

Administering Intelligent Management, Version 8.5

Administering Intelligent Management



Note

Before using this information, be sure to read the general information under “Notices” on page 489.

Compilation date: June 12, 2012

© Copyright IBM Corporation 2012.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

How to send your comments	v
Using this PDF	vii
Administering Intelligent Management	1
Intelligent Management overview	1
What is new in this release	1
z/OS considerations	2
Virtualization and Intelligent Management	2
Dynamic operations.	11
Elasticity mode	19
Runtime operations.	21
Supported middleware server types.	22
Middleware nodes and servers	22
Migrating from WebSphere Virtual Enterprise	24
Migrating to WebSphere Application Server Version 8.5 Intelligent Management cells	25
Preparing the hosting environment for dynamic operations	29
Configuring VMware Infrastructure 3 platforms and Intelligent Management	29
Creating and configuring ODRs	36
Creating dynamic clusters	75
Adding middleware servers to configurations	93
Deploying applications with defined service levels	118
Defining a service policy	123
Defining service policy rules	146
Deploying and managing application editions without loss of service	155
Application edition manager concepts	155
Installing an application edition	164
Activating an edition	165
Creating routing policies for application editions	167
Validating an edition	168
Performing a rollout on an edition	171
Performing a rollback on an edition	176
Deleting an installed edition	176
Troubleshooting application edition manager	177
Managing the Intelligent Management environment	180
Setting maintenance mode	180
Routing to servers that are in maintenance mode	182
Generating Simple Network Management Protocol (SNMP) traps	182
Configuring the autonomic managers.	183
Routing requests to nodes that are not running Intelligent Management	241
Configuring a high availability deployment manager environment	251
Using centralized logging to diagnose problems	260
Monitoring operations	262
Runtime operations overview.	263
Creating and managing reports	264
Creating and managing reports	265
Configuring the visualization data service	267
Task management.	268
Troubleshooting extended administration	272
Configuring Intelligent Management to work with other IBM products	272
Enabling the on demand router to work with IBM Enterprise Workload Manager	272
Intelligent Management reference	291
Port number settings.	291

Intelligent Management scripts	294
Administrative roles and privileges	330
Administrator scripting interfaces	332
Custom properties for Intelligent Management	439
Performance logs	478
Notices	489
Trademarks and service marks	491
Index	493

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
 1. Display the article in your Web browser and scroll to the end of the article.
 2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an email form appears.
 3. Fill out the email form as instructed, and submit your feedback.
- To send comments on PDF books, you can email your comments to: **wasdoc@us.ibm.com**.

Your comment should pertain to specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer. When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about your comments.

Using this PDF

Links

Because the content within this PDF is designed for an online information center deliverable, you might experience broken links. You can expect the following link behavior within this PDF:

- Links to Web addresses beginning with `http://` work.
- Links that refer to specific page numbers within the same PDF book work.
- The remaining links will *not* work. You receive an error message when you click them.

Print sections directly from the information center navigation

PDF books are provided as a convenience format for easy printing, reading, and offline use. The information center is the official delivery format for IBM WebSphere Application Server documentation. If you use the PDF books primarily for convenient printing, it is now easier to print various parts of the information center as needed, quickly and directly from the information center navigation tree.

To print a section of the information center navigation:

1. Hover your cursor over an entry in the information center navigation until the **Open Quick Menu** icon is displayed beside the entry.
2. Right-click the icon to display a menu for printing or searching your selected section of the navigation tree.
3. If you select **Print this topic and subtopics** from the menu, the selected section is launched in a separate browser window as one HTML file. The HTML file includes each of the topics in the section, with a table of contents at the top.
4. Print the HTML file.

For performance reasons, the number of topics you can print at one time is limited. You are notified if your selection contains too many topics. If the current limit is too restrictive, use the feedback link to suggest a preferable limit. The feedback link is available at the end of most information center pages.

Administering Intelligent Management

Use Intelligent Management for application edition management, intelligent routing, application server health management, and to improve performance by using dynamic clustering and overload protection. You can prepare the hosting environment for dynamic operations, manage application editions, monitor operations, configure Intelligent Management to work with other IBM® products, and so on.

With application edition management, you can verify that a new version of your application works in your production environment before sending real traffic to it. You can manage the health of your application server environment with a policy-driven approach that allows action to be taken when monitored criteria is met. With dynamic clusters, automatically scale up and down the number of running cluster members as needed in order to meet response time goals for your users. With overload protection, limit the rate at which the on-demand router (ODR) forwards traffic to application servers in order to prevent heap exhaustion, CPU exhaustion, or both types of exhaustion from occurring.

Intelligent Management overview

Intelligent Management provides an enhanced quality of service in dynamic operations and extended manageability.

What is Intelligent Management?

Intelligent Management provides application server virtualization, resource management, and a host of advanced operational facilities, such as performance visualization, health management, and application edition management. Use Intelligent Management to enhance operational efficiency by deploying dynamic operations, servicing high-volume transactional workloads with linear scalability and high availability, and by managing large scale, continuously available application server environments.

Application edition management provides the ability to roll out new versions of applications without experiencing downtime for a maintenance window. Health management offers you the ability to specify conditions to monitor and corrective actions to take when the conditions are observed. Both of these capabilities improve the resiliency and availability of applications in an Intelligent Management environment. Dynamic workload management allows you to automatically change the application footprint during runtime, based on incoming application demand. As a result, multiple applications sharing a common resource pool can provision the average usage, and can share a set of resources to handle peak usage scenarios.

Intelligent Management does not change any Application Programming Interfaces (APIs) or WebSphere® Application Server Network Deployment binary files. Specific Intelligent Management features can only be used with specific applications; any limitations depend on the application design, performance, and quality of service requirement. Most limitations can be overcome by tuning your topology or configuring your applications.

Restriction:  The product does not support Session Initiation Protocol (SIP) features on the z/OS® operating system.

What is new in this release

If you are transitioning from WebSphere Virtual Enterprise, this section provides a high level view of the new Installation Manager features.

New predefined health policy

The garbage collection health policy helps you avoid performance issues caused by garbage collection. If the percentage of time used for garbage collection exceeds the defined value for a specified time period,

the garbage collection health policy takes corrective actions maintaining the health of your environment.

Dynamic cluster support for message-driven beans loaded by MQ

You can use dynamic clusters for message-driven beans loaded by the MQ messaging provider.

Intelligent Management can be enabled or disabled in the administrative console

You can enable or disable the Intelligent Management functions in order to control the performance in your environment.

z/OS considerations

Considerations for using Intelligent Management for z/OS are as follows:

- Federating middleware agents from distributed platforms into a z/OS deployment manager is not supported.
- The on demand router (ODR) does not support Session Initiation Protocol (SIP) features on the z/OS operating system.

Virtualization and Intelligent Management

By configuring application infrastructure virtualization with Intelligent Management, you can pool together resources to accommodate the fluctuations of workload in your environment and increase the quality of service. You can also use application infrastructure virtualization with hardware virtualization capabilities that are provided by the physical hardware on which the product is hosted.

Application infrastructure virtualization

With *application infrastructure virtualization*, you can separate applications from the physical infrastructure on which they are hosted. Workloads can then be dynamically placed and migrated across a pool of application server resources, which allows the infrastructure to dynamically adapt and respond to business needs. Requests are prioritized and intelligently routed to respond to the most critical applications and users.

Typically, applications and Java Platform, Enterprise Edition (Java EE) resources are statically bound to a specific server. Some of these applications might experience periodic increases in load that last a short time. The most costly time for an application to become unavailable is during a period of high demand. You must build your IT infrastructures to be able to accommodate these peaks. During the majority of time when your systems experience normal load, a large percentage of your computing capacity might go unused, making inefficient use of IT investments.

In a static environment, applications often span multiple enterprise archives (.ear files), and are not comprehensively defined so that the application can be portable between environments. Statically deployed applications rely on information that is found in the server to which they are deployed.

In the virtualized dynamic operations environment of Intelligent Management, the static relationship is replaced with a dynamic relationship with looser coupling of applications or resources and server instances. Instead of statically binding applications to servers or clusters, you deploy applications to dynamic clusters, which are application deployment targets that can expand and contract depending on the workload in your environment.

After you deploy your applications to be mobile by using dynamic clusters, the placement of the applications is determined by the operational policies that you define. Autonomic managers control the placement of the server instances and how workload is routed for each application. If workload increases

for a specific application, the number of server instances for the dynamic cluster that is hosting the application can increase, using available resources from other applications that are not experiencing increased workload.

Application infrastructure virtualization benefits:

- **Improved management of software and applications:** Management processes become more repeatable and less error-prone by using automated services and operational policies.
- **Allocation of software resources:** Dynamic reallocation of resources can occur based on shifting distributions of load among applications.
- **Increased number of applications:** More applications can run in a virtualized application environment than in a static configuration.
- **Reduced configuration complexity:** Loosened coupling between applications and the application server instances reduces the overall complexity and provides for a better, more usable environment.

Application infrastructure virtualization example

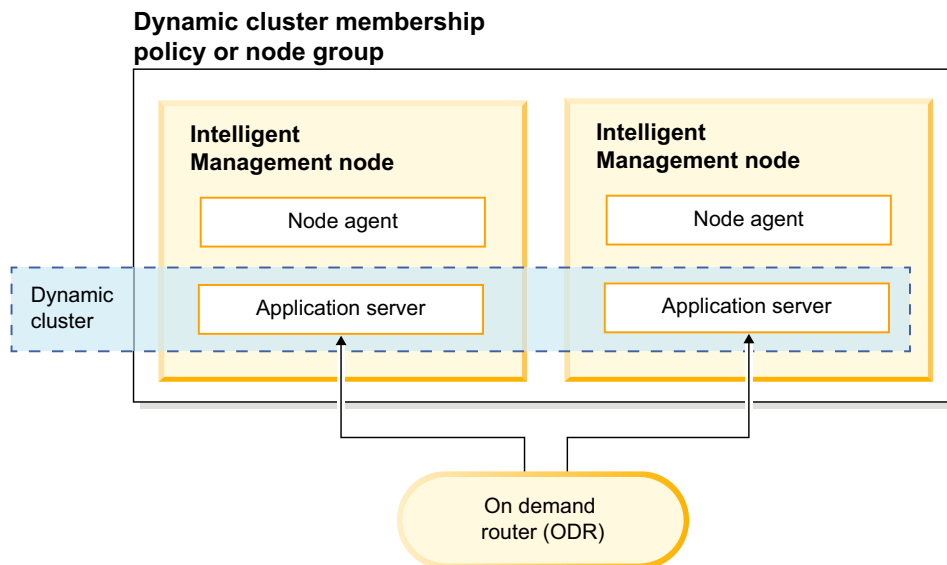


Figure 1. Application infrastructure virtualization in an Intelligent Management environment. You deploy an application to a dynamic cluster that has a specified membership policy or node group. You do not deploy your applications to specific application servers. Instead, the application placement controller starts application server instances for your dynamic cluster based on the settings that you chose for the dynamic cluster.

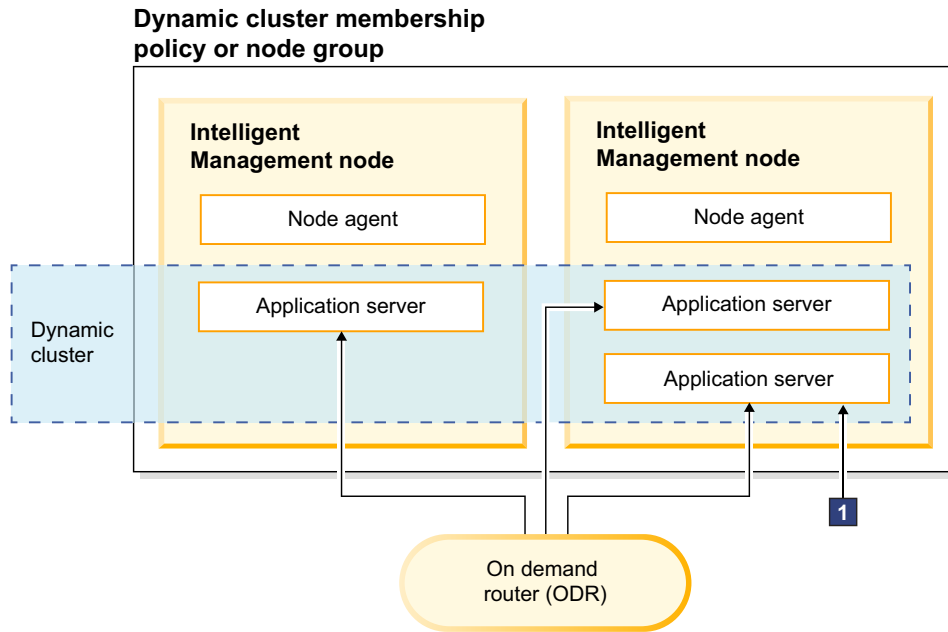


Figure 2. The starting of an additional application server to react to changes in application load. Additional application servers can start on the nodes that are selected by your dynamic cluster membership policy to handle additional requests that are coming in for the application.

¹ If the workload increases for a specific application, the number of server instances for the dynamic cluster that is hosting the application can increase. These server instances use available resources from other applications that are not experiencing increased workload.

Hardware virtualization

While Intelligent Management provides virtualization of applications in your environment, you can also use Intelligent Management on virtualized hardware, such as ESX, to take advantage of the hardware virtualization capabilities provided by the hosting environment.

Hardware virtualization benefits:

- **Reduced amount of hardware in your environment:** You can run multiple Intelligent Management nodes on the same physical hardware.
- **Improved hardware management:** You can more easily manage your environment because you have fewer physical machines and can use the hardware virtualization software to manage your images.
- **High availability of hardware:** By configuring server failover, your physical hardware can be highly available. When one server fails, it can be replaced by another server.
- **Dynamic allocation of hardware:** The physical resources, such as processors and memory, on your hosting computers can be shared among the virtual servers in your environment and dynamically allocated as needed. Because the resources are dynamically allocated, restarting the servers is not necessary.
- **Shared storage:** Multiple virtual servers or logical partitions can share the same physical storage. You do not need a physical hard drive for each virtual machine or LPAR.

Intelligent Management in an environment with server virtualization

Intelligent Management can operate in supported virtualized hardware environments. Different hardware vendors provide different virtualization capabilities, so the behavior of Intelligent Management in different hardware virtualization environments can vary. However, common themes exist in hardware virtualization environments, such as the ability to share hardware resources across the virtual servers or logical

partitions. Server virtualization environments can run in *shared processor mode* or *dedicated processor mode*. When you use shared processor mode, the physical processors are pooled and shared between the servers or logical partitions that are running on the physical machine. When you use dedicated processor mode, the physical processors are statically assigned to each virtual server or logical partition.

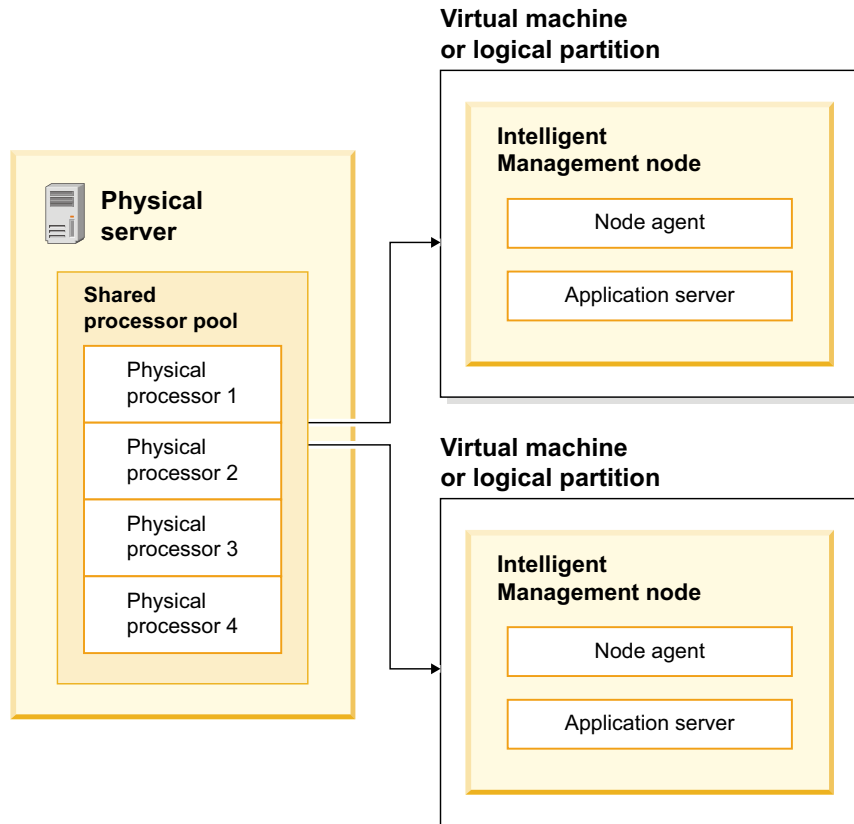


Figure 3. Shared processor mode. In shared processor mode, the physical processors are pooled and shared among the virtual servers or logical partitions.

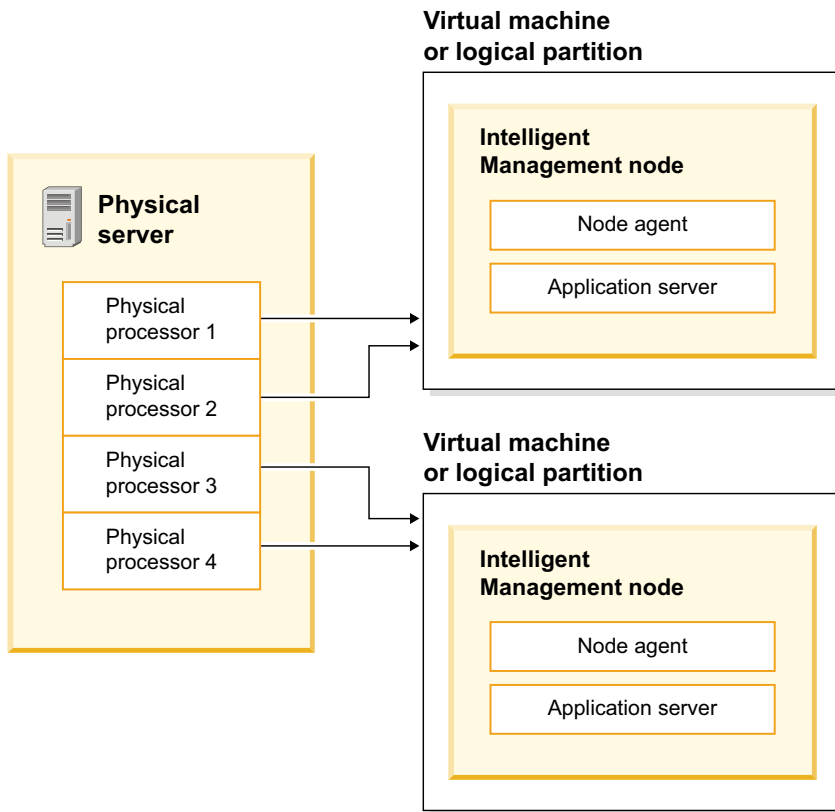


Figure 4. Dedicated processor mode. In dedicated processor mode, the physical processors are statically assigned to each virtual server or logical partition.

Intelligent Management can also run in hardware virtualization environments with dedicated processor mode. The processor capacity is statically fixed to each virtual server or logical partition. The capacity and assignment do not change dynamically. Because the processor resource does not change for each virtual server or logical partition, using dedicated processor mode does not affect the traffic management and virtualization features of Intelligent Management.

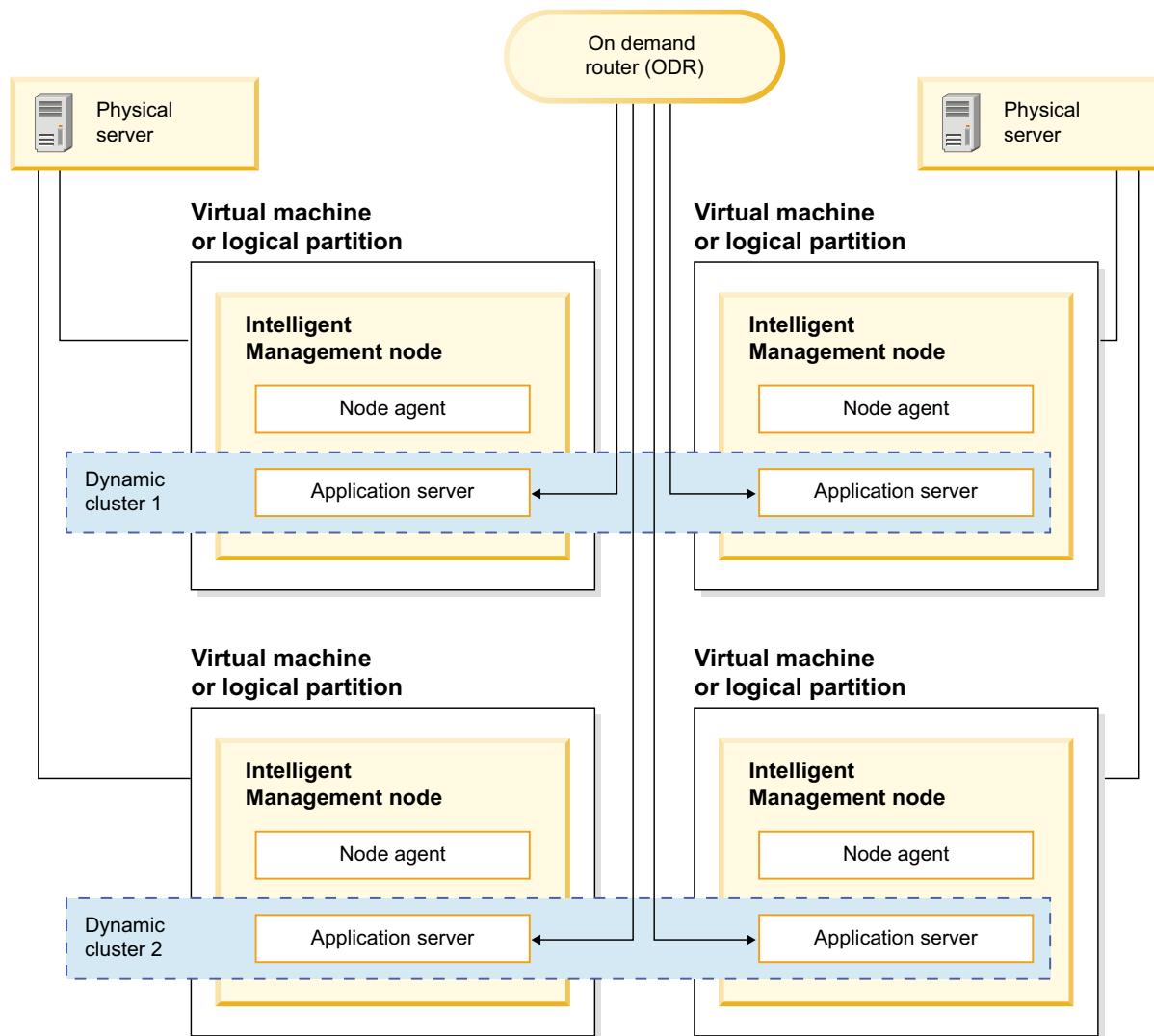


Figure 5. Coexistence of application infrastructure and hardware virtualization.

On-demand router in an environment with server virtualization

The on-demand router (ODR) should never be constrained by either CPU or memory usage. Therefore, when you install the ODR in an environment with server virtualization, configure the virtual machine or LPAR in which the ODR runs in dedicated processor mode, or configure it in a mode that guarantees the ODR receives enough CPU resources and dedicated memory when the ODR runs.

Deploying the product on Solaris 10 Zones with shared CPU

You can install and deploy the product on Solaris 10 Zones in non-dedicated mode in which CPU and memory are shared between zones.

If Solaris 10 Zones are configured to share CPU and memory between zones, the product must be installed on the global zone of each Solaris machine. This installation can be a standard node installation.

The node agent that is running in the global zone of each Solaris machine provides the necessary information for all zones on the machine. While the node agent process is running, it automatically gathers the appropriate information required for the product to function appropriately in this environment.

Configuring AIX 5.3 and AIX 6.1 on POWER5 and POWER6 (Micro-Partitioning)

To configure AIX® Micro-Partitioning® in shared and uncapped mode to work with Intelligent Management, you must enable performance collection in the Hardware management console (HMC).

Before you begin

Ensure you have access to the HMC for your AIX servers.

About this task

If you do not configure this setting in your HMC, but you are using AIX micro-partitioning in shared and uncapped mode, the following message displays in your system out log of the node agent process:

```
APMI_XDSYSTEM_HW_INFO_WARN=ASPS0023W: The logical partition on which this node resides has a mode of shared, a type of uncapped, but performance collection is not enabled.
```

You must enable this setting so that Intelligent Management can recognize more than the entitled capacity for a logical partition (LPAR).

Procedure

1. Enable the performance hardware collection setting for each LPAR. In the HMC, click **Systems management > Servers > *physical_server_name* > *LPAR_name***. Select **Allow performance hardware collection**.
2. Click **OK**.
3. Restart the LPAR.

Results

Intelligent Management can recognize more than the entitled capacity for a logical partition (LPAR).

Intelligent Management in a micro-partitioned environment:

You can use Intelligent Management in a micro-partitioned environment. You can use shared processor partitions with Intelligent Management.

Terminology for server virtualization on POWER® AIX systems

Review the following terms when using Intelligent Management on a POWER AIX system:

Logical partitioning

The ability to divide the resources of a system to create multiple separate servers. Each server runs its own operating system.

Micro-partitioning

The ability to share a pool of physical processors across multiple logical partitions. Physical processors can be allocated to a partition in increments of 0.1 a processor.

Shared processor partition

A partition that is configured to use a shared processor pool. A shared processor partition is a type of micro-partition.

Entitled capacity

The percentage of processor usage that is granted to a partition; specified in terms of .01 of a processor.

Capped partition

A partition that cannot be granted more processing units beyond the configured entitlement for the partition.

Uncapped partition

A partition that can exceed its configured entitlement when needed, if resources permit.

Entitled capacity

A shared processor partition has a metric called *Entitled Capacity Percentage*. This metric represents the percentage the partition is using of its entitlement at a given point in time. The metric is visible in popular AIX system monitoring tools like the `lparstat` command, the `nmon` command, and the `topas` command.

An uncapped, shared processor partition can be assigned more processing capacity beyond its entitlement. The amount of processing capacity beyond its entitlement depends on the availability of the availability of processors in the shared pool and the maximum amount that the virtual processor configuration allows.

Intelligent Management and shared processor partitions

Intelligent Management can be used in environments with shared processor partitions or with dynamically shared processor pools on physical hardware.

Shared processor partition statistics are available. These statistics help you understand the utilization of a shared processor partition and the dynamic capacity of the shared processor pool. The statistics monitoring Intelligent Management when operating in a micro-partitioned environment.

Supported server virtualization environments

Before you deploy Intelligent Management on virtualized servers, you must understand the limitations for the server virtualization platform that you are using.

Remember: This topic lists the most up to date information on the support of various server virtualization platforms.

Table 1. Server virtualization environments

Virtualization platform	Restrictions	Supported processor sharing mode
AIX 5.3 and AIX 6.1 on POWER5 and POWER6® (Micro-Partitioning)	<p>For more information about required configuration steps, read about configuring AIX 5.3 and AIX 6.1 on POWER5 and POWER6 (Micro-Partitioning).</p> <p>For uncapped shared processor partitions, equal partition weights are recommended. Intelligent Management does not have a partition to physical machine mapping and thus does not use the partition weight when making workload distribution and server placement decisions.</p>	<p>All Intelligent Management features are supported on AIX Micro-partitioning in capped or uncapped mode.</p> <p>Shared capped, shared uncapped, and dedicated modes are supported for both POWER5 and POWER6. Support for dedicated donating and multiple shared pools modes are available on POWER6 only.</p>
AIX 7.1 on POWER7®	None.	<p>POWER7 processor in POWER6 compatibility mode</p> <p>POWER7 processor mode</p>
Application Workload Partition (WPAR) AIX 6.1 and 7.1	None.	Shared.
KVM in SUSE Linux Enterprise Server (SLES) 11	None.	Dedicated and shared.

Table 1. Server virtualization environments (continued)

Virtualization platform	Restrictions	Supported processor sharing mode
Red Hat KVM as delivered with Red Hat Enterprise Linux (RHEL) 5.4, 5.5, 6.0	None.	Dedicated and shared.
Linux on POWER (Micro-Partitioning)	None.	Dedicated.
Linux on z/VM®	<ul style="list-style-type: none"> The guest operating system must be Red Hat Enterprise Linux (RHEL) 5.0, 5.1, 5.2 or SUSE Linux Enterprise Server (SLES) 10. Mapping a Linux on zSeries® operating system image to a physical logical partition (LPAR): Intelligent Management can balance workload across multiple z/VM virtual machines that are running Linux as the guest operating system. However, it does not have knowledge of the LPAR that is hosting the virtual machines on the z/VM image and is therefore unable to make workload balancing decisions based on the workload at the LPAR level. Service policy goals: Workload that is not under the control of Intelligent Management might be running on other Virtual Machines on the same z/VM image. This workload might affect service policy goals. 	Dedicated and shared.
Solaris Operating Environment 10 on Sun (Zones)	None for dedicated.	Dedicated and shared.
VMware ESX 3.5, 4.0, 4.1 VMware vSphere 4	<ul style="list-style-type: none"> For more information about limitations, read about VMware Infrastructure 3 platforms and Intelligent Management. 	Dedicated and shared modes are supported for ESX or VMware vSphere 4, but you must configure Intelligent Management to communicate with the ESX or VMware vSphere 4 hypervisor or the vCenter that is in control of the hypervisors where Intelligent Management is running.
Linux Xen	<ul style="list-style-type: none"> You must be using SLES 11 or later. You must use the following guidelines for setting up the clock correctly: Virtual Machine Clock Settings. Use para-virtualized mode over full-virtualized mode. Full-virtualized mode can lead to network bottlenecks that can adversely affect performance of the system. This performance problem is particularly possible for large topologies. 	Dedicated.

Table 1. Server virtualization environments (continued)

Virtualization platform	Restrictions	Supported processor sharing mode
HP-UX 11i v3 on HP using Virtual Partitions (vPars) or Integrity VMs	Currently not supported.	None.
Microsoft Hyper-V	Currently not supported.	None.

Virtualization options

Running Intelligent Management in an environment with server virtualization has advantages, because of the juxtaposition to hardware and the operating system. With application infrastructure virtualization, you can separate applications from the physical infrastructure on which they are hosted. Workloads can then be dynamically placed and migrated across a pool of application server resources, which allows the infrastructure to dynamically adapt and respond to business needs.

While there are benefits to server virtualization, this virtualization option can sometimes be too far removed from the application layer, and can therefore lack a quality of service based on application information, such as response times and response codes. Application infrastructure virtualization, on the other hand, can provide a quality of service at the application layer. Additionally, this virtualization option provides optimization and autonomic health corrections for the middleware layer through the use of Java virtual machines (JVMs). In other words, Intelligent Management minimizes the footprint that is required by the middleware layer, and keeps the middleware highly available through health management. A third benefit of application infrastructure virtualization is the application-demand-driven optimization of the operating system and virtual machine layers. In this sense, the application infrastructure virtualization layer optimizes the server virtualization layer by creating and deleting virtual machines as needed to meet application demand. This is a fully autonomic and optimized private cloud, minimizing the overhead, especially memory, that is incurred by idle virtual machines.

Table 2. Differences between virtualization options

Server virtualization	Application infrastructure virtualization
Hardware is separated from guest operating systems	Application server container is separated from the application
Increases server hardware utilization with server virtualization being used to create a pool of server resources	Increases application server container utilization with application infrastructure virtualization being used to create pools of application server resources
Leverages a layer of insulation that is called the hypervisor	Leverages a layer of insulation that is called the application fabric or application server virtual layer
Resource groups can be assigned resource profiles that are connected to quality of service needs	Application server resources are assigned to dynamic clusters to which applications are deployed
Equates to pinning more cores to the guest operating system (vertical scaling) as opposed to assigning more guests to the server (horizontal scaling)	New JVMs can be restarted in the dynamic cluster to scale out or can be shut down to scale down
Resource sharing is controlled by policies that are expressed in terms of hardware units (CPU cores, memory size)	Resource sharing is controlled by policies that are expressed in terms of what end users see (response time)
Has visibility to the hardware and guest operating system layers	Has visibility to the application, middleware, guest operating system, and hardware layers

Dynamic operations

Intelligent Management increases the quality of service by monitoring the virtualized application server environment, and by making workload management optimizations or recommendations based on observed data. This capability is referred to as *dynamic operations*.

Increasingly, businesses are rigidly tied to the availability and speed of applications that deliver essential services to customers. Loss of availability translates into lost business, which means lost opportunity and lost revenue. Dynamic operations is a fluid and dynamic environment, supporting the continuous availability of applications through application server virtualization and application virtualization, the dynamic placement of applications, prioritization and flow control of work to the applications, and integrating with overall dynamic operations infrastructure management.

In a typical WebSphere Application Server environment, there are sometimes static islands of dedicated resources to particular applications. This static structure leads to an inefficient use of resources. Some servers are not used to their full capability, and other servers are overloaded.

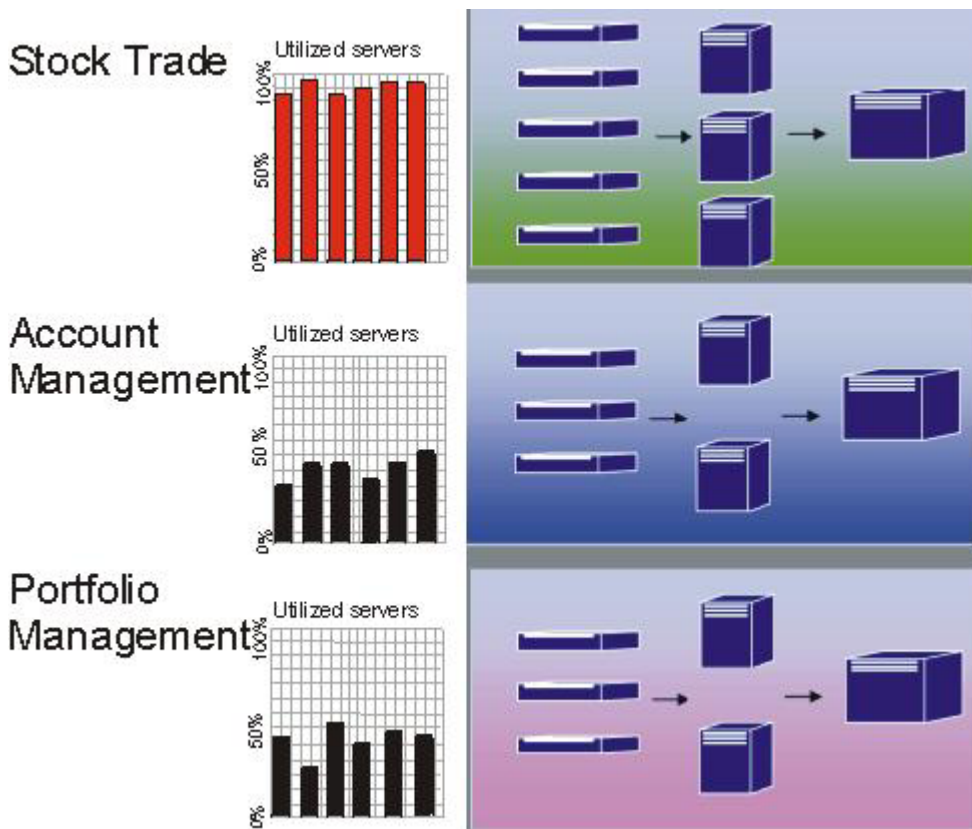


Figure 6. Typical WebSphere Application Server environment

Intelligent Management supports a far more flexible environment using its dynamic operations features. Dynamic operations consists of autonomic managers that maximize utilization using defined business goals. These autonomic managers monitor performance metrics, analyze the monitored data, offer a plan for running actions, and can start these actions in response to the flow of work.

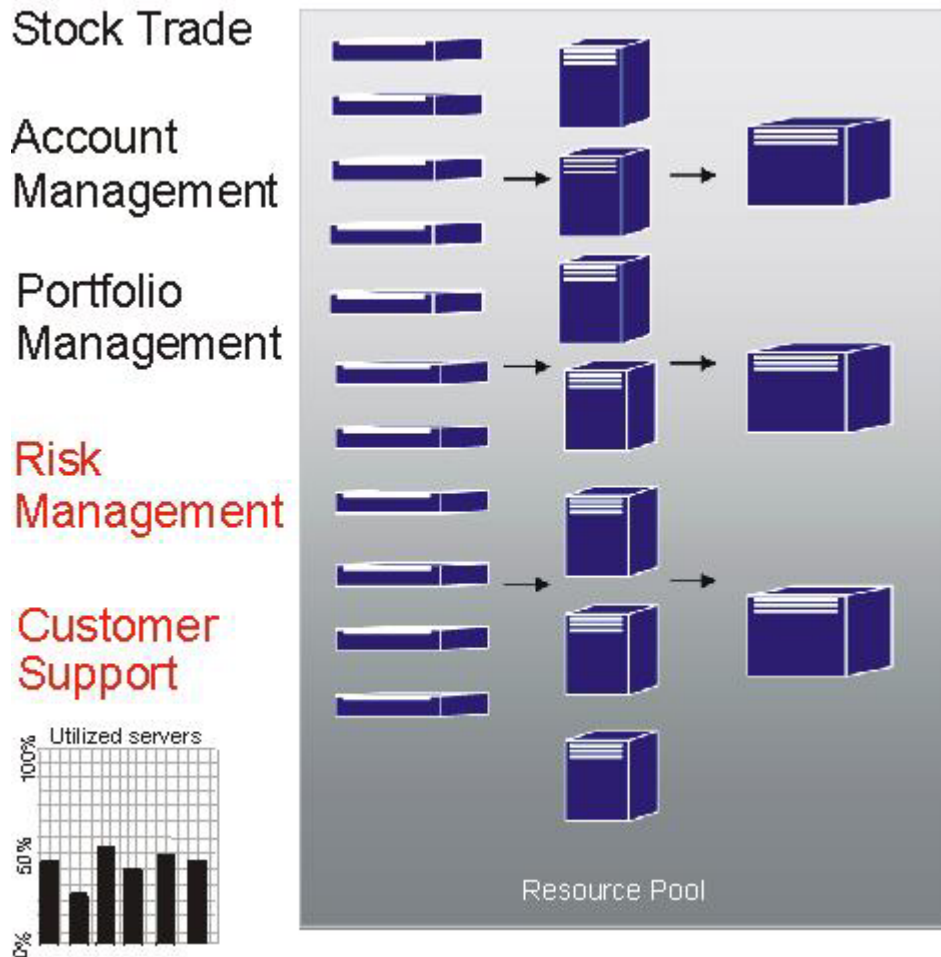


Figure 7. Intelligent Management environment

Intelligent Management offers the following autonomic managers as part of the dynamic operation functionality:

Autonomic request flow manager

Controls the order of requests into the application server tier and the rate of request flows. Using classification and the defined service goals, the autonomic request flow manager (ARFM) decides how and when to dispatch HTTP requests to the next tier. The ARFM also decides when Internet Inter-ORB Protocol (IIOP) and Java Message Service (JMS) requests are run at the application server tier, even though these requests are not routed through the ODR. For IIOP requests, only standalone Enterprise JavaBeans (EJB) clients are supported. Note that JMS support is only for message-driven beans.

Dynamic workload manager (DWLM)

Performs load balancing across available application servers. In particular, for a given request flow, DWLM balances requests across the available nodes to regulate response times. DWLM dynamically updates the application status as the application placement controller modifies a running application infrastructure.

Application placement controller

Creates and removes application instances to manage HTTP, SIP, and IIOP traffic. The application placement controller can dynamically address periods of intense workflow that otherwise require the manual intervention of a system administrator. For IIOP requests, only standalone EJB clients are supported.

Health management

Maintains a robust application server environment using a health policy to identify the criteria that require action. When the criteria is met, action is taken to ensure that the environment remains healthy.

Autonomic managers with the on demand router (ODR) are the primary functional parts of dynamic operations. An ODR can be defined and started before any service policies are defined, but operational policies can be defined before the appearance of the work to which they apply. However, if policies are not defined, the early work is handled by the default policies. When work enters the ODR, an optimization effort achieves a balance of performance results. As the work flows, the dynamic workload manager balances the load. As work variations change and the balance of work in the nodes is upset, the application placement controller, autonomic request flow manager, and the dynamic workload manager rebalance running applications to ensure efficient work flows.

The combination of these autonomic managers provides a seamless, end-to-end dynamic runtime ability.

trns: In WebSphere Virtual Enterprise, controllers start automatically as highly available managed items unless explicitly suppressed. In Intelligent Management, controllers are dormant until they detect that they are needed. When the controllers are needed, they become active. When the controllers detect that they are no longer needed, they become dormant again. Controllers that are disabled or are in manual mode do not use this detection capability.

Overview of dynamic operations

Intelligent Management is built upon a virtualized infrastructure that redefines the traditional concepts of Java Platform, Enterprise Edition (Java EE) resources and applications and their relationships with one another. This application infrastructure virtualization facilitates the product's ability to automate operations in an optimal manner, increasing the quality of service. By introducing an automated operating environment with workload management, you can reduce your total cost of ownership by performing more work while using less hardware.

Increasingly, businesses are tied rigidly to the availability and speed of applications that deliver essential services to customers. Loss of availability translates into lost business, which means lost opportunity and lost revenue. To meet this need, the dynamic operations environment is a fluid and dynamic environment, enabling applications to be available continuously through application virtualization, the virtualization of WebSphere resources, provisioning of WebSphere applications, prioritization and scheduling of applications, and integrating with overall dynamic operations environment infrastructure management.

The dynamic operations environment consists of autonomic managers whose purpose is to maximize utilization using business goals that you have defined. You can monitor performance metrics, analyze the monitored data, offer a plan for executing actions, and have the capability to execute these actions in response to the flow of work.

Overview of request flow prioritization

With Intelligent Management, you can define performance goals and bind them to specific subsets of the incoming traffic. The on demand router (ODR) and associated autonomic managers support business goals in times of high load by making smart workload management decisions about the work that is being sent through the ODR. Not all the work in your configuration is equally important. The ODR can support this concept by forwarding different flows of requests more or less quickly to achieve the best balanced result and maintain the quality of service.

Role of the ODR

The ODR is a server that acts as an HTTP proxy or a SIP proxy. An ODR contains the autonomic request flow manager (ARFM). ARFM prioritizes inbound traffic according to service policy configuration and protects downstream servers from being overloaded. Traffic is managed to achieve the best balanced performance results, considering the configured service policies and the offered load. Note that for an

inbound User Datagram Protocol (UDP) or Session Initiation Protocol (SIP) message, the ODR can route the message to another ODR to properly check for and handle UDP retransmissions.

The on demand configuration (ODC) component allows the ODR to sense its environment. ODC dynamically configures the routing rules at runtime to allow the ODR to accurately route traffic to those application servers. An ODR is able to route HTTP requests to WebSphere Application Server Network Deployment servers, and servers that are not running WebSphere software. The ODR, like the Web server plug-in for WebSphere Application Server, uses session affinity for routing work requests. After a session is established on a server, later work requests for the same session go to the original server, which maximizes cache usage and reduces queries to back-end resources.

Service policies

A service policy is a user-defined categorization that is assigned to potential work as an attribute that is read by the ARFM. You can use a service policy to classify requests based on request attributes, including the URI, the client name and address, and the user ID or group. By configuring service policies, you apply varying levels of importance to the actual work. You can use multiple service policies to deliver differentiated services to different categories of requests. Service policy goals can differ in performance targets as well as importances.

The autonomic request flow manager (ARFM)

The ARFM exists in the ODR and controls request prioritization. When the following components that the ARFM contains are working together, they are able to properly prioritize incoming requests:

- A compute power controller per target cell. That is, a cell to which some ARFM gateway directly sends work. This is an `HAManagedItem` that can run in any node agent, ODR, or deployment manager.
- A gateway per a used combination of protocol family, proxy process, and deployment target. A gateway runs in its proxy process. For HTTP and SIP, the proxy processes are the on demand routers; for Java Message Service (JMS) and IIOP, the proxy processes are the WebSphere application servers.
- A work factor estimator per target cell. This is an `HAManagedItem` that can run in any node agent, ODR, or deployment manager.

Dynamic workload management (DWLM)

Dynamic workload management (DWLM) is a feature of the ODR that applies the same principles as workload management (WLM), such as routing based on a weight system, which establishes a prioritized routing system. DWLM autonomically sets the routing weights to WLM. With WLM, you manually set static weights in the administrative console. With DWLM, the system can dynamically modify the weights to stay current with the business goals. DWLM can be shut off. If you intend to use the automatic operating modes for the components of dynamic operations, then setting a static WLM weight on any of your dynamic clusters could get in the way of allowing the on demand aspect of the product to function properly. The WebSphere Application Server Network Deployment WLM is not limited to the on demand routers, but also applies to IIOP traffic when the client is using the WebSphere Application Server Java Development Kit (JDK) and object request broker (ORB) and *prefer local* routing is not employed.

The following diagram shows an equal amount of requests flow into the ODR. Platinum, gold, and bronze are used to depict a descending order of importance, respectively. After the work is categorized, prioritized, and queued, a higher volume of more important work (platinum) is processed, while a lower volume of less important (bronze) work is queued. Because bronze is delayed, the long-term average rate of bronze coming out of the ODR is not less than the long-term average rate of bronze going in. Dynamic operations keep the work within the target time allotted for completion.

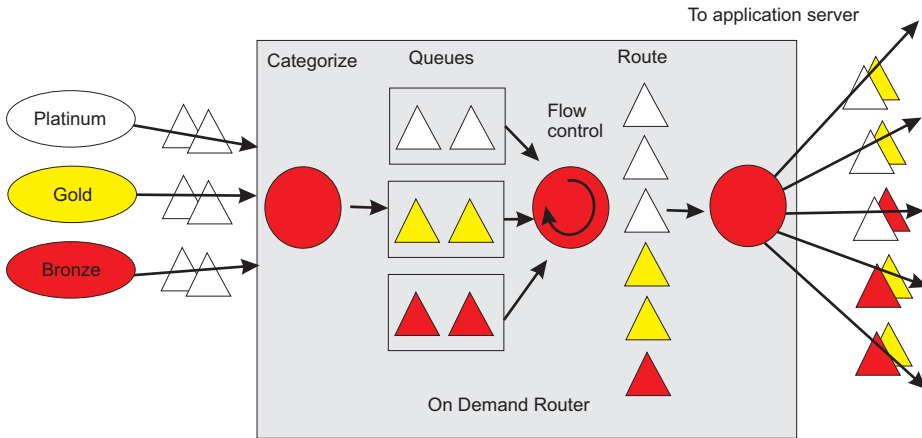


Figure 8. Flow of requests into and through the on demand router

Dynamic operations environment

The Intelligent Management dynamic operations environment supports three modes of operation: manual, supervised, and automatic. This environment is supported by autonomic managers that perform tasks such as application placement, workload management, health management, and request flow management.

Defining the mode of the operating environment involves configuring the Intelligent Management components. In a mixed environment, some components run automatically, or autonomically, others do not run automatically. Intelligent Management has the flexibility to support logical systems, rather than a switch to turn things on or off. In addition, the product supports a supervised operating mode, where the system administrator accepts or denies the recommendations of the autonomic managers.

The following are dynamic operations features, and operating modes they support:

Autonomic manager	Description
Application placement controller	<p>Characteristics:</p> <ul style="list-style-type: none"> Supports two modes: enabled or disabled. <p>If enabled, the application placement controller supports automatic, supervised, and manual mode for each dynamic cluster with which it is affiliated. The mode is set on the dynamic cluster.</p> <ul style="list-style-type: none"> Applicable to a dynamic cluster. Not autonomous functionality, as this is dependent upon the dynamic cluster setting. Is a distributed resource managed by the high availability manager. <p>The application placement controller exists within all node agent and deployment manager processes. The controller is active in one of these processes. If the active process fails, the controller becomes active on another node agent or deployment manager process.</p>
Autonomic request flow manager (ARFM)	<p>Characteristics:</p> <ul style="list-style-type: none"> Supports two modes: manual or automatic. Autonomous functionality, as this can be enabled individually. If you are not in ARFM manual mode, you can set the <code>arfmManualAllocation</code> custom property to disable the autonomic setting of the ARFM control knobs.

Autonomic manager	Description
Dynamic WLM (DWLM)	Characteristics: <ul style="list-style-type: none"> • Supports two modes: enabled or disabled. • Applicable to static clusters and dynamic clusters. DWLM is enabled on dynamic clusters by default, and disabled on static clusters by default. To enable DWLM on a static cluster, click Servers > Clusters <i>cluster_name</i> > Dynamic workload management (DWLM). To disable DWLM on a dynamic cluster, click Servers > Dynamic clusters > <i>dynamic_cluster_name</i> > Dynamic workload management (DWLM). • Autonomous functionality, as this can be enabled individually.
Health controller	Characteristics: <ul style="list-style-type: none"> • Supports two modes: enabled or disabled. • Applicable to a cell, or subset of a cell. • Autonomous functionality, as this can be enabled individually. • Is a distributed resource managed by the high availability manager. The health controller exists within all node agent and deployment manager processes. The controller is active in one of these processes. If the active process fails, the controller becomes active on another node agent or deployment manager process.

Components of dynamic operations

Intelligent Management is built on autonomic computing features in the dynamic operations environment. With these features, the virtualized application server environment can expand and contract according to business demand. Using the autonomic managers in the product environment, dynamic operations make logical decisions based on business goals.

Restriction: Intelligent Management does not support Session Initiation Protocol (SIP) features on the z/OS operating system.

The dynamic operations environment has several components:

Operational policy

An operational policy is a business, or performance objective that supports specific goals for specific requests. Operational policies include service and health policies. A service policy defines a business goal and an importance and contains one or more transaction classes. For a given work class, a rule condition maps to a transaction class which belongs to a service policy. The service policy contains the business goal requirements and the work class contains the work description upon which the service policy is applicable. The combination of these policies is read by the dynamic operations environment to make decisions on HTTP, SOAP, JMS, SIP, and IIOP work requests.

Dynamic clusters

A dynamic cluster is an application deployment target that is expandable and contractible, as needed by the dynamic operations environment. Dynamic clusters provide the core functionality of application infrastructure virtualization. Dynamic cluster instances are created on nodes that meet the criteria of a *membership policy* that you specify when you create the dynamic cluster.

Autonomic request flow manager

The autonomic request flow manager (ARFM) has numerous functions:

- Uses queuing of incoming messages in edge-based gateways to provide computing power overload protection and differentiated service. The computing resource protected from overload is typically CPU power. The differentiated service aims to provide the best balanced performance results among various flows of traffic, relative to the given operational policy and current offered load.

- Can optionally exert dialog/session-oriented admission control for the sake of computing power overload protection.
- Sends information to the placement controller about each cluster to enable the placement controller to optimize the placement for the operational policy and currently offered load. The information about a given cluster is the relationship between computing power and service utility for that cluster.

On demand router

An on demand routers (ODR) is an intelligent HTTP proxy or a SIP proxy. ODRs are the point of entry into a product environment and are gateways through which HTTP requests and SIP messages flow to back-end application servers. It can momentarily queue requests for less important applications in order to allow request from more important applications to be handled more quickly or to protect back-end application servers from being overloaded. The ODR is aware of the current location of a dynamic cluster's server's instances, so that requests can be routed to the correct endpoint. The ODR can also dynamically adjust the amount of traffic sent to each individual server instance based on process utilization and response times. Note that for an inbound SIP/UDP message, the ODR might route the message to another ODR in order to properly check for and handle UDP retransmission.

Dynamic workload manager

The autonomic request flow manager classifies and prioritizes requests to application servers based on the demand and policies. The dynamic workload manager then distributes the requests among the nodes in a dynamic cluster to balance the work.

Application placement controller

The application placement controller is an autonomic manager in the dynamic operations infrastructure that supports application virtualization, or the fluid mobility of applications within a dynamic cluster. The application placement controller adds application server instances when the work is more than can be handled by the current application, and stops application servers when there are too few requests for the number of started applications.

Health controller

The health controller constantly monitors the defined health policies. When a condition specified by a health policy is not met in the environment, the health controller assures that the configured actions are taken to correct the problem.

EWLM

The enterprise workload manager (EWLM) manages sub-goals and resource allocations for the larger environment which contains Intelligent Management.

Operational policies:

Intelligent Management uses service level management and policy-driven goals to achieve a healthy and robust environment with a high quality of service. Operational policies consist of service policies and health policies. With service policies, you can differentiate between applications according to their perceived level of importance and target values. With health policies, you can identify conditions to watch for and the product acts on these conditions to ensure a healthy environment.

Service policy

Service policies and, for most kinds of work requests, work classes are used to categorize and prioritize work requests. A service policy consists of a user-defined performance goal and (in some cases) an importance level. Service policies are related to work requests through transaction classes. Each work request belongs to exactly one transaction class, and each transaction class belongs to exactly one service policy. For most kinds of work requests, work classes are used to map incoming requests to transaction classes. Each work class is attached to a Java Platform, Enterprise Edition (Java EE) application and a basic request feature; Uniform Resource Identifier (URI) prefix for HTTP, method name for Internet Inter-ORB Protocol. (IIOP), and bus and destination for Java Message Service (JMS). Each

work class specifies how the relevant requests are classified into transaction classes. For generic server clusters, and for Session Initiation Protocol (SIP), work classes are not used; instead, the rules for classifying requests to transaction classes are configured on the on demand routers.

A health policy, in contrast to a service policy, is the definition of specific health criteria that you want your product to protect itself against. The health management function uses the defined policy to search the environment for software malfunctions.

Different service policies can have different kinds of goals. The discretionary goal has no associated value or importance. An average response time goal has an associated response time threshold and importance, whereas a response time percentile goal has two associated values; percentage and time, and an importance.

The performance management done by the autonomic request flow manager, the dynamic workflow manager and the application placement controller achieves a defined balance of the performance results. The defined balance among nondiscretionary flows is achieved by either having the flows all under threshold by the same relative amount, expressed as a fraction of the threshold, or by exceeding the threshold by a relative amount that is inversely proportional, $100 - \text{importance}$. The discretionary flows are given a minimal allocation.

A performance goal strategy requires a monitoring capability by the autonomic manager to determine whether specified performance goals are met, and a reporting capability to notify a provisioning module when changes are required. Furthermore, to account for the case when it is impossible to satisfy all performance goals, it is possible to assign a business value to each performance goal. Administrators must have an in-depth understanding of deployed applications so that they can create realistic performance goals.

Health policy

A health policy works much the same as the service policy, except that the health policy provides a health goal for the environment. A health policy consists of a *health condition* and a *health action*. A health condition specifies a problematic scenario in your environment. If this scenario occurs, the specified health action runs to make the condition better. You can specify a health policy to monitor different targets, such as a dynamic cluster or a server. Servers can be simultaneously monitored by multiple health policies. Armed with a set of conditions to look for, Intelligent Management monitors the environment until a problem is detected, and action is taken.

Elasticity mode

Use elasticity mode to add logic that causes the application placement controller to minimize the number of nodes that are used, as well as remove nodes that are not needed, while still meeting service policy goals. Additionally, you can use elasticity mode to add logic so that, when the controller recognizes that a particular dynamic cluster is not meeting service policies and has started all possible servers, the controller adds a node.

Overview

Elasticity mode enables a WebSphere cell to grow or shrink dynamically by adding or removing nodes. An elasticity operation defines the runtime behaviors to monitor, and the corrective actions to take when the behaviors are present. As part of the steps for configuring elasticity mode, you create custom actions to define the actions that are associated with the elasticity operations: the add operation and the remove operation. The add operation is issued when all of the resources of the application placement controller are being used, but more resources are still needed to meet the current demand. The remove operation is issued when the application placement controller has an excessive amount of resources.

If the elasticity mode is disabled, dynamic clusters start and stop cluster members in the following situations:

- Servers are started to:
 - Maintain minimum active instances.
 - Meet the CPU or memory demand for a cluster.
- Servers are stopped to:
 - Ensure that the maximum number of instances is not exceeded.
 - Meet the CPU or memory demand for a cluster.
 - Stop cluster members if lazy start or proactive idle stop (custom property) is enabled.
 - Balance resources and make them available to another cluster.

With elasticity enabled, additional options are:

- Increase in demand: Define custom operations (for example, `wasadmin` scripts) to expand a dynamic cluster.

Note: For IBM Workload Deployer or WebSphere Application Server Hypervisor Edition Intelligent Management Pack, predefined tasks add virtual machines and federating nodes to decrease the capacity of a dynamic cluster.

- Decrease in demand: Define custom operations (for example, `wasadmin` scripts) to contract a dynamic cluster.

Note: For IBM Workload Deployer or WebSphere Application Server Hypervisor Edition Intelligent Management Pack, predefined tasks remove virtual machines and federating nodes to decrease the capacity of a dynamic cluster.

Add operation

When elasticity mode is enabled, the application placement controller issues an add operation when all of the members of a dynamic cluster cannot meet the current demand. The controller attempts to consolidate and start all of the servers on the minimum number of nodes as possible.

When the action associated with the add operation is complete, the controller starts a server on the new node. The new node must be added as a member of the dynamic cluster that requested the addition. If a new node is not added, the controller continues to issue the add operation until all required resources are received or the demand decreases.

Remove operation

The remove operation first stops any started dynamic cluster instances before starting the associated actions. It is important to know that if the dynamic cluster is set to manual mode, the remove operation is issued on any nodes that do not have any started application servers. When elasticity mode is enabled and a node is no longer required to meet the current demand, the application placement controller issues a remove operation. Any nodes that are not part of any dynamic clusters with no running application servers are first removed. Next, an attempt is made to remove a node that does contain a dynamic cluster instance as long as the instance is not running and no other application servers are running. Finally, an attempt is made to remove nodes that only have one or more started dynamic clusters. The remove operation occurs only if that node is not required to meet the minimum number of instances for a dynamic cluster, or is not required to meet the current demand.

When the application placement controller is running without elasticity mode enabled, the controller issues start and stop operations for application servers. Servers are started due to an increased demand for CPU or memory, but the servers are not stopped after they are started. When elasticity mode is enabled and the servers are not needed, however, the stop operation is issued and the servers are stopped even after

they are started. After all of the servers on the physical machine or virtual machine are stopped, the application placement controller issues the remove operation.

Considerations when using elasticity mode

Consider the following information when using elasticity mode.

- The application placement controller will not issue a remove operation on any node that contains a deployment manager or a stand-alone on demand router (ODR).
- Do not enable application lazy start with elasticity mode. The application placement controller issues the remove operation for all of the nodes on that dynamic cluster. In certain environments, this might cause problems, because all of the custom nodes can then be lost.
- You must configure the application placement controller to always start on the deployment manager or node that will not be removed. Doing so prevents the controller from issuing a remove operation for the node that the controller is active on. If you do not configure the controller to start on the deployment manager, an attempt to remove the node in which the controller is running might occur. As a result, data can be lost, any actions defined by the remove operation do not occur, and the runtime tasks in the administrative console are not properly updated.
- When you use elasticity mode in an environment in which multi-cell performance management is configured, you must configure certain controllers to start on the deployment managers of the center cell and the point cells. Configure the application placement controller to start on the deployment manager of the center cell. Configure the cell agent to start on the deployment managers of the point cells.
- Change the `minTimeBetweenPlacementChange` custom property from 15 minutes to 3 minutes to ensure that the application placement controller does not wait too long to issue an add operation. If the default value of 15 minutes is used, the controller might issue two add operations over a period of 30 minutes.

Considerations when using elasticity to manage JMS traffic that originates from WebSphere MQ

- By default, the application placement controller (APC) uses information produced by the ODR to determine when to start or stop application servers in a dynamic cluster. Set the `APC.predictor` custom property to `CPU` to remove the APC dependency on ODR input. This enables Intelligent Management to support dynamic clusters of message-driven beans when loaded by WebSphere MQ.
- When you use elasticity mode to manage JMS traffic that originates from WebSphere MQ (Version 7.0.1.6 or later is required), go to **System administration > Cell > Custom properties > New**, and set the cell custom property `JMS.CPU` to `true`. Restart the cell.

Runtime operations

Use runtime operations to configure the dynamic operation environment and the visualization capability to understand the operational state of the environment.

Use runtime operations in the administrative console to visualize the inner workings of the environment. You can see what types of decisions are made regarding deployment of applications and allocation of hardware. You can also discover where in a virtual resource pool the applications are running to facilitate debugging, manageability, and resilience. Use runtime operations to view the current state of the environment and take administrative actions. For instance, you can override decisions made by the provisioner, or modify application priority and availability through the operations console.

Runtime operations are divided into several functions. With the runtime operations navigational group, you can view the status of the Intelligent Management environment and create reports to view the performance of your environment over time. With the task manager, you can approve and deny system suggestions through a one-click option.

Important: JavaScript must be enabled in your browser for the runtime operations panels to work properly.

Supported middleware server types

You can create representations of servers that run other middleware software and then manage the servers.

Middleware servers encompass all servers in the middleware tier that provide the infrastructure for applications or their data. The steps vary depending on if you are configuring assisted lifecycle management middleware servers or complete lifecycle management middleware servers.

With assisted lifecycle servers, you use templates to create representations of the servers. However, these servers still exist within the administrative domain of their respective middleware platform.

The following server types are supported with assisted lifecycle middleware server support through the administrative console and `wsadmin` scripting. You can control the servers operationally, monitor, and view server health and performance, and configure the administrative console to display log files and configuration files for these servers.

- Apache HTTP Server Version 1.3, Version 2.0, and Version 2.2
- Apache Geronimo Version 1.0 and Version 1.1
- WebSphere Application Server Community Edition

The following server types are supported with assisted lifecycle middleware server support through `wsadmin` scripting.

- Apache Tomcat Version 4.1.x, Version 5.0.x, Version 5.5.x, Version 6.0
- JBoss Version 4.0.x
- BEA WebLogic Server Version 8.x, Version 9.x, Version 10.x
- External WebSphere application servers Version 5.1 and later

Complete lifecycle servers include any servers that the environment can instantiate or create. You can control the servers operationally, deploy applications, and monitor and view server health and performance. The following server types are supported with complete lifecycle middleware server support through the administrative console and `wsadmin` scripting:

- WebSphere Application Server Community Edition Version 2.x, Version 3.x
- PHP Version 4.x and Version 5.x

To support any other server that supports HTTP traffic, you can create a custom HTTP server representation of the server in the administrative console.

Middleware nodes and servers

The term *middleware server* refers to a server on any middleware platform. Middleware servers include the following types: WebSphere Application Server, Apache Tomcat servers, JBoss servers, BEA WebLogic servers, PHP servers, and so on.

Support

Intelligent Management offers enhanced support for environments outside of the product domain. Application servers that run on other middleware platforms are represented more thoroughly in the product administrative domain because the node agent is installed on these machines. The application placement controller can manage dynamic clusters that are made up of these server types. Some health management is also supported for the other middleware platform servers.

Middleware server types

- **Complete lifecycle management servers**

Complete lifecycle management servers include servers that the product can both create and manage to support application server virtualization. The following managed middleware server types are supported:

- WebSphere Application Server related servers, including application servers, on demand routers (ODR), proxy servers, and so on
- PHP servers
- WebSphere Application Server Community Edition 2.0 (all releases)

All servers that are administered by complete lifecycle support must be managed from the administrative console. For example, do not manage complete lifecycle WebSphere Application Server Community Edition servers from the WebSphere Application Server Community Edition console.

Create complete lifecycle WebSphere Application Server Community Edition servers from previously unused installations of WebSphere Application Server Community Edition Version 2.x or Version 3.x.

- **Assisted lifecycle management servers**

Assisted lifecycle management servers include servers that Intelligent Management can manage, but are created outside of the Intelligent Management administrative domain, usually specific to the system with which the server is associated. Install the node agent on these hosts and associate them with the Intelligent Management administrative domain. On configured hosts, Intelligent Management can then start and stop the servers, route traffic to the servers, prioritize requests to the servers, and so on. Intelligent Management supplies templates for the following assisted lifecycle middleware server types:

- Apache HTTP Server Version 1.3, Version 2.0, and Version 2.2
- Apache Tomcat Version 4.1.x, Version 5.0.x, Version 5.5.x, and Version 6.0
- Apache Geronimo Version 1.0 and Version 1.1
- JBoss Version 4.0.x
- BEA WebLogic Server Version 8.x, Version 9.x, Version 10.x
- WebSphere Application Server Community Edition
- External WebSphere application servers, Version 5.1 and later. External WebSphere application servers are application servers that run outside of your Intelligent Management cell.
- Custom HTTP servers

For any other server types, you can develop your own templates.

Administer assisted lifecycle servers from their respective administrative consoles. For example, assisted lifecycle WebSphere Application Server Community Edition servers are administered from the WebSphere Application Server Community Edition console.

For Apache Tomcat, BEA WebLogic Server, JBoss, or External WebSphere application servers, any representational revisions must be entered using `wsadmin` scripting. For other assisted lifecycle management servers, any representational revisions must be entered into the administrative console.

- **Discovered servers**

Discovered servers are servers that are found by middleware discovery. Middleware discovery can find existing installations of WebSphere Application Server Community Edition and create representations of these servers in the administrative console. These servers are represented as assisted lifecycle servers. You can group these servers together into a dynamic cluster, but the dynamic clusters must have manual membership. You cannot create expression-based dynamic clusters of discovered servers. Middleware discovery can also find the applications that are deployed to these servers and represent them in the administrative console as unmanaged applications.

Like assisted lifecycle servers, administer discovered servers from their respective administrative consoles. For example, administer WebSphere Application Server Community Edition servers from the WebSphere Application Server Community Edition console. Make any representational revisions in the administrative console.

Definition

The Intelligent Management support is specific for each of the following server categories: complete, discovered, and assisted lifecycle servers.

- For complete lifecycle management middleware servers, define the servers in the administrative console. Management of these servers is the same as in previous releases.
- For assisted lifecycle middleware servers, use the following approach to register the servers with the Intelligent Management configuration:

Install the node agent on the server and manually define the server in the administrative console. Use *server templates* to define the middleware servers. Server templates include the following information:

- Default ports for routing traffic to the server
- Commands for starting and stopping the server
- Default list of core configuration files that make up the external server for the external configuration editing service
- Default list of directories where log files exist for the external log viewing service

Intelligent Management includes a set of default server templates for the different middleware server types.

- For discovered servers, after you install the node agent, middleware discovery locates existing installations of WebSphere Application Server Community Edition and their installed applications, and creates a representation of these servers and applications in the Intelligent Management cell. These servers and applications are represented as assisted lifecycle servers and unmanaged applications.

Management

You can manage certain aspects of assisted lifecycle and discovered middleware servers with the administrative console.

- With *server operations*, you can run a Java or non-Java executable file on your middleware server from the administrative console.
- With the *log viewer*, you can view the log files for your middleware servers from the administrative console.
- With *external configuration*, you can configure the administrative console so that you can view and edit configuration documents for your middleware servers.

Other Intelligent Management management functions, such as dynamic clusters, health policies, service policies, runtime tasks, and reporting, offer varied support for assisted lifecycle and discovered middleware servers.

Migrating from WebSphere Virtual Enterprise

Migrating involves copying the configuration from a previous release of WebSphere Virtual Enterprise into WebSphere Application Server Network Deployment Version 8.5.

About this task

You can perform an incremental migration of your WebSphere Virtual Enterprise Version 6.1.0.x, Version 6.1.1.x, or Version 7.0 configuration to WebSphere Application Server Network Deployment Version 8.5. As a result, your system temporarily operates in a mixed cell environment in which you migrate the deployment manager to WebSphere Application Server Network Deployment Version 8.5 before you migrate any nodes.

Note: Even though the method of incremental migration creates a mixed cell environment, avoid running the mixed cells for an extended period of time. You should have an existing plan in place to migrate your entire cell to the highest level to ensure consistent administration of the nodes.

