>10.2.3.2</td>
</tr>
</tbody>
</table>

The configuration steps

WebSphere Extended Deployment is installed on the dmgr2 node. The dmgr1, odr1, and odr2 nodes are up and running. Adding dmgr2 as an HA deployment manager to the environment involves the following tasks:

- Creating a new deployment manager profile on the shared file system for dmgr2.
- Running the `xd_hadmgrAdd` command from dmgr2’s profile directory to convert dmgr2 into an HA deployment manager peer.
- Restarting the deployment managers and ODRs.

Creating a new deployment manager profile

Follow the steps in section 5.1, “Creating the deployment manager” on page 64 to create a new deployment manager profile in WebSphere Extended Deployment. Use the following values:

- Enter the new profile name. In this example, `dmgr2` is used as the profile name.
- Enter the path to the WebSphere profiles directory on the shared file system as the profile directory. The directory name defaults to the profile name, so in the example, the profile directory is as follows:

  `/opt/IBM/WebSphere/AppServer/profiles/dmgr2`
Enter the node name for the new deployment manager in the Node name field. In this example, the node name is `dmgr2`.

Enter the fully qualified host name of the new deployment manager system into the Host name field. In this example, the host name is `dmgr2.ibmredbooks.com`.

Enter any value into the Cell name field. This value does not matter as it will be overwritten when the primary deployment manager's configuration is cloned using the `xd_hadmgrAdd` command.

**The `xd_hadmgrAdd` command**

WebSphere Extended Deployment provides the `xd_hadmgrAdd` command to incorporate a new deployment manager profile into a cell. The new deployment manager profile is converted into an HA deployment manager peer in the same cell as the existing deployment manager profile. The command must be run from the new deployment manager profile that is being added.

Following is the format of the `xd_hadmgrAdd` command:

```
x_d_hadmgrAdd -hostname (primary_dmgr_host) [-port (primary_dmgr_port)]
-configRoot (fully_qualified_path_to_shared_configuration)]
-workspaceRoot (fully_qualified_path_to_workspace)
-proxyServerJmxSoapAddress (JMX_SOAP_host):(JMX_SOAP_host)
-proxyServerHttpPort (HTTP_port)(HTTP_secure_port) [-uniquePort] [-user (uid)] [-password (pwd)] [-quiet] [-logfile (filename)] [-replacelog] [-trace] [-help]
```

Use the following steps to add `dmgr2` as an HA deployment manager peer:

1. Switch to the `app_server_root/profiles/dmgr2/bin` directory.
2. Run the `xd_hadmgrAdd` command with at least the following parameters:
   a. `-hostname`  
      Enter the IP address of the existing deployment manager, in our case `10.2.3.1`.
   b. `-configRoot`  
      Enter the path to the existing deployment manager's configuration directory, in our case `/usr/IBM/WebSphere/AppServer/profiles/dmgr1/config`.
   c. `-workspaceRoot`  
      Enter the path to the existing deployment manager's profile directory, in our case `/usr/IBM/WebSphere/AppServer/profiles/dmgr1`.
d. `-proxyServerJmxSoapAddress`

Enter the IP address and JMX Proxy port of your ODR. In our case this is `10.2.2.1:8882`.

You can find the ODR port in the administrative console as follows: Click `Servers → On Demand Routers → ODR_router_name → Ports` (under Communications). You need to pick a new port number that is not listed here. If you have more than one ODR, you should use a port number that is not assigned on any ODR.

e. `-proxyServerHttpPort`

Enter the ODR’s port numbers for `WC_adminhost` and `WC_adminhost_secure`. Again, you should pick port numbers that are not assigned on the ODRs. In our case they are 9060 and 9043.

**Note:** The `-proxyServerJmxSoapAddress` and `-proxyServerHttpPort` ports are used to construct new ODR end points. Hence these new ports could conflict with any ODR, application server, deployment manager, or node agent running on the ODR machines. You are required to ensure that the selected ports are free on the ODR machines, or add the `-uniquePort` option to the command to generate unique port numbers.

-uniquePort makes sure that there are no port conflicts for the `WC_adminhost` and `WC_adminhost_secure` HTTP transport channels as well as for the JMX_SOAP_PROXY_ADDRESS endpoint.

You still need to add port numbers to the command syntax, but if a selected port is in use on the ODR, this port number is incremented by one until a free port is found.

For example, you could get a port conflict for the 9060 and 9043 adminhost ports if you have application servers configured on the ODR nodes. These ports do not show up in the ODR port list but only in the port assignment list of the respective application server.

Refer to the command usage or the Information Center article `xd_hadmgrAdd command` to discern if you need additional parameters for your environment. For example, if security is enabled for your cell, then you also must specify the `-user` and `-password` parameters to connect to the existing deployment manager. Or you must add the deployment manager's SOAP port after the `-hostname` parameter, if you are not using the default port of 8879.

**`xd_hadmgrAdd example`**

Example 5-1 on page 77 shows the command used for the example environment.
Example 5-1  
**xd_hadmgrAdd command to add dmgr2 to xdcell**

```
/opt/IBM/WebSphere/AppServer/profiles/dmgr2/bin >./xd_hadmgrAdd.sh -hostname 10.2.3.1 -configRoot /usr/IBM/WebSphere/AppServer/profiles/dmgr1/config -workspaceRoot /usr/IBM/WebSphere/AppServer/profiles/dmgr1 -proxyServerJmxSoapAddress 10.2.2.1:8882 -proxyServerHttpPort 9060 9043
```

3. The command might run for several minutes. The progress of the command is output to your command prompt window and is shown in Example 5-2.

**Example 5-2  Adding dmgr2 to xdcell**

```
ADMU0116I: Tool information is being logged in file
    /opt/IBM/WebSphere/AppServer/profiles/dmgr2/logs/xd_hadmgrAdd.log
ADMU0128I: Starting tool with the dmgr2 profile
XHAD1001I: Connecting to primary deployment manager host 10.2.3.1 using port 8879
XHAD1002I: Begin addition of standby deployment manager node dmgr2 to cell xdcell
XHAD1029I: Updating endpoints in On Demand Router odr1
XHAD1028I: WC_adminhost(proxyServerHttpPort) is assigned to port 9060
XHAD1028I: WC_adminhost_secure(secure proxyServerHttpPort) is assigned to port 9043
XHAD1028I: SOAP_PROXY_CONNECTOR_ADDRESS(proxyServerJmxSoapAddress) is assigned to port 8882
XHAD1029I: Updating endpoints in On Demand Router odr2
XHAD1028I: WC_adminhost(proxyServerHttpPort) is assigned to port 9060
XHAD1028I: WC_adminhost_secure(secure proxyServerHttpPort) is assigned to port 9043
XHAD1028I: SOAP_PROXY_CONNECTOR_ADDRESS(proxyServerJmxSoapAddress) is assigned to port 8882
XHAD1030I: Updating endpoints in deployment manager of node dmgr1
XHAD1028I: SOAP_PROXY_CONNECTOR_ADDRESS is assigned to port 8879
XHAD1028I: SOAP_CONNECTOR_ADDRESS is assigned to port 8882
XHAD1003I: Standby deployment manager node dmgr2 has been added successfully to cell xdcell
XHAD1004I: Please restart the active deployment manager in node dmgr1
XHAD1005I: Please restart the On Demand Router odr1
XHAD1005I: Please restart the On Demand Router odr2
```

**Notes:**

- As you can see in Example 5-2, the command automatically discovers that there are two ODRs in the environment and updates both ODRs.
- The messages indicate that you need to restart the active deployment manager and the ODRs.

**Configuration changes performed by xd_hadmgrAdd command**

The **xd_hadmgrAdd** command makes the following changes to the cell configuration:

1. The new deployment manager is configured to use the same configuration repository instance as the existing deployment manager, which must be on a shared device with the existing deployment manager.

Chapter 5. Creating the basic environment  77
2. The new deployment manager is configured to use the same specified workspace shared directory, which must be on a device shared with the existing deployment manager.

3. The on demand routers are configured with the following:
   a. WC_adminhost and WC_adminhost_secure HTTP transport channels.
   b. JMX_SOAP_PROXY_ADDRESS endpoint.

4. The deployment managers are reconfigured as follows:
   a. Their JMX_SOAP_CONNECTOR addresses point to the ODR’s JMX_SOAP_PROXY_ADDRESS.
   b. They have a JMX_SOAP_PROXY_ADDRESS added that holds the same configured host or port as their original JMX_SOAP_ADDRESS.

**Important:** The address specified using `-proxyServerJmxSoapAddress` becomes a single point of failure. In an HA deployment manager environment, node agents connect to this address to perform administrative tasks such as synchronizing the local repository with the master repository.

To eliminate the ODR as a single point of failure you need at least two ODRs and an IP sprayer fronting them.

If you have an IP sprayer in front of your ODRs, then the deployment manager’s JMX_SOAP_ADDRESS configured host name should point to the IP sprayer instead of to an ODR directly. For more information about this configuration please refer to the Information Center article `xd_hadmgrAdd command`.

**Starting the deployment managers and ODRs**

To finish the HA deployment manager configuration, you must now restart the active deployment manager and the ODRs and also start the newly configured deployment manager.

1. To restart the active deployment manager, go to its `profile_home/bin` directory, and issue the `stopManager` command. When it stops, run the `startManager` command.

2. To restart the ODRs:
   a. Open the administrative console and go to Servers → On Demand Routers.
   b. If you can stop all active ODRs at the same time (because you have no active requests in your WebSphere system), select all ODRs, and press the Stop button.
c. Select them again, and click the **Start** button. If your WebSphere system is actively processing user requests, then you should stop and start one ODR after the other.

3. The last step is to also start the new deployment manager. Go to the dmgr2's `profile_home/bin` directory, and issue the `startManager` command.

The first deployment manager that is successfully started becomes the active deployment manager, while the one started later becomes the standby. If you then stop the active deployment manager, or if it fails, the standby takes over and becomes active. It stays active until you stop it or until it fails, that is, the deployment manager does not automatically fail back when the original deployment manager becomes active.

**Verifying the HA deployment manager configuration**

There are a few ways to verify that the configuration of the HA deployment manager was successful:

- Look at the SystemOut.log files of both deployment managers. The following messages in the dmgr2 SystemOut.log indicate that this deployment manager is in standby mode because it was started after dmgr1 was already active.

  XHAD0001I: The Deployment Manager process is in standby mode.
  ...
  ASND0002I: Detected server dmgr started on node dmgr01

- After stopping dmgr1 using the `stopManager` command, dmgr2 takes over processing and becomes the active deployment manager. The following messages in dmgr2's SystemOut.log indicate the transition.

  ASND0002I: Detected server dmgr started on node dmgr2
  WSVR0001I: Server dmgr open for e-business
  ....
  XHAD0002I: The Deployment Manager process is in active mode.
  ....
  ASND0003I: Detected server dmgr stopped on node dmgr1
  ....

- Verify the port assignments on both ODRs using the administrative console. Click **Servers → On Demand Routers → ODR_name → Ports** (under Communications). Notice that WC_adminhost (Figure 5-7 on page 80) on
both ODRs is assigned to port number 9060 and WC_adminhost_secure is assigned to 9043.

**Accessing the administrative console:**

Use the ODR WC_adminhost or WC_adminhost_secure port to access the deployment manager, for example:

http://odr1:9060/ibm/console

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOTSTRAP_ADDRESS</td>
<td>9810</td>
</tr>
<tr>
<td>SOAP_CONNECTOR_ADDRESS</td>
<td>8880</td>
</tr>
<tr>
<td>DRS_CLIENT_ADDRESS</td>
<td>7873</td>
</tr>
<tr>
<td>SAS_SSL_SERVERAUTH_LISTENER_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>WC_defaulthost</td>
<td>9080</td>
</tr>
<tr>
<td>DCS_UNICAST_ADDRESS</td>
<td>9354</td>
</tr>
<tr>
<td>WC_defaulthost_secure</td>
<td>9443</td>
</tr>
<tr>
<td>PROXY_HTTP_ADDRESS</td>
<td>80</td>
</tr>
<tr>
<td>PROXY_HTTPS_ADDRESS</td>
<td>443</td>
</tr>
<tr>
<td>ORB_LISTENER_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>WC_adminhost</td>
<td>9060</td>
</tr>
<tr>
<td>WC_adminhost_secure</td>
<td>9043</td>
</tr>
<tr>
<td>SOAP_PROXY_CONNECTOR_ADDRESS</td>
<td>8882</td>
</tr>
</tbody>
</table>

*Figure 5-7  odr2’s ports*

- Make sure you can access the administrative console through all ODRs in your topology.
- Initiate a failover by stopping the active deployment manager. We recommend verifying that all nodes are synchronized before performing this test. Verify that the console is accessible after the failover occurred. Be aware that it might take a while until the failover was successful—this is mainly due to the time it takes for the shared file system to release the lock that the active deployment manager holds in the shared file system.
5.2.2 Creating a new cell with an HA deployment manager

This scenario assumes that you want to set up a new cell. No configuration is done yet, and you need to use the following steps to configure your HA deployment manager environment:

1. Install WebSphere Application Server Network Deployment and WebSphere Extended Deployment on all systems. The HA deployment manager systems must have access to a shared file system. For the deployment manager systems, you can install the WebSphere binaries locally or on the shared file system.

2. Create a deployment manager profile on the first deployment manager system. The profile must reside on the shared file system.

3. Start the deployment manager.

4. Create a custom profile on the ODR system and federate it with the deployment manager.

5. Use `wsadmin` or the administrative console to create and configure an ODR on the federated node.

6. Create a deployment manager profile on the second deployment manager system.

7. Run the `xd_hadmgrAdd` command within the deployment manager profile on the second deployment manager system to convert it into a standby deployment manager.

8. Restart the deployment managers and the ODRs. The first deployment manager that starts is the active deployment manager, and the other is the standby deployment manager.

5.2.3 Converting existing cell to HA deployment manager configuration

We have not tested this scenario, but following are the steps that must occur in order to convert an existing cell configuration:

1. Ensure that you have a shared file system set up that both deployment manager systems can access.

2. Install WebSphere Application Server Network Deployment and WebSphere Extended Deployment on the shared file system. Alternatively, keep the WebSphere binaries locally and create only the deployment manager profile on the shared file system.
3. Create a deployment manager profile using the first deployment manager's system information (such as IP address) and with the same cell and node name as the existing deployment manager. The profile must be created on the shared file system.

4. Use the `backupConfig` and `restoreConfig` command line utilities to relocate the existing cell configuration to the shared file system.

5. Ensure that the existing cell configuration contains an ODR. If an ODR does not exist, create one.

6. Create a deployment manager profile using the second deployment manager system information.

7. Run the `xd_hadmgrAdd` command within the second deployment manager profile to convert it into a standby deployment manager.

8. Restart the deployment managers and ODRs. The first deployment manager that starts is the active deployment manager, and the other is the standby deployment manager.

### 5.2.4 Removing an HA deployment manager from a cell

The `xd_hadmgrRemove` command removes a highly available deployment manager from a cell. The command must be run within the deployment manager profile that is being removed.

The command syntax is as follows:

```
xd_hadmgrRemove -host_name (primary_dmgr_host) [-port (primary_dmgr_port)]
[-user (uid)] [-password (pwd)] [-quiet] [-logfile (filename)]
[-replacelog] [-trace] [-help]
```

Example 5-3 shows the output from the command window when removing the HA deployment manager from the configuration.

#### Example 5-3  `xd_hadmgrRemove` command for dmgr2

```
/opt/IBM/WebSphere/AppServer/profiles/dmgr2/bin >./xd_hadmgrRemove.sh -hostname
dmgr1
ADMU0116I: Tool information is being logged in file
/opt/IBM/WebSphere/AppServer/profiles/dmgr2/logs/xd_hadmgrRemove.log
XHAD1001I: Connecting to primary deployment manager host dmgr1 using port 8879
XHAD1007I: Begin removal of standby deployment manager node dmgr2 from cell
xdcell
ADMU2018I: Node dmgr2 has been removed from the Deployment Manager configuration.
```

The `xd_hadmgrRemove` command reconfigures all settings that were changed when adding the HA deployment manager, as explained in “Configuration
changes performed by xd_hadmgrAdd command” on page 77. For example, the WC_adminhost and WC_adminhost_secure HTTP transport channels and the JMX_SOAP_PROXY_ADDRESS endpoint are set back to their original status or removed. Also, after removing the second deployment manager you can no longer access the administrative console through the ODRs.

5.3 Adding WebSphere nodes to the cell

A node is a logical grouping of servers. A node usually corresponds to a logical or physical computer system with a distinct IP host address. Nodes cannot span multiple computers although multiple nodes can exist on one computer.

The process discussed in this section was used to add nodes XDNODE1, XDNODE2, and XDNODE3 in the sample topology. See 3.1, “The sample topology” on page 26. These nodes will host WebSphere application servers.

5.3.1 Creating and federating the nodes

Nodes are added to a cell by creating a custom profile on the host system for the node and federating it to the cell.

1. Launch the profile management tool using the appropriate command for your platform.
   - On Windows systems:
     `app_server_root\bin\ProfileManagement\pmt.bat`
   - On UNIX: `app_server_root/bin/ProfileManagement/pmt.sh`
2. To create a new profile, select Create.
3. The next panel is a Welcome window. Click Next.
4. Select WebSphere Extended Deployment Operations Optimization as the environment. Any other selection results in a profile that has not been augmented for Extended Deployment.
5. Click Next.
6. Select Custom profile as the profile type, and click Next.
7. Select Advanced profile creation, and click Next.
8. Enter the name for the new profile and the directory to store the profile in, and click Next.
9. Enter the name for the new node and the host name of the system.
10. Click Next.
11. The next panel asks if you want to federate now or later. The deployment manager must be up and running in order to federate the new node to the cell.

Leave the **Federate this node later box** unchecked (allow the federation to happen now).

Enter the host name or IP address of the deployment manager and the SOAP port number for the deployment manager.

If you enabled administrative security, enter the user ID and password you specified when creating the deployment manager profile.

12. The next panel allows you to specify port values to use. Ports must be unique on the system. The profile management tool selects a set of ports for this profile unique to the WebSphere installation.

13. Review the profile summary, and click **Create**.

### 5.3.2 Starting the node

To start the node on distributed systems, open a command prompt, and type the following command:

- **Windows**: `profile_home\bin\startNode`
- **UNIX and z/OS**: `profile_home/bin/startNode.sh`

To start a node agent on z/OS using the START command, use the following format:

- `START nodeagent_procname, JOBNAME=server_shortname, ENV=cell_shortname.node_shortname.server_shortname`
Working with other middleware servers

This chapter discusses how to configure middleware servers with assisted lifecycle support to WebSphere Extended Deployment.

Assisted lifecycle support applies to the following servers:
- Apache HTTP Server
- Apache Tomcat
- Apache Geronimo
- JBoss
- BEA WebLogic Server
- WebSphere Application Server Community Edition

Because the PHP server is a special case of a non-WebSphere server with full lifecycle support, we address it here also.

This chapter includes the following topics:
- Working with assisted lifecycle servers
- Working with PHP servers
6.1 Working with assisted lifecycle servers

Assisted lifecycle servers have significant support in WebSphere Extended Deployment, but do not have the full range of support. Servers are created and applications are deployed using the middleware native tools and then are defined to WebSphere.

Configuring middleware servers with assisted lifecycle support on WebSphere Extended Deployment involves the following steps:

1. Install the server product, configure the server(s), and install applications to it.
   This occurs outside the realm of WebSphere Extended Deployment and is not discussed in this book.
2. Install the middleware agent.
   The middleware agent is a light-weight agent, which you can install on the nodes that you want WebSphere Extended Deployment to manage. No other WebSphere Extended Deployment components are required on other middleware servers. It can be installed to the systems from the deployment manager using the centralized installation manager.
   The middleware agent replaces the remote agent used in V6.0.
   This step is discussed in section 6.1.1, “Installing the middleware agent” on page 86.
3. Define the server(s) to the WebSphere configuration.
   This step makes the cell aware of the server and allows manageability of it. It is discussed in section 6.1.4, “Defining the server to the WebSphere configuration” on page 94.
4. Create a dynamic cluster for the server(s).
   You can now manage the servers and configure dynamic clusters from WebSphere.
   Creating dynamic clusters is discussed in Chapter 8, “Working with dynamic clusters” on page 147.
5. Define the existing applications. This step is discussed in Chapter 9, “Managing applications” on page 175.

6.1.1 Installing the middleware agent

The middleware agent can be installed locally on the system from the Extended Deployment media, or it can be installed remotely from the deployment manager using the centralized installation manager. Using the centralized installation
manager can significantly reduce installation time and enables simplified administration.

This chapter focuses on using centralized installation manager to install the agent. The process is the same, regardless of the server type to which you are installing the middleware agent.

**Note:** Installation of a middleware agent will not only install the binaries for the product, but will create a new node and federate it to the deployment manager. The federation process requires the following:

- The date / time on the deployment manager host operating system, and the remote operating system must be synchronized to within 5 minutes.
- The deployment manager must be able to communicate with the remote node and must be able to resolve the host name of the remote node to an IP address.
- The remote node must be able to communicate with the deployment manager node and must be able to resolve the host name of the deployment manager node to an IP address.

Before proceeding, make sure that these conditions are met.

The following process describes the use of centralized installation manager to install the agent.

1. Log in to the administrative console for the cell.
2. Define the new host as a deployment target.
   - Navigate to **System administration → Centralized Installation Manager → Installation targets**.
3. Click **Add Installation Target**.
4. Enter the information required to define the target host.
   - Host name: Used to access the remote system
   - User name: The administrative ID of the target host for use when installing or uninstalling packages.
   - Password
5. Click **OK** to add the new target. The new target will now be displayed in the list of targets.
6. Next, navigate to **System administration → Centralized Installation Manager → Available installations**.
7. Select **Product install** as the package type and the **WebSphere Extended Deployment middleware agent** as the installation package.

![Available Installations](image)

*Select a package type*

- Product install

*Select an installation package*

- WebSphere Extended Deployment middleware agent for non WebSphere servers - 6.1

Show installation targets  Show uninstallation targets

**Figure 6-1  Select installation package**

8. Click **Show installation targets**. The panel expands to show the available installation target hosts. Select the hosts to install to, and click **Install**.

9. A wizard will start to collect the information and perform the install. Accept the license agreement, and click **Next**.

10. Select the authentication method (Figure 6-2).

![Authentication Method](image)

*Select an authentication method to access the installation targets.*

- Use user name and password.
- Use Secure Shell (SSH) public/private key authentication.

**Figure 6-2  Select authentication method**

If you select user name and password, the next panel allows you to enter the values for them. These fields are pre-filled with the user ID and password stored for each target host selected.

If you select SSH, the next panel asks for the SSH private key location and keystore password.

In this example, user name and password were entered. In the next panel the proper ID and password were entered.

Click **Next**.
11. The next panel asks you to enter the directory location for the middleware installation (Figure 6-3).

![Location Settings]

*Figure 6-3  Select the installation location*

12. The next panel allows you to disable the prerequisite check. Unless you are experiencing problems with the installation and understand the consequences of disabling this feature, leave it unselected, and allow the prerequisite check to take place.

   Click **Next**.

13. The next panel (Figure 6-4 on page 90) shows a summary of the installation options.
Review the settings and click **Finish**.

14. Messages appear on the console stating that the request for installation was submitted. You will be returned to the initial panel.

15. Select **Installations in progress** to monitor the installation. (Figure 6-5).

   If your console preference is to automatically refresh the workspace, you will see the status change as the installation progresses. If not, click the Status refresh icon to check the status.
The stages of installation are as follows:

- Transferring files: the middleware agent files are being transferred to the host.
- Installation in progress: the software is being installed.
- Running post-installation commands: The new node representing the installation is started and synchronized to the deployment manager.

When the installation is complete the target disappears from the list of installations in progress.

16. You are able to see the new middleware nodes, including their status, by navigating to **System administration → Middleware nodes**.

17. Navigate to **System administration → Centralized Installation Manager → Installation history**. This view shows the status of the installation.

When you install the middleware agent, two steps are performed: the installation of the middleware agent code and the post configuration, which includes adding the new node to the cell. If the status of the installation is Succeeded, that indicates that both steps completed successfully.

<table>
<thead>
<tr>
<th>Select</th>
<th>Host Name</th>
<th>Operation</th>
<th>Package and features</th>
<th>Creation time</th>
<th>Completion time</th>
<th>Completion status</th>
</tr>
</thead>
</table>

*Figure 6-6  Installation history*

If either step fails, click **View details** to review the logs.

If the status is **Failed**, that means that the installation failed, and you should perform the following:

a. Click the log.txt file to review the install.
b. Correct the problem.
c. Retry the installation.

If the status indicates that the installation Succeeded, but the configuration failed, complete the following tasks:

a. Click the addAgent.log file to review the federation process.
b. Correct the problem. This is most often a failure to be able to contact the deployment manager host from the target host.
c. Logon to the target host, and uninstall the middleware agent using `agent_home/uninstall_xd_operations/uninstall.exe`.
d. Delete the `agent_home` directory.
e. Retry the installation.

6.1.2 Starting the middleware agent

**Important:** Do not start servers natively from the operating system automatically or manually. For example, if the server is defined as a Windows service, change the startup property to “manual”. The server must be started from WebSphere, either using the `startMWServer` command from the `agent_home/bin` directory or from the WebSphere administrative console for WebSphere to be able to manage it.

You can start the agent remotely from the administrative console or locally using the `startAgent` command.

To start the agent from the administrative console, perform the following steps:

1. Select **System Administration → Middleware nodes**.
2. Check the box to the left of the node.
3. Select **Start agent** in the **Select operational action** drop-down.
4. Click **Run**.
5. Enter a user ID and password to be used to execute the command under on the remote machine. The user ID must have admin or execute privileges for the command.
6. Click **Ok**.

To start the agent from the middleware system, perform the following steps:

1. Open a command window, and change the directory to the `agent_home/bin` directory.
2. Enter **startAgent** (Windows) or **startAgent.sh** (Unix). See Figure 6-7 for an example.

![startAgent](C:%WebSphere%XD%bin>startAgent)

CWXDA0033I: Launching the agent, check logfile
C:%WebSphere%XD%logs%middlewareagent\startServer.log for launch errors.
CWXDA0018I: Tool actions are being logged in file
C:%WebSphere%XD%logs%middlewareagent\startServer.log.
CWXDA0034I: Agent launched successfully. Check logs for messages.

*Figure 6-7  Starting a middleware agent*

### 6.1.3 Checking the status of the middleware agent

You can view the status of the middleware agent remotely from the administrative console or locally using the **mwServerStatus** command.

To view the status from the administrative console, perform the following steps:

1. Select **System Administration → Middleware nodes**.
2. The status will appear as an icon in the Status column. Hold the cursor over the icon to see a pop-up that describes the status.

To view the status from the middleware system, perform the following steps:

1. Open a command window, and change the directory to the `agent_home/bin` directory.
2. Use the **mwServerStatus -server middlewareagent** (Windows) or **mwServerStatus.sh -server middlewareagent** (Unix) command. See Figure 6-8 for an example.

![mwServerStatus](C:%WebSphere%XD%bin>mwServerStatus -server middlewareagent)

CWXDA0018I: Tool actions are being logged in file
C:%WebSphere%XD%logs%middlewareagent\status.log.
CWXDA9014I: Invoking action status on Server middlewareagent. Check
C:%WebSphere%XD%logs%middlewareagent\status.log file for log messages.
CWXDA9011I: **Server middlewareagent type XDAGENT on node XDTomcat1 cell XDcell1** is running.

*Figure 6-8  Viewing the status of the middleware agent*
If the middleware server is not running you will see the following message:

```
CWXDA9013E: Error invoking action on Server middlewareagent type XDAGENT on node XDTomcat1 cell XDcell1.
```

*Figure 6-9  Status message when agent is not running*

### 6.1.4 Defining the server to the WebSphere configuration

The next step is to define the new server to the cell.

1. Navigate to **Servers → Add a server**.

   ![Create new middleware server](image)

   *Figure 6-10  Add a server*

2. You have two options.
   
   - You can create a new server. This can be any type that has full lifecycle support.
   
   - You can add an existing server. This can be any type of server that has assisted or generic lifecycle support.

   Since the middleware servers are already created, select the option to **Add an existing server**, select the server type, and click **Next**.
3. Select the node from the drop-down, enter the server name, and click **Next**.

4. Select a template. Templates are provided for the supported versions of the server type. You can also choose from any user-defined templates for that server type. Click **Next**.

5. Confirm the selections, and click **Finish**.

6. Perform any additional configuration activities required for the specific installation, such as modifying environment variables.