Procedure

1. Migrate the deployment manager to WebSphere Application Server Network Deployment Version 8.5. The deployment manager must be at the highest release and fix level within the cell.
2. Migrate the on demand routers (ODR) one at a time to WebSphere Application Server Network Deployment Version 8.5. Each ODR must be at the same release and fix level as the deployment manager.
3. Migrate the application server nodes one at a time.

Migrating to WebSphere Application Server Version 8.5 Intelligent Management cells

Use the **VEUpgrade** command to migrate a previous release of WebSphere Virtual Enterprise to a WebSphere Application Server Version 8.5 Intelligent Management cell.

Before you begin

Update WebSphere Application Server Network Deployment Version 8.5.x to the latest maintenance level that is available. For more information about the latest fix packs, see the IBM Support Portal.






gotcha: If you run **VEUpgrade** with no parameters to get command line help, existing migration logs are overwritten. To retain the migration logs, copy the logs to a backup directory.

About this task

First migrate the deployment manager. Then migrate each application server node one at a time.

Note: If you are migrating a WebSphere Virtual Enterprise Version 6.1.1.x or 7.0.0.x cell with middleware agents to WebSphere Application Server Network Deployment Version 8.5.x, the middleware agents are not migrated. The middleware agents remain at the previous version. You can continue to use the middleware agents and use IBM Update Installer to install fix packs to the middleware agents separately.

Procedure

1. Run the **backupConfig** command on the deployment manager and all nodes. The **backupConfig** command is a utility that is used to back up the configuration of the deployment manager and the nodes.
2. Install WebSphere Application Server Network Deployment Version 8.5.x on each deployment manager and node. Specify a different installation location than the location in which the previous version exists. Read about installing the product and additional software in the information center.
3. Use the Profile Management Tool or the **manageprofiles** command to create your profiles for Version 8.5.x, but do not federate the nodes. Read about managing profiles.
4. Migrate the deployment manager to your Version 8.5.x cell. You can use the **WASPreUpgrade** and **WASPostUpgrade** commands. The **WASPreUpgrade** command saves the previous configuration to a migration-specific backup directory, and the **WASPostUpgrade** command retrieves the saved configuration. Read about the **WASPreUpgrade** and the **WASPostUpgrade** command.
5. Use the **VEUpgrade** command to migrate the deployment manager to Version 8.5. To migrate the product using the **VEUpgrade** command, run the following script:
 -     target_profile/bin/VEUpgrade.sh
 -  target_profile\bin\VEUpgrade.bat
6. Migrate the remaining nodes one at a time. The procedure to migrate each node is similar to the preceding procedure, except WebSphere Application Server Network Deployment already exists in your environment, and you create custom profiles instead of a deployment manager profile.

- a. Use the Profile Management Tool or the **manageprofiles** command to create a WebSphere Application Server Network Deployment custom profile.
- b. Migrate the node to your WebSphere Application Server Network Deployment Version 8.5.x cell. Run the **WASPreUpgrade** and **WASPostUpgrade** commands.
- c. Migrate the node to Version 8.5. Run the **VEUpgrade** command.

Results

Your cell is now at Version 8.5 level.

VEUpgrade command

Use the **VEUpgrade** command to migrate product configuration data from an existing WebSphere Virtual Enterprise Version 6.1.0.5, Version 6.1.1, or Version 7.0 cell to a new Intelligent Management Version 8.5 cell.

Location

Run the command from the `target_profile_root/bin` directory.

Parameters

-userID

Specifies the administrative user name for authentication if security is enabled on the workstation. Because the migration utility does not check for the **-userID** parameter, you must specify it or the migration process fails. (Required)

-password

Specifies the administrative password for authentication if security is enabled on the workstation. Because the migration utility does not check for the **-password** parameter, you must specify it or the migration process fails. (Required)

-sourceWasHome

Specifies the `install_root` directory of the existing product installations from which the configuration is migrated. (Required)

-sourceProfilePath

Specifies the fully qualified path of the existing profile from which the configuration is migrated. (Required)

-targetProfilePath

Specifies the fully qualified path to the target profile. (Required)

-traceLevel

Specifies the trace level for generating diagnostic messages. (Optional)

-traceDir

Specifies the directory of the `XDUUpgrade.log` and `XDUUpgrade.trace` files. The default directory is `was_home/logs`. (Optional)

Usage

The following example migrates the deployment manager to the new cell:

```
VEUpgrade.sh -userid user_name -password user_password
-sourcewashome install_root/DeploymentManager -sourceprofilepath install_root/DeploymentManager/profiles/default
-targetprofilepath target_profile_path/DeploymentManager/profiles/default
```

The following example migrates an application server to the new cell:

```
VEUpgrade.sh -userid user_name -password user_password
-sourcewashome install_root/AppServer -sourceprofilepath install_root/AppServer/profiles/default
-targetprofilepath target_profile_path/AppServer/profiles/default
```

Migrating large topologies of external WebSphere application servers

You can manage previous versions of WebSphere product application servers in the Intelligent Management administrative console. If you have a large topology and want to automate a migration from Version 6.1, 7.0, or 8.0 to Version 8.5, you can use the migration toolkit to detect your configurations and import them into a Version 8.5 cell.

Before you begin

- Install and federate your Version 8.5 middleware agents and deployment manager on your previous version machines. For the purposes of this task, the previous version deployment manager is referred to as the *previous deployment manager*. If you do not have the middleware agent running on your previous deployment manager, you can copy the `AGENT_HOME/lib/legacycell.jar` file and `AGENT_HOME/bin/Extractor.sh|.bat` file from another workstation that has the middleware agent deployed to the previous deployment manager.
- Install or augment your deployment manager for Version 8.5.

About this task

The migration toolkit can detect previous WebSphere configurations and import them into a Version 8.5 cell.

The migration toolkit uses scripts to connect to one cell, read the data, and import the data into the other cell. The scripts also function as an adaptor or integration interface. The migration script logic handles the routing information and environment detection. The scripts also perform integrity checks to ensure a stable configuration.

For more information read about the migration toolkit.

Procedure

1. Generate the mappings file. The mappings file defines additional data that cannot be gathered automatically. You must generate the mappings file before you import the middleware deployment.
 - a. Log in to the previous deployment manager computer.
 - b. On the deployment manager computer, go to the `AGENT_HOME/bin` directory, or the directory to which you copied the middleware agent files from another workstation.
 - c. Run the `Extractor.sh` or `Extractor.bat` script. For example, run the following command:

```
./Extractor.sh app_server_root/config
```

The single `install_root` directory is the default for the deployment manager profile directory. If `install_root` does not specify the deployment manager profile directory as the default, add the following argument:

```
app_server_root/profiles/deployment_manager_profile/config
```

When this script completes, a `Mapping.dat` file is in the directory from which you ran the script.

- d. Copy the `Mapping.dat` file to the `app_server_root/bin` directory on your Intelligent Management Version 8.5 deployment manager.
2. Run the automation script.
 - a. Log in to your Intelligent Management Version 8.5 deployment manager computer.
 - b. On the deployment manager computer, navigate to the `app_server_root/bin` directory.
 - c. Run the `MirrorCell.bat` or `MirrorCell.sh` script. The `MirrorCell` script uses the following format:

```
MirrorCell.* [-props PROPS] [-mode MODE] [-nosec, -nosave]
```

The only required parameter is the `-props` parameter, which specifies the full path to the properties file. For more information about the `MirrorCell` script parameters, read about the `MirrorCell.bat` or `MirrorCell.sh` script.

Note: If you include paths that contain spaces in your script call, surround these paths with quotation marks, for example:

```
"C:\Program Files"
```

By default, the script connects to the previous deployment manager and generates any necessary delta information between the newly detected configuration and a copy of the last known configuration. The script then uses this delta information to recreate the configuration within the Version 8.5 cell. Status updates are reported at each milestone.

- d. Log in to the Intelligent Management cell and verify that the configuration was created properly. To check that your servers are represented in the console, click **Servers > Other middleware servers > External WebSphere application servers**. To check that representations of your applications were created, click **Applications > All applications**.

What to do next

You can manage your previous configurations from the Intelligent Management cell.

Migration toolkit:

You can use the migration toolkit to automatically detect your previous configurations of WebSphere Application Server and import the configuration into the Intelligent Management cell.

Toolkit capabilities

- **Bidirectional and atomic automation:** With the migration toolkit, you can read from an older cell configuration and create the Version 8.5 configuration in an atomic process. You do not need to manually make any updates after the migration process begins.
- **Multiple migration modes:** You can also phase your migration. For example, you can read the data from the previous cell and manipulate the data before continuing with the migration.
- **Delta generation:** The migration toolkit scripts persist a copy of the previously generated environment data. When you rerun the scripts, the scripts create a delta between the configuration versions and can update and delete servers or applications as needed. Only major configurations such as servers, applications, and modules are factored into the delta. If port or endpoint information changes, for example, you must modify the data directly from the administrative console.
- **Automatic data entry:** The migration toolkit scripts automatically preload data into your cell.
- **Limited dynamic clustering:** After the scripts complete, your static clusters are represented as dynamic clusters. Dynamic clusters can start and stop servers as your service policies or demand requires. The dynamic clustering capabilities for your migrated servers are limited. The limitations that apply to dynamic clusters of assisted life cycle middleware servers also apply to the dynamic clusters that the migration toolkit creates.
- **HTTP traffic shaping:** Because the applications are registered within the Intelligent Management configuration, the on demand router (ODR) can route to your clustered applications based on the demand and statistical analysis of your HTTP traffic. Traffic can be routed to your Version 6.1.x applications in the same way that traffic can be routed to Version 8.5 and later applications. Service and routing policies can be applied.

Functional limitations

- The representations of your servers receive server status from the middleware agent. The middleware agent is installed on your previous server and provides the status information. However, this status might not be accurately represented at all times.
- You cannot fully monitor middleware applications from your Version 8.5 cell. The status of the applications is tied to the status of the target servers or clusters with which they are associated. When the corresponding servers or clusters start, so do the middleware applications, independent of the actual application status.

Preparing the hosting environment for dynamic operations

Dynamic operations monitor the middleware server environment and make recommendations or changes based on the data that is observed. To prepare the hosting environment for dynamic operations, federate middleware nodes, create the on demand router (ODR), create middleware servers and dynamic clusters, and deploy applications to your servers and dynamic clusters with an associated service policy.

About this task

To set up the dynamic operations environment, create ODRs, servers and dynamic clusters. These actions set up the application server virtualization portion. You can then deploy your applications and define service policies, to configure application virtualization. The autonomic managers work with these elements to maximize the performance of your environment.

Procedure

1. Federate middleware nodes.
2. Create and configure ODRs.
3. Create and configure dynamic clusters.
4. Add middleware servers to the environment.
5. Deploy and manage applications through service policies.

Configuring VMware Infrastructure 3 platforms and Intelligent Management

To configure Intelligent Management to work with VMware Infrastructure 3 platforms, you must configure security so that the servers can communicate with each other and configure custom properties on your deployment manager to define the vCenter or ESX servers.

Before you begin

- Configure the VMware Infrastructure 3 platforms environment on your physical servers. Your VMware Infrastructure 3 platforms environment must meet the following requirements:
 - Your VMware Infrastructure 3 platforms environment must be on servers that are running Solaris Operating Environment on Intel hardware, Windows, or Linux x86 operating systems.
 - You must use VMware products that support VMware Infrastructure 3 platforms. The supported versions are:
 - VMware VirtualCenter Version 2.5
 - VMware ESX Version 3.5
 - VMware vSphere Version 4.0 and Version 4.1, both of which include VMware ESXi and VMware vCenter Server

The documentation generically refers to these servers with the following terminology:

- **ESX server:** Refers to VMware ESX Version 3.5 or a VMware ESXi server in VMware vSphere Version 4.0 and Version 4.1.
- **vCenter server:** Refers to VMware VirtualCenter Version 2.5 or a VMware vCenter server in VMware vSphere Version 4.0 and Version 4.1.

About this task

When you have multiple nodes running on a physical computer with VMware Infrastructure 3 platforms, Intelligent Management can contact VMware through Web services. You can configure this communication in the administrative console by creating cell-wide custom properties. These custom properties define the URL, user ID, and password for the vCenter or ESX servers. You also must configure your key stores to retrieve signers from the vCenter or ESX servers.

The Intelligent Management configuration depends on your VMware configuration. Create the custom properties for the servers in your environment so that Intelligent Management can monitor all of the virtual machines and physical computers. To set the custom properties on the cell level, click **System administration > Cell > Custom properties**.

- If you are using only ESX servers, you must configure enough of the individual servers to make Intelligent Management aware of the physical servers and virtual machines in the environment.
- If you are using a vCenter server to manage your environment, you can connect to the vCenter server, which establishes communication with all of the virtual machines and servers that the vCenter server manages. You do not need to connect to each ESX server. If a vCenter is available, the best practice is to connect to the vCenter server instead of each ESX server.
- If you are running multiple vCenter servers with a Microsoft Cluster Server (MSCS) to provide high availability, you can configure the key stores and custom properties for each vCenter server.

If you do not configure Intelligent Management to work with VMware Infrastructure 3 platforms, the Intelligent Management environment does not understand that the nodes are on virtual machines, and as a result, the machine processor or memory might be overloaded.

Procedure

• If you are configuring Intelligent Management to communicate with a vCenter server:

1. Retrieve and store a signer certificate from the vCenter server and configure Intelligent Management to communicate with the vCenter server:

```
./wsadmin.sh -lang jython -f retrieveVMwareCertificate.py
  -host:<vmware_virtual_center_host_name> -port:<vmware_virtual_center_ssl_port_number>
  -user:<vmware_user_id>
  -password:<vmware_password>
```

Where *<vmware_virtual_center_host_name>* is the host name of the vCenter server, *<vmware_virtual_center_ssl_port_number>* is the secure SSL port of the vCenter server, *<vmware_user_id>* is the VMWare userid that is used to access the vCenter server, and *<vmware_password>* is the password associated with the *<vmware_user_id>*.

• If you are configuring Intelligent Management to communicate with ESX servers:

1. Retrieve and store a signer certificate from the ESX server and configure Intelligent Management to communicate with the ESX server.

```
./wsadmin.sh -lang jython -f retrieveVMwareCertificate.py
  -host:<vmware_esx_server_host_name> -port:<vmware_esx_server_ssl_port_number>
  -user:<vmware_user_id>
  -password:<vmware_password>
```

Where *<vmware_esx_server_host_name>* is the host name of the ESX server, *<vmware_esx_server_ssl_port_number>* is the secure SSL port of the ESX server, *<vmware_user_id>* is the VMWare userid that is used to access the ESX server, and *<vmware_password>* is the password associated with the *<vmware_user_id>* value. For the **-host** parameter, use the host name and not the IP address.

2. Repeat the previous step for all of your ESX servers by using the script to retrieve and store a signer certificate for each ESX server.

Results

By configuring Intelligent Management to work with vCenter or ESX, you obtain better service differentiation management results than by using vCenter or ESX alone. With Intelligent Management, you can add application-level goals and characteristics, so that the autonomic managers can perform the necessary flow control in your virtualized environment.

What to do next

If timeout errors occur, you can increase the `com.ibm.websphere.webservices.http.connectionTimeout` and `com.ibm.websphere.webservices.http.SocketTimeout` custom property values from the default of 300 seconds to 600 seconds. Consider making this change when you have a virtualized environment with a large number of physical and virtual machines. For example, if your environment has 400 physical machines, when requests are sent from Intelligent Management to the hypervisor for configuration information, the hypervisor contacts each of the 400 physical machines. If each request takes 1 second to complete, the default timeout of 300 seconds is not long enough to process all of the requests, and a read timeout results. For more information about the custom properties, read about HTTP transport custom properties for Web services applications.

Configure middleware servers on your WebSphere nodes.

VMware Infrastructure 3 platforms and Intelligent Management

Intelligent Management uses the VMware Infrastructure SDK (VI SDK) to communicate through Web services with VMware Infrastructure 3 platforms. Any VMware Infrastructure 3 platforms that expose the VMware Infrastructure SDK (VI SDK) as a Web service and can work with Intelligent Management, such as ESX or vCenter. VMware software provides operating system level virtualization, while Intelligent Management provides virtualization at an application level.

Supported VMware releases

The supported versions include:

- VMware VirtualCenter Version 2.5
- VMware ESX Version 3.5
- VMware vSphere Version 4.0, which includes VMware ESXi and VMware vCenter Server

The documentation generically refers to these servers with the following terminology:

- **ESX server:** Refers to VMware ESX Version 3.5 or a VMware ESXi server in VMware vSphere Version 4.0.
- **vCenter server:** Refers to VMware VirtualCenter Version 2.5 or a VMware vCenter server in VMware vSphere Version 4.0.

Intelligent Management and virtualization technologies

When you use virtualization technologies with Intelligent Management, a node is no longer a representation of a physical computer. Because several virtual machines can be created on a single physical computer, you can create multiple Intelligent Management nodes on the same physical computer.

Intelligent Management communicates with VMware Infrastructure 3 platforms that expose the VMware Infrastructure SDK (VI SDK) as a Web service. With this integration, the runtime environments can communicate with each other through Web services. By using the SDK, Intelligent Management can make decisions about the placement of servers with the knowledge about where nodes are running in the cell. This integration is only supported if you are using VMware Infrastructure 3 platforms.

VMware Infrastructure 3 platforms and Intelligent Management function

Intelligent Management is aware that nodes are not necessarily distinct, standalone resources. In some cases, a node might be sharing memory and processor space with other nodes, with parameters specified that control and manage the systems.

When you have multiple nodes running on a physical computer with VMware Infrastructure 3 platforms, Intelligent Management can communicate with the VMware product through Web services. You can configure this communication in the administrative console by creating cell-wide custom properties. These

custom properties define the URL, user ID, and password for a vCenter or ESX server. You configure the custom properties with information about all of the virtual machines and physical servers in your environment. For example, if you are running vCenter, you can create a custom property for the vCenter server.

Virtual machines and nodes are correlated by Media Access Control (MAC) addresses.

By configuring Intelligent Management to know about the vCenter or ESX servers, the application placement controller can use the VMware settings to properly set the maximum processor utilization limits for each node. Intelligent Management sets the processor utilization limit for a given virtual machine to be the minimum of either the processor utilization limit that is configured in vCenter or ESX, or the number of processors that are assigned to the virtual machine divided by the actual number of processors on the physical workstation. For example, if the physical computer has 3 processors, and the virtual machine is assigned 1 processor, then the processor utilization limit for the virtual machine is 33% of the total processor on the physical workstation.

Benefit of using Intelligent Management in addition to vCenter or ESX

Although VMware Infrastructure 3 platforms provide service differentiation and management, you cannot control performance at an application level. Intelligent Management adds the capability to define application level goals. The autonomic managers work to maintain the defined goals.

Sample topology

The following sample topology shows how multiple virtual machines can be created on a single physical workstation. In this topology, the deployment manager and on demand router (ODR) are on the same physical workstation, but two different virtual machines.

Cell

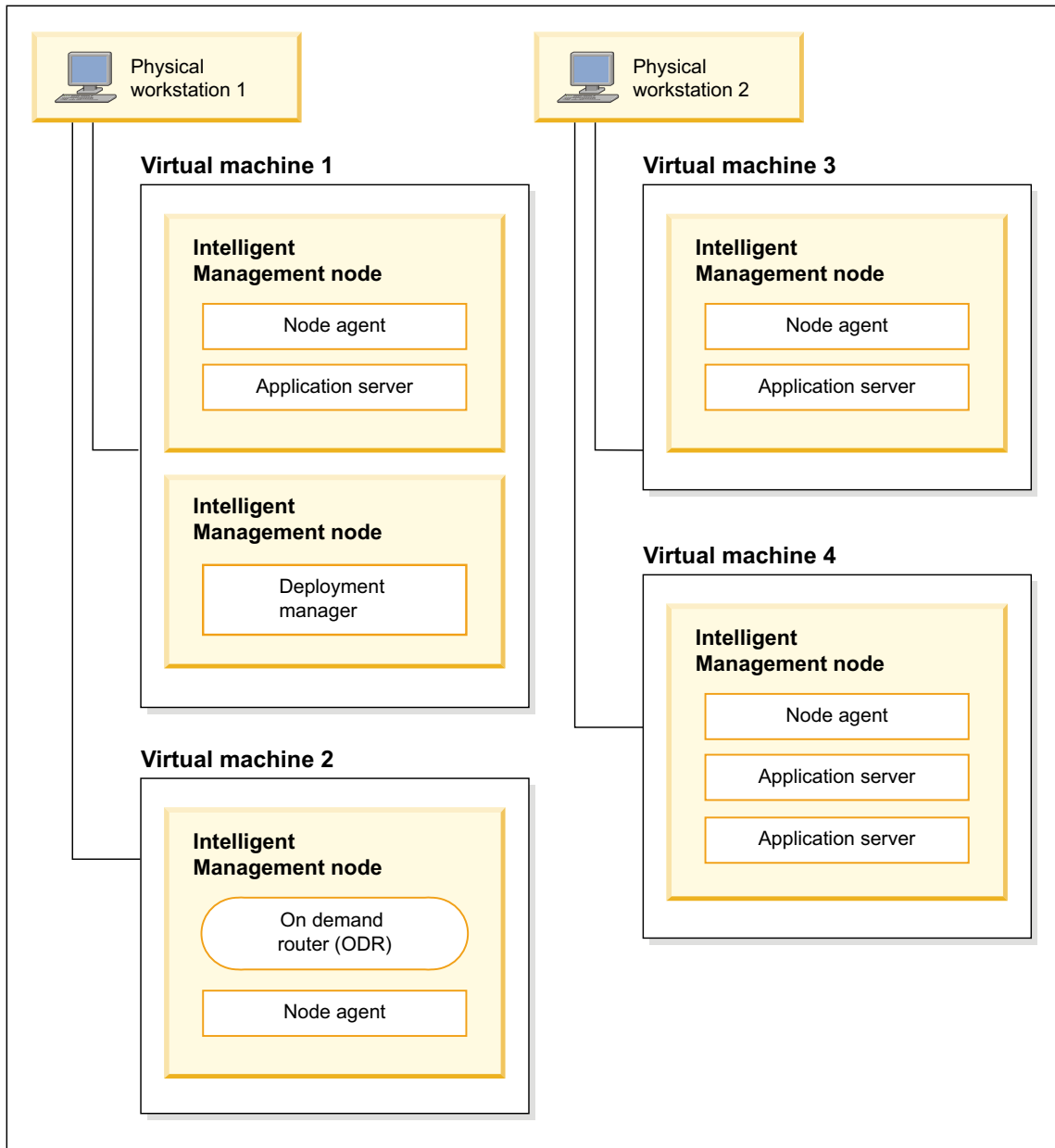


Figure 9. Virtual machines integrated into a Intelligent Management environment

Advanced configuration for VMware Infrastructure 3 platforms and Intelligent Management

The `retrieveVMwareCertificate.py` script can complete all of the steps that are needed to configure VMware Infrastructure 3 platforms and Intelligent Management. However, you can also complete these steps manually by creating the signer certificate and required custom properties in the administrative console.

Before you begin

- Configure the VMware Infrastructure 3 platforms environment on your physical servers. Your VMware Infrastructure 3 platforms environment must meet the following requirements:

- Your VMware Infrastructure 3 platforms environment must be on servers that are running Solaris Operating Environment on Intel hardware, Windows, or Linux x86 operating systems.
- You must use VMware products that support VMware Infrastructure 3 platforms. The supported versions are:
 - VMware VirtualCenter Version 2.5
 - VMware ESX Version 3.5
 - VMware vSphere Version 4.0 and Version 4.1, both of which include VMware ESXi and VMware vCenter Server

The documentation generically refers to these servers with the following terminology:

- **ESX server:** Refers to VMware ESX Version 3.5 or a VMware ESXi server in VMware vSphere Version 4.0 and Version 4.1.
- **vCenter server:** Refers to VMware VirtualCenter Version 2.5 or a VMware vCenter server in VMware vSphere Version 4.0 and Version 4.1.

About this task

You can retrieve a signer certificate with a script or in the administrative console, and then define the required custom properties in the administrative console. You can also complete these steps with the script only. For more information, read about configuring VMware Infrastructure 3 platforms and Intelligent Management.

Procedure

- **If you are configuring Intelligent Management to communicate with a vCenter server:**

1. Retrieve a signer from the vCenter server and store the signers in the CellDefaultTrustStore key store. To retrieve the signer, you can either use the administrative console or run the `retrieveVMwareCertificate.py` script.

To retrieve the signer certificate by running the script:

```
./wsadmin.sh -lang jython -f retrieveVMwareCertificate.py
  -host:<vmware_virtual_center_host_name> -port:<vmware_virtual_center_ssl_port_number>
```

Where `<vmware_virtual_center_host_name>` is the host name of the vCenter and `<vmware_virtual_center_ssl_port_number>` is the secure SSL port of the vCenter.

To retrieve the signer certificate using the administrative console:

- a. Navigate to the signer certificates administrative console panel. In the administrative console, click **Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates > Retrieve from port**.
- b. Enter the host and port information for the vCenter server and an alias or name for the certificate. The alias should follow the syntax: `<vmware_virtual_center_short_host>-vmware`. For example, if the hostname of the vCenter server is `myvmwarevc.foo.net`, the alias name would be `myvmwarevc-vmware`. For Hypertext Transfer Protocol Secure (HTTPS), the default port value is 443.
- c. Click **Retrieve signer information**.
- d. Click **Apply**. This action indicates that you accept the credentials of the signer.

The signer certificate that is retrieved from the vCenter server is stored in the CellDefaultTrustStore keystore.

2. Configure custom properties for the vCenter server so that Intelligent Management can use Web services to communicate with the VMware Infrastructure SDK (VI SDK). In the administrative console, click **Cells > Custom properties > New**. Create the following cell-wide custom properties:
 - `vmware.service.unique_id.url`
 - `vmware.service.unique_id.userid`
 - `vmware.service.unique_id.password`

Note: For the `vmware.service.unique_id.userid` custom property, the following privileges are required by Intelligent Management to read certain properties and to perform various operations:

- System.Anonymous
- System.Read
- System.View
- Sessions.TerminateSession

The `unique_id` value is a unique identifier that represents the vCenter. For example, if the host name of the vCenter server is `myvmwarevc.foo.net` and the port is 443, the `unique_id` value would be `myvmwarevc_foo_net_443`. Following the same example, the names of the custom properties would be:

```
vmware.service.myvmwarevc_foo_net_443.url
vmware.service.myvmwarevc_foo_net_443.userid
vmware.service.myvmwarevc_foo_net_443.password
```

• **If you are configuring Intelligent Management to communicate with ESX servers:**

1. Retrieve a signer from the ESX server and store the signers in the `CellDefaultTrustStore` key store. To retrieve the signer, you can either use the administrative console or run the `retrieveVMwareCertificate.py` script.

To retrieve the signer certificate by running the script:

```
./wsadmin.sh -lang jython -f retrieveVMwareCertificate.py
-host:<vmware_esx_server_host_name> -port:<vmware_esx_server_ssl_port_number>
```

Where `<vmware_esx_server_host_name>` is the host name of the ESX server and `<vmware_esx_server_ssl_port_number>` is the secure SSL port of the ESX server.

To retrieve the signer certificate using the administrative console:

- a. Navigate to the signer certificates administrative console panel. In the administrative console, click **Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates > Retrieve from port**.
- b. Enter the host and port information for the ESX server and an alias name for the certificate. The alias should follow the syntax: `<vmware_esx_server_short_host>-vmware`. For example, if the hostname of the ESX server is `myvmwareesx.foo.net`, the alias name would be `myvmwareesx-vmware`. For Hypertext Transfer Protocol Secure (HTTPS), the default port value is 443.
- c. Click **Retrieve signer information**.
- d. Click **Apply**. This action indicates that you accept the credentials of the signer.

The signer certificate that is retrieved from the ESX server is stored in the `CellDefaultTrustStore` keystore.

2. Configure custom properties for the ESX servers so that Intelligent Management can use Web services to communicate with the VMware Infrastructure SDK (VI SDK). In the administrative console, click **Cells > Custom Properties > New**. Create the following cell-wide custom properties:

- `vmware.service.unique_id.url`
- `vmware.service.unique_id.userid`
- `vmware.service.unique_id.password`

The `unique_id` value is a unique identifier that represents the ESX server. For example, if the host name of the ESX server is `myvmwareesx.foo.net` and the port is 443, the `unique_id` value would be `myvmwareesx_foo_net_443`. Following the same example, the names of the custom properties would be:

```
vmware.service.myvmwareesx_foo_net_443.url
vmware.service.myvmwareesx_foo_net_443.userid
vmware.service.myvmwareesx_foo_net_443.password
```

Repeat these steps for each ESX server in your configuration.

Creating and configuring ODRs

The on demand router (ODR) is an intelligent HTTP and Session Initiation Protocol (SIP) proxy server in Intelligent Management. The ODR is the point of entry into an Intelligent Management environment and is a gateway through which HTTP requests and Session Initiation Protocol (SIP) messages flow to back-end application servers. You can configure the ODR to determine how it handles failure scenarios and how it tunes certain work requests.

z/OS

Before you begin

SIP is not supported on the z/OS operating system.

About this task

The ODR can momentarily queue requests for less important applications in order to allow requests from more important applications to be handled more quickly or to protect back-end application servers from being overloaded. The ODR is aware of the current location of a dynamic cluster instance, so that requests can be routed to the correct endpoint. The ODR can also dynamically adjust the amount of traffic sent to each individual server instance based on process utilization and response times. By default, the ODR binds to ports 80 and 443 for listening on HTTP and HTTPS, which requires running the ODR as a root user. If you want to run the ODR as a non-root user, you must change the PROXY listening ports to values greater than 1024.

The ODR is fully aware of the dynamic state of the cell, so that if one server in the cell fails, the requests are routed to another server. When the ODR is notified that the application has initialized on the restarted server, the ODR will route requests to that server again.

The ODR will not route any requests to the application on the application server until the application is through starting/initializing. If the application is started on other application server(s), then the requests will be routed to them. If the application is not started on any other servers, then the ODR will still not route to the starting-in-progress application server. Instead, a 503 is returned.

gotcha: The SIP ODR is not currently recommended for a production environment which requires high availability. If you require SIP high availability in production, use the SIP proxy server instead. If your solution does not require high availability, you can use a non-clustered SIP ODR.

Procedure

- For more information about ODRs, read about creating ODRs.

An ODR is a proxy server with advanced capabilities that are used to route work to server nodes. Note that the configuration of the ODR in the DMZ is not supported. To configure ODRs to perform a SSL offload, see read about configuring SSL offload for all HTTPS traffic. For information on other custom properties, read about the on demand router system and custom properties.

- Follow the WebSphere Application Server Network Deployment instructions in the proxy server settings topic to configure ODRs. For more information about Intelligent Management specific fields, read about configuring ODRs..

Note: In the Intelligent Management administrative console, use the following path to define the configuration of the ODR: **Servers > Server types > On demand routers > odr_name > On demand router settings > On demand router properties.**

- By default, the ODR matches the incoming protocol to the outgoing protocol. For inbound HTTP requests, the request is forwarded over outbound HTTP. For inbound HTTPS, the request is forwarded over outbound HTTPS. This default behavior can either be changed for all HTTP and HTTPS traffic that is handled by the ODR, or on a per-Web module basis. For more information, read about configuring the SSL offload for all HTTPS traffic.

- You can use ODR custom properties to change the behavior of your ODR. For example, you can change the error code that the ODR returns when messages are rejected because of processor or memory overload. For more information, read about the on demand router system and custom properties.
- A Web server should be configured as a trusted secure proxy because a trusted security proxy is allowed to pass information such as the virtual host name, or user identity to the ODR in private HTTP headers. For more information, read about configuring a web server as a trusted proxy server.
- Define routing policies for generic server clusters.
- Routing and service policies for SIP are defined at the ODRs. For more information, read about defining a service policy.


What to do next

Configure the middleware servers and dynamic clusters for your environment.

Creating ODRs

You can create on demand routers (ODRs) to route requests to Intelligent Management nodes. The ODR is fully aware of the dynamic state of the cell, so that if one server in the cell fails, the requests are routed to another server. The configuration of the ODR in the DMZ is not supported.

Before you begin

- The ODR is a server that acts as an intermediary for HTTP requests that are serviced by application servers or Web servers. Make sure you have an application or Web server installed. By default, the ODR binds to ports 80 and 443 for listening on HTTP and HTTPS, which requires running the ODR as a root user. If you want to run the ODR as a non-root user, you must change the PROXY listening ports to values greater than 1024.
-  SIP is not supported on the z/OS operating system.

About this task


A deployment manager profile cannot be used as the target profile for an ODR. Only an application server profile can be used as the target node when creating an ODR. To collocate an ODR with a deployment manager, a separate Intelligent Management-enabled application server profile must be created and federated on the same machine as the Intelligent Management-enabled deployment manager profile. You can then create an ODR within the application server profile.

gotcha: Ensure that the nodes hosting an ODR are at the same level as the deployment manager.

gotcha: The Session Initiation Protocol (SIP) ODR is not currently recommended for a production environment which requires high availability. If you require SIP high availability in production, use the SIP proxy server instead. If your solution does not require high availability, you can use a non-clustered SIP ODR.

Procedure

1. Navigate to the ODR creation wizard in the administrative console. In the administrative console, select **Servers > Server types > On demand routers**. Click **New**.
2. Select the node on which you want the ODR to be created. The selected node is pre-populated with available nodes in the cell. If your environment is a heterogeneous mix of Intelligent Management and nodes that are running other middleware software, select a product node. If you select a node that is not running the product, your ODR does not start.

Tip:  Select a node to run the ODR that does not run any dynamic clusters, unless you are using the z/OS operating system. In a z/OS environment, you can co-locate the ODR on a node that hosts application servers.

3. Determine whether to generate HTTP ports, SIP ports, or both. The Generate unique ports option is selected by default and is the recommended option. If you deselect this option, port conflicts might occur.
4. Select a server template to base your new ODR. You can use an application server template on which to model the servers for the new ODR. You can use a default template or map an existing application server. Mapping preexisting ODRs can save time. You can build one ODR, apply all of the configurations that your environment needs, and use that ODR as a template.

If you are running Intelligent Management for z/OS, ensure that the SIP option is not selected.

Results

The ODR that you created automatically routes HTTP requests to product cells.

What to do next

To enable routing to another product cell, configure your cell to communicate with other product cells.

You might want to configure the ODR to route work to nodes that are not running Intelligent Management nodes. After you create the ODR and apply any optional configuration parameters, you can define the ability to route work to nodes that are not running Intelligent Management. Note that the configuration of the ODR in the DMZ is not supported.

Configuring ODRs

You can configure an on demand router (ODR) to determine how it handles failure scenarios and tunes certain work requests. You can configure the connections and requests to the application server, configure the requests that must be rejected, define how error responses are handled, and specify the location of the proxy logs. The configuration of the ODR in the DMZ is not supported.

Before you begin

You must first create an ODR, a proxy with advanced capabilities that Intelligent Management uses to route work to application server nodes. For more information, read about creating ODRs.

About this task

You can define the configuration of the ODR by editing its proxy configuration. Define the configuration further by clicking **Servers > Server types > On demand routers > odr_name > On demand router properties > On demand router settings**. Note that the configuration of the ODR in the DMZ is not supported. To configure ODRs to perform a SSL offload, read about configuring SSL offload for all HTTPS traffic. For information on other custom properties, read about on demand router system and custom properties.

Procedure

1. Configure the ODRs. For more information, read about proxy server settings and refer to the overview of custom error page policy. The Intelligent Management-specific field for Compression Policy information follows.
2. Compression Policy. Check **Enable** to enables the compression of the HTTP response message body before sending it to the client. Choices include:

Option	Description
gzip-only	Compress the response using the gzip compression mechanism. The response is compressed only if it is acceptable to the client, based on the Accept-Encoding request header.

Option	Description
deflate-only	Compress the response using the deflate compression mechanism. The response is compressed only if it is acceptable to the client, based on the Accept-Encoding request header.
auto-only	Use either gzip, deflate, or no compression as determined by the client preference.

What to do next

After you create and configure the ODR and apply any configuration parameters, you can define the ability to route work to nodes that are not a part of your Intelligent Management environment.

Configuring a Web server as a trusted proxy server

If your environment includes a Web server, you must configure it as a trusted secure proxy server. By configuring the secure proxy server, you can inform the on demand router (ODR) that the Web server is a trusted secure proxy so that the ODR can receive requests.

Before you begin

You must first create an ODR, a proxy with advanced capabilities that Intelligent Management uses to route work to application server nodes. For more information, read about creating ODRs.

About this task

Configure a Web server as a trusted secure proxy; a trusted security proxy is allowed to pass information such as the virtual host name, or user identity to the ODR in private HTTP headers. Web servers read incoming requests to verify which ODR they are routed to. Private headers received from an untrusted proxy are discarded by the ODR. This configuration field enables intermediaries other than the ODR server to handle the request by explicitly telling the ODR that it is to trust them. A trusted security proxy receives requests before the ODR and then forwards requests to the ODR. For example, when the Web server with the WebSphere Application Server plugin forwards requests to the ODR, the Web server must be configured as a trusted security proxy.

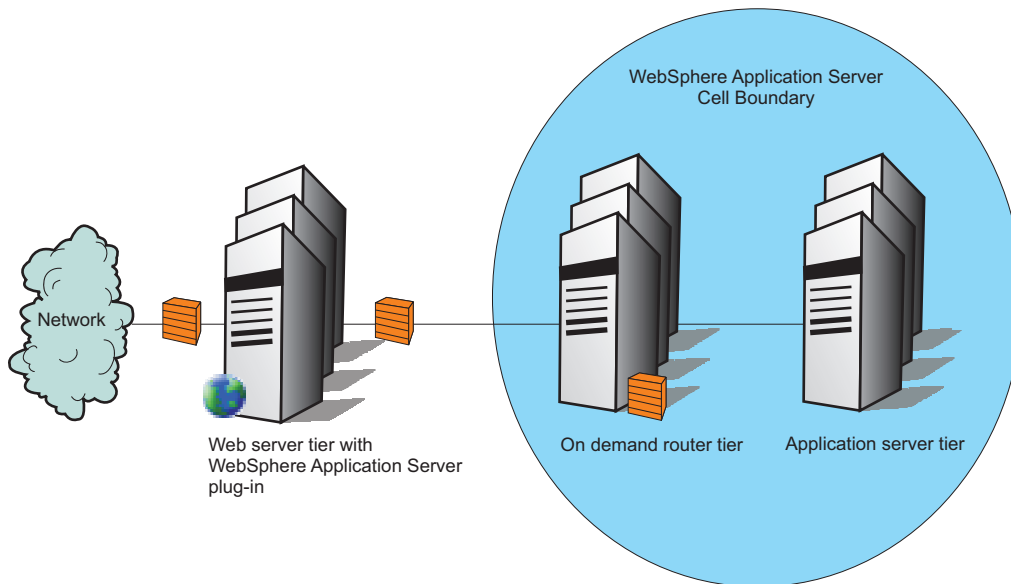


Figure 10. Example topology of a simple Intelligent Management environment supported configuration including a Web server

Procedure

1. To configure a Web server as a trusted proxy server, in the administrative console, click **Servers > Server types > On demand routers > *on_demand_router_name* > On demand router settings > On demand router properties**. If you are using dynamic clusters of on demand routers, click **Servers > Clusters > Dynamic clusters > *server_template* > On demand router properties > On demand router settings**.
2. Specify the name of the Web server in the **Trusted security proxies**. This configuration field enables intermediaries other than the ODR server to handle the request by explicitly telling the ODR that it can trust the Web server you specify. Use an internet protocol or fully qualified host name in this field. For example, myhost.com or an IP address such as 10.1.1.1.
3. Click **Apply**.
4. Click **Save**.

Results

Your Web server is now configured as a trusted proxy server.

Enabling cell affinity

The cell affinity function preserves sessions even in the event of an on demand router (ODR) outage when you configure unbridged, on demand router (ODR) topologies.

About this task

The cell affinity function prevents the loss of sessions when there are multiple ODRs within multiple, unbridged cells, and the IBM HTTP Server is configured to forward session traffic, either through load-balancing or failover, to more than one ODR.

Procedure

1. In the administrative console, select **System administration > Cell > Custom properties**.
2. Set the custom cell property odrSessionAffinityEnabled to true. For more information read about custom properties.

3. Configure generic server clusters for other ODR cells. For more information, read about defining generic server clusters for remote ODR cells.
4. If you are configuring a multi-tiered configuration, the ODRs in one tier must be configured and operate independently from the ODRs in the other tiers. For more information read about configuring cell affinity in a multi-tiered environment. Note that you can optionally perform this step in a single-tiered environment if you simply want to use a cookie name other than ODRSESSIONID.
5. Generate a `plugin-cfg.xml`.
6. Merge the `plugin-cfg.xml` files into one `plugin-cfg.xml` file. For more information, read about the `pluginmerge.bat` or `pluginmerge.sh` script.
7. Install the new `plugin-cfg.xml` file on the IBM HTTP server. For more information on how to install a new `plugin-cfg.xml` file, read about configuring an ODR to dynamically update the web server plug-in configuration.

Cell affinity function:

Using the cell affinity function, you can configure unbridged, on demand router (ODR) topologies to preserve sessions even in the event of on demand router (ODR) outages. With this function, you can configure your topology in such a way that, when an ODR receives misrouted in-session traffic, the ODR reroutes the traffic back to a working ODR in the original cell. Thus, you can configure an IBM HTTP Server to route to ODRs in multiple cells and still preserve session affinity.

The cell affinity function prevents the loss of sessions when there are multiple ODRs within multiple, unbridged cells, and the IBM HTTP Server is configured to forward session traffic, either through load-balancing or failover, to more than one ODR. For example, in a network configuration where ODRs are situated between the IBM HTTP Server and the back-end application servers, the IBM HTTP Server is not able to recognize the servers identified on the JSESSIONID cookies contained within in-session traffic because it is configured to recognize and route to the ODRs. Thus, the IBM HTTP Server generally selects different ODRs and sprays the session requests. If the IBM HTTP Server selects a router within the same cell as the hosting application server, or if all application servers share session data through a common database, the risk of losing sessions is not a concern. However, without cell affinity, if the IBM HTTP Server selects an ODR within another cell, the ODR does not recognize the server ID, does not route the request, and consequently the session is lost. The cell affinity function helps you avoid session loss.

Aspects of the cell affinity function

There are two aspects of the cell affinity function. The first aspect enables the IBM HTTP Server to affine or always to route to a specific ODR after a session is established. Configuring the IBM HTTP Server to preserve session affinity through a particular ODR is accomplished by enabling cell affinity, producing a `plugin-cfg.xml`, moving the `plugin-cfg.xml` to the IBM HTTP Server, and restarting that server. The produced `plugin-cfg.xml` instructs the IBM HTTP Server plug-in to use the ODRSESSIONID cookie for its session ID, enabling session affinity to the ODRs.

The second aspect of cell affinity is the ability to route session traffic across cell boundaries in order to direct misrouted traffic into the correct cell. To enable this function, in addition to enabling cell affinity, generic server clusters (GSCs) must be configured for each cell for which an ODR can receive traffic. The GSC members must be the ODRs in the remote cells. When an ODR receives misrouted session traffic and cell affinity is enabled, it checks the GSC lists to find the ODR associated with the ODR session ID. If a match is identified, the traffic is rerouted to the appropriate GSC. If the reroute is successful, the final ODR adopts the session and routes the traffic to the appropriate back-end server for the session.

Using generic server clusters with cell affinity:

As part of the cell affinity function, in order for the on demand router (ODR) to route to other cells, generic server clusters must be defined to represent those cells. Defining generic server clusters allows each ODR to recognize servers in remote cells.

About this task

Defining the generic server clusters provides a way for the ODRs in one cell to send traffic to the servers in the another cell. There are several reasons why this is important. First, if all the application servers in a cell are unavailable, the ODRs in that cell send the requests to the generic server clusters, representing the servers in another cell). The ODRs in the other cell routes the requests to application servers within their own cell, ensuring that the requests are handled successfully. If a request which has session data associated with an application server in a cell, *Cell1*, is mistakenly sent to an ODR in another cell, *Cell2*, the ODR in *Cell2* must be able to forward the request to an ODR in *Cell1*. Only the ODRs in *Cell1* can send the request to the appropriate application server (The ODRs in *Cell2* cannot send requests directly to an application server in *Cell1*). The generic server cluster allows the ODR in *Cell2* to forward the request to an ODR in *Cell1*, which then handles the request.

Procedure

1. Configure trusted proxies. An ODR needs to include as trusted proxies all other ODRs (including those in remote cells), and all web server or IHS servers that send traffic to this ODR. For ODR configuration instructions, read about proxy server settings. For further details, read about configuring ODRs.
2. From the administrative console, create a new generic server cluster. Select **Servers > Clusters > Generic server clusters > New**.
3. Provide a name, select a protocol, and click **OK**.
4. Click the new generic server cluster, and click **Ports**.
5. Click **New**, and specify the host name and port number of the ODR as the member of the generic server cluster.

When defining generic server clusters in which more than one member (including members in different generic server clusters) resides on the same node (i.e. has the same host name), use the *server=<uniqueServerName>* custom property to ensure that member names are unique in the on demand cluster. If this custom property is not set, the host name is used as generic server cluster member name. The host names are not unique if two members reside on the same node. The host name duplication causes improper routing.

The ODR calculates the *cloneID* for a server from the host name and the non-SSL ODR port. The same mechanism is used to calculate generic server cluster member *cloneIDs*. Thus, when an ODR in one cell is represented by a generic server cluster member in another cell, the *cloneIDs* automatically match. However, an override mechanism is provided which allows the *cloneID* to be specified manually. To do so, define a custom generic server cluster member property named *ODRCloneID* with the value set to the desired *cloneID*. This value must match the calculated *cloneID* of the ODR in the remote cell.

6. Repeat step 5 for each ODR in the cell that the generic server cluster represents.
7. Repeat steps 1 through 5 for each cell in your topology.
8. Save your configuration changes.

Results

Configuring the generic server cluster representations of the ODRs in the remote cell allows incorrectly routed traffic to be forwarded to the correct cell, maintaining cell affinity. If an ODR has no available servers in its cell, the request is not serviced. To enable the ODR to send the request to an application server in a remote cell, in the case that no local servers are available, read about configuring the on demand router for multi-cluster failover and load balancing routing.

Configuring cell affinity in a multi-tiered environment:

You can configure your network to contain multiple tiers of on demand routers (ODRs). In multi-tiered environments, configure the ODRs to have unique ODRSESSIONIDs for the different tiers.

Before you begin

You must enable cell affinity and configure the ODRs. For details, read about creating and configuring ODRs and enabling cell affinity.

About this task

An example environment for configuring cell affinity might include a cluster of ODRs that distribute load to multiple IBM Portal application servers at the front end of the network, while at the back end there are clusters of ODRs that distribute load to multiple Web content managers.

There are three important configuration requirements for this task:

1. Within a tier, any generic server cluster configurations you create for cell affinity must refer only to ODRs within that tier.
2. Within a tier, the ODR session identifier cookie name, which is named ODRSESSIONID by default, must be unique for all the other tiers. You must configure this cookie name as a custom property on each ODR within that tier, as described in the procedure in this topic.
3. If you are using the highly available (HA) plugin configuration generator, each plugin-cfg.xml configuration must define the ODR session identifier cookie name to use the IBM HTTP Server to configure ODR affinity. To define the ODR session identifier cookie name, define an additional cell-wide custom property associated with the plugin-cfg.xml generation configuration:
ODCPluginCfgOdrSessionIdCookie_<configname>=CookieName>. For more information about configuring the HA plug-in configuration generator, read about configuring an ODR to dynamically update the web server plug-in configuration.

Procedure

1. From the administrative console, select **Servers > Server types > On demand routers > odr_name**.
2. Expand **On demand router properties**.
3. Click **On demand router settings > Custom properties > New**.
4. Type **odrSessionIdCookieName** for the name value.
5. Type the new cookie name. For example, ODRSID_TIER1.
6. Repeat steps 3 through 5 for every ODR within the tier.

Configuring the multi-cluster routing policy:

In order to configure the on demand router (ODR) to route requests to a generic server cluster, you must configure a multi-cluster routing policy (MCRP) on each ODR. This allows the ODRs in one cell to send requests to a remote cell if no application servers are available in the local cell.

About this task

Configure a multi-cluster routing policy (MCRP) on each ODR, in all cells, in order to configure the on demand router (ODR) to route requests to a generic server cluster. To configure all ODRs in each cell, complete the following steps:

Procedure

1. Click **Servers > Server types > On demand routers > ODR_name > On demand router properties > On demand router settings > Custom properties > New**. \$<applicationName> is optional. If not specified, this rule applies to all traffic sent to this ODR. If specified, it only applies to traffic destined for the application with the name <applicationName>.

2. For the property name, type `MCRP@<thisCellName>${applicationName}`.
3. For the property value, type `failover@<thisCellName>${PrimaryDynamicCluster},<thisCellName>${GSCName}`. This tells the ODR to first attempt to send traffic matching the multicluster routing policy rule (based on the application name specified in step 2, if any) to the local dynamic cluster of application servers within the cell. If none of the local application servers are available, the request fails over to the specified generic server cluster (the generic cluster members in the other cell, which then route the traffic to the application servers within the cell).

Generating a plugin-cfg.xml:

In an environment with an IBM HTTP server routing requests to an on demand router (ODR), the `plugin-cfg.xml` configuration file ensures that requests are forwarded to the appropriate target servers.

About this task

Each ODR can generate a `plugin-cfg.xml` file, which includes all the handled URIs.

Procedure

1. Generate the `plugin-cfg.xml` file using the ODR. You can override the default attributes. For details, read about configuring an ODR to dynamically update the web server plug-in configuration. Generate `plugin-cfg.xml` in a high availability environment. This ensures that, despite the other processes available, the configuration is always generated by one process in the cell. For more information, read about this task.
 - In a multi-cell environment, ODRs in different cells generate cell-specific `plugin-cfg.xml` files. You can merge `plugin-cfg.xml` files for different cells into a single `plugin-cfg.xml` file by using `pluginmerge.bat` or `pluginmerge.sh`, depending on your operating system.
2. Install the new `plugin-cfg.xml` file on the IBM HTTP server. For details, read about configuring custom HTTP servers.

Cell affinity when an ODR fails:

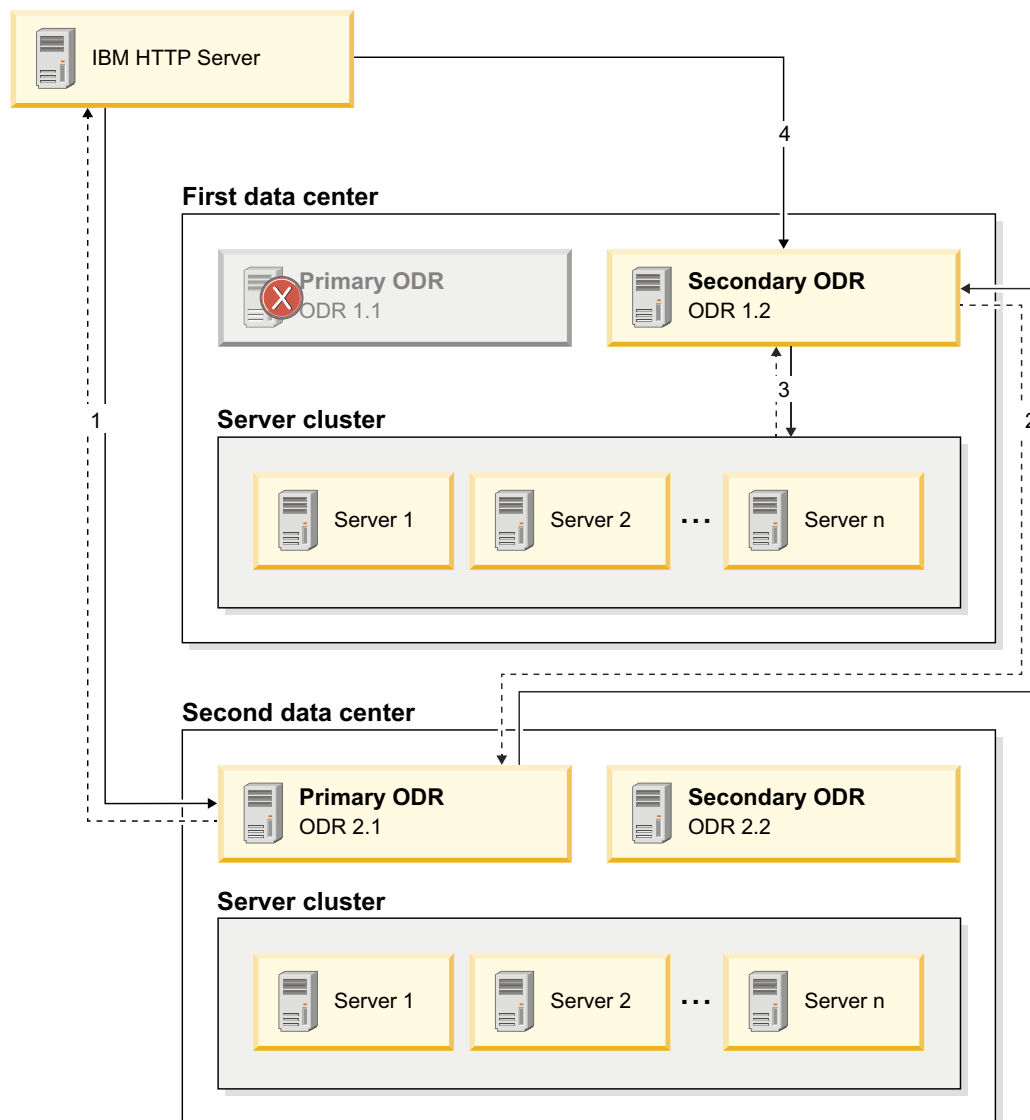
The cell affinity function enables you to configure unbridged, multi-cell, on demand router (ODR) topologies that preserve sessions even in the event of ODR outages.

A scenario featuring a request/response flow is shown in the following diagram. In this scenario, the browser has sent an in-session request to the IBM HTTP server. The IBM HTTP server has determined it cannot forward the request to the original ODR 1.1, so it has chosen to forward the request to ODR 2.1 instead (normally, this would break the session). The solid arrows in the diagram represent requests, whereas the broken arrows represent responses. The flows are explained in the following sequence:

1. The browser sends a request to IBM HTTP server. ODR 1.1 is not functioning. In an attempt to failover, IBM HTTP Server routes to ODR 2.1.
2. ODR2.1 notes that the request was originally destined for ODR 1.1, so ODR 2.1 finds a generic server cluster containing ODR 1.1, and routes back to an active ODR within the generic server cluster, namely, ODR 1.2.
3. ODR1.2 marks this session for adoption during response processing and forwards the request to the original back-end target cluster.

The IBM HTTP server could route directly to ODR 1.2 after detecting that ODR 1.1 failed. In this case, ODR 1.2 would forward the request to the correct back-end target cluster and adopts the session during response processing as described in #3 and #4 previously.

The following diagram illustrates a request/response flow scenario in which a browser sends an in-session request to the IBM HTTP Server.



Using the binary trace facility

Intelligent Management provides a binary trace facility (BTF) that permits the gathering of trace information in a production environment with a minimal impact on performance. The BTF is useful for both customers and development teams because its efficiency facilitates leaving a basic level of trace on all the time; thus, it is frequently the case that the data necessary for troubleshooting a problem has been captured the first time it occurs.

About this task

The binary trace facility (BTF) is rule-based, and deeper tracing can be enabled based on conditions detected at runtime. The BTF can:

- Use the trace data that is generated by default.
- Use the cell custom property to capture additional debug information and not impact your performance.

gotcha: Do not modify the `trace-cfg.xml` file syntax. Instead, use of the cell custom properties to control the BTF. The `trace-cfg.xml` contains the initial trace enablement settings used by the trace facility. After being loaded, `trace-cfg.xml` is not loaded again by the JVM.

The trace command has two main parameters:

The "read" parameter:

Reads the BTF file and displays it.

The "info" parameter:

Displays information about what types of information are going to be printed.

For more information, read about binary trace facility custom properties.

Procedure

1. To translate the binary trace files that are produced by the BTF into a human readable form. Run the following command from the `app_server_root/bin/trace` directory:

```
app_server_root/bin/trace read <logDirectory> [<traceSpec>]
```

<logDirectory>

Specifies the path to a server log directory.

<traceSpec>

Specifies the trace specification, which is a comma-separated list of elements of the form **<groupOrRecord>[<:traceLevel>]**, where **groupOrRecord** specifies the name of a group or record, and **traceLevel** identifies the level of debug desired, as defined by the group or record.

-filePrefix

Indicates that a non-default trace file prefix name is being supplied.

<prefix>

Specifies the trace file prefix name. The default value is `btrace`

For example, to read the `btrace` files in the current directory for all records, run the following command,

```
app_server_root/bin/trace read .
```

To see all `btrace` in the `http` group in the logs from the current directory, down to level 2 trace, run the following:

```
app_server_root/bin/trace read . http:2
```

2. To get a description of everything that can be traced, run the following command:

```
app_server_root/bin/trace info [<groupOrRecordOrFieldName>] [-depth <maxDepth>]
```

<groupOrRecordOrFieldName>

Specifies the name of a group, record, or field for which information is needed.

-depth

Indicates that a maximum depth is being supplied.

<maxDepth>

The depth that you specify will include all groups, records, and fields that are within the specified depth from the initial argument provided.

Use the field descriptors to limit your output, and `-depth` to further refine that by limiting how deep into the hierarchy the filter should go. If you do not specify a maximum depth, all of the information is displayed.

For example, for information on the `http.request` group, but no deeper, run the following command:

```
app_server_root/bin/trace info http.request -depth 1
```

Tuning the on demand router (ODR)

ODRs are intelligent routers for SIP and HTTP traffic, acting as intermediaries for application servers and web servers. There are many factors that affect ODR performance; in order to get the optimal performance from your ODRs, it is sometimes necessary to tune them.

Before you begin

- Tune Java virtual machines. For more information, read about tuning the IBM virtual machine for Java.
- Remove all tracing information except for `*=info`, because this type of tracing impacts ODR performance. To change the tracing information, perform the following steps:
 1. Select **Servers > Server types > On demand routers > *on_demand_router* > Troubleshooting > Logging and tracing > Diagnostic trace service > Change log detail levels.**
 2. Ensure that only `*=info` is specified.
- The ODR should never be constrained by either CPU or memory usage. Therefore, when you install the ODR in an environment with server virtualization, configure the virtual machine or LPAR in which the ODR runs in dedicated processor mode, or configure it in a mode that guarantees the ODR receives a sufficient amount of CPU resources and dedicated memory when the ODR runs.
- Binary Trace Facility (BTF) has minimal impact on performance and can be left enabled

About this task

The default settings of the ODR work for most people, most of the time. For your installation, it may be necessary to carry out some or all of the following steps to obtain maximum performance. The steps are prioritized in order of importance.

Procedure

1. Check the JVM settings. For more information about the JVM settings, read about modifying the JVM heap size for the on demand router.
2. **HP-UX** HP and Sun provide additional tuning parameters to optimize garbage collection. For generational garbage collection JVMs such as Sun and HP, or IBM's J9 JVM when using gencon garbage collection, set the permanent memory region to approximately 100MB to encompass the base 90MB foot print that exists in the ODR. Further, a SurvivorRatio of 16 further optimizes processing in the young generation. On HP JVM, you can turn NIO to yield an increase in performance by using the `-Djava.nio.channels.spi.SelectorProvider=sun.nio.ch.DevPollSelectorProvider` selector provider and disabling the extra poll before a read: `-XX:-ExtraPollBeforeRead`.
3. Tune the connection keep alive settings. For more information, read about tuning ODR persistent connections.
4. Tune the ODR maximum connections per server. For more information, read about tuning ODR maximum connections.
5. Disable ODR caching when not in use. When the ODR caching is enabled, the ODR must go through the process of determining whether a request should be cached, then examine the cache repository to check whether the request was previously cached. This additional overhead on the ODR may create a bottleneck at the ODR.
6. Disable access logging if not needed. If you do need access logging, then the proxy logging is preferred over the HTTP Channel/NCSA logging as the proxy access logging happens outside of the request/reply path. Thus, it does not affect the response time of the request. Access logging on a fairly fast disk is typically 5% overhead, but the percentage is highly dependant on disk performance.
7. Use the same thread group for both inbound and outbound work, which will avoid moving requests across threads and eliminates the resulting overhead. The ODR has a set of threads that tune itself under most circumstances. Queuing and throttling requests are dispatched to the default thread pool, which you can tune so that it will only handle the overflow requests. The primary thread group will continue to handle most requests. All requests on the thread pool are asynchronous with no blocking calls, so the number should not be more than one or two threads per CPU. Complete the following steps to use the same thread group for both inbound and outbound work:
 - a. Select **Servers > Server types > On demand routers > *on_demand_router* > Thread pools > *default_thread_pool* > Custom properties > New.**
 - b. Specify **combineSelectors** for the name.
 - c. Specify **1** for the value.

- d. Click **OK**.
- e. Click **Save**.

Modifying the JVM heap size for the on demand router:

The Java virtual machine (JVM) heap size setting is the single most critical factor for efficient on demand router (ODR) performance. You might need to modify the JVM heap size setting based on your environment configuration.

About this task

The base setting for the ODR under 100% load is approximately 90MB. However, the ODR also needs memory for temporary working space. Since the ODR acts the same for every request, the major factor to determine how much working space is required is throughput. The ODR generates approximately 4 to 5KB of temporary objects per request. The desired time between garbage collection cycles should be at least 10 seconds to minimize heap contention, and generally under 60 seconds to avoid excess delays in completing garbage collection cycles. Test this value, as may factors such as high numbers of queued requests or large in-memory caches can increase the required memory. Testing is important, because values cannot be determined arbitrarily for all possible applications. Therefore, the recommended initial heap setting is calculated as:

$90MB + 0.05 \text{ MB/request} \times \text{peak requests per second} \times \text{desired time between garbage collection (sec)}$

Procedure

1. In the administrative console click **Servers > Server types > On demand routers > on demand router name > Server infrastructure > Java and process management > Process definition > Java virtual machine**.
2. Specify the initial heap size and maximum heap size.

Tuning ODR persistent connections:

Tune the persistent connection settings for ODR performance to minimize the time you spend setting up new connections and taking down older connections during a barrage of requests. Primarily, this tuning is relevant to the connections between the ODR and the application servers.

Before you begin

Modify your JVM heap settings. For more information about the JVM settings, read about modifying the JVM heap size for the on demand router.

About this task

Adjust the application server's number of persistent requests so that the ODR reuses the connection to the application server for as long as it can. To do this, configure the ODR transport chain to tune inbound connections to the ODR. To tune outbound connections from the ODR, configure the transport chains of the application servers that are members of a dynamic cluster. To do so, configure the transport chain of the dynamic cluster from within the dynamic cluster server template. Changes to the dynamic cluster's server template are automatically applied to all cluster members.

You should set number of persistent requests high enough to avoid having to bring down and recreate a connection during a barrage of requests, so set the value to at least the number of requests in a single barrage of request. The barrage can be a single page with associated data with a long think time, or a series of pages and associated data in short succession. Setting a value higher than the number of requests in a single barrage of request will not negatively affect performance provided the persistent timeout is set appropriately to close idle connections.

Procedure

- Tune the ODR's HTTP_PROXY_ADDRESS and HTTPS_PROXY_ADDRESS transport chains. In the administrative console, click **Servers > Server types > On demand routers > odr_name > Ports**.
 1. Click **View associated transports** for the port whose transport chains you want to view.
 2. Click **transport_chain_name**.
 3. Click **HTTP inbound channel**.
 4. Ensure that **Use persistent (keep-alive) connections** is checked.
 5. Specify an unlimited number of persistent requests per connection. Select **Unlimited persistent requests per connection**.
 6. Specify a maximum number of persistent requests per connection. Select **Maximum persistent requests per connection**. Setting this value to **-1** yields the best performance.
 7. Click **Apply** and **OK** to save your changes.
- Tune the application servers' HTTP_PROXY_ADDRESS and HTTPS_PROXY_ADDRESS transport chains. In the administrative console, click **Servers > Clusters > Dynamic clusters > dynamic_cluster_name > Server template > Web container settings > Web container transport chains**.
 1. Configure the WCInboundDefault transport chain.
 - a. Click **WCInboundDefault**.
 - b. Click **HTTP inbound channel**.
 - c. Ensure that **Use persistent (keep-alive) connections** is checked.
 - d. Specify an unlimited number of persistent requests per connection. Select **Unlimited persistent requests per connection**.
 - e. Specify a maximum number of persistent requests per connection. Select **Maximum persistent requests per connection**, and set the value to **-1**. This allows an infinite number of requests to flow over a single connection. Connections that are not used for an extended period of time are still closed due to the inactivity timeout for the connection.
 - f. Click **Apply** and **OK** to save your changes.
 2. Configure the WCInboundDefaultSecure transport chain.
 - a. Click **WCInboundDefaultSecure**.
 - b. Click **HTTP inbound channel**.
 - c. Ensure that **Use persistent (keep-alive) connections** is checked.
 - Select **Unlimited persistent requests per connection** to specify an unlimited number of persistent requests per connection.
 - Select **Maximum persistent requests per connection** to specify a maximum number of persistent requests per connection. Setting this value to **-1** yields the best performance.
 - d. Click **Apply** and **OK** to save your changes.

Tuning ODR maximum connections:

Increasing ODR maximum connections per server allows for more connections to be created and pooled, and enables keeping connections for a significant amount of time.

Before you begin

Modify your JVM heap settings. For more information about the JVM settings, read about modifying the JVM heap size for the on demand router.

About this task

To enable keeping connections for a significant time, the ODR maximum connections per server should be increased to allow for more connections to be created and pooled. The ODR allocates connections beyond the setting specified, but will not pool those connections when they are returned to the pool if the pool already contains this amount of idle connections. A general setting for the maximum connections per server can be calculated as the peak number of concurrent clients divided by the number of servers allotted for an application. For dynamic server clusters, the number of servers is the minimum number of servers the cluster is defined to.

The following steps show you how to adjust the application server's maximum connections per server. From the administrative console:

Procedure

1. Click **Servers > Server types > On demand routers > odr_name > On demand router properties > On demand router settings**.
2. Use the calculation described previously and enter your result in the **maximum connections per server** field.

Configuring custom logs

By configuring custom logs, you can determine what data to capture in the logs and how to display the logged data. You can configure custom logs using the `wsadmin` command, through rule set administrative tasks, or by using the `manageODR.py` script.

About this task

With custom logging, the on demand router (ODR) records customized entries to log files based on defined conditions. For each request, the ODR evaluates the rule conditions. For each fulfilled condition, the ODR logs an entry in the custom log file. For example, a specific URL or a specific virtual host name.

Validation is limited to type checking. Thorough validation is performed during the ODR startup. Problems are reported through runtime exceptions recorded in the `SystemOut` log file.

Procedure

1. Create a rule set:

```
wsadmin>$AdminTask createRuleset {-odrname odr1 -nodename myNode01 -rulesetName myRuleset -rulesetType HTTP -defaultContinue true}
```

2. Add a rule to the rule set that you created, with the expression set to match when the virtual host is NOT `badvhost`.

```
wsadmin>$AdminTask addRuleToRuleset {-odrname odr1 -nodename myNode01 -rulesetName myRuleset -ruleName myRule -rulePriority 0 -expression "virtualhost <> 'badvhost'}
```

3. Add a custom log action. Note the @ symbol that is used in the `actionValue` argument. The following example shows the `wsadmin` command for listing the configurable attributes for the action and the results:

```
wsadmin>$AdminTask addActionToRule {-odrname odr1 -nodename myNode01 -rulesetName myRuleset -ruleName myRule -actionName myCustomLogAction -actionType log -actionValue "Custom1.log %r %B %h %z %Z@Custom2.log %t %r %s" -actionContinue true}
```

4. Save your changes.

```
wsadmin>$AdminConfig save
```

Example

Two examples of custom log outputs follow. These examples are based on the formats specified in the previous actions, after running several HTTP requests.

```
"GET /BADAPP HTTP/1.1" 70 myremotehost.mycompany.com - -  
"GET /A/ServerInfo HTTP/1.1" 422 myremotehost.mycompany.com 9.44.111.191 myCell01/myNode01/MyClusterA_myNode01  
"GET /A/ServerInfo HTTP/1.1" 90 myremotehost.mycompany.com - -  
"GET /favicon.ico HTTP/1.1" 70 myremotehost.mycompany.com - -
```

```

"GET /favicon.ico HTTP/1.1" 70 myremotehost.mycompany.com - -
"GET /A/ServerInfo HTTP/1.1" 422 myremotehost.mycompany.com 9.44.111.191 myCell101/myNode01/MyClusterA_myNode01
"GET /A/ServerInfo HTTP/1.1" 422 myremotehost.mycompany.com 9.44.111.191 myCell101/myNode01/MyClusterA_myNode01
01/Jul/2009:20:35:35 EDT "GET /BADAPP HTTP/1.1" 404
01/Jul/2009:20:35:43 EDT "GET /A/ServerInfo HTTP/1.1" 200
01/Jul/2009:20:37:21 EDT "GET /A/ServerInfo HTTP/1.1" 503
01/Jul/2009:20:37:21 EDT "GET /favicon.ico HTTP/1.1" 404
01/Jul/2009:20:37:24 EDT "GET /favicon.ico HTTP/1.1" 404
01/Jul/2009:20:42:55 EDT "GET /A/ServerInfo HTTP/1.1" 200
01/Jul/2009:20:42:59 EDT "GET /A/ServerInfo HTTP/1.1" 200

```

Custom log file format:

When a request is processed, you can create rule sets to specify what data to include in the logs, the display order, the conditional logic for filtering which requests are included, and the log file number and names.

Custom property objects

Custom property objects can be used at the rule set, rule, and action level. Rules determine the conditions based upon which log actions are executed. An expression field is used to determine the requests and responses to log. This expression uses the standard HTTP language operands, and the *response.code* operand to filter by HTTP response codes, such as 404, or 503. For example, *response.code = 200* filters all responses for response code 200. The HTTP operand *targetserver* is used to show the server where the request was sent in a WebSphere Application Server format. The HTTP custom log operand *service.time* is the time frame between when the request is sent to the application server, and when a response is received from the application server. The HTTP custom log operand *response.time* is the time frame between when the request is received in the ODR, and when the request response is sent from the ODR. For more information, read about HTTP custom log operands.

The rule priority determines the order of the rule evaluation. Specific log actions occur when rules are matched. The log actions have a *continue* field. When *continue* is set to true, subsequent log actions can be executed. However, if *continue* is set to false, once the log action is finished, no subsequent log actions are performed. Use the *manageODR.py* script to create a cluster of ODRs and to manage custom logging. For more information, read about the *manageODR.py* script.

ruleset

name

Specifies the name of the rule set. (String, required)

type

Specifies the type of rule set. (String, protocol identifier such as HTTP)

continueDefault

Determines the default value for *continue*. (Boolean, required)

properties

Optional

rule

Required

rule

name

Specifies the name of the rule. (String, unique within ruleset)

priority

Required, integer (unique within ruleset)

expression
String, required.

action
Required.

properties
Optional

Action

name
Specifies the name of the action. (String, unique within rule)

priority
Optional, integer (unique within rule).

type
String, required, log

value
Required.(String, contains a series of logFormats delimited by @. For example:
value="custom01.log %a%z@custom02.log @(#) 10 1.6@(#)Z"

continue
Optional, boolean. If not set, defaults to `continueDefault` value; true, false.

properties
Optional

customProperties

name
Specifies the name custom property. (String)

value
Required.(String)

continue
Optional, boolean. If not set, defaults to `continueDefault` value; true, false.

Log action format

The action value attribute contains a set of arguments to indicate the file name and format of the log. To create multiple logs in the value attribute, delimit the arguments with a @. The first argument for the log action is the name of the log file, relative to the ODR logs directory. The following arguments are parameters to specify what information to place into the log. The data is shown in the order in which the parameters are specified.

Table 3. Custom log parameters

Parameter	Description
%a	Remote IP address.
%A	Local IP - address.
%B	Bytes sent, excluding HTTP headers.
%b	Bytes sent, excluding HTTP headers. In CLF format, that is, a '-' rather than a zero when no bytes are sent.
%{FOOBAR}C	The contents of the cookie in the request sent to the server.

Table 3. Custom log parameters (continued)

Parameter	Description
%{FOOBAR}i	The contents of Foobar: header lines in the request sent to the server.
%I	Bytes received, including request and headers, cannot be zero.
%{FOOBAR}e	The contents of the debug argument ("-D") FOOBAR specified for the JVM of the ODR process.
%h	Remote host.
%H	The request protocol HTTP or HTTPS.
%m	The request method.
%{FOOBAR}o	The contents of Foobar, header lines in the reply.
%O	Bytes sent, including headers; the number cannot be zero.
%p	The canonical port of the server serving the request.
%q	The query string, with ? prefix, if a query string exists. Otherwise, it is an empty string.
%r	First line of request.
%R	Response time in milliseconds. Combined time spent in ODR and application server.
%s	Status, HTTP response code, that is, 503, 404, 200.
%t	Time, in common log format time format, standard English format.
%{format}t	The time, in the form given by format, which should be in strftime(3) format. Potentially localized.
%T	The time taken to serve the request, in milliseconds.
%U	The URL path requested, not including any query string.
%v	The canonical ServerName of the server serving the request.
%z	Destination IP address.
%Z	Destination server name (cell/node/server).

Configuring SSL offload for all HTTPS traffic

You can change the default outgoing protocol for requests that go through the on demand router. By default, the on demand router matches the incoming protocol to the outgoing protocol.

Before you begin

Create the on demand router. For more information, read about creating ODRs.

About this task

The on demand router has the following default behaviors:

- For inbound Hypertext Transfer Protocol (HTTP) requests, the request is forwarded over outbound HTTP.
- For inbound Hypertext Transfer Protocol Secure (HTTPS) requests, the request is forwarded over outbound HTTPS.

You can change this default behavior for all HTTP and HTTPS traffic that the on demand router handles, or on a per-web module basis.

Procedure

1. Configure the on demand router (ODR) to perform Secure Sockets Layer (SSL) offload for all HTTPS traffic. In the administrative console, click **Servers > Server types > On demand routers > on_demand_router > On demand router properties > On demand router settings > Custom properties > New**.
2. Specify a custom property named **http.protocolMap** and a value of **SSL-offload**.
3. Click **Apply**.

Configuring SSL offload for partial HTTPS traffic

Configure the ODR to perform SSL offload for partial HTTPS traffic, or on a per-web module basis.


Procedure

1. In the administrative console, click **Applications > Application types > WebSphere enterprise applications > application_name > Manage modules > Web_module_name > Web module proxy configuration**.
2. For **Web module transport protocol**, select **HTTP**.

Integrating the SIP on demand router with Load Balancer

You can integrate the Session Initiation Protocol (SIP) on demand router (ODR) with Load Balancer. Load Balancer for IBM WebSphere Application Server can help maximize the potential of your website by providing a powerful, flexible, and scalable solution to peak-demand problems.

Before you begin

Restriction:  Intelligent Management does not support SIP features on the z/OS operating system.

Install Load Balancer for IBM WebSphere Application Server. See the Edge Components Information Center for installation information.

gotcha: The SIP ODR is not currently recommended for a production environment which requires high availability. If you require SIP high availability in production, use the SIP proxy server instead. If your solution does not require high availability, you can use a non-clustered SIP ODR.

Procedure

1. Start the Load Balancer.
 - a. From the command prompt, type **dsserver start**.
 - b. Then type **lbadmin** to start the administrative console for the Load Balancer.
 - c. From the administrative console, right click **Dispatcher**, and then select **Connect to host**.
 - d. Right click the hostname and select **Start executor**.
2. Start the configuration wizard for the load balancer. Right-click **Dispatch > Start configuration wizard**.
 - a. Select default host.
 - b. Type a cluster address. The cluster address is not available before the executor starts. You must specify this same value for host when you create a user-defined port.
 - c. Type a port number, such as **5060**.
 - d. Add servers to the port. Add each server to which the load balancer will proxy traffic. In your configuration, the load-balanced server is the ODR server for your WebSphere Application Server configuration.

- e. Start an advisor by clicking **Yes**. For example, for HTTP traffic, start the HTTP advisor. For SIP traffic, start the SIP advisor. The advisor tells the manager whether a specific port is accepting traffic.
3. Alias the cluster address on the SIP on demand router loopback adapter. For example, type `lb-alias.sh cluster_ip_address:`

```
#!/bin/sh

CLUSTER=$1

if [ -f /proc/sys/net/ipv4/conf/all/arp_ignore -a -f /proc/sys/net/ipv4/conf/all/arp_announce ]
then
echo Using arp_ignore
echo "3" >/proc/sys/net/ipv4/conf/all/arp_ignore
echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
ip addr add $CLUSTER/32 scope host dev lo
elif [ -f /proc/sys/net/ipv4/conf/all/hidden -a -f /proc/sys/net/ipv4/conf/lo/hidden ]
then
echo Using hidden
echo "1" > /proc/sys/net/ipv4/conf/all/hidden
echo "1" > /proc/sys/net/ipv4/conf/lo/hidden
ip addr add $CLUSTER/32 dev lo
elif [ ! -z "$( which arptables )" ]
then
echo Using arptables
arptables -A IN -s $CLUSTER -j DROP
arptables -A OUT -s $CLUSTER -j mangle --mangle-ip-s $(hostname)
ip addr add $CLUSTER/32 dev lo
else
echo Using iptables
iptables -t nat -A PREROUTING -d $CLUSTER -j REDIRECT
fi
```

Load Balancer Administration Guide for more information.

4. Define SIP ODR custom properties from the Intelligent Management administrative console.
 - a. From the administrative console, click **Servers > Server types > On demand routers > odr_name > SIP on demand router settings**.
 - b. Under General Properties, in the Additional Properties section, click **Custom properties**.
 - c. Create the following SIP ODR custom properties:
 - LBIPAddr** : The IP address of the load balancer machine.
 - SIPAdvisorMethodName**: The type of messages sent by the Load Balancer advisory. This name is INFO.
 - serverUDPInterface**: The IP address of the server machine.
 - serverUDPPort** : Specify an unused port number. For example, 5080.
 - udp.IPSprayer.host**: You specified this value in step 2b.
5. Create a user-defined port from the Intelligent Management administrative console.
 - a. From the administrative console, click **Servers > Server types > On demand routers > on_demand_router > Communications > > Ports**.
 - b. Click **New**.
 - c. Select User-defined port.
 - d. Enter SIP_LB_Address for the **Port name**.
 - e. Enter a value for the **Host**. You specified this value in step 2b.
 - f. Enter a value for the **Port**. You specified this value in step 2c. Click **OK**. **Save** changes.
 - g. Under Ports, modify the **PROXY_SIP_ADDRESS** from * to the actual host name of the proxy server machine.
 - h. Click **Apply**, and then click **Save**.
6. Modify the SIP on demand router transports.

- a. From the administrative console, click **Servers > Server types > On demand routers > odr_name > SIP on demand router settings > SIP on demand router transports > UDP_SIP_PROXY_CHAIN > UDPInbound Channel (UDP_1)**.
- b. From the **Port** drop-down list, select **SIP_LB_Address**.
- c. Click **Apply**, and then click **Save**.
7. Verify that the host name of the ODR machine is set for the PROXY_SIP_ADDRESS. To verify from the administrative console, click **Servers > Server types > On demand routers > odr_name > Communications > Port > PROXY_SIP_ADDRESS**.
8. Create a UDP transport chain. Click **Servers > Server types > On demand routers > odr_name > SIP on demand router settings > SIP on demand router transports > New**.
 - a. Type a name for the new chain, such as ODR_SIP_PROXY_CHAIN.
 - b. In the drop-down list, select **Proxy-UDP**.
 - c. Click **Next**.
 - d. Select **Use existing port** and **PROXY_SIP_ADDRESS**. Click **Next** and then **Finish** on the summary page. If **PROXY_SIP_ADDRESS** does not show in the list of existing ports, select any of the ports to complete the transport chain creation, then modify the UDP inbound channel of the new chain. Select the new transport chain, then select the UDP inbound channel and change the port to **PROXY_SIP_ADDRESS**.
9. Create a TCP transport chain. Click **Servers > Server types > On demand routers > odr_name > SIP on demand router settings > SIP on demand router transports > New**.
 - a. Type a name for the new chain, such as TCP_SIP_PROXY_CHAIN.
 - b. In the drop-down list, select **SIP_LB_Address**.
 - c. Click **Next** and then **Finish** on the summary page.
 - d. Restart the ODR to pick up the changes.
10. Create a system property named `clientUDPInterface` on each ODR and set it to the IP address of the cluster address. To create the system property, select **Servers > Server types > On demand routers > odr_name > Java and process management > Process definition > Java Virtual Machine > Custom properties > New**.
11. **Save** and synchronize the configuration.
12. Restart the ODR.

Defining routing policies for generic server clusters

You can define routing policies for generic server clusters. A routing policy is a set of rules that determine how the server routes incoming requests.

Procedure

1. In the administrative console, click **Servers > Server types > On demand routers > ODR_name > On demand router properties > Generic server cluster routing policies**.
2. Click **Work classes for HTTP requests** and click **New**.
3. Type a name for the work class in the **Name** field.
4. Click **Finish** to confirm your new work class.
5. To set a URI match condition for an existing rule, type the name of the URI in the **If URI matches** field, and from the **For virtual host** list, select a host name.
6. Assign a work class with no rules and provide choices for the other fields in the panel.
7. Click **Apply** or **OK** to commit your new rule settings. The new ODR has a default routing policy, a URI pattern of asterisk (*), a virtual host selection of `default_host` and a default routing action of `Reject` with return code with the reject return code set to 404.

What to do next

After you create the ODR and apply any optional configuration parameters, you can define the ability to route work to nodes that are not running Intelligent Management.

Configuring rewriting rules:

You can configure rewriting rules to redirect requests to the proxy server, which then routes the requests to the target server. The proxy server acts as an intermediary for HTTP requests that are serviced by application servers or Web servers.

About this task

Typically, the proxy server recognizes which target server is active and routes all administrative communication to that target. In some cases, however, a target server might receive a request without the proxy server intercepting that request. Rewriting rules modify URLs to point requests to the proxy server instead of pointing requests directly to a target server.

Procedure

1. Click **Servers > Server types > On demand routers > odr_name > On demand router properties > Rewriting rules > New** in the administrative console.
2. Type the original URL pattern in the **From URL pattern** field. The pattern can include the following wild card symbol: *
3. Type the modification for the URL in the **To URL pattern** field. The pattern can include the following wild card symbol: *

For example, you receive a redirected response with the original location header of `http://internalserver/secure/page.html`. You configure a rule with the following patterns:

- `http://internalserver/*` as the **From URL pattern**
- `http://publicserver/*` as the **To URL pattern**

As a result, the rule rewrites the location header as `http://publicserver/secure/page.html`.

Defining service policies for generic server clusters

Optionally, you can define service policies for generic server clusters.

About this task

Use the administrative console to define service policies for generic server clusters by clicking **Servers > Server types > On demand routers > ODR_name > On demand router properties > Generic server cluster service properties**.

Procedure

1. Select a work class for HTTP requests and click **New**.
2. Type a name for the work class in the **Name** field.
3. Click **Finish** to confirm your new work class.
4. To set a URI match condition for an existing rule, type the name of the URI in the **If URI matches** field, and from the **For virtual host** list, select a host name. These fields are required.
5. If a work class has no rules, from the **If no classification rules apply, then classify to this transaction class** list assign the work class to a transaction class. This field is required whether or not rules exist and is the default action in the absence of rules.
6. Click **Apply** or **OK** to commit your new rule settings.

What to do next

After you create the ODR and apply any optional configuration parameters, you can define the ability to route work to nodes that are not running Intelligent Management.

BBSON bulletin board

Intelligent Management offers a bulletin board service overly network (BBSON) that is independent of the WebSphere Application Server Network Deployment high availability manager. The high availability manager provides a mechanism that allows servers to easily exchange state data. This mechanism is commonly referred to as the bulletin board.

Overview

BBSON, which is enabled by default, alleviates the Intelligent Management dependency on the WebSphere Application Server Network Deployment high availability manager. By using BBSON you avoid constructing and managing core groups and bridges required for Intelligent Management. You can use BBSON for new cells bridged to existing cells running WebSphere Virtual Enterprise Version 7.0 or later versions. For configuration information about the WebSphere Application Server Network Deployment high availability manager, see the topic on setting up a high availability environment.

Considerations for using BBSON

Typically, a cell with 50 or more Java virtual machines (JVMs) requires core group configuration and bridging. If you anticipate your cell to increase to 50 or more JVMs, use BBSON.

Troubleshooting

For problem diagnosis and debugging purposes, use the `manageBBSON.py` script to collect BBSON state dumps from every process in the product cell. For more information, read about the `manageBBSON.py` script.

trns: The WebSphere Virtual Enterprise command that equates to `manageBBSON.py` is `manageWVEBB.py`. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the `manageWVEBB.py` command, which operates the same as the `manageBBSON.py` command.

Restrictions

The following restrictions apply when using BBSON:

- Do not use BBSON if you have an older version than WebSphere Virtual Enterprise 7.0 in your cell.

Configuring Intelligent Management for cross-cell communication

You can enable cross-cell communication when an Intelligent Management cell contains servers that are enabled with an on demand router (ODR) that routes to other WebSphere Application Server administrative cells.

Before you begin

Configure the cells that need to communicate with each other and create the on demand router.

About this task

Use this method of configuring cross-cell communication only if the bulletin board service overly network (BBSON) has been explicitly disabled. Otherwise use the procedure described in Configuring multi-cell performance management.

When the ODR needs to route work to servers that are in different cells, you can configure cross-cell communication by creating a core group bridge.

You must use the **crossCellCGBCfg** command to configure Intelligent Management cells to communicate. Do not create a core group bridge in the administrative console.

The **crossCellCGBCfg** command performs the following configuration actions:

- Enables the core group bridge service between the cell that is running the ODR and another WebSphere Application Server administrative cell.
- Configures core group bridges on all of the node agents in both cells.
- Enables the Intelligent Management overlay communication function between the cells.

Running the **crossCellCGBCfg** does not remove any existing core group bridge configurations. After you run the script, you can update the core group bridge configuration for your environment.

Procedure

1. Start the node agents and deployment managers that you want to participate in cross-cell communication.
2. Verify that the XD-CGB-EXPORT file is created for each cell. The XD-CGB-EXPORT file is in the *app_server_root/profiles/deployment_manager_profile_name/config/cells/cell_name* directory. If you are using a firewall, verify that the firewall is open for the ports that are listed in the XD-CGB-EXPORT file to support communication.
3. Verify that the overlaynodes.config file is created for each cell. The overlaynodes.config file is in the *app_server_root/profiles/deployment_manager_profile_name/config/cells/cell_name* directory. If you are using a firewall, verify that the firewall is open for the ports that are listed in the overlaynodes.config file to support communication.
4. Enable communication. For more information, read about enabling communication between cells that have security enabled.

If the cells at you are using for cross-cell communication have some form of security enabled, such as Lightweight Directory Access Protocol (LDAP) administration security, set up communication between the cells.

5. Run the **crossCellCGBCfg** command in each cell.
 - a. From a command prompt, type the following command against the ODR deployment manager process for each back-end cell to which the ODR cell routes:

```
crossCellCGBCfg create deployment_manager_host deployment_manager_SOAP_port  
path_to_external_cell_XD-CGB-EXPORT path_to_external_cell_overlaynodes.config_file
```

When security is enabled for the **crossCellCGBCfg** command, include the user ID and password as additional parameters: `crossCellCGBCfg create deployment_manager_host deployment_manager_SOAP_port path_to_external_cell_XD-CGB-EXPORT path_to_external_cell_overlaynodes.config_file user_name password`

This command identifies to the ODR the external cell that is represented by the XD-CGB-EXPORT and overlaynodes.config files. The XD-CGB-EXPORT file is a subset of the serverindex.xml file that isolates the required endpoints for cross-cell high availability manager communication in a runtime environment. If you examine the contents of the serverindex.xml file directly or by way of the endpoint manager for a node agent in the administrative console, the endpoint that is used by the cross cell function is P2P_CGBS_UNICAST_ADDRESS. Any firewall that is employed between the ODR cell and the back-end cell needs to allow the traffic to flow through the ports that are represented by this endpoint on each node in the ODR and back-end cells.

- b. Run the **crossCellCGBCfg** command on each back-end cell, against the deployment manager process of that cell, to import the XD-CGB-EXPORT and overlaynodes.config files of the ODR cell. Importing the files between two back-end cells is not supported.
6. Stop all of the processes in each cell.
 7. Restart each cell.

Results

The on demand router routes work to servers that are in different cells.

Example

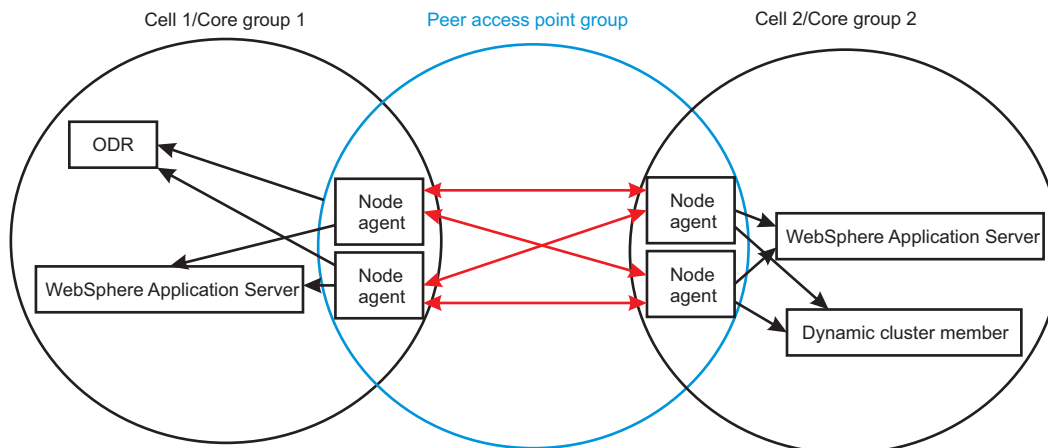


Figure 11. Cross-cell communication topology diagram. This topology shows the logical division between two cells where each cell is running several servers. Each node agent runs a peer access point bridge between the cells, making the bridges part of a peer access point group.

What to do next

To clear previous imports and start over, run the following command:

```
crossCellCGBCfg clear deployment_manager_host deployment_manager_SOAP_port
```

Or, if you want the cross cell communication to use Secure Sockets Layer (SSL), set the `CGBS_USE_SSL` system property to a non-null value on each server in the ODR and backend cells.

Enabling communication between cells that have security enabled:

When two cells have security enabled, such as Lightweight Directory Access Protocol (LDAP), you must perform additional steps so that these cells can communicate with each other.

Before you begin

You must be able to access the deployment manager for each cell you want to communicate.

About this task

You can add a signer certificate to the `trust.p12` file, allowing that cell to securely communicate with another cell. You edit the `trust.p12` file at the cell level for each cell, and then run the `bin/retrieveSigners.sh` script in each cell. After you run the script, the cells can communicate through Secure Socket Layer (SSL) connections.

Procedure

1. Start the deployment manager for each cell.
2. On each deployment manager, edit the `deployment_manager_profile/properties/ssl.client.props` file to change the `com.ibm.ssl.trustStore` value to the cell-level default trust store.

For example, change the line `com.ibm.ssl.trustStore=${user.root}/etc/trust.p12` to `com.ibm.ssl.trustStore=deployment_manager_profile/config/cells/cell_name/trust.p12`. Remember the original value. You change this value back to the original value after you run the script.

3. Run the `bin/retrieveSigners.sh` script from the first cell, including information for the second cell in the script.

For example:

```
retrieveSigners.sh CellDefaultTrustStore ClientDefaultTrustStore -autoAcceptBootstrapSigner -conntype SOAP
-port 8879 -host secondmgr.host.ibm.com
```

4. On the first cell, edit the `deployment_manager_profile/properties/ssl.client.props` file and change the value back to the original `com.ibm.ssl.trustStore` value.
5. On the second deployment manager, check the `deployment_manager_profile/properties/ssl.client.props` file that the `com.ibm.ssl.trustStore` value is the cell-level default trust store. Run the `bin/retrieveSigners.sh` script from the second cell, including information for the first cell.
6. On the second cell, edit the `deployment_manager_profile/properties/ssl.client.props` file to change back to the original `com.ibm.ssl.trustStore` value.
7. Restart all of the cells that you are configuring or ensure that all of the cells have been fully synchronized.

Results

The two cells can establish SSL connections with each other.

Configuring an ODR to dynamically update the web server plug-in configuration

In a topology with a web server which forwards requests to an on demand router (ODR), the ODR can be set to dynamically generate and update the web server plug-in configuration file `plugin-cfg.xml`. By configuring an ODR to dynamically update `plugin-cfg.xml`, you avoid manually updating the file during administrative actions, which can affect the content.

Before you begin

Consider the environment topology before configuring the `plugin-cfg.xml` generation; each ODR can generate a `plugin-cfg.xml` file which includes all URIs that it can handle. The generated `plugin-cfg.xml` file can be configured to: `all`, `cell`, `node`, and `server`. The configuration scope determines which ODRs are included in `plugin-cfg.xml`, and therefore, what ODRs a web server forwards requests to. By default, `plugin-cfg.xml` includes only running ODRs.

The `plugin-cfg.xml` configuration scope can be:

- `All`: Includes all ODRs in the environment.
- `Cell`: Includes all ODRs within the same cell as the ODR that generates `plugin-cfg.xml`.
- `Node`: Includes all ODRs on the same node as the ODR that generates `plugin-cfg.xml`.
- `Server`: Includes only the ODR that generates `plugin-cfg.xml`.

About this task

An ODR configured to generate `plugin-cfg.xml` must be running in order for dynamic updates to occur. If a single ODR is configured to generate the `plugin-cfg.xml` file, and that ODR is stopped, `plugin-cfg.xml` is not regenerated until the ODR is restarted. Therefore, run the `plugin-cfg.xml` file generator as a highly available service. For more information, read about generating the plug-in configuration in a high availability environment.

The `plugin-cfg.xml` file is generated by an ODR process. Therefore the file is generated on the file system that hosts the generating ODR. For `all`, `cell`, and `node`, `WAS_HOME/profiles/profile_name/etc/` is the default directory where the ODR generates the `plugin-cfg.xml` file. The default directory for `server` is `WAS_HOME/profiles/profile_name/etc/odr_name/`. The default generation directory can be overridden by

setting a Java virtual machine (JVM) system property on the generating ODR. To set the JVM system property, go to **Servers > Server types > On demand routers > ODR_name > Java and process management > Process definition > Java Virtual Machine > Custom properties > New**. Name the JVM system property `ODCPluginXmlDir` and enter a value that contains the absolute path to the generation directory.

Procedure

1. In the administrative console, go to the **On demand router settings** page:
 - For stand-alone ODR processes, click **Servers > Server types > On demand routers > ODR_name > On demand router properties > On demand router settings**.
 - For a static cluster of ODR processes, click **Servers > Clusters > On demand router clusters > ODR_static_cluster_name > On demand router properties > On demand router settings**.
 - For a dynamic cluster of ODR processes, go to **Servers > Clusters > Dynamic clusters > ODR_dynamic_cluster_name > Server template > > On demand router properties > On demand router settings**.
2. In the **Proxy plugin configuration policy** section, select the scope from the **Generate plugin configuration** drop-down list.
3. Optional: In the **Proxy plugin configuration policy** section, define an absolute path to the script that you enter in the **Plugin config change script** text box. The defined script is invoked every time the `plugin-cfg.xml` is dynamically updated. This approach promotes the newly generated `plugin-cfg.xml` to a remote web server that does not reside on the same system as the ODR.
4. Click **Apply**, and save and synchronize the changes.
5. Restart the ODR for the changes to take effect. `plugin-cfg.xml` is generated every time the ODR starts, when there are administrative actions performed that impact what URIs the ODR handles, and when changes in the environment affect the content of the `plugin-cfg.xml` file.

Results

The ODR automatically updates the `plugin-cfg.xml` file when configuration changes occur.

What to do next

You can override the `plugin-cfg.xml` default attribute values automatically generated by the ODR. For more information, read about controlling the generation of the `plugin-cfg.xml` file.

Generating the plug-in configuration in a high availability environment

You can configure the web server plug-in configuration to generate in a high availability environment to ensure that the configuration is always generated by one process in the cell regardless of other processes that might not be available.

Before you begin

- Establish your high availability environment. For more information, read about setting up a high availability environment.
- Using the high availability plugin-cfg generation service does not automatically disable the on demand router (ODR) plugin-cfg generation if it is enabled. The two services are independent. It is redundant to enable both services, so any ODR plugin-cfg generation that was configured should be disabled when enabling the high availability plugin-cfg generation service. For more information, read about configuring an ODR to dynamically update the web server plug-in configuration.

About this task

Define a set of custom properties at the cell level for each `plugin-cfg.xml` file that you want to generate. The custom properties are formatted as `ODCPluginCfg<property>_<config>=value`, where `<property>` is one of the following elements:

- OdrList
- OutputPath
- UpdateScript
- OdrClusterList
- Disabled

The <config> variable is the name of the configuration that you choose. As a result, properties ending in the same <config> variable define a single generation definition within the high availability plugin configuration generation service.

Procedure

1. In the administrative console, click **System administration > Cell > Custom properties > New**.
2. Define the following custom properties:

ODCPluginCfgOdrList_<configName>=cell1:node1:odr1,cell2:node2:*

This property specifies the ODRs that you want to include in the plugin-cfg.xml file. Use the * symbol as a valid wildcard for each path segment.

ODCPluginCfgOutputPath_<configName>=/path/file_name.txt

This property specifies the location in which the plugin-cfg.xml file is placed after the file is generated. Because you can generate the plug-in configuration on any node in the cell, you must ensure the output directory exists on each node.

ODCPluginCfgUpdateScript_<configName>=/path/script <parameter1> <parameter2>

This property defines the absolute path to your script and the arguments to be passed to your defined script. The defined script will be invoked each time a plugin-cfg.xml is generated.

ODCPluginCfgOdrClusterList_<configName>=cell1:cluster1

This property specifies a cluster of ODRs that you want to include in the plugin-cfg.xml file.

ODCPluginCfgDisabled_<configName>

This property disables the generation of a particular configuration without undefining all of the properties for that configuration. The default value is false.

ODCPluginCfgOdrSessionIdCookie_<configName>

Defines the name of the cookie used to maintain IBM HTTP server/ODR affinity when using ODR cell affinity.

ODCPluginCfgIHSCfgProperties_<configName>

This property is used instead of configuring JVM properties (as is done during non-HA plugin-cfg generation) to set IBM HTTP server specific configuration properties. The value of this property is a comma separated list of ATTRIBUTE_NAME=value pairs where ATTRIBUTE_NAME is the name of an attribute represented in the plugin-cfg.xml.

For example, if a configuration name of 1 is being used a cell property named ODCPluginCfgIHSCfgProperties_1 with a value of TrustedProxyEnable=true,LogLevel=INFO,CloneSeparatorChange=true,ServerIOTimeout=60 would be created to set the TrustedProxyEnable, LogeLevel, CloneSeparatorChange, and ServerIOTimeout attributes contained in the generated plugin-cfg.xml.

ODCPluginCfgTrustedProxyList_<configName>=trustedproxy1,trustedproxy2

This property is used to specify the trusted proxies that you want to include in the plugin-cfg.xml file.

See Controlling the generation of the plugin-cfg.xml file for the list of valid property names and values.

Set the following cell custom properties to generate the plug-in configuration for a collection of ODRs that are not in an ODR cluster. Note that all properties end with _1, which ties them together into a single configuration.

Table 4. Custom properties used to generate a plug-in configuration for ODRs that are not in a cluster

Property name	Property value	Description
ODCPluginCfgOdrList_1	myCell:*.*	Generates a plug-in configuration that will route to all ODRs in the myCell cell.
ODCPluginCfgOutputPath_1	/tmp/plugin-cfg1.xml	Writes the generated plug-in configuration to the /tmp/plugin-cfg1.xml file.
ODCPluginCfgUpdateScript_1	/root/bin/pluginCfgUpdate1	The path to the script that will be invoked each time the /tmp/plugin-cfg1.xml file is updated.

Set the following custom properties to generate the plug-in configuration for a cluster of ODRs named myCell/myOdrCluster. Note that all properties end with _2, which ties them together into a single configuration.

Table 5. Custom properties used to generate the plug-in configuration for myCell/myOdrCluster

Property name	Property value	Description
ODCPluginCfgOdrClusterList_2	myCell:myOdrCluster	Generates a plug-in configuration that will route to all ODRs of the myOdrCluster cluster in the myCell cell.
ODCPluginCfgOutputPath_2	/tmp/plugin-cfg2.xml	Writes the generated plug-in configuration to the /tmp/plugin-cfg2.xml file.
ODCPluginCfgUpdateScript_2	/root/bin/pluginCfgUpdate2	The path to the script that is ran each time the /tmp/plugin-cfg2.xml file is updated.
ODCPluginCfgODRIncludeStopped_2	true or false	Includes or excludes stopped ODRs.

What to do next

Because the generation of the plugin-cfg.xml file can occur on any node in the cell, you can determine the specific location in which the generation service so that it only references the one way of determining where the controller is running

In the administrative console, click **Runtime operations > Component stability > Core components**. Verify that HAPPluginCfgGenerator is displayed in the table.

Segregating HTTP traffic by ODR clusters

The high availability plugin-cfg generation service automatically regenerates and propagates the web server plug-in configuration file each time a change is made, which should affect how the plug-in routes. When you use this service, the web server plug-in routes traffic by application, deployment target, or node group to a specific ODR cluster.

Before you begin

The ability to segregate HTTP traffic by ODR cluster is supported only through the use of the high availability plugin-cfg generation service. It is not supported when a specific ODR is configured to generate a plug-in configuration file.

Procedure

Set the following custom property to configure the high availability plugin-cfg generation service and direct traffic to different ODR clusters.

1. In the administrative console, select **System administration > Cell > Custom properties > New**.
2. Specify the name of the custom property as `ODCPluginCfgRoutingPolicy_<config>`. The `<config>` variable is the name of the configuration that you choose. As a result, the properties end in the same variable to define a single configuration for the generation of the high availability `plugin-cfg.xml` file.
3. Specify the value of the custom property as a comma-separated list of elements formatted as `<webModuleSpec>=ODR=<odrClusterName>`. You can specify the `<webModuleSpec>` portion of the value as one of the following formats:
 - `WebModule:<cell>[/<application>/[<edition>/[<webModule>]]]`

Specifies one or more web modules. Note that the brackets indicate optional elements. If any part is omitted or if the value of any part equals `""`, it matches any value. For example, a custom property named `ODCPluginCfgRoutingPolicy_1` with a value of `WebModule:myCell/myApp1=ODR=odrCluster1,WebModule:mycell/myApp2=ODR=odrCluster2` routes traffic for the `myApp1` application through `odrCluster1` and traffic for the `myApp2` application through `odrCluster2`.
 - `Cluster:<cell>/<cluster>`

Specifies web modules that are deployed to the cluster. For example, a custom property named `ODCPluginCfgRoutingPolicy_1` with a value of `Cluster:myCell/myCluster1=ODR=odrCluster1,Cluster:myCell/myCluster2=ODR=odrCluster2` routes traffic targeted for any application, which is deployed to `myCluster1`, through `odrCluster1`. Similarly, traffic targeted for any application that is deployed to `myCluster2` is routed through `odrCluster2`.
 - `NodeGroup:<cell>/<nodeGroup>`

Specifies web modules that are deployed to one or more nodes in the node group. For example, a custom property named `ODCPluginCfgRoutingPolicy_1` with a value of `NodeGroup:myCell/myNodeGroup1=ODR=odrCluster1,Cluster:myCell/myNodeGroup2=ODR=odrCluster2` routes traffic targeted for any application, which is deployed to a node in `nodeGroup1`, through `odrCluster1`. Traffic targeted for any application that is deployed to a node in `nodeGroup2` is routed through `odrCluster2`.
 - `VirtualHost:<cell>[/<VirtualHost>]`

Specifies all of the web modules that are deployed to the virtual host. For example, a custom property named `ODCPluginCfgRoutingPolicy_1` with a value of `VirtualHost:myCell/default_host=ODR=odrCluster1,VirtualHost:myCell/special_host=ODR=odrCluster2` routes traffic for applications that are deployed to `default_host` through `odrCluster1`. Traffic targeted for applications that are deployed to `special_host` is routed through `odrCluster2`.

You can specify the `<odrClusterName>` variable as a simple cluster name, or you can format it as `<cell>/<cluster>` to reference an ODR cluster in a different cell from which the service is running. In general, the value is read as meaning all traffic that matches the `<webModuleSpec>` variable is routed through the ODR cluster named `<odrClusterName>`.

Controlling the generation of the plugin-cfg.xml file

You can override the default attribute values of the `plugin-cfg.xml` file that is automatically generated by the on demand router (ODR).

The following elements are formatted as `com.ibm.ws.odr.plugincfg.<element>.<attributeName>`, where `<element>` is the name of the configuration file keyword that denotes the element that contains an attribute, and `<attributeName>` is the name of the attribute that is located within that element. Configure the elements as JVM system properties. To set a JVM system property on the ODR in the administrative console, click **Servers > On demand routers > on_demand_router > Java and Process Management > Process definition > Java Virtual Machine > Custom properties > New**.

Config

This element starts the plug-in configuration file. It can include one or more of the following elements and attributes.

com.ibm.ws.odr.pluginfg.config.ASDisableNagle

Specifies whether the user wants to disable Nagle's algorithm for the connection between the plug-in and the server.

The default value is `false`.

com.ibm.ws.odr.pluginfg.config.AcceptAllContent

Specifies whether or not you can include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header. You can specify one of the following values for this attribute:

- True if content is to be expected and read for all requests
- False if content only is only to be expected and read for POST and PUT requests.

The default value is `false`.

com.ibm.ws.odr.pluginfg.config.AppServerPortPreference

Specifies which port number is used to build URI's for a sendRedirect.

The default value is `HostHeader`.

com.ibm.ws.odr.pluginfg.config.ChunkedResponse

Specifies whether the plug-in groups the response to the client when a Transfer-Encoding : Chunked response header is present in the response.

You can specify one of the following values for this attribute:

- True if the plug-in is to chunk the response to the client when a Transfer-Encoding : Chunked response header is present in the response.
- `false` if the response is not to be chunked.

The default value is `false`.

com.ibm.ws.odr.pluginfg.config.IISDisableNagle

Specifies if you want to disable nagle algorithm.

The default value is `false`.

com.ibm.ws.odr.pluginfg.config.IISPluginPriority

Specifies the priority in which the Web server loads the plug-in. You can specify one of the following values for this attribute:

- High
- Medium
- Low

The default value is `High`.

com.ibm.ws.odr.pluginfg.config.IgnoreDNSFailures

Specifies if the plug-in ignores DNS failures within a configuration when started. When set to `true`, the plug-in ignores DNS failures within a configuration and starts successfully if at least one server in each ServerCluster resolves the host name. Any server for which the host name is not resolved is marked *unavailable* for the life of the configuration. The host name is not resolved later during the routing of requests. If a DNS failure occurs, a log message is written to the plug-in log file and the plug-in continues initializing instead of the Web server not starting.

The default value is `false`.

com.ibm.ws.odr.pluginfg.config.RefreshInterval

The time interval, in seconds, at which the plug-in should check the configuration file for updates or changes. The plug-in checks the file for any modifications that occur since the plug-in configuration was loaded.

The default value is `60`. In a development environment in which changes are frequent, set the time interval to less than 60 seconds. In production, set a higher value than the default, because updates to the configuration do not occur as often. If the plug-in reload is not successful, the

plug-in log file contains a message, and the previous configuration is used until the plug-in configuration file successfully reloads. Refer to the plug-in log file for more information if an error occurs.

com.ibm.ws.odr.pluginconfig.config.ResponseChunkSize

The plug-in file reads the response body in 64k chunks until all of the response data is read, which causes a performance problem for requests whose response body contains large amounts of data.

Use this attribute to specify the maximum chunk size to use when reading the response body. For example, `Config ResponseChunkSize="N">`, where N equals the chunk size in kilobytes.

If the content length of the response body is unknown, a buffer size of N kilobytes is allocated and the body is read in N kilobyte size chunks, until the entire body is read. If the content length is known, then a buffer size of either content length or N is used to read the response body.

The default chunk size is 64k.

com.ibm.ws.odr.pluginconfig.config.VHostMatchingCompat

Specifies to use the port number for virtual host matching. The following values can be specified:

- True for physically matching by using the port number for which the request is received.
- False for logically matching by using the port number contained in the host header.

The default is False.

com.ibm.ws.odr.pluginconfig.odrIncludeStopped

Specifies to allow the inclusion of stopped ODRs. The following values can be specified:

- True for including stopped ODRs.
- False for not including stopped ODRs.

The default is False.

com.ibm.ws.odr.pluginconfig.config.TrustedProxyEnable

Specifies the trusted proxies. The following values can be specified:

- True for including trusted proxies.
- False for not including trusted proxies.

The default is False. The trusted proxies are collected from the defined trusted security proxies on the **On demand router properties > On demand router settings** administrative console page.

Log

The log describes the location and level of log messages that are written by the plug-in. If a log is not specified within the configuration file, the log messages might be written to the error log.

com.ibm.ws.odr.pluginconfig.log.Name

The fully qualified path to the log file to which the plug-in writes error messages.

The default value is `profileRoot/logs/http_plugin.log`.

com.ibm.ws.odr.pluginconfig.log.LogLevel

The level of detail of the log messages that the plug-in writes to the log. You can specify one of the following values for this attribute:

- Trace. All of the steps in the request process are logged in detail.
- Stats. The server selected for each request and other load balancing information relating to request handling is logged.
- Warn. All warning and error messages resulting from abnormal request processing are logged.
- Error. Only error messages resulting from abnormal request processing are logged.
- Debug. All of the critical steps performed in processing requests are logged.
- Detail. All of the information about requests and responses are logged.

The default value is Error.

CAUTION: A lot of messages are logged at the trace level, which can cause the file system to fill up very quickly. Never use a trace setting in a normally functioning environment as it adversely affects performance.

ServerCluster

This element is a group of servers that are generally configured to service the same type of requests.

In the simplest case, the cluster contains only one server definition. When you define more than one server, the plug-in completes a load balance across the defined servers by using either a Round Robin or a Random algorithm. The default is Round Robin.

com.ibm.ws.odr.pluginfg.cluster.CloneSeparatorChange

Some pervasive devices cannot handle the colon character (:) that is used to separate clone IDs in conjunction with session affinity. This attribute tells the plug-in to expect the plus character (+) as the clone separator. You must change the server configurations so that the server separates clone IDs with the plus character as well.

The default value is `false`.

com.ibm.ws.odr.pluginfg.cluster.LoadBalance

The Round Robin implementation has a random starting point. The first server is selected randomly, and the Round Robin value is used to select servers from that point forward. This implementation ensures that in multiple process-based Web servers, all of the processes do not start by sending the first request to the same server.

The default load balancing type is Round Robin.

com.ibm.ws.odr.pluginfg.cluster.PostSizeLimit

The maximum number of bytes of request content allowed for the plug-in to attempt to send the request to a server. If a request is received that is greater than this size, the plug-in ends the request.

The default value is `-1` bytes, which indicates that there is no limit for the post size.

com.ibm.ws.odr.pluginfg.cluster.RemoveSpecialHeaders

The plug-in adds special headers to the request before it is forwarded to the server. These headers store information about the request that the application then uses. By default, the plug-in removes these headers from incoming requests before adding the required headers.

The default value is `true`. If you set the attribute to `false`, you introduce a potential security exposure by not removing headers from incoming requests.

com.ibm.ws.odr.pluginfg.cluster.RetryInterval

An integer that specifies the length of time that elapses from the time a server is marked down to the time that the plug-in reattempts to make a connection.

The default value is `60` seconds.

Server

This element is a server instance that is configured to handle requests routed to it based on the routing rules of the plug-in configuration. The server element corresponds to an application server running on either the local workstation or a remote workstation.

com.ibm.ws.odr.pluginfg.server.ServerIOTimeout

Enables the plug-in to set a timeout value, in seconds, for sending requests to and reading responses from the server.

If a value is not set, the plug-in, by default, uses blocked I/O to write request to and read response from the server until the TCP connection times out. When selecting a value for this attribute, remember that it might take several minutes for a server to process a request. Setting the value of the `ServerIOTimeout` attribute too low might cause the plug-in to send a false server error response to the client.

The default value is `60`.

For more information about configuring the `ServerIOTimeout` attribute, read the web server plugin configuration technote.

com.ibm.ws.odr.plugincfg.server.ConnectTimeout

Enables the plug-in to perform non-blocking connections with the application server, which are beneficial when the plug-in is unable to contact the destination to determine if the port is available or unavailable.

If no value is specified, the plug-in performs a blocking connect in which the plug-in waits until an operating system times out and allows the plug-in to mark the server *unavailable*. A value greater than 0 specifies the number of seconds the plug-in waits for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server *unavailable*, and proceeds with one of the other servers defined in the cluster.

The default value is 0.

com.ibm.ws.odr.plugincfg.server.ExtendedHandShake

Used when a proxy firewall is between the plug-in and the application server. In such a case, the plug-in is not failing over, as expected.

The plug-in marks a server as stopped when the connect() ends. However, when a proxy firewall is in between the plug-in and the application server, the connect() succeeds, even though the back-end application server is stopped. This causes the plug-in to not failover correctly to other application servers.

The plug-in contains this attribute to ensure that it is started before sending the request.

The default value is `false`.

com.ibm.ws.odr.plugincfg.server.MaxConnections

Specifies the maximum number of pending connections to a server that flows through a Web server process at any point in time.

The default value is set to -1.

com.ibm.ws.odr.plugincfg.cluster.WaitForContinue

Specifies whether to use the HTTP 1.1 100 Continue support before sending the request content to the application server. The default value is `false`. The plug-in does not wait for the 100 Continue response from the application server before sending the request content.

This property is ignored for POST requests to prevent a failure from occurring if the application server closes a connection because of a time-out.

Enable this function when configuring the plug-in to work with certain types of proxy firewalls.

Property**com.ibm.ws.odr.plugincfg.property.ESIEnable**

Used to enable or disable the Edge Side Include (ESI) processor. If the ESI processor is disabled, the other ESI elements in the file are ignored.

The default value is `true`.

com.ibm.ws.odr.plugincfg.property.ESIMaxCacheSize

An integer specifying, in 1K byte units, the maximum size of the cache. The default maximum size of the cache is 1024K bytes (1 megabyte). If the cache is full, the first entry to be evicted from the cache is the entry that is closest to its expiration time.

com.ibm.ws.odr.plugincfg.property.ESIInvalidationMonitor

Indicates whether or not the ESI processor receives invalidations from the application server.

The default value is `false`.

com.ibm.ws.odr.plugincfg.property.https.keyring

Use this element to specify the initialization parameters when the protocol of the transport is set to HTTPS.

This attribute is a supported name that is recognized by the transport. The default value for the keyring element is `profileRoot/etc/plugin-key.kdb`.

com.ibm.ws.odr.plugincfg.property.https.stashfile

The default value for the stash file is `profileRoot/node/etc/plugin-key.sth`

com.ibm.ws.odr.plugincfg.property.PluginInstallRoot

Specifies the installation path for the plug-in. The default value is "". Set the value, however, to the fully qualified path of the plug-in installation root, or the property is not displayed in the plugin-cfg.xml file.

Routing requests directly from a web server to a back-end application server

Define a new custom property to configure the web server plug-in configuration to route requests directly from the web server to a back-end application server. You can configure certain requests to route through an on demand router (ODR) and configure other requests to route directly to a back-end application server.

About this task

Typically, an ODR acts as a gateway through which requests sent by a web server flow to a back-end application server. You can define the `ODR_Module_Routing_Policy` custom property for the `plugin-cfg.xml` file to redirect specific requests that you do not want to route through the ODR. Instead, the requests are routed directly from the web server to the back-end server. Alternatively, you can reset the custom property so that the ODR resumes intercepting requests.

The format of the custom property value is a comma-separated list of module paths, such as `cell_name/app_name/edition/module_name=value`.

Procedure

1. In the administrative console, click **System administration > Cell > Custom properties > New**.
2. Type `ODR_Module_Routing_Policy` as the name of the custom property.
3. Type the value of the custom property.
 - Set the value to `cell_name/app_name/edition/module_name=direct` to route requests directly to the back-end server.
 - Set the value to `cell_name/app_name/edition/module_name=ODR` to route requests through the ODR before the back-end server receives the requests.

For example, if you set the value to `cell/app/edition/module=direct,cell/app2/edition/module=ODR`, each module is configured independently as to whether requests for that module are sent through the ODR or directly to the back-end server. You can use a wildcard (*) in place of the `app_name`, `edition`, and `module_name` variables.

4. Click **Apply** and save your changes.

Example

In the following example, the custom property is set to route requests to a back-end application server. A wildcard is used in place of the `app_name`, `edition`, and `module_name` variables.

```
myCell/*/*/*=direct
```

Configuring passive HTTP session affinity in the on demand router

When the on demand router (ODR) processes a request, it obtains the session affinity descriptor policy of the cluster to which the server belongs. If you changed the default settings for any middleware servers, you might need to update the middleware descriptor properties so that the ODR can obtain the descriptor policy. In most cases, the on demand router (ODR) does not require configuration to support HTTP session affinity. However, some special cases exist where you must configure the ODR to learn about backup servers that the back end servers might be setting on the session affinity cookie.

Before you begin

The servers in your configuration must be in a generic server cluster or a dynamic cluster. You can use passive HTTP session affinity with a static cluster.

About this task

In environments where the ODR forwards requests to generic server cluster members and to non-federated WebSphere Application Server servers, a set of properties must be set for the ODR to correctly uphold session affinity. Passive HTTP session affinity means that the ODR passes the session cookie set by the backend server through to the client, as opposed to the ODR setting the WSJSESSIONID cookie. Passive HTTP session affinity is used in the following situations:

- When the ODR routes to servers that are not running WebSphere Application Server middleware products.
- When the ODR routes to WebSphere Application Server application servers that are in different core groups that are not connected by the core group bridge.
- When the application uses Java Platform, Enterprise Edition (Java EE) HTTP session affinity that is not standard. For example, the application's session ID cookie name is something other than JSESSIONID

Procedure

1. If any of the default values for the server have changed, modify the session affinity descriptor. In the administrative console, click **System administration > Middleware descriptors > middleware_server > default**.
2. Define the session affinity descriptor properties. Modify the values for any of the following fields that apply:
 - **Learn clone IDs**
 - **Cookie names**
 - **URL rewrite**
 - **Clone ID separator**
 - **Alternate clone ID separator**
 - **Affinity mode**

Set the value of the **Learn clone IDs** field to true so that the ODR parses the clone IDs from the response cookie that are sent back to the client. Because the ODR recognizes the server that sends the response back at this point, the parsed clone ID is associated with the server. Therefore, future requests are matched against the known set of clone IDs in order to uphold session affinity in other middleware server environments. Set the **Learn clone IDs** field to true when the ODR does not have an on demand configuration of the server. Note that the ODR can only parse the response cookie if the session ID is in a JSESSIONID format that the ODR understands.

The **Cookie names** field indicates which response header contains session ID information, and should be parsed to determine the clone ID. The **Clone ID separator** field indicates on which part of the session cookie the **Clone ID** field begins. The **Cookie names** and **Clone ID separator** fields are also used by the ODR to parse the clone IDs from the request cookie to enforce session affinity.

In cases where there is no on demand configuration information for servers, such as servers that are members of generic server clusters, set the **Learn clone IDs** field to true so that the ODR parses the session ID for the clone ID. If the session ID in the response is not in the JSESSIONID format, you must set the affinity mode to Active[-conditional] affinity. In this case, the ODR internally assigns each backend application server a clone ID, which is set in the WSJSESSIONID header. As a result, the ODR maintains session affinity when operating with backend environments that cannot generate session IDs in the JSESSIONID format. Active affinity means that the ODR always sets a WSJSESSIONID cookie with the clone ID of the backend server that is sending the response. Active-conditional affinity means that the ODR sets only the WSJSESSIONID cookie if it recognizes the Set-Cookie header in the response.

In WebSphere Application Server environments where the clone IDs are available to the ODR by way of the on demand configuration, the clone ID information is never learned by setting the **Learn clone IDs** field to true. The clone IDs are available to the ODR by way of the on demand configuration, if the

ODR is in the same core group as the application servers, if the ODR is in a different core group but the core groups are bridged, or if the bulletin board service overly network (BBSO) is enabled. BBSO is enabled by default.

Results

When the ODR processes a request, it obtains the session affinity descriptor policy that is configured for the cluster to which the server belongs. The method in which the server clone identification is obtained depends on the property values of the policy attributes.

Configuring the on demand router for multi-cluster failover and load balancing routing

To configure the on demand router (ODR) to route requests to a different cluster, you can configure custom properties for configuring multi-cluster failover and load balancing routing policies. You might route requests to a cluster in another cell if your primary cluster fails, to balance the load in your environment between multiple clusters, or to direct the ODR to route requests to a specific cluster.

Before you begin

- Create the ODR for your cell. For more information, read about creating ODRs.
- Configure and deploy the clusters and applications for your multi-cluster policy.
- If the clusters are in different cells, configure the core group bridge so that they can communicate, or use a generic server cluster definition.

About this task

Use multi-cluster routing policies for failover and load balancing. With *multi-cluster failover*, you can specify a cluster to take over the workload when the primary cluster fails. With *load balancing routing*, you can balance the request loads between multiple clusters.

Procedure

1. Create a custom property for the multi-cluster routing policy. In the administrative console, click **Servers > Server types > On demand routers > ODR_name > On demand router properties > On demand router settings > Custom properties > New** .
2. Type a name for the multi-cluster routing policy in the **Name** field. The name must start with the token MCRP@ string. The full syntax of the name field follows:

```
MCRP@cell_name[$application_name[$web_module_name[$cluster_name]]]
```

Note: The recommendation for configuring multi-cluster failover and load balancing routing policies is through the wsadmin tasks, unless you already have it configured through a custom property. For more information, read about rules for ODR routing policy administrative tasks.

Table 6. Components of the name field syntax

Option	Description
MCRP	Specifies that the custom property is a multi-cluster routing policy (MCRP). This prefix must be specified in uppercase letters.
@	Required symbol. This symbol is the separator between the policy name and the cell. In this configuration, it is generally used to separate a policy name attribute from a cell name.
cell_name	Specifies the name of the cell. This cell must be a valid cell that runs WebSphere Application Server. The case and spelling must match the WebSphere Application Server configuration.

Table 6. Components of the name field syntax (continued)

Option	Description
\$	Separates the WebSphere Application Server objects.
<i>application_name</i>	Specifies the application name without the file extension. For example, if the enterprise application name is StockTrade.ear, then specify StockTrade as the <i>application_name</i> value.
<i>web_module_name</i>	Specifies the name of the Web module without the .war file extension.
<i>cluster_name</i>	Specifies the name of the cluster in which the application is deployed.
[]	Indicates variables that are optional.

3. Type a value in the **Value** field. The full syntax of the value field follows:

policy_type@cell_name1\$cluster_name1[,cell_name2\$cluster_name2,...]

Table 7. Components of the value field syntax

Option	Description
<i>policy_type</i>	<p>The <i>policy_type</i> value is not case sensitive. The <i>failover</i>, <i>wlor</i>, or <i>wrr</i> values can be specified in uppercase or lowercase.</p> <p>Valid values:</p> <p>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 route only to the configured cell and cluster when the primary cell is down. The cell status is indicated by an HTTP status code of 503, <i>service unavailable</i>.</p> <p>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. This load balancing policy not only considers weights, but also how many outstanding HTTP requests exist in a cluster. This policy will more efficiently distribute requests to clusters that can handle them. <i>wlor</i> is recommended over <i>wrr</i>.</p> <p>New weight values are obtained every 15 seconds from the dynamic workload manager (DWLM), which takes into account the application level response time. Use the <i>mcrp.ui</i> system property to set the new update time in seconds.</p> <p>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. .</p> <p>New weight values are obtained every 15 seconds from the dynamic workload manager (DWLM), which takes into account the application level response time. Use the <i>mcrp.ui</i> system property to set the new update time in seconds</p>

Table 7. Components of the value field syntax (continued)

Option	Description
<i>cell_name</i>	Specifies the name of the cell. This cell must be a valid cell that runs WebSphere Application Server. The case and spelling must match the cell name in WebSphere Application Server.
<i>cluster_name</i>	The cluster names can be names of clusters or dynamic clusters in the local cell, a cluster that is in a cell that is bridged with the core group bridge service, or a generic server cluster. The cluster name value must be capitalized and spelled the same way that you specified the name when you created the cluster in the administrative console.
,	The comma (,) is used to separate a set of values in the list.

All Java Platform, Enterprise Edition (Java EE) artifact names such as *cell_name*, *application_name*, and *cluster_name* must be spelled the way that they were spelled in the WebSphere Application Server configuration.

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.

Examples for the value field follow: The following policy configures a failover policy. When a failure occurs, the requests can fail over to the myGSC1 generic server cluster in the thesaharaCell01 cell:

```
failover@myCell01$myCluster1,myCell01$myGSC1
```

The following policy configures a weighted least outstanding request load balancing policy.

```
wlor@thesaharaCell01$myCluster1,myCell2$myCluster2
```

The following policy configures a weighted round robin policy.

```
wrr@thesaharaCell01$myNYCGSC,cell_2$cluster_2
```

The following value balances load across all of the cell and cluster combinations where the configured application is deployed.

```
wrr@*$*
```

4. Click **Apply** or **OK** to commit your new custom settings.

Results

The ODR routes to multiple clusters, as you configured in the multi-cluster routing policy.

Configuring ODR OutOfMemory prevention

You can create cell-level custom properties to prevent the on demand router (ODR) from running out of memory when the number of concurrent requests sent to the ODR increases.

About this task

Because the ODR is completely asynchronous, it can scale to an extremely high number of concurrent connections. If application server threads begin to slow down or hang due to some condition, such as a slow database, the number of concurrent requests through a single ODR can increase dramatically. To prevent the ODR from running out of memory due to the large number of connections that can accrue in the ODR in this scenario, the ODR automatically starts rejecting requests, and thus closing connections,

when the heap utilization exceeds a maximum threshold.

Procedure

1. Set the following custom property to configure a percentage value that determines at what amount of heap usage the ODR rejects requests. If heap usage exceeds 90%, the ODR rejects any incoming request and a 503 error code is returned.
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as `ODR.heapUsage.max`.
 - c. Specify a value for the custom property. The default value is 90. To disable this feature, set the value to 100.
2. Set the following custom property to configure the type of error code that is returned when a request is received and the heap usage exceeds the maximum threshold.
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as `ODR.heapUsage.errorCode`.
 - c. Specify a value for the custom property. The default value is 503.

Creating dynamic clusters

When you create a dynamic cluster, the workload of the cluster members is dynamically balanced based on performance information collected from the cluster members. Creating dynamic clusters enables application server virtualization.

Before you begin

Complete the following prerequisites before creating a dynamic cluster:

- You must have configurator administrative privileges to create a dynamic cluster.
- Verify that the application placement controller is enabled. The application placement controller is enabled by default. The application placement controller enables the autonomic capabilities of dynamic clusters. To enable the application placement controller, click **Operational policies > Autonomic managers > Application placement controller**. For more information about the properties that you can change on the application placement controller, read about monitoring and tuning the application placement controller.
- Decide if you are going to use vertical stacking. Vertical stacking can improve bottleneck conditions in deployed applications by enabling the placement controller to start more than one instance of the dynamic cluster on a node. With vertical stacking enabled, the autonomic managers limit the processor percentage that is used by each stacked instance. The general formula is $100\% / \text{max-number-of-stacked-instances}$. For example, if you configure three stacked instances, the workload is throttled to prevent any single instance from using more than 33% of the processor capacity. For more information, read about configuring vertical stacking.
- If you are creating a dynamic cluster of externally created middleware servers with assisted life-cycle management, create representations of these servers in the product environment before you create a dynamic cluster. All of these servers must have the same applications installed and have the same version of middleware software installed. For more information, read about adding assisted life-cycle middleware servers.

About this task

A dynamic cluster is an application deployment target that can expand and contract depending on the workload in your environment. Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload manager to maximize the use of your computing resources. Dynamic clusters are required for many of the product autonomic functions, including high availability and service policies.

If you already have over 40 servers in your core group, you can use the `coregroupsplit.py` script to split your existing cell into multiple core groups. For more information, read about the `coregroupsplit.py` script.

Procedure

1. Create the dynamic cluster. In the administrative console, click **Servers > Clusters > Dynamic clusters > New**.
2. Select the dynamic cluster server type. The dynamic cluster server type determines the type of servers that are members of this dynamic cluster. Depending on the type, you enter the name of the dynamic cluster on this panel or when you select the membership method.
3. For some dynamic cluster types, you can select the membership method. The membership method defines how servers join the dynamic cluster as cluster instances.

Option	Description
Automatically define cluster members with rules	You can Automatically define cluster members with rules if you are using servers with complete life-cycle management. With this option, you create a membership policy that defines the nodes on which cluster instances can be placed.
Manually define cluster members	If you are using servers with assisted life-cycle management, you can Manually define cluster members . With this option, you select existing servers to add to the dynamic cluster. The servers that you select must be homogenous: that is, they must be of the same server type, middleware server version, and have the same applications installed. Note: To add a new middleware server to an existing dynamic cluster when one or more applications are targeted to the dynamic cluster, you must install the applications on the middleware server and target the applications to the server before you add the server as member of the dynamic cluster.

4. Define the dynamic cluster members.
 - If you selected **Automatically define cluster members with rules** in the previous step, use the subexpression builder to build a membership policy expression for your dynamic cluster. This expression is compared to all the nodes in the cell, selecting any nodes for which the subexpression is true.
 - If you selected **Manually define cluster members**, the action you take depends on what kind of servers are in your dynamic cluster.
 - If your server type is application server, choose an existing static cluster to convert to a dynamic cluster.
 - If your server type is an externally created middleware server with assisted life-cycle management, select servers from the list, and add them to your dynamic cluster. Before you add assisted life-cycle servers, be sure that they are the same type, middleware server version, and have the same set of applications installed.

Note: Before you can add a new middleware server to an existing dynamic cluster when one or more applications are targeted to the dynamic cluster, you must install the applications on the middleware server and target the applications to the server.

 - a. Deploy your unmanaged application to the middleware server. For more information read about deploying unmanaged Web applications.
 - b. Define the deployment target for the application. In the administrative console, click **Applications > All applications > *unmanaged_app_name*** . Select the target and click **Add**.

- c. Click **Apply** and save your changes.
5. Select a dynamic cluster server template.

You can select a dynamic cluster server template only for dynamic clusters that consist of servers with complete life-cycle management. You can choose an existing predefined template, or create your own server templates to use when you create your dynamic cluster. Read about creating server templates.

Note: If your configuration consists of mixed versions of WebSphere Application Server Network Deployment, specifically the deployment manager is at a higher version than the version of the node, you cannot use a predefined server template to create a dynamic cluster. If you are running a Version 6.1 node and a Version 7.0 deployment manager, for example, you can create a static cluster of servers on the node, and then convert the static cluster to a dynamic cluster.

Note: The defaultXD and defaultXDZOS server templates that are used when creating a dynamic cluster are deprecated. Use the default server template instead.

6. Specify dynamic cluster-specific properties.
 - a. Define the minimum number of cluster instances. The default minimum number of instances is one instance and the maximum default is no limit on instances. If a minimum value is excessive, performance degradation might occur.
 - b. Define the maximum number of cluster instances. The default value has no limit on the number of cluster instances.
 - c. Determine whether to enable vertical stacking. When you configure vertical stacking, more than one dynamic cluster instance can start on the same node.
 - d. Specify an isolation preference for the dynamic cluster.
7. Confirm the dynamic cluster creation. Click **Finish** > **Save** to save the changes to the master configuration.
8. Select the mode of operation. In the administrative console, click **Servers** > **Clusters** > **Dynamic clusters**. Select the dynamic clusters that you want to modify. Choose the operational mode and click **Set mode**.

Important: To use dynamic application placement, click **Automatic** or **Supervised** as the mode of operation.

z/OS If you have nodes that run on z/OS systems, use dynamic clusters in supervised or automatic mode. If you want to prevent Intelligent Management from automatically starting a cluster member on a logical partition (LPAR) that is hosting a dynamic cluster when the LPAR processor is too busy, then you must define the `cpuUtilizationThreshold` custom property. In this scenario, the product starts the cluster member only if transaction demand requires additional cluster members and the processor utilization on the LPAR is less than the `cpuUtilizationThreshold` value. For more information read about application placement custom properties.

Results

When the dynamic clusters start, at least one instance of each dynamic cluster in your environment becomes available as soon as possible. Multiple instances on the same node can start concurrently if you have multiple processors on the same node. For example, if you have two processors on a node, two instances can start concurrently. The application placement controller continues to start instances evenly across the nodes for all the dynamic clusters until the minimum number of instances for each dynamic cluster is reached.

Example

Use the following placement scenarios as a guideline for your dynamic cluster settings:

- To specify that exactly two servers are started when the dynamic cluster is running: Click **Keep multiple instances started at all times**. Set the **Number of instances** value to 2. Click **Limit the number of instances that can start** and set the **Number of instances** value to 2.
- To limit the number of started servers to five, and to stop the servers when no activity occurs, click **Stop all instances during periods of inactivity**. Set the maximum number of instances by clicking **Limit the number of instances that can start** and set the **Number of instances** value to 5.
- To keep at least one instance active at all times, and to support an unlimited number of instances to start, click **Keep one instance started at all times** and **Do not limit the number of instances that can start**.

What to do next

- To edit your dynamic cluster settings, click **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name***. If you want to make changes to all of the members of the dynamic cluster, you can edit the dynamic cluster server template. Click **Server template**.
- Deploy an application to your dynamic cluster.
- Monitor performance with reports and operations tabs. Click **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name***. Click the **Reports** tab or the **Operations** tab.
- If you are using supervised mode, the autonomic managers generate recommended actions and runtime tasks upon which you can act. To view all the runtime tasks that the supervised operating mode created, click **System administration > Task management > Runtime tasks**. If you want to avoid monitoring the runtime tasks queue, you can define e-mail notification. You can create an e-mail notification profile by clicking **System administration > Task management > Notifications**.
- If you are using automatic mode, you can prevent servers from starting or restarting during the cell shutdown by adding commands to disable the application placement controller and health controller to the script that you use to stop the cell. For example, you might add the following lines to your script:

```
wsadmin -profile PlacementControllerProcs.jacl -c "disable"
wsadmin -profile HmmControllerProcs.jacl -c "disable"
```

Remember: To change your server ports after the server has been created, you must change the ports on each server instance. You cannot change the ports through the dynamic cluster template. Because multiple servers can be on the same node by configuring vertical stacking, the ports must be unique for each server instance.

Creating a static cluster of ODRs

You can create a static cluster of on demand routers (ODRs). A static cluster is a group of application servers in a WebSphere Application Server Network Deployment environment that participates in workload management.

Before you begin

When an ODR is a member of a cluster, routing rules and other configuration settings that are set at the ODR level are ignored. Configuration settings at the cluster level take precedence over any configuration settings at the server level.

About this task

You might consider creating a static cluster of ODRs when you are integrating the Session Initiation Protocol (SIP) ODR with Load Balancer for IBM WebSphere Application Server.

You might consider creating a static cluster of HTTP ODRs to administer all of the ODRs at one time. If you do not create a static cluster, you must administer each ODR individually.

gotcha: The SIP ODR is not currently recommended for a production environment which requires high availability. If you require SIP high availability in production, use the SIP proxy server instead. If your solution does not require high availability, you can use a non-clustered SIP ODR.