   If you plan to add similar servers in the future, create a template from this server.

### 6.1.5 Managing the server

You can perform certain management and administrative functions for other middleware servers.

**Note:** Each middleware type has platform descriptor properties defined. You can see these in the administrative console by navigating to **System administration → Middleware descriptors → middleware_platform_name**. Review the descriptors that apply to your middleware server before using the server.

Management functions include the following:

- Starting the server
- Stopping the server
- Terminate the server (immediate stop)
- Putting the server into and out of maintenance mode
- Act on runtime tasks generated for the server

Use the following steps to perform these functions:

1. Navigate to **Servers → All servers**. Alternatively, you can go to **Servers → server_type**. Both views have options for managing servers.

2. Check the box to the left of the server, and click the appropriate button.
Administrative functions include the following:

- Viewing the server logs
- Viewing and changing configuration files.

Use the following steps to perform the administrative functions:

1. Click the server name in the list of servers.
2. To view the server logs, click the **Log Viewer** tab. Select the log from the pull-down menu, and click **Retrieve**.
3. To view and change configuration files, click the **External Configuration** tab. Select the configuration file from the pull-down, and click **Retrieve**. To alter the file, type the change into the window, and click **Apply**.

### 6.1.6 Apache Tomcat servers

The first step in integrating Tomcat servers to the Extended Deployment environment is to install and configure the Tomcat environment. This is done external to Extended Deployment and is not discussed here.

After the Tomcat environment is installed and configured, steps need to be performed to integrate the Tomcat servers into the WebSphere Extended Deployment environment. The sample topology illustrates this process.

The following two hosts in the sample topology contain Apache Tomcat installations and are integrated into the Extended Deployment cell.

- XDTomcat1
- XDTomcat2
Install the middleware agent
Before installing the middleware agent, ensure that the network connectivity is operating. In the sample topology, ping was used to ensure that the host with the deployment manager could access the two systems xdtomcat1 and xdtomcat2.

Using the process in section 6.1.1, “Installing the middleware agent” on page 86, install the middleware agent on the remote server where Tomcat is installed.

Start the new middleware agent using the instructions in section 6.1.2, “Starting the middleware agent” on page 92.

Figure 6-13 shows the newly installed nodes for the sample topology Tomcat systems.

Figure 6-13   Tomcat nodes

Add the Apache Tomcat server as a middleware server
The process to add an existing Tomcat server is as follows:

1. Use the process outlined in section 6.1.4, “Defining the server to the WebSphere configuration” on page 94 to add an existing Tomcat server to the configuration.
   - Select the option to Add an existing server, and select Apache Tomcat server.
   - The name that you select for the server can be any name you choose. It does not have to match anything on the Tomcat server system.
– Choose the template that corresponds to the Tomcat version you have installed. The template options are tomcat4x and tomcat5x.

In the sample topology, the following Tomcat servers were defined:

![Newly defined Tomcat servers and nodes](image)

Figure 6-14 Newly defined Tomcat servers and nodes

2. Click the server name to open the configuration page. Confirm that the default settings are correct for your installation. If they are not, correct them.

   **Tip:** If you have to modify the settings, consider creating a template of the existing server after you make the modifications. You can use this template instead of the default when you create future servers, ensuring that the settings are automatically set correct.

3. Update the WebSphere variables so that they match the settings on the node that is running the Apache Tomcat server.

   In the administrative console, click **Environment → WebSphere Variables**.

4. Select the cell scope, and view the properties for the CATALINA_HOME variable. This variable specifies the home directory of your Apache Tomcat installation.

   If the default value does not match your installation, edit the variable settings. If you have multiple installations and the home directory varies across nodes, create a CATALINA_HOME variable at the node level for each node.

   In the sample topology, the variables were set as shown in Figure 6-15 on page 99.
5. Save the configuration.

6. If the middleware agent is not started, start it now and verify that the server status displayed in the console is correct.

7. If you started the Tomcat server manually on the Tomcat system, stop it now.

8. Start and stop the Tomcat server from the administrative console to ensure that the server was defined to WebSphere correctly and can be managed from the console.

**External files available to the administrative console**
The following logs are configured by default and can be viewed from the administrative console:

- `${CATALINA_HOME}/logs` directory
- `${AGENT.HOME}/logs/${WAS_SERVER_NAME}` directory

The following configuration files are included by default:

- `${CATALINA_HOME}/conf/server.xml`
- `${CATALINA_HOME}/conf/catalina.policy`
- `${CATALINA_HOME}/conf/web.xml`

### 6.1.7 JBOSS servers

The first step in integrating JBoss servers to the Extended Deployment environment is to install and configure the JBoss environment. This is done external to Extended Deployment and is not discussed here.
After the JBoss environment is installed and configured, integrate the JBoss servers into WebSphere Extended Deployment environment. The sample topology illustrates this process.

The following two hosts contain JBoss installations and will be integrated into the Extended Deployment cell.

- XDJBOSS1
- XDJBOSS2

**Install the middleware agent**

Before installing the middleware agent, ensure that the network connectivity is operating. Ping was used to ensure that the host with the deployment manager could access the two systems xdjboss1 and xdjboss2.

Using the process in section 6.1.1, “Installing the middleware agent” on page 86, install the middleware agent on the remote server where Jboss is installed.

Start the new middleware agent using the instructions in section 6.1.2, “Starting the middleware agent” on page 92.

Figure 6-16 shows the newly installed nodes for the sample topology JBoss systems.
Add a JBoss server as a middleware server
The process to add an existing JBoss server is as follows:

1. Use the process outlined in section 6.1.4, “Defining the server to the WebSphere configuration” on page 94 to add an existing JBoss server to the configuration.

   Select the option to **Add an existing server**, select **JBoss server**, and click **Next**.

   **Note:** The name that you select for the server **must match** the server name on the JBoss installation. This name is also used to set the value of the WAS_SERVER_NAME variable at the server scope.

2. The new server definition should appear in the Middleware servers list. Navigate to **Servers → Other middleware servers → JBoss servers** to view the new definition.

   In the sample topology, the following JBoss servers were defined:

   ![Figure 6-17 Newly defined JBoss servers and nodes](image)

3. Click the server name to open the configuration page. Confirm that the default settings are correct for your installation. If they are not, correct them.

   **Tip:** If you had to modify the settings, consider creating a template of the existing server. You can use this template instead of the default when you create future servers, ensuring that the settings are automatically set correct.

4. Update the WebSphere variables so that they match the settings on the node that is running the JBoss server.

   In the administrative console, click **Environment → WebSphere Variables**.
5. View the properties for the following variables. If the default value does not match your installation, edit the variable settings.

- **JBOS_HOME**

  Defined at the cell scope. If you have multiple installations and the home directory varies across nodes, create a new variable by the same name at the node level for each node.

  Specifies the home directory of your JBoss installation. Usually, the default installation location includes the JBoss version, so it is likely that you need to update the default values.

  The value used for the sample topology is shown in Figure 6-18.

  ![JBOS_HOME variable](image)

  **Figure 6-18  JBOSS_HOME variable**

- **JBOS_PROFILE and WAS_SERVER_NAME**

  Defined at the server scope when you define the JBoss server to WebSphere.

  JBOSS_PROFILE specifies the configuration set in use for the JBoss server. The default value is ${WAS_SERVER_NAME}.

  Make sure the WAS_SERVER_NAME variable matches the server name on the JBoss system.

  Figure 6-19 shows the server level variables for the JBoss server. The server name is “default”, matching the server name on the JBoss system.

  ![Server scope variables for JBoss servers](image)

  **Figure 6-19  Server scope variables for JBoss servers**

- **JAVA_HOME**

  (Windows systems) Define this variable at the node or server scope. This is required for JBoss start operations on Windows. See Figure 6-20 on page 103 for an example.
6. Save the configuration.

7. If the middleware agent is not started, start it now, and verify that the server status displayed in the console is correct.

8. Navigate to **Servers → Other middleware servers → JBoss servers → jboss_server**.

9. Click the Configuration tab, if it is not already selected, and then click **Server operations** (underneath Additional Properties).

   ![Server operations for JBoss servers](image)

10. Click the start and stop links to edit the start and stop server operations. Here you can enter the user name and password, if required, to run these commands. Verify that variables used in these operations are defined.

11. Start and stop the server from the administrative console to ensure that the server was defined to WebSphere correctly and can be managed from the console.

**External files available to WebSphere**

The following logs are configured by default and can be retrieved from the administrative console:

- Logs in the `{JBOSS_HOME}/server/server_name` directory
- Logs in the `agent_home/logs/server_name` directory

Configuration files in the following locations are available from the administrative console:

- `{JBOSS_HOME}/server/server_name/conf` directory
- `{JBOSS_HOME}/server/server_name/deploy` directory
6.1.8 WebSphere Application Server Community Edition

The first step in integrating WebSphere Application Server Community Edition servers to the Extended Deployment environment is to install and configure the Community Edition environment. This is done external to Extended Deployment and is not discussed here.

After the Community Edition environment is installed and configured, integrate the Community Edition servers into the WebSphere Extended Deployment environment. The sample topology illustrates this process.

The following two hosts contain Community Edition installations and are integrated into the Extended Deployment cell.

- XDWASCE1
- XDWASCE2

**Install the middleware agent**

Before installing the middleware agent, ensure that the network connectivity is operating. In the sample topology, ping was used to ensure that the host with the deployment manager could access the two systems xdwasce1 and xdwasce2.

Using the process in section 6.1.1, “Installing the middleware agent” on page 86, install the middleware agent on the remote server where Community Edition is installed.

Start the new middleware agent using the instructions in section 6.1.2, “Starting the middleware agent” on page 92.

Figure 6-22 on page 105 shows the newly installed nodes for the sample topology Community Edition systems.
Adding Community Edition as a middleware server

The process to add an existing WebSphere Application Server Community Edition server is as follows:

1. Use the process outlined in 6.1.4, “Defining the server to the WebSphere configuration” on page 94 to add an existing Community Edition server to the configuration.

Select the option to Add an existing server and select WebSphere Application Server Community Edition server.

The server name does not have to match anything on the Community Edition installation.

In the sample topology, the Community Edition servers in Figure 6-23 on page 106 were defined:
2. Click the server name to open the configuration page. Confirm that the default settings are correct for your installation. If they are not, correct them.

**Tip:** If you had to modify the settings, consider creating a template of the existing server. You can use this template instead of the default when you create future servers, ensuring that the settings are set correctly automatically.

3. Update the WebSphere variables so that they match the settings on the node that are running the Community Edition server.

   In the administrative console, click **Environment → WebSphere Variables**.

4. Select the cell scope and view the properties for the following variables.

   If the default value does not match your installation, edit the variable settings.

   If you have multiple installations and the home directory varies across nodes, create a new variable by the same name at the node level for each node.

   - **GERONIMO_HOME**

     Specifies the home directory of your WebSphere Application Server Community Edition installation. Default values are set at the cell scope level.

     The value used for the sample topology is shown in Figure 6-24 on page 107.
5. Save the configuration.

6. If the middleware agent is not started, start it now, and verify that the server status displayed in the console is correct.

7. Navigate to Servers → Other middleware servers → WebSphere Application Server Community Edition servers → server_name.

8. Click the Configuration tab if it is not already selected, and then click Server operations (underneath Additional Properties).

9. Click the start and stop links to edit the start and stop server operations. Here you can enter the user name and password, if required, to run these commands. Figure 6-26 on page 108 shows the properties for the start command.
10. Start and stop the server from the administrative console to ensure that the server was defined to WebSphere correctly and can be managed from the console.

**External files available to WebSphere**

The following logs are configured by default and can be viewed from the administrative console:

- `${GERONIMO_HOME}/var/log` directory
- `${AGENT_HOME}/logs/${WAS_SERVER_NAME}` directory

The following configuration files are included by default:

- `${GERONIMO_HOME}/conf/config.xml`
6.2 Working with PHP servers

PHP servers have full lifecycle support in WebSphere Extended Deployment. They can be created on nodes that are running WebSphere Extended Deployment or the middleware agent.

Because PHP is a non-WebSphere product, the process of integrating a PHP node to the cell is similar to that of other non-WebSphere nodes. However, while assisted lifecycle servers are created separate from WebSphere and then defined to the cell, PHP servers can be created directly from WebSphere.

The first step in integrating PHP servers to the Extended Deployment environment is to install and configure the PHP environment. This is done external to Extended Deployment and is not discussed here.

The sample topology illustrates how to integrate the PHP installations into WebSphere Extended Deployment environment.

The following two hosts in the sample topology contain PHP installations.

- XDPHP1
- XDPHP2

6.2.1 Install the middleware agent

Note: If PHP is installed on a system that has a WebSphere node, you do not have to install the middleware agent. You can create a new server instance for PHP on an existing node. If PHP is installed on a server without a WebSphere presence, you need to install the middleware agent.

Before installing the middleware agent, ensure that the network connectivity is operating. In the sample topology, ping was used to ensure that the host with the deployment manager could access the two systems xdp1 and xdp2.

Using the process in section 6.1.1, “Installing the middleware agent” on page 86, install the middleware agent on the remote server where PHP is installed.

Start the new middleware agent using the instructions in section 6.1.2, “Starting the middleware agent” on page 92.

Figure 6-27 on page 110 shows the newly installed nodes for the sample topology PHP systems.
6.2.2 Verify the PHP install location in the middleware descriptor

Prior to adding a PHP server as a middleware server, use the following steps to verify that the middleware descriptors reflect the correct location of the PHP runtime:

1. In the administrative console, click System administration → Middleware descriptors → `middleware_platform_name` → phpRuntime.
2. Click the name of the server (underneath version).
3. Verify the installation location for the PHP server.
4. Return to the Middleware Platform Descriptors page, and click **apacheWebServerRuntime** to open the configuration page. Verify the installation location for the Apache HTTP Server.

### 6.2.3 Create a PHP server

**Note:** PHP servers to be used in dynamic clusters are created when you create the dynamic cluster. You would only use the process described in this section if a) you wanted to create a server template, or b) if you wanted to use a PHP server outside of a dynamic cluster.
The process to add an existing PHP server is as follows:
1. Navigate to Servers → Add a server.
2. Select the option to Create a new server instance, select PHP server, and click Next.
3. Select the node where the server will be created, and enter a name for the server.
4. Select the Apache runtime that matches your installation, and click Next.
5. Select a PHP runtime that matches your installation, and click Next.
6. Select a server template that corresponds to the PHP server version you have installed or a user-defined template.

![Select a PHP template](image)

*Figure 6-31  Select a template*

7. Click **Finish** to create the server.

![Newly defined PHP server](image)

*Figure 6-32  Newly defined PHP server*

8. Click the server name to open the configuration page. Confirm that the default settings are correct for your installation. If they are not, correct them.

**Tip:** If you had to modify the settings, consider creating a template of the existing server. You can use this template instead of the default when you create future servers, ensuring that the settings are set correctly automatically.

9. Verify the following PHP related WebSphere variable to make sure it is defined correctly for the PHP installation.

   If the default value does not match your installation, edit the variable value setting. If you have multiple installations and the value varies across nodes or servers, create a new variable by the same name at that scope for each node...
In the administrative console, click **Environment → WebSphere Variables**.

- **APACHE_SERVERROOT**
  
  Defined at the cell scope.
  
  Specifies the installation root of the Apache installation. It is used in the start server operation for the PHP server.

10. Save the configuration.

11. If the middleware or node agent is not started, start it now and verify that the server status displayed in the console is correct.

12. Navigate to **Servers → Other middleware servers → PHP servers → server_name → Server operations**.

13. Edit the start and stop server operations for the server to include the user name and password, if required, to run these commands.

14. Start and stop the server from the administrative console to ensure that the server was defined to WebSphere correctly and can be managed from the console.
Chapter 7. Working with ODRs and autonomic managers

The on demand router (ODR) represents the entry point for HTTP traffic into your environment. This chapter discusses how to implement and manage one or more ODRs in a WebSphere Extended Deployment environment. It also discusses configuration options for the autonomic request flow manager (ARFM), the application placement controller (APC), and the dynamic workload manager. Before creating and configuring the ODRs, you should plan your installation using the information in section 4.4, “ODR planning considerations” on page 41.

This chapter includes the following topics:

- Creating an on demand router
- Configuring an ODR
- Proxy plug-in generation
- Caching in the ODR
- Configuration of custom error pages
7.1 Introduction

The ODR is responsible for orchestrating the queuing and dispatching of requests in accordance with your service policy. In optimizing queue lengths and dispatch rates, several factors are considered, including the processing capacity of the nodes, performance goals, relative importance, and load balancing.

The ODR is a component that logically replaces and extends the functionality of the WebSphere Web server plug-in. However the ODR can and often does work in concert with existing Web servers that use the WebSphere Web server plug-in.

The ODR provides the standard functionality of an HTTP/1.0 and HTTP/1.1 compliant proxy server with the following added on demand features:

- Request classification and prioritization
- Request queuing
- Request routing
- Dynamic load balancing
- HTTP session affinity
- SSL ID affinity
- WebSphere Partitioning Facility partition affinity

**Important:** Because the ODR is the entry point into your application server environment it is extremely important that you make the ODR highly available by using multiple ODRs.

ODRs run the ARFM gateway for HTTP requests. The gateway intercepts and queues requests. The request flows are managed to achieve the best balanced performance results, considering the configured service policies and the offered load.

The ODR uses session affinity for routing work requests. After a session is established on a server, subsequent requests for the same session go to the original server, maximizing cache usage and reducing queries to back-end resources.
There can be multiple ODRs in a topology. Each request goes through only one ODR, but for environments with more than one ODR, the initial request could be routed to any one of them.

The ODR is aware of its environment because of a component called the on demand configuration service (ODC). The ODC collects information about all of the WebSphere Extended Deployment and WebSphere Network Deployment application servers and applications that are deployed in the cell. It also gathers information about the defined non-WebSphere middleware servers and middleware applications. This information is used by the ODR to dynamically route HTTP requests to both WebSphere and non-WebSphere servers. The ODC dynamically configures the routing rules at runtime.

The ODR provides routing of HTTP, SOAP, and SIP traffic to the following items:

- Dynamic clusters
- Static clusters
- Generic server clusters (deprecated)

**Note:**
- Generic server clusters are deprecated in V6.1.
- IIOP and JMS traffic bypass the ODR.

### 7.2 Creating an on demand router

When you create an ODR, you are actually creating an intelligent HTTP proxy server. The ODR must belong to a federated node.

There are two ways to create an ODR:

- Using the administrative console
- Using a wsadmin script

**Creating an ODR using the administrative console**

In order to create an ODR using the administrative console, do the following:

1. Start the ODR node.
2. Navigate to **Servers → On Demand Routers**.
3. Click **New**.
4. Select the node on which you want to install the ODR (for example: xdodr1).
5. Provide a name for the ODR server (for example: odr1).
6. Click **Next**.
7. Select the supported protocols. Choose HTTP, SIP, or both.
8. Select to **Generate Unique HTTP Ports**, and click **Next**.
9. Select the **Server template**, and click **Next**.
10. Review the summary, and click **Finish**.
11. Make sure you save your configuration and synchronize with nodes.

**Creating an ODR using a wsadmin script**

You can create the ODR using a wsadmin script.

1. Start the ODR node.
2. Update the createodr.jacl script with the name you want to use for the ODR.
   
   The `createodr.jacl` script is located on the deployment manager node in the `app_server_home/bin` directory.

   ```bash
   ./wsadmin.sh -f createodr.jacl
   ODRNode
   
   This command creates an ODR on node `ODRNode`.
   
   **Using a name other than “odr”**: When you want to create an ODR server with a different name than the default of odr, you need to change the `createodr.jacl` script. Make sure you have a backup copy of the original script.

   You must make the following changes:
   1. Search for the following line and change the odr name:
      ```java
      set serverName "odr"
      
      You will find this line twice.
      
      2. Do a global change on the file to change all instances of `-name odr` to `-name your_new_name`.
      
     3. Execute the script as follows:
        ```bash
        ./wsadmin.sh -f createodr.jacl ODRNode
        
        This command creates an ODR on node `ODRNode`.
        ```

7.3 Configuring an ODR

To change an ODR configuration, log on to the administrative console and go to **Servers → On Demand Routers**. Figure 7-1 on page 119 shows the ODR’s overview page:
Click an ODR name to change its configuration. In addition to the configuration options for the ODR server, you will see a set of ODR-specific settings, shown in Figure 7-2 on page 120.
The basic On Demand Router settings are described in the following section.

### 7.3.1 On Demand Router settings

Basic ODR settings can be viewed and modified by expanding the **On Demand Router Properties** section and clicking on **On Demand Router settings** (circled in Figure 7-2).

The On Demand Router settings configuration page contains a number of settings.

- Content Server Connection
The Content Server Connection settings allow you to configure connections between the ODR and application servers. Using this pane, you can also configure SSL secured connections. When doing so, we recommend that you first create a new SSL alias under Security → SSL → New JSSE repertoire.

- **Caching**

  Enables static and dynamic caching in the ODR. See section 7.5, “Caching in the ODR” on page 127 for information.

- **Compression policy**

  Selecting this indicates that the ODR should compress the response sent to the client using the selected response compression type (gzip only, deflate only, auto).

- **Exclusions**

  You can disable certain HTTP request methods using the Exclusions pane. If a requested HTTP method matches any of the excluded methods, the ODR rejects the request with a METHOD DISALLOWED error.
7.4 Proxy plug-in generation

When using the classic topology with a Web server in the DMZ, you must generate a special Web server plug-in. This special plug-in routes incoming traffic to the ODRs instead of directly to the application servers—which is what happens when you use the default Web server plug-in. This special plug-in replaces the plug-in that is usually automatically generated for the Web servers.
The name of this plug-in file is identical to the normal Web server plug-in file, which is plugin-cfg.xml.

The new plugin-cfg.xml is automatically created by the ODR every time a configuration change happens that affects the file, such as adding, deleting, modifying applications or when the number of ODRs changes.

Following is where the new plug-in file is located on the ODR system:

profile_home/etc/plugin-cfg.xml

This file must be copied to the Web servers afterwards. In the ODR properties, you can point to a user supplied script that copies the plug-in file to the appropriate directory on the Web servers.

**Note:** The ODR that is configured to generate plugin-cfg.xml files must be running for dynamic updates to occur.

Following are the steps required to configure the proxy plug-in:

- Disable automatic plug-in generation on the Web servers.
- Create a script to propagate the plug-in.
- Configure the ODR proxy plug-in configuration policy.

### 7.4.1 Disable automatic plug-in generation on the Web servers

If your Web servers are running on managed nodes (that is, a node agent is present on the Web server system) or if you are using the special case of the IBM HTTP Server V6.0 running on an unmanaged node, which supports the same functions as when using a managed node, then you must disable the automatic generation and propagation of the Web server plug-in. This makes sure that you use the ODR’s plug-in file rather than the Web server plug-in file.

The reason for this is that the Web servers automatically update their plugin-cfg.xml file when plug-in related configurations occur in the environment. This has to be disabled because the plugin-cfg.xml that is generated on the Web servers will direct requests directly to the application server environment rather than to the ODRs.

To disable automatic generation and propagation of the Web server plug-in, go to **Servers → Web servers → Web_server_name → Plug-in properties**. Then follow these steps:

1. Deselect the **Automatically generate the plug-in configuration file** check box, as you can see in Figure 7-6 on page 124.
2. Deselect **Automatically propagate plug-in configuration file.**
3. Click **OK**, and save the configuration.

![Web server plug-in configuration](image)

### Figure 7-6   Web server plug-in configuration

#### 7.4.2 Create a script to propagate the plug-in

Plug-in configuration files are automatically generated by the ODR when a change occurs. However, the plug-in file is not automatically available to the Web servers. You can create a script that copies the configuration file to the Web server, and define the name of the script in the ODR proxy plug-in configuration. The script contains the appropriate commands for your environment (for example, scp or FTP) to copy the file to the Web server system. The ODR runs this script every time it generates a new plugin-cfg.xml.

The script must reside on the ODR system, for example:

```
/opt/IBM/WebSphere/AppServer/profiles/odr1/bin/copy_plugin.sh
```

Example 7-1 on page 125 shows a sample copy script using scp on a Linux system. The script copies the configuration file from the ODR system to two Web server systems.
Example 7-1  Plug-in config change script

```bash
#!/bin/sh
/usr/bin/scp /opt/IBM/WebSphere/AppServer/profiles/odr1/etc/plugin-cfg.xml
web2:/usr/IBM/WebSphere/Plugins/config/web2
/usr/bin/scp /opt/IBM/WebSphere/AppServer/profiles/odr1/etc/plugin-cfg.xml
web1:/usr/IBM/WebSphere/Plugins/config/web1
exit 0
```

The server names in this example (odr1, web1 and web2) have to be replaced by your actual ODR and Web server names.

**Tip:** If you are having problems getting the script to execute, you can create a trace of the command. Enable the diagnostic trace on the ODR with the following log detail level:

```
com.ibm.ws.odc.*=all
```

The trace should contain an entry when the script is invoked. The entry will have the following form:

```
notificationCmd: command
```

where `command` is what it actually attempts to execute.

A simple test to verify that the script is working is to define a new virtual host (such as a host alias for default host) in the administrative console. This causes the plug-in to be generated and the script to be invoked. You can do this by navigating to Environment → Virtual Hosts.

Using scp to copy the plug-in file

Example 7-1 uses the `scp` command to copy the plugin-cfg.xml file. If you do not have an identity file under `~/.ssh/identity`, you need to specify where the ssh key is located with the `-i filename` parameter.

The identity has to be in the .ssh directory of the home directory of the WebSphere Extended Deployment user for example, `/root/.ssh`.

To generate the server ssh key use the `ssh-keygen` command:

```
ssh-keygen -t rsa -f "~/.ssh/identity" -C "ODR servername"
```

Leave the passphrase empty; otherwise, the copy plug-in script will not work. Be aware that this is a security risk because if someone is root on the ODR server,
he can get root rights on the Web servers as well. A possible work-around is to copy the files as a different, non-root user, using a different identity file.

Your public key is located in the following file:

`~/.ssh/identity.pub`

After creating the ssh-key you must insert your public key in the `authorized_keys` file on the Web servers:

`~/.ssh/authorized_keys`

For further information about ssh, see the following Web site:

- OpenSSH
  
  [http://www.openssh.com](http://www.openssh.com)

### 7.4.3 Configure the ODR proxy plug-in configuration policy

First, you must decide which ODR generates a plugin-cfg.xml file. This depends on how many unique plugin-cfg.xml files must be generated. Each ODR can generate a plugin-cfg.xml file and it depends on your environment which scope you need.

The Proxy Plug-in Configuration Policy is used by the ODR to determine how to generate a plugin-cfg.xml file.

Use the following steps to configure this setting:

1. The plug-in configuration file is regenerated by the ODR, but you have to propagate this file to the Web server system. The first step in setting up the proxy plug-in is to create a script on the ODR system that can be used to copy the plug-in configuration file from the ODR system to the Web server. See section 7.4.2, “Create a script to propagate the plug-in” on page 124.

   You will specify the name of this file during the proxy plug-in configuration.

2. Go to **Servers → On Demand Routers → ODR_name → On Demand Router settings**.

3. On the Configuration panel scroll down to the Proxy Plugin Configuration Policy pane, which is illustrated in Figure 7-7 on page 127.
4. Select one of the following scopes:
   - **All**: The generated plugin-cfg.xml file includes all ODRs in the environment.
   - **Cell**: The plug-in file includes all ODRs in the same cell.
   - **Node**: The plug-in file includes all ODRs on the same node.
   - **Server**: The plug-in file includes only a single ODR (itself).
   - **None**: Disables generation of the plugin-cfg.xml file.

   If all Web server plug-ins route to all ODRs, choose any single ODR to generate a plugin-cfg.xml file, and select **All** as the scope.

   If you want to tie particular Web servers to particular ODRs, select the appropriate scope, such as **Cell** or **Node**.

   If you want each ODR to be fronted by a single Web server, select **Server** and configure each ODR to generate a plugin-cfg.xml file.

5. Enter the name of the script that will copy the configuration file to the Web server.

6. To create the first plug-in configuration file, restart the ODR. The configuration file is automatically regenerated whenever an applicable change occurs in the system.

### 7.5 Caching in the ODR

The ODR is derived from the Network Deployment proxy server and inherits its caching capabilities. The ODR caches according to the HTTP 1.1 specification (RFC 2616) heuristics (static caching) and also according to WebSphere Application Server dynamic caching heuristics (dynamic caching).

- Static caching refers to the proxy's ability to cache content based on the cache heuristics defined in RFC 2616. An example of such heuristics are the Cache-Control header and the Expires header.

- Dynamic caching refers to the proxy's ability to cache content based on the WebSphere Application Server dynamic caching heuristics. An example of
these heuristics is setting the time to live of a servlet in the cachespec.xml of
an application.

Static caching is enabled by default while dynamic caching is disabled by default. Dynamic caching also requires additional set up on the application server.

For more information about caching see the following:

- *Caching content in the proxy server* at:
  
c=/com.ibm.websphere.nd.doc/info/ae/ae/tjpx_cachestatdyn.html

## 7.6 Configuration of custom error pages

The ODR can send out custom error pages instead of normal HTTP error pages. This allows you to present a more friendly and personalized error message, than the default message, to your Web site users should an error occur.

These error pages are part of an application that runs on the ODRs. Because this is a full J2EE™ application, the opportunity of still helping your users even though there is an error is maximized.

The custom error pages are extremely helpful for lazy application start. When the application is not yet started, the user gets an HTTP error 503 - Server unavailable message. As an example, you could customize this error page so that it automatically retries the server every 30 seconds and explains the reason why the server is not running to the user at the same time.

As a starting point, you can use the default error-handling application that comes with WebSphere Extended Deployment called HTTPErrorHandler.ear, which is located in the *app_server_root/installableApps* directory. You can create your own error handling application based on this one.

**Note:** The interim fix for PK49152 or a superseding fix is required in order for the ODR to serve custom error pages.

To use this application, follow the succeeding steps:

1. Install the application onto all ODRs in your environment. You cannot map an ODR to an application through the administrative console, so you need to use the AdminApp script command to perform the install.