Procedure

1. Create an ODR to use as a template when you create your cluster. You can create the ODR in the administrative console or with the `createodr.jacl` script. For more information, read about creating ODRs, and about the `createodr.jacl` script.

2. Run the following `wsadmin` command to create a static cluster.

```
$AdminTask createCluster {-clusterConfig {-clusterName odr_cluster_name -clusterType ONDEMAND_ROUTER}
-convertServer {-serverNode node_name -serverName my_odr}}
```

For the `my_odr` value, specify the name of the ODR that you created as a template.

3. Add ODRs to the static cluster. Run the following command for each ODR cluster member:

```
$AdminTask createClusterMember {-clusterName odr_cluster_name -memberConfig {-memberNode node_name
-memberName odr_cluster_member_name}}
```

4. Save your changes. Run the following command:

```
$AdminConfig save
```

What to do next

To modify the settings of an ODR static cluster, such as the action type and rule expression, use the `wsadmin` tool to complete administrative tasks at the cluster level. For more information, read about rules for ODR routing policy administrative tasks and rules for ODR service policy administrative tasks.

Creating a dynamic cluster of ODRs

You can create a dynamic cluster of on demand routers (ODRs), which means that the application placement controller selects the best node on which to start the minimum number of ODRs. If an ODR stops for any reason, the application placement controller will then start a new instance.

Before you begin

- The size of an ODR dynamic cluster will not currently increase beyond the minimum number of instances.
- Before running the `manageODR.py` script, ensure that you have the `WAS_HOME` environment variable configured to point to the directory of your WebSphere installation.
- When an ODR is a member of a cluster, routing rules and other configuration settings that are set at the ODR level are ignored. Configuration settings at the cluster level take precedence over any configuration settings at the server level.
- For Session Initiation Protocol (SIP), an ODR cluster defines the other ODRs to which an ODR can route. An ODR might route a UDP message to another ODR to guarantee that server affinity is maintained in the case that the UDP message is a retransmission.

gotcha: The SIP ODR is not currently recommended for a production environment which requires high availability. If you require SIP high availability in production, use the SIP proxy server instead. If your solution does not require high availability, you can use a non-clustered SIP ODR.

About this task

You can configure a dynamic cluster isolation policy to prevent ODRs from running on the same node with application servers. You can use the administrative console, or you can run the `manageODR.py` script to create ODR dynamic clusters. You can also set a cell-level custom property to enable elasticity mode for ODR dynamic clusters.

Procedure

- Create ODR dynamic clusters in the administrative console to create ODR dynamic clusters.
 1. Click **Servers > Clusters > Dynamic clusters > New**.
 2. Select **On demand router** as the dynamic cluster server type.
 3. Select the membership method, which defines how servers join the dynamic cluster as cluster instances. Select **Automatically define cluster members with rules** to create a membership policy that defines the nodes on which cluster instances can be placed. Select **Manually define cluster members** to add existing servers to the dynamic cluster. The servers that you select must be homogenous: that is, they must be of the same server type, middleware server version, and have the same applications installed.
 4. Define the dynamic cluster members.
 5. Select a dynamic cluster template.

Note: The defaultXD and defaultXDZOS server templates that are used when creating a dynamic cluster are deprecated. Use the default server template instead.

6. Specify dynamic cluster-specific properties.
 7. Confirm the creation of the ODR dynamic cluster. Click **Finish > Save** to save the changes to the master configuration.
 8. Select the mode of operation. Select **Servers > Clusters > Dynamic clusters > ODR_dynamic_cluster**. Choose the operational mode, and click **Set mode**.
- Run the manageODR.py script to create ODR dynamic clusters. The manageODR.py script is in the install_root/bin directory.

In the following example, an ODR dynamic cluster named myOdrCluster is created on all nodes in the node group named myOdrNodeGroup.

```
./wsadmin.sh -f manageODR.py createDynamicCluster myOdrCluster myOdrNodeGroup
```

- Set the APC.predictor custom property to enable elasticity mode for ODR dynamic clusters. By setting the custom property, the application placement controller starts and stops servers based on CPU usage alone. Furthermore, the controller no longer retrieves data from the autonomic request flow manager (ARFM) regarding what servers to start and stop.
 1. Click **System administration > Cell > Custom properties > New**.
 2. Specify the name of the custom property as APC.predictor.
 3. Specify the value of the custom property as CPU.
 4. Click **OK**, and save and synchronize your changes.

What to do next

- After you create your ODR dynamic clusters, you can manage them from the Dynamic clusters panel in the administrative console. Click **Servers > Clusters > Dynamic clusters**.
- Monitor performance with reports and operations tabs. Click **Servers > Clusters > Dynamic clusters > ODR_dynamic_cluster**. Select the **Reports** tab or the **Operations** tab.

Dynamic clusters

A dynamic cluster is a server cluster that uses weights and workload management to balance the workloads of its cluster members dynamically, based on performance information that is collected from the cluster members. Dynamic clusters enable application server virtualization.

A *dynamic cluster* is an application deployment target that can expand and contract depending on the workload in your environment. Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload manager to maximize the use of your computing resources. Dynamic clusters are required for many of the Intelligent Management autonomic functions, including high availability and service policies.

For complete life-cycle management servers, the product controls the creation and deletion of server instances and can start and stop servers. For assisted life-cycle management servers, the product can control the state of servers by stopping and starting servers from a pool of predefined server instances.

Dynamic cluster membership

Two options exist for adding members to a dynamic cluster: automatically define cluster members with rules or manually define cluster members.

- Automatically define cluster members with rules

By automatically defining cluster members with rules, you can create a subexpression that automatically selects nodes to host dynamic cluster members based on different node properties. This subexpression is called a *membership policy*. After you create the membership policy, you can preview the node membership before you finish creating the dynamic cluster.

Automatically defining cluster members with rules is available only for servers that have complete life-cycle management. After you create the dynamic cluster with a membership policy, dynamic cluster instances can start on any of the selected nodes. If nodes become available that meet the criteria of your membership policy, dynamic cluster instances can also start on these nodes.

- Manually define cluster members

When you manually define cluster members, you statically define which servers are cluster members by selecting servers to add to the cluster. You use this option instead of the membership policy for the following reasons:

- You have an existing static cluster that you want to convert to a dynamic cluster.
- You are using *assisted life-cycle management servers*. Assisted life-cycle management servers cannot be created from the administrative console. With this option, you create representations of the servers as cluster members. These members must be homogeneous, that is, be all of the same server type, for example, a group of BEA WebLogic servers. The same version of the middleware software must be installed on all the nodes in the dynamic cluster, and you must deploy the same applications to these servers before you create the dynamic cluster.

Server templates

A server template is a copy of a server configuration that can be used as a starting point when a server is added to the dynamic cluster. Predefined templates exist for different middleware server types. You can also define your own server templates.

Dynamic cluster server templates

After you create a dynamic cluster, the *dynamic cluster server template* defines the properties for all of the members in the dynamic cluster.

Note: When you make a change to a dynamic cluster server template, this will overwrite any changes you made that are unique to an individual member of that cluster.

Cluster instances

You can control the creation and management of cluster instances for your dynamic cluster. These options include:

- Creating a minimum and maximum number of cluster instances.
- Stopping cluster instances when other dynamic clusters need resources.
- Allowing more than one cluster instance to start on the same node, also known as *vertical stacking*. With vertical stacking, you can improve bottleneck conditions within an application. The *stacking number* defines how many cluster instances can start on a single node.

- Specifying if cluster instances from other dynamic clusters can start on the same node, also known as *dynamic cluster isolation*.

Operating modes

Dynamic clusters act differently depending on the operating mode. Choose one of the following options for mode of operation:

- **Manual.** In manual mode, the dynamic cluster is no different from the standard application server environments with static clusters. Manual mode does not support application placement, or runtime task suggestions. The autonomic request flow manager and dynamic workload management (DWLM) can work with the cluster.
- **Supervised.** In supervised mode, the environment provides information about required corrective actions by generating runtime tasks. You can accept or deny the recommendations of the autonomic managers in the task management panel in the administrative console. To manage runtime tasks, click **System administration > Task management > Runtime tasks**.
- **Automatic.** In automatic mode, the environment takes corrective actions automatically.

Important: To use dynamic application placement, click **automatic** or **supervised** as the mode of operation.

If you are using **manual** mode for the autonomic request flow manager (ARFM) on either the cell level or deployment target level, then you also must put your dynamic clusters in manual mode. You can also use static clusters when ARFM is in manual mode for the cell or deployment target. If ARFM is in automatic mode, you can use any of the operating modes for the cluster.

Static clusters versus dynamic clusters

Static clusters in a WebSphere Application Server Network Deployment configuration are different from the dynamic clusters that you can define in Intelligent Management. Both types of clusters support workload balancing, however, dynamic clusters are controlled by autonomic managers that can optimize the performance of the cluster.

Static cluster

A static cluster is a group of application servers in a WebSphere Application Server Network Deployment environment that participates in workload management.

Dynamic cluster

A dynamic cluster is a fundamental building block of the dynamic operations environment. You must configure dynamic clusters to get Intelligent Management functionality, such as high availability and service policies. Dynamic cluster members can be started and stopped by the application placement controller when running in supervised or automatic mode. Therefore, dynamic clusters are more scalable than static clusters.

Table 8. *Static clusters versus dynamic clusters*

Characteristic	Static clusters	Dynamic clusters
Membership	You must manually add application servers to static clusters.	You can automatically define cluster members with rules or you can manually define the cluster members. If you are using servers that have complete life cycle management, you can create a membership policy that automatically selects nodes on which to host dynamic cluster members based on different node properties. If you are using servers that have assisted life cycle management, you can statically define the servers that are cluster members.

Table 8. Static clusters versus dynamic clusters (continued)

Characteristic	Static clusters	Dynamic clusters
Cluster management	You define the application servers that are in the static cluster and then start or stop all the application servers in the cluster.	A dynamic cluster can start and stop instances of servers as required. If the dynamic cluster is in automatic mode, then the server instances stop and start automatically. If the dynamic cluster is in supervised mode, runtime tasks generate to advise the administrator to start and stop servers at certain times. In manual mode, the administrator decides when instances of servers are stopped and started.
Cluster templates	When you define a static cluster, you can select an application server template on which to base all the application server instances that you create. However, any changes that you make to the template after creating the instances do not change the instances.	When you define a dynamic cluster, you can define an application server template for the application server instances. After you define the dynamic cluster, you can use the dynamic cluster server template to edit the cluster member properties. The changes are propagated to all of the application server instances.
Application server weights	You explicitly assign a weight value to each application server instance. You can also enable the dynamic workload manager to assign weight values. To enable the dynamic workload manager, click Servers > Clusters > cluster_name > Dynamic WLM.	The dynamic workload manager is enabled by default and assigns weights to the application server instances. To disable the dynamic workload manager, click Servers > Clusters > cluster_name > Dynamic WLM.
Applicability	You can use static clusters in a WebSphere Application Server Network Deployment environment, or an Intelligent Management environment. If you use static clusters in an Intelligent Management environment, then the behavior is identical to static clusters in a WebSphere Application Server Network Deployment environment.	You can use dynamic clusters in an Intelligent Management environment only.

Dynamic cluster isolation

You can use *dynamic cluster isolation* to isolate applications from other applications that are deployed in the cell. For example, you might create a dynamic cluster isolation configuration to isolate the critical applications that an external customer uses from your internal applications, which can tolerate some instability.

Scenario

Your company hosts Web applications for external customers and for internal departments in the company. To provide the most stable and secure service for your external customers, you want to be sure that their applications run on separate computers. Your hosting environment must adhere to the following requirements:

- All customer applications must run on different servers than applications from other companies for optimal security. For example, customer_1 applications must run on a different set of servers than customer_2 applications.
- Customer_2 also has a critical application that must be completely separate from both their other applications and any other applications in the cell.
- You also must host several internal applications that do not have performance and security requirements, including the company employee directory.

Solution

Use dynamic cluster isolation to meet the requirements of your customers. *Dynamic cluster isolation* specifies whether the dynamic cluster runs on the same nodes as other instances of dynamic clusters, or whether the dynamic cluster is the only dynamic cluster that is running on a node. The following configurations meet your customer requirements:

- Create dynamic clusters for customer_1 and customer_2. For the isolation requirement for each of the dynamic clusters, click group isolation when you create the dynamic cluster. With *group isolation*, a dynamic cluster instance can run on the same node only with instances of dynamic clusters that are a part of the same isolation group. Create an isolation group for customer_1 dynamic clusters, and another isolation group for customer_2 dynamic clusters.
- To separate the critical customer_2 application, define strict isolation for the dynamic cluster that is hosting the critical application. With *strict isolation*, a dynamic cluster instance can run only with other instances of the same dynamic cluster on a node.
- Because the internal employee directory application does not have any isolation requirements, click **No isolation requirements** when you create the dynamic cluster for that application.

By configuring the dynamic cluster with specific isolation requirements, you are providing the most stable and secure service for customers while also hosting internal applications for your company.

Priority of isolated dynamic clusters

Dynamic cluster isolation ensures that dynamic cluster instances from different dynamic clusters do not run on the same node, but it does not make guarantees about how the system avoids a violation of the isolation mode. Configuring strict isolation does not give a dynamic cluster priority over any other dynamic clusters.

For example, you might have an environment with one available node, and two dynamic clusters. Each dynamic cluster has the minimum number of cluster instances set to 1. Consider the following isolation configuration scenarios:

- Both of the dynamic clusters are configured with strict isolation.
- One of the dynamic clusters is configured with strict isolation, and the other dynamic cluster does not have strict isolation defined.

In both of these scenarios, the application placement controller can place a single cluster instance for one of the dynamic clusters. Dynamic cluster instances cannot be placed for both dynamic clusters because only one node is available. In both scenarios, no guarantee is made about which dynamic cluster starts an instance. Even in the second scenario, where one of the dynamic clusters has strict isolation defined, the isolated dynamic cluster does not have priority over the other dynamic cluster. This restriction can be problematic for small systems, such as a single node environment, or environments with a large number of constraints.

Configuring application lazy start

By configuring application lazy start, you can release resources that are being consumed by inactive dynamic clusters so that other cluster instances in the cell can use these resources.

Before you begin

- Application lazy start requires that requests are routed through the on demand router. For more information about creating an on demand router, read about creating ODRs.
- Internet Inter-ORB Protocol (IIOP) and Java Message Service (JMS) requests cannot be used because they are not routed through the ODR. Do not use application lazy start on dynamic clusters that run Session Initiation Protocol (SIP) applications.
- Create a dynamic cluster. For more information, read about creating dynamic clusters. You can also configure application lazy start when you are creating a dynamic cluster.

About this task

An *application lazy start* is the activation of the first application server instance of a deactivated dynamic cluster when an application request arrives. You decide which applications to deactivate and subsequently lazily start. Use application lazy start if you have an environment in which the ratio of the number of dynamic clusters to the number of nodes is high, and if many dynamic clusters are not accessed for a long period. By using application lazy start, you can increase the performance and efficiency of your environment.

Procedure

1. Edit the dynamic cluster properties. In the administrative console, click **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name***.
2. Select the **If other dynamic clusters need resources, stop all instances of this cluster during periods of inactivity** option.
3. Set the **Time to wait before stopping instances** in minutes if the application placement controller determines that the resource is required by some other dynamic application cluster.

Results

The application placement controller tracks the amount of time that a dynamic cluster has been inactive. If another dynamic cluster needs resources, the application placement controller can stop the inactive dynamic cluster after the specified time period. The resources that were consumed by the inactive instance are released and made available for use by the dynamic cluster that requires additional resources. If a request arrives for the stopped dynamic cluster instance, the lazy start controller is activated and at least one server instance is started. In the meantime, HTTP error code 503 (server unavailable) is generated. The error page informs you that the requested application is starting and the request is resubmitted shortly.

What to do next

- You can configure the application lazy start without the `proactiveIdleStop` custom property configured, once a application request arrives, the application server instance will start, but lazy start will never stop it in future.
- You can configure the application lazy start with the `proactiveIdleStop` custom property configured, the application server instance will stop after a specified time period.
- You can configure a custom error page for the ODR that can be used for the 503 error. The error page can include an HTTP meta refresh tag that causes the browser to automatically send the request again after a certain waiting period. For more information about configuring custom error pages, read about configuring ODRs.

Application lazy start:

An *application lazy start* is the activation of the first application server instance of a deactivated dynamic cluster when an application request arrives. You decide which applications to deactivate and subsequently lazily start. Use application lazy start if you have an environment in which the ratio of the number of dynamic clusters to the number of nodes is high, and if many dynamic clusters are not accessed for a long time period.

Application lazy start is available for requests that are routed through the on demand router (ODR). Internet Inter-ORB Protocol (IIOP) and Java Message Service (JMS) requests cannot be used because they are not routed through the ODR. Do not use application lazy start on dynamic clusters that run Session Initiation Protocol (SIP) applications.

Application lazy start process

To make valuable resources available for other dynamic clusters in an environment that routes requests through the ODR, you can temporarily deactivate idle dynamic clusters, stop all server instances, and release valuable resources for other active clusters. Later, when a request arrives for one of the deactivated clusters, the cluster is activated and at least one server instance is started. In the meantime, a HTTP error code 503 (server unavailable) page is displayed when a user tries to access the server. The error page informs you that the requested application is starting and to resubmit the request shortly. You can configure the ODR to display a special error page that includes an HTTP meta refresh tag so that the browser can automatically re-send the request after a certain time period.

A lazy start controller monitors request activity for dynamic clusters that can be deactivated when idle, and lazily started when a request arrives. When a request arrives at the ODR for an inactive dynamic cluster, lazy start controller triggers placement controller to run off cycle and start an instance for that cluster. The lazy start controller also advises the placement controller when to deactivate the inactive clusters.

The following diagram demonstrates the activity flows of the lazy start and placement controller:

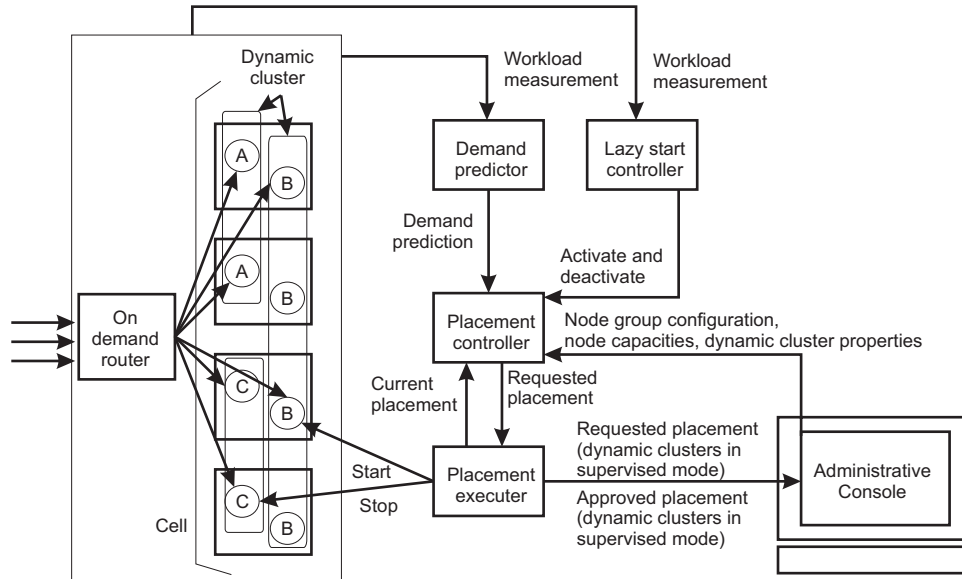


Figure 12. Application lazy start activity flow

Note: You can set the inactivity timeout on a dynamic cluster in automatic mode, but the application placement controller does not necessarily stop an instance after that period of inactivity if no memory contention exists on the computer that hosts this server instance. The application placement controller leverages the inactivity timeout to stop a dynamic cluster instance only if a host does not have enough memory to keep the current number of server instances running. The lazy start controller does not stop instances unless absolutely necessary or `proactivelyIdleStop` is in use. For more information about the `proactivelyIdleStop` custom property, read about dynamic cluster custom properties.

The lazy start controller:

The lazy start controller monitors the workload for dynamic clusters that you explicitly mark for deactivation. This controller triggers the placement controller to run off cycle and activate an inactive dynamic cluster when workload is detected. The lazy start controller also informs the placement controller when it is appropriate to deactivate these clusters.

The lazy start controller maintains the following information for each dynamic cluster that can be deactivated:

Deactivation timeout

Specifies the minimum idle time before the deactivation of a dynamic cluster, which can be deactivated. Deactivation stops all instances of a dynamic cluster. The lazy start controller deactivates a cluster only if it perceives a condition of memory contention on the node that is running an idle application server.

Time of last activity


Specifies the time at which the cluster was last active. The lazy start controller compares the time of last activity to the current time to determine how long the cluster has been inactive. This value is then compared to the deactivation timeout value to determine if the dynamic cluster should be deactivated if another dynamic cluster needs resources.

State Specifies if the dynamic cluster is in Active, Inactive, and Unknown state. A dynamic cluster state is initially Unknown. If queue activity or active instances are detected, then the state is changed to Active. If the cluster is idle for a sufficient time, according to the deactivation timeout property, the state is switched to Inactive.

Configuring vertical stacking

You can configure vertical stacking to enable the application placement controller to start more than one instance of the dynamic cluster to improve bottleneck conditions within an application.

Before you begin

-  Vertical stacking is supported on the z/OS operating system, but the WebSphere Application Server for z/OS multiple servant feature is the preferred alternative. Consider using vertical stacking for application servers on the z/OS operating systems only for application servers that host applications that are constrained to run in a single servant application server.
- To use vertical stacking, the nodes in your dynamic cluster must be at the same product version as the deployment manager, or at a later version. Nodes that are at an earlier release than the deployment manager cannot start multiple server instances on the same node.
- Vertical stacking can occur only on nodes that have complete life cycle management. You cannot configure vertical stacking for dynamic clusters that have manually defined cluster members.

About this task

Use vertical stacking to improve bottleneck conditions in your application. With vertical stacking, multiple server instances can start on a node. By configuring multiple server instances, you can use all the power that is available on the nodes when a large workload exists for the application.

Procedure

1. Determine whether your application has an internal bottleneck by profiling the application. The application placement controller assumes that the only resource bottleneck that an application might have is either in the processor or in memory. Otherwise, the application might have an internal bottleneck.
 - a. Run an instance of the application on each node.
 - b. Increase the load for the application.
 - c. Analyze the runtime reports, the throughput curves and processor utilization charts for the nodes that are running the application. Click **Runtime operation > Reports**.

If the throughput curve saturates while increasing the load, but the processor utilization remains well under 100%, the application might have an internal bottleneck. Configure vertical stacking to improve these bottleneck conditions. If the server instances are able to reach processor utilization near 100%, do not enable vertical stacking.

2. Determine the appropriate stacking number. The stacking number is the number of application servers for a cluster that are needed to use all the power of a node. For more information about determining a stacking number, read about determining the stacking number for a dynamic cluster.
3. Enable vertical stacking and the stacking number. If your dynamic cluster is deployed to a node group that has homogeneous nodes, you can configure the stacking number one time in the administrative console. If your dynamic cluster is deployed to a node group that has heterogeneous nodes, you must configure the stacking number by specifying custom properties.
 - For more information about enabling vertical stacking for a dynamic cluster that is on homogenous nodes, read about configuring a dynamic cluster with homogeneous nodes to support vertical stacking.
 - For more information about enabling vertical stacking for a dynamic cluster that is on heterogeneous nodes, read about configuring a dynamic cluster with heterogeneous nodes to support vertical stacking.

Results

When a large workload exists, the application placement controller can fully use the power of the nodes by starting multiple cluster instances on each node.

The general formula to determine the percentage of the processor that is used by each instance on a node is $100\% / \text{maximum_number_of_stacked_instances}$. For example, if three instances are stacked, then workload is throttled to prevent any one of these instances from using more than 33% of the processor.

What to do next

Tune and monitor your application placement configuration.

Determining the stacking number for a dynamic cluster:

You can use profiling to determine the stacking number for your dynamic cluster. The stacking number is the number of cluster instances that are needed to use all the power of a node.

Before you begin

- Determine if your applications have an internal bottleneck. For more information, read about configuring vertical stacking.

About this task

The **Allow more than one instance to start on the same node** setting in the administrative console enables vertical stacking on the node. When you select this option, you must provide the number of instances to start on the same node. This value is the stacking number.

You can determine the stacking number by profiling of your application. If your dynamic cluster is homogeneous, and all of the nodes for the dynamic cluster have similar power, you need to profile and determine the stacking number on one node only. If the member nodes for the dynamic cluster are heterogeneous and vary in power, you must profile each node and determine the stacking number for each node.

Procedure

1. Start one server instance on the test node.
2. View **CPU utilization** and average throughput curves for the node. To view the charts, click **Runtime operations > Reports**.
3. Increase the load on the node, with the goal of reaching 90-100% node utilization.

4. If you cannot reach 90-100% node utilization, add and start another server instance on the test node. Continue adding server instances until the node utilization reaches 90-100%.
5. The current number of server instances on the test node is the stacking number. Remember this number so that you can configure the stacking number in the administrative console.
6. If your dynamic cluster has member nodes that are heterogeneous, repeat these steps for each node and record a stacking number for each node.

What to do next

Enable vertical stacking for your cluster. With vertical stacking enabled, the autonomic managers can limit the processor percentage that is used by each stacked server instance. The general formula to determine the amount of processor that can be used by a single instance is: $100\% / \text{maximum_number_of_stacked_instances}$. For example, if you have three stacked server instances on a node, the workload is throttled to prevent any single server instance from using more than 33 percent of the processor.

Configuring a dynamic cluster with homogeneous nodes to support vertical stacking:

You can configure vertical stacking for a dynamic cluster with nodes that have similar computing power by updating settings in the administrative console.

Before you begin

- Determine the stacking number for your dynamic cluster. For more information, read about this task. If you determine that your application does not have an internal bottleneck when you perform application profiling, do not enable vertical stacking.

About this task

You can enable vertical stacking when you create the dynamic cluster, or you can modify the settings on an existing dynamic cluster to enable vertical stacking. Perform the following steps to modify the settings on an existing dynamic cluster.

Procedure

1. In the administrative console, click **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name***.
2. To enable vertical stacking, click **Allow more than one instance to start on the same node**.
3. Enter the number of instances to start on the same node. This value is the stacking number that you calculated.

What to do next

Tune and monitor your application placement configuration.

Configuring a dynamic cluster with heterogeneous nodes to support vertical stacking:

You can configure vertical stacking for a dynamic cluster with nodes that each have different computing power by configuring custom properties for each node.

Before you begin

- Determine the stacking number for each of your dynamic cluster nodes. For more information, read about this task. If you determine that your application does not have an internal bottleneck when you perform application profiling, do not enable vertical stacking.

About this task

If your dynamic cluster is deployed to a group of nodes that are heterogeneous, you must configure the stacking number for each individual node. You can configure the stacking number for each node by creating custom properties on the dynamic cluster.

Procedure

1. Enable vertical stacking on the dynamic cluster.
 - a. In the administrative console, click **Servers > Clusters > Dynamic clusters > *cluster_name***.
 - b. To enable vertical stacking, click **Allow more than one instance to start on the same node**.
 - c. Enter the number of instances to start on the same node. Set this value to the largest stacking number that you have determined for your nodes. When you define the individual stacking numbers as custom properties for each node, this value is overridden. However, if no custom property value is specified, the node uses the number of instances value that is defined on the administrative console page.
2. Create a custom property for the stacking number on one of your nodes. In the administrative console, click **Servers > Clusters > Dynamic clusters > *cluster_name* > Custom properties > New**.
3. Enter the name of the custom property. The name is `numVerticalInstances.node_name`, where *node_name* is the name of your node.
For example: If your node is named `node1`, the name of your custom property is `numVerticalInstances.node1`.
4. Enter the value of the custom property. The value of this custom property is the stacking number that you calculated for the specified node.
5. Add a custom property that defines the stacking number for each one of your nodes in the dynamic cluster.

Example

You have determined that among the three nodes on which your dynamic cluster runs, that each node has a different stacking number.

Table 9. Heterogeneous node stacking number scenario

Node name	Stacking number
node_1	3
node_2	5
node_3	8

To configure these stacking numbers in the administrative console, use the following configuration:

1. Set the **number of instances** value for the dynamic cluster to 8. The number of instances value must be greater than or equal to your largest stacking number.
2. Specify the following custom properties:
 - `numVerticalInstances.node_1 = 3`
 - `numVerticalInstances.node_2 = 5`
 - `numVerticalInstances.node_3 = 8`

What to do next

Tune and monitor your application placement configuration.

HTTP session rebalancing

You can dynamically and actively balance the distribution of HTTP sessions among application servers by using HTTP session rebalancing.

WebSphere Application Server assigns HTTP sessions based on application server *affinity*. After a session is established in a particular application server, subsequent requests that belong to the same session are assigned to the application server that has the established session affinity.

Session affinities cause the session load to balance out for a cluster in the long term, and generally any imbalances that occur are short-lived and tolerable. However, you can use session rebalancing to help some of the shorter term imbalances that can occur, without modifying the performance benefits of session affinities.

Intelligent Management uses session rebalancing to expedite the balancing of sessions across a cluster. You can use session rebalancing if you use distributed sessions and track your sessions with cookies. By default, Intelligent Management application servers are configured with session tracking enabled using cookies, but without distributed sessions.

The steps that you use for configuring servers in static clusters can also be used for servers in dynamic clusters. If you are configuring dynamic clusters, leave the dynamic cluster in manual mode initially so that the servers in the cluster are not started automatically. After the distributed environment settings of all the servers in the cluster are changed, switch the dynamic cluster to automatic mode. If you do not switch modes, you must stop and restart the servers that are already started in the dynamic cluster to pick up the distributed environment setting changes.

When a new server becomes available in a dynamic cluster, that server does not have any assigned HTTP sessions. However, the existing servers in the dynamic cluster have sessions with assigned affinity.

The goal of session rebalancing is to reassign sessions so that the number of sessions that are running in each server in the dynamic cluster is proportional to the assigned weight of the servers. The dynamic workload manager (DWLM) performs the session rebalancing function. DWLM decides how many sessions to move and where to move the sessions based on information about the entire dynamic cluster. The DWLM component factors in the session location and can initiate session moves in addition to changing the weight on specific servers. Any events that DWLM monitors can initiate the moves, such as new starting servers or overloaded servers.

DWLM often picks an uneven distribution of routing weights. This approach is allowed because the primary objective of DWLM is to equalize service times. A secondary goal of DWLM is to even out the routing weights when possible without significant degradation of the primary objective of equalizing service times.

If you are using sessions that are maintained by Uniform Resource Locator (URL) rewriting or Secure Sockets Layer (SSL) IDs, no rebalancing is performed. Rebalancing is also not performed on HTTP POST requests. Sessions are not rebalanced if the session is accessed by issuing the HTTP request directly to the application server HTTP port, bypassing the on demand router. The PMI counter for memory sessions reflects the moving sessions. Specifically, the PMI counter decreases on existing servers as sessions move, and increases on the new servers that are recently started. However, new sessions are load balanced across any of the servers in the cluster. For more information, read about analyzing PMI data and best practices for using HTTP sessions.

The session rebalancer request and response filters in ODR track which sessions are being sent to which servers, to get a live session count. Using information from the dynamic workload manager (DWLM) for each dynamic cluster, the ODR also tracks the number of sessions that are being sent to each server, ranking the servers in the dynamic cluster. This ranking is used to determine how many sessions to move between servers. The default configurations vary depending on the type of dynamic cluster that you are using:

- For dynamic clusters that are made of WebSphere Application Server application servers, the distributed session configuration is detected. Sessions are automatically rebalanced unless you turn off the session rebalancing by setting the `HttpSessionRebalanceOff` custom property to true.
- For dynamic clusters that have other types of servers, the runtime cannot detect if the server is using distributed sessions. Session rebalancing is disabled by default. You can set the `HttpSessionRebalanceOff` custom property to false on the dynamic cluster to enable session rebalancing.

HTTP session balancing is supported when you are using eXtreme Scale-based HTTP session support instead of the regular WebSphere Application Server session management.

Session rebalancer configuration

You can enable or disable session rebalancing by adjusting the `HttpSessionRebalanceOff` custom property on the dynamic cluster in the administrative console.

To enable session rebalancing on the other middleware server types and eXtreme Scale servers, you must set the `HttpSessionRebalanceOff` custom property to false for the dynamic cluster. Session rebalancing is automatically enabled for dynamic clusters that have application servers. Set the custom property on the specific dynamic cluster.

Intelligent Management rebalancing scenario

The following example illustrates session rebalancing in an Intelligent Management environment. In this example, dynamic clusters exist in the configuration. The server performs session rebalancing by sending information from DWLM to the ODR. The ODR then routes the HTTP sessions to the appropriate server.

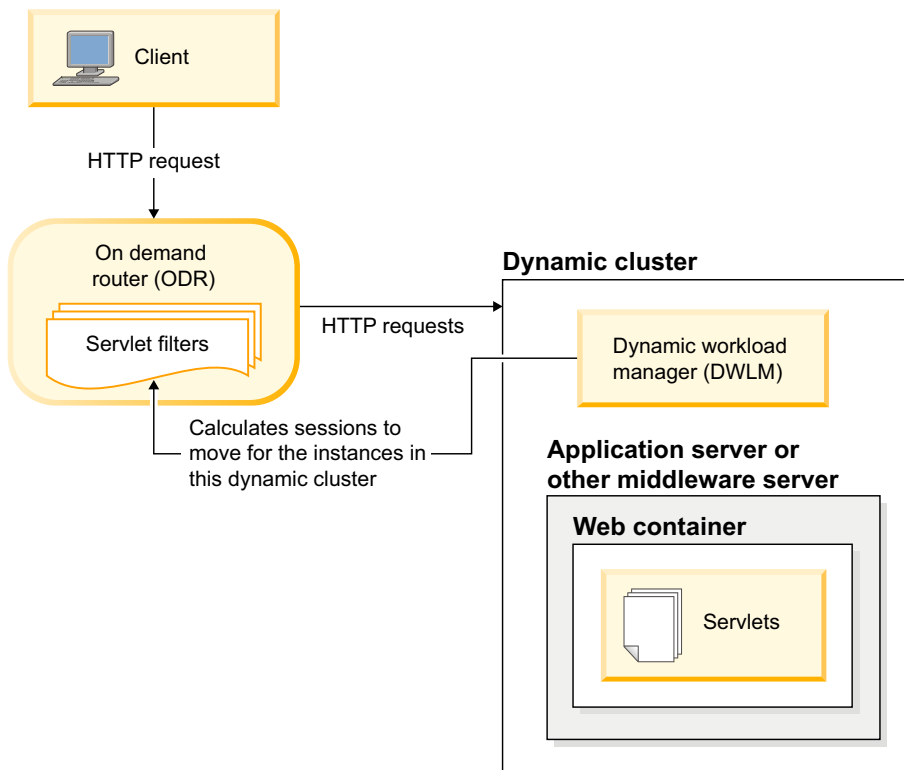


Figure 13. HTTP session rebalancing in Intelligent Management

Adding middleware servers to configurations

Using middleware servers, you can manage all of the servers in your environment, including servers that are not created with Intelligent Management, by using the product administrative domain.

Before you begin

If you are configuring representations in the domain of externally created middleware servers, perform the following tasks on the nodes:

1. Install and configure the middleware software on the nodes.
2. The node agent on each node must be at the same fix pack level as the deployment manager.
3. Federate nodes to an Intelligent Management cell.

You can use the **addNode** command or the administrative console to federate the node. For more information, read about utilizing the **addNode** command and the topic on adding, managing, and removing nodes.

4. Create and configure ODRs. Read about creating ODRs for more details.
5. Each of your nodes must have the same network view for each endpoint and resource that is related to WebSphere Application Server. All host entries or host alias entries that are related to WebSphere Application Server within a `hosts` file must be included in the `hosts` file on each of the other nodes in the configuration, including any `hosts` file on nodes that run the node agent. For example, if you define a product-related host alias called `xdagentA` in the `hosts` file for the `nodeA` node, then the `hosts` file on each middleware node must contain an entry for the `xdagentA` host alias.

About this task

Middleware servers encompass all servers in the middleware tier that provide the infrastructure for applications or their data. The steps vary depending on if you are configuring complete lifecycle management middleware servers or assisted lifecycle management middleware servers.

Procedure

- Complete lifecycle servers include any servers that the environment can instantiate, or create. These server types include WebSphere Application Server types such as application servers, generic servers, Web servers, and proxy servers. The product adds complete lifecycle management for PHP servers and WebSphere Application Server Community Edition servers. For more information, read about adding complete lifecycle middleware servers.
- With assisted lifecycle servers, you use templates to create representations of servers. However, these servers still exist within the administrative domain of their respective middleware platform. These servers can be controlled operationally. Server health and performance are monitored. You can configure the administrative console to display log files and configuration files for these servers, except for JBoss, BEA WebLogic, Apache Tomcat, and external WebSphere Application Server. For more information, read about adding assisted lifecycle middleware servers.

What to do next

To easily manage groups of servers to host an application, configure dynamic clusters. You can create clusters of either type of server. However, dynamic application placement is supported only on complete lifecycle servers. When you create a dynamic cluster of assisted lifecycle middleware servers, the same version of middleware software and the same applications must be installed on the servers. If the servers in a dynamic cluster have varying sets of applications installed, routing through the on-demand router might result in failed requests.

The logs and trace views in the administrative console that you access by clicking **Troubleshooting > Logs and trace** are not supported for externally created middleware servers and PHP servers. You can

use the external log viewing service to view the log files for these server types in the administrative console. To configure the external log viewing service, navigate to the configuration panel for the server and click **External log service**.

gotcha: You cannot use the external log viewing service with an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can view logs for these middleware servers on the machines that host them.

Adding complete life-cycle middleware servers

You can configure complete life-cycle middleware servers so that the dynamic operations environment can govern all aspects of server operations, create and remove server instances, and deploy applications to the servers.

About this task

By configuring complete life-cycle servers, the environment performs the following tasks on your middleware servers:

- Create and remove server instances based on the needs of the dynamic operations environment
- Govern all aspects of server configuration
- Provide operational control over the server
- Deploy applications to the server
- Visualize and monitor server health and performance
- Use expression-based dynamic clustering (for PHP servers, WebSphere Application Server Community Edition servers, and application servers)

Note: On the administrative console, when you select **Servers > All servers > *middleware_server***, and then click **Stop** for a middleware server that displays a status of Started, you can intermittently receive an error. The following message is an example of the error:

Error xdblade31b04/WASMaintModeDC1_xdblade31b04 has not been started.

The server cannot be stopped from this page when this situation occurs.

As a work-around, complete one of the following actions to stop the started middleware server:

- Click **Servers > Server Types > *type_of_middleware_server* > *middleware_server***. Then click **Stop** for a middleware server that displays a status of Started.
- Stop and then start the servers from the command line.

Procedure

Follow the steps to create your complete life-cycle middleware server. The steps vary depending on the server type:

- **On demand router (ODR):** The on demand router is a specialized proxy server that can route work to application server nodes. The ODR is required for Intelligent Management environment. Read about creating and configuring ODRs for more information.
- **PHP server:** Use Intelligent Management to create PHP servers. Read about creating PHP servers and PHP dynamic clusters for more details.
- **WebSphere Application Server Community Edition server:** Intelligent Management offers assisted and complete life-cycle management for WebSphere Application Server Community Edition servers. Complete life-cycle management is supported for WebSphere Application Server Community Edition Version 2 and later servers. WebSphere Application Server Community Edition Version 1 and Version 2 servers can be represented in the administrative console as assisted life-cycle servers. Read about creating life-cycle servers and dynamic clusters for more information.

- **Generic server:** Generic servers are managed in, but not supplied by WebSphere Application Server. These servers might be a Java server, a C or C++ server or process, a CORBA server, or a Remote Method Invocation (RMI) server. Read about creating generic servers for more information.
- **Proxy server:** Proxy servers, like the ODR, route requests to application server nodes. However, the ODR is the required proxy for dynamic operations. For more information read about creating proxy servers.
- **Web server:** Intelligent Management works with a web server to route requests for dynamic content, such as servlets, from web applications. Read about communicating with web servers for more information.
- **Application server:** Application servers are supported for both WebSphere Application Server and Intelligent Management. By creating dynamic clusters of application servers, you enable application server virtualization. Read about creating application servers for more information.

What to do next

You can create dynamic clusters of application servers, WebSphere Application Server Community Edition servers, and PHP servers.

Creating PHP servers and PHP dynamic clusters:

By creating PHP Hypertext Preprocessor (PHP) servers or dynamic clusters, you can deploy PHP applications. You can use the product to manage these applications and servers.

Before you begin

- Install Apache HTTP Server and PHP on the nodes that you want to host PHP servers. The supported Apache HTTP Server versions are 1.3 (all releases), 2.0 (all releases), and 2.2 (all releases). The supported PHP versions are 4 and 5 (all releases). Default server templates are provided for these combinations of Apache HTTP Server and PHP.
- Federate these nodes into the cell.
- **Attention:** If you are using Apache HTTP Server Version 2.2 and PHP Version 5.2 (all releases), then you must perform some manual edits to the `httpd.conf` file. Instead of performing these edits each time you create a PHP server or dynamic cluster, create a PHP server with the `APACHE22_PHP5` template, edit the `httpd.conf` file, and create a server template from that server, which you can use to create the rest of your PHP servers.

You can also choose an existing server to use as the template for a new PHP server by using the PHP server administrative tasks, or by selecting a predefined server instead of a template. You cannot select a user-defined template from the PHP server wizard.

About this task

A PHP server is defined in Intelligent Management as an Apache HTTP Server with the `mod_php` module. You can define a PHP server on nodes that are running the product or the node agent. PHP server configuration consists of three documents: the `server.xml`, `httpd.conf` and `php.ini` files. The `server.xml` file contains properties that describe the Apache and PHP runtime locations and server operations. The `httpd.conf` file is the Apache HTTP Server configuration file that includes PHP modules. You can define variables in the administrative console to be used in the `httpd.conf` file. The `php.ini` file contains configuration data that is used by the PHP interpreter.

Procedure

1. Configure the middleware descriptors so that you can run discovery to find the Apache and PHP runtimes. In the administrative console, click **System administration > Middleware descriptors > *middleware_platform_name***. The default locations are listed in the `apacheWebServerRuntime` and `phpRuntime` middleware descriptors. The discovery service uses middleware descriptors to define

where to look for the Apache and PHP runtimes. Verify that the middleware descriptors contain the correct installation locations for your environment. Use a semicolon delimiter for Windows paths, and a colon delimiter for UNIX paths.

2. Run discovery to find the PHP and Apache HTTP Server runtimes on the nodes that you want to host PHP servers. Discovery runs automatically when the node agent starts. Discovery also runs at a predefined time interval that you can specify in the middleware descriptor. However, you can also invoke discovery:
 - a. In the administrative console, click **System administration > Middleware nodes**.
 - b. Select the nodes that you want to run through discovery, and select the **Run discovery** operational action.
 - c. Click **Run**.
 - d. Verify that the discovery service found the run time. In the administrative console, click **System administration > Middleware nodes > node_name > Node installation properties**. If any runtime environments exist, then properties that begin with APACHE or PHP strings are displayed.

Only the default installation locations are listed in the apacheWebServerRuntime and phpRuntime middleware descriptors. You can modify the middleware descriptor so that the discovery looks for the correct installation location.

3. Create PHP deployment targets. The deployment target is where you deploy the PHP application. You can create individual PHP servers or PHP dynamic clusters.
 - Create a PHP server.
 - a. In the administrative console, click **Servers > New server**. Choose **Create a new server instance**, and select the **PHP server** type. You can also click **Servers > Server types > PHP servers > New**.
 - b. Select the node on which you want the PHP server to run. This node must have the PHP and Apache HTTP Server runtime environments.
 - c. Choose the appropriate Apache HTTP Server and PHP runtime environments for the node.

For both runtimes, a list of variables displays at different levels of granularity such as APACHE, APACHE_2, APACHE_2_0, and APACHE_2_0_59. With these variables, you can select the runtime that corresponds to either a specific or a general version that you need. For example, the most generic variables are APACHE and PHP, which always point to the newest versions that are installed on your node. Select these variables if you always want to use the newest runtimes and you do not need a specific version. You can also choose a runtime that is based on major version such as APACHE_2. With this variable, you can use any Apache 2 runtime. The newest Apache 2 runtime that is installed on your node is used. By always using the newest Apache Version 2 runtime, you can upgrade between minor versions without any manual server configuration changes. If you update from Apache Version 2.0.58 to Apache Version 2.0.59 and add the new runtime location to the middleware descriptor, the runtime service automatically detects the newer runtime and updates the server configuration to point to the new runtime.
 - d. Choose the PHP server template.
 - e. Confirm and save the PHP server.
 - Create a PHP dynamic cluster.
 - a. In the administrative console, click **Servers > Clusters > Dynamic clusters > New**.
 - b. Choose the PHP server dynamic cluster type, and name the dynamic cluster.
 - c. With PHP servers, you must use automatic membership. Define a membership policy to identify which nodes host the PHP servers in the dynamic cluster. These nodes must have the PHP and Apache HTTP Server runtime environments installed.

For example, you might use the following membership policy:

```
node_property$APACHE IS NOT NULL and node_property$PHP IS NOT NULL
```

Restriction: PHP dynamic cluster members must be at the same two digit Apache version and one digit PHP version. This limitation is caused by the differences in configuration

between releases. If multiple versions are installed on your system, update the membership policy to include the version number, like in the following example:
node_property\$APACHE_2_0 IS NOT NULL and node_property\$PHP_4 IS NOT NULL

- d. Select the server template for the PHP runtime and Apache HTTP Server runtime that you are using.
 - e. Specify other dynamic cluster properties.
 - f. Save your PHP dynamic cluster.
4. If you are using PHP Version 5.2 (any release), then you must manually edit the httpd.conf file to use the php5apache2_2.d11 module, instead of the default php5apache2.d11 module.
- a. Verify that you created the PHP server with the APACHE22_PHP5 template.
 - b. Open the httpd.conf file. In the administrative console, click **Servers > Server types > PHP servers > php_server_name > External configuration**. Choose the httpd.conf file, and click **Retrieve**.
 - c. Modify the file to use the php5apache2_2.d11 module. Search for the following string in the httpd.conf file, where *php_server_root* is the location of your PHP server:

```
LoadModule php5_module "php_server_root/php5apache2.d11"
```

Edit this line to reference the php5apache2_2.d11 module. See the following example:

```
LoadModule php5_module "php_server_root/php5apache2_2.d11"
```
 - d. Apply and save your changes.
 - e. Optional: To avoid repeating these steps for each PHP Version 5.2 server that you create, make a template of your current server that you can use when you create subsequent PHP servers. In the administrative console, click **Servers > Server types > PHP servers > Templates > New**. Select the server from which you want to create a template, specify properties for the template, and save your changes. Choose this template when you create other PHP servers or dynamic clusters. You can also choose an existing server to create your new PHP server from the PHP server wizard.
5. Optional: Update the HTTP and HTTPS ports for your servers. The default port is uniquely generated. modifying the port changes the port for the Apache server, which updates the httpd.conf file.
- a. In the administrative console, click **Servers > Server types > PHP servers > php_server**.
 - b. Edit **HTTP connector** or **HTTPS connector**. Click **OK** and save and synchronize your changes.

What to do next

Deploy PHP applications to your PHP servers and dynamic clusters.

Creating complete life-cycle WebSphere Application Server Community Edition servers and dynamic clusters:

You can create WebSphere Application Server Community Edition complete life-cycle management servers and expression-based dynamic clusters. You can deploy managed applications to those servers and dynamic clusters.

Before you begin

- Determine the version of WebSphere Application Server Community Edition that you want to run:
 - For complete life-cycle management, use WebSphere Application Server Community Edition Version 2 or later.
 - Additional steps are required to enable Java Management Extensions (JMX) security in WebSphere Application Server Community Edition Version 2.1. For more information, read about configuring JMX security for WebSphere Application Server Community Edition Version 2.1 servers.
 - If you are using WebSphere Application Server Community Edition Version 1 (all releases), you can only use assisted life-cycle management. For information about configuring assisted life-cycle servers, read about configuring assisted lifecycle WebSphere Application Server Community Edition.

- Install the WebSphere Application Server Community Edition runtime environment to the nodes on which you want to run servers. You can either use the installation wizard that is included in WebSphere Application Server Community Edition, or you can configure the centralized installation manager to install the runtime environment.
- Federate your nodes into the cell. Read the topics on the **addNode** command and adding, managing, and removing nodes for more information.

Procedure

1. Configure the middleware descriptors so that you can run discovery to find the WebSphere Application Server Community Edition runtime environment. If you use the centralized installation manager to install WebSphere Application Server Community Edition, you can skip this step. In the administrative console, click **System administration > Middleware descriptors > wasceRuntime**. The discovery service uses middleware descriptors to define where to look for the runtime environment. Verify that the middleware descriptors contain the correct installation locations for your environment. The default values:
 - C:\Program Files\IBM\WebSphere\AppServerCommunityEdition
 - /opt/IBM/WebSphere/AppServerCommunityEdition
2. Run discovery to find the WebSphere Application Server Community Edition runtimes on the nodes that you want to host servers. If you use the centralized installation manager to install WebSphere Application Server Community Edition, you can skip this step. Discovery runs automatically when the node agent starts. Discovery also runs at a predefined time interval that you can specify in the middleware descriptor. However, you can also invoke discovery with the following steps:
 - a. In the administrative console, click **System administration > Middleware nodes**.
 - b. Select the nodes that you want to run through discovery, and select the **Run discovery** operational action.
 - c. Click **Run**.
 - d. Verify that the discovery service found the runtime environment. In the administrative console, click **System administration > Middleware nodes > node_name > Node installation properties**. If any runtime environments exist, then properties that begin with WASCE strings are displayed.

Only the default installation location is listed in the `wasceRuntime` middleware descriptor. You can modify the middleware descriptor so that the discovery looks for the correct installation location.

3. Create WebSphere Application Server Community Edition deployment targets. The deployment target is where you deploy the WebSphere Application Server Community Edition application. You can create individual WebSphere Application Server Community Edition servers or dynamic clusters.
 - Create a WebSphere Application Server Community Edition server.
 - a. In the administrative console, click **Servers > New server**. Choose **Create a new server instance**, and select the **WebSphere Application Server Community Edition server** type. You can also click **Servers > Server types > WebSphere Application Server Community Edition server > New**.
 - b. Select the node on which you want the WebSphere Application Server Community Edition server to run. This node must have the WebSphere Application Server Community Edition runtime.
 - c. Choose the WebSphere Application Server Community Edition server template.
 - d. Confirm and save the WebSphere Application Server Community Edition server.
 - Create a WebSphere Application Server Community Edition dynamic cluster.
 - a. In the administrative console, click **Servers > Clusters > Dynamic clusters > New**.
 - b. Choose the WebSphere Application Server Community Edition server dynamic cluster type, and name the dynamic cluster.
 - c. To create a dynamic cluster with complete life-cycle management, choose **Automatically define cluster members with rules**. Define a membership policy to identify which nodes host the

WebSphere Application Server Community Edition servers in the dynamic cluster. These nodes must have the WebSphere Application Server Community Edition runtime environment installed. The default membership policy follows:

```
node_property$IS_WASCE_DISCOVERED = 'TRUE'
```

This membership policy looks for WebSphere Application Server Community Edition Version 2.0 and later runtimes.

If you want to specify a specific version, you might use the following membership policy:


```
node_property$WASCE_2.0.0 IS NOT NULL
```

This policy defines membership as all nodes that run WebSphere Application Server Community Edition Version 2.0.0. The group of nodes that you define must have the exact same version of WebSphere Application Server Community Edition installed. For example, you cannot combine Version 1 and 2 servers in the same dynamic cluster.

- d. Select the server template for the WebSphere Application Server Community Edition runtime environment that you are using.
- e. Specify other dynamic cluster properties.
- f. Save your WebSphere Application Server Community Edition dynamic cluster.

Results

The new server displays in the list on the **Servers > Server types > WebSphere Application Server**

Community Edition server administrative console page. If you see an error status icon () , the creation process did not complete. This error can occur because of problems in the WebSphere Application Server Community Edition server runtime environment. Delete the server and try creating the server again.

The list of servers that displays on this administrative console page includes all of your complete life-cycle, assisted life-cycle, and discovered WebSphere Application Server Community Edition servers. If you have WebSphere Application Server Community Edition dynamic clusters defined, the cluster members are also listed on this administrative console page.

What to do next

- Deploy applications to your WebSphere Application Server Community Edition servers and dynamic clusters. For more information, read about deploying WebSphere Application Server Community Edition applications.
- Stop discovery from examining the installation location that you used to create your managed servers.
 - If you used discovery to find the WebSphere Application Server Community Edition runtime environment, you can disable discovery globally or you can remove the specific installation location from the middleware descriptor.
 - If you installed the WebSphere Application Server Community Edition runtime environment with the centralized installation manager, discovery was not needed to register the configuration. Do not include centralized installation manager paths in the discovery paths in the middleware descriptor. If you used the default centralized installation manager installation location, you must globally disable discovery or change the default discovery location to a different path.

For more information, read about configuring middleware descriptors.

Do not use the WebSphere Application Server Community Edition console to edit your complete lifecycle servers, except for functions that are not available with Intelligent Management . Using the WebSphere Application Server Community Edition console can cause unexpected results. For example, if you edit the server port numbers in the WebSphere Application Server Community Edition console, the port value that is configured for Intelligent Management is overridden with the new value.

WebSphere Application Server Community Edition servers:

You can centrally manage WebSphere Application Server Community Edition servers and applications along with your other servers and applications from the Intelligent Management environment.

Complete life-cycle servers

You can create WebSphere Application Server Community Edition servers in the Intelligent Management environment that have complete life-cycle management. To take advantage of complete life-cycle management, you must install WebSphere Application Server Community Edition Version 2 or later. Complete life-cycle servers provide the following benefits:

- You can create WebSphere Application Server Community Edition servers from the administrative console, which creates a server on the WebSphere Application Server Community Edition runtime environment. These servers can also be vertically stacked on a server from a single run time.
- You can create expression-based dynamic clusters of WebSphere Application Server Community Edition servers.
- You can view the performance of WebSphere Application Server Community Edition servers in the runtime operations panels.
- You can install managed applications to WebSphere Application Server Community Edition servers and dynamic clusters. When you install managed applications, you deploy the application within the administrative console.
- You can associate complete life-cycle servers with health policies to enable health management and monitoring.

Important: Do not use the WebSphere Application Server Community Edition console to edit your complete life-cycle servers, except for functions that are not available with Intelligent Management.

Discovered servers

Middleware discovery locates existing installations of WebSphere Application Server Community Edition servers and their installed applications, and creates the corresponding configuration to include servers and applications in the Intelligent Management cell. Middleware discovery can preserve the time investment that went into building your original WebSphere Application Server Community Edition environment.

Middleware discovery can find WebSphere Application Server Community Edition Version 1 and later and Version 2 and later servers. These servers are represented as assisted life-cycle servers. You cannot create expression-based dynamic clusters of these servers, but you can group the servers together in a dynamic cluster. The members of these dynamic clusters must have the same version of WebSphere Application Server Community Edition and the same applications installed.

Any discovered applications are represented as unmanaged applications. You can install managed applications with the administrative console on discovered WebSphere Application Server Community Edition Version 2 and later servers. For WebSphere Application Server Community Edition Version 1 discovered servers, you can create representations of the applications that have been installed in the WebSphere Application Server Community Edition console as unmanaged applications only.

Assisted life-cycle servers

If you configured WebSphere Application Server Community Edition servers with Intelligent Management, you manually created representations of the servers in your environment that were assisted life-cycle servers.

You cannot create expression-based dynamic clusters of these servers, but you can group the servers together in a dynamic cluster. The members of these dynamic clusters must have the same version of WebSphere Application Server Community Edition and the same applications installed.

You can install managed applications to assisted life-cycle WebSphere Application Server Community Edition Version 2 and later servers with the administrative console or administrative tasks. For WebSphere Application Server Community Edition Version 1 assisted life-cycle servers, you can install unmanaged applications only. You install unmanaged applications within the WebSphere Application Server Community Edition administrative console and create representations of the applications in the administrative console.

Topology

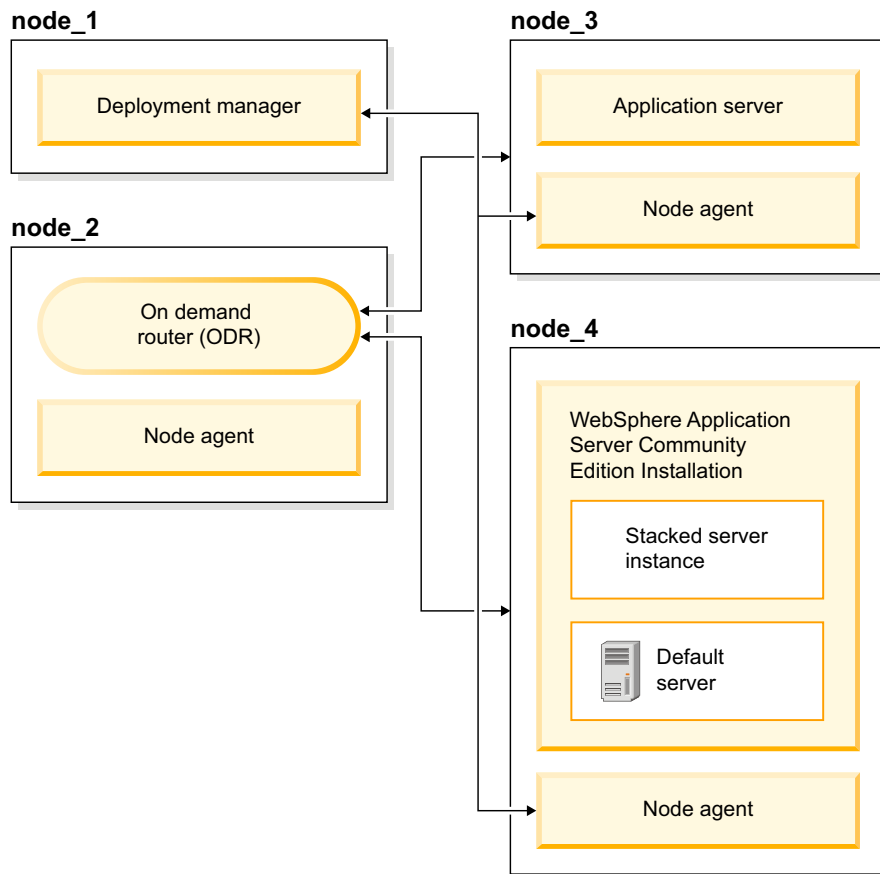


Figure 14. Intelligent Management and WebSphere Application Server Community Edition topology

Configuring JMX security for WebSphere Application Server Community Edition Version 2.1 servers:

If you use the Java Management Extensions (JMX) security feature in WebSphere Application Server Community Edition Version 2.1, additional steps are required to enable the Intelligent Management security feature.

Before you begin

- You must have WebSphere Application Server Community Edition Version 2.1 installed with JMX security enabled.

About this task

JMX security is a feature that is added in WebSphere Application Server Community Edition Version 2.1. When you create new servers or dynamic clusters in the administrative console, you can use the wasce21s server template to create servers that have JMX security enabled. If you have discovered servers, you must perform the additional steps in this topic to configure JMX security.

Procedure

1. If you have an existing server, start the WebSphere Application Server Community Edition Version 2.1 server for which JMX security needs to be enabled.
2. Import the WebSphere Application Server Community Edition certificates into the WebSphere Application Server trust store.
 - a. In the administrative console, click **Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates**.
 - b. Enter the host name of the node where WebSphere Application Server Community Edition Version 2.1 is running, the HTTPS port number, and an alias for the certificate.
 - c. Click **Retrieve signer information**.
 - d. Save and synchronize your settings.
3. Configure your WebSphere Application Server Community Edition Version 2.1 server. If you have not yet created your servers in the environment, you can create the server using the wasce21s server template. If the server has already been discovered by Intelligent Management, you must perform the additional steps that are required to enable security that are listed in this step.

To create a new WebSphere Application Server Community Edition Version 2.1 server or dynamic cluster with JMX security enabled, perform the following steps:

- a. To create a WebSphere Application Server Community Edition server in the administrative console, click **Servers > Server types > WebSphere Application Server Community Edition servers > New**. To create a WebSphere Application Server Community Edition dynamic cluster in the administrative console, click **Servers > Clusters > Dynamic clusters > New**. Choose WebSphere Application Server Community Edition as the server type.
- b. Select the wasce21s template in the wizard.
- c. Save and synchronize your changes.

For a server that has already been discovered by Intelligent Management, perform the following steps:

- a. Stop the WebSphere Application Server Community Edition Version 2.1 server.
- b. Edit the `config.xml` file to disable the JMX service and enable the JMX secure connector.
 - 1) In the administrative console, open the WebSphere Application Server Community Edition Version 2.1 server configuration. Click **Servers > Server types > WebSphere Application Server Community Edition servers > wasce_server_name**.
 - 2) Click the **External configuration** tab. Edit the `config.xml` file.
 - 3) Make the following changes, highlighted in bold text:

```
<module name="org.apache.geronimo.framework/j2ee-security/2.1.1/car">
  <gbean name="JMXService" load="false">
    <attribute name="protocol">rmi</attribute>
    <attribute name="host">${ServerHostname}</attribute>
    <attribute name="port">${JMXPort + PortOffset}</attribute>
    <attribute name="urlPath">/jndi/rmi://${ServerHostname}:${NamingPort + PortOffset}/JMXConnector</attribute>
  </gbean>
</module>
...
...
...
<module name="org.apache.geronimo.configs/clustering/2.1.1/car" load="false">
  <gbean name="Node">
    <attribute name="nodeName">${clusterNodeName}</attribute>
  </gbean>
</module>
...
...
...
<module name="org.apache.geronimo.framework/jmx-security/2.1.1/car" load="true">
  <gbean name="JMXSecureConnector" load="true">
    <attribute name="protocol">rmi</attribute>
    <attribute name="host">${ServerHostname}</attribute>
    <attribute name="port">${JMXSecurePort + PortOffset}</attribute>
    <attribute name="urlPath">/jndi/rmi://${ServerHostname}:${NamingPort + PortOffset}/JMXSecureConnector</attribute>
    <attribute name="clientAuth">>false</attribute>
  </gbean>
</module>
```

- c. Add the `JAVA_OPTS` variable.

- 1) In the administrative console, click **Servers > Server types > WebSphere Application Server Community Edition servers > *wasce_server_name* > Variables > New.**
- 2) Create a new variable called *JAVA_OPTS*.
- 3) Enter the following text in the **Value** field, specifying the appropriate values for the system properties:


```
-Djavax.net.ssl.keyStore=${GERONIMO_HOME}/var/security/keystores/geronimo-default
-Djavax.net.ssl.keyStorePassword=secret
-Djavax.net.ssl.trustStore=${GERONIMO_HOME}/var/security/keystores/geronimo-default
-Djavax.net.ssl.trustStorePassword=secret
```
- 4) Click **OK**.
- d. Add the UseJMXSecureConnector custom property.
 - 1) In the administrative console, click **Servers > Server types > WebSphere Application Server Community Edition servers > *wasce_server_name* > Custom properties > New**
 - 2) Enter UseJMXSecureConnector as the value in the **Name** field.
 - 3) Enter true in the **Value** field.
 - 4) Click **OK**.
- e. Save and synchronize your changes.
- f. Restart your server.

Results

You can use WebSphere Application Server Community Edition Version 2.1 with JMX security enabled.

What to do next

Configure your WebSphere Application Server Community Edition servers. For more information, read about configuring complete lifecycle WebSphere Application Server Community Edition servers and dynamic clusters.

Adding assisted lifecycle middleware servers

By configuring assisted lifecycle middleware servers, you can manage representations of externally created middleware servers that were created outside of the administrative domain.

Before you begin

Read about adding middleware servers to configurations for information about installing the node agent on nodes, and federating those nodes into the configuration.

About this task

With assisted lifecycle middleware servers, you can create a representation of the server. The node agent provides the information that Intelligent Management needs to manage these servers. You can configure the following assisted lifecycle middleware server types:

- Apache Tomcat
- JBoss Application Server
- Custom HTTP servers
- BEA WebLogic Server
- WebSphere Application Server Community Edition
- Apache HTTP Server
- External WebSphere application server

Note: Intelligent Management support for using the following server types is deprecated:

- Apache Tomcat
- BEA WebLogic
- JBoss
- External WebSphere application server

Support for using the administrative console to configure these server types is removed. Use **wsadmin** scripting to manage these resources.

Note: On the administrative console, when you select **Servers > All servers > *middleware_server***, and then click **Stop** for a middleware server that displays a status of Started, you can intermittently receive an error. The following message is an example of the error:

Error xdblade31b04/WASMaintModeDC1_xdblade31b04 has not been started.

The server cannot be stopped from this page when this situation occurs.

As a work-around, complete one of the following actions to stop the started middleware server:

- Click **Servers > Server Types > *type_of_middleware_server* > *middleware_server***. Then click **Stop** for a middleware server that displays a status of Started.
- Stop and then start the servers from the command line.

After you install the node agent on WebSphere Application Server Community Edition nodes and federate the nodes, middleware discovery can automatically create representations of these servers in the administrative console. You do not need to perform the manual steps to create the representation of the server in the administrative console. You can continue to manage these servers in the WebSphere Application Server Community Edition console. Make any representative changes in the administrative console.

Procedure

1. Create a representation of the middleware server.
 - For Apache Tomcat servers, use the **wsadmin** scripting command of **createTomCatServer**.
 - For BEA WebLogic servers, use the **wsadmin** scripting command of **createWebLogicServer**.
 - For JBoss servers, use the **wsadmin** scripting command of **createJBossServer**.
 - For external WebSphere application server, use the **wsadmin** scripting command of **createForeignServer**.
 - For other middleware server types, use the administrative console.
 - a. Add an existing server.

In the administrative console, click **Servers > Add a server** and select **Add an existing server**.
 - b. Select the node on which the middleware server that you are configuring runs.

The node must be running a node agent. Create a name for the server that is unique among all servers in the cell.
 - c. Specify the server template that you want to use for the middleware server.
 - d. Click **Finish**.
 - e. Click **Save** to save your changes to the master configuration.
2. Perform additional configuration steps for the middleware server, including adjusting the values for the WebSphere variables for each server type and configuring server operations to stop and start your servers:
 - For Apache Tomcat servers, read about configuring Apache Tomcat servers.
 - For BEA WebLogic servers, read about configuring BEA WebLogic servers.
 - For JBoss servers, read about configuring JBoss servers.

- For WebSphere Application Server Community Edition servers, read about configuring assisted life-cycle WebSphere Application Server Community Edition servers.
- For custom HTTP servers, read about configuring custom HTTP servers.
- For external WebSphere application server, you can manage previous versions of WebSphere application servers, from Version 5.1 and later. For more information, read about configuring external WebSphere application servers.

The WebSphere variables define settings for the middleware server such as the installation location and vary depending on the middleware server type. By editing the server start and stop operations, you can specify the username and password that is required to start and stop the servers in the middleware server.

3. Start the middleware server.

- For Apache Tomcat servers, BEA WebLogic servers, JBoss servers, and external WebSphere application server, use the `startMiddlewareServer` command.
- For other assisted lifecycle middleware servers, in the administrative console, click **Servers > All servers**. Select the server that you want to start and click **Start**.

The start server operation that is defined for the server runs.

Note: If the node agent and the server are stopped at the same time, the last known status of the server is reported. Because the last known status is reported as started, the on-demand router (ODR) tries to route to the server.

What to do next

You can create a middleware server template that is based on your configured server. After you create a middleware server template, you can create additional servers using the template, which creates servers with the same settings as your original server.

To easily manage groups of existing servers to host an application, configure dynamic clusters. By configuring a dynamic cluster, the product can adjust the number of running servers to meet the application service policy. For assisted lifecycle middleware servers, you group together the representations that you created. These servers must have the same applications installed.

The logs and trace views in the administrative console are not supported for assisted lifecycle middleware servers. For Apache Tomcat servers, BEA WebLogic servers, JBoss servers, and external WebSphere application server, you can view logs for these middleware servers on the machines that host them. For other assisted lifecycle middleware servers, configure the external log viewing service to view the log files in the administrative console.

Configuring assisted life-cycle WebSphere Application Server Community Edition servers:

After you manually create a middleware server representation of a WebSphere Application Server Community Edition server in the administrative console, you must update a few configuration settings so that they are defined specifically for your WebSphere Application Server Community Edition installation. If the middleware discovery function is enabled, you do not need to perform these steps.

Before you begin

Before you perform this task, consider the other options for configuring your WebSphere Application Server Community Edition servers:

- If you do not have an existing WebSphere Application Server Community Edition configuration, consider using complete life-cycle servers. By configuring complete life-cycle servers, you can deploy managed applications and create expression-based dynamic clusters that are made of WebSphere Application Server Community Edition servers. For more information, read about creating complete life-cycle WebSphere Application Server Community Edition servers and dynamic clusters.

- If you enable middleware discovery, representations of your existing WebSphere Application Server Community Edition servers and applications are automatically created. These representations are assisted life-cycle servers and unmanaged applications. You do not need to perform any of the additional steps in this task if middleware discovery is enabled. To enable middleware discovery for WebSphere Application Server Community Edition servers, click **System administration > Middleware descriptors > wasceRuntime**. If the **Discovery interval** value is set to -1, discovery is disabled. Change this value to indicate the interval at which you want middleware discovery to run.

If middleware discovery is disabled, you must perform the manual steps in this task to create a representation of your WebSphere Application Server Community Edition in the Intelligent Management cell. As a prerequisite, federate the node that is running the WebSphere Application Server Community Edition server and create a representation of the server in the administrative console.

About this task

Perform this task only if you have an existing WebSphere Application Server Community Edition environment and you do not want to enable middleware discovery to automatically create a representation of your environment.

Procedure

1. Update the WebSphere variables so that they match the settings on the node that is running the WebSphere Application Server Community Edition server. In the administrative console, click **Environment > WebSphere variables**.
 Edit the *WASCE_HOME* variable, which specifies the home directory of your WebSphere Application Server Community Edition installation. Default values are set at the cell scope level:
 - **Windows** C:\Program Files\IBM\WebSphere\AppServerCommunityEdition
 - **AIX** **HP-UX** **Solaris** /opt/IBM/WebSphere/AppServerCommunityEdition
 If the home directory is different on different nodes in your configuration, select the node scope and create the *WASCE_HOME* variable for the specific node.
2. Update server operations for the WebSphere Application Server Community Edition server to include the user name and password.
 - a. In the administrative console, click **Servers > Server types > WebSphere Application Server Community Edition servers > wasce_server > Server operations**.
 - b. Edit the start and stop server operations to include the user name and password that is required to run these commands.

Results

You created a representation of your WebSphere Application Server Community Edition server in the administrative console. To view all of your WebSphere Application Server Community Edition servers, click **Servers > Server types > WebSphere Application Server Community Edition servers**.

The list of servers that displays on this administrative console page includes all of your complete life-cycle, assisted life-cycle, and discovered WebSphere Application Server Community Edition servers. If you have WebSphere Application Server Community Edition dynamic clusters defined, the cluster members are also listed on this administrative console page.

What to do next

- To use this configuration when you are configuring your other WebSphere Application Server Community Edition servers, create a server template of your current WebSphere Application Server Community Edition server.

- To view the log files for your WebSphere Application Server Community Edition server in the administrative console, use the external log viewing service. This service is enabled by default. The following directories are configured by default:
 - `${WASCE_HOME}/var/logs`
 - `${WASCE_HOME}/catalina/logs`
 - `${AGENT.HOME}/logs/${WAS_SERVER_NAME}`
- To edit configuration files for your WebSphere Application Server Community Edition server, use the external configuration service. The `${WASCE_HOME}/var/config/config.xml` file is included by default.
- After you create representations of all the WebSphere Application Server Community Edition servers, you can add the representations to a dynamic cluster. To cluster these servers, each server must be running the same applications and be running the same version of WebSphere Application Server Community Edition.

Configuring Apache Tomcat servers:

After you create a middleware server representation of an Apache Tomcat server, you must update a few configuration settings so that they are defined specifically for your Apache Tomcat installation.

Before you begin

Federate the node that is running the Apache Tomcat server and create a representation of the server.

About this task

If you installed Apache Tomcat in a location other than the default directory, update the WebSphere variables so that the installation directory is correct. To perform server operations on your server, you must configure the server operations for the Apache Tomcat middleware server representation.

Note: Intelligent Management support for using Apache Tomcat servers is deprecated. Support for using the administrative console to configure these servers is removed. Use `wsadmin` scripting to manage these servers.

Procedure

1. Update the WebSphere variables so that they match the settings on the node that is running the Apache Tomcat server. Follow the directions in the topic on modifying variables using `wsadmin` scripting to edit the following variables:

CATALINA_HOME

Specifies the home directory of your Apache Tomcat installation.

Default values are set at the cell scope level:

Windows C:\Program Files\Apache Software Foundation\Tomcat

AIX **HP-UX** **Solaris** /usr/local/apache-tomcat

If the home directory is different on different nodes in your configuration, create the `CATALINA_HOME` variable for a specific node.




JAVA_HOME

Specifies the directory of your Java Development Kit (JDK) installation.

If the `JAVA_HOME` variable is not set in the environment of the node, define the `JAVA_HOME` variable at the cell scope or at the node scope.

2. Launch `wsadmin` scripting with a username and password so that you can start and stop server operations.

Follow the directions in the topic on starting the `wsadmin` scripting client to enable security. The `startMiddlewareServer` command and the `stopMiddlewareServer` command require a user name and password to run.

- Optional: Update the HTTP and HTTPS ports for your servers. The default port is 8080. If you use a value other than the default, you must change the port. Use the `modifyForeignServerProperty` command in the middleware server management administrative tasks to change the ports.
-    Optional: Disable security for the Tomcat server. By default, security is enabled when you create the server. Use the `modifyForeignServerProperty` command in the middleware server management administrative tasks to start the server without security. If you want to create a new Tomcat server that does not have security enabled, you must create a custom Tomcat server template. Read the topic on creating middleware server templates.

What to do next

- To use this configuration when you are configuring your other Apache Tomcat servers, create a server template of your current Apache Tomcat server.
- View the log files for your Apache Tomcat server on the machine that hosts it.
- Edit configuration files for your Apache Tomcat server on the machine that hosts it.
- After you create representations of all the Apache Tomcat servers, you can add the representations to a dynamic cluster.

Configuring BEA WebLogic servers:

After you create a middleware server representation of a BEA WebLogic server, you must update configuration settings so that they are defined specifically for your BEA WebLogic installation.

Before you begin

Federate the node that is running the BEA WebLogic server and create a representation of the server.

About this task

If you installed BEA WebLogic in a location other than the default directory, update the WebSphere variables so that the installation directory is correct. To perform operations on your server, you must configure the server operations for the BEA WebLogic middleware server representation.

Note: Intelligent Management support for using BEA WebLogic servers is deprecated. Support for using the administrative console to configure these servers is removed. Use `wsadmin` scripting to manage these servers.

Procedure

- Update the WebSphere variables so that they match the settings on the node that is running the BEA WebLogic server. Follow the directions in the topic on modifying variables by using `wsadmin` scripting to edit the following variables:

WEBLOGIC_ADMINHOST

Specifies the host name of the WebLogic administration server.

Default value: localhost

Scope: Cell

WEBLOGIC_ADMINPORT

Specifies the port of the WebLogic administration server.

Default value: 7001

Scope: Cell

WEBLOGIC_ADMINPROTOCOL

Specifies the protocol to use when connecting to the WebLogic administration server.

Default value:t3

Scope: Cell

WEBLOGIC_ADMINURL

Specifies the URL for the WebLogic administration server. The value is created from other variables.

Default value: `${WEBLOGIC_ADMINPROTOCOL}://${WEBLOGIC_ADMINHOST}:${WEBLOGIC_ADMINPORT}`

Scope: Cell

WEBLOGIC_SERVERROOT

Specifies an alias to the `WEBLOGIC_DOMAINDIR` variable.

Default value: `${WEBLOGIC_DOMAIN_DIR}`

Scope: Cell

WEBLOGIC_SERVERNAME

Specifies the name of the server in the WebLogic administration environment.

Default value: `${WAS_SERVER_NAME}`

Scope: Server

BEA_HOME

Specifies the root installation location.

 Default value: C:\bea

     Default value: /opt/bea

Scope: Cell

WEBLOGIC_HOME

Specifies the WebLogic root directory. The root is version-specific, for example `${BEA_HOME}\weblogic91`.

Default value:`${BEA_HOME}\weblogic`

Scope: Cell

WEBLOGIC_DOMAINNAME

Specifies the name of the WebLogic domain.

Default value: myDomain

Scope: Cell

WEBLOGIC_DOMAINDIR

Builds the path to the WebLogic domain.

Default value:`${BEA_HOME}\user_projects\domains\${WEBLOGIC_DOMAINNAME}`

Scope: Cell

If the home directory is different on different nodes in your configuration, create another variable with the same name for a specific node.

2. Launch **wsadmin** scripting with a username and password so that you can start and stop server operations.

Follow the directions in the topic on starting the **wsadmin** scripting client to enable security. The **startMiddlewareServer** command and the **stopMiddlewareServer** command require a user name and password to run.

- Optional: Update the HTTP and HTTPS ports for your servers. The default port is 7001. If you use a value other than the default, you must change the port. Use the **modifyForeignServerProperty** command in the middleware server management administrative tasks to change the ports.

What to do next

- To use this configuration when you are configuring your other BEA WebLogic servers, create a server template of your current BEA WebLogic server.
- View the log files for your BEA WebLogic server on the machine that hosts it.
- Edit configuration files for your BEA WebLogic server on the machine that hosts it.
- After you create representations of all the BEA WebLogic servers, you can add the representations to a dynamic cluster.

Configuring JBoss servers:

After you create a middleware server representation of a JBoss server, you must update a few configuration settings so that they are defined specifically for your JBoss installation.

Before you begin

Federate the node that is running the JBoss server and create a representation of the server.

About this task





If you installed JBoss in a location other than the default directory, update the WebSphere variables so that the installation directory is correct. To perform server operations on your server, you must configure the server operations for the JBoss middleware server representation.

Note: Intelligent Management support for using JBoss servers is deprecated. Support for using the administrative console to configure these servers is removed. Use **wsadmin** scripting to manage these servers.

Procedure

- Update the WebSphere variables so that they match the settings on the node that is running the JBoss server. Follow the directions in the topic on modifying variables using **wsadmin** scripting to edit the following variables:

Edit the **JBOSS_DIST** variable, which 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. Default values are set at the cell scope level:

-  `C:\Program Files\jboss`
-    `/usr/local/jboss`

Edit the **JBOSS_PROFILE** variable, which specifies the configuration set in use for the server representation, and defaults to the name of the server itself. The default value is set at the server scope level: `${WAS_SERVER_NAME}`. If these settings are different on different nodes in your configuration, create the variables for a specific node.

- Launch **wsadmin** scripting with a username and password so that you can start and stop server operations.

Follow the directions in the topic on starting the **wsadmin** scripting client to enable security. The **startMiddlewareServer** command and the **stopMiddlewareServer** command require a user name and password to run.

- Optional: Update the HTTP and HTTPS ports for your servers. The default port is 8080. If you use a value other than the default, you must change the port. Use the **modifyForeignServerProperty** command in the middleware server management administrative tasks to change the ports.

What to do next

- To use this configuration when you are configuring your other JBoss servers, create a server template of your current JBoss server.
- View the log files for your JBoss server on the machine that hosts it.
- Edit configuration files for your JBoss server on the machine that hosts it.
- After you create representations of all the JBoss servers, you can add the representations to a dynamic cluster.

Configuring custom HTTP servers:

After you create a middleware server representation of a custom HTTP server in the administrative console, you must update a few configuration settings so that they are defined specifically for your custom HTTP server installation.

Before you begin

Federate the node that is running the custom HTTP server and create a representation of the server in the administrative console.

About this task

To perform server operations on your server from the administrative console, you must configure the server operations for the custom HTTP server middleware server representation.

Procedure

Configure server operations for the custom HTTP server to include the user name and password.

1. In the administrative console, click **Servers > Server types > Custom HTTP servers > *custom_http_server* > Server operations**.
2. Edit the start and stop server operations.
3. Define the location of the process ID (PID) file on the server. Create the PID file name environment variable. The PID file name specifies the location of the PID file where the server operation runs. To define the PID file name variable, click **Servers > All servers > *custom_http_server* > Variables > New** or **Environment > WebSphere variables > New**. Enter the same name for the variable that you entered in the server operations panel, and for the value, enter the fully qualified or variable qualified location of the PID file.

What to do next

- To use this configuration when you are configuring your other custom HTTP servers, create a server template of your current custom HTTP server.
- Use the external log viewing service to view the log files for your custom HTTP server from the administrative console.
- Use the external configuration service to edit the configuration files for your custom HTTP server from the administrative console.
- After you create representations of all the custom HTTP servers, you can add the representations to a dynamic cluster.

Configuring external WebSphere application servers:

You can create middleware server representations of WebSphere application servers that are in other cells.

Before you begin

- If you have a large topology of WebSphere application servers, you can use the migration toolkit to create the representations of these servers in the environment.
- Federate the node that is running the WebSphere application server and create a representation of the server.

About this task

External WebSphere application servers are application servers that reside outside of your Intelligent Management cell. For example, you might have two cells: one cell that is running WebSphere Application Server application servers, and another cell that is running Intelligent Management. In this scenario, you can install the node agent on the WebSphere nodes in the other cell, federate these nodes into your Intelligent Management cell, and create representations of your external WebSphere application servers. Using external WebSphere application servers can be useful in a migration scenario because you can continue to route traffic to servers that are running an older version with the ODR while you work on migrating your servers and applications to the latest version. However, external WebSphere application servers are assisted life-cycle servers and therefore do not receive the full life-cycle management that is provided when you create application servers directly in your Intelligent Management cell.

When you configure the external WebSphere application servers, if you installed WebSphere Application Server in a location other than the default directory, update the WebSphere variables so that the installation directory is correct. To perform server operations on your server, you must configure the server operations for the WebSphere Application Server middleware server representation.

Note: Intelligent Management support for using external WebSphere application servers is deprecated. Support for using the administrative console to configure these servers is removed. Use `wsadmin` scripting to manage these servers.

Procedure

1. Update the WebSphere variables so that they match the settings on the node that is running the WebSphere Application Server. Follow the directions in the topic on modifying variables using `wsadmin` scripting to edit the following variables:

WAS51_HOME

Specifies the location of the WebSphere Application Server installation.

Default values are set at the cell scope level:

Windows c:\Program Files\IBM\WebSphere\AppServer
AIX **HP-UX** **Solaris** /opt/IBM/WebSphere/AppServer

If the home directory is different on different nodes in your configuration, create the `WAS51_HOME` variable for a specific node.

WAS6_HOME

Specifies the location of the WebSphere Application Server Version 6 installation location.

Default values are set at the cell scope level:

Windows c:\Program Files\IBM\WebSphere\profiles\AppServer
AIX **HP-UX** **Solaris** opt/IBM/WebSphere/profiles/AppServer

If the home directory is different on different nodes in your configuration, create the `WAS6_HOME` variable for a specific node.

WAS6_PROFILE_NAME

Specifies the name of the profile in which the external WebSphere Application Server server resides on its physical computer.

2. Launch `wsadmin` scripting with a username and password so that you can start and stop server operations.

Follow the directions in the topic on starting the `wsadmin` scripting client to enable security. The `startMiddlewareServer` command and the `stopMiddlewareServer` command require a user name and password to run.

What to do next

- To easily configure your remaining external WebSphere application servers, create a server template of your current external WebSphere Application Server.
- View the log files for your external WebSphere Application Server on the machine that hosts it.
- Edit configuration files for your external WebSphere Application Server on the machine that hosts it.
- After you create representations of all the external WebSphere application servers, you can add the representations to a dynamic cluster.

Managing middleware servers

To define middleware servers, you can use middleware server templates and middleware descriptors. To manage middleware servers that are defined, you can view your middleware server log files and configuration files and configure server operations to run on your middleware servers.

Before you begin

To configure middleware server templates, middleware server operations, the external configuration service, or the external log viewing service, you must create the middleware server. For more information, read about adding middleware servers to configurations.

You can configure middleware descriptors before or after you add your middleware servers.

Procedure

- Configure middleware descriptors. If discovery applies to your specific platform, `mMiddleware` descriptors provide information about your middleware platform, such as the installation location and the interval at which an installation can be discovered. For more information, read about configuring middleware descriptors.
- Define middleware server templates. By creating a middleware server template, you copy all of the configuration data for that server so that you can create other servers from the template. If you have many middleware servers with a similar configuration, middleware server templates can make this configuration easier. For more information, read about creating middleware server templates.
- Configure middleware server operations. For an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server, use `wsadmin` scripting to use middleware server operations to run executable files. For other middleware servers, you can use the administrative console. Middleware operations can perform tasks such as enabling or disabling tracing, starting or stopping servers, or querying the running state of the server. For more information, read about configuring middleware server operations.
- View and edit configuration files for your middleware servers. You can specify the path for the configuration files that you want to view or edit in the administrative console. For more information, read about viewing middleware server configuration files.

gotcha: You cannot use the external configuration service with an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can view and edit configuration files for these middleware servers on the machines that host them.

- Configure the external log viewing service. With the external log viewing service, you can view the log files from your middleware servers in the administrative console. For more information, read about configuring the external log viewing service.

gotcha: You cannot use the external log viewing service with an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can view logs for these middleware servers on the machines that host them.

Configuring middleware descriptors:

Middleware descriptors provide information about different middleware platform types that includes middleware discovery intervals, enablement, and installation information.

Before you begin

If you want to add or change installation information to the middleware descriptor, you must know the installation paths of the middleware software that you want to configure.

About this task

You can edit the following middleware descriptors that are provided with the product:

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

Procedure

1. Edit middleware platform descriptor properties.
 - Use the middleware descriptor administrative tasks to edit the properties for an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can also use these tasks for other middleware servers.
 - For other middleware servers, in the administrative console, click **System administration > Middleware descriptors > *middleware_platform_name***.
If you are using an Apache HTTP server, WebSphere Application Server Community Edition, or PHP run time: Configure the interval at which the discovery function of the middleware platform runs. Supply a **Discovery interval** and units for this discovery interval in seconds, minutes, hours, or days. To disable middleware discovery, enter -1 in the **Discovery interval** field. Click **OK** to save these changes. This automated discovery is supported for Apache, WebSphere Application Server Community Edition, and PHP run times only.
2. Edit the middleware platform version details.
 - Use the middleware descriptor administrative tasks to edit the version details for an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can also use these tasks for other middleware servers. The version details vary depending on the middleware platform.
 - Click the version that you want to edit. The version properties vary depending on the middleware platform type, for example, for some types, you can edit the session affinity descriptor.
For Apache web servers, WebSphere Application Server Community Edition servers, and PHP servers, if you installed the phpRuntime, wasceRuntime, or apacheWebServerRuntime to a location other than the default path, you can edit the installation location. You can also define multiple installation locations. Do not include installation locations for server installations for which discovery is not preferred, such as centralized installation manager installations of WebSphere Application

Server Community Edition. If you change or remove an installation location from servers that were previously discovered, changes to those servers are no longer discovered.

Results

When enabled, automated discovery can run for Apache, WebSphere Application Server Community Edition, and PHP run times that you configured.

Creating middleware server templates:

By creating a middleware server template, you save a copy of server configuration data that you can use as a starting point when you create a server.

Before you begin

Create a middleware server from which you want to create a template. For more information, read about adding middleware servers to configurations.

About this task

When you create a server template, you copy all of the configuration data for that server so that you can create other servers from the template. If your middleware servers each have a similar configuration, then a middleware server template can make creating additional servers easier.

Procedure

- Use middleware server template administrative tasks to create a server template for Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server.

You can also use the administrative tasks to create server templates for other middleware servers.

- For middleware servers other than Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server, you can use the administrative console to create server templates.
 1. Navigate to the server templates collection page. In the administrative console, click **All servers > Templates**. The list of existing server templates is displayed.
 2. To create a middleware server template, click **New**. Choose the type of server template that you want to create.
 3. Select an existing server from which to create the server template. Select a server and click **OK**.
 4. Specify a name and description for the server template.
 5. Click **OK**. To save the template to the master configuration, click **Save**.

Results

A server template is created based on the configuration of the existing server.

What to do next

When you create a middleware server that is the same type of server that you selected to create your template, you can choose the template as the starting point for the configuration of the new server.

Configuring middleware server operations:

You can use server operations to run executable files on your middleware servers from the administrative console.

Before you begin

- Create and configure the middleware server. For more information, read about adding middleware servers to configurations.
- Create a Java or non-Java executable file that performs the operation that you want to run on your middleware server. You specify this file when you create the server operation.

About this task

Server operations consist of Java or non-Java process definitions that you can define to run on middleware servers. You can create these operations to perform tasks such as enabling or disabling tracing, starting or stopping applications, and querying the running state of the server.

Procedure

- Use middleware server management administrative tasks to specify server operations for Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server.
You can also use the administrative tasks to specify server operations for other middleware servers.
- For middleware servers other than Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server, you can use the administrative console to specify server operations.
 1. Navigate to the server operations wizard. In the administrative console, click **Servers > All servers > *middleware_server* > Server operations > New**.
 2. Specify the properties for your server operation. The properties are different if you choose a Java or non-Java server operation.

For operations that run non-Java and Java executable files:

- Specify a name for the action, the name of the executable file, and any arguments to pass to the executable file.
- Specify variables for the user name and password, and specify the actual values to pass for the user name and password at the time the executable file runs.
- Specify the operating systems on which the executable files can run.
- Specify a working directory, which is where to run the executable file.

For operations that run Java executable files, you must also specify the following properties:

- Specify the location of the Java executable directory.
 - Specify the type of target: a Java class, or an executable Java archive (JAR) file.
 - Specify the environment variable that stores the process ID (PID) file name. The PID file name is stored in a WebSphere variable. Remember the name of the variable so that you can create the variable in a later step.
3. Create the PID file name environment variable. The PID file name specifies the location of the PID file where the server operation runs. To define the PID file name variable, click **Servers > All servers > *middleware_server* > Variables > New** or **Environment > WebSphere variables > New**. Enter the same name for the variable that you entered in the server operations panel, and for the value, enter the fully qualified or variable qualified location of the PID file.
 4. To run the server operation, select the operation from the list and click **Execute**.

Viewing middleware server configuration files:

You can view and edit configuration files for your middleware servers in the administrative console with the external configuration service.

Before you begin

- Create and configure the middleware server in the Intelligent Management environment. Read about adding middleware servers to configurations for more information.
- You must know the fully qualified path name of the configuration file that you want to view on the middleware server. This configuration file must be less than 10 megabytes.

About this task

By configuring the external configuration service, you can view and edit the configuration files for middleware servers from the administrative console.

gotcha: You cannot use the external configuration service with an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can view and edit configuration files for these middleware servers on the machines that host them.

Procedure

1. Enable the external configuration service for the middleware server. In the administrative console, click **Servers > All servers > *server_name* > External configuration service**. Select **Enable external configuration service**. Click **OK**.
2. Specify the configuration files that you want to edit or view in the administrative console. Depending on the server type, one or more configuration files might be listed by default. To add a file to this list, enter the fully qualified path name of the file, and specify if this file is writable. Click **Add** to add the file to the list. To change the file permission for a file, select the file and click **Toggle privileges**.
3. View and edit the configuration files. In the administrative console, click **Servers > All servers > *server_name***. Click the **External configuration** tab. Select the configuration file and click **Retrieve**. When you complete editing the file, click **Apply** to save your changes.

What to do next

View the log files for your middleware servers.

Configuring the external log viewing service:

To view the log files for externally created middleware servers, configure the external log viewing service. By using this service, you can view the log files for servers with the administrative console instead of viewing the log files on each node.

Before you begin

- Create and configure your middleware servers.
- You must know the path to the log files for the server.
- To add new log files to the service, you need administrator administrative privileges. If you want to view existing log file output, you need configurator administrative privileges.

About this task

By configuring the external log viewing service, you can view logs for your other middleware servers in the administrative console. By default, some log files are configured to be viewed in the external log viewing service with the server template for your middleware server type. You can add to this default list of log files.

Restriction: You cannot use the graphical logs and trace settings that are included in WebSphere Application Server for other middleware server types, including middleware agents. Instead, use the external log viewing service to view the logs for these server types.

gotcha: You cannot use the external log viewing service with an Apache Tomcat server, a JBoss application server, a BEA WebLogic server, or an external WebSphere Application Server. You can view logs for these middleware servers on the machines that host them.

Procedure

1. Navigate to the external log service panel for your middleware server. In the administrative console, click **Servers > All servers > *middleware_server* > External log service**.
2. To enable the remote log viewing services, select **Enable remote log viewing services**.
3. List all the directories that contain log files that you want to view using the remote log viewer. Type a directory name, and click **Add**. The list contains all the paths to the log files. You can use variables such as `${USER_APP_SERVER_ROOT}` in your file path.
4. Click **OK**. To save your changes to the master configuration, click **Save**.
5. View the log files. In the administrative console, click **Servers > All servers > *middleware_server***. Click the **Log viewer** tab. Select a log file to view, and click **Retrieve**.

Results

You can view the log file to troubleshoot any issues with your middleware server.

Deploying applications with defined service levels

By deploying applications to your middleware servers and dynamic clusters and defining service policies on those applications, you can meet the performance goals for your environment.

Before you begin

- Create the servers and dynamic clusters for your environment, specifically the servers to which you want to deploy applications. For more information, read about creating dynamic clusters and adding middleware servers to configurations
- Plan the classification and handling of traffic for your application.

About this task

Service policies provide the ability to differentiate applications according to levels of importance and target values that can be prioritized to meet specific goals.

Procedure

1. Deploy the application. Different processes are involved for the different application types:
 - **PHP Hypertext Preprocessor (PHP) applications:** You can deploy PHP applications to PHP deployment targets, including PHP servers and PHP dynamic clusters. For more information, read about deploying PHP applications.
 - **WebSphere Application Server Community Edition applications:** You can deploy Java Platform, Enterprise Edition (Java EE) application applications and Geronimo modules with Geronimo artifacts to WebSphere Application Server Community Edition deployment targets, including servers and dynamic clusters. For more information, read about deploying WebSphere Application Server Community Edition applications.
 - **Unmanaged web applications:** Unmanaged web applications are installed and configured outside of the product environment, typically on assisted life cycle middleware servers. Configure unmanaged web applications so that you can route HTTP requests to assisted life cycle servers. By providing some basic information about the application, including context roots, virtual hosts, and servers, the ODR can route HTTP requests to these applications. For more information, read about deploying unmanaged web applications.

- **Enterprise applications:** You can deploy enterprise applications to dynamic clusters that are running WebSphere Application Server application servers. For more information, read about deploying enterprise applications.
2. Define service levels with service policies. A service policy is a user-defined business goal, and correlates to transaction and work class components. For more information, read about defining a service policy.
 3. Classify application requests with service policy work classes. In the administrative console, click **Applications > All applications > *application_name***. Click the **Service policies** tab. Work classes contain Universal Resource Identifiers (URI), Web services, Enterprise JavaBeans (EJB), or Java Message Service (JMS) patterns and conditional rules for associating requests to transaction classes. Transaction classes are then associated with service policies. For more information, read about routing and service policies
 Alternatively, you can use the `workclassoperations.py` script to define and modify the settings of work classes. For more information, read about the `workclassoperations.py` script.
 4. Start the application.
 - **Enterprise applications:** In the administrative console, click **Applications > Application types > WebSphere enterprise applications**. Select the application, and click **Start**.
 - **PHP Hypertext Preprocessor (PHP) applications:** In the administrative console, click **Applications > All applications**. Select the PHP application that you want to start. Choose the **Start** action and click **Submit action**. This action starts all of the PHP servers that are associated with this application. To start the servers individually, click **Servers > Server types > PHP servers**.
 - **WebSphere Application Server Community Edition applications:** The application displays as started in the administrative console when the middleware server on which the application is deployed starts.
 - **Unmanaged web applications:** The application displays as started when the server on which the application is deployed starts.
Attention: If the server or node agent stops, the unmanaged web application might still show as started or partially deployed in the administrative console.

What to do next

- Use reporting to view statistics and performance of your applications, dynamic clusters, servers, and service policies.
- To achieve a healthy product environment, configure health policies. Health policies are similar to service policies, except that health policies are based on a health goal for the environment.
- To ensure positive application performance, such as applying interruption-free application updates, you can use the application edition manager to install new editions of your application.

Deploying enterprise applications

By deploying applications to dynamic clusters, requests to the application can be managed autonomically. The dynamic cluster can expand and contract with application server virtualization. By deploying applications to dynamic clusters, you enable application virtualization.

Before you begin

Verify that Intelligent Management is running and you have access to an application for installation. Configure a dynamic cluster. You must know the specific cell and dynamic cluster on which you want to deploy.

About this task

When you deploy your application to a dynamic cluster that is in automatic mode, the product can automatically control the size and placement of dynamic clusters on the nodes, according to the business goals of your application.

Procedure

1. Deploy the application to your dynamic cluster. In the administrative console, click **Applications > New application** or **Applications > Install new middleware application > Java Platform, Enterprise Edition (Java EE)**.

Restriction: The application name must be unique among all the applications that are deployed in the cell.

For example, you cannot deploy a PHP application and a Java EE application that have the same name.

2. On the **Map modules to application servers** panel in the wizard, map your application to the cell and dynamic cluster name, or the deployment target, that is specific to your dynamic cluster.
3. Complete the wizard panels and click **Finish**.

What to do next

Define a service policy for the newly deployed application.

Deploying unmanaged Web applications

You can create representations of unmanaged Web applications so that the on demand router (ODR) can route requests to applications that are installed outside of the product environment to assisted life cycle middleware servers.

Before you begin

- Create and configure your middleware servers and dynamic clusters. Read about middleware servers and dynamic clusters for more information.
- Deploy the application to your assisted life cycle middleware servers, using the steps that are specified for the software that your middleware server runs. Be sure that you know or have access to the deployment information for the application, including the module name, context root, virtual host name, and the server or cluster names on which you want to deploy the application.

About this task

To route HTTP requests to applications that are deployed on other middleware servers and classify these requests to service policies, the runtime environment needs information about the application. By creating an *unmanaged Web application*, you create a representation of the application that is installed on the middleware server that includes this information. Intelligent Management does not manage the life cycle of the application in any way, but can route requests to the application. The status of unmanaged Web applications is directly linked to the status of the server to which the unmanaged Web applications are deployed. The ODR can only route HTTP requests to these applications.

An unmanaged application for WebSphere Application Server Community Edition is a representation of an application that has been installed in the WebSphere Application Server Community Edition Version 1 console, specifically. With *middleware discovery*, an unmanaged WebSphere Application Server Community Edition application can also be a representation of a discovered application. Middleware discovery locates existing installations of WebSphere Application Server Community Edition servers and their installed applications, and creates the corresponding configuration to include servers and applications in the Intelligent Management cell.

Procedure

1. Navigate to the unmanaged Web application installation wizard. In the administrative console, click **Applications > Install new middleware application > Unmanaged web applications > New**.
2. Enter the general properties for the application. Specify a name for the application that is unique among all of the applications in the cell, and edition information about the application.

About application editions for unmanaged Web applications: Because the application is unmanaged, you cannot perform edition control commands such as rolling out or validating new editions. However, if you install another edition on the externally created servers, then you can create a new edition of the application in the administrative console by creating a new representation of the unmanaged Web application with a different edition name. You can then specify routing policies to control how to route the requests to the multiple editions.

3. Define the deployment properties of the application. Specify module, context root, and virtual host information for the application. Choose the servers and clusters on which the application is deployed by selecting deployment targets. You can filter the list of deployment targets by only clusters, only servers, or by entering a name. Click **Add** or **Remove** to specify which servers and clusters run the application. Click **Apply** to add the module and deployment targets to the module list.

The dynamic cluster to which you deploy an unmanaged Web application must be homogenous, which means the same application must be deployed to all the members of the dynamic cluster, and all members must be running the same middleware software, including the same version. When you add a cluster member that does not contain the same application as the cluster, the status of the application becomes partially deployed. The application is not automatically deployed to the newly added cluster member.

4. Confirm the new unmanaged Web application representation, and save the changes.
5. Start the application. With middleware applications, applications display as started in the administrative console when the middleware server on which the application runs is started. Read about configuring middleware server operations for more information about starting middleware servers from the administrative console.

Attention: If the server or node agent stops, the application might still show as started or partially deployed in the administrative console.

What to do next

Define service policies and routing policies for the application.

Deploying PHP applications

You can use the various qualities of service of Intelligent Management product to manage PHP-based applications in addition to Java Platform, Enterprise Edition 5 (Java EE 5) applications.

Before you begin

- Create a PHP deployment target by creating either a PHP dynamic cluster or a PHP server. Verify that you can start the deployment targets. For more information read about creating PHP servers and PHP dynamic clusters.
- Prepare your PHP application to be deployed. Your PHP application consists of an archive file and optional setup or clean-up scripts. The supported archive file types are zip, tar, tar.gz, tar.tgz and jar. The optional scripts run for each server on which the application is deployed. The setup script runs during the application installation, and the clean-up script runs when you uninstall the application to remove any residual artifacts. The setup scripts can include commands such as setting file permissions and creating or deleting directories.

About this task

You can experience the same qualities of service that the product provides for Java EE 5 applications for PHP applications.

Procedure

1. Navigate to the PHP application wizard. In the administrative console, click **Applications > Install new middleware application**. Choose the PHP application type, and click **Next**.
2. Specify the location of the application archive, setup, and clean-up scripts.

3. Specify installation options, including the application name, edition, and edition description. The application name must be unique among all of the applications that are deployed in the cell, including applications that are other types, such as Java EE 5 or unmanaged Web applications.
4. Define the deployment properties of the application.
 - a. Specify the context root and virtual host information for the application.
 - b. Choose the servers and clusters on which the application runs by selecting deployment targets. Choose PHP servers or PHP dynamic clusters. You can filter the list of deployment targets by only clusters, only servers, or by entering a name. Click **Add** or **Remove** to specify which servers and clusters run the application.
 - c. For an unmanaged Web application with multiple modules, click **Apply** to add the module and deployment targets to the module list.
5. Confirm the new PHP application, and save the changes. The application archive expands and the setup script runs.
6. Start the application. With middleware applications, applications start when you start the PHP server in the administrative console. Click **Servers > Server types > PHP servers**. Select the servers that you chose as deployment targets, and click **Start**. You can also start the application from **Applications > All applications** in the administrative console. However, doing so from this panel will start all the servers associated with this application. The same rule applies if you stop PHP applications from this panel.

Results

Your PHP application runs on the defined PHP deployment targets. If you deployed the application to a PHP dynamic cluster, then the dynamic cluster can add or remove PHP servers as needed.

What to do next

Define service policies and routing policies for the application. All application edition manager functions are available. When you roll out a new edition, the clean-up script from the first edition runs, the application archive of the new edition is expanded, and the setup script of the new edition runs.

Restriction: 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.

Deploying WebSphere Application Server Community Edition applications

From the administrative console, you can install managed WebSphere Application Server Community Edition Version 2.0 applications to servers and dynamic clusters.

Before you begin

- Create a WebSphere Application Server Community Edition deployment target by creating a complete life cycle WebSphere Application Server Community Edition server or dynamic cluster. Verify that you can start the deployment targets. For more information read about creating complete lifecycle WebSphere Application Server Community Edition servers and dynamic clusters.
- Prepare your WebSphere Application Server Community Edition application to be deployed. The application consists of an archive file and an external deployment plan, which are both deployed to the WebSphere Application Server Community Edition server.

Important: With managed WebSphere Application Server Community Edition applications, you must complete all deployment actions from within the administrative console only. Use the WebSphere Application Server Community Edition console only for functions that are not available from the administrative console.

About this task

Deploy a WebSphere Application Server Community Edition application to experience the same qualities of service that Intelligent Management provides for Java Platform, Enterprise Edition 5 (Java EE 5) applications and PHP applications.

Procedure

1. Open the WebSphere Application Server Community Edition application wizard. In the administrative console, click **Applications > Install new middleware application**. Select **WebSphere Application Server Community Edition** as the application type, and click **Next**.
2. Specify either the local or remote location of the application archive file and external deployment plan, and click **Next**.
3. Specify the name, edition, and edition description of the application. The application name must be unique among all of the applications that are deployed in the cell, including applications that are other types, such as Java EE 5 or unmanaged web applications.
4. Select the servers and clusters on which the application runs by selecting the deployment targets. You can filter the list of deployment targets by only clusters, only servers, or by entering a name. Click **Add** or **Remove** to specify which servers and clusters run the application.
5. Specify the location of the virtual host for the web modules.
6. Review the confirmation details of the WebSphere Application Server Community Edition application, and save the changes.
7. Start the application. The application displays as started in the administrative console when the middleware server on which the application is deployed starts. To start middleware servers from the administrative console, read about configuring middleware server operations.

Attention: If the server or node agent stops, the application might still show as started or partially deployed in the administrative console.

Results

Your WebSphere Application Server Community Edition application runs on the defined deployment targets.

What to do next

Define service policies and routing policies for the application. All application edition manager functions are available.

Defining a service policy

You can define service policies and, for most kinds of work requests, work classes to categorize and prioritize work requests. A service policy consists of a user-defined performance goal and an importance level, in some cases.

Before you begin

You must have administrative privileges to perform the following tasks:

- To create, modify, or remove service policies and transaction classes
- To modify rules through the rule builder

For more information, read about administrative roles and privileges

About this task

Service policies are related to work requests through transaction classes. Each work request belongs to exactly one transaction class, and each transaction class belongs to exactly one service policy. For most kinds of work requests, work classes are used to map incoming requests to transaction classes. Each work class is attached to a Java Platform, Enterprise Edition (Java EE) application and a basic request feature; URI prefix for HTTP, method name for IIOP, and bus+destination for Java Message Service (JMS). Each work class specifies how the relevant requests are classified into transaction classes. For generic server clusters and for SIP, work classes are not used; instead, the rules for classifying requests to transaction classes are You can use the service policy custom properties to provide service policy alerting for persistent service policy violations on a transaction class basis. For more information read about service policy custom properties.configured on the ODRs.

For SIP over UDP traffic, you must enable the admission control for CPU overload protection to prevent retransmissions from occurring because of CPU overload. When using admission control for CPU overload protection for SIP, the discretionary type of goal must NOT be used. Only the average response time or percentile response time goals should be used. The response time threshold specified in the goal should be well under the value of the client's T1 timer (which defaults to 500 milliseconds). The rejection average response time threshold (the value derived from the goal's response time threshold and the rejection policy configured on the ARFM control panel, should be less than the client's T1 timer. For information about enabling the admission control for CPU overload protection, read about configuring the autonomic request flow manager.

Restriction: When dialog/session orientation is enabled for HTTP or SIP, a service policy cannot apply to messages that are part of pre-existing dialogs or sessions, *and* messages that are NOT part of pre-existing dialogs or sessions.

Using the *service time*, or response time of a single request or small number of requests on a non-loaded or lightly loaded system, that is, how long a single request takes on a non-loaded system, then constructing service policies where the response times are less than that will never result in additional instances being started (either average or percentile response time goals). The system will determine that starting additional instances will not improve the capability of meeting the goal. For percentile goals, ARFM and APC are very sensitive to the relationship between the parameters in question: the Response Time Target (RTT), or the Goal Value, and Percentile Threshold (PCT), or the Goal Percentage.

Following are some sample ranges starting with the response time, increasing to the RTT of 2, 3, 4 times (and so on) of the single request service time. Low and high end PCTs are then provided. These can vary to some degree from application to application, but these ranges are given as a starting point for fine-tuning your service policies for you specific needs For percentile goals, the ARFM and the APC are aware of the relationship between the parameters, RTT, or the goal value, and PCT, or the goal percentage.

- RTT of two times the service time:
 - A tight percentile goal or PCT of 75% completing within the response time specified (RTT)
 - A loose percentile goal or PCT of 50% completing within the response time specified (RTT)
- RTT of three times the service time: PCTs from 88% to 65%
- RTT of four times the service time: PCTs from 94% to 76%
- RTT of five times the service time: PCTs from 97% to 83%
- RTT of six times the service time: PCTs from 99% to 88%
- RTT of seven times the service time: PCTs from 99% to 92%
- RTT of eight times the service time: PCTs from 99% to 94%

Procedure

1. From the administrative console click **Operational policies > Service policy**. You can select an existing service policy to edit, or click **New** to create a service policy. To edit an existing service policy, click the service policy name.
2. Create a name, description, and a goal type for your new service policy. The goal type can be either discretionary, average response time, or percentile response time:
 - A discretionary goal is the default, and indicates work that does not have significant value. As a result, work of this type can see a degradation in performance when resources are constrained.
 - Average response time goals are indicative of work with a higher priority than discretionary. The average response time goal is assigned a specific time goal.
 - Percentile response time goals are another measure for work with a higher priority than discretionary. The percentile response goals are defined with specific criteria on the following panel. The percentile response time target is the percentage of requests whose response time is T or less that should be P or more; a target has particular values for T and P.
3. Optional: If you select a goal type of average response time, or percentile response time, you are prompted to define the specifics and select an importance.

For average response time goals, type a goal value, associate an importance with the service policy, and select **Monitor for persistent policy violations** to set up the creation of a runtime task when a policy violation occurs.

When you associate an importance with the service policy, the options for importance vary from lowest to highest. Some planning is essential to select the correct importance value, because negative results can occur if all work is rated as highest. This rating can create a bottleneck within the environment. To define a policy violation, specify the **Goal delta value** and the **Time period value**:

- In the **Goal delta value** field, type an integer to indicate the maximum allowable amount of time that exceeds the configured goal value. Acceptable values are 0 to 3000 milliseconds, 0 to 300 seconds, and 0 to 2147483647 minutes.
- In the **Time period value** field, type an integer to indicate the milliseconds, seconds, or minutes after which the goal value is in violation. This value can be 0 to 1 day, inclusive.

For percentile response time, set the goal percentile to the percentage of requests that must meet the goal value that is defined in the next field. Next, type a goal value, associate an importance with the service policy, and select **Monitor for persistent policy violations** to set up the creation of a runtime task when a policy violation occurs.

For the goal value, type the maximum allowable time for the service policy. The environment tries to stay beneath the defined goals, and continually adjusts to achieve the most balanced result. When you associate an importance with the service policy, the options for importance vary from lowest to highest. Some planning is essential to select the correct importance value, because negative results can occur if all work is rated as highest. To define a policy violation, specify the **Goal delta percentage** and the **Time period value**:

- In the **Goal delta value** field, type an integer that indicates the percentage of request beneath the goal value for which to monitor. Acceptable values are 0 to 100, inclusive.
- In the **Time period value** field, type an integer to indicate the milliseconds, seconds, or minutes after which the goal value is in violation.

A runtime task is generated when certain criteria are violated. For example, in the following percentile response time example, with a percentile goal of 90% and a goal delta of 5%, the service policy is breached when less than 85% of requests meet the service time goal of 1 second (for 5 consecutive seconds), that is, when more than 15% of requests exceed the service time goal of 1 second (for 5 consecutive seconds). The system will still prioritize traffic in such a way as to attempt to meet the 90% goal, however no notification of a breach will be issued unless the 85% (90% minus 5%) threshold is not met.

Table 10. Percentile response time example

Description	Value
Goal percentile	90%
Goal value	1
Importance	1
Monitor for persistent service policy violations	true
Goal Delta Percentage:	5%
Time Period Value	5 seconds

For the goal value, type the maximum allowable time for the service policy. The environment continually adjusts all automatically adjustable controls, aiming to reach and maintain the best possible balance of relative performance results. When you associate an importance with the service policy, note that the options for importance vary from lowest to highest. Some planning is essential to select the correct importance value, because negative results can occur if all work is rated as highest. This rating can create a bottleneck within the environment.

4. Associate transaction class members to the service policy, or create a transaction class. If the transaction class that you are seeking does not exist, create a transaction class.
5. To create a work class for your service policy, from the administrative console click **Applications > Application types > WebSphere enterprise applications > application_name > Service policies**. Select an existing service policy and for the request type, click **New**.
 To create a service policy for HTTP, specify a name for the work class, select a module, and select the members to add. Optionally, to use a custom URI, type its name, and click **Add pattern** in the **Custom URI pattern** field. For example, a custom URI is necessary to do JavaServer Pages (JSP) work.
 To create a service policy for SOAP, specify a name for the work class, select a module, and select the Web service operations to add.
 To create a service policy for IIOP, specify a name for the work class, select a module, and select the EJB methods to add. Optionally, to use a custom EJB, type the information in the **Custom EJB name** and **Custom EJB method** fields, and click **Add pattern**.
 To create a service policy for JMS, type a name for the work class, select a module, select a defined bus, and select the EJB methods. Optionally, to use a custom bus, type the information in the **Custom bus name** and **Custom bus destination** fields, and click **Add pattern**.
 To create a service policy for SIP, you must create the following two policies:
 - a. Create a default SIP policy with the following values:
 - Goal Type = Average Response Time
 - Goal Value = 75 milliseconds
 - Importance = High
 - b. Create an INVITE policy with the following values:
 - Goal Type = Average Response Time
 - Goal Value = 75 milliseconds
 - Importance = Low
 - c. Set the service policy SIP rules:
 - If request.method = INVITE, then classify to transaction class Default _TC_INVITE (INVITE).
 - If no rules apply, then classify to transaction class Default _TC_def_sip (def_sip).
6. The system automatically picks up any changes you make to your service policy configuration. You do not need to restart any servers when you update your service policies and work classes.

Results

You have defined a business goal and applied that goal to application URIs using the service policy and routing rules. Your system can now categorize and prioritize work.

Routing and service policies

Two types of policies are applied to a request: routing and service. You can create routing policies for HTTP and SOAP requests, and you can create service policies for HTTP, IIOF, SOAP, JMS, and SIP requests. Additionally, work classes can contain classification rules for both policy types with the exception of JMS. Classification rules are not supported for JMS work classes.

Valid routing policies

Table 11. Routing policies

Routing policy	Description
<code>permit:application_name</code>	<code>application_name</code> is the application name to route to with an optional edition specifier.
<code>permitMM:application_name</code>	<code>application_name</code> is the application name to route to with an optional edition specifier. Routing in this way allows the request to continue as normal. Note that the server must be in maintenance mode.
<code>permitsticky:application_name</code>	<p>The <code>permitsticky</code> routing policy is the same as the <code>permit</code> routing policy, except that the on demand router (ODR) also maintains client-to-server affinity for any future requests that come from the same client. In this case, the ODR adds a <code>SET-COOKIE</code> header to the response before it sends the response to the client.</p> <p>The <code>permitsticky</code> action means that the ODR actively establishes affinity between the client and server if affinity was not already established by the application. The ODR accomplishes this by adding a <code>SET-COOKIE: WSJSESSIONID=xx:serverID; path=webModuleContextRoot</code> to the response if:</p> <ul style="list-style-type: none">• The response does not already have a <code>SET-COOKIE</code> that will establish server affinity, and• The corresponding request does not indicate that server affinity had already been established. <p>The <code>serverID</code> is the server identifier, and is also referred to as the clone ID. The <code>webModuleContextRoot</code> is the context root of the Web module to which the request was mapped.</p>
<code>permitstickyMM:application_name</code>	This routing policy is the same as the <code>permit</code> routing policy, except that the ODR also maintains client to server affinity for any future requests that come from the same client. In this case, the ODR adds a <code>SET-COOKIE</code> header to the response before it sends the response to the client. Note that the server must be in maintenance mode.
<code>reject:HTTP_error_code</code>	This routing policy causes the ODR to reject the request and return the specified HTTP error code. For example, <code>reject:503</code> returns a 503 Service is unavailable error.
<code>reject:URL</code>	With this routing policy, the ODR redirects the request to the specified URL. The URL has the pattern of <code>protocol://URI</code> . An example of a valid URL is <code>http://w3.ibm.com</code> .

Valid service policies

The valid service policies are the list of transaction class names. The transaction class refers to a single service class.

Subexpression builder operands for routing and service policies:

Operands and their associated protocols are supported from the initial installation, and can be used in the subexpression utility builder, an optional tool that helps you build complex rule conditions from subexpressions by using AND, OR, NOT and parenthetical grouping. An expression can be built from several sub-expressions using logical operators.

Intelligent Management includes routing and service policy rules that use the subexpression builder. The rule builder validates the rule when you apply the changes, and alerts you to mismatched parentheses, and unsupported logic operators. Choices of operands depend on the protocol you use.

SOAP operands:

Use SOAP operands in the subexpression utility builder, an optional tool that helps you build complex rule conditions from subexpressions by using AND, OR, NOT and parenthetical grouping.

Use the following subtopic links to find specific information:

- Rules
- “SOAP operands” on page 129
- “Xpath expression” on page 131
- “Operators” on page 132

Rules

Each work class contains an optional ordered list of rules that are evaluated for a particular request to determine the policy for that request. Each rule consists of a Boolean expression and a policy value. If the expression evaluates to true for a particular request, the policy associated with that rule is used.

The syntax and semantics of a Boolean expression for a rule are similar to the WHERE clause of a Structured Query Language (SQL) expression. More precisely, the syntax of an expression is defined by the Java Message Service (JMS) 1.1 specification. For more information about syntax and semantics of expressions, read about rule-based request classification.

In the JMS specification, identifiers refer to various attributes that can be associated with a request, for example, a specific query parameter, cookie, or HTTP header. A JMS identifier can be thought of as a request variable, or *operand*. These operands can be specific to a protocol. For example, the SOAP service name is an operand that is valid only in a SOAP work class.

Because the SOAP is over HTTP, the HTTP operands are also valid in a SOAP request. JMS specification uses literals to specify a specific value to use in a comparison to a request variable. For example, in the expression:

```
clienthost LIKE '%.ibm.com'
```

'%.ibm.com' is a literal that is used to compare to the client host name for a request. This expression is true for all requests that originate from a computer in the *ibm.com*[®] domain. Enclose string literals by single quotes. Do *not* enclose numeric literals in single quotes. Parentheses together with the AND, OR, and NOT operators can also be used to form compound Boolean expressions. See the JMS 1.1 specification for a detailed description.

SOAP operands

Table 12. Operands supported in SOAP

Operand	Syntax	Description
Client host	clienthost	The fully-qualified client host name. This is the value of the internet protocol (IP) command host name. This operand does <i>not</i> support numeric operators such as >, >=, <, <=.
Client IPV4	clientipv4	The IP address of the client using the Internet Protocol version 4 (IPv4) dotted quad address type <i>n.n.n.n</i> .
Client IPV6	clientipv6	The Internet Protocol version 6 (IPv6) 128-bit address type of <i>x:x:x:x:x:x:x:x</i> following Request for Comments 1924 (RFC 1924) of the client computer.
Cookie name	cookie\$<name>	A cookie name. For example, the expression <code>cookie\$My_Cookie_Name='My_Cookie_Value'</code> tests a request to see if it contains a cookie named <i>My_Cookie_Name</i> with a value of <i>My_Cookie_Value</i> . To test for the presence or absence of a particular cookie, use one of the following expressions: <code>cookie\$MyCookieName IS NOT NULL</code> <code>cookie\$MyCookieName IS NULL</code>
Header name	header \$<name>	A header name and value. For example, the expression <code>header\$Host='localhost'</code> tests a request to see if it contains an HTTP host header with a value of <i>localhost</i> . To test for presence or absence of the host header, use one of the following expressions: <code>header\$Host IS NOT NULL</code> <code>header\$Host IS NULL</code>
HTTP method	HTTPMethod	The HTTP method for the request. Possible values are POST, GET, PUT, and DELETE.
MIME type	MIMEType	The MIME type of the request.
Operation	operation	The name of a Web service operation.
Percentage	percentage\$<val>	The percentage operand evaluates to true, a fixed percentage of the time. For example, <code>percentage\$50</code> evaluates to true on average 50% of the time.
Port	port	The listening port on which the request was received.
Protocol	protocol	The communications protocol that transmits the request. Currently supported protocols are HTTP, HTTPS, SOAP, and SOAPS.
Query parameter	queryparm\$<name>	A header name and value. For example, the expression <code>queryparm\$timezone='EST'</code> tests a request to see if the request contains an HTTP query parameter named <i>timezone</i> with a value of <i>EST</i> . To test for presence or absence of a query parameter, use one of the following forms: <code>queryparm\$timezone IS NOT NULL</code> <code>queryparm\$timezone IS NULL</code>

Table 12. Operands supported in SOAP (continued)

Operand	Syntax	Description
Rampup	rampup\$<startTime> \$<completionTime>	<p>The rampup operand evaluates to true a variable percentage of the time. It always evaluates to false before <startTime> and to true after <completionTime>. As time progresses from <startTime> to <completionTime>, it evaluates to true, a linearly increasing percentage.</p> <p>The format of <startTime> and <completionTime> is day/month/year::hour:min:sec. where <i>day</i> is the day of the month, <i>month</i> is one of the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, <i>year</i> is the 4 digit year, <i>hour</i> is the 2 digit hour of the 24 hour clock, and <i>min</i> and <i>sec</i> are 2 digit values for minute and second, respectively.</p> <p>For example, rampup\$01/Jan/2007::08:00:00\$01/Jan/2007::17:00:00 begins to occasionally evaluate to true at 8 AM on Jan 1, 2007 and always evaluates to true by ramp up completion time at 5 PM of the same day.</p>
Server host	serverhost	The fully-qualified host name of the server. This operand does <i>not</i> support numeric operators such as >, >=, <, <=.
Server IPV4	serveripv4	The IP address of the server computer using the IPv4 dotted quad address type <i>n.n.n.n</i> .
Server IPV6	serveripv6	The IPv6 128-bit address type of <i>x:x:x:x:x:x:x:x</i> following RFC 1924 of the server computer.
Service	service	The name of a Web service.

Table 12. Operands supported in SOAP (continued)

Operand	Syntax	Description
Time	time	<p>Used to define the date and time of day that a given request must be honored. Two optional fields are StartTime and EndTime. If a request is received outside of that defined window, the request will not be processed.</p> <p>The Start Time and End Time fields each have the following format: dayOfWeek/dayOfMonth/month/year::hour:minute:second.</p> <p>For example, Thursday, the 11th of April, year 2007 at 1:03:45 PM is specified as:</p> <p>Thu/11/Apr/2007::13:03:45</p> <p>Any field can use a wild card with the value *.</p> <p>For example, the 1st of each month is specified as */1.</p> <p>The <i>dayOfWeek</i> values are Sun, Mon, Tue, Wed, Thu, Fri, Sat, and the <i>dayOfMonth</i> values range from 1-31.</p> <p>The <i>month</i> value is a non-numeric value that represents the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.</p> <p>The <i>year</i> value is comprised of the year's four digits. For example, 2007.</p> <p>The <i>hour</i> value is the hour of day in the 24-hour clock. For example, 8am is represented as ::8. The <i>minute</i> and <i>second</i> are integers ranging from 0-59.</p> <p>The forward slash (/) is used to separate date parameters, the double colon (::) is used to separate the date parameters, and the colon (:) is used to separate the time of day parameters. Note that it is the Boolean result of the entire rule in which the <i>time</i> operand is used that determines the routing action taken.</p>
Virtual Portal	virtualportal	<p>Virtual portals are created within WebSphere Portal Server, and Intelligent Management supports this operand for better integration with WebSphere Portal Server. The virtual portal is the request URL minus the context root for the WebSphere Portal application's Web module. If a given request matches the virtual portal defined, then the routing action defined for that rule is taken.</p>

Xpath expression

The syntax for the Xpath expression is xpathexpr and is supported by SOAP.

The XPath string expression has a required field that contains the XPath expression and an optional field for defining local namespace(s). If all the name spaces contained in the XPath expression are standard, then the second field can be optional. If there are multiple local namespaces, then separate each with a comma (,).

The following example shows an xpathexpr with one local namespace definition:

```

xpathexpr$/soap:Envelope/soap:Body/m:getTimeZone/n:clientId$m\=http://test.classify.ws.ibm.com,n\=http://test2.classify.ws.ibm.com
= \'1000\' or operation IS NOT NULL
    
```

In the previous expression , the XPath expression is /soap:Envelope/soap:Body/n:getTimeZone/n:clientId. It contains the local namespace of n. So, the second field of the xpathexpr is defined as n \\=http://test.classify.ws.ibm.com. The dollar sign (\$) is used to denote the start of a field definition. The double backslashes are the escape sequence in this example. The first equals sign (=) is escaped because it is part of the local namespace definition, whereas the second equals sign (=) is the operator in the classification expression and must not be escaped.

The following example shows an xpathexpr with two local namespaces:

```
xpathexpr$/soap:Envelope/soap:Body/m:getTimeZone/n:clientId$m\\=http://test.classify.ws.ibm.com,n\\=http://test2.classify.ws.ibm.com
= '\\1000\\' or operation IS NOT NULL
```

The following table shows the standard namespaces for the xpathexpr expression:

Table 13. Namespaces

Namespace	URL
soap	http://schemas.xmlsoap.org/soap/envelope/
soap-env	http://schemas.xmlsoap.org/soap/envelope/
soapenc	http://schemas.xmlsoap.org/soap/encoding/
soapbind	http://schemas.xmlsoap.org/wsdl/soap/
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
xsi	http://ws-i.org/schemas/conformanceClaim/
wsdl	http://schemas.xmlsoap.org/wsdl/

Operators

Intelligent Management supports the operators in the following table in the rules expressions. These operators are also referred to as *predicates* in SQL terminology because they appear inside of a WHERE or HAVING clause. Operators are case insensitive.

Table 14. Request classification operators

Operator	Description
OR	The logical OR operator.
AND	The logical AND operator.
NOT	The negation operator.
IN	<p>Expresses an operand with multiple values in a single expression. Its meaning is consistent with the SQL standard meaning of the operator.</p> <p>For example, if you want to express that the port value could be any or all of the values such as 9080, 9090, 9091, use the expression fragment:</p> <pre>port IN (9080,9090,9091)</pre> <p>In SQL, how the values inside the parenthesis are expressed depends on the data type of port. If the port is an integer, the values without the single quotation marks are syntactically correct. If the port is a string, the correct expression is:</p> <pre>port IN ('9080','9090','9091')</pre>

Table 14. Request classification operators (continued)

Operator	Description
LIKE	<p>Expresses pattern matching for string operand values. The value must contain the wildcard character (%) in the position where the pattern matching is expected to start.</p> <p>For example, the expression: host LIKE %blanca</p> <p>matches the word blanca and any other word that ends in blanca, while the expression: host LIKE blanca%</p> <p>matches the word blanca and any other word that starts with blanca. The expression: host LIKE %blanca%</p> <p>matches the word blanca and any word that has the token blanca imbedded in it.</p> <p>The <code>java.util.regex.Pattern</code> class is used.</p>
=	The equality operator expresses a match in case-sensitive match.
>	Greater-than operator for use with numeric operands.
>=	Greater-than or equal operator for use with numeric operands.
<	Less-than operator for use with numeric operands.
<=	Less-than or equal operator for use with numeric operands.
< >	Not-equal operator.
BETWEEN	Used with AND to select a range of values inclusive of the first (low) value and the last (high) value. Together, they operate on numbers and dates values.
IS NULL	Tests for an operand having a NULL value.
IS NOT NULL	Tests for an operand having a value other than NULL.

SIP operands:

You can use the SIP operands and their associated protocols in the subexpression utility builder, an optional tool that helps you build complex rule conditions from subexpressions by using AND, OR, NOT and parenthetical grouping.

Use the following subtopic links to find specific information:

- Rules
- “SIP operands” on page 134
- “Operators” on page 136

Rules

Each work class contains an optional ordered list of rules that are evaluated for a particular request to determine the policy for that request. Each rule consists of a Boolean expression and a policy value. If the expression evaluates to true for a particular request, the policy associated with that rule is used.

The syntax and semantics of a Boolean expression for a rule are similar to the WHERE clause of a Structured Query Language (SQL) expression. More precisely, the syntax of an expression is defined by

the Java Message Service (JMS) 1.1 specification. For more information about syntax and semantics of expressions, read about rule-based request classification.

In the JMS specification, identifiers refer to various attributes that can be associated with a request, for example, a specific query parameter, cookie, or HTTP header. A JMS identifier can be thought of as a request variable, or *operand*. These operands can be specific to a protocol. For example, the SOAP service name is an operand that is valid only in a SOAP work class.

Because the SOAP is over HTTP, the HTTP operands are also valid in a SOAP request. JMS specification uses literals to specify a specific value to use in a comparison to a request variable. For example, in the expression:

```
clienthost LIKE '%.ibm.com'
```

'%.ibm.com' is a literal that is used to compare to the client host name for a request. This expression is true for all requests that originate from a computer in the `ibm.com` domain. Enclose string literals by single quotes. Do *not* enclose numeric literals in single quotes. Parentheses together with the AND, OR, and NOT operators can also be used to form compound Boolean expressions. See the JMS 1.1 specification for a detailed description.

SIP operands

Table 15. Operands in the subexpression builder

Operand	Syntax	Description
Client host	<code>clienthost</code>	The fully-qualified client host name. This is the value of the internet protocol (IP) command host name. This operand does <i>not</i> support numeric operators such as <code>></code> , <code>>=</code> , <code><</code> , <code><=</code> .
Client IPV4	<code>clientipv4</code>	The IP address of the client using the Internet Protocol version 4 (IPv4) dotted quad address type <i>n.n.n.n</i> .
Client IPV6	<code>clientipv6</code>	The Internet Protocol version 6 (IPv6) 128-bit address type of <i>x:x:x:x:x:x:x:x</i> following Request for Comments 1924 (RFC 1924) of the client computer.
Contact header	<code>request.contact</code>	The Contact header field.
Contact URI	<code>request.contact.uri</code>	The URI of the Contact header field.
From header	<code>request.from</code>	The From header field.
From header display name	<code>request.from.display-name</code>	The display name in the From header field.
From URI	<code>request.from.uri</code>	The URI of the From header field.
From URI host	<code>request.from.uri.host</code>	The host in the From header field.
From URI port	<code>request.from.uri.port</code>	The port in the URI of the From header field.
From URI user	<code>request.from.uri.user</code>	The user in the From header field.
Header name	<code>header \$<name></code>	A header name and value. For example, the expression <code>header\$Host='localhost'</code> tests a request to see if it contains an HTTP host header with a value of <code>localhost</code> . To test for presence or absence of the host header, use one of the following expressions: <code>header\$Host IS NOT NULL</code> <code>header\$Host IS NULL</code>
Percentage	<code>percentage\$<val></code>	The percentage operand evaluates to true, a fixed percentage of the time. For example, <code>percentage\$50</code> evaluates to true on average 50% of the time.

Table 15. Operands in the subexpression builder (continued)

Operand	Syntax	Description
Port	port	The listening port on which the request was received.
Rampup	rampup\$<startTime> \$<completionTime>	<p>The rampup operand evaluates to true a variable percentage of the time. It always evaluates to false before <startTime> and to true after <completionTime>. As time progresses from <startTime> to <completionTime>, it evaluates to true, a linearly increasing percentage.</p> <p>The format of <startTime> and <completionTime> is day/month/year::hour:min:sec. where <i>day</i> is the day of the month, <i>month</i> is one of the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, <i>year</i> is the 4 digit year, <i>hour</i> is the 2 digit hour of the 24 hour clock, and <i>min</i> and <i>sec</i> are 2 digit values for minute and second, respectively.</p> <p>For example, rampup\$01/Jan/2007::08:00:00\$01/Jan/2007::17:00:00 begins to occasionally evaluate to true at 8 AM on Jan 1, 2007 and always evaluates to true by ramp up completion time at 5 PM of the same day.</p>
Request transport	request.transport	The transport of the request.
Request URI	request.uri	The request URI.
Request URI host	request.uri.host	The host in the request URI.
Request URI port	request.uri.port	The port in the request URI.
Request URI user	request.uri.user	The user in the request URI.
Scheme of From header	request.from.uri.scheme	The scheme of the From header field.
Scheme of To header	to.uri.scheme	The scheme of the To header field.
Scheme of URI	request.uri.scheme	The scheme of the URI.
Server host	serverhost	The fully-qualified host name of the server. This operand does <i>not</i> support numeric operators such as >, >=, <, <=.
Server IPV4	serveripv4	The IP address of the server computer using the IPv4 dotted quad address type <i>n.n.n.n</i> .
Server IPV6	serveripv6	The IPv6 128-bit address type of <i>x:x:x:x:x:x:x:x</i> following RFC 1924 of the server computer.
SIP method	request.method	The SIP method for the request. Possible values are INVITE, TRYING, RINGING, ACK, OK, and BYE. If the message is not a request, this operand returns null.
SIP response code	response.code	The response code of the response. If the message is not a response, this operand returns -1.
To client host	clienthost	The client host in the To header field.
To header	request.to	The To header field.
To header display name	request.to.display-name	The display name in the To header field.
To URI	request.to.uri	The URI of the To header field.
To URI host	request.to.uri.host	The port in the URI of the To header field.
To URI user	request.to.uri.user	The user in the To header field.

Operators

Intelligent Management supports the operators in the following table in the rules expressions. These operators are also referred to as *predicates* in SQL terminology because they appear inside of a WHERE or HAVING clause. Operators are case insensitive.

Table 16. Request classification operators

Operator	Description
OR	The logical OR operator.
AND	The logical AND operator.
NOT	The negation operator.
IN	<p>Expresses an operand with multiple values in a single expression. Its meaning is consistent with the SQL standard meaning of the operator.</p> <p>For example, if you want to express that the port value could be any or all of the values such as 9080, 9090, 9091, use the expression fragment:</p> <pre>port IN (9080,9090,9091)</pre> <p>In SQL, how the values inside the parenthesis are expressed depends on the data type of port. If the port is an integer, the values without the single quotation marks are syntactically correct. If the port is a string, the correct expression is:</p> <pre>port IN ('9080','9090','9091')</pre>
LIKE	<p>Expresses pattern matching for string operand values. The value must contain the wildcard character (%) in the position where the pattern matching is expected to start.</p> <p>For example, the expression:</p> <pre>host LIKE %blanca</pre> <p>matches the word blanca and any other word that ends in blanca, while the expression:</p> <pre>host LIKE blanca%</pre> <p>matches the word blanca and any other word that starts with blanca. The expression:</p> <pre>host LIKE %blanca%</pre> <p>matches the word blanca and any word that has the token blanca imbedded in it.</p> <p>The <code>java.util.regex.Pattern</code> class is used.</p>
=	The equality operator expresses a match in case-sensitive match.
>	Greater-than operator for use with numeric operands.
>=	Greater-than or equal operator for use with numeric operands.
<	Less-than operator for use with numeric operands.
<=	Less-than or equal operator for use with numeric operands.
< >	Not-equal operator.
BETWEEN	Used with AND to select a range of values inclusive of the first (low) value and the last (high) value. Together, they operate on numbers and dates values.
IS NULL	Tests for an operand having a NULL value.
IS NOT NULL	Tests for an operand having a value other than NULL.

IIOP operands:

Use IIOp operands and their associated protocols in the subexpression utility builder, an optional tool that helps you build complex rule conditions from subexpressions by using AND, OR, NOT and parenthetical grouping.

Use the following subtopic links to find specific information:

- Rules
- “IIOp operands”
- “Operators” on page 139

Rules

Each work class contains an optional ordered list of rules that are evaluated for a particular request to determine the policy for that request. Each rule consists of a Boolean expression and a policy value. If the expression evaluates to true for a particular request, the policy associated with that rule is used.

The syntax and semantics of a Boolean expression for a rule are similar to the WHERE clause of a Structured Query Language (SQL) expression. More precisely, the syntax of an expression is defined by the Java Message Service (JMS) 1.1 specification. For more information, read about rule-based request classification.