```plaintext
Example 7-2 Installing the HTTPErrorHandler.ear

C:\WebSphere\AppServer\profiles\XDDmgr1\bin>wsadmin
```
WASX7209I: Connected to process "dmgr" on node XDdmgr1node using SOAP connector; The type of process is: DeploymentManger
WASX7029I: For help, enter: "$Help help"
wsadmin>$AdminApp install
"C:/WebSphere/AppServer/installableApps/HttpErrorHandler.ear" {-server xodotr2 -node XDodr2}
ADMA5016I: Installation of HttpErrorHandler started.
ADMA5058I: Application and module versions are validated with versions of deployment targets.
ADMA5005I: The application HttpErrorHandler is configured in the WebSphere Application Server repository.
ADMA5053I: The library references for the installed optional package are created.
ADMA5005I: The application HttpErrorHandler is configured in the WebSphere Application Server repository.
ADMA5001I: The application binaries are saved in C:\WebSphere\AppServer\profiles\XDdmg1\wstemp\Script1132ff08577\workscells\XDcell1\applications\HttpErrorHandler.ear\HttpErrorHandler.ear
ADMA5005I: The application HttpErrorHandler is configured in the WebSphere Application Server repository.
SECJ0400I: Successfully updated the application HttpErrorHandler with the appContextIDForSecurity information.
ADMA5011I: The cleanup of the temp directory for application HttpErrorHandler is complete.
ADMA5013I: Application HttpErrorHandler installed successfully.
wvadmin>$AdminConfig save

The application will not be visible from the administrative console until you log out and log back in. You can also use the following wsadmin $AdminApp command to view the application:

Example 7-3 List applications

wsadmin>$AdminApp list
DefaultApplication
HttpErrorHandler
PlantsByWebSphere
SamplesGallery
WSIFSamples-edition1.0.0
2. Start the application from the administrative console or using the following wsadmin commands.

Example 7-4  Start the HttpErrorHandler application

```
wsadmin> set appManager [$AdminControl queryNames cell=XDcell1,node=Xdodr2,type=ApplicationManager,process=xdodr2,*]
WebSphere:name=ApplicationManager,process=xdodr2,platform=dynamicproxy,
node=Xdodr2,version=6.1.0.7,type=ApplicationManager,mbeanIdentifier=ApplicationManager,cell=XDcell1,spec=1.0
wsadmin> $AdminControl invoke $appManager startApplication
HttpErrorHandler
```

3. Test the application. Following is an example of the syntax for testing the application:

```
http://xdodr2/ErrorPageApp/ErrorPage
```

You should see a white page with a blue bar at the top. Note that you do not need to start the application. In fact, you cannot start the application at all.

4. The next task is to configure the ODRs. To do so, go into the administrative console, and click Servers → On Demand Routers → ODR_name → On Demand Router Properties → On Demand Router settings.

5. Scroll down to the Custom Error Page Policy section, as seen in Figure 7-8 on page 131, and perform the following tasks:

a. Type the following into the Error page generation application URI field:

```
/ErrorPageApp/ErrorPage
```

b. Optionally check the Handle remote errors box.

If you enable the handling of remote errors, all error messages from the remote application server are customized. If you do not want this, uncheck it.

c. In the HTTP status codes, which are to be recognized as errors, field enter any specific HTTP response codes that should be handled by your error page application. You can use X as a wildcard character to denote code ranges. For example, type 4XX to denote all status codes between 400 and 499. Make sure that you have each error code (range) on a separate line. Following is an example:

```
4XX
5XX
```

**Important:** There must be one error code per line. Unlike the statement in the Information Center, error codes must *not* be comma separated.
6. Click **OK**, save your changes, and synchronize with all nodes. The ODRs do not need to be restarted for these changes to become active.

![Custom Error Page Policy](image)

**Figure 7-8  Custom Error Page Policy**

This information is stored in the following:

`app_server_root/profiles/ODR/config/cells/cell_name/nodes/ODR_name/servers/ODR_name/proxy-settings.xml`:

```xml
<errorPagePolicy xmi:id="CustomErrorPagePolicy_1129309186146" errorPageAppURI="/ErrorPageApp/ErrorPage" handleRemoteErrors="true">
  <statusCodes>4XX</statusCodes>
  <statusCodes>5XX</statusCodes>
</errorPagePolicy>
```

### 7.7 Logging and tracing ODRs

You can activate logging and tracing for the ODR with the following log detail level:

- `com.ibm.ws.proxy.*=all`
  
  Gives you information about the routing, communication errors, and response codes.

- `com.ibm.ws.odc.*=all`
  
  Traces all configuration updates to each node. The on demand configuration service (ODC) is the communication protocol that the ODR uses to communicate with the nodes. Information on clusters (static and dynamic), service policies, work classes, transaction classes, and dynamic workload management is provided to the ODR via the ODC.
The following articles in the WebSphere Application Server V6.1 Information Center can help you enable a trace:

- **Enabling trace on a running server**
  

- **Enabling trace at server startup**
  

### 7.8 Routing in a multi-cell environment

It is possible to combine multiple WebSphere Extended Deployment cells and serve them with a common set of ODRs. The ODRs get configuration information automatically from all cells. Figure 7-9 on page 133 illustrates a multi-cell environment:

**Tip:** The log and trace files grow very fast. Make sure, that you set the log size to a big enough value.
In order to route to multiple cells, the ODR’s cell must be configured to allow cross cell communication. The `crossCellCGBCfg` command is provided for you to perform the configuration. See the following Information Center article:

- *Configuring Extended Deployment for cross cell communication for information*
  

Following are some things to consider regarding multi-cell topologies:

- Setting up the cross cell communication automatically provides the ODR with information required to route requests to other cells.
- Application context roots must be unique across cells.
7.8.1 Configuring multi-cell routing cells

Configuring cross-cell routing involves the following steps:

- Configure the core group bridge
- Configure the multi-cluster routing protocol (MCRP)

**Configure the core group bridge**

To create a core group bridge, you need the following files:

- `app_server_root/profiles/dmgr_profile/config/cells/cell_name/XD-CGB-EXPORT`
- `app_server_root/profiles/dmgr_profile/config/cells/cell_name/overlaynodes.config`

**Note:** Each cell has its own set of the files needed to create the core group bridge. To successfully create a core group bridge, you need the files of the other (foreign) cell. This is because these files contain cell specific details and port information, which are used to create a core group bridge. It is also recommended that core group bridge be configured using command line and not using the administrative console.

Run the `crossCellCGBCfg` command in each cell:

1. From a command prompt, type the following command against the deployment manager process for each back-end cell to which the ODR cell routes:

   ```
   crossCellCGBCfg create dmgr_host dmgr_SOAP_port
   path_to_external_cell_XD-CGB-EXPORT
   path_to_external_cell_overlaynodes.config_file
   ```

   For example, to create a core group bridge between BachCell02 and BachCell03 you would do the following:

   1. On the deployment manager system for BachCell03, the following command was executed:

      ```
      crossCellCGBCfg create bach02 8879
      C:\CGBCell\BachCell102\XD-CGB-EXPORTS
      C:\CGBCell\BachCell102\overlaynodes.config
      ```

   2. On the deployment manager system for BachCell02, the following command was executed:

      ```
      crossCellCGBCfg create bach03 8879
      C:\CGBCell\BachCell1102\XD-CGB-EXPORTS
      C:\CGBCell\BachCell1102\overlaynodes.config
      ```
The time required for the crossCellCGBCFG command to complete will vary depending on the size of the cells and network speed.

3. After the command successfully completes on all the cells, restart each entire cell. After the cell is restarted, you can view the core group bridge settings from the deployment manager.

Select **Core groups → Core group bridge settings**.

Figure 7-10 shows the core group bridge settings from the administrative console for BachCell02.

![Figure 7-10  Core group bridge settings](image)

Figure 7-11 on page 136 shows the core group bridge settings from the administrative console for BachCell03.
Configure the multi-cluster routing protocol (MCRP)

The core group bridge enables the exchange of information such as server availability and the health of the systems.

The MCRP enables the routing of requests from the ODR cell to the cell hosting the application. MCRP is configured as a custom property of the ODR properties.

Use the following steps to configure MCRP:

1. Select **Servers** → **On Demand Routers**.
2. Open the configuration page for the ODR by clicking the ODR name. In our example, it is ODR1.
3. In the On Demand Router Settings section, Select **On Demand Router Properties** → **On Demand Router settings**.
4. Select **Custom Properties**.
5. Click **New**.

6. Create custom properties to define the multi-cluster routing policies.

   The Name value has the following format:
   
   `MCRP@cell_name[$application_name[$web_module_name[$cluster_name]]]`

   The value has the following format:
   
   `policy_type@cell_name1$cluster_name1[,cell_name2$cluster_name2,...]`

   The `cell_name` and `cluster_name` values in the Name or Value field can be a wildcard (*). If you use the wildcard in place of a cell name, all the cells in the cell group are indicated. A cell group is defined by any cells that are bridged together with the core group bridge. If you use the wildcard in place of the `cluster_name` value, all of the clusters in a given cell are indicated. Using a wildcard value is only relevant when you are using multi-cluster load balancing routing.

   `policy_type` can be:

   - **failover**: When a request for the application Web module in the cell that is specified in the Name field fails, the request fails over to the cell and cluster that are specified in the Value field after the `@` symbol. Requests only route to the configured cell and cluster when the primary cell is down. The cell status is signified by an HTTP status code of 503, service unavailable. New weight values are obtained every 15 seconds from the Dynamic Work Load Manager (DWLM), which takes into account the application level response time. Use the `mcrp.ui` system property to set the new update time in seconds.

   - **wlor**: Specifies a weighted least outstanding request load balancing policy. This policy comes into effect when the ODR is active and reads its custom property configuration.

   - **wrr**: Specifies a weighted round-robin load balancing policy. This policy comes into effect when the ODR is active and reads its custom property configuration.
7. Click **OK**.
7.9 Autonomic request flow manager (ARFM)

The ARFM continuously monitors the performance of each service class and adjusts the queue dispatching weights in the ODR. The ARFM modifies the weights of queues to align flow control with business goals. When adjusting the dispatching weight, the ARFM includes the following:

1. The amount of computing resources that each class of request consumes.
2. The size and placement of each dynamic cluster.
3. The computing power of each node.
4. The performance goals and business importance of each service class.

The functions of the ARFM include the following:

- Classifying incoming requests. The work is then queued based on the classification (service policies).
- Limiting the number of requests on a given server to protect servers from being overloaded.
- Dispatching work out of the queues based on work classes.
- Dynamically adjusting weights to achieve response time goals for different classes of requests and to respond to varying load conditions and request surges.

7.9.1 Configuring the ARFM

In an environment where response time is prone to fluctuations, you may want to modify the ARFM parameters. For example, increasing the smoothing window parameter, which ultimately increases the sample size that the ARFM uses prior to rebalancing queue dispatching weights.

To view or adjust the settings of the ARFM, navigate to **Operational policies** → **Autonomic Managers** → **Autonomic Request Flow Manager** (Figure 7-14 on
page 140. For detailed information about these settings see the following document:

- **Configuring the autonomic request flow manager**
  

![Figure 7-14 ARFM properties](image)

### 7.10 Application placement controller (APC)

The application placement controller (APC) decides where applications run and how many server instances are started for each application.
The application placement controller receives information from the ARFM and together with the performance data and the configured service policies and service goals it computes the optimal allocation of available resources to running applications. The APC determines the available capacity of each node. It is aware of all processes running on each node including non-WebSphere processes. It takes CPU and memory usage into account and finds the application that fits this node best.

The APC increases or decreases the number of started application servers. It does not deploy applications on demand.

**Note:** The controller does not automatically decrease the number of started application servers when workload decreases, but it will stop one application server on a node and start another application server on the same node where the server is associated with a different dynamic cluster and resources are needed to meet defined service goals.

When a new application is deployed into a dynamic cluster, this application is directly installed on every node in the dynamic cluster. The APC decides when to start which application depending on the service policies and the current load.

Information about newly started or stopped application servers is also given back to the ODR. The ODR reconfigures the routing rules according to these changes.

There is only one APC per cell. You can determine where the APC is running from the administrative console. Select **Runtime Operations → Extended Deployment → Core Components**.

### 7.10.1 Configuring the APC

You can change the configuration in one of the two following ways:

- **Persistent configuration using the Configuration tab**
  
  Changes to the persistent configuration will not take immediate effect—they require the APC to be disabled and enabled again.

- **Runtime configuration using the Runtime tab**
  
  Changes that you perform on the runtime configuration tab take immediate effect but are not made permanent except when checking the **Save To Repository** box.

The APC is configured in the administrative console under **Operational policies → Autonomic Managers → Application Placement Controller**.
You can change the following configuration settings (see Figure 7-15 on page 143):

- **Enable** or deselect to disable the APC.
  
  If you disable the APC, you are effectively disabling all autonomic operations for the dynamic clusters. After application placement is disabled, no dynamic changes occur concerning the size and placement of applications on the dynamic cluster. This action is equivalent to turning all dynamic clusters into static clusters.

- **Approval Timeout** for the supervised mode of the dynamic cluster. Possible values are 1 to 60 minutes.

- **Server Operation Timeout** is the amount of time after a start or stop operation is considered a failure. This should be set to whatever is the predicted worst case time to start or stop a server. Possible values are one to 60 minutes.

- **Minimum Time between Placement Change** is the time the APC should wait before initiating a change. Acceptable values range from 1 minute to 24 hours.

**Tip:** The time between placement changes should depend on how long an application server start takes. Best practice for this value is to define it to be at least 20 to 30 times larger than the time necessary to start the server.
Figure 7-15   Application Placement Controller configuration
For more information about APC configuration see the Information Center article:

- *Monitoring and tuning the application placement controller*
  
  websphere.ops.doc/info/odoe_task/todmonitorapc.html

### Configuring the APC runtime with scripting

Runtime changes to the APC can be made using the PlacementControllerProcs.jacl JACL script.

For details on changing APC runtime configuration settings using scripts, see the following Information Center article:

- *Change the application placement controller runtime configuration with scripting*
  
  websphere.ops.doc/info/reference/todappscripts.html

This wsadmin script is located in `app_server_root/bin`. Following are some examples on how to use the script:

- Usage of the script, its commands and parameters can be obtained using:

  ```
  ./wsadmin.sh -profile PlacementControllerProcs.jacl -c "help"
  ```

- You can change the Approval Timeout using the following command:

  ```
  ./wsadmin.sh -profile PlacementControllerProcs.jacl -c "setApprovalTimeOut 12"
  ```

- To retrieve the current node where the APC is running, enter the following:

  ```
  ./wsadmin.sh -profile PlacementControllerProcs.jacl -c "getNodeName"
  ```

### Configuring the APC with scripting (persistent)

To make persistent changes to the APC, you need to use the xd_APCconfig.jacl script. The changes you make here are not immediately effective. They become active when the APC is disabled and enabled again.

This wsadmin script is located in `app_server_root/bin`. Following are some examples on how to use the script:

For details on changing APC persistent configuration settings using scripts, see the following Information Center article:

- *Changing application placement configurations with scripting (persistent changes)*
  
  websphere.ops.doc/info/reference/rodappscript2.html
For the script usage, type the following:

```
./wsadmin.sh -profile xd_APCconfig.jacl -c "help"
```

Example 7-5 shows the command to change the approval timeout.

---

### Example 7-5 Using the xd_APCconfig.jacl script

```
/usr/IBM/WebSphere/AppServer/bin # ./wsadmin.sh -profile xd_APCconfig.jacl
WASX7209I: Connected to process "dmgr" on node dmgr1 using SOAP connector; The
  type of process is: DeploymentManager
WASX7029I: For help, enter: "$Help help"

wsadmin>$AdminConfig show [\$AdminConfig getid "/AppPlacementController:/"]
{approvalTimeOut 10}
{enable true}
{minTimeBetweenPlacementChange 15}
{minTimeBetweenPlacementChangeUnits 2}
{properties {}}
{serverOperationTimeOut 5}

wsadmin>setAPCAttribute approvalTimeOut 15

wsadmin>$AdminConfig show [\$AdminConfig getid "/AppPlacementController:/"]
{approvalTimeOut 15}
{enable true}
{minTimeBetweenPlacementChange 15}
{minTimeBetweenPlacementChangeUnits 2}
{properties {}}
{serverOperationTimeOut 5}
```

---

### 7.10.2 Logging and tracing the APC

You can activate logging and tracing for the APC with the following setting:

```
com.ibm.ws.xd.placement*=all=enabled
```

You can either change the trace level through the administrative console or using a wsadmin script.

The following articles in the WebSphere Application Server V6.1 Information Center can help you enable a trace:

- **Enabling trace on a running server**
  
7.11 Dynamic Workload Manager

The ARFM classifies and prioritizes requests to application servers based on the demand and policies. The dynamic workload manager (DWLM) then distributes the requests among the nodes to balance the work.

It sets the load balancing weights for application servers dynamically to stay current with the business goals. The weights are then used by the router in the ODR to distribute the workload accordingly. This autonomic manager continuously monitors the response time and resource utilization of each server and uses feedback control techniques that change the dispatching weights to achieve balance across clusters and nodes.

DWLM can also dynamically update the application status as the application placement controller may make modifications to a running application infrastructure.

DWLM is enabled by default. It can be enabled or disabled through the administrative console under Servers → Dynamic Clusters → cluster_name → Dynamic workload management (WLM).
Work with dynamic clusters

Dynamic clusters are at the center of the operations optimization features of WebSphere Extended Deployment. A dynamic cluster is an application deployment target that is expandable as needed by the dynamic operations environment. This chapter discusses how to create and manage dynamic clusters.

Following are the topics we discuss in this chapter:

- Dynamic clusters versus static clusters
- Creating dynamic clusters
- Dynamic clusters with WebSphere application servers
- Dynamic clusters with PHP servers
- Dynamic clusters with other middleware servers
- Setting the operating mode of a dynamic cluster
- Changing a dynamic cluster
- Deleting a dynamic cluster
- Migrating a static cluster to a dynamic cluster
- Setting a node or server into maintenance mode
8.1 Dynamic clusters versus static clusters

A dynamic cluster is similar to a Network Deployment static cluster in terms of application deployment and application configuration but differs in terms of use and manageability. Table 8-1 compares static and dynamic clusters.

<table>
<thead>
<tr>
<th>Static cluster</th>
<th>Dynamic cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
<td>A static cluster is a group of application servers in a Network Deployment environment that participate in workload management.</td>
</tr>
<tr>
<td><strong>Cluster management</strong></td>
<td>You define the application servers that are in the cluster, and then start the application servers in the cluster.</td>
</tr>
<tr>
<td><strong>Cluster templates</strong></td>
<td>When you define a static cluster, you can select an application server template that all the application server instances you create are based on. There is no possibility to change all application servers in a cluster at once as it is not possible to change an existing server template. Therefore you need to change each application server in the cluster individually.</td>
</tr>
<tr>
<td><strong>Application server weights</strong></td>
<td>You explicitly assign a weight value to each application server instance.</td>
</tr>
<tr>
<td><strong>Applicability</strong></td>
<td>If you use static clusters in a WebSphere Extended Deployment cell, the behavior is identical to static clusters in Network Deployment.</td>
</tr>
</tbody>
</table>
8.2 Creating dynamic clusters

Dynamic clusters can be created from the administrative console by navigating to **Servers → Dynamic Clusters**. All servers in the cluster must be of the same type.

The process you follow in the administrative console for creating a dynamic cluster will vary slightly depending on the type of servers you select for the cluster.

Dynamic clusters consisting of assisted lifecycle servers consist of servers you manually add to the cluster. Dynamic clusters consisting of full lifecycle servers (WebSphere application servers and PHP) use membership rules that define nodes to host the cluster servers. The application servers are automatically generated on those nodes based on the options you define for the cluster.

Table 8-2 shows an overview of the process you go through to create a dynamic cluster for each server type when using the administrative console.

*Table 8-2   Dynamic cluster options*

<table>
<thead>
<tr>
<th>Wizard steps</th>
<th>WebSphere Application Server</th>
<th>PHP</th>
<th>Assisted lifecycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Add members ▶ automatically based on rules ▶ manually by selecting a static cluster to convert</td>
<td>Add members ▶ automatically based on rules</td>
<td>Add members ▶ manually by selecting predefined server definitions or by defining existing servers</td>
</tr>
<tr>
<td>Step 2</td>
<td>Select a template.</td>
<td>Select a template.</td>
<td>Select a template.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Tailor the dynamic cluster properties.</td>
<td>Tailor the dynamic cluster properties.</td>
<td>Tailor the dynamic cluster properties.</td>
</tr>
</tbody>
</table>

You can also use scripting to create and delete dynamic clusters with the wsadmin command line interface. Command line options include listing and setting dynamic cluster parameters. You can use the wsadmin command in interactive mode to try out some command options prior to creating a script.

When you start the wsadmin command specify the -lang option to specify whether you are using jython or jacl as your scripting language. The following example creates a Tomcat dynamic cluster using predefined Tomcat servers.