In the JMS specification, identifiers refer to various attributes that can be associated with a request, for example, a specific query parameter, cookie, or HTTP header. A JMS identifier can be thought of as a request variable, or *operand*. These operands can be specific to a protocol. For example, the SOAP service name is an operand that is valid only in a SOAP work class.

Because the SOAP is over HTTP, the HTTP operands are also valid in a SOAP request. JMS specification uses literals to specify a specific value to use in a comparison to a request variable. For example, in the expression:

```
clienthost LIKE '%.ibm.com'
```

'%.ibm.com' is a literal that is used to compare to the client host name for a request. This expression is true for all requests that originate from a computer in the *ibm.com* domain. Enclose string literals by single quotes. Do *not* enclose numeric literals in single quotes. Parentheses together with the AND, OR, and NOT operators can also be used to form compound Boolean expressions. See the JMS 1.1 specification for a detailed description.

IIOp operands

Table 17. Operands supported by IIOp

Operand	Syntax	Description
Application	application	The name of the enterprise application where the EJB is contained.
Client host	clienthost	The fully-qualified client host name. This is the value of the internet protocol (IP) command host name. This operand does <i>not</i> support numeric operators such as >, >=, <, <=.
Client port	clientport	The client port name.
EJB module	ejbmodule	The module name of an EJB.
EJB name	ejb	The name of an EJB.
EJB method	ejbmethod	The name of a method within the EJB.

Table 17. Operands supported by IIOP (continued)

Operand	Syntax	Description
Percentage	percentage\$<val>	The percentage operand evaluates to true, a fixed percentage of the time. For example, percentage\$50 evaluates to true on average 50% of the time.
Port	port	The listening port on which the request was received.
Rampup	rampup\$<startTime> \$<completionTime>	The rampup operand evaluates to true a variable percentage of the time. It always evaluates to false before <startTime> and to true after <completionTime>. As time progresses from <startTime> to <completionTime>, it evaluates to true, a linearly increasing percentage. The format of <startTime> and <completionTime> is day/month/year::hour:min:sec. where <i>day</i> is the day of the month, <i>month</i> is one of the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, <i>year</i> is the 4 digit year, <i>hour</i> is the 2 digit hour of the 24 hour clock, and <i>min</i> and <i>sec</i> are 2 digit values for minute and second, respectively. For example, rampup\$01/Jan/2007::08:00:00\$01/Jan/2007::17:00:00 begins to occasionally evaluate to true at 8 AM on Jan 1, 2007 and always evaluates to true by ramp up completion time at 5 PM of the same day.
Server host	serverhost	The fully-qualified host name of the server. This operand does <i>not</i> support numeric operators such as >, >=, <, <=.

Table 17. Operands supported by IIOP (continued)

Operand	Syntax	Description
Time	time	<p>Used to define the date and time of day that a given request must be honored. Two optional fields are StartTime and EndTime. If a request is received outside of that defined window, the request will not be processed.</p> <p>The Start Time and End Time fields each have the following format: dayOfWeek/dayOfMonth/month/year::hour:minute:second.</p> <p>For example, Thursday, the 11th of April, year 2007 at 1:03:45 PM is specified as:</p> <p>Thu/11/Apr/2007::13:03:45</p> <p>Any field can use a wild card with the value *.</p> <p>For example, the 1st of each month is specified as */1.</p> <p>The <i>dayOfWeek</i> values are Sun, Mon, Tue, Wed, Thu, Fri, Sat, and the <i>dayOfMonth</i> values range from 1-31.</p> <p>The <i>month</i> value is a non-numeric value that represents the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.</p> <p>The <i>year</i> value is comprised of the year's four digits. For example, 2007.</p> <p>The <i>hour</i> value is the hour of day in the 24-hour clock. For example, 8am is represented as ::8. The <i>minute</i> and <i>second</i> are integers ranging from 0-59.</p> <p>The forward slash (/) is used to separate date parameters, the double colon (::) is used to separate the date parameters, and the colon (:) is used to separate the time of day parameters. Note that it is the Boolean result of the entire rule in which the <i>time</i> operand is used that determines the routing action taken.</p>

Operators

Intelligent Management supports the operators in the following table in the rules expressions. These operators are also referred to as *predicates* in SQL terminology because they appear inside of a WHERE or HAVING clause. Operators are case insensitive.

Table 18. Request classification operators

Operator	Description
OR	The logical OR operator.
AND	The logical AND operator.
NOT	The negation operator.

Table 18. Request classification operators (continued)

Operator	Description
IN	<p>Expresses an operand with multiple values in a single expression. Its meaning is consistent with the SQL standard meaning of the operator.</p> <p>For example, if you want to express that the port value could be any or all of the values such as 9080, 9090, 9091, use the expression fragment:</p> <pre>port IN (9080,9090,9091)</pre> <p>In SQL, how the values inside the parenthesis are expressed depends on the data type of port. If the port is an integer, the values without the single quotation marks are syntactically correct. If the port is a string, the correct expression is:</p> <pre>port IN ('9080','9090','9091')</pre>
LIKE	<p>Expresses pattern matching for string operand values. The value must contain the wildcard character (%) in the position where the pattern matching is expected to start.</p> <p>For example, the expression:</p> <pre>host LIKE %blanca</pre> <p>matches the word blanca and any other word that ends in blanca, while the expression:</p> <pre>host LIKE blanca%</pre> <p>matches the word blanca and any other word that starts with blanca. The expression:</p> <pre>host LIKE %blanca%</pre> <p>matches the word blanca and any word that has the token blanca imbedded in it.</p> <p>The <code>java.util.regex.Pattern</code> class is used.</p>
=	The equality operator expresses a match in case-sensitive match.
>	Greater-than operator for use with numeric operands.
>=	Greater-than or equal operator for use with numeric operands.
<	Less-than operator for use with numeric operands.
<=	Less-than or equal operator for use with numeric operands.
<>	Not-equal operator.
BETWEEN	Used with AND to select a range of values inclusive of the first (low) value and the last (high) value. Together, they operate on numbers and dates values.
IS NULL	Tests for an operand having a NULL value.
IS NOT NULL	Tests for an operand having a value other than NULL.

HTTP operands:

Use the HTTP operands in the subexpression utility builder, an optional tool that helps you build complex rule conditions from subexpressions by using AND, OR, NOT and parenthetical grouping.

- Rules
- “HTTP request operands” on page 141
- “HTTP response operands” on page 143
- “Xpath expression” on page 144
- “Operators” on page 145

Rules

Each work class contains an optional ordered list of rules that are evaluated for a particular request to determine the policy for that request. Each rule consists of a Boolean expression and a policy value. If the expression evaluates to true for a particular request, the policy associated with that rule is used.

The syntax and semantics of a Boolean expression for a rule are similar to the WHERE clause of a Structured Query Language (SQL) expression. More precisely, the syntax of an expression is defined by the Java Message Service (JMS) 1.1 specification. For more information, read about rule-based request classification.

In the JMS specification, identifiers refer to various attributes that can be associated with a request, for example, a specific query parameter, cookie, or HTTP header. A JMS identifier can be thought of as a request variable, or *operand*. These operands can be specific to a protocol. For example, the SOAP service name is an operand that is valid only in a SOAP work class.

Because the SOAP is over HTTP, the HTTP operands are also valid in a SOAP request. JMS specification uses literals to specify a specific value to use in a comparison to a request variable. For example, in the expression:

```
clienthost LIKE '%.ibm.com'
```

'%.ibm.com' is a literal that is used to compare to the client host name for a request. This expression is true for all requests that originate from a computer in the `ibm.com` domain. Enclose string literals by single quotes. Do *not* enclose numeric literals in single quotes. Parentheses together with the AND, OR, and NOT operators can also be used to form compound Boolean expressions. See the JMS 1.1 specification for a detailed description.

HTTP request operands

Table 19. HTTP request operands

Operand	Syntax	Description
Client host	clienthost	The fully-qualified client host name. This is the value of the internet protocol (IP) command host name. This operand does <i>not</i> support numeric operators such as <code>></code> , <code>>=</code> , <code><</code> , <code><=</code> .
Client IPV4	clientipv4	The IP address of the client using the Internet Protocol version 4 (IPv4) dotted quad address type <i>n.n.n.n</i> .
Client IPV6	clientipv6	The Internet Protocol version 6 (IPv6) 128-bit address type of <i>x:x:x:x:x:x:x:x</i> following Request for Comments 1924 (RFC 1924) of the client computer.
Cookie name	cookie\$<name>	A cookie name. For example, the expression <code>cookie\$My_Cookie_Name='My_Cookie_Value'</code> tests a request to see if it contains a cookie named <i>My_Cookie_Name</i> with a value of <i>My_Cookie_Value</i> . To test for the presence or absence of a particular cookie, use one of the following expressions: <code>cookie\$MyCookieName IS NOT NULL</code> <code>cookie\$MyCookieName IS NULL</code>
Header name	header\$<name>	A header name and value. For example, the expression <code>header\$Host='localhost'</code> tests a request to see if it contains an HTTP host header with a value of <i>localhost</i> . To test for presence or absence of the host header, use one of the following expressions: <code>header\$Host IS NOT NULL</code> <code>header\$Host IS NULL</code>

Table 19. HTTP request operands (continued)

Operand	Syntax	Description
HTTP method	HTTPMethod	The HTTP method for the request. Possible values are POST, GET, PUT, and DELETE.
MIME type	MIMEType	The MIME type of the request.
Percentage	percentage\$<val>	The percentage operand evaluates to true, a fixed percentage of the time. For example, percentage\$50 evaluates to true on average 50% of the time.
Port	port	The listening port on which the request was received.
Protocol	protocol	The communications protocol that transmits the request. Currently supported protocols are HTTP, HTTPS, SOAP, and SOAPS.
Query parameter	queryparm\$<name>	A header name and value. For example, the expression queryparm\$timezone='EST' tests a request to see if the request contains an HTTP query parameter named <i>timezone</i> with a value of <i>EST</i> . To test for presence or absence of a query parameter, use one of the following forms: queryparm\$timezone IS NOT NULL queryparm\$timezone IS NULL
Rampup	rampup\$<startTime> \$<completionTime>	The rampup operand evaluates to true a variable percentage of the time. It always evaluates to false before <startTime> and to true after <completionTime>. As time progresses from <startTime> to <completionTime>, it evaluates to true, a linearly increasing percentage. The format of <startTime> and <completionTime> is day/month/year::hour:min:sec. where <i>day</i> is the day of the month, <i>month</i> is one of the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, <i>year</i> is the 4 digit year, <i>hour</i> is the 2 digit hour of the 24 hour clock, and <i>min</i> and <i>sec</i> are 2 digit values for minute and second, respectively. For example, rampup\$01/Jan/2007::08:00:00\$01/Jan/2007::17:00:00 begins to occasionally evaluate to true at 8 AM on Jan 1, 2007 and always evaluates to true by ramp up completion time at 5 PM of the same day.
Scheme of URI	request.uri.scheme	The scheme of the URI.
Server host	serverhost	The fully-qualified host name of the server. This operand does <i>not</i> support numeric operators such as >, >=, <, <=.
Server IPV4	serveripv4	The IP address of the server computer using the IPv4 dotted quad address type <i>n.n.n.n</i> .
Server IPV6	serveripv6	The IPv6 128-bit address type of <i>x:x:x:x:x:x:x:x</i> following RFC 1924 of the server computer.
Service	service	The name of a Web service.

Table 19. HTTP request operands (continued)

Operand	Syntax	Description
Time	time	<p>Used to define the date and time of day that a given request must be honored. Two optional fields are <code>StartTime</code> and <code>EndTime</code>. If a request is received outside of that defined window, the request will not be processed.</p> <p>The Start Time and End Time fields each have the following format: <code>dayOfWeek/dayOfMonth/month/year::hour:minute:second</code>.</p> <p>For example, Thursday, the 11th of April, year 2007 at 1:03:45 PM is specified as:</p> <p><code>Thu/11/Apr/2007::13:03:45</code></p> <p>Any field can use a wild card with the value <code>*</code>.</p> <p>For example, the 1st of each month is specified as <code>*/1</code>.</p> <p>The <i>dayOfWeek</i> values are Sun, Mon, Tue, Wed, Thu, Fri, Sat, and the <i>dayOfMonth</i> values range from 1-31.</p> <p>The <i>month</i> value is a non-numeric value that represents the twelve months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.</p> <p>The <i>year</i> value is comprised of the year's four digits. For example, 2007.</p> <p>The <i>hour</i> value is the hour of day in the 24-hour clock. For example, 8am is represented as <code>::8</code>. The <i>minute</i> and <i>second</i> are integers ranging from 0-59.</p> <p>The forward slash (/) is used to separate date parameters, the double colon (::) is used to separate the date parameters, and the colon (:) is used to separate the time of day parameters. Note that it is the Boolean result of the entire rule in which the <i>time</i> operand is used that determines the routing action taken.</p>
URI	uri	Uniform Resource Identifier
Virtual Host	virtualhost	Virtual host target of the request, used for configuring Web applications to a particular host name.
Virtual Port	numeric	Virtual port target of the request, used for configuring Web applications to a particular port.
Virtual Portal	virtualportal	Virtual portals are created within WebSphere Portal Server, and Intelligent Management supports this operand for better integration with WebSphere Portal Server. The virtual portal is the request URL minus the context root for the WebSphere Portal application's Web module. If a given request matches the virtual portal defined, then the routing action defined for that rule is taken.

HTTP response operands

When you are using a custom log file, you can use the following operands in addition to the operands in Table 1:

Table 20. HTTP response operands

Operand	Syntax	Description
Response code	response.code	Filters by HTTP response codes, such as 404, 503, and so on.

Table 20. HTTP response operands (continued)

Operand	Syntax	Description
Response time	response.time	The number of milliseconds between when the request in the ODR is received, and until the response from the ODR is sent.
Response write error	response.write.error	Logs errors that can occur when writing a response to a client.
Service time	service.time	The number of milliseconds between when the request is sent to the application server, and when a response is received from the application server.
Target server	targetserver	Shows the server where the request was sent, in a WebSphere Application Server format. For example, mycell1/mynode/myserver.

Xpath expression

The syntax for the Xpath expression is `xpathexpr` and is supported by HTTP.

The XPath string expression has a required field that contains the XPath expression and an optional field for defining local namespaces. If all the namespaces contained in the XPath expression are standard, the second field can be optional. If there are multiple local namespaces, separate each with a comma (,).

The following example shows an `xpathexpr` with one local namespace definition:

```
xpathexpr$/http:Envelope/soap:Body/m:getTimeZone/n:clientId$n\\=http://test2.classify.ws.ibm.com
= '\\1000\\' or operation IS NOT NULL
```

In the previous expression, the XPath expression is `/http:Envelope/soap:Body/n:getTimeZone/n:clientId`. It contains the local namespace of `n`. So, the second field of the `xpathexpr` is defined as `n\\=http://test.classify.ws.ibm.com`. The dollar sign (\$) is used to denote the start of a field definition. The double backslashes are the escape sequence in this example. The first equals sign (=) is escaped because it is part of the local namespace definition, whereas the second equals sign (=) is the operator in the classification expression and must not be escaped.

The following example shows an `xpathexpr` with two local namespaces:

```
xpathexpr$/http:Envelope/http:Body/m:getTimeZone/n:clientId$m\\=http://test.classify.ws.ibm.com,n\\
=http://test2.classify.ws.ibm.com = '\\1000\\' or operation IS NOT NULL
```

The following table shows the standard namespaces for the `xpathexpr` expression:

Table 21. Namespaces

Namespace	URL
soap	http://schemas.xmlsoap.org/soap/envelope/
soap-env	http://schemas.xmlsoap.org/soap/envelope/
soapenc	http://schemas.xmlsoap.org/soap/encoding/
soapbind	http://schemas.xmlsoap.org/wsdl/soap/
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
xsi	http://ws-i.org/schemas/conformanceClaim/
wsdl	http://schemas.xmlsoap.org/wsdl/

Operators

Intelligent Management supports the operators in the following table in the rules expressions. These operators are also referred to as *predicates* in SQL terminology because they appear inside of a WHERE or HAVING clause. Operators are case insensitive.

Table 22. Request classification operators

Operator	Description
OR	The logical OR operator.
AND	The logical AND operator.
NOT	The negation operator.
IN	<p>Expresses an operand with multiple values in a single expression. Its meaning is consistent with the SQL standard meaning of the operator.</p> <p>For example, if you want to express that the port value could be any or all of the values such as 9080, 9090, 9091, use the expression fragment:</p> <pre>port IN (9080,9090,9091)</pre> <p>In SQL, how the values inside the parenthesis are expressed depends on the data type of port. If the port is an integer, the values without the single quotation marks are syntactically correct. If the port is a string, the correct expression is:</p> <pre>port IN ('9080','9090','9091')</pre>
LIKE	<p>Expresses pattern matching for string operand values. The value must contain the wildcard character (%) in the position where the pattern matching is expected to start.</p> <p>For example, the expression:</p> <pre>host LIKE %blanca</pre> <p>matches the word blanca and any other word that ends in blanca, while the expression:</p> <pre>host LIKE blanca%</pre> <p>matches the word blanca and any other word that starts with blanca. The expression:</p> <pre>host LIKE %blanca%</pre> <p>matches the word blanca and any word that has the token blanca imbedded in it.</p> <p>The <code>java.util.regex.Pattern</code> class is used.</p>
=	The equality operator expresses a match in case-sensitive match.
>	Greater-than operator for use with numeric operands.
>=	Greater-than or equal operator for use with numeric operands.
<	Less-than operator for use with numeric operands.
<=	Less-than or equal operator for use with numeric operands.
<>	Not-equal operator.
BETWEEN	Used with AND to select a range of values inclusive of the first (low) value and the last (high) value. Together, they operate on numbers and dates values.
IS NULL	Tests for an operand having a NULL value.
IS NOT NULL	Tests for an operand having a value other than NULL.

Configuring service policies without response time goals

Configure service policies without response time goals when multiple service policies are configured and the response time goals cannot be accurately determined for any of the service policies, or when there might be occasional long-running or stuck requests.

Before you begin

- Enabling this feature enables it for all service policies. It is not possible to have some service policies with response time goals and others without response time goals.
- ARFM queueing cannot be disabled for this feature to work correctly. In other words, the `disableARFM.py` script cannot be used to set the `arfmManageCpu` cell custom property to false.

About this task

To aid in the ease of use of the product, performance management without response time goals is supported. Providing correct response time goals requires load testing to be performed to determine a reasonable value. This feature eliminates the need for performance testing. Instead, the product simply uses the importance of a service policy to manage performance.

To determine when a new dynamic cluster instance should be started when there are no response time goals, a utility value is calculated based upon the fraction of time that requests wait. The utility value is a value between -1 and +1 based on the importance of the service class. Requests with zero waiting time get a +1 utility value, and requests with almost all time waiting get a -1 utility value.

A utility value of zero is obtained if the fraction of time waiting equals the relative importance of the request service class, where relative importance is zero for very high importance and one for very low importance. Hence, the utility value is a function of the performance level achieved, which is measured as fraction of time waiting, and the importance of the request, which is measured as relative importance. For a given importance level, a better performance level means higher utility values. When the utility value is less than zero, the application placement controller attempts to start a new instance if capacity to do so can be found.

Procedure

1. In the administrative console, select **System administration > Cell > Custom properties > New**.
2. Specify the name of the custom property as `disableResponseTimeGoals`.
3. Specify the value of the property as `true`.

Defining service policy rules

Service policy rules define the service policy goal to use for a single piece of incoming work. This is done by associating a transaction class with a Boolean expression. Each transaction class is associated with exactly one service policy goal. The Boolean expression can be customized to match any specific piece of work.

Before you begin

Define a service policy to categorize and prioritize work requests. See the topic on defining a service policy for more information.

About this task

Optionally, you can add rules to your service policy. If the expression evaluates to true for a particular request, the policy associated with that rule is used. A work class contains a set of rules that is used to classify requests, such as an HTTP request or IIOp call. Complete the following steps to define a service policy rule. This is just one example of how to create service policy rules.

Procedure

1. Select **Applications > Application types > WebSphere enterprise applications > *application_name* > Service policies**. Click **OK**.
2. From the **Service policies** tab, expand the work request type and the work class for which you want to create a rule, and click **Add rule > Rule builder > Add**.
3. Select the type of rule, such as **Group ID**. This displays the rule builder panel. Continue to build a rule, specify the transaction class, or click **OK**.
4. Click the new rule to set its operators. A predefined set of operators displays for the type of rule condition that is selected.
5. Select the operator that you want to use and enter the appropriate information in the provided field. For example, you can classify incoming requests for the StockTradeWork work class by a group ID to use a different transaction class.
6. Select the **(=)** operator and type HTTP in the provided field, to provide a different transaction class for HTTP requests.

Rules support

Support for HTTP rules applies for both routing and service policy rules. They can be variously located and have a specific order of priority.

Overview

HTTP rules support applies for both routing and service policy rules. They can be located:

1. Inside work classes at the application level
2. Inside work classes at the ODR level
3. At the ODR level. There is no default rule supported for HTTP rules located here. The rules created take precedence over any rules at the application level.

The order of priority is

1. Custom HTTP rules at the ODR level is used to route the request.
2. If none of those rules match, custom HTTP rules located inside work classes at the application level is used to route the request.
3. If none of those rules match, the default HTTP rule located inside work classes at the application level is guaranteed to be used to route the request.

The default rule for HTTP service and routing policy rules, at the on demand router (ODR) level, is not supported.

Routing policy action types

You can create routing policies for HTTP and SOAP and SIP requests. The routing policy actions types vary for different locations of rules. The table in this topic shows the actions types for routing policies based on location and protocol.

Action types

Table 23. Action types for rules based on protocol

Rules	Location	Action type
HTTP	Inside work classes at the application level	<ul style="list-style-type: none"> • Permit routing to (permit) • Permit routing with affinity (permitsticky) • Redirect routing to (redirect) • Reject routing with return code (reject) • Permit routing to servers in maintenance mode (permitMM) • Permit routing with affinity to servers in maintenance mode (permitstickyMM)
SOAP	Inside work classes at the application level	<ul style="list-style-type: none"> • Permit routing to (permit) • Permit routing with affinity (permitsticky) • Redirect routing to (redirect) • Reject routing with return code (reject) • Permit routing to servers in maintenance mode (permitMM) • Permit routing with affinity to servers in maintenance mode (permitstickyMM)
HTTP	Inside work classes at the ODR level	<ul style="list-style-type: none"> • Permit routing to (permit) • Permit routing with affinity (permitsticky) • Redirect routing to (redirect) • Reject routing with return code (reject) • Permit routing to servers in maintenance mode (permitMM) • Permit routing with affinity to servers in maintenance mode (permitstickyMM)
HTTP	At the ODR level. There is no default rule supported for HTTP rules in this location. The rules created take precedence over any rules at the application level.	<ul style="list-style-type: none"> • Permit routing to (permit) • Permit routing with affinity (permitsticky) • Redirect routing to (redirect) • Permit routing to servers in maintenance mode (permitMM) • Permit routing with affinity to servers in maintenance mode (permitstickyMM)
SIP	At the ODR level.	<ul style="list-style-type: none"> • Permit routing to (permit) • Reject routing with return code (reject)

Selecting an action type

1. Selecting one of the following options:
 - **Permit routing to:** From the **Select edition name here** list select the edition name.
 - **Reject routing with return code:** From the **Select edition name here** list select the edition name and in the **Enter in return code** field, type the return code.
 - **Redirect routing to:** From the **Select edition name here** list select the edition name and in the **Enter URI to redirect to** field, type the URI.
 - **Permit routing with affinity to:** From the **Select edition name here** list select the edition name.
2. Click **Apply**.

Alternatively, you can apply new classification rules by clicking **Add Rule** and taking the following actions:

- If you know your rule name, type the new routing rule name, and select one of the following options from the **Then** list:
 - **Permit routing to:** From the **Select edition name here** list select the edition name.
 - **Reject routing with return code:** From the **Select edition name here** list select the edition name and in the **Enter in return code** field, type the return code.
 - **Redirect routing to:** From the **Select edition name here** list select the edition name and in the **Enter URI to redirect to** field, type the URI.
 - **Permit routing with affinity to:** From the **Select edition name here** list select the edition name.
- To build a new rule, click **Rule builder**, select the type of rule that you want to create from the rule **Condition** list, and click **Add**:
 1. Click the new rule to set its operators. A predefined set of operators displays for the type of rule condition selected.
 2. Select the operator that you want to use and enter the appropriate information in the field provided.

Overview of work classes

A work class is a grouping of work that is to be done by an application server. Work is an HTTP request, a SIP message, an IOP call, or a JMS message. Each work class contains a set of rules that is used by Intelligent Management to determine how to handle that work.

Work class overview

A work class is used to classify requests to apply a policy to the request. There are two main types of work classes:

1. Service policy work classes - Work class rules associate incoming work with a service policy, thus indicating to Intelligent Management when to forward the work to the application server.
2. Routing policy work classes - Work class rules associate incoming work with routing policy, thus indicating to Intelligent Management where to send the work. Each unit of work (request, message, or call) is associated with a single service policy work class. HTTP requests and SIP messages are also associated with a single routing work class. Routing work classes do not exist for IOP and JMS because these protocols do not flow through the on demand router (ODR); thus no routing policy is needed.

Creating a work class

A work class is applied to the patterns that are associated with it. Every pattern has an object (application module or on demand router) with which it is associated. These pattern and object pairs make up the definition of which requests the work class is mapped to. Custom work class defined patterns take precedent over the default work class patterns. Every default work class contains a `"/*` pattern, which means every request matches for that object.

Custom work class definitions are evaluated before default work class definitions when trying to find the matching definition. If an incoming request cannot be matched by any custom work class definitions, the default work class definition is used.

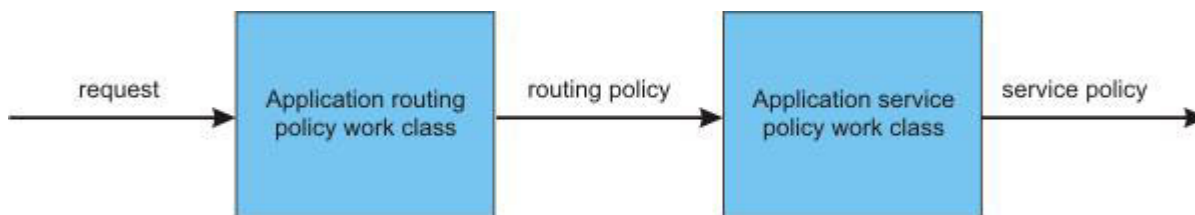
Work class types

There are different types of work classes for service and routing policies:

Table 24. Work class types

Work class	Description
Application routing policy	Specifies how to determine the routing policy for a request to an application that is installed in the Intelligent Management environment.
Application service policy	Specifies how to determine the service policy for a request to an application that is installed in the Intelligent Management environment.
Generic server cluster routing policy	Specifies how to determine the routing policy for a request to a generic server cluster.
Generic server cluster service policy	Specifies how to determine the service policy for a request to a generic server cluster.

The following diagram shows the flow of a request that is targeted for an application that is installed in the Intelligent Management environment. The request is applied to an application routing policy work class to determine the routing policy. If the resulting routing policy is `permit` or `permitsticky`, the request continues to an application service policy work class to determine the service policy and the transaction class name.



Each application contains default application routing and application service policy work classes. You can also create additional non-default work classes. Each work class has a default match action. The default routing policy for an application is `permit:application_name`. The default service policy is `Default_TC`, the default transaction class.

Work class types

You can use either default work classes that are created during a system application installation or define your own. Default work classes and directories for system applications are created during profile augmentation to support the high availability deployment manager. Default and new application work classes are defined on a per application edition basis.

Default application work classes

Each default work class has membership that is equivalent to a wildcard expression for all the work of that protocol type for that application. This work class is matched to last, with any new user defined work classes taking precedence. Default work classes cannot have their membership altered manually nor can they be deleted. They are meant to define how any work directed toward the application that does not get classified into any user-defined work classes into a service policy definition. While the membership cannot be deleted, classification rules can be defined on the default work class. This is especially useful if the

environment does not need to classify based on work class membership, but does need to classify on some advanced criteria such as group identification or host name.

The default matchAction on the default work classes for the application is to classify to the default transaction class of the default service policy. This can be changed to select an alternate transaction class/service policy pair.

z/OS Routing and service policies for work classes are not supported for IIOp or JMS for applications deployed to z/OS platforms. WebSphere Application Server z/OS provides IIOp and JMS service classification.

New application work classes

Each edition of the application has its own definitions on how it should be classified into service policies. After the on demand router (ODR) determines which application edition should be routed to, its service policy work class definitions are evaluated to determine how its work should be classified. When a new edition of an application is installed, you can choose an edition of the application to clone or choose none at all. If an edition is chosen, all of its work classes are cloned with the defaults appropriately renamed with the new application edition name. If no edition is chosen, only the defaults are created.

Configuration placement location

The configuration placement location for work classes for applications is:

```
<context>
  <context-name>applications</context-name>
  <child-context-names>
    <child-context-name>deployments</child-context-name>
    <child-context-name>workclasses</child-context-name>
  </child-context-names>
</context>
<context>
  <context-name>deployments</context-name>
  <child-context-names>
    <child-context-name>workclasses</child-context-name>
  </child-context-names>
</context>
<context>
  <context-name>workclasses</context-name>
  <root-document-type>WorkClass</root-document-type>
  <child-document-names>
    <child-document-name>WorkClass</child-document-name>
  </child-document-names>
</context>
```

Default system application work classes

Default work classes and directories for system applications, such as the adminconsole.ear, are created during profile augmentation to support the high availability deployment manager. An xd directory under the cell context mimics the systemApps structure and contains the default work classes. The default work classes are created under the following contexts:

```
cells/<cellName>/xd/systemApps/<earName>/workclasses/<workclass>/
```

```
cells/<cellName>/xd/systemApps/<earName>/xddeployments/<appName>/workclasses/<workclass>/
```

Intelligent Management listens for changes to the systemapps.xml file under the node context for any updates:

```
cells/<cellName>/nodes/<nodeName>/systemapps.xml
```

Middleware application work class location

The middleware application work classes location is:

```
<context>
  <context-name>middlewareapps</context-name>
  <child-context-names>
    <child-context-name>middlewareappeditions</child-context-name>
    <child-context-name>workclasses</child-context-name>
    <child-context-name>preferences</child-context-name>
  </child-context-names>
</context>
<context>
  <context-name>middlewareappeditions</context-name>
  <child-context-names>
    <child-context-name>workclasses</child-context-name>
  </child-context-names>
</context>
```

Intelligent Management listens for changes to the `systemapps.xml` file under the node context for any updates:

```
cells/<cellName>/nodes/<nodeName>/systemapps.xml
```

Request classification operators

Intelligent Management supports operators in the rules expressions. These operators are also referred to as *predicates* in Structured Query Language (SQL) terminology because they appear inside of a WHERE or HAVING clause. Operators are case insensitive.

Operators

Table 25. Request classification operators

Operator	Description
OR	The logical OR operator.
AND	The logical AND operator.
NOT	The negation operator.
IN	<p>Expresses an operand with multiple values in a single expression. Its meaning is consistent with the SQL standard meaning of the operator.</p> <p>For example, if you want to express that the port value could be any or all of the values such as 9080, 9090, 9091, use the expression fragment:</p> <pre>port IN (9080,9090,9091)</pre> <p>In SQL, how the values inside the parenthesis are expressed depends on the data type of port. If the port is an integer, the values without the single quotation marks are syntactically correct. If the port is a string, the correct expression is:</p> <pre>port IN ('9080','9090','9091')</pre>

Table 25. Request classification operators (continued)

Operator	Description
LIKE	<p>Expresses pattern matching for string operand values. The value must contain the wildcard character (%) in the position where the pattern matching is expected to start.</p> <p>For example, the expression: <code>host LIKE %blanca</code></p> <p>matches the word <code>blanca</code> and any other word that ends in <code>blanca</code>, while the expression: <code>host LIKE blanca%</code></p> <p>matches the word <code>blanca</code> and any other word that starts with <code>blanca</code>. The expression: <code>host LIKE %blanca%</code></p> <p>matches the word <code>blanca</code> and any word that has the token <code>blanca</code> imbedded in it.</p> <p>From a code implementation viewpoint, the <code>java.util.regex.Pattern</code> class is used.</p>
=	The equality operator expresses a match in a case-sensitive fashion.
>	Greater-than operator for use with numeric operands.
>=	Greater-than or equal operator for use with numeric operands.
<	Less-than operator for use with numeric operands.
<=	Less-than or equal operator for use with numeric operands.
<>	Not equal operator.
BETWEEN	Used with AND to select a range of values inclusive of the first (low) value and the last (high) value. Together, they operate on numbers and dates values.
IS NULL	Tests for an operand having a NULL value.
IS NOT NULL	Tests for an operand having a value other than NULL.

Rule-based request classification

With any rule-based technology, rule-based processing involves three basic areas: the vocabulary that forms the language, the grammar for expressing the vocabulary in statements, and the rule-processing engine.

Vocabulary

Vocabulary consists of the operators, variable keywords known as operands, and control flow statements. The language is the Java Message Service (JMS 1.1), and the Message Selector Syntax. The message selector is a string whose syntax is based on a subset of the SQL92 conditional expression. The general statement syntax is:

```
operand operator literalExpression [| compoundOperator expression] .
```

In the following example,

```
serverhost like '%ibm.com' or clienthost = 'myhost.raleigh.ibm.com'
```

serverhost and clienthost are operands, like and = are operators, or is the compound operator, and '%ibm.com' and 'myhost.raleigh.ibm.com' are the literalExpression. The [] indicates that the expression enclosed within is optional, and the | indicates that after the operator, a literalExpression or a compound operator can be followed by more expressions.

The result of an expression is an action that is taken. From a grammatical viewpoint, these actions are literals that are provided by a policy provider. Two types of policies are supported:

1. Routing
2. Service

The actions that are taken are dictated by the policy provider. For routing, the actions are:

- permit
- reject
- permitsticky
- redirect

Each of the actions has the appropriate target; the recipient of an action. If the result of evaluating an expression is to take the action of permit, the target of that action is the application for which routing is permitted. For service policies, the target is encapsulated in the action, and the action is a transaction class.

A complete statement consists of the rule expression, and the action to be taken is represented differently depending on the input source. In the administrative console, the actions are separated into forms and fields that are easily selectable. If you are using scripting, the complete statement might look as follows:

```
expression<delimiter>action  
For example, clienthost='localhost' and serverhost like  
'%.ibm.com'?permit?DefaultApplication.ear
```

From an implementation viewpoint, work classes, which are XML documents, are used to capture the rules expression, matched actions, and other implementation artifacts. Therefore, a work class is an XML document that contains zero or more matchRules elements, and one or more workClassModules elements.

Operators

Intelligent Management supports the operators in the rules expressions. In general, you might not know the true data type of a given operand. If you use the Hypertext Transfer Protocol (HTTP), every operand is treated as a data type string and use the operator as an indicator for the real data type of the operand for data validation purposes. An example of an operator which tests for an operand that has a null value is: IS NULL.

Operands

When new protocols are added and new sets of operands defined, operands are valid within protocol scopes. If an operand is specified in a scope for which it is not valid, an error condition is indicated. In this release, supported protocols are:

- HTTP
- **Distributed operating systems** JMS
- **z/OS** SOAP over HTTP represented simply as SOAP
- **Distributed operating systems** Internet Inter-ORB Protocol (IIOP)

An example of an operand for HTTP requests is: MIMEType.

Deploying and managing application editions without loss of service

The application edition manager controls interruption-free production application deployments. Interruption-free deployment prevents loss of service when you install an application update in your environment.

Before you begin

If you are a user with either a monitor or an operator role, you can only view the edition manager information. If you have the role of configurator or administrator, you have all the configuration privileges for the application edition manager.

About this task

Deploy and use application editions when you install or update applications. You can perform an upgrade while maintaining continuous application availability.

Procedure

- Install an application edition.
- Activate an application edition.
- Update your routing rules.
- Validate a new application edition before activation.
- Roll out an edition.
- Roll back an edition.
- Delete an edition.
- Troubleshoot.

Application edition manager concepts

By knowing the difference between application edition manager versions and editions, the methods of deploying and upgrading your application, and edition validation and compatibility, you can fully use the application edition manager to manage your application deployments.

Unmanaged web applications can be defined with an edition. However, a rollout cannot be performed on them nor can they be placed into validation mode. Unmanaged web applications are supported with the exception that not all capability is available due to their nature as applications of assisted lifecycle management.

Versions and editions

The terms *version* and *edition* distinguish between what occurs in your development and build environment from what occurs in your deployment and operational environment. A version is a successive generation of an interface, function, implementation, or entire application, and so forth. Version is a development and build concept. An edition is a successive deployment generation, for example, the deployment of a particular set of versioned artifacts. Edition is a deployment and operational concept.

An *application edition* is a unique deployment of a particular application. In the WebSphere Application Server administrative environment, an application edition is an application that is uniquely identified by the combination of an application name and an edition name. Multiple editions of the same application have the same application name, but different edition names. The edition name can be numeric, such as 1.0 or 2.0, or the name can be descriptive, such as *first edition* or *blue edition*.

- *Base edition* refers to a deployed application that has no associated edition information.

- *Edition activation* distinguishes between two states in which an application edition might exist. When an edition is first installed, the edition is in the *inactive* state. You can start the edition only when it is in the *active* state. The transition from inactive to active is known as *activation*.
- *Concurrent activation* exists when multiple editions of the same application are active and started simultaneously. Concurrently active editions can provide one set of users access to one edition and other users with access to another. For example, if you introduce a new edition of an application, you might want a select group of users to test the edition, and not want all users to have access to the edition. With concurrent activation, you must establish a *routing policy* to distinguish which users have access to an edition. A routing policy is stored as part of the configuration metadata for an application. Also, a routing policy prevents ambiguity and determines which edition receives control. The following example is a diagram of concurrently active editions.

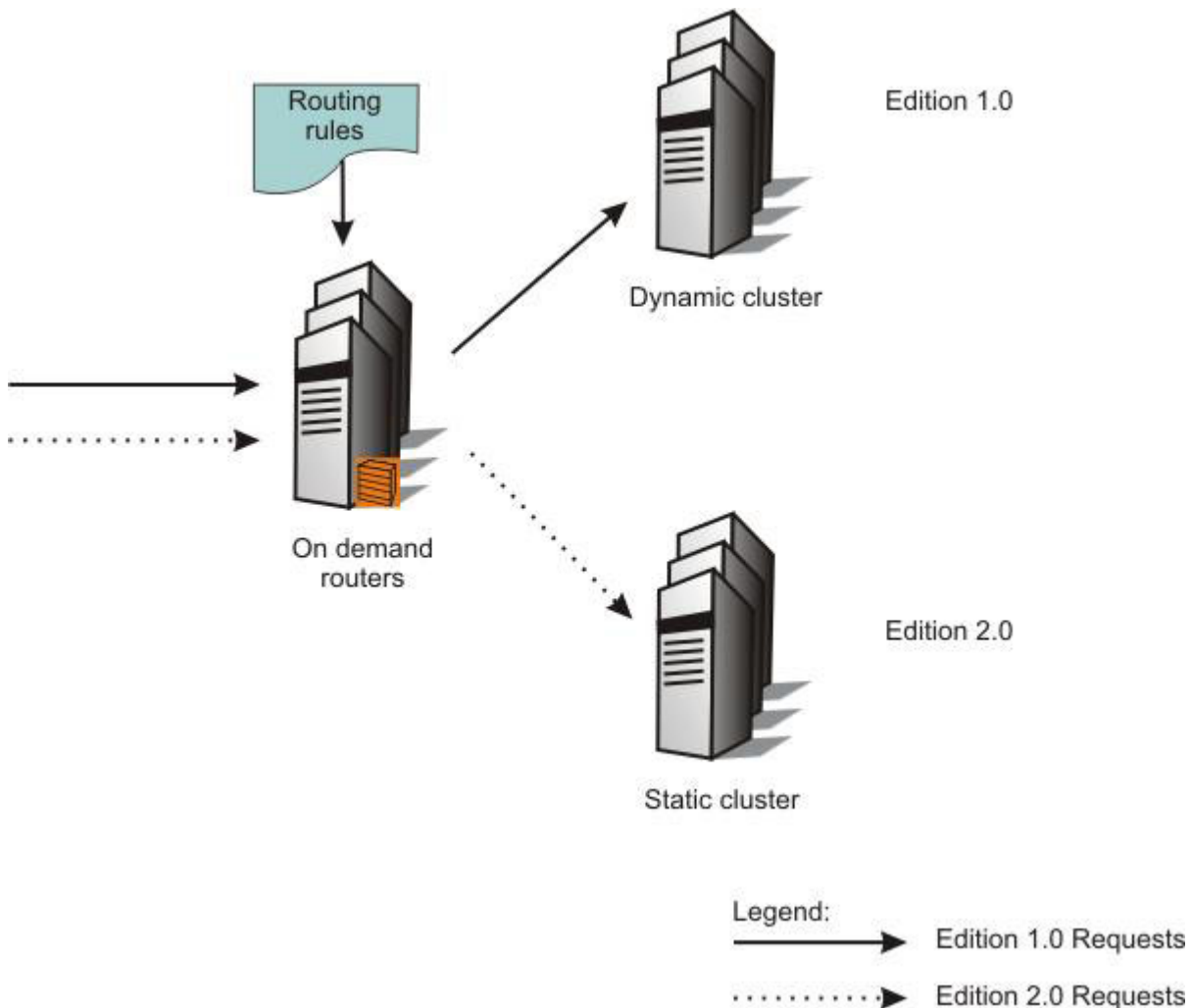


Figure 15. Concurrently active editions

Application upgrade and deployment

Many business applications require constant availability. The standard for application availability asserts that applications are deployed on application server clusters. The redundancy of a cluster is essential to provide continuous availability. *Interruption-free application upgrade* refers to the ability to upgrade while maintaining application continuous availability. In other words, users of the application experience no loss of service during the application upgrade.

When you perform a *rollout* to an edition, you replace an active edition with a new edition. To provide interruption-free application upgrades, performing a rollout to an edition includes the following items:

- Fencing a server from receiving new requests.
- Quiescing requests for the application in a particular server.
- Stopping the currently active edition.
- Starting the new edition.
- Resuming the flow of requests to the edition.

When you perform a rollout to an edition across an application server cluster, you complete the following activities across the set of the servers that are in that cluster:

To perform a rollout to a target cluster, you can divide the cluster into groups, and define a group size, which specifies the number of nodes to process at a time. Performing a rollout to a group results in the servers in each group being upgraded to the new edition at the same time. Each server in the group is quiesced, stopped, and reset. A rollout can be performed on only one group at a time in the administrative console. When any member in the new edition becomes available, all requests are routed to the new edition.

As you perform a rollout to the edition, some servers in the cluster move from the previous edition to the new edition, some servers are in the process of making the transition, and other servers have not started the transition. All application requests are sent to any server that has an active, running instance of the latest edition of the requested application. For example, when you perform a rollout from edition 1.0 to 2.0, all application requests are served by edition 2.0 when edition 2.0 becomes available on a server. Any servers that are still running edition 1.0 do not serve requests until this server is updated to edition 2.0. The following example is a diagram of a group rollout.

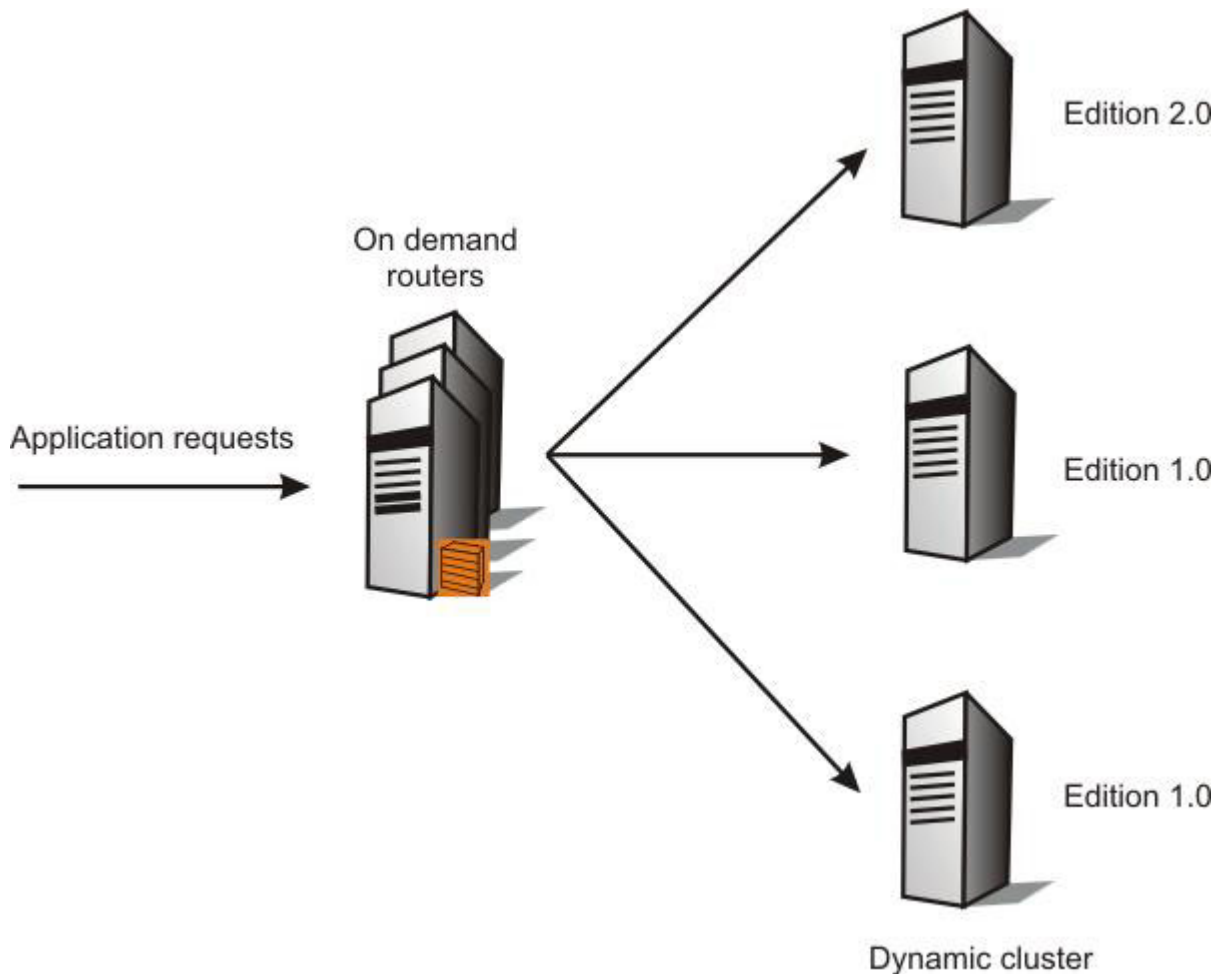


Figure 16. Group rollout

Performing an *atomic rollout* to an edition replaces an edition on half of the cluster at a time to serve all user requests with a consistent edition of the application. All user requests are served by either the previous or the new edition; user requests are never served by both editions.

An atomic rollout ensures that all application requests are served by a consistent edition, for example, either edition 1.0 or 2.0, but not by both. The currently available edition is taken offline from half of the servers that comprise the cluster. In those servers, the new edition is activated and started, but those servers are held offline until the next step completes. The next step is to take the currently active edition offline in the remaining servers. At this point, no server has an instance of either edition available to serve application requests. The ODR temporarily queues any request that arrives for this application. After the application is fully offline, the first half of the cluster is brought back online. The second half of the cluster transitions from the previous edition to the next edition and is brought back online. The following example is a diagram of an atomic rollout:

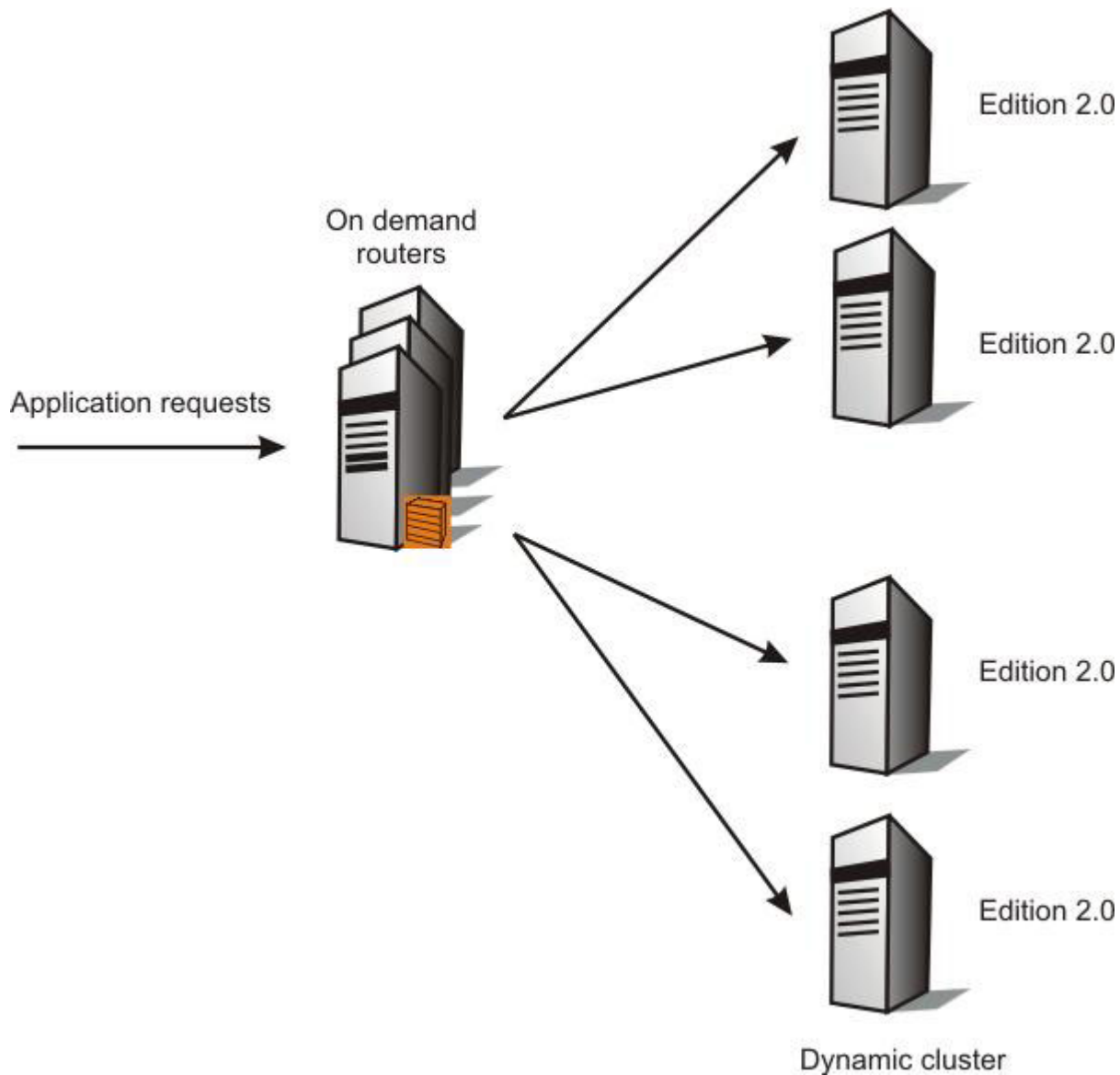


Figure 17. Atomic rollout

Edition validation and compatibility

The *Edition validation* refers to a special case of concurrent activation, where an assigned deployment target of the edition, for example, a dynamic cluster, is cloned and the edition is made ready to be started on the cloned deployment target. The cloned deployment target is known as a *validation target*. Routing rules must be used to designate which application requests are sent to the edition undergoing validation. When an edition is in validation, it is in *validation mode*.

Validation mode ensures that a new edition of an application functions in its production environment without taking the currently available edition offline. Typically, a test load is sent to an edition in validation mode to confirm that aspects of the application environment and setup, such as connectivity and database access, work as expected. When you perform a rollout to an edition validation mode, the operation occurs on the deployment target on which the edition was originally installed. Performing a rollout causes the edition to exit validation mode. When validation completes, the validation target is deleted. The following example is a diagram of edition validation.

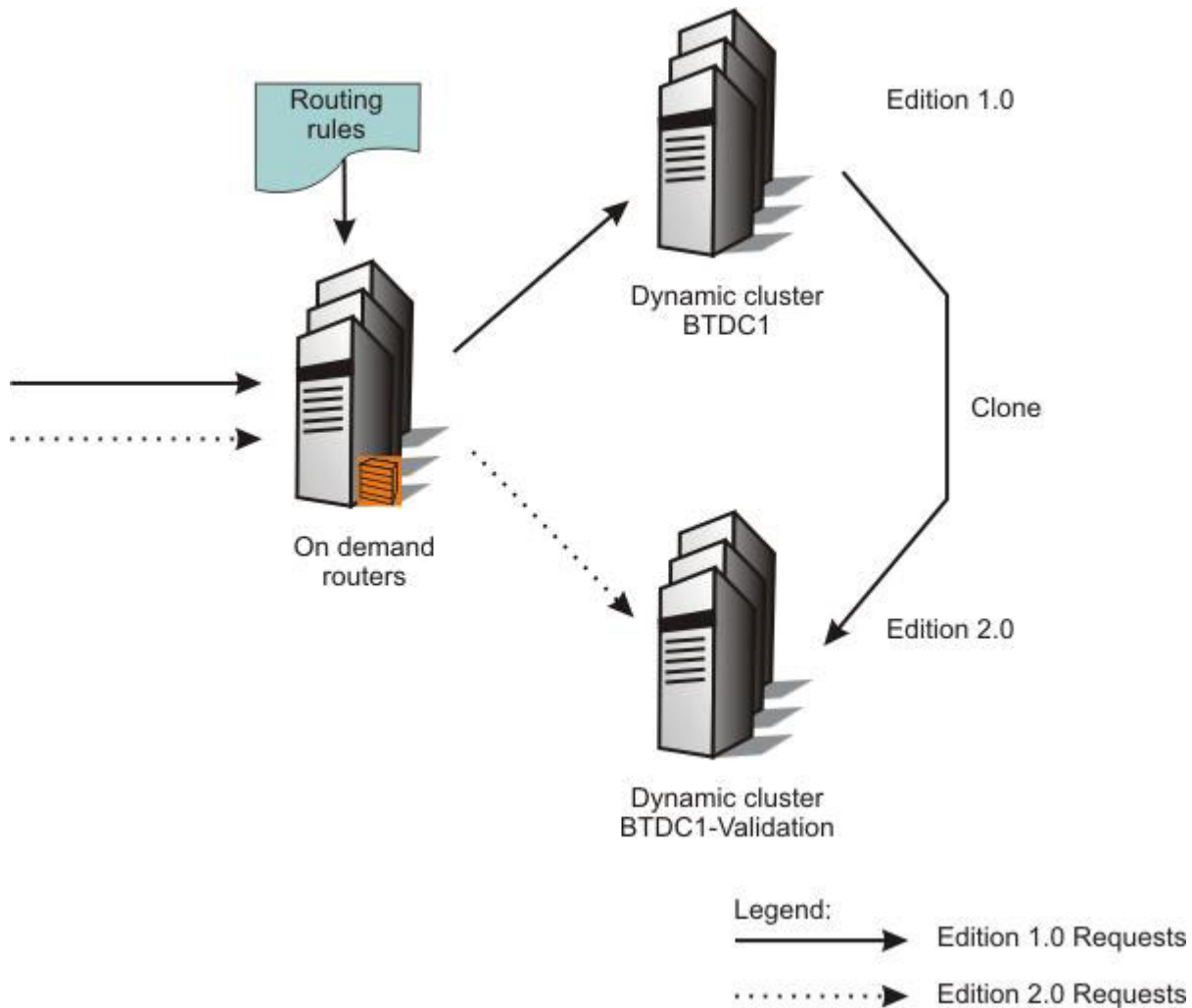


Figure 18. Edition validation

Some application changes are transparent, while other application changes are seen by the end user. When an application upgrade delivers at least the same application programming interfaces as the last change did and no semantic change to essential behavior, that application edition is an upgrade that is *compatible* with previous versions. As an existing user, you can use the upgraded application without changing how you use it and without recognizing a difference between the current and former editions.

An application upgrade that requires existing users to change how they use the application is an *incompatible* upgrade. You might need to drop former function, or change interfaces, for example, to improve maintainability or other factors, and might introduce incompatible changes to your deployment environment. Incompatible changes require careful planning to manage the impact to existing users.

Application edition manager

The application edition manager ensures interruption-free production application deployments. Interruption-free deployment prevents loss of service when you install an application update in your environment.

The application edition manager provides an application versioning model that supports multiple deployments of the same application in the Intelligent Management cell. Each deployment has a unique

edition name. The application edition manager allows you to select the edition to activate on an Intelligent Management cluster, so that you can perform a rollout of an application update or revert to a previous level.

The application edition manager is fully integrated with Intelligent Management, interacting with the on demand router (ODR), dynamic workload balancing, and the application placement manager. This integration ensures predictable application behavior when you apply application updates, and a smooth transition from one application edition to another while the system continues to manage your application performance goals. You can access application update processes with the administrative console, including edition activation across the application servers. Scripting application programming interfaces enable the integration of edition management functions with automated application deployment.

Application editions are supported for Java Platform, Enterprise Edition (Java EE) applications, including enterprise archive (EAR) files, and batch applications that conform to one of the batch programming models. The following table summarizes the supported features, which vary depending on the type of application that you are using.

Table 26. Application edition manager support for applications

Application type	Multiple editions of applications	Application rollout	Interruption-free application update, driven by the ODR and HTTP communication	Validation mode	Concurrent activation
Managed Web applications, including PHP Hypertext Preprocessor (PHP) applications	Supported	Supported	Supported	Supported	Supported
IIOP, EJB, or JMS applications	Supported	Supported	Not supported ¹	Not supported	Not supported
Unmanaged Web applications	Supported	Not supported	Not supported	Not supported	Supported
WebSphere Application Server Community Edition applications	Supported	Supported	Supported	Supported	Supported
Session Initiation Protocol (SIP) applications	Supported	Supported	Supported ²	Supported	Supported
Converged HTTP-SIP applications	Supported	Supported	Supported ²	Supported	Supported

¹ The ODR does not support communication with IIOP. Any EJB, JMS, or IIOP components called directly by an external client cannot utilize this feature.

² This feature is driven by the ODR and HTTP/SIP communication.

WebSphere Application Server includes an administrative function called *rollout update*. Rollout update provides a basic application upgrade, but is not interruption-free. The application edition manager is the preferred way to upgrade applications.

The application edition manager supports your overall application life cycle, and enables application updates and seamless, interruption-free application deployments to your production environment.

Application edition manager states

Application edition manager editions transition into multiple states. The transition of an application between the running and stopped runtime states occurs through start and stop operations. The transition of an application between the non-existent or existent configuration states occurs when you install or uninstall the application. As the application edition transitions from one state to another, various actions occur, such as installing, validating, activating, performing a rollout, deactivating, or uninstalling. The following table features a list of the transitions and the resulting actions:

Table 27. State transitions

From state	To state	Transition action	Notes
Nonexistent	Inactive	Install application edition	None
Inactive	Validation	Validate ¹	Validation deployment target is created.
Inactive	Active	Activate or perform a rollout ¹	Performing a rollout deactivates the previous edition.
Validation	Active	Perform a rollout ¹	Validation deployment target is deleted.
Active	Inactive	Deactivate ¹	None
Active	Nonexistent	Uninstall application edition	None
Validation	Inactive	Cancel validation	Validation deployment target is deleted.
Inactive	Nonexistent	Uninstall application edition	None

¹Validate, activate, perform a rollout, and deactivate are new application edition operations introduced by the application edition manager.

Operational environment

The operational environment in which the application edition manager runs is an Intelligent Management cell.

In the following illustration, the application edition manager manages applications that are deployed to dynamic clusters that receive work requests through an on demand router (ODR).

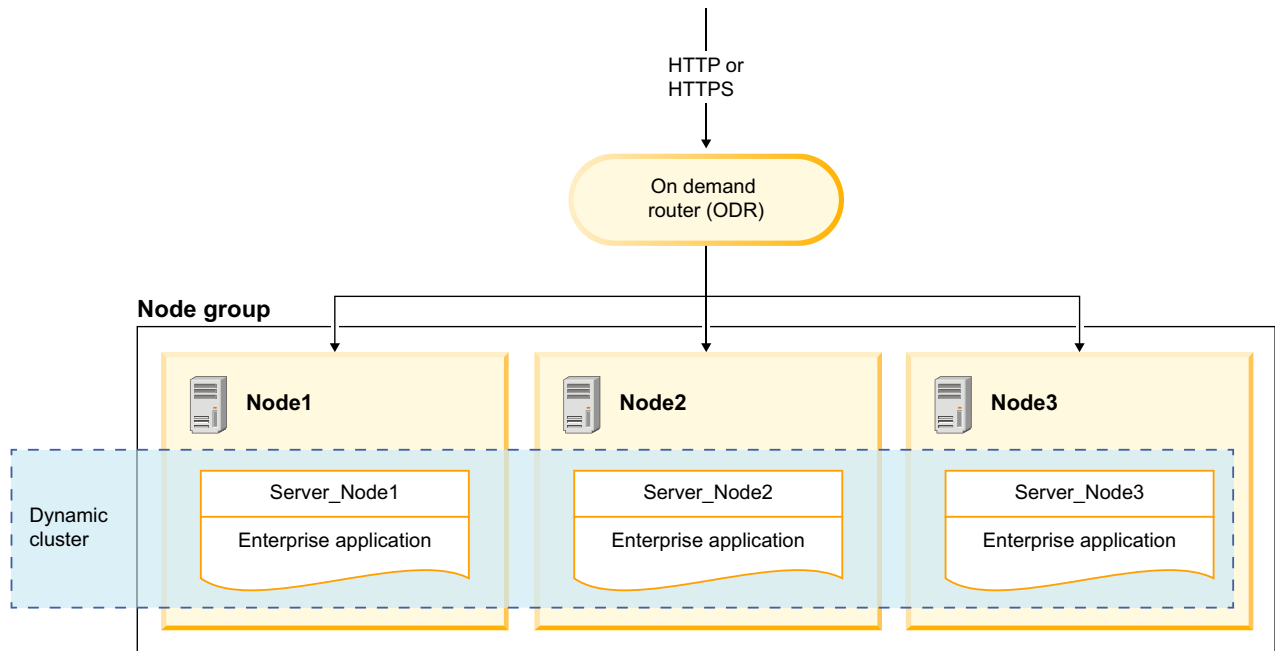


Figure 19. Application edition manager with an ODR and dynamic cluster

The SIP protocol is supported in Intelligent Management.

The application edition manager provides support for interruption-free application upgrades only for applications accessed through the ODR by way of the hypertext transfer protocol (HTTP) or hypertext transfer protocol running under the secure sockets layer (HTTPS). Service continuity during application upgrade is not assured for inter application access, for example, one Java Platform, Enterprise Edition (Java EE) application calling another, unless the inter application access is accomplished by way of HTTP or HTTPS through another ODR layer.

Edition compatibility

The application edition manager supports only compatible application upgrades for edition roll out, which means that interruption-free upgrade is only for editions that are compatible with previous editions. When you deploy editions with incompatible changes, the editions require the concurrent activation pattern and use of routing rules to separate request traffic between users of the former edition and users of the current edition. With concurrent activation, you can host multiple editions of the same application simultaneously with each edition supporting a distinct, non-intersecting set of users. However, concurrent activation might not provide interruption-free upgrade.

Consider the following compatibility issues when you deploy editions:

- **Application interfaces or semantics:** Active users who are currently using the application might be affected if application interfaces or semantics are changed between editions when a roll out is attempted. Examples of changes include those to existing interfaces, including modification or removal of an existing interface. Also, a change to the semantic behavior of an interface might also affect active users. For example, if an interface formerly allowed a parameter to be null, then the change requires that the same parameter is non-null. Changes that impact existing clients are not considered to be backward compatible and cannot be a part of an interruption-free upgrade. If impact to existing clients is not an issue, then use the WebSphere Application Server roll out update.
- **Hypertext transfer protocol (HTTP) session state:** If the HTTP session state is persisted or replicated, then application changes that add or change the data types that are stored in session also represent incompatible change. The current edition might be unable to use the session state created by a former edition.

- **Web content caching:** If a new application edition includes changed static web content and you are using the on demand router (ODR) to cache content, then you might need to flush the cache as part of the edition roll out. To perform this task manually, read about the cache monitor.

Installing an application edition

Installing an application edition is similar to installing an application. When installing a new edition of an application, the edition must be specified.

Before you begin

- Before you install your application edition, create a deployment target for your application.
- If you are a user with either a monitor or an operator role, then you can view the application edition manager information only. If you have the role of configurator or administrator, then you have all the configuration privileges for the application edition manager.

About this task

Multiple versions of the same shared library are installed if you give them different names during creation. Use a naming scheme that adds a version number to the shared library name, for example, Production Library 1.0 and Production Library 2.0. Each edition of an application is updated to use a distinct version of the shared library by binding the edition to the appropriate library. You can also clone an existing edition.

Procedure

1. Install the application. In the administrative console, select **Applications > Install new application** or **Applications > Install new middleware application**. Enter the following information in the wizard panels.
 - a. Specify the application type.
 - b. Specify the EAR, WAR, JAR, or SAR module to upload and install.
 - c. Optional: **For enterprise applications:** If you want to clone work classes from another edition of the application, select **Show me all installation options and parameters**.

2. In the **Application edition** field, specify your edition information. For example, type 1.0. The edition identifier does not have to be numeric and can be any combination of letters, numbers, and certain special characters, such as ~!@#%\$. -.

Session Initiation Protocol (SIP), Java EE applications: The edition identifier becomes part of a directory name under which the application is installed. Any character that you use in a file name on your system can also be used in the edition identifier.

PHP Hypertext Preprocessor (PHP), WebSphere Application Server Community Edition, unmanaged web applications: The edition identifier becomes a configuration attribute in the repository and accepts any characters.

Note: Ensure you enter a value for the edition; otherwise, a new edition of an application cannot be created.

3. Specify the installation options. In the **Application description** field, specify the type of edition you are installing. For example, type First edition.
4. Specify the deployment targets for the application.
 - Enterprise applications, Session Initiation Protocol (SIP) applications: If you are installing an enterprise application, you can use split deployment. With split deployment, you deploy a module in a single Java EE application archive and divide the module across multiple deployment targets. For example, you might deploy an enterprise archive (EAR) file that contains a web application module and the enterprise beans module is installed in the Intelligent Management environment. As a result, the web application module is installed on a server and the enterprise beans module is deployed on a cluster.

- Unmanaged Web applications: Define the deployment properties of the application. Specify module, context root, and virtual host information for the application. Choose the servers and clusters on which the application is deployed by selecting deployment targets.
 - PHP, WebSphere Application Server Community Edition applications: Choose the servers and clusters on which the application runs by selecting the deployment targets. Choose PHP servers or PHP dynamic clusters for PHP applications. Choose WebSphere Application Server Community Edition servers or dynamic clusters for WebSphere Application Server Community Edition applications.
5. Optional: If you are installing an enterprise application: To save time and reuse work classes, you can choose an existing edition of the application to clone. On the first administrative console panel of the application installation, you must select the **Show me all installation options and parameters** option to see the clone work classes option. From the **Clone existing work classes from this application edition** list, select the work class, and click **Next**. The work class establishes default routing rules for the application edition. The work classes of an application constitute the routing policy of that application. If you select an existing application edition, all of its work classes are cloned with the defaults appropriately renamed with the new application edition name. If you do not select an application edition, the defaults are created.
 6. Specify the location of the virtual host for the web modules, and edit the context root that is defined in the deployment descriptor file.
 7. Save and synchronize your nodes.
 8. Start the application.
 - Enterprise applications: In the administrative console, select **Applications > Enterprise applications > application_name > Start**.
 - PHP applications: In the administrative console, select **All applications**. Select the PHP application that you want to start. Choose the **Start** action, and click **Submit**. This action starts all of the PHP servers that are associated with this application. To start the servers individually, select **Servers > Other middleware servers > PHP servers**.
 - Unmanaged web applications: The application is displayed as started when the server on which the application is deployed starts.
 - **WebSphere Application Server Community Edition applications:** Start application editions individually. In the administrative console, select **Applications > Edition control center > application_name > Activate**.

What to do next

Now that your application edition is installed, you can perform a rollout, backout, concurrent activation, or validation.

Activating an edition

Edition activation distinguishes between two states in which an application edition might exist. When an application edition is first installed, the application edition is in the inactive state. You can start the application edition only when it is in the active state. The transition from inactive to active is known as activation. Three methods exist to activate an application edition: activating, performing a rollout to a new application edition, or validating the application edition. as activation.

Before you begin

- You must have an application installed and in the inactive state.
- To activate an application edition, you must have a configurator or administrator administrative role.

Note: The first installed edition of an application is automatically activated.

Procedure

1. In the administrative console, select **Applications > Edition control center**.
2. Select the application edition that you want to activate.
3. Complete one of the following actions.
 - a. Click **Activate**. Activating marks an application edition as available to be started. After you activate an application edition, you must update the routing rules, and click **Apply** and **Save** regardless of what item is displayed in the list. For more information about activating editions, read about [Creating routing policies for application editions](#).

Note: After you update the routing rules, you must click **Apply** and **Save** even if only one edition of the application is displayed in the menu list.

- b. Click **Roll out**. Performing a rollout to an application edition activates one edition in place of another. The new application edition automatically starts because it replaces a running application edition. For more information, read about [performing a rollout on an edition](#).
- c. Click **Validate**. Validation activates an application edition on a clone of its original deployment target. For more information, read about [validating an edition](#).

Results

The application edition is activated.

What to do next

If you used the activate or validate options, you can start your edition application.

Activating concurrent application editions

Activate multiple editions of the same application concurrently when you are validating before production, piloting an application to a select group of users, or rolling out a branch when an application upgrade requires a corresponding change on identifiable branches of client machines.

Before you begin

- You must have at least two editions of the same application installed. For example, the `my_application` application edition 1.0 is installed on the `dynamic_cluster_1` dynamic cluster, and application edition 2.0 is installed on the `dynamic_cluster_2` dynamic cluster.
- Privileges for the application edition manager differ, depending on the various roles. Roles include monitor, operator, configurator, and administrator. If you are a user with either a monitor or an operator role, you can only view the application edition manager information. If you have the role of configurator or administrator, you have all the configuration privileges for the application edition manager.

About this task

Each application edition must be active on a separate deployment target. When multiple editions of the same application are concurrently available to users in the same environment, the on demand router (ODR) cannot differentiate between the active editions without some information available to process the request and route it to the intended edition. You can use routing rules or unique interfaces for each application edition to prevent ambiguity.

Restriction: Only one active edition of a PHP Hypertext Preprocessor (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.

Procedure

1. Activate the application editions. Click **Applications > Edition control center > *application_name***. Select the inactive edition, and click **Activate**. For example, select the `my_application` application and activate application edition 2.0.
2. Create routing policies for each application edition. See “Creating routing policies for application editions” for more information.
3. Verify that the ODR is running. Click **Servers > On demand routers**. To route requests, the status must be **Started**.
4. Test the concurrent access to application editions. Select the two application editions by selecting the servers associated with the two dynamic clusters, and click **Start**.

Results

Edition 1.0 is serviced by the routing rule that you create for that particular edition, and edition 2.0 is serviced by its specific routing rule.

Example

To run a preproduction test of an application edition in the production environment with a selected set of users, you can clone the deployment target, including its resource and security definitions, and activate the target edition on the cloned environment. Use routing rules to direct the ODR to divert a selected subset of users to the application edition.

Additionally, to pilot your application, you can use routing rules to separate the pilot users on edition 2.0 from the general users on edition 1.0.

In the case of branch rollout, use routing rules to direct each branch to the appropriate edition. As the client code at each successive branch updates, the server-side routing rules can be updated to qualify the clients from the newly updated branch to be sent to the appropriate edition.

For cases where the routing rules are insufficient to differentiate user requests or where the user prefers an alternative to routing rules, each edition can be given its own unique URI and Enterprise JavaBeans (EJB) Java Naming and Directory Interface (JNDI) name. Unlike routing rules, unique interfaces for each edition are exposed to the application users. Therefore, you must choose the appropriate name to drive the appropriate edition.

What to do next

Perform validation to test the availability and resiliency of your new edition under realistic conditions. For more information, read about creating routing policies for application editions.

Creating routing policies for application editions

Update your routing rules after you activate an application edition to ensure that the on demand router (ODR) can route requests to the appropriate application edition.

Before you begin

Activate your application edition.

About this task

A routing policy is stored as part of the configuration metadata for an application. With a routing policy, you can express rules that instruct an on demand router (ODR) to send particular application requests to one

edition or another based on a set of criteria. You can use various criteria that specify which requests are sent to a particular application edition. Use this process to send requests from certain users to one edition and requests from other users to another edition.

Note: If your application has an existing application edition with a defined multi-cluster routing policy and you install a new edition, you must create a new multi-cluster routing policy for that specific edition. Use the administrative tasks to add multi-cluster routing policies for your new editions. For information about routing administrative tasks, read about rules for ODR routing policy administrative tasks.

Procedure

1. Navigate to the routing policies for the application. Click **Applications > Enterprise applications > *application_name*** or **Applications > All applications > *application_name***. Click the **Routing policies** tab. For example, select the `my_application` application.
2. Expand **Work classes for HTTP requests**. Because no routing rules are specified, all requests are routed to the edition that is displayed on this page. For example, all requests are routed to application edition, `my_application-edition2.0`.
3. Click **Rule builder**.
4. From the rule list, select a rule. For example, select **Client host (clienthost)**, and click **Add**.
5. Select criteria for the rule. For example, select an operator of **Equals (=)** and type a value of your client host name. Click **OK**.
6. Click **OK** again.
7. Expand **Work classes for HTTP requests**.
8. Set the action associated with the new rule. For example, requests from the host route to edition, `my_application-edition1.0`. Select the corresponding action from the **Then** list, and click **Apply** to save the rule.
9. From the **Routing policies** tab, click **Apply**.
10. Save the changes to the configuration repository and synchronize the nodes.

What to do next

If you are routing to multiple editions of the same application, you can validate your new application edition before moving to the new edition. For more information, read about validating an edition.

Validating an edition

Validating an edition is the process of determining if a new edition is available and ready to move into production and replace the current edition. You can install and validate a new edition under realistic conditions while your production application edition continues to serve requests.

Before you begin

- Ensure that all of the modules for your application are deployed to the same deployment targets.
- Define unique routing rules for edition 2.0. Routing rules enable editions to run concurrently and hypertext transfer protocol (HTTP) requests for the validation edition to correctly route to the validation target without interfering with edition 1.0. For this scenario, use the `my_application` application. Install both application editions, 1.0 and 2.0, on the `dynamic_cluster_1` dynamic cluster. For more information about routing rules, read about creating routing policies for application editions.
- To set the operational mode of the cloned validation cluster to a different mode from the production cluster, create the `VALIDATION_OPERATIONALMODE` custom property in the administrative console. Otherwise, the validation cluster is set to the same operational mode as the production cluster after it is created. Set the value to `automatic`, `manual`, or `supervised`. If you specify any other value or you do not specify a value, the validation dynamic cluster is set to manual mode.

Restriction: Only two cluster members can be used or created in validation mode. You can map routing and service policies to the validation mode application, but no more than two cluster members are started to maintain the work. You can overwrite this setting after the validation cluster is created by changing the minimum and maximum number of dynamic cluster instances.

- If you are a user with either a monitor or an operator role, then you can only view the application edition manager information. If you have the role of configurator or administrator, then you have all the configuration privileges for the application edition manager.

gotcha: Restart the browser before you validate an edition. By restarting the browser, you ensure that previous sessions expired, and that the requests are routed to the application under validation.

About this task

Consider the following scenario as an example of how validation is performed on an edition: Edition 1.0 of an application is installed, active, and running on a dynamic cluster. Edition 2.0 is the candidate validation edition and is installed on the same deployment target in the inactive state. Validating edition 2.0 clones the edition 2.0 deployment target. For example, the validation might create a new dynamic cluster, such as the DC-Validation dynamic cluster, and map edition 2.0 to this new cluster. The cloned cluster uses the existing cluster members as the server template for the creation of the cloned servers.

After the validation clone target is created, edition 2.0 is activated, and the routing rules are defined, you can start, stop, and reconfigure the edition.

Procedure

1. Click **Applications > Edition control center** to verify that the application has two installed editions, with only one active edition.
2. Optional: If you want to create a validation cluster that has a different operational mode than your production cluster, you can define the `VALIDATION_OPERATIONALMODE` custom property on the production cluster. Add the validation cluster to the service integration bus (SIB). If you do not define this custom property, your validation cluster has the same operational mode as the production cluster.
3. Update the Enterprise JavaBeans (EJB) reference bindings to specify the new cluster name. Before rolling out the application from the validation cluster, the bindings must be changed back to the original value.
4. Click the `my_application` application.
5. Select edition 2.0 and click **Validate**. The validation status page shows each step of validating the `dynamic_cluster_1` dynamic cluster and deploying edition 2.0 to the cloned cluster. The application edition control center shows that one of the editions is in validation mode, and the manage editions page shows that the edition 2.0 target is now the `dynamic_cluster_1-Validation` dynamic cluster. The dynamic cluster page shows that the `dynamic_cluster_1-Validation` dynamic cluster is created, and the servers page shows the cloned servers.

Tip: If you want to save the validation cluster after you perform the rollout, you can create the `saveClonedCluster` custom property on the validation cluster. Otherwise, the validation target is deleted after the edition rollout or after the validation is canceled for all of the applications on the validation target. For example, if you have two applications deployed to the validation target, and one of the applications is validated and rolled out, the validation target is not deleted until the second application is validated. The `saveClonedCluster` custom property applies only to dynamic clusters. For more information, read about application edition manager custom properties.

6. Verify that the validation occurred correctly. Click **Applications > Enterprise applications** or **Applications > All applications**. Edit the `my_application-edition2.0` application.
 - **For PHP and WebSphere Application Server Community Edition applications:**
Verify that the context root, deployment targets, and so on are pointing to the cloned cluster.

- **For enterprise (Java Platform, Enterprise Edition (Java EE)) applications:**

Select **Manage modules**. Verify that edition 2.0 is mapped to the validation cluster. From the Map Enterprise JavaBeans (EJB) references to beans detail view, verify that the Java Naming and Directory Interface name is adjusted for the new cloned target name.

For an application edition with fully qualified bindings based on the original deployment target name to operate correctly on a validation deployment target, you must change its binding names to reflect the fully qualified binding names based on the validation deployment target name. For example, an application with a resource reference bound to `/clusters/clusterb1/jdbc/CustomerData` must have the binding changed to `/clusters/cluster1-validation/jdbc/CustomerData` as the application is prepared to run on the deployment target clone.

7. Test the new edition. Start the validation cluster, and with your routing rules in place, try sending a request load to the edition 2.0 edition to test the edition. The edition 1.0 edition remains in production.

What to do next

If you successfully complete the edition 2.0 edition testing, you can replace the edition 1.0 edition with the edition 2.0 edition. If you encounter errors in your testing, you can cancel the validation mode.

- To replace edition 1.0 with edition 2.0:
 1. Stop the validation target, for example, `dynamic_cluster_1-Validation`.
 2. Delete the routing rules that are specific to edition 2.0 to route all requests for the application to a single edition.
 3. Save your changes and synchronize the nodes.
 4. Perform rollout to the new edition. Click **Applications > Edition control center > *application_name***. Select edition 2.0 and click **Rollout**. During the rollout, edition 2.0 is retargeted to its original deployment target, for example, `dynamic_cluster_1`. The edition state transitions from validation to active.
- If edition 2.0 has errors, you can cancel validation mode and move edition 2.0 back to its original inactive state. As a result, the duplicate dynamic cluster that was created for validation is removed. For more information about cancelling the validation mode, read about canceling an application validation.

Canceling an application validation

You can cancel validation of an edition and return the application edition to inactive state.

Before you begin

You must have an application installed and in the validate state.

About this task

By canceling an application in validation mode, the application returns to the inactive state. The deployment target for the edition is reset to its original deployment target. When you cancel an application in validation mode, the validation target is deleted.

Procedure

Complete the following steps to cancel an application in validation mode:

1. In the administrative console, click **Applications > Edition control center > *application_name***.
2. Select the application that is in the validate state and click **Cancel validation**.
3. View the cancel validation status page to view each step of canceling the validation. During the validation cancellation, the following actions occur:
 - The application is set to the inactive state.
 - The validation cleanup process is performed

- The validation target is deleted. If the validation target is a dynamic cluster, you can prevent it from being deleted by setting the `saveClonedCluster` custom property on dynamic cluster. For more information, read about application edition manager custom properties

The edition control center shows that edition is now in the inactive state, and the manage editions page shows that the deployment target for the edition is restored to its original deployment target. The dynamic cluster page shows that the validation dynamic cluster no longer exists, and the servers page shows the cloned servers no longer exist.

Remember: The validation deployment target is deleted unless other applications are still deployed on the validation deployment target.

4. Verify that the cancelation of validation occurred correctly. Click **Applications > Enterprise applications** or **Applications > All applications**. Edit your edition 2.0 application.
 - **For PHP and WebSphere Application Server Community Edition applications:**
Verify that the context root, deployment targets, and so on are restored to the original cluster.
 - **For Java Platform, Enterprise Edition (Java EE) applications:**
Select **Manage modules**. Verify that edition 2.0 is mapped to the original cluster. From the **Map Enterprise JavaBeans (EJB) references to beans** detail view, verify that the Java Naming and Directory Interface name is adjusted for the original target name.
5. Delete the routing rules that are set up to direct requests to the edition in the validation state. Click **Applications > All applications > *application_name* > Routing policies**. Save and synchronize the nodes.

Results

The application edition is inactive and is restored back to its original deployment target.

Performing a rollout on an edition

When you perform a rollout on an edition, you replace an active edition with a new edition. The new edition might be a simple modification to the application, or contain a more substantial change. If the new edition is compatible with earlier versions, then you can perform a rollout to replace the active edition without impacting existing clients. To perform a rollout on a new edition, you must first install the application edition with the new edition information.

Before you begin

You must have an application edition that is installed and started, and have *configurator* or *administrator* privileges to perform a rollout.

- Performing a rollout fails when two user IDs on two administrative consoles attempt to complete the process in parallel.
- Tune the SOAP connector properties to set the request timeout value for the deployment manager to be greater than the total time required to perform a rollout on your system, and restart the deployment manager. Not setting the property can cause the rollout process to fail when the `requestTimeout` value expires. The formula to estimate the value to set is `number-of-groups-to-rollout * (drainage-interval + internal-quiesce-timeouts-approximately-5minutes + application-or-server-restart-times-approximately-10minutes)`. Alternatively, you can set the value to zero to disable the timeout.
 - If you are performing the rollout using the `wsadmin` tool, adjust the request timeout value by setting the `com.ibm.SOAP.requestTimeout` property in the `soap.client.props` in the deployment manager profile. The default value is 180 seconds and should be increased adequately.
 - If you are performing the rollout using the administrative console, adjust the request timeout value by clicking **System administration > Deployment manager > Administration services > JMX connectors > SOAPConnector > Custom properties > requestTimeout**. The default value is 600 seconds and should be increased adequately.

For more information, read about Java Management Extensions connector properties.

- If you are performing a rollout within the administrative console, set the session expiration for the administrative console to a value greater than the amount of time required for the entire rollout process to end. Multiply the request timeout value by the number of groups processed during the rollout. For more information about session expiration for the administrative console, read about changing the console session expiration.
- You must define a multi-cluster routing policy for each new edition you install before you can perform a rollout. Use the administrative tasks to add multi-cluster routing policies for your new editions. For more information about multi-cluster routing policies, read about rules for ODR routing policy administrative tasks.

About this task

You can also use the application edition manager if you want to perform a rollout to batch applications. These are Java Enterprise Edition 5 (Java EE 5) applications that conform to one of the batch programming models.

Procedure

1. Install the new edition. Use the same steps that are described in installing an application edition, but specify your new edition information. For example, type 2.0 in the **Application edition** field and Second edition in the **Application description** field. Select the same deployment targets that are used for the current edition.
2. Save and synchronize your nodes.
3. Specify the rollout settings. Click **Applications > Edition control center > application_name**. Select your new edition, for example, 2.0, and click **Roll out**.

Specify the following settings for enterprise or other middleware applications:

- a. Select **Atomic** or **Grouped** rollout type.

Use *group rollout* to replace editions on members of the target cluster in a group of one. Group rollout is the most typical choice, and is useful when the cluster contains four or more members. Alternatively, you can perform group rollout with a specified group size through scripting. For more information about group rollout, read about application edition management administrative tasks. When the new edition becomes available during group rollout, all requests are directed to the new edition.

Use *atomic rollout* to replace one edition with another on half of the cluster at a time. This rollout type serves all user requests with a consistent edition of the application. Because all user requests are served a consistent edition, your cluster runs at half capacity. If your cluster has four or more members, consider dividing up the cluster into smaller groups by performing a group rollout. Atomic mode is also used with a single server deployment target. In a single server deployment target, the actions that are carried out against the second half of the cluster are omitted. If you stop your deployment targets before you start atomic rollout, the deployment targets are started when the new edition replaces the active edition regardless of the reset strategy you choose. This procedure provides better availability to the requests that are serviced during the rollout period.

Note: Before you perform an atomic rollout, determine the load capability of the target server cluster. Performing an atomic rollout activates the new edition on half of the cluster first, and then activates the edition on the remaining half of the cluster. While the first half of the cluster is taken offline and updated, application requests are routed to the second half of the cluster. Verify that half the cluster can handle the entire load during the rollout period.

- b. Select the reset strategy. The reset strategy instructs the application edition manager how each deployment target loads the new edition into the server runtime.

Use a *soft* reset strategy to reset the application by stopping or restarting the application in each server of the cluster as the next edition replaces the old edition in that server. Soft reset is the most typical choice and the most optimal performing application reset because it results in loading

the new edition by recycling the application in the running application server. The server stays up during this process. With soft reset, 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, monitor the application server process to ensure that sufficient virtual memory exists.

A *hard* reset strategy recycles the entire application server of the cluster as the next edition replaces the former edition in the server, refreshing both process memory and any native libraries used by the application. This strategy prevents virtual storage exhaustion and allows new versions of native libraries to be loaded. Select hard reset as your reset strategy when you perform a rollout on an edition that depends on new versions of native libraries or other dependencies that are refreshed only by recycling the entire application server, or if you have large applications that consume a lot of memory for just-in-time compilation (JIT).

- c. Set the drainage interval in seconds. The drainage interval gives the HTTP sessions time to complete before the application or server is reset. The drainage interval specifies the amount of time that the application edition manager waits before the reset strategy starts.

Affinities, such as transaction, activity, and compensation-scope, and activities unknown to Intelligent Management, lengthen the effective drainage interval because the server does not stop until these units of work complete. Applications with activities unknown to Intelligent Management can use the `AppEditionManager` MBean quiesce initiated notification as a trigger to begin shutdown processing and use the drainage interval as a time period during which to complete the shutdown. This process is unnecessary for persistent sessions, for example, those backed in database or replicated through VMware Distributed Resource Scheduler (DRS), but is important for transient (in memory) sessions.

The goal of the drainage interval is to allow requests with affinities and inflight requests to complete. To prevent the loss of transient sessions, set the drainage interval to exceed the application session timeout interval. After the rollout starts, as each server updates, the server is marked as ineligible to begin any new sessions. Set this value to 0 to not wait for sessions to complete.

The application edition quiesce manager might not wait the full length of the drainage interval. Performance Monitoring Infrastructure (PMI) statistics are available for the quiesce manager to determine if all active requests on a server have been quiesced. If all requests are quiesced before the drainage interval, the application edition quiesce manager does not need to wait for the full drainage interval. To force a soft reset to wait the entire drainage interval, you can set the `apedition.rollout.softreset.fulldrainageinterval` system property to true on the deployment manager.

The drainage interval allows existing sessions to complete. However, at the end of the drainage interval, a period exists during which inflight requests can still arrive. In such cases, the on-demand router (ODR) provides a timeout value of 60 seconds within which to complete the quiesce operation. If the requests end within 60 seconds or the 60 seconds expire, the application, or the server in the case of a hard reset strategy, is stopped. Next, if inflight requests have still not ended, WebSphere Application Server Network Deployment provides a quiesce time of 60 seconds before stopping the application or the server instance. For hard reset strategies, WebSphere Application Server Network Deployment provides a quiesce time of 180 seconds before stopping the server instance. You can use the `com.ibm.ws.webcontainer.ServletDestroyWaitTime` custom property to define the amount of time that the Web container waits for the requests to complete. For more information, read about web container custom properties.

You can use the `com.ibm.ejs.sm.server.quiesceTimeout` custom property to define the amount of the time that the server instance waits for the requests to complete before initiating shutdown. For more information about the timeout property, read about Java virtual machine custom properties. You must set both the `com.ibm.ws.webcontainer.ServletDestroyWaitTime` custom property and the `com.ibm.ejs.sm.server.quiesceTimeout` custom property on each of the server instances on which the application editions are deployed.

Specify the following settings for Session Initiation Protocol (SIP) applications:

- a. Choose a quiesce strategy. The quiesce strategy specifies how old servers or cluster members that host the current edition are removed. This setting does not affect the new edition that is being rolled out.

Quiesce server or cluster members after all active sessions or dialogs are completed:

Removes the server or cluster member when all of the active sessions and dialogs for the server or cluster member complete.

Quiesce servers or cluster member after the specified interval:

Removes the server or cluster member after a specified time period. Specify an amount of time, in seconds, minutes, or hours.

Attention: Performing a rollout is not supported for SIP applications that are deployed on a dynamic cluster that has been converted from a static cluster.

4. Start the rollout. Click **OK**. This action launches an interruption-free replacement of the previous edition with your new edition.

Results

For an edition that is not in validation mode, the new edition replaces the current edition after the rollout completes. An edition that is in validation rolls out on the original deployment target and the cloned environment is deleted. The routing rules are updated to begin routing to your new edition.

What to do next

To validate the results, click **Applications > Edition control center > *application_name***. Your new edition is the active edition on the deployment target. The new edition automatically starts, because it replaces a running edition.

When you perform a rollout on an edition in validation mode, the binding names are changed back to the original values. For example, `/clusters/cluster1-validation/jdbc/CustomerData` is changed back to `/clusters/cluster1/jdbc/CustomerData`.

Algorithm for performing a rollout

The algorithm for performing a rollout to a new edition has operational implications on your environment. The installation and distribution of an application edition is separate from its activation.

Two basic patterns exist for interruption-free replacement: group rollout or atomic rollout. The steps that occur to perform a rollout to a new edition vary depending on which of these options you choose.

The dynamic clusters are put in manual mode during the rollout. If your load becomes heavy during the rollout, application placement does not occur. Plan your rollouts so that you avoid peak periods or heavy loads. When the rollout completes, the dynamic cluster is put back into its original mode.

Note: Do not perform a rollout during periods of heavy traffic.

Group rollout

When you choose to perform a group rollout, the rollout occurs across the clusters in groups of servers. The following steps occur for each server:

1. Quiesce work to the server.
2. Stop the application or stop the server.
3. Update the server configuration.
4. Restart the application or the server.
5. The server is ready with the new edition.

Atomic rollout

Before you perform an atomic rollout, determine the load capability of the target server cluster. Performing an atomic rollout activates the new edition on half of the cluster first, and then activates the edition on the remaining half of the cluster. While the first half of the cluster is taken offline and updated, application requests are routed to the second half of the cluster. Verify that half the cluster can handle the entire load during the rollout period.

When you choose to perform an atomic rollout, the following steps occur:

1. Quiesce work to half of the servers.
2. Stop the applications or servers in the first half of the servers.
3. Update the configurations.
4. Start the applications or servers in the first half of the servers.
5. Quiesce work to the second half of the servers.
6. Start routing requests to the new edition, which is running on the first half of servers.
7. On the second half of the servers, stop the applications or servers, update the configurations, and start the applications or servers.
8. Rollout is complete.

Default rollout settings

The following options are preset for the rollout actions in the administrative console:

- Group rollout:
 - rollout strategy = group, group size = 1
 - reset strategy = application
 - drainage interval = 30 seconds
- Atomic rollout:
 - rollout strategy = atomic
 - reset strategy = application
 - drainage interval = 30 seconds

Scripting interface rollout options

The group and atomic rollout options in the administrative console offer a preset selection of rollout options. Greater flexibility over these options is possible through the scripting interface. Read about application edition management administrative tasks for more information. The following scripting options exist:

- **Rollout strategy:** Specifies the rollout method, either groups of nodes updated serially or the divide-and-swap atomic strategy.
 - **Group:** Specifies that the target cluster is divided into groups for rollout. Group rollout is most effective when your cluster is large. You can specify the group size with a sub-option. The group size gives the number of nodes to process at a time. The default is 1.
 - **Atomic:** Specifies that only one edition of the application can serve requests during the rollout period. This results in half of the application server cluster being taken offline and updated, and then the other. Application requests that arrive while both halves of the cluster are offline are queued by the on demand router (ODR).
- **Reset strategy:** Specifies whether to recycle, for example, stop and restart, the application or the entire application server.
 - **Application:** Activates the new edition in each application server by recycling the application. The application server continues to run.

- **Server:** Activates the new edition in each application server by recycling the server itself. This option is necessary if you need to refresh connectors, native libraries, or reset the Java virtual machine.
- **Drainage interval:** Specifies the amount of time to wait for processing requests to complete before the application or application server is stopped. The default is 30 seconds.

Performing a rollback on an edition

If you perform a rollout on a new edition and an error exists in your application, you can undo your change and easily revert back to the previous edition. To rollback to the edition, perform the rollout steps, but select a former edition to replace the current edition.

Before you begin

If you are a user with either a monitor or an operator role, you can only view the application edition manager information. If you have the role of configurator or administrator, then you have all the configuration privileges for the application edition manager.

The edition to which you want to roll back must be in the system. If your edition is not in the system, install the edition and specify edition information that indicates that the edition is earlier than your current edition. For example, if the current edition is 2.0, specify the edition that you want to rollback to as 1.0. Select the same deployment targets for both editions. Clone the work classes from the current edition to the previous edition.

About this task

For application resiliency, rollback an edition when there are errors with your currently active application edition. You can revert to the previous edition of the application. Rollback the current edition by rolling out a previous edition.

Procedure

1. Expand **Applications > Edition control center** in the administrative console. Select the former edition, for example 1.0.
2. Click **Rollout**. This action launches an interruption-free replacement of the current edition with the selected previous edition.

Results

The former application edition is active.

What to do next

Troubleshoot errors with the newer application edition, and perform an edition rollout to activate the newer edition.

Deleting an installed edition

After multiple successive editions of an application are deployed and rolled out, which is not unusual in the life span of many production applications, you might have a number of editions stored in your repository that are no longer needed. You can delete an active edition without performing a rollout to a new edition.

Before you begin

- You must have an installed, active edition.
- You must have a configurator or administrator administrative role.

About this task

An edition can exist in one of three states. Therefore, the method for deleting or uninstalling an edition varies slightly based on its current state.

- **Inactive:** The edition is neither assigned for use in any deployment target, nor the default. You can remove an inactive edition at any time.
- **Active:** The edition is currently assigned for use in one or more deployment targets. Use caution when you delete an active edition, especially if the edition is started. Stop the edition first, remove any routing rules that reference the edition, deactivate it, and you can then remove or uninstall the edition. After you delete an active edition, you must update the routing rules, and click **Apply** and **Save** regardless of what item is displayed in the list.
- **Validate:** The edition is currently assigned for use in the cloned deployment targets. For more information about removing an application in validation mode, read about canceling an application validation.

Procedure

1. Remove any routing rules that reference the edition you are uninstalling. Click **Applications > Enterprise applications > application_name** or **Applications > All applications > application_name**. Click the **Routing policies** tab and select the appropriate routing rules.
2. Click **Applications > Edition control center**. Select the target application.
3. Stop the application edition.
4. Deactivate the active edition. Select the edition, and click **Deactivate**.
Instead of deactivating an active edition, you can replace it with a new edition. You can perform a rollout to a new edition, which makes the old edition inactive. Use rollout to ensure an interruption-free deactivation of the edition you want to delete.
5. Uninstall the application. Click **Applications > Enterprise applications** or **Applications > All Applications**. For an enterprise application, select the edition that you want to uninstall, click **Uninstall**. For a middleware application, select the edition, and click **Remove**. You are not required to restart the servers.
6. Save and synchronize your changes.

Results

You removed the edition.

Troubleshooting application edition manager

When you encounter a problem with the application edition manager, verify that the servers are running or refer to the log files to further investigate the errors.

Procedure

- While performing a rollout, verify that your servers are running. Node agents, the on demand router (ODR), and servers must be running to perform the rollout action.
- Check the log files of the deployment manager, the node agents, and the servers that are involved in the current operation.
 - **Distributed operating systems** On distributed operating systems, the log file is the SystemOut.1log file.
 - **z/OS** On z/OS systems, the log file is the servant job log.

You can use the log files when you encounter errors. If you report the problem, it is helpful to collect and save these files. The preferred mechanism is to use the collector utility on each affected node, including the deployment manager, and be prepared to send the output. The collector utility is in the app_server_root/bin directory on each product installation directory.

- If you are frequently experiencing deployment manager timeout on a z/OS system, increase or decrease the timeout values.

Configuring timeout values for application edition manager on z/OS systems

If you are using application edition manager on a z/OS system, and are experiencing deployment manager timeouts, you should either increase the deployment manager timeout values or set the timeout values to zero.

Before you begin

Configure your application editions.

About this task

To avoid deployment manager timeout, you might want to either increase the system timeout values based on the system responses and the site environment, or you might disable timeout. By disabling the deployment manager timeout, you can prevent timeout from occurring and keep the deployment manager servant region from stopping and restarting.

Procedure

1. Change the workload manager (WLM) timeout value on the object request broker (ORB) to zero.
 - a. In the administrative console, click **System administration > Deployment manager > ORB service > z/OS additional settings**.
 - b. Change the **Workload manager timeout** value to 0. The default value is 300 seconds.
2. Modify the `ConnectionResponseTimeout` custom properties. These properties specify the amount of time that the Java Platform, Enterprise Edition (Java EE) server waits for an application component to respond to a Hypertext Transfer Protocol (HTTP) request.
3. Update the timeout values in the Web container transport chains.
 - a. In the administrative console, click **System administration > Deployment manager > Web container transport chains**.
 - b. Modify the `WCInboundAdmin` variable. Modify the following transport channels:
 - Click **TCP inbound channel (TCP_1)** and set the **Inactivity timeout** to 0 seconds.
 - Click **HTTP inbound channel (HTTP_1)** and set the **Read timeout**, **Write timeout**, and **Persistent timeout** values to 0 seconds.

This setting affects Internet Inter-ORB Protocol (IIOP) work that is queued to the servants.

The **Web container inbound channel (WCC_1)** channel does not require any changes.

- c. Modify the `WCInboundAdminSecure` variable. Modify the following transport channels:
 - Click **TCP inbound channel (TCP_3)** and set the **Inactivity timeout** to 0 seconds.
 - Click **HTTP inbound channel (HTTP_2)** and set the **Read timeout**, **Write timeout**, and **Persistent timeout** values to 0 seconds.

The **SSL inbound channel (SSL_1)** and **Web container inbound channel (WCC_2)** channels do not require changes.

What to do next

Configure other application edition manager features.

Application edition manager frequently asked questions

When you start to use application edition manager, you might come across some questions about the application edition manager functionality.

Can the application edition manager ensure interruption-free application upgrade for workloads that are driven by Internet InterORB Protocol (IIOP) or Java Message Service (JMS) protocols?

No. The application edition manager can ensure interruption-free application upgrades only for workloads managed by the on demand router (ODR). The ODR supports HTTP and HTTPS only.

Can the application edition manager ensure interruption-free application upgrades for workloads that include calls between Java Platform, Enterprise Edition (Java EE) Version 5 applications?

Yes, the application edition manager can ensure an interruption-free update if each interapplication request uses HTTP or HTTPS protocol through an on demand router.

Can the application edition manager ensure interruption-free application upgrades for workloads that are hosted on static clusters?

Yes, if an on demand router is managing the work sent to that static cluster.

Can the application edition manager support the deployment of one edition of an application to one cluster and the deployment of another edition to a different cluster?

Yes. The application edition manager supports different editions simultaneously deployed on different targets. You must establish routing policies for routing traffic to different editions.

Does the application edition manager support the activation of different editions on different members of the same cluster?

No.

Does the application edition manager restart the application or the entire server to activate a new edition?

No, the application edition manager does not restart the application or server. You must perform the following steps:

1. Stop the active edition.
2. Deactivate the active edition.
3. Activate the new edition.
4. Start the new edition.

Can the application edition manager handle the rollout of editions with incompatible changes?

Yes, the application edition manager can handle this situation if the on demand router is configured to perform edition-aware routing. For more information about edition-aware routing, read about edition compatibility.

Can the application edition manager handle the rollout of editions that include database schema changes?

Yes, you can perform rollout of editions that include database schema changes if the database changes are compatible with earlier versions.

Managing the Intelligent Management environment

You can configure additional functions to enhance the autonomic functions of the application infrastructure virtualization environment. These functions include maintenance mode, health management, centralized installation manager, high availability deployment manager, repository checkpoints, and autonomic managers.

Before you begin

Configure the on demand router (ODR), servers, dynamic clusters, applications, and service policies.

About this task

After you set up your basic hosting environment with application server virtualization, you can tune the behavior of the managers that make decisions in your environment. You can also configure several features to help with administration.

Procedure

- Set the maintenance mode.
- Use fine-grained security in the Intelligent Management environment.
- Configure the autonomic managers.
- Manage routing requests to nodes outside of the Intelligent Management cell.
- Set up a high availability deployment manager environment.

Setting maintenance mode

Set maintenance mode before you perform diagnostic tests, maintenance, or tuning on a node or server. Maintenance mode can prevent the disruption of client requests by routing client traffic that is targeted for a server or node that is in maintenance mode to another server or node.

Before you begin

- You can set maintenance mode on a server or a node that is represented in the administrative console.
- To view the nodes that are configured, click **System administration > Middleware nodes**. To view all of the servers that are configured, click **Servers > All servers**.
- You can use maintenance mode when you route requests with on demand routers (ODR) or proxy servers. If you are using a web server to route requests, maintenance mode is not recognized.
- ODR maintenance mode is supported when the ODR is fronted by a web server plug-in. The `plugin-cfg.xml` file generator generates the plug-in appropriately to enforce the various modes of node and server maintenance mode. You can use node and server maintenance mode to apply service to the ODR tier in the same way that you use maintenance mode to apply service to the application server tier.

About this task

When a node or server is in *running* state, the dynamic operations environment considers that node or server to be available for servicing application requests. When problems occur, you must perform diagnostics, maintenance, or tuning on the node or server. Use *maintenance mode* to stop routing traffic to the node or server while you determine the problem.

When a server is in maintenance mode, the application placement controller cannot control that server. The server does not count toward the minimum or maximum running instances setting for the dynamic cluster. For vertical stacking, the configured maximum minus the number of servers in maintenance mode on the node is used to determine if any servers need to be created on the node. By using this method, the

dynamic operations environment does not create or destroy server instances to support the configured maximum number of instances as the dynamic cluster instances move in and out of maintenance mode.

You can use maintenance mode to perform node-level maintenance mode for the ODR nodes just as you can perform node-level maintenance mode for application server nodes. Similarly, you can use server maintenance mode for your ODRs just as you can for application servers.

Two routing policies, the `permitMM` and `permitstickyMM` routing policies, are work class match actions that you can use to route HTTP traffic only to servers in maintenance mode. For more information, read about routing and service policies.

Procedure

1. Place nodes into or out of maintenance mode.
 - a. Select the nodes. In the administrative console, click **System administration > Middleware nodes**. Select the nodes for which you want to change the maintenance mode.
 - b. Select the maintenance mode.

Maintenance mode

The on demand router (ODR) continues to send requests with affinity to an application server in this mode, but typically does not send requests without affinity (unless the request matches a rule which explicitly permits it to be routed to an application server in maintenance mode).

Maintenance immediate stop

This mode is the same as the break affinity mode, except that the application server is also immediately stopped. When the application server is restarted, it is still in break affinity mode.

Normal

The ODR sends requests with and without affinity to an application server in this mode.

- c. Click **Set mode**.
2. Place servers into or out of maintenance mode. In the administrative console, click **Servers > All servers**. Select the servers for which you want to change the maintenance mode.
 - a. Select the servers.
 - b. Select the maintenance mode.

Maintenance mode

The ODR continues to send requests with affinity to an application server in this mode, but typically does not send requests without affinity (unless the request matches a rule which explicitly permits it to be routed to an application server in maintenance mode).

Maintenance mode - break affinity

The ODR typically does not send any requests to an application server in this mode (unless the request matches a rule which explicitly permits it to be routed to an application server in maintenance mode).

Maintenance immediate stop

This mode is the same as the break affinity mode, except that the application server is also immediately stopped. When the application server is restarted, it is still in break affinity mode.

Normal

The ODR sends requests with and without affinity to an application server in this mode.

- c. Click **Set mode**.

After a server is placed into maintenance mode, the changes are synchronized in all nodes. If the cell level custom property `maintenanceModeSyncContainingNodeOnly` is set to true, the changes are synchronized with the node on which the server resides. This custom property takes effect dynamically. No restart is required.

Results

HTTP and Session Initiation Protocol (SIP) traffic is no longer routed to the nodes or servers that you have in maintenance mode. Enterprise JavaBeans (EJB) affinity is not broken. Requests with EJB affinity and Java Message Service (JMS) traffic are still routed to the server that is in maintenance mode.

What to do next


The health controller also uses server maintenance mode as an action that is taken when the health policy is breached. For example, if you are using an excessive response time condition, and the response time exceeds the specified maximum for a server, then the server can move to maintenance mode.

Routing to servers that are in maintenance mode

You can build complex rule conditions from subexpressions to modify routing policy Session Initiation Protocol (SIP) rules, and route certain messages to servers that are in maintenance mode while routing other traffic to servers that are not in maintenance mode from within the same dynamic cluster.

Before you begin

To modify rules through the rule builder, you must have administrator privileges.

Restriction:  Intelligent Management does not support SIP features on the z/OS operating system.

About this task

Typically, only servers that are not in maintenance mode receive and service application requests. You can also route to servers that are in maintenance mode, so you can route to a subset of servers to verify configuration changes or troubleshoot issues without interfering with availability. You can direct SIP traffic to specific clusters with routing rules. The expressions for SIP routing rules are enhanced to support an expression that provides more functionality and flexibility in defining the target cluster. For example, the expression `cluster='TestCell/TestClusterA'` and `serverMaintenanceMode='affinity'` direct the on demand router (ODR) to route calls to members of TestClusterA that are in maintenance mode. You can also use this function in conjunction with the prioritized routing rules.

Procedure

1. From the administrative console, click **Servers > Server types > On demand routers > odr_name > SIP on demand router settings > Routing policy SIP rules > Specify by > Expression > Subexpression builder**.
2. Select **and** as the logical operator.
3. Select **Server maintenance mode** as the operand.
4. Select **Equals (=)** as the operator.
5. Select **Affinity** as the value. Click **Generate subexpression**.
6. Click **Append** to add the subexpression to your rule.

Generating Simple Network Management Protocol (SNMP) traps

You can generate Simple Network Management Protocol (SNMP) Version 1 traps for health events and dynamic cluster start and stop application server events.

Before you begin

Any standard SNMP management endpoint that is capable of receiving Version 1 traps can receive traps that are sent by Intelligent Management. See the documentation associated with your SNMP management endpoint for instructions on how to import the Management Information Base (MIB) file that is provided with Intelligent Management.

About this task

You can generate SNMP Version 1 traps for the following two types of events:

- Health events: In this case, sending an SNMP trap is one of the actions that can be performed when a health policy is triggered. The SNMP trap contains the name of the health policy that was triggered, when the health policy was triggered, and where the health policy was triggered.
- Dynamic cluster start and stop application server events: When the application placement controller starts or stops an application server, the controller can send an SNMP trap notifying an SNMP management endpoint.

The precise events are defined by a MIB file in the `WAS_HOME/etc/wvemib.txt` file. You can use this MIB file to import the events into your SNMP management endpoint.

Complete the following procedure to generate SNMP traps.

Procedure

1. Configure the host name and an optional port of the SNMP management endpoint. Set the following custom property:
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as `snmp.manager`.
 - c. Specify the value of the custom property as `hostname[:port]`. This is the host name and UDP port to which the SNMP traps are sent. The default port value is 162.

For example, setting the value of the custom property to `mySnmpManagerHost:162` indicates that SNMP Version 1 traps should be sent to UDP port 162 of the host named `mySnmpManagerHost`.
2. Configure the product to send an SNMP trap for health policy violations. Create a new health policy and select the **Generate an SNMP trap** predefined health policy action as one of the actions to perform when the health policy is triggered. In the administrative console, select **Operational policies > Health policies > New**.
3. Set the following custom property to disable SNMP traps from being sent when the application controller starts or stops a member of a dynamic cluster.
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as `snmp.disabledTraps`.
 - c. Specify the value of the custom property as `dynamicClusterStartStopServer`.
4. Configure the SNMP community string. By default, the product sends an SNMP trap with a community string value equal to `public`. Set the following custom property to override the default value.
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as `snmp.community`.
 - c. Specify a value for the custom property. The value of the custom property is then used as the community string value.

Configuring the autonomic managers

Use these instructions to configure the behavior of the autonomic managers for Intelligent Management.

Before you begin

For more information on setting up your system, read about preparing the hosting environment for dynamic operations.

About this task

After preparing your environment to support autonomic management, configure autonomic managers to maximize the environment utilization based on your defined business goals. Autonomic managers monitor performance metrics, analyze the monitored data, offer a plan for running actions, and can start these actions in response to the flow of work. The following autonomic managers work in or with the Intelligent Management environment.

Procedure

- Configure the autonomic request flow manager.
- Set up and monitor dynamic application placement.
- Monitor performance. Configure and monitor the database tier.
- Enable health management.

Configuring the autonomic request flow manager

You can fine tune the autonomic request flow manager (ARFM) by changing the default settings in the administrative console.

Before you begin

To change the settings on the autonomic request flow manager, you must have operator, configurator, or administrator administrative privileges. Operators can only view the information on the configuration tab, but can change the settings on the runtime tab. Configurators can change settings on the configuration tab, but cannot change settings on the runtime tab. Administrators have all privileges.

When security is enabled, some fields are not editable without proper security authorization.

About this task

The autonomic request flow manager contains the following components:

- A controller per target cell, such as a cell to which an ARFM gateway directly sends work. This is an HAManagedItem process that runs in any node agent or deployment manager.
- A gateway per used combination of protocol family, proxy process, and deployment target. A gateway runs in its proxy process. For HTTP and Session Initiation Protocol (SIP), the proxy processes are the on demand routers; for Java Message Service (JMS) and Internet Inter-ORB Protocol (IIOP), the proxy processes are the WebSphere Application Server application servers.
- A work factor estimator per target cell, which is an HAManagedItem process that can run in any node agent, ODR, or deployment manager.

The gateways intercept and queue the incoming HTTP, SIP, JMS, and IIOP requests, while the controller provides control signals, or directions, to the gateways and the placement controller. The work profiler continually estimates the computational requirements of the various kinds of requests, based on observations of the system in operation. Working together, these components properly prioritize incoming requests.

z/OS Dynamic placement function with job scheduler is not supported on z/OS servers.

Procedure

1. Modify the appropriate ARFM settings. In the administrative console, click **Operational policies > Autonomic managers > Autonomic request flow manager**.

2. Click **OK** or **Apply** when you have completed your changes.
3. Click **Save** to save the changes to the master repository.
4. Test the settings you have just defined and iterate as often as necessary to get the request flow performance that you want.

Example

The following table provides specific guidance for configuring each setting.

Table 28. ARFM configuration properties

Field	Purpose	Tips for setting
Aggregation period	<p>Each ARFM gateway broadcasts aggregated statistics periodically, and this parameter specifies the period.</p> <p>The statistics reported by the gateways support: the runtime charting in the administrative console, the operation of ARFM controllers, the operation of the application placement controller, and the operation of work profilers.</p>	<p>When setting the aggregation period, ensure the value is high enough to allow for the collection of a sufficient number of performance samples. Samples are collected by the gateways for each request. A few hundred samples are necessary to produce a good statistical measure.</p> <p>Using an example - requests associated with a service class run in 250 milliseconds, and on average 10 requests run concurrently. The concurrency value is calculated automatically, based on the cluster size and the resources in the environment. The concurrency value can be seen on the visualization panels, under the Runtime Operations category in the console. As a result, the service class handles about 40 requests per second. Therefore, setting the aggregation period value to 15 seconds results in the collection of 600 samples for each aggregation period. The metrics provided by a 600 sample survey are useful and reliable.</p> <p>Setting an aggregation period value too low results in unreliable performance metrics. Performance metrics derived from fewer samples are more noisy and less reliable, then a higher sample size. Because the ARFM controller is activated when new statistics are produced, setting an aggregation period value that is too long results in less frequent recomputation of the control settings. Therefore, Intelligent Management becomes less responsive to sudden changes in traffic intensities and patterns.</p>
Control cycle length minimum	<p>This parameter defines how often the ARFM controller is activated.</p> <p>Controller activation is the process of evaluating inputs and producing new control settings as a result of the input received. The activation process for an ARFM controller is initiated when new statistics are received from one of its gateways AND the elapsed time since the previous activation is greater than or equal to the control cycle minimum length, or the controller has never activated before.</p>	<p>This setting determines the control cycle length giving it a lower bound. For example, if you have just one ODR and set the aggregation period to 30 seconds and the control cycle minimum length to 60 seconds, you might find that one activation occurs at 12:00:00.0 and the next occurs 90.1 seconds later at 12:01:30.1 because the previous statistics arrival time was 12:00:59.9. To ensure a reliable control cycle of around 60 seconds, set the control cycle minimum length to 58 or 59 seconds.</p>

Table 28. ARFM configuration properties (continued)

Field	Purpose	Tips for setting
Smoothing window	<p>This setting defines how sensitive the ARFM controller reaction is to the incoming gateway statistics, by allowing a concatenation of gateway statistics. For any gateway, its ARFM controller uses a running average of the last few statistics reports from that gateway. The smoothing window controls the number of reports that are combined.</p>	<p>A low smoothing window setting makes the controller more sensitive and react more quickly. However, a low parameter also creates a sensitive reaction to noise, or anomalies, in the data.</p> <p>The product of the smoothing window and the aggregation period should be roughly the same as the actual control cycle length, which is sometimes slightly greater than the configured control cycle minimum length.</p>
Maximum queue length	<p>This parameter is used to bound the length of each ARFM queue to a maximum number of requests that can be held in queue. ARFM divides all incoming traffic into flows, and has a separate queue for each flow. Flow particulars include requests that have a particular service class, are served on a particular deployment target, or go through a particular ODR.</p> <p>When a request arrives and its queue is full, the request is rejected.</p>	<p>A lower parameter in this field increases the possibility that a request will be rejected due to short-term traffic bursts, while a higher parameter in this field can allow requests to linger longer in the queues. Queued requests consume memory. The default setting is 1000, but you can experiment with this setting to find the one that is a best match for your environment.</p>
Maximum CPU usage	<p>The ARFM provides overload protection, in addition to its prioritization capabilities. An ARFM will queue requests in its gateways to avoid overloading the application servers.</p> <p>For this release, load is determined in terms of processor utilization on the first tier of application servers. The maximum CPU utilization parameter tells ARFM how heavily to load the servers. During severe peak conditions this utilization limit might be briefly exceeded.</p>	<p>Higher values give better resource utilization; lower values give more robust operation. Real load is noisy and variable. The performance management techniques in Intelligent Management react to changes in the load, but with some time delay. During that reaction time, the system might operate outside its configured region; this includes having higher processor utilization than configured. Operation with one application server at 100 percent processor utilization for multiple minutes has been observed to break some internal communication mechanisms, to the detriment of many features.</p> <p>The performance management in this release of Intelligent Management does not work well if the first tier of application server machines are loaded with other work besides WebSphere requests that arrive through HTTP through the ODRs.</p> <p>This setting affects application placement. If the total predicted demand succeeds the Maximum CPU utilization limit, the placement controller uniformly reduces the demand of all the dynamic clusters before calculating best placement.</p> <p>Set the <code>arfmManageCpu</code> custom property to <code>false</code> to disable processor overload protection and request prioritization. The <code>arfmManageCpu</code> is a cell custom property that you need to create.</p>

Table 28. ARFM configuration properties (continued)

Field	Purpose	Tips for setting
Admission control for CPU overload protection	<p>The purpose of admission control for processor overload protection is to deliberately <i>not</i> accept dialogs based on judgments concerning how much can be accepted without overloading the compute power in the nodes being managed and compromising the response time of the accepted messages.</p> <p>The Admission control for CPU overload protection value applies only to HTTP and Session Initiation Protocol (SIP); it does not apply to IIOOP and JMS.</p> <p>Enable it when queuing for processor overload protection is not enough; when it is important to make deliberate refusals of some offered load.</p>	<p>Disabled by default. To configure:</p> <ol style="list-style-type: none"> 1. Define service policies with achievable performance goals, and set the goal type of the policies either to response time or percentile, not discretionary. 2. In the ARFM panel, set the CPU utilization limit to no higher than 90%. Select the third button for Rejection policy. The rejection policy determines whether the admission control for processor overload protection is enabled and, if so, how the response time threshold used for admission control is related to the response time threshold that appears in the performance goal. 3. At the cell level, set a cell custom property named <code>arfmInitialMsgDlgRatio</code>. The value is a decimal-formatted float that is the initial estimate for the ratio of each of the dialog-continuing message flows to the dialog-initiating message flow within the same (protocol family, deployment target). That is, it is the number of incoming follow-up messages per dialog. Set <code>arfmInitialMsgDlgRatio</code> to a value that is comparable among the collection of all dialog-continuing message flows. <p>This custom property is also relevant when dialog orientation for processor overload protection and differentiated service is enabled.</p> <ol style="list-style-type: none"> 4. Save your changes. <p>The admission control for processor overload protection is working if, in a heavily loaded system, the processor utilization is about the same as the setting for processor overload protection.</p>
Read about memory overload protection	Specifies the maximum percentage of the heap size to be used for each application server.	Maximum percentage of the WebSphere Application Server heap size to use. Set the value to less than 100.
Request rejection policy	Specifies the behavior for HTTP, SIP and SOAP requests that are associated with a performance goal when an overload condition is detected.	<p>Choose among the options to determine when to reject messages to prevent the CPU from being overloaded. You can reject no messages, or specify a rejection threshold value that determines when to reject messages. The default is to reject no messages.</p> <p>Discretionary work is assumed to have a response time threshold of 60 seconds.</p>

What to do next

Use mustGather documents to troubleshoot autonomic request flow manager and application placement issues.

Rate-based autonomic request flow manager (ARFM):

The autonomic request flow manager (ARFM) uses a rate-based algorithm that results in a more consistent loading and protecting of application server resources by ARFM.

ARFM controls the flow of requests for HTTP and SIP traffic through the on demand router (ODR) and for IIOOP and MDB traffic from within an application server.

The rate-based ARFM feature is enabled by default. Complete the following steps to disable the feature.

1. In the administrative console, select **System administration > Cell > Custom properties > New**.
2. Specify the name of the custom property as `enableRateBasedARFM`.
3. Specify the value of the custom property as `false`.
4. Restart the cell.

Configuring emergency throttle:

The on demand router (ODR) and associated autonomic managers are able to support business goals in times of intense request flows by making smart decisions about the work coming into the server. The autonomic request flow manager (ARFM) controls HTTP request prioritization in the ODR. At times, emergency conditions result when certain sensors detect such overloaded situations. These overload situations include extremely high node utilization, intermittent communication failures between ARFM controller and request scheduling gateways, and intermittent communication failures between AsyncPMI monitoring data producers and the gateways. To prevent prolonging of these conditions, if they occur, and the accompanying degradation in performance, the gateways are equipped with emergency throttle controllers that control, and safeguard request dispatch rates to backend nodes. ARFM is handled in the back end for IOP/JMS requests.

The ARFM contains two parts: a controller and a gateway. The ARFM function is implemented, for each node group, by a controller plus a collection of gateways in the ODRs. The ARFM controller (triggered by the eWLM controller if available on the system) might initiate typical throttling directives to the gateways. In a typical mode, throttling directives come from the ARFM controller by way of the `RatesMessages`, and are immediately enforced at the gateway by the throttle controller.

A throttle is attached to each queue in the gateway, and is not in the throttle state by default. When an emergency occurs or when rate messages arrive from the ARFM controller, it receives directives from the throttle controller and changes to the throttled state.

If one or more overload sensors detect overload condition, despite typical throttling, the gateway throttle controller enters emergency mode. An emergency blackout sensor senses communication failures between an ARFM controller and request scheduling gateways, or communication failures between AsyncPMI monitoring data producers and the gateways. The term *blackout* means that the sensor does not receive expected messages. In emergency mode, the throttle controller gradually reduces the dispatch rates of the gateway queues until the overloaded sensors stops firing. Then it gradually restores the rates to their original, pre-emergency mode settings. While restoring the rates, the throttle controller ensures that rate directives from ARFM controller are never exceeded, thus preserving the integrity of throttling decisions made by different controllers. Working together, these components can properly limit incoming requests.

Multiple sensors detect emergency conditions, resulting in the throttle controller going into emergency mode. Each sensor can be in one of two states: fired or unfired. During an emergency, there are two phases for the throttle controller: `emergency_throttle` and `emergency_unthrottle`. During the `emergency_throttle` phase, the throttle reduces all queue rates as long as one of the sensors still fires. In the `emergency_unthrottle` phase, all the sensors return to the unfired state and gradually restore all queue rates to their original values they had before entering emergency mode.

Emergency throttling is disabled by default. Enable emergency throttling only if IBM Support instructs you to do so. ARFM4998W messages might still be displayed in the log if an emergency condition is detected, however this condition does not throttle traffic. You can enable emergency throttling by adding the following entry to the `WAS_HOME/profiles/node/properties/arfm.cfg` file on the ODR host.

```
EnableEmergencyThrottling=true
```

Enforcing rate directives from ARFM controller (initiated by eWLM) is enabled by default. You can disable it by adding the following entry to the `arfm.cfg` file.

```
EnableExternalThrottling=false
```

See the following list for other configuration parameters that you can add to the `arfm.cfg` file.

- `EmergencyRateChangeStep=x` where `x` is an integer in the range 0 - 100, specifying the percentage change in rate in each step of the gradual reduction/increase of throttle rate. The default value is 20.
- `EmergencyRateChangeInterval=x` where `x` is the time between two successive rate change steps in emergency mode in milliseconds. The default value is 15000.
- `EmergencyBlackoutMultiplier=x` where `x` is a multiplier multiplied by different normal message cycles used as input to emergency blackout sensor. The `EmergencyBlackoutMultiplier` parameter is a configuration parameter which tells the sensor indirectly how long to wait before firing. This interval is determined as the product (multiplication) of the parameter and the normal anticipated interval between successive messages. The default value is 2.
- `EmergencyCPUUtilLimit=x` where `x` is an integer in the range of 0 -100, specifying the processor utilization watermark on backend nodes, that triggers emergency throttling. The default value is 100.
- `TokenBucketSizeMillis=x` where `x` is the number of tokens that can be accumulated in the token bucket queue. The default value is 1000.

Memory overload protection:

Memory overload protection limits the rate at which the on demand router (ODR) forwards traffic in order to prevent an out of memory exception from occurring in an application server. If traffic without server affinity arrives at the ODR and the rate for all potential servers has been exceeded, the traffic is rejected. Memory overload protection does not reject traffic that has server affinity. For example, HTTP requests with session affinity or SIP in-dialog messages.

To protect against memory overload, memory overload protection must initially discover the maximum rate, that is, calls per second, that can be sustained without exceeding the maximum percentage of the maximum heap size. As it is discovering the maximum rate, memory overload protection slowly allows more traffic through without affinity, but will reject the remainder. Initially, a potentially large number of HTTP requests or SIP messages without affinity are rejected with a 503 (unless the error code is changed). Intelligent Management persists the maximum rate across server restarts, so it needs to learn the maximum rate once. The maximum rate can change over time due to changes in the session or dialog lifetimes, but these lifetimes generally change relatively slowly and memory overload protection is able to react to such changes. To discover the maximum rate, the product must keep the rate relatively steady for at least an averaging window. The averaging window must be at least as long as the lifetime of most of the HTTP sessions, SIP dialogs, or application sessions. Therefore, the longer the averaging window, the longer it will take to initialize.

For SIP and HTTP, memory overload protection and CPU overload protection are not guaranteed to work if a dynamic cluster is in automatic mode, because of the CPU and heap overhead incurred by replication

WebSphere eXtreme Scale considerations

WebSphere eXtreme Scale might allocate additional memory in a running application server when another application server starts or stops. Memory overload protection does not currently control this memory allocation. So, if the memory utilization is already high, the additional uncontrolled allocation of memory can cause an out of memory exception to occur. For example, if the maximum percentage memory setting is 90% and the current heap utilization is near 90% in application server AS1, and application server AS2 starts or stops, an out of memory exception might occur in AS1 due to replication to AS2. The maximum heap percentage should be set low enough so that there is always enough memory in reserve for the potential replication needs when an application server starts or stops. Memory overload protection will prevent an `OutOfMemoryException` in a dynamic cluster if the maximum percentage memory setting is set to 56%.

Configuring memory overload protection:

Follow these instructions to configure memory overload protection from the administrative console.

Before you begin

Read about configuring the autonomic request flow manager.

About this task

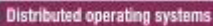




To protect against memory overload, the memory overload protection must initially discover the maximum rate, that is, calls per second, that can be sustained without exceeding the maximum percentage of the maximum heap size. As it is discovering the maximum rate, memory overload protection slowly allows more traffic through without affinity, but will reject the remainder. Initially, a potentially large number of HTTP requests or SIP messages without affinity will be rejected with a 503 (unless the error code is changed). Intelligent Management persists the maximum rate across server restarts, so it must discover the maximum rate once. The maximum rate can change over time due to changes in the session or dialog lifetimes, but these lifetimes generally change relatively slowly and memory overload protection is able to react to such changes. When discovering the maximum rate, Intelligent Management must keep the rate relatively steady for at least an averaging window. The averaging window must be at least as long as the lifetime of most of the HTTP sessions, SIP dialogs, or application sessions. Therefore, the longer the averaging window, the longer it will take to start.

Memory overload protection is disabled by default.

gotcha: To ensure the memory overload protection to be configured properly, the `-agentlib` parameter needs to be set in the **Generic JVM arguments** field.

To enable, follow these steps from the administrative console:

Procedure

1. Expand **Operational policies > Autonomic controllers > Autonomic managers > Autonomic request flow manager**.
2. Type a value less than 100 in the **Memory overload protection: Maximum percentage of the WebSphere Application Server maximum heap size to use** field.
3.  Expand **All servers > odr_name > Process definition > Java Virtual Machine**.
4.  Expand **All servers > odr_name > Process definition > Control > Java Virtual Machine**.
5.  Expand **All servers > odr_name > Process definition > Servant > Java Virtual Machine**.
6.  Set **Generic JVM arguments** `-agentlib:HeapDetect` for 32-bit platforms or `-agentlib:HeapDetect64` for 64-bit platforms.
7.  Set **Generic JVM arguments** for both **Control** and **Servant** `-agentlib:HeapDetect` for 31-bit platforms or `-agentlib:HeapDetect64` for 64-bit platforms.

Trace settings for autonomic request flow manager and application placement:

To troubleshoot the autonomic request flow manager and application placement, you can enable diagnostic trace.

To enable diagnostic trace for the autonomic request flow manager, run the `arfmMustgather.py` script located in the `app_server_root/bin` directory :

```
./wsadmin.sh -f arfmMustgather.py
```

To disable the trace, run:

```
./wsadmin.sh -f arfmMustgather.py disable
```

For information about troubleshooting the mustGather script, see the IBM Support Portal.

Request prioritization problems:

Occasionally, you might encounter flow prioritization behavior that is unexpected. You can look for some common things when request flow prioritization is not working as expected.

HTTP requests are all discretionary

If your environment treats all of the incoming requests equally, you might not have a service policy defined and applied to the proper application module Uniform Resource Identifiers (URI). A best effort approach, also called *discretionary*, is the default policy. Take the following actions to ensure that the service policy is configured and applied:

Table 29. Service policy configurations. The following table outlines the actions required to ensure the service policy is configured correctly.

Action	Performed by
Verify that your service policies are created.	From the administrative console, click Operational policies > Service policies . All of the currently defined service policies display. If you do not see your service policy listed, configure a new service policy by clicking New .
Verify that your service policies are applied to the proper application URIs.	From the administrative console, click Operational policies > Service policies > Select an existing service policy . Verify that the assigned transaction classes are in the transaction classes field. You can create a transaction class by clicking New . If you do not see the transaction class members you are looking for, check that they are not already assigned to another service policy. Also, ensure that the application you are applying a service policy to is deployed in your environment.

Differentiation of a service policy not occurring

Differentiation in response time becomes apparent as the node group approaches its maximum CPU usage threshold, which occurs when all of the nodes in the node group are fully used. In a dynamic environment, the application placement controller can be configured to start more application instances to keep up with the work requests and reduce the load on individual servers.

CPU use remains at 100% on one or more back-end nodes

ARFM is continuously computing the load of each transaction class on the system and continuously optimizes the load distribution. To ensure that the ARFM is optimized, allow ARFM to gather load information for a period of time so that it can fine-tune itself.

Verify that you are consistently grouping transaction classes. For example, avoid grouping URIs with long service times in the same transaction class as URIs with low service times. Mixing requests with widely varying demands in the same transaction class causes the ARFM to produce inaccurate estimates. To modify your transaction classes, click **Operational policies > Service policies > Select an existing service policy** and verify the transaction class field to ensure consistent groupings.

In a heterogeneous cell, not all the nodes in the cell are used equally

The system is working as designed. The load balancer tries to equalize response time on all the back-end nodes in a cluster. If one node is less powerful than another, then the load balancer might distribute less work to the less powerful node so that response time can be similar to the response time from a faster node.

Configuring dynamic application placement

Dynamic application placement is the process by which the features of the Intelligent Management dynamic operations work together to start and stop application instances to meet the fluctuating demand of work requests of varying service policy definitions. This set of actions is controlled by the application placement controller.

Before you begin

- Before configuring application placement, create an on demand router (ODR) and configure the autonomic request flow manager. For more information about configuration, read about configuring the autonomic request flow manager.
- Create a service policy. For more information, read about defining a service policy.
- Verify that you have the correct administrative role. If you want to change the configuration of the application placement controller, you must have a Configurator or Administrator administrative role. If you want to change runtime settings, you must have an Operator or Administrator administrative role.

About this task

Dynamic clusters are the deployment target for your applications. When you create a dynamic cluster, you define a minimum and maximum number of application instances, or cluster members. The application placement controller works to keep these numbers of instances available along with working to meet the demand of your defined service policies. For more information about dynamic clusters, read about creating dynamic clusters.

Procedure

1. Enable or disable the application placement controller and adjust other settings such as the minimum amount of time between placement changes, the server operation timeout, and so on. For more information about enabling, disabling and adjusting settings, read about monitoring and tuning the application placement controller.
2. Configure and manage multi-cell performance in your environment to avoid overprovisioning resources, such as CPU and memory utilization. For more information, read about configuring multi-cell performance management.
3. enable the elasticity mode to add logic that causes the application placement controller to minimize the number of nodes that are used, as well as remove nodes that are not needed, while still meeting service policy goals. For more information about the elasticity mode, read about configuring elasticity mode.

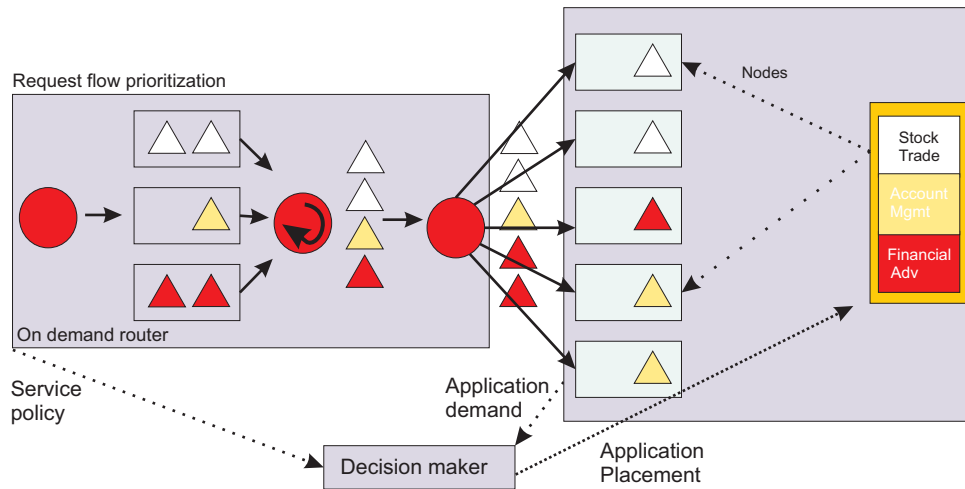
Overview of application placement:

Dynamic application placement for Internet InterORB Protocol (IIOP), Java Message Service (JMS), and HTTP request types is achieved through the cooperation of various product components. These components include dynamic clusters, the on demand router (ODR) and associated autonomic managers, and the application placement controller. The fluctuating volume of work requests for applications is dynamically met on the application server side by the application placement feature, which ensures the integrity of the defined business goals.

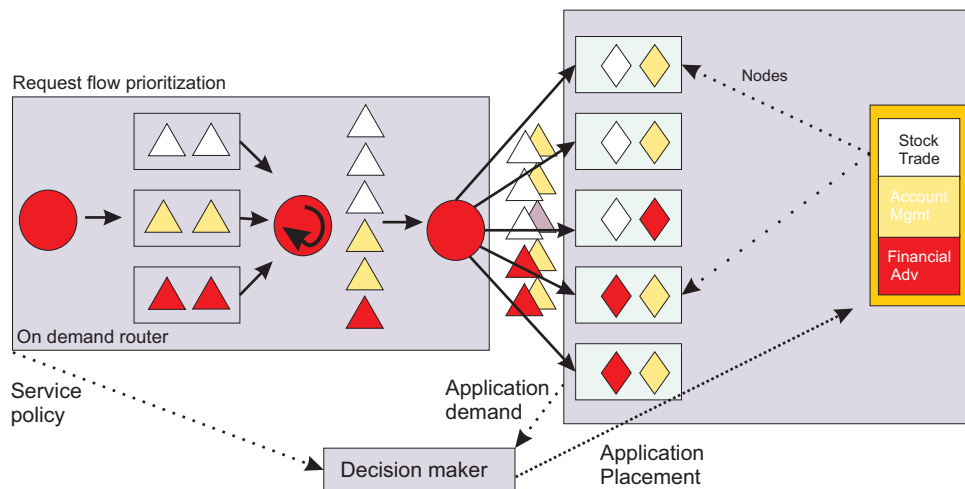
Application placement is a natural extension to the request flow prioritization feature. While the ODR and its associated autonomic managers ensure that the work flows appropriately according to the defined policy, the application placement feature ensures that the applications and the nodes that they run on are kept at appropriate levels to support the influx of work. In times of less work flow, the application instances that run within the resource pool are kept at a minimum. In times of significant work flow, the application instances that run are increased to keep pace with the requests.

The following image shows the application placement process for a stock trading firm in times of less demanding workflow; that is, the request flow is during an off-peak hour. The ODR and its managers are

controlling the incoming request flow prioritization. The decision making unit of the application placement function is made aware of the service policy for the incoming Universal Resource Identifier (URI) requests, as well as the demand level for the given application server resources. As a result, the application instances that run are comfortable for the given demand.