```java
AdminTask.createDynamicClusterFromForeignServers('MyTomcat_DC', ['-foreignServers [[TomcatNode TomcatServer] [TomcatNode2 TomcatServer2]]']
```

Chapter 8. Working with dynamic clusters  149
A complete table of dynamic cluster scripting commands can be found in the WebSphere Extended Deployment V6.1 Information Center at the following Web site:

- Dynamic cluster administrative tasks
  

### 8.2.1 Dynamic cluster membership policy

Before starting the process of defining a new dynamic cluster for WebSphere and PHP servers, give some thought as to how you want to define the membership policy for the cluster.

The membership policy is compared against the nodes in your cell. WebSphere Extended Deployment creates servers for the dynamic cluster using nodes that match the membership policy that you define. When new nodes are added to your environment, they are automatically added to the dynamic cluster if they match the defined membership policy.

Prior to WebSphere Extended Deployment V6.1 members of a dynamic cluster were scoped explicitly by node group. In WebSphere Extended Deployment V6.1 you have more options available to delimit your dynamic cluster. A subexpression builder is included within the process that helps you create a membership policy. A default membership policy is predefined for you.

You can use the following criteria to define your rule:

- Node group
- Node name
- Node host name
- Node property values

The following are examples of membership rules:

- The following rule is the default. This includes all WebSphere nodes in the DefaultNodeGroup with Extended Deployment. By default, servers are placed in the DefaultNodeGroup.
  
  ```
  node_nodegroup = 'DefaultNodeGroup' AND
  node_property$com.ibm.websphere.wxdopProductShortName = 'WXDOP'
  ```

- The following rule includes all nodes that start with the characters XDn.
  
  ```
  node_name LIKE 'XDn*' 
  ```

  In the sample topology, this membership would include XDnode1, XDnode2, and XDnode3.
**Node properties**

Node property values include the product type and versions installed on the node. To view these properties, click **System administration → Middleware nodes → node_name → Node installation properties.**

Figure 8-1 shows the node properties associated with a WebSphere application server.

![Node properties for WebSphere application servers](image)

Figure 8-2 on page 152 shows the node properties associated with a PHP server.
8.2.2 Dynamic cluster properties

Each dynamic cluster has a set of properties that you set when you create the dynamic cluster. These can be modified later by navigating to Servers → Dynamic clusters → *cluster_name*.

The properties available vary depending on the server type in the cluster.

<table>
<thead>
<tr>
<th>Property type</th>
<th>WebSphere Application Server</th>
<th>PHP</th>
<th>Assisted lifecycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum number of cluster instances</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Maximum number of cluster instances</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 8-3  Dynamic cluster properties

![Node properties for PHP servers](image)

Figure 8-2  Node properties for PHP servers
Table 8-1 on page 148 shows an example of the property definition page you will see when you create a dynamic cluster using the administrative console. This particular example is from a dynamic cluster using WebSphere application servers.

![Figure 8-3 Dynamic cluster properties](image-url)
The following list explains the fields and selections on Figure 8-3 on page 153.

- **Minimum number of cluster instances.** You can select to have one or more application servers started at all times or to stop all application servers in times of inactivity. The latter is called lazy application start. For information about this topic, please read section 8.2.3, “Lazy application start” on page 154.

- **Maximum number of cluster instances.** For this example take the default value.

- **Vertical stacking of instances on node.** Check the Allow more than one instance to start on the same node box if you want to allow more than one application server instance to be started on the same node. See section 8.2.4, “Vertical stacking” on page 155 for more information.

- **Isolation Preference** is a new property in WebSphere Extended Deployment V6.1 with the following settings:
  - **No Isolation Requirements** allows cluster members of one dynamic cluster to run on the same node as cluster members of a different dynamic cluster.
  - **Strict Isolation** mandates that only cluster members of the same dynamic cluster may run on the same node.
  - The **Associate with an Isolation group** option specifies that cluster members from this dynamic cluster can run only with members of other dynamic cluster members that belong to the same isolation group. To create a new isolation group, type a unique name in the **Isolation Group Name** field.

Note that for assisted lifecycle servers, WebSphere Extended Deployment can only enforce the isolation properties within the context of WebSphere Extended Deployment. An administrator could still manually create a server running on the same node as servers that belong to another dynamic cluster, and they could start that server using the server's native startup commands.

### 8.2.3 Lazy application start

The lazy application start feature optimizes server resource allocation during times of inactivity. The smallest size for a dynamic cluster is zero, implying that an application can be configured for execution but not running in any application server instances. When a request for that application is received by the ODR, an application server for that application is automatically started on any node in the dynamic cluster. Requests for the application are not queued and clients receive an “HTTP Error 503 - server unavailable” error until the server is up and the application is available. There is a special Web page for this error, which can be individually set up with an automatic redirect after a few seconds. See section
7.6, “Configuration of custom error pages” on page 128 for details on how to do this.

A typical environment where the lazy application start feature is beneficial is an environment on which the ratio of the number of dynamic clusters to the number of servers is high, and where many dynamic clusters are not accessed for long periods of time. In such an environment, it is beneficial to hibernate idle dynamic clusters temporarily (stopping all server instances), thereby releasing valuable resources to be used by active dynamic clusters.

**Configuring lazy application start**

Lazy application start can only be configured for the whole dynamic cluster.

Use the following steps to enable lazy application start:

1. Go to **Servers → Dynamic Clusters** in the administrative console.
2. Click your dynamic cluster name.
3. On the upcoming panel check the **Stop all instances during periods of inactivity** radio button.

The second setting for lazy application start is **Time to wait before stopping instances**. By default it is set to 60 minutes. But this value really depends on your application and user behavior. You can look at the server activity logs if you cannot ask your users directly for information. In these logs, you can discern if and how long servers are idle during the day and when an application has its peak time.

It is also possible to configure lazy application start during creation of the dynamic cluster. See 8.3, “Dynamic clusters with WebSphere application servers” on page 158 for more information.

**8.2.4 Vertical stacking**

*Vertical stacking* allows you to have more than one application server instance in a dynamic cluster on the same node. The benefit of this capability is better hardware utilization if a system’s CPU and memory is not fully used with a single application server on a node.

Vertical stacking should only be used when a single application server instance cannot consume the full CPU resources of a node.

To determine how many application server instances can run in parallel on a node, take a look at the resources needed when only one instance is active. This can be achieved by use of operating system-specific system monitoring tools.
We recommend that you determine the stacking number for a dynamic cluster with no other application servers running on that node. First, start one instance. While monitoring the effective throughput, increase the workload intensity. When throughput saturates, start one more instance. When adding a new instance does not improve the throughput, the stacking number is one. If adding a new instance improves the throughput, you can conclude that the application has some internal bottleneck that prevents it from effectively using the entire box within a single application server. Thus you can continue to increase workload and (possibly) the number of active instances until no improvement in throughput can be achieved.

Repeat this approach individually for every application that can possibly run on that node. This way, you can decide for each cluster (and thus application) whether it should be using stacking or not.

When determining the stacking number for a dynamic cluster, you do not have to consider any other dynamic clusters because you only want to learn how many instances of this dynamic cluster are needed to fully utilize the system. So when you have multiple clusters/applications, each one of them can fully utilize the system if no other application is running concurrently (for example, because these applications only run at certain times of the day, week, or month). The stacking number becomes the maximum number of instances that are allowed to execute on a node for this dynamic cluster, but at any time, a smaller number may be started, depending on the current workload. At run time the APC, ODR, and DWLM will work together to make sure that the node is not overloaded and that all applications meet their policies.

**Configuring vertical stacking**

You configure vertical stacking for the whole dynamic cluster. Use the following steps to enable this feature:

1. Select **Servers → Dynamic Clusters** in the administrative console.
2. Click the desired cluster.
3. Select **Allow more than one instance to start on the same node**.
4. Under **Number of instances**, enter the number of instances you want to be started in parallel on each node.
It is also possible to configure vertical stacking during creation of the dynamic cluster. See section 8.3, “Dynamic clusters with WebSphere application servers” on page 158 for more information.

8.2.5 Using a server template

Servers in a dynamic cluster are based on a common template. A change to the template is propagated to all the members in the cluster.

Tip: The vertical stacking number must be determined for each node. When you have heterogeneous nodes you might have different stacking numbers for each node. You can define different vertical stacking numbers per node within the dynamic cluster by using a custom property named numVerticalInstances.nodename (for example numVerticalInstances.node1).

You can enter this custom property in the administrative console under Servers → Dynamic Clusters. Click your cluster and then click Custom Properties on the right.

It is also possible to configure vertical stacking during creation of the dynamic cluster. See section 8.3, “Dynamic clusters with WebSphere application servers” on page 158 for more information.

Attention: Changes made to the template within the dynamic cluster (under Servers → Dynamic Clusters → cluster_name → Server template) are propagated to all application servers within this cluster.

You cannot change the originating server template itself (under Servers → Application servers → Templates...). You can only delete it, and create a new one based on an application server that contains all configuration settings you need.

WebSphere Extended Deployment provides default templates for each server type. You can use one of these provided templates, or you can create one of your own based on an existing middleware server.

Following is a summary of the process of using a server template:

1. Create the first server.
2. Configure the first server with the desired settings.
3. Create a template based on this server.
4. Use this template as a basis for future servers (including those created in a dynamic cluster) of this type.
5. (Optional) Delete the original server if you created it solely for the purpose of creating a template.
It is important to plan this step since you can create servers or server representations as part of the dynamic cluster creation. If you wait until you create the dynamic clusters, you will not have the opportunity to create and use a unique template. The server and template must be created before you create the dynamic clusters.

To create an application server template based on an existing server:

1. Select **Servers → All servers.**
2. Click the **Templates...** button at the top of the server list.
3. Click **New.**
4. Select a server from the list to build the template from, and click **OK.**
5. Enter a name and description for the template, and click **OK.**
6. Save and synchronize your configuration.

The new template is added to the list of templates and available for selection the next time you create a new server. WebSphere Application Server templates can also be used when creating a new cluster (static or dynamic).

### 8.3 Dynamic clusters with WebSphere application servers

Creating a dynamic cluster is similar to creating a static cluster, but includes some dynamic cluster specific configuration settings:

1. In the administrative console go to **Servers → Dynamic Clusters**, and click **New.**
2. Select **WebSphere application server** as the server type (Figure 8-4).
3. Click **Next**.

4. The next panel allows you to determine how cluster members are added to the dynamic cluster: automatically using rules or manually defined.

---

**Automatically define cluster members with rules**

- Type the dynamic cluster name, for example **XDCluster1**.
- Check the **Prefer local enabled** option if you want to optimize EJB client request routing. This option insures that out-of-process calls (from one application server to another or an application client to an application server) are first attempted on the local node, and requests are only directed to a remote node if the EJB is not running locally.

   **Note:** The best practice is to deploy EJB (Web) clients in the same EAR as the EJB within a single application server. This eliminates the need for out-of-process calls which degrade performance.

- Select whether to create a replication domain for the cluster. Replication domains provide the ability for cluster members to transfer data, objects, or events. One example of this is the replication of HTTP session data. More information regarding replication domains reside in the WebSphere Application Server Information Center and in *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

---

**Manually define cluster member**

*(New in V6.1)* This option allows you to convert a pre-existing static cluster into a dynamic cluster. An existing static cluster must already exist in order
to use the manual option. The name of the cluster remains the same after it is converted to a dynamic cluster.

Because this example is using WebSphere Application Servers in a new installation, the automatic option is selected.

5. Click Next.

6. Because you chose to use rules to add servers to the dynamic cluster, the next page of the wizard allows you to build the expression, which is evaluated to determine members of the dynamic cluster. The rule ultimately defines the nodes to use in the dynamic cluster both now and for nodes that are added to your environment in the future.

The Define dynamic cluster members panel initially contains a default membership policy (Figure 8-6).

![Figure 8-6 Create a dynamic cluster—define cluster members](image)

Click the link for Syntax help to review your options.

The default expression defines membership policy, specifies all nodes that belong to the DefaultNodeGroup, and specifies all nodes that have the node_property $com.ibm.websphere.wxdopProductShortName = 'WXDOP'.

Click the link for Syntax help to review your options.

The default expression defines membership policy, specifies all nodes that belong to the DefaultNodeGroup, and specifies all nodes that have the node_property $com.ibm.websphere.wxdopProductShortName = 'WXDOP'.
The product shortname WXDOP tests that WebSphere Extended Deployment Operations Optimization is installed on the node. You may have a non-WebSphere Extended Deployment node in your WebSphere cell in which case the product shortname property is not set for that specific node, and the node could not be added to the dynamic cluster.

To change the rule, type the new rule in over it, or click **Subexpression builder** to use a GUI-driven interface.

Our example will delimit dynamic cluster members by node name where all nodes in the dynamic cluster must contain the string “XDn” in their node name.

7. Click **Subexpression builder**.

8. In the Subexpression builder pane enter the following values (see Figure 8-7 on page 162).
   - Select operand: Node name
   - Operator: Like
   - Value: %XDn% Note that your values may be different here based on the node names in your environment. Also note that you can use the “%” character as a wildcard.

**Tip:** Typically you would not set node_nodegroup = ‘DefaultNodeGroup’ because this includes your On Demand Router (ODR) nodes as a location for your cluster members. As a best practice if you use node groups as part of your membership policy, then define a new node group prior to creating your dynamic cluster. Another alternative is to not use the node group parameter at all when defining your membership policy.
9. Click the **Generate subexpression** button to create the rule.

10. Click the **Append** button.

    Note that the new expression is appended to the original rule. You can type directly into this window to modify the expression, so if you want to replace the original rule, rather than append, you can manually modify the policy here.

11. Click **Close**, and review the new membership policy. For our example, we deleted the `node_nodeGroup` parameter.
12. Click **Preview membership** to make sure your rule selects the nodes you expect. You should not see ODR nodes.

![Figure 8-9  Membership policy preview](image)

13. Click **Next** to go to the next step.

14. Select the **Server template**. For our purposes, select **DefaultXD**, which is the recommended template; nevertheless, there are three server template options. Following is a description of all of the available templates.

   - **DeveloperServer**
     
     This template is used for development environments. It is not recommended for an Extended Deployment environment.

   - **Default**
     
     This template is identical to the defaultXD template except that the Web container thread pool maximum size value is set to 50 in the default template and is set to 500 in the defaultXD template.

   - **DefaultXD**
     
     This is the recommended template.

   This panel also displays any user-defined templates. After you select **DefaultXD** as your Server template, click **Next**.

15. The final panel allows you to set the dynamic cluster properties. In this example, the defaults were taken (see Figure 8-3 on page 153).

   After entering all necessary information click **Next**, and then click **Finish**.

16. Save your configuration.
8.4 Dynamic clusters with PHP servers

Note: If you are using Apache HTTP Server Version 2.2 and PHP Version 5.2.x, then you must perform some manual edits to the httpd.conf file.

For information, see the following article:
- Creating PHP servers and PHP dynamic clusters

To create a dynamic cluster that consists of PHP servers, do the following:

1. In the administrative console go to Servers → Dynamic Clusters, and click New.
2. Select PHP server as the server type.
   Enter a name for the new dynamic cluster, and click Next.
3. The only option for determining cluster membership for PHP dynamic clusters is using rules. Leave the Automatically define cluster members with rules option selected, and click Next.
4. The next page of the wizard allows you to build the expression which is evaluated to determine members of the dynamic cluster. The rule ultimately defines the nodes to use in the dynamic cluster both now and for nodes that are added to your environment in the future.
   The Define dynamic cluster members panel initially contains a default membership policy (Figure 8-6 on page 160).
   To change the rule, type the new rule in over it, or click Subexpression builder to use a GUI-driven interface.
   Our example will use the default policy. See Figure 8-10 on page 165.
5. Click **Preview membership** to make sure your rule selects the nodes you expect. Close the window.

6. Click **Next** to go to the next step.
7. Select the correct **Server template**. There are templates corresponding to the supported PHP releases. Select the appropriate template, and click **Next** to go to the next step.

8. The final panel allows you to set the dynamic cluster properties.

9. For our example we use the default values (see Figure 8-3 on page 153). After entering all necessary information click **Next**, and then click **Finish**.

10. Save your configuration.

### 8.5 Dynamic clusters with other middleware servers

The process for creating a dynamic cluster with any of the following node types is similar:

- Apache
- Custom HTTP Server
- JBoss
- Apache Tomcat
- WebSphere Application Server Community Edition
- BEA WebLogic

To create a dynamic cluster for these middleware servers:

1. In the administrative console go to **Servers → Dynamic Clusters**, and click **New**.

2. Select the server type, as shown in Figure 8-12 on page 167. In this example, a dynamic cluster with Apache Tomcat servers are created with a cluster name of Tomcat_DC. Click **Next**.
3. The administrative console automatically skips to **Step 3: Define dynamic cluster members** panel. This panel allows you to define which servers will be members of the dynamic cluster. You can either choose from existing server representations or you can create new server representations.

In either case the server representation is only a definition of the middleware server. When you define a Tomcat server to WebSphere Extended Deployment it does not create the actual server implementation. Creation of the actual server must be done manually outside of WebSphere.
As you can see in Figure 8-13 we select both pre-existing Tomcat server definitions. Click each server in the list that is to be a dynamic cluster member, and then click the **Add Member** button. (You can use the shift key to select multiple servers simultaneously.)

Click **Next**.

4. The next panel allows you to set the dynamic cluster properties. This is similar to the panel you would see for WebSphere and PHP servers but without the vertical stacking option. Information regarding these properties can be found earlier in this chapter.
For our example we use the default values. Make any changes based on your cluster requirements. Click **Next**, and then click **Finish**.
5. Save your configuration.

8.6 Setting the operating mode of a dynamic cluster

After creating the dynamic cluster, you can change the operating mode of the cluster. WebSphere Extended Deployment supports different operating modes.

Use the following steps to change the mode:

1. Select **Servers → Dynamic clusters**.
2. Check the box to the left of the dynamic cluster.
3. Select the mode from the pull-down menu, and click **Set mode**. See Figure 8-16 on page 171 for an example, where we also explain the available Mode options on the window.
Chapter 8. Working with dynamic clusters

Figure 8-16   Set operational mode

- **Automatic mode**
  The APC automatically starts and stops application servers as needed when the workload changes.

- **Supervised operating mode**
  Identical to the automatic mode except that all actions have to be approved by an administrator. Tasks needing approval are displayed in the administrative console under **System administration → Task Management → Runtime Tasks**.

  **Note:** All runtime tasks have an approval timeout. When the timeout is reached and the administrator did not accept the task, then the task is automatically declined. As a result, for an application with a high availability or continuous availability requirement, you would need to have an administrator to handle runtime tasks at all times.

Therefore, unless you have around the clock staffing, the supervised mode is only feasible for high availability or continuous availability applications during the introduction phase of WebSphere Extended Deployment. After application rollout and operation validation finishes successfully, the cluster should be set to automatic mode.
For more information about Runtime Tasks and how to accept or decline the tasks, see section 12.5, “Monitoring runtime tasks” on page 305.

- Manual operating mode

Does not support automated application placement or runtime task suggestions. However, dynamic workload management is supported. When using the manual mode you basically combine a dynamic cluster with a static cluster: all automated processes are disabled but you take advantage of dynamic workload management.

### 8.7 Changing a dynamic cluster

After you create a dynamic cluster you should only edit the dynamic cluster settings and not the single application server settings. Individual node settings are overwritten by cluster settings.

**Note:** The best practice is to create a single application server first and tune it for your application. When you find the best settings for your application, create a template from this application server, then create a dynamic cluster from the template. You can then delete the original application server.

### 8.8 Deleting a dynamic cluster

Before deleting a dynamic cluster, you must first uninstall any application deployed to that cluster. Then go to Servers → Dynamic Clusters. Before a dynamic cluster can be deleted, it must be set to manual mode. When it is in manual mode, select the dynamic cluster, and click the Delete button. Save and synchronize your changes.

### 8.9 Migrating a static cluster to a dynamic cluster

It is possible to migrate a static cluster to a dynamic cluster using the following steps:

1. From the Dynamic Clusters panel, click **New** to create a new dynamic cluster.
2. At the Select the Membership method pane, click the **Manually define cluster members** radio button.
3. Select the static cluster to convert and to complete the steps to create the dynamic cluster.
8.10 Setting a node or server into maintenance mode

When managing a WebSphere Extended Deployment environment you may need to make changes to nodes or servers in your environment. Those changes may include troubleshooting, performing maintenance, or tuning the environment. In a dynamic environment the challenge is how to achieve this without disrupting the end user community.

The solution to this challenge is to place the node or server into maintenance mode. When a node or server is in maintenance mode the ODR stops routing requests to that server. The APC also excludes this node or server from all automatic application placement. This allows administrators to perform maintenance or changes to that specific server or node with minimal disruption to the environment.

**Note:** If you bypass the ODR by routing directly to the server or directly from a Web server, then maintenance mode will have no impact.

**Setting a node into maintenance mode**

Use the following steps to set a node into maintenance mode:

1. Open the administrative console, and go to **System administration → Middleware Nodes**.
2. Check the box to the left of the node for which you want to perform maintenance.
3. Select the appropriate mode from the Select mode pull-down, and click **Set Mode**.
   - **Maintenance mode**
     This puts the node into maintenance mode but does not stop processing on that node. Requests on the node are served and open sessions are routed to the node until the session ends or times out. After all requests are completed, the node is moved into maintenance mode. Any new requests are routed to nodes that are not in maintenance mode. After servers are in maintenance mode you can stop them manually.
   - **Maintenance immediate stop mode**
     Stops all processes on the node immediately and puts the node into maintenance mode.

You can recognize that a node is in maintenance mode by the small icon in the Maintenance column of the nodes overview page.
To take the node off of maintenance mode, follow the same process listed above, except when you get to step number three, select **Normal** mode (which is not listed).

**Note:** After resetting the maintenance mode of a node, the node agent or middleware agent is not automatically started.

Node agents cannot be started from the administrative console. Use the `startNode` command in the `profile_name/bin` directory on that system.

Middleware agents can be started from the administrative console or by using the `startAgent` command, which can be found in the `xd_home/bin` directory.

### Setting a server into maintenance mode

Use the following step to set a server into maintenance mode:

1. From the administrative console, click **Servers** → **All Servers**.

2. Follow the same steps as described for setting a node in maintenance mode. Note that servers provide one additional mode:
   - **Maintenance mode - break affinity**
     
     This moves the server into maintenance mode and breaks HTTP or Session Initiation Protocol (SIP) session affinity to the selected server. EJB affinity is not effected.
Managing applications

This chapter discusses how to deploy WebSphere and PHP applications, as well as how to define existing applications on assisted lifecycle servers.

It also discusses using the application edition management features of WebSphere Extended Deployment to make application management and deployment into a production environment easier, more reliable, and interruption-free.

This chapter includes the following topics:

- Installing WebSphere applications
- Installing PHP applications
- Defining other middleware applications
- Introduction to application edition management
- Installing application editions
- Application activation and rollout
- Tracing the application edition manager
9.1 Installing WebSphere applications

Following is a quick overview that can help you distinguish between the process for installing a WebSphere application from the process used for defining applications on other middleware servers. This overview does not address any of the preparations you might need to take for WebSphere applications, such as defining resources required by the application or defining virtual hosts.

Application packaging and deployment for WebSphere applications is covered thoroughly in the IBM Redbooks publication *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

Use the following steps to deploy the application:

1. Select **Applications → Install New Application** from the administrative console navigation bar.

2. Check the **Local file system** or **Remote file system** box, and click the **Browse** button to locate the application EAR file.

   You can install files that are located either on the same machine as your browser (local file system option) or on any node in the WebSphere Application Server cell (the remote file system option).

   If you select the **Prompt me only when additional information is required** option, only the windows where you actually need to fill out some information during installation are shown.

   Select **Prompt me only when additional information is required**. Then click **Next**.

   The rest of the wizard is divided into steps. The number of steps depends on your application, for example, if it contains EJB modules or Web modules, you will see windows prompting for the information necessary to deploy them.

3. Step 1: Select installation options.

   Step 1 gives you a chance to review the installation options. You can specify various deployment options, such as JSP™ precompiling and whether you want to generate EJB deployment code. You can see some of the options on this page in Figure 9-1 on page 177.

   Note that the Application edition, and Edition description fields are specific to Extended Deployment.
We discuss information about application editions later in section 9.4, “Introduction to application edition management” on page 186.

Click Next.

4. Step 2: Map modules to servers.

Select the deployment target from the pull-down list of clusters and servers. Dynamic clusters are unique to WebSphere Extended Deployment.

Click the icon to select all modules, or individually select modules from the list. Select the dynamic cluster in the pull-down, and click Apply.

**Note:** Unlike Network Deployment, you do not have to select a Web server. The proxy plug-in generation on the ODR creates the appropriate configuration required for routing application requests from the Web server.
5. Click **Next**.

6. **Step 3: Summary.**
   
   The Summary window gives an overview of application deployment settings. If those settings are fine, click **Finish** to deploy the application.

7. **Save the configuration.**

   If you are working in a distributed server environment, make sure you synchronize the changes with the nodes so that the application is propagated to the target application server (s).
9.2 Installing PHP applications

Note: Only one active edition of a PHP application is supported on a node. If you have multiple active editions of the same PHP application, ensure that the editions are not deployed to servers that are on the same node.

Use the following steps to deploy an application to PHP:

1. Select Applications → Install New Middleware Application from the administrative console navigation bar.

2. Select PHP as the middleware application type, and click Next.

3. Check the Local file system or Remote file system box, and click the Browse button to locate the application file.

   The file must have one of the following extensions:
   - zip
   - tar
   - gz
   - tgz
   - jar

   You can install files that are located either on the same machine as your browser (local file system option) or on any node in the cell (the remote file system option).
Resources in a PHP application are defined by PHP scripts. You can specify one setup script and one cleanup script. These scripts are optional depending on the application needs.

The setup script is executed during application install. The script can perform actions such as the following:

- Set file permissions
- Create or delete directories
- Verify the version of PHP
- Verify the availability of PHP extensions, XML Parsers etc.,
- Load hooks
- Load database drivers
- Create and populate database tables

The cleanup script is called during application uninstall to clean up files and any other artifacts created by the application that are no longer needed.

Click Next.
4. Type a name for the application. This name is for WebSphere administrative purposes and does not have to match anything on the PHP system. Click Next.

5. As shown in Figure 9-6 on page 182, enter the following values into the appropriate field. These values are specific to WebSphere and are used to route requests to the PHP system.

   - The context root. This value is used to route requests for the application to the PHP server. It does not correspond to any values on the PHP server or application.
   - The virtual host.
   - Select the deployment target, in this case the PHP dynamic cluster, and click **Add** to add it to the list of deployment targets.
6. Click **Next**.

7. Confirm the settings, and click **Finish**. Note that the module name representing the PHP application module is automatically populated (module1). You will see this module later when you define service policies for the application.
9.3 Defining other middleware applications

This section describes middleware applications deployed to platforms other than PHP or WebSphere Application Server.

9.3.1 Defining a middleware application

In this section we define a middleware application to WebSphere Extended Deployment. This allows the ODR to route requests to applications that were deployed to other middleware platforms, such as Apache Tomcat or WebSphere Application Server Community Edition. Defining this application does not cause the application to get deployed to the actual middleware server where it will run.

Define the middleware server environment, which serves as the deployment target, to WebSphere Extended Deployment prior to defining the application.

In the following example the deployment target is a Tomcat dynamic cluster, which was previously defined.

Use the following steps to define the application:

1. Select Applications → Install New Middleware Application.
2. Select Unmanaged Web Applications for the application type, and click Next.

![Select the type of middleware application]

Figure 9-8  Defining a middleware application - select the type of middleware application

3. Type a name for the application. The name is for administrative purposes.
Edition manager capabilities such as roll out and validate are not supported for unmanaged Web applications.

However, if a second edition of an application is deployed on the externally created servers, you can create a new edition of the application in the administrative console by creating a new representation of the application with a different value entered in the Application edition field. You can then specify routing policies to control how to route the requests to the multiple editions.

Click **Next**.

4. The next panel, shown in Figure 9-10 on page 185, defines deployment information.
   a. If you have a large number of deployment targets defined, you can use the Filter box to select the type of deployment targets to display. If you plan to use this filter, select it first. Changing the filter clears all the fields on this panel.
   b. Enter a module name and context root for the application. The context root routes requests for the application to the server. It does not correspond to any values on the middleware server or application. The module name is used when building service policies for the application.
   c. Select the virtual host.
   d. Select the deployment target in the left box, and click **Add >>** to move it to the right.
   e. Click **Apply**. You will see an entry for the deployment target at the bottom of the panel.

Click **Next**.
5. On the **Confirm the new application representation** panel click **Finish**
6. Save your changes. HTTP requests can now be routed through the ODR to the middleware application.

9.4 Introduction to application edition management

WebSphere Extended Deployment contains an application edition manager that provides advanced application management capabilities. These capabilities address five functional requirements:

- Application versioning capability.
- Interruption-free application rollout.
- Ability to back-out an application update.
- A validation mode to test and verify an application before rolling it out to production.
- Ability to host concurrent versions of an application to support needs of more complicated rollout strategies, such as piloting and branch upgrade.

There are different levels of support for the application edition manager in WebSphere Extended Edition V6.1 based on the type of application that is being managed. They are summarized in Figure 9-12 on page 187.
9.5 Installing application editions

Companies commonly have a build and deploy process that their application moves through as it goes from development to production. A software library is normally used to store the application source code and related artifacts. These library systems are typically designed to store multiple versions of these parts. The concept of an application version is well established in the context of software libraries and build processes.

With WebSphere Extended Deployment, it is possible to store these application versions in the system management repository and deploy them as needed.

Note: The application edition manager provides limited support for unmanaged Web applications. This means that applications that run on middleware platforms other than WebSphere Application Server and PHP cannot leverage most of the edition manager capabilities such as managed rollout and edition validation.
9.5.1 Application editions

An application edition represents a unique instance of an application in the WebSphere Extended Deployment environment. An application edition encompasses both application versions and deployment bindings.

Edition names
The application edition manager enables you to install multiple editions of the same application. Each edition is identified with an application edition name and description. The edition name is a free-form text field in which you can specify a value to uniquely identify one application edition distinctly from other editions of the same application. A version number scheme, such as 1.0, 2.0, and so on, may be a useful approach for naming editions, but you are free to employ whatever scheme is most useful in your environment. An application installed with no edition name given is called a base edition.

When deploying an application you can also specify an edition description next to the edition name, which gives you the ability to store additional information.

Non-destructive update
The existing application install and update functions in Network Deployment are destructive: they replace the old instance of the application with a new one. Installing an application edition is additive: you may install any number of application editions and keep them in the system management repository.

9.5.2 Split deployment

Split deployment refers to the case when the modules deployed in a single J2EE application archive are divided across two or more deployment targets. For example, an EAR file containing a Web application module and an EJB module is installed in the WebSphere Application Server environment. The Web application module is installed on a server and the EJB module is deployed on a cluster. With the application edition manager it is possible to install and activate applications that are split as described.

9.5.3 Installing a new edition—full lifecycle

This section discusses using the application edition manager for WebSphere application servers and PHP servers. With full lifecycle support, applications are installed from the deployment manager.

Installing a new application edition is similar to installing a new application in other WebSphere editions such as Network Deployment.

7. Go to Applications → Install New Application in the administrative console.
8. Select your application’s EAR file, and click **Next**. Make any necessary changes on the Preparing for the application installation panel, and then click **Next** again.

9. Click **Continue** if you receive the Application Security Warnings panel.

10. The next panel differs from the traditional application installation procedure. You can specify the edition name/number and description. If no Application Edition name or number is entered, then this is considered the base edition. Make sure that you specify the same application name as for your previous edition plus an Application Edition number and optionally an Edition Description, as shown in Figure 9-13:

![Figure 9-13 Application edition installation](image)

**Tip:** Although possible, we do not recommend entering a space in the application edition name because WebSphere Extended Deployment creates a directory that includes the name of the ear file and the name of the application editor. Values such as 1.0 or 1.1.0 or 1.0.1_FIX are less problematic.

11. During installation of your new edition, you are asked if you want to clone existing work classes. If you created work classes for your application prior to
installing this new edition, you should clone them to the new edition. Otherwise, you have to create them manually again. See Figure 9-14. More on work classes can be found in section 10.2, “Work classes” on page 217.

![Clone Existing Work Classes](image)

Figure 9-14  Clone existing work classes

12. Step through all other necessary installation steps to finish the installation of the application edition.

After the installation completes, the application edition is not yet active. Section 9.6, “Application activation and rollout” on page 198 discusses the options for activation and rollout.

### 9.5.4 Defining and activating a new edition—assisted lifecycle

This section discusses using the application edition manager for servers with assisted lifecycle support. With assisted lifecycle support, applications are not installed from the deployment manager. Existing applications on the server are defined to the cell manually.

The application edition manager has limited support for applications that reside on these servers. It does not support application rollout to these servers nor does it provide edition validation support. WebSphere Extended Deployment describes these applications as unmanaged Web applications.

The following scenario describes how the edition manager could be used with unmanaged Web applications where there are two or more versions of the same application. In the following scenario each version of the application resides on a different deployment target. For example edition-1 of the application runs on
dynamic_cluster-1, and edition-2 runs on dynamic_cluster-2. You can route requests to one edition of the application or to a different edition of the application based on some criteria that is defined via routing rules.

To accomplish this, define each edition of the application to WebSphere Extended Deployment. Next define routing rules to be used by the ODR that determine which requests were routed to one version of the application and which requests were routed to a different version. For instance you could have requests from one set of client IPs routed to edition 1.0 of your application and have requests from a second set of client IPs, which were assigned to a user test group, routed to edition 2.0 of your application. It is important to note that the deployment targets for each specific edition must be different.

1. To install a new edition of a middleware application, go to Applications → Install New Middleware Application in the administrative console.

   While the link is titled “Install New Application” you are not installing a new application. You are defining an existing middleware application to WebSphere Extended Deployment. You must have already manually deployed the application to your middleware server.

   Defining the application provides WebSphere Extended Deployment with information about the application, allowing the ODR to route requests to the application.

   Select Unmanaged Web Applications as the application type, and click Next.

2. The next panel is the Define the general properties panel. Here you specify the application name as well as the application edition.

   To define a new edition of an application that was previously defined to WebSphere Extended Deployment, complete the following tasks:

   a. Specify the same application name that was used when the application was initially defined.

   b. Enter a new value for the Application edition.

      The value that is entered for the application edition can be any arbitrary string that allows you to identify the versions (for example, 2.0.0 or blue version, and so on).
Click **Next**.

3. Define the module name, the context root, the virtual host, and deployment targets where the new version is deployed. This is the same whether this is the base instance of the application or a new edition.

   Click **Next**.

4. Click **Finish**, and save your changes to complete the task.

   After the installation completes, you should see the new edition of your application listed. Note that the new edition is not yet activated.
Leveraging routing policies with application edition manager

In our test scenario we need to define criteria that will determine to which version of the application each request will be routed.

In this example, requests from the test user group should be routed to one edition of the application and have all other requests routed to the other edition. To do this we need to define a routing policy using the following steps.

1. In the list of applications, click the application name for which you want to add the rule. It does not matter which edition you select.

2. Click the Routing Policies tab.

3. Expand Work classes for HTTP requests, and then expand Default_HTTP_WC.
4. Click the **Add Rule** button.
5. Click the **Build subexpression** link.

---

**Figure 9-17** Application edition management for other middleware servers — New Routing Policies

**Figure 9-18** Build a routing rule
This causes a new browser window to open that will display the subexpression builder.

6. For this test, we use the Client IPV4 operand and set it to a range of IP addresses, including the one where our Internet browser resides. Use the pull-down menu to look at the other operands that are available.

Following are the values entered for this example:

- Select operand: Client IPV4
- Operator: Like (LIKE)
- Value 9.%.%.%

7. Click the **Build subexpression** button to populate the Generated subexpression field.

8. Copy the generated subexpression, and close the window.

9. Paste the subexpression into the empty routing rule box.
10. Specify which edition of the application to which you want the client to route.

11. Click the **Validate rule** button.

**Important:** You must click the **Validate rule** button or your rule will not be saved.

12. Update the following fields, which are the rules specifying that any requests that do not meet the rule are sent to TestApp-edition1:
   - Select action: Permit routing to
   - Select edition name: TestApp-edition1

Any requests from users with an IP address such as 9.%.%.% get routed to TestApp-edition2. All other requests are routed to TestApp-edition1.
13. Click **OK** at the top of the panel, and **save** your changes.

**Application activation**

After the middleware application and routing rules are defined, the final step in this scenario is to activate the new edition. Use the following steps to activate the new edition:

1. From the administrative console navigate to **Applications → Edition Control Center**.

2. Click the application name (TestApp in our scenario).

3. Select the check box next to the new edition of the application, and click the **Activate** button to change the application state to **Active**.
4. Save your changes.

5. Try submitting requests to the application. The requests should go to different versions of the application based on the client IP.

### 9.6 Application activation and rollout

The application edition manager separates distribution of an application from its activation. The first or base edition is automatically activated. Subsequent editions are distributed to a target server or cluster in an inactive state and are explicitly activated as a separate step.

Only a single edition can be active on a given deployment target at a specific time. Multiple editions can be concurrently active as long as they are on different deployment targets. When multiple editions are active, routing rules can be assigned to the application to instruct the ODR which application requests are to be sent to each edition.

There are four methods for activating an edition:

1. **Simple activation**
   
   Simple activation marks an application edition as available to be started. After it is activated, an edition can then be started as a separate step. Before activating an edition, the current activated edition has to be deactivated. See section 9.6.1, “Simple edition activation” on page 199 for more information.

2. **Concurrent activation**
   
   Concurrent activation enables you to activate the multiple editions of the same application on different servers or clusters. For more information about this topic see section 9.6.2, “Concurrent activation” on page 200.
3. Validation activation

Validation activation is a special form of concurrent activation. It activates an edition on a clone of its original deployment target. The clone is created on activation. After the validation rollout to the original deployment target, the clone is removed automatically. See section 9.6.3, “Validation mode” on page 202 for more information.

4. Rollout activation

Rollout activation activates one edition in place of another, ensuring an interruption-free update in the process.

There are two different types of rollout activation:

- Group rollout
- Atomic rollout

See section 9.6.4, “Rollout activation” on page 204 for more information.

9.6.1 Simple edition activation

An application is distributed to the deployment targets at the time of installation. Activation is a configuration change that marks the application to become eligible to be started.

Following are the four steps to use to do a simple edition activation:

1. Stop the current application.

   You stop an application on the Enterprise Applications panel. Go to **Applications → Enterprise Applications**, select the application, and click **Stop**.

2. Deactivate the current edition.

   You deactivate an edition by selecting the edition to be deactivated and clicking the **Deactivate** button in the application edition control center. Go to **Applications → Edition Control Center → application_name**. The edition control center is shown in Figure 9-23 on page 200.
   You activate an edition by selecting the edition and clicking the **Activate** button in the application edition control center.

4. Start the new application.
   You start an application by selecting the application and clicking the **Start** button on the Enterprise Applications panel.

**Attention:** This approach is not interruption-free. If you want an interruption-free activation follow section 9.6.4, “Rollout activation” on page 204.

### 9.6.2 Concurrent activation

Multiple editions of the same application can be concurrently activated on different deployment targets. For example, edition 1.0 of an application may be deployed on dynamic cluster XDCluster1, and edition 2.0 on dynamic cluster may be deployed on XDCluster2. Or you may want to have the same application version installed twice in different clusters—for different customers or departments.

**Note:** Only one active edition of a PHP application is supported on a node. If you have active editions of the same PHP application, do not deploy the application to servers that are on the same node.

**Separating user requests among concurrent editions**

When multiple editions of the same application are concurrently available to users in the same environment, the ODR needs some kind of information to differentiate between the active editions. Based on that information it then routes
the request to the intended edition. This can be accomplished by using either routing rules or unique URIs.

**Routing rules**

Routing rules are the recommended mechanism to separate request traffic among concurrent editions. The benefit of using routing rules is that they are only visible to the administrator. The application users can access a consistent set of application interfaces, meaning Web application URIs and EJB JNDI names, and still be directed to distinct editions as per the strategy established by the application administrator. For this reason, routing rules are considered administratively directed routing differentiation.

Routing rules are discussed further in section 10.7, “Routing policies” on page 240.

In Figure 9-24 you can see two clusters hosting different application editions. The ODR delivers the requests based on the defined routing rules. These routing rules could be based on a unique Uniform Resource Identifier (URI), client IP address, or other criteria.
**Unique URIs**
For cases where routing rules are insufficient to satisfactorily separate user requests, or if you prefer a more familiar mechanism instead of routing rules, you can give each edition its own unique URIs and EJB JNDI names. When doing this, however, the unique interfaces (Web application URIs and EJB JNDI names) for each edition are exposed to the application users and they must choose the right name(s) to use the appropriate edition.

For this reason, unique interfaces are considered *client directed routing differentiation*.

**Activating an edition in concurrent mode**
You need to set up routing rules to have multiple editions of the same application running in a WebSphere Extended Deployment environment. You must create the routing rules *before* you activate the new edition. Otherwise the ODR will route all requests to the new edition.

You need to specify different deployment targets for your concurrent editions during installation as you can only have one active edition in a single cluster.

### 9.6.3 Validation mode
Validation mode is a special case of concurrent activation that enables you to perform final pre-production testing of an application edition in the actual production environment with a selected set of users. This is accomplished by cloning the actual deployment target, including its resource and security definitions, and then activating the target edition on the cloned environment. Routing rules direct the ODR to divert a selected subset of users to the new edition.

**Note:** Only the application and the resources within WebSphere Extended Deployment are cloned. External components such as third-party software or databases are not cloned!

Figure 9-25 on page 203 is similar to Figure 9-24 on page 201 except that edition 2.0 is installed on a cloned cluster in the production environment. Again, the ODR delivers the request to the clusters based on the defined routing rules.
When your validation mode testing is deemed successful and you want to rollout the application, you can perform the rollout directly against the edition in validation mode. The edition is then rolled out on the original deployment target and the cloned environment is deleted. If you want to maintain the cloned validation cluster after you roll out the new edition, set the dynamic cluster custom property, `saveClonedCluster=true`.

**Important:** Remember that you cannot roll back any data changes that happened during a test in your production environment. So be very careful when it comes to changing data—it is much better to only read information/data.

### Activating an edition in validation mode

In order to validate an edition, you need to install the application as described in section 9.5.3, “Installing a new edition—full lifecycle” on page 188.

Before you validate a new edition you need to set up routing rules. Otherwise the ODR will start routing requests to the new application. For example, you can define a routing rule so only certain client IP addresses are routed to the new edition.
In the application edition control center (Applications → Edition Control Center → application_name) select your new edition, and click Validate as seen in Figure 9-23 on page 200.

A new dynamic cluster is created with the same specification as your original cluster. You can change the cluster configuration if needed. This might be useful if you want to use a different database, for example.

**Note:** If the application in validation mode is not good and you do not want to roll it out, you need to uninstall it manually, stop the running application server, and delete the validation cluster.

### 9.6.4 Rollout activation

Interruption-free rollout enables a smooth transition from one edition of an application to another without loss of service. This means that all application requests are serviced during the rollout—none are lost. This ensures the perception of continuous application operation from the perspective of the application's customers. To do this, the application edition manager carefully coordinates the activation of the edition and the routing of requests to the application.

Interruption-free rollout is supported only for HTTP and SOAP/HTTP driven application requests.

You can configure rollout options. To see the options, select Applications → Edition Control Center → application_name. Select your new edition, and click Rollout. This opens the option page illustrated in Figure 9-26 on page 205.
We discuss the options in Figure 9-26 in more detail in the following sections.

**Application edition compatibility**

Replacement of one edition with another in a production environment requires certain discipline in the application's evolution. Because edition replacement happens while application users are potentially accessing the previous application edition, the new edition should be backward compatible with the old edition. New interfaces may be added. Existing interfaces may be algorithmically corrected, and in some cases, even extended, and still remain compatible with older application users.
**Group rollout strategy**

When using the group rollout strategy, the administrator defines the number of servers in the deployment target. That is, how many clustered application servers the application edition manager should update at the same time. This is called the group size. For example, you can specify to rollout a new edition to a server cluster using a group size of two. If you have a four server cluster, this means that the new edition is rolled out in two passes, two servers at a time. The default group size is one. The group size cannot exceed the number of servers comprising a deployment target.

Naturally, servers that are changed from one edition to another are offline and not processing application requests during the transition. Therefore, you should tailor the group size setting to get the best balance of capacity availability and rollout expedience for your environment. A smaller group size provides greater capacity during rollout. A larger group size updates more servers at a time, thus reducing the total time required to complete the rollout. The group size number always depends on the environment’s requirements and therefore we cannot recommend a best or right group size number.

---

**Note:** A basic requirement for interruption-free rollout is that you use the ODR. In other words, interruption-free rollout is only supported for applications that are accessed through the ODR. This is because the ODR acts as a routing agent. So the flow of requests may be controlled during the rollout process, diverting requests away from servers that are transitioning from one active edition to another and queuing requests if there is temporarily no server available to process the application request.
Atomic rollout strategy

Atomic is a rollout strategy whereby all user requests are served by the same application version. After it is online, the active new edition completely replaces the old edition. For a server cluster, this is done by releasing (rolling out) the new edition to half the cluster at a time. The previous edition is served until the first half of the cluster is available to serve the next edition. At that time, the old edition is taken offline and the remaining half of the cluster is released. So the cluster either serves requests exclusively from the original edition or exclusively from the new edition. It would never serve editions from both editions concurrently.

There might be a window in which there are temporarily no servers available to process requests for the application. During this time, requests are queued in the
ODR until the servers are brought back online. This rollout strategy also ensures no loss of service for single server deployment targets.

Figure 9-28 Atomic rollout sequence
Use the atomic rollout option if it is important for you to serve all user requests with a consistent edition of the application. However, this means that your cluster runs at half capacity. If your cluster is very large, group rollout might be the better option for you.

**Attention:** A basic requirement for the atomic rollout is that all application servers are started. WebSphere Extended Deployment does not verify which servers are currently up and which are down. As a worst case scenario, imagine that only half of your application servers are running and these are shut down for the update first!

### Activating an edition using group or atomic rollout

To rollout a new edition, perform the following steps:

1. Install the application as described in section 9.5.3, “Installing a new edition—full lifecycle” on page 188.
2. Select **Applications → Edition Control Center → application_name**.
3. Select your new edition, and click **Rollout**. This opens the following option page.
4. Choose either **Group Rollout** or **Atomic Rollout**.

The SystemOut.log of the deployment manager contains detailed logging information about the rollout process. The administrative console will also display some information, as shown in Example 9-1.

#### Example 9-1  Group rollout output

WPVR0015I: Processing node1/RedbookCluster11_node1.
WPVR0025I: Draining node1/RedbookCluster11_node1.
WPVR0020I: Synchronizing node1.
WPVR0015I: Processing node2/RedbookCluster11_node2.
WPVR0025I: Draining node2/RedbookCluster11_node2.
WPVR0020I: Synchronizing node2.

Manage Editions: XDStockTradeEdition

---

You can also see which applications are active in this file:

$WASHOME/config/cells/cell_name/applications/application_name/ibm-edition-metadata.props

# File contains metadata for all editions of the application
# Wed Nov 09 16:39:32 EST 2005
config.state-edition2.0=INACTIVE
edition.desc-edition2.0=Version 2
cfg.state-editionNew_2.0=INACTIVE
edition.desc-editionNew_2.0=New 2.0
cfg.state-edition1.0=ACTIVE
edition.desc-edition1.0=Initial Edition2

---

### 9.6.5 Reset strategy

A reset strategy instructs the application edition manager in how each of the deployment targets will load the new edition into the server run time. Reset is accomplished by recycling either the application in each affected server or by recycling the entire server itself. Recycling is the combined action of stopping, and then restarting the affected object.

The reset strategy is not executed until there are no active requests in the server. Client affinities to the server are allowed to complete according to the drainage interval, which is described in the section that discusses “Drainage interval” on page 211.

The following two different reset strategies available:

- Soft reset
- Hard reset

#### Soft reset

A soft reset attempts to minimize the impact of edition transition in a server by recycling the affected application. This allows the server to continue serving requests for other applications, if there are any. It also avoids the system impact of restarting the entire server.

It is important to note that certain supporting elements in the environment do not benefit from a soft reset. Specifically, certain native resources do not get cleaned...
up. In particular, process storage containing code that is JIT is not cleaned up; furthermore, native libraries are not unloaded from memory. Soft reset is generally safe for applications that use no native libraries. When soft reset is used in a production environment, we recommend that you monitor the application server process(es) to ensure there is sufficient virtual memory.

**Hard reset**
A hard reset recycles the *entire application server*. This impacts all applications running on that server, since they will all go offline while the server is recycled. Hard reset ensures that both process memory and any native libraries used by the application are refreshed. This prevents virtual storage exhaustion and allows for new versions of native libraries to be loaded. When rolling out an application edition that is accompanied by new versions of native libraries upon which it depends, you must select hard reset as your reset strategy for rollout.

**Drainage interval**
As the rollout process is initiated in a particular application server, the server is removed as a candidate for new requests for the application being updated. The drainage interval specifies the amount of time an application server is allowed to serve clients with affinity to that server after the rollout process has begun and before the reset strategy is executed.

In addition, certain types of affinities, such as transaction, activity, and compensation-scope will lengthen the effective drainage interval, since the server does not actually stop until these units of work complete.

The default drainage interval is 30 seconds.

**9.6.6 Recovery from failure during a rollout**
There is no provision for automatic rollback for recovery. If a failure takes place during rollout, then the rollout fails. The administrator needs to rollout the previous edition manually using the back-out process described in the next section.

**9.6.7 Application back-out**
After an edition rollout is complete, you might find that there is a problem with this application edition and you want to revert back to the previously active edition. This is called *back-out*. Back-out is accomplished by simply rolling out the previous edition.
Back-out is not a routine action. It is an action taken to recover from unexpected application results that are serious enough to require reverting back to a prior edition.

**Important:** Reverting to an older edition might cause some compatibility disruption to new clients that are expecting access to the new edition.

### 9.6.8 Application activation and rollout with scripting

It is possible to use wsadmin scripts for application activation and rollout, which are AdminTasks that are built into wsadmin.

The following tasks can be used:

- **activateEdition**
  
  Batch mode example usage:
  - Using Jacl:
    ```
    $AdminTask activateEdition {-appName BeenThere -edition 1.0}
    ```
  
  - Using Jython:
    ```
    AdminTask.activateEdition('[-appName BeenThere -edition 1.0] ') 
    ```

  Interactive mode example usage:
  - Using Jacl:
    ```
    $AdminTask activateEdition {-interactive}
    ```
  
  - Using Jython:
    ```
    AdminTask.activateEdition('[-interactive] ')
    ```

- **deactivateEdition**
  
  Batch mode example usage:
  - Using Jacl:
    ```
    $AdminTask deactivateEdition {-appName BeenThere -edition 1.0}
    ```
  
  - Using Jython:
    ```
    AdminTask.deactivateEdition('[-appName BeenThere -edition 1.0]')
    ```

  Interactive mode example usage:
  - Using Jacl:
    ```
    $AdminTask deactivateEdition {-interactive}
    ```
  
  - Using Jython:
    ```
    AdminTask.deactivateEdition('[-interactive]')
    ```

- **rolloutEdition**
  
  Batch mode example usage:
  - Using Jacl:
    ```
    $AdminTask rolloutEdition {-appName BeenThere -edition 1.0 -params "{{rolloutStrategy [grouped|atomic]}{resetStrategy [soft|hard]}{groupSize [int]}{drainageInterval [int]}}"}
    ```
9.7 Tracing the application edition manager

You can activate logging and tracing for the application edition manager using the following setting:

```
com.ibm.ws.xd.appeditionmgr.* = debug
```

You can either change the trace level through the administrative console or using a wsadmin script.

For information about how to use scripting in WebSphere Extended Deployment see:

- **Getting started with scripting**
  
The following articles in the WebSphere Application Server V6.1 Information Center can help you enable a trace:

- **Enabling trace on a running server**
  

- **Enabling trace at server startup**
  
Operational policies

This chapter covers service policies and routing policies, which are used in conjunction with the ODR to provide the desired quality of service for your applications. Service policies define goals used to achieve service levels. They classify and prioritize requests. Routing policies define how work is routed to application editions or generic server clusters.

This chapter includes the following topics:

- Service policies
- Work classes
- Implementing service policies
- Managing transaction classes
- Managing work classes
- Test the service policy
- Routing policies
- Verifying service policies and routing policies
10.1 Service policies

WebSphere Extended Deployment uses a service policy to categorize and prioritize work requests. Through service policies, you can designate the performance goal and business importance of different request types. WebSphere Extended Deployment uses this information to manage the environment and to help achieve the goals defined in service policies. Service policies also allow the performance of the environment to degrade in a controlled manner in periods of over-utilization.

Service policies are assigned response time targets that are valid for specified throughput conditions. The performance management done by the ARFM, the dynamic workflow manager, and the APC achieves a defined balance of the performance results.

The service policy creates the goal, while work classes connect specified information such as URIs (Uniform Resource Identifiers) to that goal. IIOP type work classes use EJB and EJB method names to map to the goal. JMS type work classes use bus and destination names to map to the goal.

10.1.1 Transaction classes

Service policies contain one or more transaction class definitions. The service policy creates the goal, while the transaction and work classes are used to map requests to that goal.

Transaction classes are a subcontainer of the service policy for work being classified into the service policy that can be used for finer-grained monitoring. They can also be used as a mechanism of grouping cross application work together for common monitoring.

The relationship between service policies and transaction classes is one to many: A single service policy can have multiple transaction class definitions, but each transaction class belongs to exactly one service policy.

Every service policy has a default transaction class, which in most scenarios is sufficient. Additional transaction classes are created when finer grained monitoring is necessary for the environment. Each transaction class name must be unique within the cell.

Figure 10-1 on page 217 shows the relationship between service policies and transaction classes. The URIs are grouped together in work classes. When a request for a specific URI arrives, this URI is checked against the classification rules. Based on these rules different transaction classes are addressed. These
transaction classes are uniquely assigned to a service policy. Based on the service policy the request is processed. The ARFM performs these calculations.

### Quick Overview of Classification Terms

<table>
<thead>
<tr>
<th>Module</th>
<th>Work Class</th>
<th>Transaction Class</th>
<th>Service Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE Module A</td>
<td>URI group</td>
<td>Rules</td>
<td>Gold</td>
</tr>
<tr>
<td>J2EE Module B</td>
<td></td>
<td>Rules</td>
<td>Silver</td>
</tr>
<tr>
<td>J2EE Module C</td>
<td></td>
<td>Rules</td>
<td>Bronze</td>
</tr>
<tr>
<td>J2EE Module D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 10-1  Classification terms overview**

### 10.2 Work classes

Work classes are a mechanism for grouping specific work together that must be associated with a common service policy or routing policy. When a WebSphere J2EE application is installed, a default work class is created for each protocol discovered to be supported in the application. For example, if the application has only Web modules, a single default HTTP work class is created. If the application contains EJB modules an IIOP work class is created. Note that a default JMS work class is always created for each application because it is possible to have a JMS resource without a resource reference in the application definition. For applications that run on platforms other than WebSphere Application Server, only an HTTP work class is created.
It is not necessary to associate the whole J2EE Web application to a service policy in WebSphere Extended Deployment, but you can define a finer and more granular mapping by using work classes in conjunction with transaction classes.

Work classes combined with classification rules allow the ARFM to prioritize a request. For example the `/shop/checkout` URI should get more resources than the `/shop/info` URI. This could be necessary because the checkout takes more time or because you think the checkout is more important.

There are four possible types of work classes based on the supported protocols in the application:

- HTTP work classes
- SOAP work classes
- IIOP work classes
- JMS work classes

Note: For applications running on platforms other than WebSphere Application Server, only work classes based on the HTTP protocol are supported.

Work classes can have classification rules. The rules classify the requests by a number of variables:

- Virtual host or Uniform Resource Identifier (URI)
- HTTP headers, query parameters, and cookies
- Web service and operation name
- Client or server IP address, port, and host names
- User ID, group IDs, and roles
- Protocol
- Virtual portal
- Time
- EJB name or method

You can define work classes for service policies and for routing rules.

### 10.3 Implementing service policies

Before creating service policies, you must look at your applications and environment. WebSphere Extended Deployment cannot guarantee that your service goals are met with absolute certainty, but it will strive to manage the system in the best way possible to make it happen. Therefore, the best practice is to run performance tests for your applications and determine their performance profiles on your production environment before defining service policies.
After you analyze your applications and you know which applications or parts of applications you want to have a higher priority, you can then create the service policies.

In summary, the process to implement a service policy is as follows:

1. Evaluate the applications and the desired service levels for your environment, and design a strategy for assigning service policies to your applications.
2. Create a service policy and define the transaction classes for the policy.
3. For each work class in the application, define work class rules to classify requests and associate the request to a transaction class (and by extension, a service policy).

If the URL of the request matches the pattern defined in the rule, then the request is classified as belonging to the transaction class defined for the rule.

### 10.3.1 Service policy definitions

Service policy definitions are made up of the following two key items:

- **Goal**

  The goal portion of the service policy defines how incoming work is evaluated and managed in order to ensure that and detect if work is meeting its assigned service policy levels. Service policies can have four different kinds of goals: discretionary, average response time, response time percentile, or queue time. These are explained in detail in “Service goal types” on page 219.

- **Importance**

  Importance is used in times of resource contention to identify the most important work in the system and to give it higher priority. The options for importance vary from lowest to highest.

Administrators must have an in-depth understanding of deployed applications so that they can create realistic performance goals.

### Service goal types

There are four performance goal types in IBM WebSphere Extended Deployment V6.0:

- **Discretionary** indicates work that does not have significant value. Requests are processed when no higher request is waiting. As a result, work of this type can see a degradation in performance when resources are constrained. This is the default service goal.
- **Average response time** allows you to specify the average response time goal in milliseconds or seconds. The system will try to achieve this goal at a target percentage of 90%.

  For example, if you specify two seconds as the goal value, 90% of the requests must be answered in less than two seconds to meet the goal. If you need a different percentage, use the percentile response time goal to set it.

- **Percentile response time** allows you to specify both the average response time goal and the target percentage. For example, 95% of all requests should be answered in less than 1000 milliseconds. This performance goal type is useful for applications that have application response times that occasionally deviate from the norm and can skew the average response time.

- **Queue completion time** is a service goal for long-running applications. When the goal reaches this limit, more servers are needed. Note that this option is only displayed if the WebSphere Compute Grid component is installed in your environment.

**Importance**

Administrators can specify the relative level of importance of a service policy. A request associated with a service policy of higher importance is given priority over a request associated with a service policy of lower importance. This guarantees that if performance goals for all service policies cannot be met due to prolonged intense overload to your environment, WebSphere Extended Deployment can use importance to decide which service policy takes priority.

You can select from the following seven importance levels:

- Lowest
- Lower
- Low
- Medium
- High
- Higher
- Highest

Some planning is essential to select the right importance value because negative results can occur if all work is rated as highest. Such a rating can create a bottleneck within the environment. A better approach might be to leave the majority of your applications with a discretionary goal, to assign a higher goal to the important applications via service policies, and to use the highest importance levels only if you need further differentiation between the higher goal applications.
10.3.2 Creating a service policy

This section takes you through the process of creating a service policy using the administrative console.

**Scripting:** Service policies can also be managed with scripting. A python script is provided to manage service policies. The script is located in `app_server_root/bin/` and is called `servicepolicy.py`.

For a detailed description of the script, its operations, and options as well as examples, see the following Information Center article:

- *Managing service policies with scripts*
  

You can access service policies in the administrative console by navigating to Operational Policies → Service Policies (Figure 10-2).

![Service Policies overview panel](image)
Use the following steps to create a service policy:

1. Click **New**.
2. Define the general property values (Figure 10-3):
   a. Enter a name and description for the new service policy.

   **Note:** The name field cannot contain special characters and cannot begin with a period (.) or a space. Leading and trailing spaces are automatically deleted.

   b. Select a goal type:
      – Average response time
      – Discretionary
      – Percentile response time

3. Click **Next**.
4. Define the properties specific to the goal type.
   – The options for the goal value will vary depending on the goal type you selected. Enter a realistic goal value.
   – Select the importance level.
If you want to monitor for persistent service policy violations and have a runtime task created, check the Monitor for persistent violation box, and enter values for the goal delta and time period.

WebSphere Extended Deployment v6.1 offers two new options when defining a new service policy:

- **Goal Delta Value**: This is the allowable amount of time difference between the configured goal value and the actual average response time of requests that are served.

- **Time Period Value**: This value signifies how long that goal delta value can be violated before it is considered breached, and a runtime task is generated.

The intent of these two new options is to ensure some tolerance for any sort of temporary anomaly, before the APC component takes an action, such as starting a server.

5. Click **Next**.

6. Define service policy membership.

In this step a transaction class is defined for the service policy. You can select the default transaction class, or create a new one. You can have more than one transaction class. This is usually done in order to chart metrics at a more fine-grained level.

A default transaction class is automatically added to the membership (Figure 10-5 on page 224).
Figure 10-5  Define service policy memberships (transaction classes)

See section 10.1.1, “Transaction classes” on page 216 for more information about transaction classes.

To create a new transaction class click **New**. This will take you through a two-step process that defines a new transaction class. The only parameters you enter are the name and the description.

To continue with only the default transaction class, click **Next**.

7. Verify the summary, and click **Finish**.

8. Save and synchronize your new configuration. The new service policy will appear in the list of service policies.
10.4 Managing transaction classes

Transaction classes can be managed in the administrative console from the service policy configuration.

1. Navigate to Operational Policies → Service Policies.

2. Select the appropriate service policy to open the configuration page for it. The transaction classes for the policy is listed at the bottom of the page. (Figure 10-7 on page 226)

You can create, remove, or modify transaction classes from this page. You can also move a transaction class from one service policy to another.
You can create one or more transaction classes for your application and map the service policy to these transaction classes. It is possible to move the transaction classes from one service policy to another when you find that some transactions need a higher prioritization.
10.5 Managing work classes

Applications have default work classes defined for each appropriate protocol. You can change the default work class and add new work classes. If more than one work class is defined to handle a request, the request is assigned to the higher service policy.

**Tip:** Name the transaction classes according to their application and their prioritization within this application to make the transaction class to service policy mapping easier.

Imagine that you have more than one transaction class per application, and the applications have different priorities within your company. Let us assume the following:

- Application 1 has two transaction classes: tc_app1_high and tc_app1_mid.
- Application 2 has three transaction classes: tc_app2_high, tc_app2_mid, and tc_app2_low.
- It is most important that Application 2 has a good quality of service.

Based on these assumptions, the mapping could look as follows:

- tc_app2_high is mapped to sp_platinum
- tc_app2_mid and tc_app1_high are mapped to sp_silver
- tc_app2_low and tc_app1_mid are mapped to sp_bronze

**Note:** Modify the default work class if you want to change the service policy for the whole application. However, you cannot change the URIs, IIOP pattern, JMS pattern, or Web services associated with the default work class; instead, you can only alter the transaction class and service policy to which it maps.

You must create a new work class if you want to assign different patterns (URIs, IIOP, JMS, and SOAP requests) in your application to different service policies.

10.5.1 Creating work classes

This section discusses creating work classes using the administrative console.
Scripting: It is also possible to edit work classes using a python script called workclassoperations.py. The script can be found in the app_server_root/bin directory.

You can find detailed information and examples about the use of this script in the output of this command or in the following Information Center article:

- Managing work classes with scripts

Use the following steps to create a work class:

1. Go to Applications → Enterprise Applications in the administrative console.
2. Click the desired application, and select the Service Policies tab.
3. Expand the work class type that you want to create (for example the work classes for HTTP requests. For example, expand Work Classes for HTTP Requests.

![Figure 10-8 Work class to service policy assignment](image)
4. Click **New**.

5. Enter a name for the work class, and click **Next**.

6. Select the application module you want to create the work class for.
   - *Full lifecycle support*: The modules in the application are available in the pull-down menu.
   - *Assisted lifecycle support*: You will only see the module you specified while defining the application to Extended Deployment.

7. Add the URIs to match to this work class. Select one or more URIs from the list, and click **Add >>**.
   - *Full lifecycle support*: The URIs are listed for the selected application module.
   - *Assisted lifecycle support*: The Extended Deployment configuration does not have the applications deployment descriptors, so it is not aware of the structure of the application. No URIs will be listed.

   For each URI you want to add to the work class, enter the pattern in the Custom URI Pattern field, and click **Add Pattern >>**.

   While defining the custom HTTP pattern, the URI entered is appended by the context root of the application. For example if the context root of Application A is /Platinum, then any entry in the custom HTTP Pattern follows the /Platinum context root.
8. Click **Next**.

9. Click **Finish**.

### 10.5.2 Assigning a transaction class to a work class

After you finish creating the work class, you can assign a transaction class (and by extension, a service policy) to it. The service policy and transaction class must already exist.

Use the following steps to assign a transaction class to a work class:

1. Navigate to **Applications → Enterprise Applications**.
2. Click the desired application, and select the **Service Policies** tab. Refer to Figure 10-8 on page 228.
3. Expand **Work Classes for HTTP Requests**.
4. Expand the work class.
5. Scroll down, and select your transaction class from the **Select Transaction class** pull-down menu. This is shown in Figure 10-10.

6. Click **OK** at the top.

7. Save with **Synchronize to nodes** selected.

**Note:** The information of this work class is stored at the following location:

```
app_server_root/profiles/dmgr_profile/config/cells/cell_name/applications/app_name/deployments/appname-editionname/workclasses/wc_name/workclass.xml
```
Work classes can be qualified by URI, IIOP patterns, JMS patterns, SOAP patterns and also by specific rules. Instructions for creating these rules are in section 10.5.3, “Work class request classification rules” on page 232.

10.5.3 Work class request classification rules

Work classes can be classified by rules. The syntax and semantics of a boolean expression for a rule are similar to the WHERE clause of a structured query language (SQL) expression. You can combine the expressions with operators.

Note: Classification rules are not supported for JMS work classes.

Expressions are formed from operators and operands. Operators act on the operands. Operands can be variables, strings, or constants. Following is an example:

\[ a = b \text{ and } c > b \]

The operands are a, b, c, and d. The operators are ‘=’, ‘and’ and ‘>’.

A table of the operands and operators supported in rules expressions can be found in the following article:

- Routing and service policies for work classes

Use the following steps to add a classification rule:

1. Expand your work class as shown in Figure 10-10 on page 231, and click Add Rule.

   A new line opens in the classification rules area.
2. You can type your rule directly into the window, or you can click **Build subexpression** to use the rule builder. Using the Rule Builder takes the guess-work out of selecting operands, operators, and format.

3. Click **Build subexpression** to use the Rule Builder.

   A new window opens, as shown in Figure 10-12 on page 234.
4. Use the pull-down menus to select a list of operators and operands required for your classification.

5. Click the **Build subexpression button** to generate the subexpression.

**Note:** The generated subexpression is not transferred to the rule you are building. You must copy the expression, close the window, and then paste the expression into the classification rule.
Figure 10-13  Build a subexpression for the rule

6. Copy the subexpression, and close the window.

7. Paste the subexpression into the classification rules window.

Figure 10-14  Add the subexpression to the rule
8. Select the transaction class from the pull-down menu.
9. Click **Validate Rule**.
10. Click **OK** (above the Work classes area on the panel).

Following are some classification rule examples (see also Figure 10-15):

- `clientipv4 = '10.1.1.1'
- `clienthost like '%.ibm.com'
- `clientipv4 = '10.1.1.1' and protocol = 'HTTPS'
- `port in (9080, 9090, 9091)

As the new rules are created, you will see the rule summary in the classification area.

![Figure 10-15](image)

**Tip:** Remember that if you use Web servers in your topology, the clienthost and client IP address that the ODR receives are from the Web servers. Therefore, you cannot create a rule based on client IP ranges when using Web servers.

### 10.5.4 Logging and tracing of routing rules

Logging and tracing information can be obtained by activating logging for this component:

```
com.ibm.ws.classify.*=all
```

You can either change the trace level through the administrative console or using a wsadmin script.
The following articles in the WebSphere Application Server V6.1 Information Center can help you enable a trace:

- *Enabling trace on a running server*
  

- *Enabling trace at server startup*
  

### 10.6 Test the service policy

Test the service policy and ensure that the APC is functioning by creating a condition that would cause the service policy goals to be exceeded. For example, assume a service policy was created that defined an average response time goal. You can create a situation where this goal is not met by making fewer resources available than are required to meet the service objectives. This can be done by having only one of the servers in a dynamic cluster active and using a load generating tool to create a high load.

This might be an iterative process to orchestrate the correct load to simulate a breach in service policy and cause subsequent corrective action to be taken by the APC to meet the service policy goals.

Set the operational mode of the dynamic cluster to supervised mode, so you can monitor how the APC reacts to the situation.

When the server gets overloaded, a task is generated by the APC component to start another server instance to meet the service objectives.

Use the following steps to set the dynamic cluster to supervised mode using the administrative console:

1. Navigate to **Servers → Dynamic Cluster**.
2. Select the Supervised mode from the pull-down.
3. Check the box to the left of the dynamic cluster, and click **Select Mode**.

While the load is being generated, you can view the response time metric of the dynamic cluster or the servers using the runtime monitoring features.

Use the following steps to configure customized charting report of the response time by the dynamic cluster:

1. Go to the administrative console.
2. Select **Runtime Operations → Reports**.
3. Click **Open new chart tab**.
4. Select **Add Data** to customize the organization of metric by data set.
5. Enter the desired metric(s) to be charted, and click **OK**.
   
   In our example, the following combinations were used:
   
   **Metric set 1**:
   - Data set type: Dynamic cluster
   - Data set: Tomcat_Platinum_DC
   - Metric: Average response time
   
   **Metric set 2**:
   - Data set type: Application server
   - Data set: XDTomcat1:xdtomcat_platinum1
   - Metric: Average response time
   
   **Metric set 3**:
   - Data set type: Application server
   - Data set: XDTomcat2:xdtomcat_platinum2
   - Metric: Average response time
   
   **Metric set 4**:
   - Data set type: Service Policy Goal
   - Data set: Platinum_Goal
   - Metric: Average response time
   
   This allowed us to see how the application servers and dynamic clusters were responding and how the service policy goals were being met in one chart.
6. After selecting the data set click **OK**.

After a short delay the response time metric can be seen, as illustrated in Figure 10-16 on page 239. The chart is a graph with time in milliseconds or seconds (depending on service policy configuration) on the Y-axis and time of day on the X-axis.
You should initially see data for the one active server. As we increase the load, a few spikes in the graph above the set service policy goal trigger the APC to kick off a task to start another instance of application server in a dynamic cluster.

A task is generated because the dynamic cluster mode is set to Supervise. To view the tasks list using the administrative console, select **System Administration** → **Task Management** → **Runtime Tasks**.

This panel (Figure 10-17 on page 240) shows all the tasks generated by autonomic and supervised actions. Action from the administrator is required before the task is executed.
You can click the task ID for more information about the task and to receive help. The information about the page includes the action plan that will be taken if you accept the task.

You can accept the action by changing the pull-down to **Accept**, and clicking **Submit**.

To ensure that the task of starting another instance of a middleware server was successful, verify the server status by selecting **Servers → Other Middleware Servers → Apache Tomcat Servers**.

**Figure 10-17  Task generated to start a server**

**Figure 10-18  Verify that the server was started**

### 10.7 Routing policies

Routing policies define how work should be routed to an application edition or to a generic server cluster.
With routing policies you can route requests to different editions of the same application, depending on specific rules (work classes) that you can define, or to restrict access to an application. For example, you can install the same application into two dynamic clusters and define a routing policy that routes a request for the application to a specific edition depending on the IP address.

Routing policies can be configured for each application including all of its editions. Routing policies are comprised of work classes and routing rules.

**10.7.1 Creating a work class and routing rule for a routing policy**

Creating a work class for a routing policy is similar to creating one for a service policy. To define a work class for a routing policy:

1. Open the administrative console, and select **Applications → All Applications**.
2. Click the desired application, then select the **Routing Policies** tab.

Work classes for HTTP and SOAP requests are supported for applications deployed to WebSphere Application Server. Work classes for HTTP requests are supported for applications deployed to other server types.

Follow the instructions from section 10.5, “Managing work classes” on page 227.

**Routing rules**

For each work class you must define a routing rule. There are four valid routing rules, as listed in Table 10-1.

<table>
<thead>
<tr>
<th>Routing rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permit routing to</td>
<td>Permits routing to every edition of this application.</td>
</tr>
<tr>
<td>Permit routing with affinity to</td>
<td>Permits routing to every edition of this application, but the ODR maintains client to server affinity.</td>
</tr>
<tr>
<td>Redirect routing to</td>
<td>With this routing policy the ODR redirects the request to another URL.</td>
</tr>
</tbody>
</table>
You can define specific routing rules in the same way that you define request classification rules (see 10.5.3, “Work class request classification rules” on page 232). You also have to define a general routing rule—that is, what should happen when no specific routing rules apply. Figure 10-19 on page 243 shows these configuration options.

<table>
<thead>
<tr>
<th>Routing rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reject routing with error code</td>
<td>The ODR rejects the request with a given error code. For example, 403 - Forbidden.</td>
</tr>
</tbody>
</table>
Figure 10-19  Defining a routing rule
10.8 Verifying service policies and routing policies

When you create service and routing policies it is difficult to see if they work or not. For our tests we generated load on our application using Apache JMeter. This way we could verify whether the policies worked in two ways: In the WebSphere Extended Deployment runtime topology and in the JMeter results.

As an example, we created a routing rule that allowed HTTP POST requests to be forwarded to the application, while HTTP GET requests were rejected with an HTTP error message, as shown in Figure 10-20.

![Routing rule](image)

**Figure 10-20  Routing rule**

Table 10-2 shows the JMeter results. You can see that all HTTP GET requests failed.

<table>
<thead>
<tr>
<th>URL</th>
<th>Count</th>
<th>Average</th>
<th>Min</th>
<th>Max</th>
<th>Error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>54</td>
<td>69</td>
<td>0</td>
<td>550</td>
<td>100.00%</td>
</tr>
<tr>
<td>POST</td>
<td>49</td>
<td>1355</td>
<td>100</td>
<td>3615</td>
<td>0.00%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>103</td>
<td>681</td>
<td>0</td>
<td>3615</td>
<td>52.53%</td>
</tr>
</tbody>
</table>
Monitoring operations

Visualization features in WebSphere Extended Deployment allow you to view the inner workings of the runtime environment. Administrators can see what types of decisions are made regarding deployment of applications and allocation of hardware.

Reporting is quite granular and can be set at different scopes, providing views with various intersecting data points. This can be particularly important and useful in problem diagnosis, manageability, and chargeability.

This chapter describes the visualization features in IBM WebSphere Extended Deployment V6.1. We discuss the following topics in this chapter:

- Introduction
- Runtime operations views
- Reports
- Visualization data service

For information about using the visualization features for chargeback in a WebSphere environment running applications from a number of business areas, see Best Practices for Implementing WebSphere Extended Deployment, SG24-7343, Chapter 2. Application hosting and chargeback.
11.1 Introduction

The visualization features of WebSphere Extended Deployment provide a wide range of functions designed to enhance the manageability of the complex runtime environments that you can build with WebSphere Extended Deployment.

- Runtime operation views in the administrative console provide the status of the Extended Deployment environment. These views are designed to provide the IT staff with valuable insight into the topology and status of the key elements.
  - A single view of all ODRs provides a summary of the stability and average throughput of each ODR. This can be especially helpful in observing anomalies in ODR operation.
  - A single view of all core groups provides valuable information useful in planning for scalability and capacity.
  - A single view of the core components provides a quick way to verify the stability of the Extended Deployment components, including the dynamic workload manager for dynamic clusters and autonomic controllers.
  - A single view of all nodes provides insight into the stability and CPU usage of each node.

- Reports can be created to show the performance of your environment over time. This feature appeals to various sections of IT operations and business communities alike. Dynamic charting allows Lines of Business (LOBs) to better plan and provision their IT resources. In the event of a problem, the charting and stability monitors provide a starting point to narrow down the problem domain.

- The visualization data service provides valuable usage data that can be mined to determine the chargeability per application or an enterprise resource. This is important for planning, controlling, and budgeting for IT resources. You can log statistics data for further use with external tools, for example spreadsheet programs.

- Operation alerts are immediately visible to the operations staff.

Tip: Although Internet Explorer® 6.0 and Mozilla 1.4 or 1.7 are supported browsers for the administrative console, we recommend that you use Internet Explorer to display the runtime operations views.

What is new in WebSphere Extended Deployment V6.1

Runtime topology and runtime maps were replaced with new reporting and operational summary features. You can view, at a glance, the runtime information for ODRs, nodes, core groups, and core components in your environment. Core
components include a variety of autonomic controllers and managers. In addition to viewing runtime information, you can create reports and charts to monitor statistics such as availability, response time, traffic, and throughput.

### 11.2 Runtime operations views

Runtime operation views enable administrators to view the health of cell-wide resources. This feature provides one view of the enterprise environment and is important in troubleshooting problems with Extended Deployment operations.

Runtime operations has extensive reporting capabilities that can be tailored to fit the interests of the technical and business communities alike. For example, reports can be generated for utilization in the chargeability analysis, which is usually performed by IT operations and business units.

To access the runtime operations data, navigate to **Runtime Operations → Extended Deployment** (Figure 11-1).

**Note:** Due to the dynamic nature of information collection, it may take a few minutes for the information to be displayed on the console. Be patient.
The runtime operation views contain the following distinct information areas:

- **Operations Alerts**
- **Extended Deployment component views.** Each view is accessed by clicking the corresponding tab.
  - On demand routers view
  - Core Groups view
  - Core components view
  - Nodes view
- **Reports**

Operations alerts are in the top portion of the panel. Clicking on the plus sign expands that area, so you can view the alerts. The remaining information areas can be accessed by clicking the relevant tab.

### 11.2.1 Operation alerts

Operation alerts were introduced in WebSphere Extended Deployment V6.1. They provide a brief summary of operational anomalies, alerts, and recommended actions. Depending on the severity of the alert, the administrator can drill deeper into the affected component and take a corrective action or establish an appropriate health policy.

Alerts are dynamic in nature and are removed from the view when the issues are resolved.

![Operation Alerts](image)

**Figure 11-2  Runtime operations alerts**
11.2.2 On Demand Routers view

While it is best practice to cluster ODRs for high availability and scalability, it is also vital that you ensure each ODR is tuned to perform optimally and maintain expected throughput. For example, we have seen in our labs that an increase in the number of clients can impact ODR throughput.

The On Demand Routers view provides a stability report of all the ODRs in a cell in a single view. It contains ODR stability and throughput information that can assist you in evaluating ODR strategy and topology. Each ODR is listed, along with its current stability rating. Clicking the ODR name opens the configuration page for the ODR (taking you out of the Runtime Operations views).

Note: Because the ODR is a vital component of the Extended Deployment environment, we recommend that you view this report on a routine basis to ensure the stability of the ODRs.

Figure 11-3 shows the On Demand Router view.

<table>
<thead>
<tr>
<th>Name</th>
<th>Node</th>
<th>Type</th>
<th>Stability</th>
<th>Avg. Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>xodr1</td>
<td>XDodr1</td>
<td>On demand router</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>xodr2</td>
<td>XDodr2</td>
<td>On demand router</td>
<td></td>
<td>0.0</td>
</tr>
</tbody>
</table>

Total 2

Figure 11-3  On demand router runtime operations view
The *Stability column* contains one of the following icons to indicate the state of the ODR:

- 🟢 **Stable** The ODR is operational, without issues.
- 🟢 **Questionable** The ODR has some problems, but is still operational. You may see this status when the node agent is not available. You may also see this if the ODR and node agent are active but are not routing any work and not posting statistics.
- 🟢 **Unstable** The ODR has severe issues.
- 🟢 **Unknown** The ODR stability cannot be reported.

The *Average Throughput column* specifies the average number of requests per second that the ODR routes.

### 11.2.3 Core Groups view

The Core Groups view provides a stability report of all the core groups in one view. This view can be helpful in troubleshooting problems that might occur when routing between multiple cells and servers that are configured with multiple core groups.

**Important:** Well-formed core groups are crucial when you are routing requests between different cells. When an ODR is used to route requests between cells in a multi-cell and multi-cluster environment, the server state information is delivered to the ODR by the core groups via a core group bridge. The ODR makes routing decisions based on this server availability information. Monitoring core group stability is vital to ensuring that the ODR gets the server availability information posted to the bulletin board (an HA manager construct).

Figure 11-4 on page 251 shows a Core Groups view.
The Name column specifies the name of the core group. To configure the settings for the core group, click the core group name.

The Stability column uses icons to specify the operational status of the core group.

- **Stable** The core group is operational, without issues.
- **Questionable** The core group has some problems, but is still operational. Click the core group to display the members. Check that each node agent is active.
- **Unstable** The core group has severe issues. This state indicates several issues such as problems with view synchrony protocol or inability to view other core group members due to network issues.
- **Unknown** The Core group stability cannot be reported. This state is typically reported when a node is out of commission.

Note that in Figure 11-4, the core group is unstable. There is a corresponding message in the alerts that provides information and suggests remedial action to resolve the issue.

Core groups maintain a view of all servers using the view synchrony protocol (VSP). In this example, many of the nodes are out of the view. Using the
suggestion in the operations alert, the operations staff will find that the core group has several members that are not in the view. A minimum number of members need to be available in a core group for a stable condition. This number is configurable using core group policies. The operator in this example should take the necessary steps to ensure that the servers are running and the channels are tuned. This ensures that the view of each node member is in sync.

11.2.4 Core components view

The Core components view provides a stability report for all the core components in a cell. The core components include the DWLM controllers, the ARFM controller, the health controller, the APC, and work profiler controller.

The core components are HA managed. If the process hosting the controller becomes unavailable, the HA manager starts the controller on another available process. The view shows where the controllers are running. The information is updated every time a component is moved to a different node in a cell or core group.

**Note:** This view is a good starting point for diagnosing problems with the Extended deployment core components.

The Core components view is shown in Figure 11-5 on page 253.
Figure 11-5  Core components runtime operations view

- The **Name column** specifies the name of the autonomic manager, controller, or component.
- The **Scope column** indicates the operational scope in which the component operates. For example, the scope can be a certain cell or dynamic cluster.
- The **Stability column** specifies the operational status of the core component.
  - ✅ **Stable** The core component is operational, without issues.
  - ⚠ **Questionable** The core component has some problems, but is still operational.
- **Unstable** The core component has severe issues.
- **Unknown** The core component stability cannot be reported.

> The *Current location* column tells you where the component is active.

### 11.2.5 Nodes view

The Nodes tab provides a complete inventory and stability report of all the nodes in a cell. It contains information about the node name, type, WebSphere packages installed on the node, CPU utilization, and whether the node is in maintenance mode.

From a troubleshooting standpoint this would be the first place to start to get an inventory of the environment and assess overall health of all the nodes.

Figure 11-6 on page 255 shows the Nodes view.
Figure 11-6  Node runtime operations view

- The **Name column** specifies the node name.
- The **Agent column** specifies the type of agent that is running on the node: deployment manager, node agent, or middleware agent.
- The **Version column** specifies the product version and product name of the WebSphere Application Server or WebSphere Extended Deployment installed node.
  - **Base**: WebSphere Application Server
  - **Express**: WebSphere Application Server Express
  - **ND**: WebSphere Application Server Network Deployment
  - **WXDOP**: WebSphere Extended Deployment Operations Optimization
  - **WXDCG**: WebSphere Extended Deployment Compute Grid
  - **WXDDG**: WebSphere Extended Deployment Data Grid
  - **XDA**: Middleware agent for WebSphere Extended Deployment
► The CPU usage column specifies the percentage of the CPU that is currently in use for the node, as reported by the node or middleware agent.

► The Stability column uses icons to specify the operational status of the node.
  
  – ☑️ **Stable** The node is operational, without issues.
  
  – ☭ **Questionable** The node has some problems, but is still operational. Check the status of the node agent to see if it is operational.
  
  – ⚠️ **Unstable** The node has severe issues. This state indicates a problem such as CPU performance issues, memory bottlenecks, or an inability to communicate with the deployment manager to provide server state information.
  
  – ☮️ **Unknown** The node stability cannot be reported. This state is typically reported when a node is out of commission.

► The Maintenance mode column specifies the maintenance mode status of the node.
  
  – **Started Maintenance mode:** The node is in maintenance mode and is no longer serving requests, so you can perform administrative actions on this node. While in maintenance mode, the node can be either started or stopped.

  When the node is started and in maintenance mode, the ODR does not route any traffic to the node.

  – **Normal mode:** The node is serving requests.

### 11.3 Reports

Reports are charts that show runtime data. With reports, you can view the performance of your virtualized environment. You can view statistics such as availability, response time, traffic, and throughput. You can use this data to monitor your environment, and to take correctional actions when necessary.

The reporting features available in WebSphere Extended Deployment V6.1 provide the capability to produce granular charting and reports at various scope levels of deployment targets, including cell, nodes, servers, and dynamic clusters. You can also include various data set types to be displayed. This allows you to view the health and performance data at various levels.

Multiple charts can be created in a tabbed fashion for structured viewing. This can be useful for various interest groups. For example, administrators within each LOB can monitor reports that contain the performance and usage data of a particular application deployed on a particular deployment target.
You can access the reports from multiple locations in the administrative console:

- **Runtime operations → Reports**
  This is the shortest path to the reports.
- **Servers → All servers → server_name → Reports**
  The Reports tab is not available for on demand router (ODR) servers.
- **Servers → Dynamic clusters → dynamic_cluster_name → Reports**
- **Servers → Clusters → cluster_name → Reports**
- **Applications → All applications → application_name → Reports**
- **Operational policies → Service policies → service_policy_name → Reports**

### 11.3.1 Charts

To benefit from a virtualized environment, you must know how your applications are performing. Customized charts can help you see if goals are being met. Dynamic charting provides a visual perspective of this performance. Statistics such as availability, response time, traffic, and throughput are supported. A wide range of options from which you can create various charts is provided.

A chart provides a graphical view of a situation so it can be monitored and can serve as a starting point in problem discovery.

Various color coded data metrics make it easier to correlate the data on a single chart. For example, a chart configured for a dynamic cluster as a data set type can show information such as average response time, average throughput, average wait time in queue, and can provide significant information about how the request is being handled by the dynamic cluster. This information coupled with the stability monitor can quickly provide you with initial information regarding the application performance.

You can organize your charts into chart groups. Using chart groups allows you to group multiple charts into a single view. This view is then saved and can be retrieved when needed.
11.3.2 Customized chart options

For each chart, you can select three options that determine the information contained in the chart.

- **Data set type**: Indicates the type of data that is being charted. You can choose from the following types:
  - On Demand Router
  - Proxy
  - Node group
  - Node
  - Dynamic cluster
  - Cluster
  - Generic server cluster
  - Application server
  - Service policy
  - Transaction class
  - Application
  - J2EE module
  - J2EE module instance
  - Database node
  - Database server

- **Data set**: Indicates the object that is providing the runtime data. After you select the data set type, the pull-down menu for this field is populated with the valid choices. For example, if you select On Demand Router as the data set type, this field lets you choose from the ODRs.

- **Data metric**: This option is based on an average calculation and is a predetermined set of values. The options you see are based on the data set type you select. Table 11-1 shows the possible data metric options.

<table>
<thead>
<tr>
<th>Data metric options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent requests</td>
<td>This metric measures the number of concurrent requests per the defined data set selection within the data name selection.</td>
</tr>
<tr>
<td>Average throughput</td>
<td>The average throughput metric is calculated in seconds based on the total workflow requests to the on demand router, as defined by the data scope, data name, and data set.</td>
</tr>
<tr>
<td>Average response times</td>
<td>The average response time metric is the time in milliseconds it takes for a work request to complete. This metric includes the request from inception, until the request is returned to the client. It is the equivalent of the service time metric plus the wait time in queue metric.</td>
</tr>
</tbody>
</table>
### Data metric options

<table>
<thead>
<tr>
<th>Data metric options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average wait times in queue (ms)</td>
<td>This metric is the average time in milliseconds that a work request spends waiting to service.</td>
</tr>
<tr>
<td>Average service times (ms)</td>
<td>The time in milliseconds to service a request for the on demand router. This metric indicates the time that the on demand router spends getting work to the node where it is completed.</td>
</tr>
<tr>
<td>Average queue length</td>
<td>This metric is the average time, in milliseconds, for any work queue in the environment that is defined by the data scope, data set, and data name.</td>
</tr>
<tr>
<td>Average drop rate</td>
<td>The average drop rate is calculated based on the number of work requests that are not processed due to a full queue. Work requests that encounter a full queue are returned to the requestor with an error.</td>
</tr>
<tr>
<td>Average relative performance</td>
<td>The relative performance of the configured goal compared to the actual goal value.</td>
</tr>
<tr>
<td>Percentile response time</td>
<td>The percentage of requests of the service policy percentile goal that meet the specified response time.</td>
</tr>
<tr>
<td>Concurrency requests</td>
<td>The number of seats desired for a given dynamic cluster's load coming through a given gateway.</td>
</tr>
<tr>
<td>Used memory</td>
<td>Memory used by a process.</td>
</tr>
<tr>
<td>CPU utilization</td>
<td>CPU utilization by a process.</td>
</tr>
<tr>
<td>Total requests</td>
<td>Total requests for Web modules deployed to a process.</td>
</tr>
<tr>
<td>Up time</td>
<td>Total time in milliseconds that the process was running.</td>
</tr>
<tr>
<td>Total EJB method calls</td>
<td>Total method calls for EJB modules deployed to a process.</td>
</tr>
<tr>
<td>Total transactions</td>
<td>Total transactions for a partition.</td>
</tr>
<tr>
<td>Free memory</td>
<td>Free memory on a node.</td>
</tr>
<tr>
<td>Utilization</td>
<td>The percentage of the resources that are being used and how much resource is available. Utilization measures the node speed and process CPU.</td>
</tr>
<tr>
<td>Jobs requested</td>
<td>The number of jobs that arrive at the execution environment (endpoint application) for processing.</td>
</tr>
<tr>
<td>Jobs completed</td>
<td>The number of jobs that run to completion at the execution environment.</td>
</tr>
<tr>
<td>Execution time</td>
<td>The total time in milliseconds that jobs spend executing.</td>
</tr>
<tr>
<td>Maximum concurrency</td>
<td>The maximum concurrency level that is attained.</td>
</tr>
</tbody>
</table>
11.3.3 Building charts and chart groups

To open the Reports view, navigate to Runtime Operations → Reports.

There are no chart groups when you first open the Reports view. An initial chart is displayed (Figure 11-7 on page 261). To begin customizing the reports for your use, you must build a series of charts and then store them into a chart group.
**Set the preferences**

Before getting started, look at the preferences, and alter them to suit your needs.

- You can change the preferences for new charts by expanding **Reports Preferences** at the top left of the window.
- To change the preferences for an active chart, click **Preferences** at the top right of the chart.
Figure 11-8  Report preferences

- Default chart group
  Specifies the chart group that displays on this page by default.

- Default chart type
  Specifies the default chart type when you create a new chart: line, bar, or pie.

- Default chart size
  Specifies the default chart size when you create a new chart: small, medium, or large.

- Display policy goals
  If enabled, displays the target value that you set for the data metric, which you can compare to the current values.

- Display data set shapes
  If enabled, the data set shapes are displayed for each data point.
Enable automatic refresh
If enabled, the charts automatically refresh at the specified chart refresh interval.

Chart refresh interval
Specifies a number of seconds between chart refreshes. If you have a lot of data in your charts, consider setting this value to a longer time.

Add a new chart and save it to a chart group
Use the following steps to build your first chart:

1. Click the Open new chart tab. A new tab opens with a blank chart.
2. Click Add data to specify the data set and metrics that you want to monitor.
   In this example, we build a chart that shows the average throughput for the ODR, xoddr1.
3. Select the following options for the chart in this order.
   a. Select the options for data set type.
   b. Select the options for data set.
   c. Select the data metrics.
   You can also filter the data sets for traffic that flows through a specific ODR. If you check this box, you will see a list of ODRs from which to select.
   Figure 11-9 on page 264 shows the options selected for this example.
4. Click **OK**.

5. Figure 11-10 on page 265 shows the results for the new chart in this example. You can use the **Change scope** button to modify the parameters for the chart. Note that you cannot change the name of a chart.
Note that the combination of chart properties you defined (data set type, data set, and data metric) appears as a line item near the bottom of the panel.

You can add more data to this same chart by clicking the **Add data** button and selecting a new combination of chart properties. Each new set appears as a separate line item. To remove a set of properties, check the select box for the item, and click **Remove**.

*Figure 11-10  Chart for ODR average response time*
Clicking the **Change scope** button lets you create a new set of properties, *replacing* all sets that you defined before.

6. Save the new chart into a chart group. Type the name in the field at the bottom of the panel, and click **Save**. In this example (Figure 11-10 on page 265), the chart group is named Test.

   You can continue to add charts to the group by using the **Open new chart** button, then saving the new chart to the group.

### 11.4 Visualization data service

The visualization data service logs historic data in text files for reuse with other charting programs. Historic data is logged in comma-separated values with time stamps in standard long value from the `java.util.Date` class. Using the visualization data service, you can log historical data, calculate chargeback values, or perform capacity planning.

IT departments face unprecedented pressure to operate as a business and to align IT costs with company priorities. This can be difficult to accomplish, especially when WebSphere Extended Deployment is used to consolidate IT assets. In such a complex environment there is a need to collect, analyze, and report the data that can be used in monitoring usage and to bill (charge back) departments based on usage information. Since every LOB is responsible for its IT costs, it becomes increasingly important to evaluate the usage and potential costs associated with the usage of shared IT resources.

The visualization data service provides usage data that can be mined to determine the chargeability per application or an enterprise resource. This is instrumental in planning, controlling, and budgeting for IT resources. For example, the log files generated by the visualization data service can be imported into a simple spreadsheet application or a well developed application such as IBM Tivoli Usage and Accounting Manager. The data from the logs can then be interpreted by the application to evaluate and determine usage of shared IT resources by application. Multi-variate and multi-dimensional analysis can be performed on the historical data to mine other usage patterns and request trends.

The visualization data service is configurable. You can specify the location of the visualization logs, maximum file size, and other parameters that affect the way the logs are recorded.

By default this logging is not enabled.
Use the following steps to enable the logging or to view and change the logging settings from the administrative console:

1. Click **System administration → Visualization data service** (Figure 11-11).

![Visualization Data Service settings](image)

**Figure 11-11 Visualization data service settings**

2. Select **Enable log** to start logging historic data.

3. [Version 6.1 and later] Enter a value in the Time stamp format field. The time stamp format specifies a time and date pattern that is used when logging the visualization data. Use the SimpleDateFormat Java class to format your time stamp. For example, to output the 12.06.2006 5:26:30:978 PM PDT time stamp, use the following time stamp format value:

   MM.dd.yyyy hh:mm:ss:SSS aaa z
If you are using IBM Tivoli Usage and Accounting Manager, then use a format that separates the date and time into different fields:

'MM.dd.yyy, hh:mm:ss:SSS'
'yyyy.MMMMM.dd, hh:mm:ss'

4. In the Maximum file size field, type a whole integer for the maximum file size for logs.

5. In the Maximum number of historical files field, type a whole integer for the maximum number of logs to be generated per historic cache type.

6. In the File name field, type the path where the log files are generated. You can use a variable in the file name value, for example:

${LOG_ROOT}/visualization

7. In the Data log write interval field, type a whole number between 1 and 365 for the interval in which the logs are generated in seconds, minutes, hours, or days.

8. From the Data transformer action list, select **Average** or **Skip** to specify how to transform data when the interval reaches its maximum value. More data points are provided than you might want to use. The AVERAGE option averages the existing data points between the specified interval, and the SKIP option skips the data points to only use the points specifically at the intervals.
Health management

This chapter describes the IBM WebSphere Extended Deployment V6.1 health monitoring capabilities, how they can be used to make your environment more reliable, and how they can help minimize potential downtime. There are several different health conditions your servers can be checked for. Automatic or supervised actions can be taken in the event of a breached health condition.

This chapter includes the following topics:

- Introduction to health management
- Health policies
- The health controller
- Defining health policies
- Monitoring runtime tasks
12.1 Introduction to health management

*Health management* is the ability of the system to take a policy-driven approach to monitoring the application server environment and taking action when certain predefined criteria are discovered. The health monitoring and management subsystem continuously monitors the operation of servers to detect functional degradation that is related to user application malfunctions.

The health management subsystem consists of two main elements: the *health controller* and *health policies*. The health controller runs on one of the nodes in your environment. Health policies define the health conditions that you want to monitor for and the health actions to take if these conditions occur.

You can disable or enable health management. If your health policy includes server restarts as an action, you can apply limits to the frequency of server restarts or prohibit restarts during certain periods, as dictated by your business requirements.

The monitoring subsystem introduces a small performance impact. If you enable health monitoring to manage existing problems, a shorter data collection interval is appropriate. If you enable health management features in a stable environment, we recommend that you maximize the data collection interval.

*New in V6.1:* You can define custom conditions and actions for your health policy when the predefined options do not suit your needs.

Implementing health management involves the following:

1. Identify the conditions you want to monitor each target for and the actions you want to take. This includes understanding whether the predefined conditions are sufficient or whether you need to create a custom condition.

2. Determine the type of reaction mode to use. It is a good idea to start with the supervised mode (requires operator intervention for the action to take place) when you first define a health policy. After you feel certain the health condition is being monitored at the appropriate level and the correction action should always be taken, you can switch to the automatic reaction mode.

3. Define the health policy.

4. Enable e-mail notifications.

5. Configure the health controller.

6. Ensure health monitoring is enabled. Health monitoring is enabled by default.
12.1.1 Health monitoring tips

Following are a few tips regarding health monitoring:

- Health monitoring can help to make your environment more reliable, but like any automation feature, it should be used carefully. Health policies should only be defined and assigned to servers if you think that a particular health policy is actually needed.

- You should think about your peak load times and whether a health policy triggered action causes more problems than it solves during this time. Remember that you can define prohibited restart times.

- When rolling out a brand new application or a new version of an application, you never know what to expect in production. Unexpected bugs might crop up regardless of what happened during testing. For some shakeout period we recommend setting up the following health policies for all new or updated applications:
  - Memory leak condition
  - Excessive request timeout condition

  The policy action should start off in supervised mode until you learn whether there is a problem. After you know this, you can either turn the policies off or move to automatic mode.

- If your application causes a health condition breach, then try to fix the problem within the application. Health monitoring provides a work-around to make your system more reliable, but it should not be the final solution. Using health monitoring should help you to keep your environment stable until your application fixes are available.

- Health policy and monitoring should only be used as a tool to enable and exploit the continuous availability capability of WebSphere Extended Deployment. This important feature is not meant to replace the testing and benchmarking phases of the application development lifecycle. We therefore recommend that every application prior to being deployed an in Extended
Deployment environment, should be tested and benchmarked for performance.

The following articles on troubleshooting and monitoring might be of interest to you:

- *Troubleshooting health management*
  

- *Monitoring and tuning health management*
  

### 12.2 Health policies

A health policy defines a set of conditions that are interpreted by WebSphere Extended Deployment as a degradation of server function. Health policies are a combination of a health condition to monitor, actions to take if the condition occurs, the deployment target to be monitored, and a reaction mode that defines whether the action occurs automatically or with operator intervention. You can assign policies to members on any deployment target including the cell, dynamic clusters, static clusters, application servers, and nodes.

**Note:** Custom health policies must be defined using scripting. See section 12.4.2, “Defining custom health policies” on page 284.

#### 12.2.1 Health conditions

*Health conditions* define the variables that you want to monitor in your environment. Several categories of health conditions are predefined. V6.1 introduces the concept of a custom health policy that allows you to define a custom condition when the predefined health conditions do not fit your needs.

All of the available conditions support server restarts as a reaction. The age and workload conditions are preventative policies, and the other conditions are detection-based policies.

The predefined health conditions are as follows:

- **Age-based:** Triggers when members associated with this policy reach a certain age value. You can use the age-based condition on all server types.
► **Excessive request timeout**: Triggers if requests that are directed to an associated member time out and the percentage of timeouts exceed the specified value. This condition supports thread dumps in addition to the server restart reactions.

The excessive request timeout condition does not apply to JMS and IIOP traffic.

You can use the excessive request timeout condition on all server types.

► **Excessive response time**: Triggers if the members that are associated with this detection-based policy have an average response time for requests that exceed a certain amount of time. You can use the excessive response time condition on all server types.

► **Excessive memory usage**: Triggers if the members that are associated with this detection-based policy have a memory usage that exceeds a percentage of the maximum heap size for a certain amount of time. The excessive memory usage condition is supported on servers that are running WebSphere Application Server or WebSphere Extended Deployment only.

► **Memory leak**: Looks for consistent downward trends in free memory available to a server in the Java heap. The detection level setting determines when these trends are detected. The slower detection level setting requires the most historical data. The normal and faster detection level settings require the same amount of historical data, but the faster setting allows analysis before the Java heap expands to its maximum configured size. This provides earlier detection capability, but it is also more prone to false positives. This condition supports heap dumps in addition to server restarts as reactions.

The memory leak condition is supported on servers that run WebSphere Application Server or WebSphere Extended Deployment only.

► **Storm drain**: Detects situations where requests are shifted toward a faulty cluster member that advertises low response times. Storm drain relies on change point detection on a given time series data. To detect change points, the health controller calculates a left mean and a right mean for a given point. For a point, the left mean consists of the mean value of N samples that arrive prior to this sample, and the right mean is the mean value of N samples, including the current point, that arrive later. The difference of the left and the right mean values is stored and compared with other differences in a set of values to N to determine if this difference is a local maxima. If this difference is the maximum difference, then the point to which this difference corresponds is declared as a change point. The two metrics that are used for detecting storm drain are the response times and dynamic workload manager weights that are observed for the server.
The **Faster Detection, Higher Probability of False Alarms** policy uses fewer samples (N=10) for both response times and dynamic workload manager weights and detects a change point in each of the metrics based on the sample set. As a result, this policy reaches a conclusion faster because it waits for 20 samples: 10 for the left mean and 10 for the right mean, for calculating a difference of means and looking for a local maxima. The samples are collected at intervals of 15 seconds. Therefore, the storm drain can be detected within five minutes of its occurrence. However, because the samples are fewer, if the samples have multiple transient peaks or dips, then there is a higher probability for false alarms.

The **Slower Detection, Lower Probability of False Alarms** policy uses more samples (N=15) for both response times and dynamic workload manager weights. As a result, this policy reaches a conclusion slower because the policy has to wait for 30 samples: 15 for the left mean and 15 for the right mean for calculating a difference of means. The detection time is seven minutes and 30 seconds. However, because there are more samples, the presence of samples with transient peaks or dips does not overly affect the mean values. Therefore the probability of false alarms is lower.

**Restriction:** The storm drain condition does not apply to JMS and IIOP traffic.

You can use the storm drain condition on all server types.

- **Workload:** Triggers if the members associated with this policy have served a user-defined number of requests. You can use the workload condition on all server types.

### 12.2.2 Actions to take

Depending on the policy type, there are different actions that are performed if a policy breach is detected. Following are the possible actions:

- **Restarting the application server.**
  This action is available for all policy types. When a server is a member of a dynamic cluster, another instance of the dynamic cluster is started to serve user requests before the server that has triggered the policy breach is shutdown. This allows WebSphere Extended Deployment to handle potential issues with the least amount of impact to its consumers.

- **Taking a thread dump.** (javacore)
  This action is only available for the excessive request timeout condition. The option to take thread dumps is only supported for application servers running in IBM JVMs.
Taking JVM heap dumps on IBM Java Development Kit (JDK™).

This reaction is only available for the memory leak policy type. This option also works for IBM JVMs only.

**New in V6.1:** Put server into maintenance mode

Maintenance mode is used to perform diagnostics, maintenance, or tuning on a node or server without disrupting incoming traffic. Putting a server into maintenance mode allows the remaining requests on the server to be processed.

Any requests that have an open session on the server are routed to the server until the session ends or times out. After all requests are completed, the server is moved to maintenance mode. Any new requests are routed to servers that are not in maintenance mode.

**New in V6.1:** Put server into maintenance mode, and break HTTP and SIP request affinity to the server.

The HTTP and SIP session affinity is broken, and the session is moved to another server running in normal mode.
**New in V6.1:** Take server out of maintenance mode.

After the server reaches a healthy state, it can be reinstated to serve requests.

For example, if a server exceeds a memory threshold, putting it in maintenance mode gives it a chance to recover via garbage collection while no new requests were being sent to it. After heap utilization is back to below the threshold, you can then have the server taken out of maintenance mode.

**Custom action.**

With a custom action, you define an executable file to run when the health condition breaches. A Custom action must be created before you can use it in a health policy.

### 12.2.3 Reaction modes

After selecting the policy type and the action that should be performed in the event of a health policy breach, you must specify the reaction mode. The reaction mode defines the level of user-interaction when the health condition determines corrective action is needed. There are two possible reaction modes:

**Supervised**

The supervised reaction mode is the mode where health policies are active and recommendations on actions are sent to the administrator. If the administrator follows the recommendations, the only action required is selecting a button, and the actions are performed. This option is widely preferred by the administrators who are not yet comfortable with giving the WebSphere Extended Deployment total control in performing autonomic actions.

**Automatic**

The automatic reaction mode means that health policies are active, logging data, and the defined reaction is taken automatically.

### 12.3 The health controller

To use health monitoring you must make sure that the health controller is enabled. It is enabled by default. Select **Operational Policies → Autonomic Managers → Health Controller** from the administrative console. The configuration panel is shown in Figure 12-1 on page 277.
On this panel you can also configure the behavior of your health controller:

- You can set the **control cycle length** in minutes (1-60), which is the time between two consecutive health checks to determine if health policy conditions were breached. Longer control cycle lengths reduce the load caused by health monitoring. The disadvantage of long cycles is that they
take longer to detect a policy breach that occurs between the two cycles. You
may vary here depending on the policies you defined. There is a trade-off
between reduced load, due to longer control cycle, and time taken to detect
breach in health policy.

► In the Maximum Consecutive Restarts field you can enter the number of
restart attempts to be performed until an instance restart is declared as failed.

► The Restart Timeout field specifies the time in minutes that a restart can take
until it is a failure. The restart means the sequence of stopping and starting a
server. We recommend that the heuristics of the administration team be
applied from past experience in determining this value. Depending on the
environment the server restart time may vary.

► The Minimum Restart Interval parameter controls the minimum amount of
time that must pass between consecutive restarts of an application server
instance. If a health condition for an application server is breached during this
time, the restart is set to a pending state. The restart occurs when the
minimum restart interval passes. The value can range from 15 minutes to
365 days, inclusive. A value of zero (0) disables the minimum restart value.

► Under Prohibited Restart Times you can define one or more time frames
where no restart is allowed. This can be used to prevent servers from being
restarted for example during your highest load times or when special
functions are used that only run on an single machine. If a restart event is
triggered during such a time frame, it is postponed until after the end of the
time frame. You must define the start and end time for every single row, and
select at least one day of the week. You cannot define time frames over
midnight. In this case you must define two separate time frames: one until
midnight and one starting at midnight.

You can also define health controller settings using a script. It is called
HmmControllerProcs.jacl (located in the /bin directory) and allows you to do the
following:

► Enable or disable the health management controller.
► Enable or disable automated restarts for the server.
► Set or get values for other health management settings.

See the following Information Center article for information about using this
script:

► Changing health management controller configurations with scripting

websphere.ops.doc/info/reference/todhmmscripts.html
12.3.1 Locating the health controller

After you define and enable the health controller, it runs as part of the cell. There is one controller per cell. The controller is HA managed and can run in the deployment manager or a node agent. You can use the runtime topology to learn which node hosts the health controller. Select Runtime Operations → Extended Deployment → Core Components.

In Figure 12-2 the health controller is on the XDnode3 node agent.

<table>
<thead>
<tr>
<th>Name</th>
<th>Scope</th>
<th>Stability</th>
<th>Current location</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARFMCcontroller</td>
<td>XDcell1</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>Application Placement Controller</td>
<td>XDcell1</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>Async PMI Bridge</td>
<td>XDcell1</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>XDCluster1 (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>XDcluster2 (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>deletemeCluster (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>WASCE_DC (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>JBoss_DC (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>PHP_DC (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>Tomcat_DC (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>Tomcat_Platinum_DC (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>DWLM Controller</td>
<td>tomcat_platinum_dc (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
<tr>
<td>Health Controller</td>
<td>gcluster1 (XDcell1)</td>
<td></td>
<td>XDcell1/XDnode3/nodeagent</td>
</tr>
</tbody>
</table>

![Core Component tab showing health controller]

Locating the health controller can also be done by a wsadmin jacl script. The checkHmmLocation.jacl script is located in the app_server_root/bin directory. Example 12-1 shows the output of this script:

Example 12-1  wsadmin script to locate the health controller

```
/usr/IBM/WebSphere/AppServer/bin # ./wsadmin.sh -f checkHmmLocation.jacl

WASX7209I: Connected to process "dmgr" on node dmgr1 using SOAP connector; The
type of process is: DeploymentManager
looking for group name:
HAMItemGroup:com.ibm.ws.xd.hmm.controller.HmmControllerI
mpl_com.ibm.ws.xd.hmm.controller.HmmController_Default

Results:
{SERVER_NAME nodeagent}
{MEMBER_NAME
{NODE_NAME node3}
{GROUP_ID
/usr/IBM/WebSphere/AppServer/bin #

In Example 12-1 on page 279 the health controller is now located on node3. You can see that the health controller can move through the entire cell. Its location changing automatically. Note that health controller is run as a singleton service and is managed by the high availability manager.

12.4 Defining health policies

A health policy is the definition of specific health criteria that you want your WebSphere Extended Deployment to protect itself against. The health management function uses the defined policy to search the environment for software malfunctions.

A health policy consists of the following:

- A condition to monitor (for example, excessive memory use)
- An action to take when the condition is detected (for example, restart the server)
- Targets to monitor (for example, specific servers, nodes, dynamic clusters)
- A reaction mode (actions are taken automatically or with operator intervention)

You have the following options for creating health policies:

- Create a health policy using predefined conditions and actions (provided by WebSphere Extended Deployment) using scripting or the administrative console.
- Define a custom action using scripting or the administrative console, and then create a health policy that uses the custom action. The action can be a
Java-based action or non-Java based (an executable valid for the operating system).

- Define a custom health policy using scripting. This allows you to define an expression defining the condition for which to monitor.

12.4.1 Creating custom actions

You can create custom actions that define corrective tasks to perform when a health condition breaches. A custom action can be a Java or non-Java executable file. Custom actions can run on the deployment manager, a node that is hosting servers that breach health conditions, the node where the health management controller is running, or a node that you specify.

Create a custom action before you create a health policy. When you define the health policy, select the custom action as part of the action plan that runs when the health condition breaches. You can also define a custom action when you are creating an action plan in the health policy creation wizard.

Use the following steps to create a custom action:

1. From the administrative console, select **Operational Policies → Custom Action**.

2. Click **New** to define a new custom health action. You have two options from which to select from the Select Custom Action Type pull-down menu: Non-Java action or Java action. Read further to receive a description of both options.

![Health Management Custom Action](Image)

*Figure 12-3  Select custom action type*
– A non-Java action

Non-Java actions can include shell scripts on UNIX platforms or batch files (.bat) on Windows. It can also include any binaries that perform a certain function against the application server environment.

– A Java action

Java actions can include executable JAR or Java class files.

Be sure to test these actions thoroughly prior to using them in a custom health action.

Select the option type, and click Next.

3. The next panel, shown in Figure 12-4 on page 283, varies depending on the option you select. It provides fields for defining the custom action.

In this example, the custom action consists of a shell script that restarts the server when the health condition is breached. This may be drastic in some cases, but this example is provided to demonstrate a feature.
The next step asks you to confirm your settings. Click Finish.

5. Save the changes to the master repository.

The new action is now available for use in a health policy.

Custom actions can also be created using the `customHealthAction` AdminTask command. For information about using this command, see the following article:

The ability to create a custom health policy is new in V6.1. This was mainly introduced to address the health monitoring needs of the growing number of supported middleware servers. Using a custom health policy enables you to create expressions, defining what “unhealthy” means in your environment, rather than having Extended Deployment define it.

For example, because excessive memory consumption (which often indicates a memory leak) is not a predefined health condition supported for WebSphere Application Server Community Edition, you have to write a custom health condition in order to monitor this.

Custom conditions are supported on all platforms, but with varying levels of support. For example, metrics from an ODR can be used for a health policy for all server types. However, a custom health condition for a non-WebSphere server cannot make use of the WebSphere PMI server metrics.

While health policies using predefined health conditions can be configured from the administrative console, a health policy using custom conditions must be defined using jython scripts or interactively using the `createHealthPolicy` AdminTask command.

For information about using the `createHealthPolicy` AdminTask command, see the following article:

- **Health policy administrative tasks**
  

**Custom condition expression**

Custom conditions are created from a set of operands. These operands, their use, their format, and examples are discussed in the following Information Center article:

- **Custom health condition expressions**
  
The following is a summary of the operands you can use to build a custom condition:

- **PMIMetric** \_FromServerStart
  
  You can use metrics for the following set of PMI modules. All metrics that are in these modules are supported:
  
  - beanModule
  - webAppModule
  - systemModule
  - jvmRuntimeModule
  - connectionPoolModule (Java Database Connectivity, or JDBC™)
  - threadPoolModule
  - xdProcessModule (for other servers)

- **PMIIntervalMetric** \_FromLastInterval
  
  You can use the same set of modules as the PMIMetric\_FromServerStart operand. The PMIIntervalMetric\_FromLastInterval operand uses an average of the reported values in the last interval. The interval is the length of the health controller cycle.

- **ODRServerMetric** \_FromServerStart
  
  You can use a subset of server-level metrics that the on demand router (ODR) publishes. These metrics are cumulative and reported when the server starts.
  
  - departs
  - waitTime
  - responseTime
  - serviceTime
  - serviced
  - timeouts
  - errors
  - currentlyExecuting

- **ODRServerMetric** \_FromLastInterval
  
  You can use the same set of metrics as the ODRServerMetric\_FromServerStart operand. The ODRServerMetric\_FromLastInterval operand uses an average of the reported values in the last interval. The interval is the length of the health controller cycle.

- **ODRCellMetric** \_FromServerStart
  
  You can use a subset of cell-level metrics that the ODR publishes. These metrics are cumulative and reported since the server starts.
  
  - currentQueueLength
You can use the same set of metrics as the ODRCellMetric_FromServerStart operand. The ODRCellMetric_FromLastInterval operand uses an average of the reported values in the last interval. The interval is the length of the health controller cycle.

For managed bean (MBean) operands, you must specify the ObjectName query string, the method name, and the return type.

These metrics are used for querying an attribute of an MBean rather than invoking a method on the MBean.

With this operand, you can ping any relative path, or Uniform Resource Identifier (URI), on the server that is the target of this policy. The return value is used in the condition expression for the custom health policy.

Example of a custom health policy

Note: Custom policies are not visible from the administrative console. To see all health policies, including custom policies, use the AdminTask listHealthPolicies command.

To use create a custom health policy, perform the following steps:

1. Open a a shell prompt on Unix or a command prompt in a Windows environment, and invoke wsadmin.

   wsadmin -lang jython
2. Invoke the AdminTask `createHealthPolicy` command in the interactive mode.

   wsadmin > AdminTask.createHealthPolicy (’[-interactive]’)

3. Respond to the prompts to build the health policy.

4. Save the configuration.

5. Use the AdminTask `listHealthPolicies` command to make sure the policy is created.

Example 12-2 illustrates the use of the AdminTask `createHealthPolicy` command to create a new health policy. The text in bold is the text entered by the administrator. It creates a health policy that defines the condition `ODRServerMetric_FromServerStart$errors > 100L`. This condition is met when the number of requests returned from the server with an error indicator during the reported interval exceeds 100.

```
Example 12-2   Creating a custom health policy

wsadmin>AdminTask.createHealthPolicy (’[-interactive]’)
Create Health Policy

Command to create a health policy

*Health Policy Name (name): TestCustomPolicy
Health Policy Description (description): Restart node on ODR metric
Reaction Mode (reactionMode): SUPERVISE

Command to create a health policy

-> *1. Add Member Step (addMember)
   *2. Add Condition Step (addCondition)
   *3. Add Action Step (addAction)

S (Select)
N (Next)
C (Cancel)
H (Help)

Select [S, N, C, H]: [S] S

Add Member Step (addMember)

   | Member Type (type), Member Name (name), Member Node Name (node)
```
A (Add Row)
F (Finish)
H (Help)

Select [A, F, H]: [A] A
*Member Type (type): server
*Member Name (name): XDCluster1_XDnode1
Member Node Name (node): XDnode1

Add Member Step (addMember)

<table>
<thead>
<tr>
<th>Member Type (type), Member Name (name), Member Node Name (node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>server, XDCluster1_XDnode1, XDnode1</td>
</tr>
</tbody>
</table>

Member Type, Member Name, Member Node Name must be provided to specific a row in batch mode.

P (Previous)
A (Add Row Before)
F (Finish)
H (Help)

Select [P, A, F, H]: [F] F
Create Health Policy

Command to create a health policy

1. Add Member Step (addMember)
   -> *2. Add Condition Step (addCondition)
      *3. Add Action Step (addAction)

S (Select Row)
N (Next)
P (Previous)
C (Cancel)
H (Help)

Select [S, N, P, C, H]: [S] S
Add Condition Step (addCondition)

*Condition Type (type):
Condition Expression (expression):
Condition Parameters (params):

Select [C (Cancel), E (Edit)]: [E] E
*Condition Type (type): CUSTOM
Condition Expression (expression): ODRServerMetric_FromServerStart$errors > 100L
Condition Parameters (params): <enter>
Create Health Policy

Command to create a health policy

1. Add Member Step (addMember)
2. Add Condition Step (addCondition)
-> *3. Add Action Step (addAction)

S (Select Row)
P (Previous)
C (Cancel)
H (Help)

Select [S, P, C, H]: [S] S

Add Action Step (addAction)

<table>
<thead>
<tr>
<th>Action Type (type), Step Number (stepNum), Action Name (name), Action Node Name (node), Action Server Name (server)</th>
</tr>
</thead>
</table>

A (Add Row)
F (Finish)
H (Help)

Select [A, F, H]: [A] A
*Action Type (type): CUSTOM
*Step Number (stepNum): 1
Action Name (name): TestCustomAction
Action Node Name (node): XDnode1
Action Server Name (server): XDCluster1_XDnode1
Add Action Step (addAction)

<table>
<thead>
<tr>
<th>Action Type (type), Step Number (stepNum), Action Name (name), Action Node Name (node), Action Server Name (server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOM, 1, TestCustomAction, XDnode1, XDCluster1_XDnode1</td>
</tr>
</tbody>
</table>

Action Type, Step Number, Action Name, Action Node Name, Action Server Name must be provided to specific a row in batch mode.

P (Previous)
A (Add Row Before)
F (Finish)
H (Help)

Select [P, A, F, H]: [F] F
Create Health Policy

Command to create a health policy

1. Add Member Step (addMember)
2. Add Condition Step (addCondition)
3. Add Action Step (addAction)

-> End of task

P (Previous)
F (Finish)
C (Cancel)
H (Help)


WASX7278I: Generated command line: AdminTask.createHealthPolicy('[-name TestCustomPolicy -description "Restart node on ODR metric" -reactionMode SUPERVISE -addMember [[server XDCluster1_XDnode1 XDnode1]] -addCondition [-type CUSTOM -expression "ODRServerMetric_FromServerStart$errors > 100L"] -addAction [[CUSTOM 1 TestCustomAction XDnode1 XDCluster1_XDnode1]]]')
12.4.3 Defining health policies using predefined conditions

This section provides an overview of creating a new health policy using the administrative console. This process assumes that the policy will use predefined conditions.

**Note:** If you want to use scripting to create a new policy, or if you want to create a custom health policy, see the following Information Center article for information about using the createHealthPolicy AdminTask command:

- **Health policy administrative tasks**
  

Use the following steps to define a health policy using the administrative console:

1. Navigate to **Operational Policies → Health Policies**.
2. Click **New**.

3. The first panel (Figure 12-6 on page 293) allows you to name the health policy and to select the policy condition type. More information about specific condition types are discussed in later sections.
4. Define the health policy conditions.
   a. Define the health condition properties. The options are tailored to the health condition you selected on the previous panel.
   b. Select the reaction mode (supervise or automatic) from the pull-down menu.
   c. Select one or more steps to take if the condition occurs.

Click **Next**.
You can define one or more actions to be taken, in a specific order, when the condition occurs. Select actions from the list of predefined actions, or you can select a custom action (this must have been created before you begin the process).

A default action is shown in the panel. You can leave this selection as is, add additional actions, or delete the default action, and replace it with new action(s).

To perform an action on a step shown in the table, click the box under Select, and then click the appropriate button.
If you define multiple steps, they can be ordered using the **Move up** and **Move down** buttons.

5. In this example, we want to place the server in maintenance mode. Select the predefined action, and click **Delete step**.

6. Click **Add step**. Select action type and action.

![Figure 12-8   Add a new step](image)

7. Click **Next**, and then click **Finish** to complete the step selection.
8. Click **Next**.

9. The next panel, Figure 12-10 on page 297, allows you to select the targets to monitor. The type of target available depends on the health condition you selected. Use the **Add >>** button to move targets for monitoring to the Members list on the right.
You can add targets of more than one type. For instance, you can select Servers/Nodes as the member type and add one or more servers to the list. Then change the member type to Dynamic clusters and add one or more dynamic clusters.

Click **Next**.

10. The last panel summarizes your selections. Click **Finish** to create the new policy.

**Age-based condition**

A health policy that uses an age-based condition monitors for servers that have run longer than a configured time period. This is used to refresh the servers periodically, for example to prevent them from crashing in case of heap fragmentation.

Defining an age-based conditioned health policy is very simple. You set the amount of time in hours or days, and select if an event causes an automatic or supervised action. See Figure 12-11 on page 298.

**Note:** The **Maximum age** field does not accept fractional numbers, so you must select hours if you want to specify half days. Acceptable values are 1 - 365.

The only possible action for an age-based condition is a server restart.
Excessive request timeout condition
This condition detects, for each server that is a member of this policy, the percentage of timed out requests routed to this server by the ODR. You configure the percentage of timed out requests to breach this condition, and choose if the action is taken automatically or supervised.

There are two actions available for the excessive request timeout condition, and you can choose one or both of them. The possible actions are Restart server and Take thread dumps, as shown in Figure 12-12 on page 299.
Important: The option to take thread dumps is not supported for application servers running non-IBM JVMs. A warning is displayed when you assign such servers to this policy.

Excessive response time condition
With the excessive response time condition, you can create a policy that discovers hung servers. You define an average response time value from one millisecond to one hour. When this value is exceeded by a server to which this
policy is assigned, a supervised or automatic server restart is triggered. See Figure 12-13.

**Note:** Be careful with the automatic reaction mode. You could have a problem with external resources rather than with the servers themselves. For example, the database is not available, which causes high response times for all of your servers.

If you do not want your servers to be restarted automatically in such an event, then do not select the automatic reaction mode.

---

**Memory condition: excessive memory usage**

This condition creates policies that monitor each defined server for memory usage in excess of a specified percentage of that server's JVM heap size for a certain amount of time.
As you can see in Figure 12-14, you must define the percentage of heap size usage that must be reached and how long the usage must stay on or above the given value to trigger the reaction. The only possible action is an automatic or supervised server restart.

The time period can be set from one second to 60 minutes to ensure that the policy only triggers when a server's memory usage is in excess of the threshold for a certain period of time.

**Tip:** If you have too many events triggered by such a policy, you should think about changing your servers’ heap size.
Memory condition: memory leak
The memory leak algorithm, which this policy monitors the servers for, profiles the JVM heap size after a garbage collection occurs and looks for trends of increased consumption.

As you can see in Figure 12-15, there are three different detection levels that cause faster detection or lower probability of false alarms. The slower detection needs more historical data than the other two. The standard detection and the faster detection use the same amount of historical data, but the faster detection can start the analysis even before the JVM has increased its heap to the maximum allowed heap size.

Possible actions when this health condition is breached are to restart the server and to take JVM heap dumps on IBM Java Developer Kit (JDK) only. You can select both or only one of these actions. This gives you the freedom to define two different health policies with the same detection level, but with different reactions. For example, you can create one policy to take JVM heap dumps automatically and another one that restarts the server in supervised mode.

Figure 12-15 Memory condition: memory leak properties
Storm drain condition

The storm drain condition can be assigned only to dynamic clusters; therefore, you can assign it only on the cell and dynamic cluster level. This condition detects for each cluster member a significant drop in the average response time coupled with changes to the dynamic workload manager weights for the cluster member. This helps to prevent the system from shifting all the load to a faulty server with low response time.

The configuration steps are limited to choose only between Standard detection, standard probability of false alarms and Slow detection, lower probability of false alarms. Also you can select between an automatic or supervised server restart. See Figure 12-16.

The detection levels work with different amounts of samples used to calculate the left and right mean of response times and dynamic workload manager weights. The standard detection works with 20 samples, the slower detection with 30 samples. The samples are taken in a 15 second interval; therefore, it takes
five minutes for the standard detection and seven minutes and 30 seconds for the slower detection to trigger the reaction.

As an example for the standard detection, the left mean is built by the 10 samples that arrived before the current sample, and the right mean is built by the 10 samples that arrive later, including itself. The difference between the left and the right mean is stored and compared with other differences. When an unusual change is detected, this policy raises a health event.

**Workload condition**

Policies with workload conditions monitor servers for the total amount of requests that a server has already handled. You can configure the policy to restart a server automatically or in supervised mode if the given number of requests was handled by a server. This policy can be particularly instrumental when there is a known condition that was not resolved. For example, Application A is known to cause performance degradation after five days of running continuously. A workload condition can be applied to the deployment targets of Application A to enable a planned outage and graceful recovery. The configuration panel is shown in Figure 12-17.

![Figure 12-17 Workload condition properties](image)

"Define health policy health condition properties"

- **Health condition properties**
  - Total requests: 20000000

- **Health management monitor reaction**
  - Reaction mode: Supervise

- **Take the following actions when the health condition breaches**
  - Add step: Restart server, Target server: Sick server, Target node: Node hosting sick server
  - Add step: Take thread dumps, Target server: Sick server, Target node: Node hosting sick server
12.4.4 Default health policies

In WebSphere Extended Deployment Version 6.1, five predefined health policies are installed with the product. These default health policies are configured for cell-level and use the supervised reaction mode. You can modify the default policies to suit your environment, or delete them if they are not needed.

Because the default health policies monitor all the servers in supervised mode, these policies can be used as a first line of defense against health problems. You can define policies with more stringent settings or automated mode operation for particular servers or collections of servers in addition to the default policies. The following default health policies are created at installation:

- Default memory leak: Default standard detection level.
- Default excessive memory: Set to 95% of the JVM heap size for 15 minutes.
- Default request timeout: Set for 5% of the requests timing out.
- Default excessive response time: Set to 120 seconds.
- Default storm drain: Default standard detection level.

12.5 Monitoring runtime tasks

After you define your health policies and enable the health controller you can monitor the results in the administrative console by navigating to System Administration → Task Management → Runtime Tasks.

In Figure 12-18 on page 306 you can see that tasks were performed automatically or are waiting for administrator intervention, depending on the configuration of the health policies.
When a health policy breach is detected by the health controller, an automatic or supervised task is created. The task steps are not only based on the health policy, there are also steps inserted based on decisions by other WebSphere Extended Deployment components. To see which steps are planned for a task, you can click the text under the Task Explanation column.

If, for example, a health policy breach happens in a dynamic cluster, then the restart of the server is not a real restart. The steps can look similar to the following.

Step 1: Start server RedbookCluster10_node1_1 on node node1.
Step 2: Stop server RedbookCluster10_node2 on node node2.

You can see that the start and stop action is planned on different nodes, and it is based on the dynamic cluster settings. The different components act together to decide how a server restart must be managed. The environment ensures that no other rule, such as the minimum amount of running servers per cluster, is broken during a server restart. In this example, this is achieved by starting a new server before shutting down another one. The shut down server is not restarted until other events trigger a server start. This could be, for example, more workload or other health policy events that cause this server to take the place of another now unhealthy server.
12.5.1 Configuring e-mail notification for tasks

Event notification by e-mail can be used to prevent the administrator from having to monitor administrative console for runtime tasks. When the notification e-mail is enabled, the administrator is informed about all tasks generated by the health or the placement controller.

Use the following steps to set up the notification:

1. Select System administration → Task Management → Notifications.

![General Properties and E-mail Addresses](image)

- In the SMTP Host Name field, type the simple mail transfer protocol (SMTP) server to connect to when sending mail. Validation is not performed on the server host name.
- In the Port number field, type the SMTP port number to connect to when sending mail. Valid port numbers are between one and 64767. Validation that the port is the correct SMTP port is not performed.
- In the SMTP user ID field, type your user ID if your mail transport host requires authentication. Leave this field blank if your mail server does not require authentication.
– In the SMTP password field, type your password if your mail transport host requires authentication. Leave this field blank if your mail server does not require authentication.

– Select **Enable notifications** to use e-mail notifications. When notifications are enabled and a task is generated, an e-mail notification is sent to each of the e-mail addresses specified.

– In the E-mail address field, type the address for notification, and click **Add**. Each e-mail address is validated for syntax. You can enter multiple addresses to this list. To take an existing address off the list, select the address, and click **Remove**.

2. Click **OK**, and save your changes.
Roadmap to implementation

This chapter provides a summary of the steps required to create a new WebSphere Extended Deployment environment. This roadmap assumes that you are setting up a very basic “starter” environment and that you have planned your topology.
Creating the basic environment

The first step is to set up the basic WebSphere environment, including the deployment manager, On Demand Router (ODR), and Web servers.

Install the WebSphere software

1. Every WebSphere system: Install WebSphere Application Server Network Deployment V6.1 and the required fixpacks and interim fixes on every system that will host a WebSphere node, including the systems that will host the deployment manager, ODRs, and application servers.

A list of the required fixpacks and interim fixes can be found at the following Web location:

http://www-1.ibm.com/support/docview.wss?rs=3023&uid=swg27009516

Select the appropriate operating system.

2. Every WebSphere system: Install WebSphere Extended Deployment Operations Optimization V6.1 on each of these systems. Install all available fixpacks and interim fixes.

Create the deployment manager

1. Deployment manager system: Create the deployment manager profile. Start the deployment manager, and logon to the administrative console.

The steps required to perform these tasks are discussed in section 5.1, “Creating the deployment manager” on page 64.

Web servers

On each Web server system, perform the following tasks:

1. Install a supported Web server.

2. Install the Web server plug-in. The Web server plug-ins are included with WebSphere Application Server Network Deployment.

3. Define the Web servers to the WebSphere cell.

Information on these steps can be found in the WebSphere Application Server Information Center at the following Web site:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp
Create the on demand routers

*On each ODR system, perform the following steps:*

1. Create a custom profile and federate it to the cell.
2. Start the node agent.
   
The steps required to create the profile and start the node are discussed in section 5.3, “Adding WebSphere nodes to the cell” on page 83.

*Using the WebSphere administrative console, perform the following steps:*

3. Create an On Demand Router server on each node.
   
The steps required to create an ODR are discussed in section 7.2, “Creating an on demand router” on page 117.

4. Configure the proxy plug-in generation, and disable the automatic generation and propagation of the Web server plug-in configuration file.
   
The steps required to perform this task are discussed in section 7.4, “Proxy plug-in generation” on page 122.

Integrating WebSphere application servers

For every system that will host an ODR or WebSphere application server, perform the following tasks:

1. Create a custom profile and federate it to the cell.
   
The steps required to perform this task are discussed in section 5.3.1, “Creating and federating the nodes” on page 83.

2. Start the node agent.
   
The steps required to perform this task are discussed in section 5.3.2, “Starting the node” on page 84.

*Using the WebSphere administrative console, perform the following steps:*

3. Create one or more dynamic clusters. Creating a dynamic cluster automatically creates the servers.
   
The steps required to perform this task are discussed in section 8.3, “Dynamic clusters with WebSphere application servers” on page 158.

4. Deploy applications to the dynamic clusters.
   
The steps required to perform this task are discussed in section 9.1, “Installing WebSphere applications” on page 176.
5. Define service policies and associate them with the applications.
   – Create a service policy and transaction class. See section 10.3.2,
     “Creating a service policy” on page 221.
   – Create a work class for the application. See section 10.5.1, “Creating work
     classes” on page 227.
   – Assign the transaction class to the application’s work class. See section
     10.5.2, “Assigning a transaction class to a work class” on page 230.

**Integrating PHP servers**

On each PHP system, perform the following tasks:

1. Install and configure the PHP software.

Using the WebSphere administrative console, perform the following tasks:

2. Define the PHP system as a deployment target to WebSphere, and install the
   WebSphere Extended Deployment middleware agent to the PHP system
   using the centralized installation manager.
   This step is not necessary if PHP is on a system with a WebSphere node.
   The steps required to perform this task are discussed in section 6.2, “Working
   with PHP servers” on page 109.

3. Start the middleware agent.
   The steps required to perform this task are discussed in section 6.1.2,
   “Starting the middleware agent” on page 92.

4. Create one or more dynamic clusters. Creating a dynamic cluster
   automatically creates the servers.
   The steps required to perform this task are discussed in section 8.4,
   “Dynamic clusters with PHP servers” on page 164.

5. Deploy applications to the dynamic clusters.
   The steps required to perform this task are discussed in section 9.2,
   “Installing PHP applications” on page 179.

6. Define service policies and associate them with the applications.
   – Create a service policy and transaction class. See section 10.3.2,
     “Creating a service policy” on page 221.
   – Create a work class for the application. See section 10.5.1, “Creating work
     classes” on page 227.
Assign the transaction class to the application’s work class. See section 10.5.2, “Assigning a transaction class to a work class” on page 230.

**Integrating other middleware servers**

On each system that will host the other middleware servers, perform the following tasks:

1. Install and configure the software for each server type (for example Tomcat, WebSphere Application Server Community Edition, and so on).
2. Install applications on the server using the middleware's native tools and procedures.

Using the WebSphere administrative console, perform the following tasks:

3. Define the middleware system as a deployment target to WebSphere, and install the WebSphere Extended Deployment middleware agent to the system using the centralized installation manager.
   The steps required to perform this task are discussed in section 6.1, “Working with assisted lifecycle servers” on page 86.
4. Start the middleware agent.
   The steps required to perform this task are discussed in section 6.1.2, “Starting the middleware agent” on page 92.
5. Define middleware server definitions to represent the servers to the cell.
   The steps required to perform this task are discussed in section 6.1.4, “Defining the server to the WebSphere configuration” on page 94.
6. Create one or more dynamic clusters.
   The steps required to perform this task are discussed in section 8.5, “Dynamic clusters with other middleware servers” on page 166.
7. Define the existing applications to the cell. Map the application to the appropriate dynamic cluster.
   The steps required to perform this task are discussed in section 9.3, “Defining other middleware applications” on page 183.
8. Define service policies and associate them with the applications.
   
   - Create a service policy and transaction class. See section 10.3.2, “Creating a service policy” on page 221.
   
   - Create a work class for the application. See section 10.5.1, “Creating work classes” on page 227.
- Assign the transaction class to the application’s work class. See section 10.5.2, “Assigning a transaction class to a work class” on page 230.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 316. Note that some of the documents referenced here may be available in softcopy only.

- *Best Practices for Implementing WebSphere Extended Deployment*, SG24-7343

Online resources

The following Web sites are also relevant as further information sources:

- WebSphere Extended Deployment Information Center
- WebSphere Application Server V6.1 Information Center
- Storage networking solutions home page at:
- Apache JMeter
- *Exploring new network topologies made possible by WebSphere XD and the On Demand Router* by Kyle Brown, Keys Botzum, and Bill Hines
- OpenSSH
  [http://www.openssh.com](http://www.openssh.com)
Enabling verbosegc in WebSphere Application Server

http://www-1.ibm.com/support/docview.wss?uid=swg21114927

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at the following Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
Symbols
$com.ibm.websphere.wxdopProductShortName

A
access logging 52, 122
activate application edition 200
activation methods 198
activation state 198
addAgent.log 92
administrative console port 67
administratively directed routing differentiation 201
age-based 297
age-based condition 272
alerts 248
Apache 166
Apache Geronimo 8–9, 40, 85
Apache HTTP Server 8–9, 40, 85
Apache JMeter 244
Apache Tomcat 13, 29, 40, 85, 166, 183
Apache Tomcat Server 240
Apache Tomcat server 8–9, 96–98, 166
APC 17, 140–142, 144–145, 171, 216, 239
application activation 197–198
application edition 187–188
application edition compatibility 205
application edition management 4, 186
application edition manager 24, 188, 191, 198, 203
application versioning 24
application placement manager 24
application rollout 186, 198
application server weight 148
application version 187
application versioning 24, 186
ARFM 17, 115, 139, 141, 146, 216–218
ARFM controller 17
ARFM gateway 17, 20, 116, 260
assisted lifecycle 8
assisted lifecycle server 13, 29, 86, 149
assisted lifecycle support 5, 8, 10, 40, 85–86, 190, 229
atomic rollout 207, 209
autonomic managers 4, 16
autonomic request flow manager 17
average drop rate 259
average queue length 259
average relative performance 259
average response time 220, 222
average throughput 250

B
back-out 211
backupConfig 82
base edition 188
batch workload 6
BEA WebLogic 13, 166
BEA WebLogic Server 8–9, 85
BEA WebLogic Server WebSphere Application
Server Community Edition Custom HTTP servers 40

C
Cache-Control header 127
cachespec.xml 128
caching 51, 127
centralized installation manager 5, 10–12, 29–30,
39, 54, 86–87, 312–313
chargeback 245
chart 257–258, 260
chart group 257, 260, 263
charts 256
checkHmmLocation.jacl 279
checkpoint types 55
classic topology 42
classification rule 218, 232, 235–236
Cluster
Static 148
Compute Grid 6
compute-intensive workload 6
concurrency 19
concurrency limits 19
concurrency requests 259
concurrent activation 198, 200, 202
concurrent mode 202
connection keep-alive 50
Content Server Connection 121
### Table of Content

- **context root**: 229
- **core components**: 246, 252
- **core group**: 46, 246, 250–251
- **core group bridge**: 45–46
- **CPU**: 2–3
- **CPU capacity**: 260
- **CPU consumption**: 260
- **CPU demand**: 260
- **CPU usage**: 256
- **CPU utilization**: 259
- **createHealthPolicy**: 284, 287
- **cross-cell communication**: 44–45
- **custom action**: 276, 280–282, 294
- **custom condition expression**: 284
- **custom error page**: 128
- **custom health policy**: 281, 284, 286
- **custom HTTP Server**: 166
- **custom HTTP server**: 8–9, 39
- **customHealthAction**: 283
- **cycle length**: 277

### Data Grid
- **data metric**: 257–258, 263
- **data set**: 258, 263
- **data set type**: 258, 263
- **deactivate application edition**: 199
- **default excessive memory**: 305
- **default excessive response time**: 305
- **default memory leak**: 305
- **default policies**: 305
- **default request timeout**: 305
- **default storm drain**: 305
- **default work class**: 227
- **DefaultNodeGroup**: 160
- **defaultXD template**: 163
- **deployment manager**: 27, 38, 64, 68, 70, 310
- **discretionary**: 219, 222
- **dispatch error time**: 260
- **dispatch time**: 260
- **distributed server environment**: 178
- **DMZ**: 43
- **drainage interval**: 211
- **DWLM**: 146
- **dynamic application placement**: 3, 14
- **dynamic cluster**: 13–14, 18, 52, 86, 117, 147–149, 158, 164, 172, 204, 311–312
- **dynamic cluster membership**: 14, 150
- **dynamic cluster migration**: 172
- **dynamic cluster operating mode**: 170
- **dynamic operations**: 3, 13
- **dynamic workflow manager**: 216
- **dynamic workload management**: 172
- **dynamic workload manager**: 18, 146, 148, 303

### E
- **Edge Components**: 26
- **e-mail notification**: 270, 307
- **Error 503**: 154
- **event notification**: 307
- **excessive memory consumption**: 284
- **excessive memory usage**: 300
- **excessive memory usage condition**: 273
- **excessive request timeout**: 298
- **excessive request timeout condition**: 271, 273
- **excessive response time**: 299
- **excessive response time condition**: 273
- **execution time**: 259
- **expires header**: 127
- **extended repository service**: 54

### F
- **Fast lock recovery**: 70
- **Fix Packs**: 28
- **free memory**: 259
- **full lifecycle server**: 13
- **full lifecycle support**: 5, 8–9, 14, 40, 188, 229

### G
- **garbage collection**: 302
- **generic lifecycle**: 8
- **generic lifecycle server**: 13
- **generic lifecycle support**: 5
- **generic Lifecycle Support**: 8, 11–12, 40
- **generic server cluster**: 16, 117, 218, 240
- **goal**: 16, 219
- **goal delta value**: 223
- **goal type**: 219
- **group rollout**: 206, 209
- **group size**: 206
- **guided activities**: 271

### H
- **HA deployment manager**: 69, 72–73, 78–79, 81–82
Index

hard reset 211
health condition 272
health condition properties 293
health controller 271, 276–277, 279, 306
health management 4, 270–271
health monitoring 269, 271
health policy 270, 272, 280, 291
health policy breach 306
Heap size 302
high memory watermark 260
HmmControllerProcs.jacl 278
HTTP 1.1 127
HTTP endpoint 16
HTTP method exclusion 121

I
IBM HACMP 27
IBM HTTP Server V6.0 27
IBM Java Developer Kit 302
IBM JDK 302
IBM JVM 275
IBM SAN FS 70
IBM Storage Area Network File System 27, 70
IBM Tivoli System Automation 27
IIOP traffic shaping 20
importance 219–220
interim fix 28
interruption-free rollout 204

J
J2EE 218
J2EE application 188
J2EE Web application 218
Java virtual machine 302
javacore 274
JBoss 29, 40, 85, 166
JBoss server 8–9, 99, 101
JBoss, Apache 13
JMS traffic shaping 20
JMX 70–71
JMX_SOAP_ADDRESS 78
JMX_SOAP_CONNECTOR 78
JMX_SOAP_PROXY_ADDRESS 78, 83
jobs completed 259
jobs dispatched 260
jobs error 260
jobs failed 260
jobs queued 260
jobs requested 259
JSP precompile 176
JVM 302
Heap dump 302
JVM heap dump 275
JVM settings 49

L
lazy application start 15, 128, 154–155
listHealthPolicies 287
load balancer 26
logging 131, 236
long-running workload 6

M
maintenance mode 173–174, 256, 275
map modules to servers 177
master configuration repository 70
maximum concurrency 259
Maximum Consecutive Restarts 278
maximum open connections 50
maximum persistent requests 50
memory 2–3
memory leak 284, 302
memory leak condition 271, 273
message size 48
middleware agent 8, 10–12, 16, 27, 29–30, 36,
39–40, 54, 86–87, 91–93, 97, 99–100, 103–104,
107, 109, 174, 255–256, 312–313
middleware components 34
middleware server 313
middleware server support 8
migration 56
minimum restart interval 278
multi-cell environment 132
mwServerStatus 93

N
Network File System Version 4 27, 70
NFS 72–73
node 52, 246, 254
node agent 39, 311
node group 4, 39, 52, 150, 161, 173, 258
node types 39
non-destructive update 188
numVerticalInstances custom property 157
ODC  17, 117
ODR  117–118, 200, 206, 241, 249
On Demand Configuration manager  17
On Demand Router  3, 14, 27–28, 41, 48, 70, 78, 115, 141, 311
operation alerts  248
operational policy  15
Operations Optimization  2, 7
other middleware application  183
other middleware nodes  39
other middleware servers  13, 85, 95, 166, 313

percentile response time  220, 222, 259
performance goal  216
PHP  40
PHP application  175, 179
PHP server  5, 8, 13, 27, 30, 39, 85, 109–114, 150–151, 164, 168, 181, 188, 312
PlacementControllerProcs.jacl  144
plug-in  122, 127
plug-in configuration  47
plug-in script  124
plugin-cfg.xml  47, 123
pmt  64, 83
Policy  219
policy-based workload management  14–15
power coefficient  260
pre-production testing  202
profile creation  74
profile management tool  64
prohibited restart times  278
proxy plug-in generation  29, 311
proxy server  122

Quality of service  1
queue time  260
queue wait time  220

reaction mode  270, 272, 276, 293
Redbooks Web site  316
Contact us  xiii
reformed topology  43
relative importance  16

remote agent  86
report  256
repository checkpoint  54
reset strategy  210–211
resident memory  260
resource contention  219
resource sharing  13–14
restart server  302
restart timeout  278
restoreConfig  82
RFC 2616  127
RMI  70
rollback for recovery  211
rollout activation  199, 204, 206–207
route to multiple cells  44
routing policy  16, 193, 240–241, 244
routing rule  49, 198, 201, 241
runtime operation views  247
runtime task  305
runtime task approval timeout  171
runtime tasks  305

SAN FS  27, 72–73
server restart  297–298, 300–301
server type  40, 52
server weight  18
Service  219
service goal type  16
service policy  14–15, 18, 116, 216–221, 237, 244, 312–313
service policy membership  223
servicepolicy.py  221
Shared file system  70
Simple activation  198
simple activation  199
smoothing window  139
SOAP  70
Soft reset  210
split deployment  188
SSH  88
SSL  49, 121
stability  250–251, 253, 256
stack product  41, 44
startAgent  92, 174
startManager  68, 78
startNode  84, 174
static cluster  117, 148
stopManager 78
storm drain condition 273, 303
SystemOut.log 79

T
takeover 72
task 239, 305
task ID 240
thread dump 274, 298
time period value 223
time stamp format field 267
Tivoli Usage and Accounting Manager 268
topology 37, 42
total CPU capacity 260
total EJB method calls 259
total memory 260
total requests 259
total transactions 259
Trace 131
trace 145, 236
traffic shaping 4, 14, 18
transaction class 18, 216, 223–226, 230, 312–313

U
unmanaged Web application 191
up time 259
URI 216
used memory 259
utilization 259

V
Validation activation 199
Validation mode 202
validation mode 186, 203
verbosegc 49
vertical stacking 15, 155–157
view synchrony protocol 45, 251
visualization 245–246
visualization data service 246, 266

W
WASPostUpgrade 58
WASPreUpgrade 58
WC_adminhost 78–79, 83
WC_adminhost_secure 78, 80, 83
Web server 27, 29, 41, 236, 310
Web server keep-alive settings 51
Web server plug-in 27, 123, 310
Web server plug-in configuration file 47
WebSphere application 176
WebSphere Application Server 40
WebSphere application server 13
WebSphere Application Server Community Edition 8, 13, 29, 85, 104–105, 166, 183
WebSphere Application Server Network Deployment node 39
WebSphere Extended Deployment node 39
Weights 303
work coefficient 260
work profiler 17
workload 3, 304
workload condition 274
workspace 70
wsadmin 70
WXDOP 161

X
xd_APCconfig.jacl 144
xd_augment 59–60
xd_hadmgrAdd 74–77, 81–82
xd_hadmgrRemove 82
XD-CGB-EXPORT 46
XDUpgrade 58–60

Z
z/OS Automatic Restart Management (ARM) 67
Implement dynamic operations and resource sharing

Operations Optimization enables application services to share a common pool of heterogeneous resources, resulting in the following:

- Better resource utilization
- Predictable and consistent attainment of application service levels
- Increased flexibility to respond to changing application demands
- Improved ability to manage complex and heterogeneous environments

Build a high-performance infrastructure

Monitor performance and health

This IBM® Redbooks publication will help you understand how WebSphere® Extended Deployment Operations Optimization can improve the performance and reliability of your systems. It takes you through the steps required to implement Operations Optimization.