The following image shows a sudden increase in incoming URI requests occurs. For example, it is now lunch time and all online trade customers are now free to access the Web for personal use. The sudden work request fluctuation, which in other environments requires manual intervention to address, is handled by the autonomic features of dynamic operations.



During times of more robust request flow, the application placement function must ensure that the business goals created for the many application URIs are met. As a result, the autonomic managers of dynamic operations must work together to bring a balance to the environment. You can see this balance achieved in the previous graphic. The work flow fluctuation is met by increasing the application instances that run on the available nodes, while balancing the service policy definitions. When the peak declines, the application placement function reduces the running number of application instances accordingly.

The functionality of dynamic application placement requires fundamental Intelligent Management components. Application placement requires dynamic clusters and service policies. Dynamic clusters are application deployment targets. Service policies are performance goals that are assigned to a specific application URI.

These components, when integrated with the autonomic managers, support dynamic application placement.

The dynamic application placement function with the job scheduler is supported. The application placement controller, along with the scheduler and autonomic request flow manager, provides overload protection of servers as long as both online and batch workloads are on dynamic clusters. This overload protection is not supported for static cluster members. Because batch jobs can consume a lot of processor capacity and run for a long period of time, the utilization limit might be exceeded.

The application placement controller is consulted by the job scheduler during its endpoint selection process. You can configure the `UseAPCEndpointSelection` custom property on the job scheduler to `false` to disable the application placement controller and job scheduler integration. Using this custom property to prevents the job scheduler from asking the application placement controller to choose an endpoint. The particular endpoint is chosen by the job scheduler when the custom property is set to `false`.

Dynamic application placement:

Dynamic application placement is based on load distribution, service policy, and available resources. You can run applications at various levels within a Intelligent Management environment, ranging from changing relative weights of applications to expanding the Intelligent Management cell. As changes become more drastic and their impacts become more extreme, applications take more time to run.

By dynamically placing resources, Intelligent Management allows you to utilize your hardware more efficiently. It is unlikely that all applications are in high demand at the same time, assuming a varied assortment of deployed applications. Intelligent Management exploits this situation by supporting resource allocation where needed, thereby increasing the utilization of hardware. The result is that an enterprise no longer requires enough hardware to satisfy each application maximum load simultaneously, which can translate to a significant reduction of requisite IT investments.

The dynamic application placement capability is contained by three autonomic managers: the application placement controller, dynamic workload manager (DWLM), and autonomic request flow manager (ARFM).

Each autonomic manager provides capabilities that work to achieve a common goal, which is to allocate the available capacity among the deployed applications and configured services classes. When it is necessary to modify the capacity that is allocated to a given application, the autonomic manager takes actions starting with the first or fastest level. As changing demand renders these lightweight adjustments ineffective, more drastic measures are required. For example, Intelligent Management autonomic managers might start by changing the dispatching priorities of requests that are associated with a service policy. Intelligent Management makes such adjustments every few seconds. However, such adjustments might be ineffective if the size of a given cluster is too small. Intelligent Management changes the sizes of dynamic clusters within tens of minutes.

The challenge of dynamically placing a given application includes increasing and decreasing its capacity, which can be accomplished at several levels. The decisions about the resource allocation changes are made using the autonomic managers, which monitor various performance metrics, analyze the monitored data, and then guide the planned actions. Intelligent Management supports various configurable levels of autonomicity. In the most autonomic mode, autonomic managers are granted the freedom to run the actions they plan. In the most manual mode, autonomic managers suggest actions, which an administrator must run manually.

Configuring application placement for cells that share the same nodes:

When two cells share the same workstation, such as the same physical machine, the same blade in a blade center, and so on, overuse of memory can occur. You can configure a custom property so that the application placement controller does not overuse resources on the shared hosts.

Before you begin

Configure the cells and create the nodes within each cell.

About this task

For application placement controller to properly use resources on the cells that share the same nodes, you can define the `OverlappingCells` custom property in each cell to give information about the hosts that are sharing the same node.

For example, if you have a `cell_1` cell and a `cell_2` cell, you can create this custom property in `cell_1` using information about `cell_2`, and create the custom property in `cell_2` using information about `cell_1`.

Procedure

1. Enable communication between cells that have security enabled.
If your cells have security enabled, such as Lightweight Directory Access Protocol (LDAP) administration security, you must set up communication between the cells.
2. Optional: If you do not want the password for the deployment manager cells to display as plain text in the administrative console, you can encode the passwords for each of your cells. To encode a password, use the `encodePassword` script. The script is in the `app_server_root\bin` directory. To run the script, use the following command:

```
./encodePassword.sh my_password
```

The encoded password is displayed. You can use this encoded password when you define the custom property value.

3. Set the `OverlappingCells` custom property in each cell.
 - a. In the administrative console, click **Operational policies > Autonomic managers > Application placement controller > Custom properties > New**.
 - b. In the **Name**, type `OverlappingCells`.
 - c. Enter the custom property value. Use the following format, where each one of these variables is referring to the information for the other cell:

```
user_name:password@deployment_manager_host:port.SOAP
```

For example, you might enter: `user1:user1pwd@mydmgrhost.mycompany.com:8879.SOAP` in the value field. .

If you are connecting three or more cells, you might enter multiple values by separating the values with a comma, for example: `user1:user1pwd@mydmgrhost.mycompany.com:8879.SOAP, user2:user2pwd@mydmgrhost2.mycompany.com:8879.SOAP`

4. Save and synchronize your changes.

Results

When the custom property is configured in two or more cells, the application placement controller for each cell contacts the application placement controllers in the other cells to ensure that only one application placement controller is running. By having one application placement controller among all the connected cells running at a single point in time, the application placement controllers do not try to manipulate the same node.

Configuring multi-cell performance management:

Configure and manage multi-cell performance in your environment to avoid overprovisioning resources, such as CPU and memory utilization. You can configure multi-cell performance management in a star topology either automatically or manually.

Before you begin

Create and configure the cells that communicate with each other.

About this task

When you configure multi-cell performance management, multiple cells are managed as a single unit, because the cells share common resources. From a functional perspective, performance management, such as dynamic clusters and overload protection, operates on multiple cells of the same hardware in the same way that performance management operates in a single cell. This feature applies directly to the application placement controller to allow management, through dynamic placement and elasticity, throughout the topology.

A star topology applies to server virtualized environments, such as AIX LPARs/WPARs, Linux on System z®, VMware, and Solaris Zones, as well as non-virtualized environments in which multiple cells share the physical hardware. In a star topology, a single cell is designated as the center cell while the other cells are designated as the point cells. The center cell can perform work, such as contain ODRs and application servers, and only the center cell makes autonomic decisions to start or stop application servers.

The following procedure describes a sample scenario in which multi-cell performance management is configured in a star topology environment so that work requests can be routed from an ODR to dynamic cluster members across cells. The ODR is installed and running on CellA, which is the center cell. The two-point cells, CellB and CellC, contain the dynamic clusters and applications.

gotcha: Complete either the automatic step or the manual step to configure multi-cell performance management. Do not complete both steps.

Procedure

- Automatically configure multi-cell performance management.
 1. Set the following custom property to designate CellA as the center cell.
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as CenterCell.
 - c. Specify the value of the custom property as true.
 - d. Click **OK**, and save and synchronize your changes.

gotcha: Set only one custom property to true.
 2. Set the following custom property individually for each cell that you want to designate as a point cell. In this sample scenario, complete steps 2a through 2c to designate CellB as one-point cell. Next, repeat steps 2a through 2c to designate CellC as another point cell.
 - a. In the administrative console, select **System administration > Cell > Custom properties > New**.
 - b. Specify the name of the custom property as CenterCell.
 - c. Specify the value of the custom property as false.
 - d. Click **OK**, and save and synchronize your changes.
 3. Run the **linkCells.sh** script from CellA to configure the overlay communication between the cells. The **linkCells.sh** script is in the `install_root/bin` directory. Run the script to link the center cell with each point cell individually. Do not run the script to link the center cell to multiple point cells.

Note: Run the **linkCells.sh** script from the cell with the oldest product version.

```
./linkCells.sh CellA_deployment_manager_host:CellA_soap_port:user_id:password CellB_deployment_manager_host:  
CellB_soap_port:user_id:password
```

```
./linkCells.sh CellA_deployment_manager_host:CellA_soap_port:user_id:password CellC_deployment_manager_host:
CellC_soap_port:user_id:password
```

- Manually configure multi-cell performance management.

1. Set the following custom property to designate CellA as the center cell.

- a. In the administrative console, select **System administration > Cell > Custom properties > New**.
- b. Specify the name of the custom property as CenterCell.
- c. Specify the value of the custom property as true.
- d. Click **OK**, and save and synchronize your changes.

gotcha: Set only one custom property to true.

2. Set the following custom property individually for each cell that you want to designate as a point cell. In this sample scenario, complete steps 2a through 2c to designate CellB as one-point cell. Next, repeat steps 2a through 2c to designate CellC as another point cell.

- a. In the administrative console, select **System administration > Cell > Custom properties > New**.
- b. Specify the name of the custom property as CenterCell.
- c. Specify the value of the custom property as false.
- d. Click **OK**, and save and synchronize your changes.

3. Configure the overlay communication between the cells. Run the **importOverlayConfig.py** script, and specify the **overlaynodes.config** file of each cell. The **overlaynodes.config** file is in the **WAS_HOME/profiles/deployment_manager_profile_name/config/cells/cell_name** directory of each cell.

- Run the **importOverlayConfig.py** script from the deployment manager on CellA to link the overlay anchor transports of CellA to CellB. Specify the path to the **overlaynodes.config** file of CellB.

```
WAS_HOME/bin/wsadmin.sh -profileName profile_name -f importOverlayConfig.py
link path_to_CellB_overlaynodes.config_file
```

- Run the **importOverlayConfig.py** script from the deployment manager on CellA to link the overlay anchor transports of CellA to CellC. Specify the path to the **overlaynodes.config** file of CellC.

```
WAS_HOME/bin/wsadmin.sh -profileName profile_name -f importOverlayConfig.py
link path_to_CellC_overlaynodes.config_file
```

- Run the **importOverlayConfig.py** script from the deployment manager on CellB to link the overlay anchor transports of CellB to CellA. Specify the path to the **overlaynodes.config** file of CellA.

```
WAS_HOME/bin/wsadmin.sh -profileName profile_name -f importOverlayConfig.py
link path_to_CellA_overlaynodes.config_file
```

- Run the **importOverlayConfig.py** script from the deployment manager on CellC to link the overlay anchor transports of CellC to CellA. Specify the path to the **overlaynodes.config** file of CellA.

```
WAS_HOME/bin/wsadmin.sh -profileName profile_name -f importOverlayConfig.py
link path_to_CellA_overlaynodes.config_file
```

4. Exchange the root certificates between the cells. Complete the following steps multiple times to extract and copy the root certificate of CellA to each point cell, and to extract and copy the root certificate of each point cell to CellA.

- a. In the administrative console, select **Security > SSL certificate and key management > Key stores and certificates**.
- b. Select **Root certificates keystore** from the **Keystore usages** menu. A new panel is displayed. Click **DmgrDefaultRootStore**.
- c. Select **Personal certificates** from the list of **Additional properties**, select the root certificate, and click **Extract**.

- d. Specify the fully qualified path to where the file is stored on the deployment manager in the **Certificate file name** field, for example `/tmp/myrootcertificate`. Click **OK**.
 - e. Save and synchronize your changes.
 - f. Copy the root certificate to the deployment manager of the target cell.
5. Specify the location of each root certificate between the cells. Complete the following steps multiple times to specify the location of the root certificate of CellA to each point cell, and to specify the location of the root certificate of each point cell to CellA. In this example, the location of the root certificate of CellA is specified in the administrative console of CellB.
 - a. In the administrative console of CellB, select **Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore**.
 - b. Select **Signer certificates** from the list of **Additional properties**. A new panel is displayed. Click **Add**.
 - c. Specify the location of the root certificate on the deployment manager of CellA, and click **OK**.
 - d. Save and synchronize your changes.

Results

You configured multi-cell performance management for your environment so that you can manage your cells as a single unit. You did the configuration by using either the automatic step or the manual step.

Manually disabling communication between multiple cells:

You can complete a set of steps to manually disable the overlay communication between multiple cells.

About this task

Typically, you enable the overlay communication between multiple cells when you configure multi-cell performance management in a star topology. Each cell contains a `overlaynodes.config` file, and the files must be copied between the cells to enable multi-cell performance management. You can manually disable the overlay communication by completing the steps in this topic. As an alternative, you can run the `unlinkCells` script, which is described in another topic, to disable the overlay communication.

For example, assume that you set up a star topology in which three cells are running, and that you want to disable communication between certain cells without disrupting service to the other cells in the environment. CellA is the center cell, and CellB, and CellC are the point cells. Complete the following procedure to manually disable communication between CellA and CellB while maintaining communication between CellA and CellC.

Procedure

1. Copy the `overlaynodes.config` file from the deployment manager of CellB to a deployment manager temporary directory on CellA. For example, copy the file to the `/tmp/CellB_overlaynodes.config` directory. The `overlaynodes.config` file is in the `WAS_HOME/profiles/deployment_manager_profile_name/config/cells/cell_name` directory of each cell.
2. Run the `importOverlayConfig.py` script from the deployment manager on CellA. Specify the path to the temporary directory in which you saved the `overlaynodes.config` file of CellB.

```
WAS_HOME/bin/wsadmin.sh -lang jython -f ./importOverlayConfig.py unlink /tmp/CellB_overlaynodes.config
```

3. Copy the `overlaynodes.config` file from the deployment manager of CellA to a deployment manager temporary directory on CellB. For example, copy the file to the `/tmp/CellA_overlaynodes.config` directory.

4. Run the `importOverlayConfig.py` script from the deployment manager on CellB. Specify the path to the temporary directory in which you saved the `overlaynodes.config` file of CellA.

```
WAS_HOME/bin/wsadmin.sh -lang jython -f ./importOverlayConfig.py unlink /tmp/CellA_overlaynodes.config
```

5. Optional: If security is enabled, you might need to delete the security certificates that were exchanged when the cells were linked together.
 - a. Log on to the administrative console of CellA, and delete the security certificate of CellB. Click **Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore**.
 - b. Select **Signer certificates** from the list of **Additional properties**. A new panel is displayed.
 - c. Select the signer certificate for the remote cell, and click **Delete**.
 - d. Save and synchronize your changes.
 - e. Log on to the administrative console of CellB, and repeat the steps to delete the security certificate of CellA.
6. Delete the CenterCell custom property from each cell.
 - a. In the administrative console of CellA, click **System administration > Cell > Custom properties**.
 - b. Select the CenterCell custom property, and click **Delete**.
 - c. Save and synchronize your changes.
 - d. Log on to the administrative console of CellB, and repeat the steps to delete the CenterCell custom property from CellB.

Configuring elasticity mode:

Configure elasticity mode to allow the application placement controller to minimize the number of used nodes, and to remove nodes that are not needed, while still meeting service policy goals. Additionally, you can configure elasticity mode so that if the controller recognizes that a particular dynamic cluster is not meeting service policies although all possible servers are in use, the controller adds a node.

Before you begin

- For optimal performance, ensure that your dynamic clusters are running in supervised mode or automatic mode. It is not recommended to have elasticity mode enabled when your dynamic clusters are running in manual mode. However, if the dynamic clusters are running in manual mode with elasticity enabled, consider the following items:
 - The application placement controller does not add nodes to dynamic clusters in manual mode.
 - The application placement controller does not remove nodes from dynamic clusters in manual mode when a server is started on the specific nodes.
 - The application placement controller does remove nodes from dynamic clusters in manual mode when a server is not started on the specific nodes.
- It is not recommended to enable elasticity mode when the following option is set in the administrative console for one or more dynamic clusters:

If other dynamic clusters need resources, stop all instances of this cluster during periods of inactivity. If you have elasticity mode enabled and the option set, the application placement controller can remove all of the custom nodes in the cell.

Procedure

1. Set the HAManagedItemPreferred_apc custom property to configure the application placement controller to start on a deployment manager or a node that will not be removed.

Complete the following steps to configure the application placement controller to start on the deployment manager:

 - a. In the administrative console, select **System administration > Deployment manager > Java and process management > Process definition > Java virtual machine > Custom properties**.
 - b. Enter the name of the custom property as HAManagedItemPreferred_apc.
 - c. Set the value of the custom property to true.
 - d. Click **Apply**, and save your changes.

e. Restart the current process in which the application placement controller is running.

Complete the following steps to configure the application placement controller to start on the node:

- a. In the administrative console, select **System administration > Nodes > node_name > node_agent_name > Java and process management > Process definition > Java virtual machine > Custom properties**.
 - b. Enter the name of the custom property as `HAManagedItemPreferred_apc`.
 - c. Set the value of the custom property to `true`.
 - d. Click **Apply**, and save your changes.
 - e. Restart the current process in which the application placement controller is running.
2. Set the `HAManagedItemPreferred_apc` and `HAManagedItemPreferred_cellagent` custom properties to configure the controllers to start on the deployment managers of the center cell and the point cells when you enable elasticity mode in an environment in which multi-cell performance management is configured.

Complete the following steps to configure the application placement controller to start on the deployment manager of the center cell:

- a. In the administrative console, select **System administration > Deployment manager > Java and process management > Process definition > Java virtual machine > Custom properties**.
- b. Enter the name of the custom property as `HAManagedItemPreferred_apc`.
- c. Set the value of the custom property to `true`.
- d. Click **Apply**, and save your changes.
- e. Restart the current process in which the application placement controller is running.

Complete the following steps to configure the cell agent to start on the deployment managers of the point cells:

- a. In the administrative console, select **System administration > Deployment manager > Java and process management > Process definition > Java virtual machine > Custom properties**.
 - b. Enter the name of the custom property as `HAManagedItemPreferred_cellagent`.
 - c. Set the value of the custom property to `true`.
 - d. Click **Apply**, and save your changes.
 - e. Restart the current process in which the application placement controller is running.
3. Enable elasticity mode. Select **Operational policies > Autonomic controllers > Application placement controller**.
- a. Select the **Enable elasticity operation** check box.
 - b. Set elasticity to either supervised or automatic mode.
 - c. Set the timeout value. This value defines the allotted period of time that the application placement controller allows the task to complete. If the task does not complete within the specified amount of time, the application placement controller marks the task as failed. Specify the value in minutes. The default value is 10.
 - d. Click **Apply**, and save your changes.
4. Define the actions of the elasticity operations. An elasticity operation defines the runtime behaviors to monitor, and the corrective actions to take when the behaviors are present. Select **Operational policies > Autonomic controllers > Application placement controller > Elasticity operation > operation..**
- a. To add additional actions to the add operation, click **Add Action...** Select the custom action from the list of custom elasticity operation actions. If no custom actions are defined, select **Operational policies > Autonomic controllers > Application placement controller > Elasticity custom actions > New**.

- b. To add additional actions to the remove operation, click **Add action...**. Select the custom action from the list of custom elasticity operation actions. If no custom actions are defined, select **Operational policies > Autonomic controllers > Application placement controller > Elasticity custom actions > New**.

Elasticity mode:

Use elasticity mode to add logic that causes the application placement controller to minimize the number of nodes that are used, as well as remove nodes that are not needed, while still meeting service policy goals. Additionally, you can use elasticity mode to add logic so that, when the controller recognizes that a particular dynamic cluster is not meeting service policies and has started all possible servers, the controller adds a node.

Overview

Elasticity mode enables a WebSphere cell to grow or shrink dynamically by adding or removing nodes. An elasticity operation defines the runtime behaviors to monitor, and the corrective actions to take when the behaviors are present. As part of the steps for configuring elasticity mode, you create custom actions to define the actions that are associated with the elasticity operations: the add operation and the remove operation. The add operation is issued when all of the resources of the application placement controller are being used, but more resources are still needed to meet the current demand. The remove operation is issued when the application placement controller has an excessive amount of resources.

If the elasticity mode is disabled, dynamic clusters start and stop cluster members in the following situations:

- Servers are started to:
 - Maintain minimum active instances.
 - Meet the CPU or memory demand for a cluster.
- Servers are stopped to:
 - Ensure that the maximum number of instances is not exceeded.
 - Meet the CPU or memory demand for a cluster.
 - Stop cluster members if lazy start or proactive idle stop (custom property) is enabled.
 - Balance resources and make them available to another cluster.

With elasticity enabled, additional options are:

- Increase in demand: Define custom operations (for example, **wasdmn** scripts) to expand a dynamic cluster.

Note: For IBM Workload Deployer or WebSphere Application Server Hypervisor Edition Intelligent Management Pack, predefined tasks add virtual machines and federating nodes to decrease the capacity of a dynamic cluster.

- Decrease in demand: Define custom operations (for example, **wasdmn** scripts) to contract a dynamic cluster.

Note: For IBM Workload Deployer or WebSphere Application Server Hypervisor Edition Intelligent Management Pack, predefined tasks remove virtual machines and federating nodes to decrease the capacity of a dynamic cluster.

Add operation

When elasticity mode is enabled, the application placement controller issues an add operation when all of the members of a dynamic cluster cannot meet the current demand. The controller attempts to consolidate and start all of the servers on the minimum number of nodes as possible.

When the action associated with the add operation is complete, the controller starts a server on the new node. The new node must be added as a member of the dynamic cluster that requested the addition. If a new node is not added, the controller continues to issue the add operation until all required resources are received or the demand decreases.

Remove operation

The remove operation first stops any started dynamic cluster instances before starting the associated actions. It is important to know that if the dynamic cluster is set to manual mode, the remove operation is issued on any nodes that do not have any started application servers. When elasticity mode is enabled and a node is no longer required to meet the current demand, the application placement controller issues a remove operation. Any nodes that are not part of any dynamic clusters with no running application servers are first removed. Next, an attempt is made to remove a node that does contain a dynamic cluster instance as long as the instance is not running and no other application servers are running. Finally, an attempt is made to remove nodes that only have one or more started dynamic clusters. The remove operation occurs only if that node is not required to meet the minimum number of instances for a dynamic cluster, or is not required to meet the current demand.

When the application placement controller is running without elasticity mode enabled, the controller issues start and stop operations for application servers. Servers are started due to an increased demand for CPU or memory, but the servers are not stopped after they are started. When elasticity mode is enabled and the servers are not needed, however, the stop operation is issued and the servers are stopped even after they are started. After all of the servers on the physical machine or virtual machine are stopped, the application placement controller issues the remove operation.

Considerations when using elasticity mode

Consider the following information when using elasticity mode.

- The application placement controller will not issue a remove operation on any node that contains a deployment manager or a stand-alone on demand router (ODR).
- Do not enable application lazy start with elasticity mode. The application placement controller issues the remove operation for all of the nodes on that dynamic cluster. In certain environments, this might cause problems, because all of the custom nodes can then be lost.
- You must configure the application placement controller to always start on the deployment manager or node that will not be removed. Doing so prevents the controller from issuing a remove operation for the node that the controller is active on. If you do not configure the controller to start on the deployment manager, an attempt to remove the node in which the controller is running might occur. As a result, data can be lost, any actions defined by the remove operation do not occur, and the runtime tasks in the administrative console are not properly updated.
- When you use elasticity mode in an environment in which multi-cell performance management is configured, you must configure certain controllers to start on the deployment managers of the center cell and the point cells. Configure the application placement controller to start on the deployment manager of the center cell. Configure the cell agent to start on the deployment managers of the point cells.
- Change the `minTimeBetweenPlacementChange` custom property from 15 minutes to 3 minutes to ensure that the application placement controller does not wait too long to issue an add operation. If the default value of 15 minutes is used, the controller might issue two add operations over a period of 30 minutes.

Considerations when using elasticity to manage JMS traffic that originates from WebSphere MQ

- By default, the application placement controller (APC) uses information produced by the ODR to determine when to start or stop application servers in a dynamic cluster. Set the `APC.predictor` custom property to `CPU` to remove the APC dependency on ODR input. This enables Intelligent Management to support dynamic clusters of message-driven beans when loaded by WebSphere MQ.

- When you use elasticity mode to manage JMS traffic that originates from WebSphere MQ (Version 7.0.1.6 or later is required), go to **System administration > Cell > Custom properties > New**, and set the cell custom property `JMS.CPU` to `true`. Restart the cell.

Monitoring and tuning the application placement controller:

The application placement controller is designed to work with the default settings. However, there can be times when fine tuning the application placement controller becomes necessary to ensure the best results.

Before you begin

Before you can tune application placement, you need to have dynamic application placement running. For more information about enabling application placement, read about configuring dynamic application placement

Depending on your administrative role, you are allowed the following specific privileges when you are configuring the application placement controller:

- **Monitor:** Can view the information.
- **Operator:** Can view the information on the configuration tab. Can change the settings on the runtime tab.
- **Configurator:** Can change the configuration but not the runtime settings.
- **Administrator:** Has all privileges.

About this task

You might tune the application placement controller for the following reasons:

- To adjust the amount of time before runtime tasks expire
- To adjust the amount of time that passes before starting or stopping a server is considered a failure
- To adjust the amount of time between placement changes for applications

Procedure

1. In the administrative console, click **Operational policies > Autonomic managers > Application placement controller**.
2. Determine whether you want to modify the persistent settings in the configuration tab, which are permanent between stops and starts of the application placement controller, or test runtime settings in the runtime tab, which last for the life of the application placement controller. You can choose to modify the runtime settings and then find those settings are actually what you want to use as permanent settings. You can make runtime settings persistent settings by clicking **Save to repository**.
3. Modify the application placement controller settings, as needed.
The table following these steps contains the general configuration settings.
4. Define custom properties to change placement behavior.
You can specify the `reservedMemoryFixed` and `reservedMemoryPercent` custom properties to define how much node memory to reserve for processes that are not related to WebSphere Application Server or Intelligent Management. For information about custom properties that you can set on the application placement controller, read about application placement custom properties.
5. Click **Apply** or **OK**. Save your changes.
6. Iterate and modify the settings as needed. Make changes until you discover the best possible configuration for your environment.

Results

Table 30. General configuration settings

Setting	Description
Enable	<p>Enables or disables the application placement controller. If you disable the application placement controller, you are disabling all autonomic operations for the dynamic clusters. After the placement controller is disabled, no dynamic changes occur with regard to the size and placement of applications on the dynamic cluster. This action is equivalent to turning all dynamic clusters to static clusters.</p>
Approval timeout	<p>Controls how long the runtime task sits in the queue awaiting action before it expires automatically. Acceptable time values are 1 to 60 minutes. This setting is particular to the supervised operating mode. When the operating environment is set to run in supervised mode, the application placement controller creates tasks, but must wait for approval from the system administrator before making changes.</p> <p>The application placement controller treats runtime tasks that are timed out as user-rejected tasks.</p>
Server operation timeout	<p>Represents the amount of time, in minutes, that the application placement controller waits for a start or stop operation to complete before the operation is considered a failure.</p> <p>Set this value to the predicted worst case time to start or stop a server. Acceptable values are between 1 to 60 minutes. If a server fails to start before the timeout, the server is put into maintenance mode. The application placement controller avoids trying to start that server again until you manually fix the start issue and manually start the server. When the server starts, the application placement controller detects that the server has started.</p> <p>Servers are not automatically placed in maintenance mode when the servers fail to start before the server operation timeout. Instead, a runtime task is generated a when the server fails to start. The application placement controller does not attempt to start this server again until the server is successfully started manually. When you successfully start the server manually, the application placement controller receives a notification that the server has started.</p> <p>If you want the servers to be put into maintenance mode, you can set the <code>maintenanceModeOnOperationFail</code> custom property on the application placement controller.</p> <p>If you want to take servers out of maintenance mode after starting successfully, you can specify the <code>unsetMaintenanceModeAfterStart</code> custom property on the application placement controller. For information about custom properties that you can set on the application placement controller, read about application placement custom properties.</p>

Table 30. General configuration settings (continued)

Setting	Description
Minimum time between placement change	<p>Specifies the amount of time that the application placement controller waits before initiating a new batch of changes. The application placement controller might wait for a batch of changes after completing previous changes, or encountering a timeout. Acceptable values can range from 1 minute to 24 hours.</p> <p>When setting this value, consider the overhead that is associated with starting and stopping servers. Starting or stopping servers can take several minutes and might introduce an additional load to the nodes.</p> <p>If you allow the placement controllers to readjust application placement too often, the added overhead negates the increased performance gains that can be earned by readjusting the size of dynamic clusters. For example, if a server takes one minute to start, and the setting for the minimum time between placement changes is 20 minutes, then placement changes have a performance impact of approximately 5%.</p> <p>Set this value at least 20 to 30 times larger than the time necessary to start a server. A value greater than several hours prevents application placement changes from happening more than once a day. If you believe that traffic load and application demands adjust several times during a day, you might want to allow placement changes to occur more frequently.</p>

The application placement controller is tuned for its best performance.

Application placement frequently asked questions:

Occasionally, you might encounter application placement behavior that is not expected. This topic describes some commonly asked questions and things to look for when application placement is not working as you expect.

Where is the application placement controller running?

To find where the application placement controller is running, you can use the administrative console or scripting. To check the location in the administrative console, click **Runtime Operations > Component stability > Core components**. You also run the `checkPlacementLocation.jacl` script to display the server on which the application placement controller is running.

When does the application placement controller start a server?

The application placement controller starts servers for the following reasons:

- To meet the minimum number of application instances that are defined for the dynamic cluster.
- When a request routed through the on demand router for a deactivated dynamic cluster.
- When a dynamic cluster could benefit from additional capacity. The autonomous request flow manager sends a signal indicating how beneficial it would be for a dynamic cluster to have more additional capacity, and additional instances start for the dynamic cluster.

For a view of what is running in view of the application placement controller, see the `SystemOut.log` messages.

When does the application placement controller stop a server?

The application placement controller stops a server for the following reasons:

- A memory constraint on a node exists. The application placement controller understands the minima for the dynamic cluster, or the amount of capacity needed for that dynamic cluster and the processor constraints and memory constraints of the system. If available memory becomes low on a node, the application placement controller attempts to stop instances to try and prevent the node from swapping.
- The dynamic cluster is configured for application lazy start and proactive idle stop, and no demand exists for the dynamic cluster. If the dynamic cluster has no demand, the application placement controller attempts to stop instances of that cluster to eliminate the resource consumption of an inactive dynamic cluster.

Why did the application placement controller not start a server?

The application placement controller does not display that the server is started for one of the following reasons:

- The configuration did not enable dynamic application placement:
 1. Verify that the placement controller is enabled. In the administrative console, click **Operational policies > Autonomic managers > Application placement controller**.
 2. Verify that the subject cluster or clusters are dynamic clusters. The application placement controller acts upon dynamic clusters only. In the administrative console, click **Servers > Dynamic clusters**. Check that the **Operational mode** field for each of the subject clusters is **Automatic**. If not, select the dynamic clusters and click **Automatic**. After you select automatic for your dynamic clusters, click **Set mode**.
 3. Verify that the configured minimum time between placement change parameters is not set too high. In the administrative console, click **Operational policies > Autonomic managers > Application placement controller**. Set the value in the **Minimum time between placement changes** field to a suitable value. Acceptable values range from 1 minute to 24 hours.

- Server operation timeout value set too low.

Sometimes the application placement controller does not start a server because the server operation times out. You can configure the amount of time before a timeout occurs in the administrative console. Click **Operational policies > Autonomic managers > Application placement controller**. Edit the **Server operation timeout** field. If your cell is large, your system is slow, or your system is under high workload, set this field to a higher value. This value represents the amount of time for each server to start, but the timeout occurs based on the number of servers in your cell. For example, if you have five servers and set the value to 10 minutes, then a timeout occurs after 50 minutes.

- Not enough memory available:

- You can diagnose when not enough memory is available by looking at the failed starts in the `SystemOut.log` file.

- The application placement controller uses the following formulas to calculate the memory consumption of a dynamic cluster member:

- If no other dynamic cluster instances are running (cold start):

Server memory consumption = $1.2 * \text{maxHeapSize} + 64 \text{ MB}$

- If other dynamic cluster instances are running, the application placement controller memory profiler uses the following formula:

Server memory consumption = $.667 * \text{resident memory size} + .333 * \text{virtual memory size}$

- Memory profiles are not persisted when the application placement controller restarts.

If you want to debug, you can disable the application placement controller memory profiler by setting the `memoryProfile.isDisabled` custom property to `true`.

Viewing failed start information

Remember: The failed start list is not persisted when the application placement controller restarts or moves between nodes.

You can view failed start information with one of the following options:

- Use the `PlacementControllerProcs.jacl` script to query failed server operations.

Run the following command:

```
./wsadmin.sh -profile PlacementControllerProcs.jacl -c "anyFailedServerOperations"
```

- Use commands in the `wsadmin` tool to display failed starts.

For example, you might run the following commands:

```
wsadmin>apc = AdminControl.queryNames('WebSphere:type=PlacementControllerMBean,process=dmgr,*')
wsadmin>print AdminControl.invoke(apc,'anyFailedServerOperations')
```

When the server becomes available, the failed to start flag is removed. You can use the following `wsadmin` tool command to list the servers that have the failed to start flag enabled:

```
wsadmin>print AdminControl.invoke(apc,'anyFailedServerOperations') OpsManTestCell/xdblade09b09/DC1_xdblade09b09
```

- View the failed starts in the `SystemOut.log` file.

Why did the application placement controller start more servers than I expected?

More servers can start than expected when network or communication issues prevent the application placement controller from receiving confirmation that a server started. When the application placement controller does not receive confirmation, it might start an additional server.

How do I know when the application placement controller has completed or will complete an action?

You can check the actions of the application placement controller with runtime tasks. To view the runtime tasks, click **System administration > Task management > Runtime tasks**. The list of runtime tasks includes tasks that the application placement controller is completing, and confirmation that changes were made. Each runtime task has a status of succeeded, failed, or unknown. An unknown status means that there was no confirmation either way whether the task was successful.

How does the application placement controller work with VMware? Which hardware virtualization environments are supported?

For more information about how the application placement controller works with VMware and other hardware virtualization environments, read about virtualization and Intelligent Management and supported server virtualization environments.

How can I start or stop a server without interfering with the application placement controller?

If you start or stop a server while the dynamic cluster is in automatic mode, the application placement controller might decide to change your actions. To avoid interfering with the application placement controller when you start or stop a server, put the dynamic cluster into manual mode before you start or stop a server.

In a heterogeneous system (mixed hardware or operating systems), how does the application placement controller pick where to start a server?

The membership policy for a dynamic cluster defines the eligible nodes on which servers can start. From this set of nodes, the application placement controller selects a node on which to start a server by considering system constraints such as available processor and memory capacity. The application placement controller does not determine the server placement based on operating systems.

When my dynamic cluster is under load, when does the application placement controller start another server?

The application placement controller works with the autonomic request flow manager (ARFM) and defined service policies to determine when to start servers. Service policies set the performance and priority maximums for applications and guide the autonomic controllers in traffic shaping and capacity provisioning decisions. Service policy goals indirectly influence the actions that are taken by the application placement controller. The application placement controller provisions more servers based on information from the ARFM about how much capacity is required for the number of concurrent requests that are being serviced by the ARFM queues. This number is determined based on how much capacity each request uses when it is serviced and how many concurrent requests ARFM determines is appropriate. The number of concurrent requests is based on application priority, goal, and so on.

The performance goals that are defined by service policies are not guarantees. Intelligent Management cannot make your application respond faster than its limit. In addition, more capacity is not provisioned if enough capacity is already provisioned to meet the demand, even if the service policy goal is being breached. Intelligent Management can prevent unrealistic service policy goals from introducing instability into the environment.

How does the application placement controller determine the maximum heap size of my server?

You can change the heap size of the server in the dynamic cluster template. For more information, read about modifying the JVM heap size.

Why are the dynamic cluster members not inheriting properties from the template?

You must save dynamic clusters to the master repository before changing the server template. If you have dynamic cluster members that do not inherit the properties from the template, the server template probably incurred changes in an unsaved workspace. To fix this issue, delete the dynamic cluster, then recreate it.

Save your changes to the master repository. You can ensure that your changes are saved to the master repository after clicking **Finish**, by clicking **Save** in the message window. Click **Save** again in the **Save to master configuration** window. Click **Synchronize changes with nodes**.

Why does my dynamic cluster have too few active servers?

If you encounter problems where not enough servers are running in the dynamic cluster, try the following actions:

- When the nodes in the node group are not highly utilized, verify that the service policy is met. At times, the policy might not be defined clearly and although the system is able to meet them, although not to your expectations. To check or change a service policy in the administrative console click **Operational policies > Service policies > Select an existing policy**. Check the goal type, goal value, and importance of the policy, and make any necessary changes.
- When the nodes in the node group are highly utilized, compare the service policy goals of this cluster to service policy goals of other active clusters. If the traffic that belongs to this cluster has lesser importance or looser target service goals relative to the other clusters, it is more likely that the system instantiates fewer servers for this cluster. To check or change a service policy in the administrative console, click **Operational policies > Service policies > Select an existing policy**.
- When the node group seems to have some extra capacity, but your service policies are not met, check the configuration settings on the dynamic cluster. There might be too few instances of the dynamic cluster created as a result of the `maxInstances` policy setting.

In a dynamic cluster environment, why does the application placement controller fail to distribute available servers across nodes?

Dynamic application placement is based on load distribution, service policy, and available resources. When reducing the maximum number of application instances in a dynamic cluster, the application placement controller stops servers on nodes with highest workload until the number of servers is reduced to the set maximum value. If all the nodes are available, the application placement controller selects the first node in the list, and continues with the next node in the list until the maximum number is met.

When reducing the maximum number of application instances in multiple dynamic clusters running on the same nodes, the same process applies: the application placement controller stops servers in each dynamic cluster until the number of servers meets the set maximum for each dynamic cluster. Because all the servers in each dynamic cluster are running on the same nodes, the node selection order for stopping servers is the same for each dynamic cluster.

Note: If at least one node is under load, the application placement controller initiates a more distributed placement solution.

Application placement controller logs:

You can use the log files to troubleshoot issues with the application placement controller, such as the placement of new instances.

apc.log file

The application placement controller logs information regarding application placement decisions the `apc.log` file. You can submit this file when you contact IBM support so that decisions that are made by the application placement controller can be understood.

You can enable the application placement controller logs with the `apc.log.enablePlacementLog` custom property. To change the size and number of log files, edit the `apc.log.LogFileSize` and `apc.log.numHistoricalFiles` custom properties.

SystemOut.log file

You can look for DCPC400 messages in the `SystemOut.log` file of the node that runs the application placement controller. These messages contain information regarding the input and result of a placement cycle. This information can be useful in understanding the decisions made by the placement controller. For example:

```
[7/23/08 10:42:12:086 EDT] 0000006f APCXDMaPHELpe I DCPC400I: Application Placement Controller Input is
Cluster name is dc1:
Cluster's properties: Type=DYNAMIC, Operational Mode=AUTOMATIC, Minimum Instances=2, Maximum Instances=2,
Isolation Preference=None, Vertical Stacking Number=Disabled, Lazy Start Time=Disabled,
Idle Stop Time=Disabled, Current Memory Size=371.2 MB, Maximum Heap Size=256.0 MB
Cluster's Demands: CPU Demand=0.0, Memory Demand=0.0
Cluster's Utility: {UtilityFunctionLinear extantDemand=0.0, utility=1.0}
Cluster's Node Membership: jpcammar01Cell01/elara11Node01, jpcammar01Cell01/elara11Node02,
Cluster's Members: jpcammar01Cell01/elara11Node01/dc1_elara11Node01,
jpcammar01Cell01/elara11Node02/dc1_elara11Node02,
Cluster's Placements: None
[7/23/08 10:42:12:090 EDT] 0000006f APCXDMaPHELpe I DCPC400I: Application Placement Controller Output is
Cluster name is dc1:
Cluster's properties: Type=DYNAMIC, Operational Mode=AUTOMATIC, Minimum Instances=2, Maximum Instances=2,
Isolation Preference=None, Vertical Stacking Number=Disabled, Lazy Start Time=Disabled,
Idle Stop Time=Disabled, Current Memory Size=371.2 MB, Maximum Heap Size=256.0 MB
Cluster's Demands: CPU Demand=0.0, Memory Demand=0.0
Cluster's Utility: {UtilityFunctionLinear extantDemand=0.0, utility=1.0}
Cluster's Node Membership: jpcammar01Cell01/elara11Node01, jpcammar01Cell01/elara11Node02,
```

```
Cluster's Members: jpcammar01Cell01/elara11Node01/dc1_elara11Node01,  
jpcammar01Cell01/elara11Node02/dc1_elara11Node02,  
Cluster's Placements: elara11Node02/dc1_elara11Node02, elara11Node01/dc1_elara11Node01,
```

If you look at the first DCPC400 message, you can see important information about the clusters:

- **Cluster's properties:** Displays cluster configuration information.
- **Cluster's Demands:** Displays the cluster demands on processor and memory.
- **Cluster's Utility:** Displays the cluster capacity calculation.
- **Cluster's Node Membership:** Displays the nodes on which a cluster member can reside.
- **Cluster's Members:** Displays the members of the cluster.
- **Cluster's Placements:** Displays the current placements of the cluster.

By comparing the first DCPC400 message, which is the application placement controller input, to the second DCPC400 message, which is the application placement controller output, you can determine that the application placement controller started instances of the dc1 dynamic cluster on the elara11Node02 node and the elara11Node01 node. This conclusion was determined from the following data:

1. From the **Cluster's properties** value in the first message, the minimum and maximum number of cluster instances is set to 2.
2. From the **Cluster's Placements** value on the first message, you can conclude that no instances are running. Because no instances are running, a breach of the minimum and maximum number of cluster instances has occurred. As a result, two instances are started.
3. From the **Cluster's Placements** value on the second message, you can see that two instances were started: one instance on the elara11Node1 node and one instance on the elara11Node2 node. The instance names are in the format of ***node_name/server_name***.

Overview of IIOp and JMS request flow prioritization

Internet Inter-ORB Protocol (IIOp) and Java Message Service (JMS) request flow prioritization is achieved with the autonomic managers that control the flow of requests, prioritization of requests, and dynamic workload management. Only IIOp requests from a stand-alone Enterprise JavaBeans (EJB) client are handled by Intelligent Management IIOp request flow prioritization. EJB calls from servlets, for example, are not prioritized. This limitation exists because the system cannot prioritize at multiple tiers, such as the Web tier and EJB tier, requests associated with the same overall user request. However, given the asynchronous nature of JMS, no restrictions exist on where the requests originate.

For IIOp and JMS, the back-end application server processes that are hosting the applications run the autonomic request flow manager (ARFM) gateways. These ARFM gateways prioritize request flow. The request flows are managed to achieve the best balanced performance results, considering the configured service policies and the offered load.

With Intelligent Management, you can define performance goals and bind them to specific subsets of the incoming traffic. The ARFM and associated autonomic managers are able to support business goals in times of high load by making smart decisions about the work coming into the application servers. Not all work in your configuration is created equal. The ARFM is able to support this concept by forwarding different flows of requests on for execution more or less quickly to achieve the best balanced result.

The ARFM is aware of its environment because of a component called an on demand configuration (ODC). The ODC automatically gets information about all the Intelligent Management application servers and applications that are deployed in the cell, and any service policy and work classes associated with those configuration artifacts.

Note: The ODC cannot read environments other than a homogeneous Intelligent Management environment.

A service policy is a user-defined categorization that is assigned to potential work as an attribute that is read by the ARFM. For IOP, you can use a service policy to classify requests based on request attributes, including the application name, EJB method name, EJB module name, such as the EJB jar file, and the EJB name. For JMS, you can classify based on destination name, either topics or queues. By configuring service policies, you apply varying levels of importance to the actual work. You can use multiple service policies to deliver differentiated services to different categories of requests. Service policy goals can differ in performance targets and importance.

The ARFM exists in the application server process and controls request prioritization. The autonomic request flow manager contains two parts: a controller and a gateway. The ARFM function is implemented, for each cell, by a controller plus a collection of gateways in the application servers. The gateways intercept and queue the incoming IOP requests, while the controller provides control signals, or directions, to the gateways, and the placement controller. The ARFM also includes the work profiler, which estimates the computational load characteristics of the different flows of requests. Working together, these components can properly prioritize incoming requests.

Dynamic workload management (DWLM) is a feature that applies the same principles as workload management (WLM), such as routing based on a weight system, which establishes a prioritized routing system. DWLM is an optional add-on that adds autonomic setting of the routing weights to WLM. With WLM, manually set static weights in the administrative console. With DWLM, the system can dynamically modify the weights to stay current with the business goals. DWLM can be shut off. If you intend to use the automatic operating modes for the components of dynamic operations, then setting a static WLM weight on any of your dynamic clusters might get in the way of allowing the on demand aspect of the product to function properly. For IOP, these weights are consumed by base WebSphere EJB WLM and factor to where new EJB client requests are directed, as shown in the following diagram:

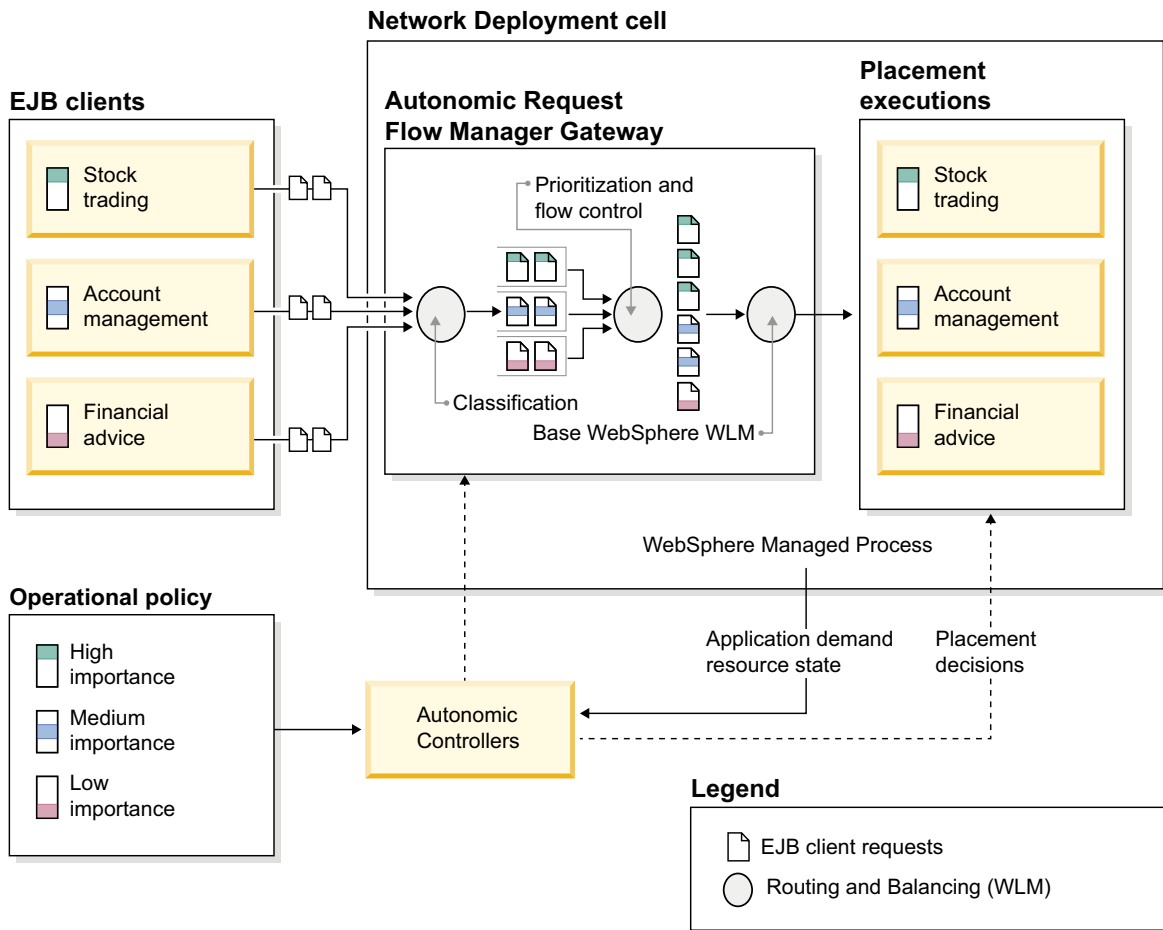


Figure 20. IIOF flow

DWLM does not influence JMS traffic. The destinations shown in the following figure can be running in the same WebSphere Managed process or a different WebSphere Managed process.

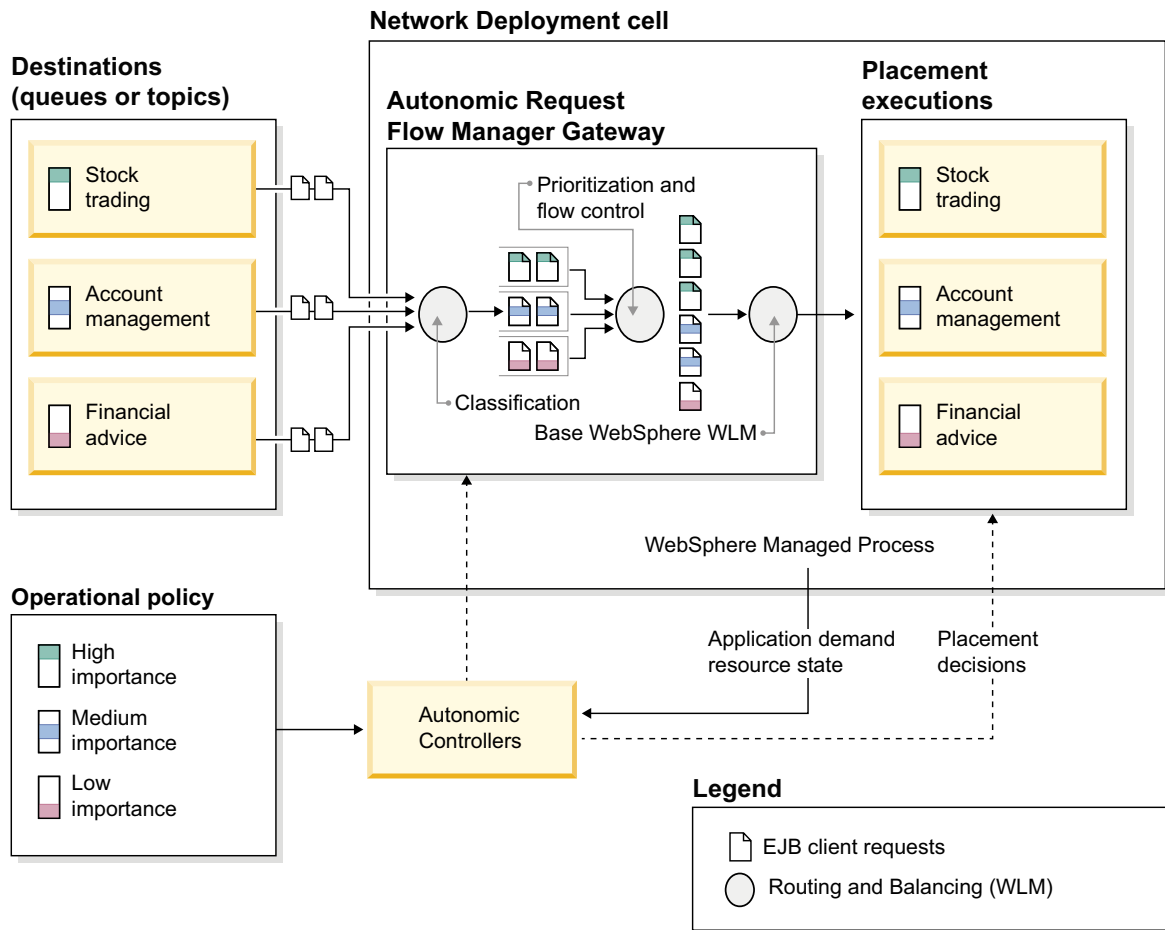


Figure 21. JMS flow

As the previous diagrams depicts, an equal amount of requests flow into the application server, but after the work is categorized, prioritized, and queued, a higher volume of more important Platinum work is sent to be processed while a lower volume of less important Bronze work waits to get queued. However, just because the lower priority work is delayed the most, that does not make the long-term average rate of Bronze work that runs in the application server less than the long-term average rate of Bronze requests that come in. In the end, the features of the dynamic operations attempt to keep the work within the target time allotted for completion.

JMS support in the Intelligent Management environment:

Intelligent Management extends the dynamic operations environment to Java Message Service (JMS) by providing the capability to categorize and prioritize JMS messages. JMS messages can be classified by destination, that is, topic or queue. Service policies can be defined to prioritize messages differently based on the destination name.

While JMS messages can be consumed by standalone clients or enterprise applications in a client container or enterprise applications running in an application server, Intelligent Management *only* supports classification and flow control of messages for enterprise applications inside application server. For JMS, autonomic request flow manager gateways runs in the application server hosting the application, which means that JMS traffic is not handled by the ODR server.

For Message Driven Beans, service policy goals are based on the *service times* of the *onMessage* method. For synchronous JMS message consumers, service time is the interval between time of message delivery to the application and the application returning for the next message on the same consumer session.

Restriction: Routing policies including application edition based routing are not supported for JMS. Classification and flow control of JMS messages are only supported only when using the default message provider of WebSphere Application Server. A given destination is allowed to be part of only one JMS work class.

Optimizing the database tier for performance monitoring

You can configure Intelligent Management to provide real time information about the database tier, including central processing unit utilization, average response time, and throughput. You can access this information through the runtime topology in order to make adjustments when bottlenecks occur.

About this task

Bottlenecks can occur in the database tier and cause a slower response time. If there is a problem in the database tier, Intelligent Management can provide you with the information you need to identify and address the problem.

Procedure

1. Configure the database tier. Read about configuring the database tier.
Follow these steps to set up Intelligent Management for monitoring the database tier.
2. Monitor the database tier. Read about monitoring the database tier.
After configuring the database tier, you can view real time information in order to detect and prevent bottlenecks.

Configuring the database tier:

To access useful information about the database tier, configure Intelligent Management for performance monitoring. With database tier performance monitoring, you can view central processing unit utilization, average response time, and throughput. The information provided in performance monitoring is valuable for detecting and preventing bottlenecks.

Before you begin

Ensure that a DB2® or Oracle database is installed on the node, and the product is installed with an augmented profile.

About this task

After you have installed either a DB2 or Oracle database on the node and installed the product with an augmented profile, you can then configure the database tier for performance monitoring. Monitoring the database tier provides the information needed to prevent the system from experiencing slower response time due to bottlenecks.

Procedure

1. Create a database server in the WebSphere Application Server configuration.
 - a. Create a new generic server and give it a name, for example `DB2Instance1`. Select the database node from the list.
 - b. Click **Generic servers > DB2Instance1**.
 - c. Under **Additional properties**, click **Process definition > Java Virtual Machine > Custom properties**.
 - d. Add the following entries:

```
GENERIC_SERVER_TYPE = DBSERVER
GENERIC_SERVER_SUB_TYPE = DB2 (for DB2) or ORACLE (for Oracle)
INSTANCE_NAME = name of database instance (DB2) or SID (Oracle)
```
2. Configure data sources for Java Database Connectivity (JDBC) monitoring:

- a. Click **Resources > JDBC > JDBC providers > *database_server_name* > Data sources > *data_source_name* > Custom properties.**
Add the property: `dbServerName = generic server name`, for example, `DB2Instance1`.
 - b. Follow these steps, depending on the database type and Java database connectivity (JDBC) driver type you use:
 - For type 2 DB2 data sources. If the database name on the server is different from the database alias name used on the local DB2 client, take these actions. For the **Database name** field on the **Data source configuration** panel, specify the alias name. Under **Additional properties**, click **Custom property**. Add a remote database name custom property, using database name as the value on the database server.
 - For Oracle data sources: If the database name does not display in the connection URL, under **Additional properties**, click **Custom property**. Set the value of the database name property to the name of the Oracle database.
3. Enable performance monitoring infrastructure (PMI) for JDBC statistics collection:
 - a. Click **Monitoring and tuning > Performance Monitoring Infrastructure (PMI)**
This action displays a list of servers in the cell.
 - b. For every server in the list that has a data source:
 - 1) Click the server name.
 - 2) In the **Configuration** panel, select the **Enable Performance Monitoring Infrastructure (PMI)** check box. Set the **Currently monitored statistic set** to **Extended** or **All**. However, if the currently monitored statistic set is set to **Custom**, and for improved performance, enable the JDBC counters in the Custom Level.

Results

Your system is now ready for extended monitoring of Intelligent Management components, including JDBC. You can view your database from the runtime topology.

What to do next

Read about monitoring the database tier for more information.

Monitoring the database tier:

To prevent bottlenecks, Intelligent Management has several tools that provide the information necessary to monitor the database tier. Use the following list of monitoring tools and links to configure Intelligent Management to fit your needs.

Before you begin

Configure the database tier. Read about configuring the database tier for more information.

About this task

Before a bottleneck occurs, Intelligent Management can help you anticipate potential problems and adjust the system accordingly. Customize your system using the following information.

Procedure

- Enable historic logging of statistical information in the environment.
For more information, read about configuring the visualization data service.
- Evaluate historical information by using the data exporter to filter out historical logs or parse logs directly.

For more information, read about configuring the operations reports to log data for reuse in external programs for more information.

- View CPU utilization at the database node level, and view average response time and throughput at the database server level.

For more information, read about creating and managing reports.

- Enable automatic e-mail notifications for new tasks.

For more information, read about defining email notification.

- Validate incoming tasks by viewing the details and the corresponding monitors to determine if the task is valid.

For more information, read about managing runtime tasks.

- Use the topology and charting feature when a problem is detected or to validate a task.

For more information, read about monitoring operations.

Configuring health management

Health management is the process by which Intelligent Management dynamic operations monitor and manage servers to preserve an optimal server environment.

Before you begin

Create the environment that you want to monitor. When you create a health policy, you select members for the policy, so these members must exist before you create the policy.

About this task

Health management is the ability of the system to take a policy-driven approach to monitoring the application server environment, and to take action when certain criteria are discovered.

Procedure

1. Enable health management.
2. Create health policies.
3. Monitor and tune health management.

Enabling and disabling health management:

Health management monitoring is enabled by default. You can enable or disable health management monitoring.

Before you begin

The health monitor uses health policies. A health policy is a combination of health conditions and actions. Health conditions define triggers from which the system can protect itself, for example, a memory leak. Health actions are the specific steps that the system takes when a health condition is triggered. For example, with a memory leak condition, the health action can be to restart the associated servers. A number of predefined health conditions are installed with the product. You can use the predefined health conditions to create default health policies. In addition to these broad default health policies, you can define specific policies that apply for your environment. For more information about the default health policies, read about health management. For more information, read about creating health policies.

Depending on your administrative role, you are allowed specific privileges when configuring the autonomic managers. The following list shows the administrative roles and privileges for configuring the autonomic managers:

Monitor: Can view the information.

Operator: Can view the information on the configuration tab. Can change the settings on the runtime tab.

Configurator: Can change the configuration but not the runtime settings.

Administrator: Has all privileges.

About this task

Health management is enabled by default. Use health management to protect your system from many user application malfunctions, including memory leaks and application hangs. Health management uses health policies to define a set of conditions. Intelligent Management uses the health conditions to monitor the health of the system.

Procedure

1. In the administrative console, click **Operational policies > Autonomic managers > Health controller**.
2. Enable or disable health monitoring. When the check box is selected, the health condition of the environment is monitored. When the check box is not selected, health monitoring is turned off.

Results

When health management is enabled, the system is monitored for conditions that are defined by your health policies.

What to do next

Manage other health monitoring settings. For more information, read about monitoring and tuning health management and about the `HmmControllerProcs.jac1` script.

Health management:

With the health monitoring and management subsystem, you can take a policy-driven approach to monitoring the application server environment and take action when certain criteria are discovered.

Health monitoring and management subsystem

The health management subsystem continuously monitors the state of servers and the work that is performed by the servers in your environment. The health management subsystem consists of two main elements: the *health controller* and *health policies*.

The health controller is the autonomic manager that controls the health monitoring and management subsystem, and acts on your health policies to ensure that certain conditions exist. The health controller is a distributed resource managed by the high availability manager, and exists within all node agent and deployment manager processes. The health controller is active in one of these processes. If the active process fails, the health controller can become active on another node agent or deployment manager process.

The health controller runs on a *control cycle*. The control cycle length defines the amount of time between environment checks initiated by the health controller. At the end of the control cycle, the health controller checks the environment and generates runtime tasks to resolve any breaches in the health conditions.

You define the health policies, which include the *health conditions* that you want to monitor in your environment and the *health actions* to take if these conditions are not met.

You can disable or enable health management using the health controller, while still having multiple health policies defined on the system. You can limit the server restart frequency or prohibit restarts during certain periods.

The health management subsystem functions when Intelligent Management is in automatic or supervised operating mode. When the reaction mode on the policy is set to automatic, the health management system takes action when a health policy violation is detected. In supervised mode, the health management system creates a runtime task that offers one or more reactions. The system administrator can approve or deny the proposed actions.

Health conditions

Health conditions define the variables that you want to monitor in your environment. Several categories of health policy conditions exist. You can choose from the following predefined health conditions:

Age-based condition

Tracks the amount of time that the server is running. If the amount of time exceeds the defined threshold, the health actions run.

Excessive request timeout condition

Specifies a percentage of HTTP requests that can time out. When the percentage of requests exceeds the defined value, the health actions run. The timeout value depends on your environment configuration. For more information about the excessive request timeout health condition, see excessive request timeout health policy target timeout value.

Excessive response time condition

Tracks the amount of time that requests take to complete. If the time exceeds the defined response time threshold, the health actions run.

Attention: Requests that exceed the timeout value configured for the excessive request timeout condition are not counted towards this health condition. For example, if the default timeout value of 60 seconds is being used for the excessive request timeout condition, any request that exceeds 60 seconds does not activate the health actions defined for the excessive response time condition. This restriction applies even if you do not have the excessive request timeout condition defined in your environment.

Memory condition: excessive memory usage

Tracks the memory usage for a member. When the memory usage exceeds a percentage of the heap size for a specified time, health actions run to correct this situation.

Memory condition: memory leak

Tracks consistent downward trends in free memory that is available to a server in the Java heap. When the Java heap approaches the maximum configured size, you can perform either heap dumps or server restarts.

Storm drain condition

Tracks requests that have a significantly decreased response time. This policy relies on change point detection on given time series data.

Workload condition

Specifies a number of requests that are serviced before policy members restart to clean out memory and cache data.

Garbage collection percentage condition

Monitors a Java virtual machine (JVM) or set of JVM's to determine whether they spend more than a defined percentage of time in garbage collection during a specified time period.

For more information about these conditions, click the help icon on the Define health policy general properties panel in the administrative console.

With these predefined health policy conditions, actions have been taken to optimize the distribution of the needed data, minimize the impact of monitoring, and enforce the health policy in your environment.

You can also define custom conditions for your health policy if the predefined health conditions do not fit your needs. You define custom conditions as a subexpression that is tested against metrics in your environment. When you define a custom condition, consider the cost of collecting the data, analyzing the data, and if needed, enforcing the health policy. This cost can increase depending on the amount of traffic and number of servers in your network. Analyze the performance of your custom health conditions before you use them in production.

Example:

```
PMIMetric_FromServerStart$systemModule$cpuUtilization > 90L
```

Health actions

Health actions define the process to use when a health condition is not met. Depending on the conditions that you define, the actions can vary. The following table lists the health actions that are supported in various server environments:

Table 31. Predefined health action support for different server types

Health action	WebSphere application servers that run in the same cell as the health controller	Other middleware servers, including external WebSphere application servers, that run the node agent
Restart server	Supported	Supported
Take thread dumps	Supported for servers that are running on the IBM Software Development Kit	Not supported
Take Java virtual machine (JVM) heap dumps	Supported for servers that are running on the IBM Software Development Kit	Not supported
Put server into maintenance mode	Supported	Supported
Put server into maintenance mode and break HTTP and SIP request affinity to the server	Supported	Supported
Take server out of maintenance mode	Supported	Supported

Note: In a dynamic cluster, a restart can take one of several forms:

- Restart in place (stop server, start server). This restart always occurs when a dynamic cluster is in manual mode.
- Start a server instance on another node, and stop the failing one.
- Stop the failing server only, assuming that the remaining application instances can satisfy demand.

You can also define a custom action. With a custom action, you define an executable file to run when the health condition breaches. You must define custom actions before you create the health policy that contains the custom actions.

Health policy targets

Health policy targets can be a single server, each of the servers in a cluster or dynamic cluster, the on demand router (ODR), or each of the servers in a cell. You can define multiple health policies to monitor the same set of servers.

If you are using predefined health conditions, the support varies depending on the server type. Certain middleware servers do not support all of the policy types. The following table summarizes the health policy support, by server type:

Table 32. Health policy support for different server types

Predefined health policy	WebSphere application servers that run in the same cell as the health controller	Other middleware servers, including external WebSphere application servers, that run the node agent
Age-based policy	Supported	Supported
Workload policy	Supported	Supported
Memory leak detection	Supported	Not supported
Excessive memory usage	Supported	Supported for WebSphere Application Server Community Edition servers. Not supported for other middleware server types.
Excessive request timeout	Supported	Supported for other middleware servers to which the ODR routes requests.
Excessive response time	Supported	Supported
Storm drain detection	Supported	Supported
Garbage collection percentage	Supported	Not supported

Default health policies

You can create default health policies using predefined health conditions installed with the product.

To create a default health policy, click **Operational policies > Health policies > New**, and select one of the predefined health conditions.

Because the default health policies monitor each server in supervised mode, you can use these policies to prevent health problems. In addition to the default policies, you can define policies with more detailed settings or automated mode operation for particular servers or collections of servers. The following list shows the default cell-wide health policies that you can create using the predefined health conditions:

- **Default memory leak:** Default standard detection level. The default memory leak health policy uses the performance advisor function. The performance advisor is enabled when this policy is enabled. To disable the performance advisor, remove this health policy or narrow the membership of the health policy. To preserve the health policy for future use, keep the default memory leak policy, but remove all members. To change the members, click **Operational policies > Health policies > Default_Memory_Leak**. You can edit the health policy memberships by adding and removing members from the policy.
- **Default excessive memory usage:** Set to 95 percent of the JVM heap size for 15 minutes
- **Default excessive request timeout:** Set for 5 percent of the requests timing out
- **Default excessive response time:** Set to 120 seconds
- **Default storm drain:** Default standard detection level
- **Garbage collection percentage:** Set to 10 percent. The default sampling time is 2 minutes.

To view the recommendations made by default health policies and to take actions on these recommendations, click **System administration > Task management > Runtime tasks**.

Excessive request timeout health policy target timeout value:

The excessive request timeout health policy specifies an acceptable percentage of requests that can time out after being routed from the on demand router (ODR). The target timeout value can vary depending on settings in your environment.

About the excessive request timeout health policy

The excessive request timeout health policy defines an acceptable percentage of timed out requests that are routed through the ODR. For each control cycle of the health controller, the number of requests that time out after being routed from the ODR is examined. If the percentage of timed out requests exceeds the configured percentage, runtime tasks are generated to correct the situation.

Default values

By default, the target timeout value for requests is 60 seconds. The default value for the percentage of requests that can time out in the excessive request timeout health policy is 5%. However, setting other properties in your environment can affect the target timeout value that is used.

Settings that affect the timeout value

The following settings in the administrative console affect the target timeout value:

- **ODR outbound request timeout** setting
To change the setting, click **Servers > On demand routers > *on_demand_router_name* > On demand router properties > On demand router settings** and set the **ODR outbound request timeout** value to a number of seconds.
- `goodServiceTimeLimitSpec` custom property
For more information, read about autonomic request flow manager custom properties.
- `timeoutFactor` custom property
For more information, read about autonomic request flow manager custom properties.
- Average response time value that is defined in the service policy
To change the setting, click **Operational policy > Service policy > *service_policy_name***. Specify **Average response time** as the **Goal type** and enter a **Goal value**.

Determining the target timeout value

Depending your configuration, the timeout value used as target value for the excessive request timeout health policy can vary. The following decision tree helps you determine the target timeout value for your environment. The default setting is the path indicated with red arrows, or a target timeout value of 60 seconds, which is the default setting for the ODR outbound request timeout.

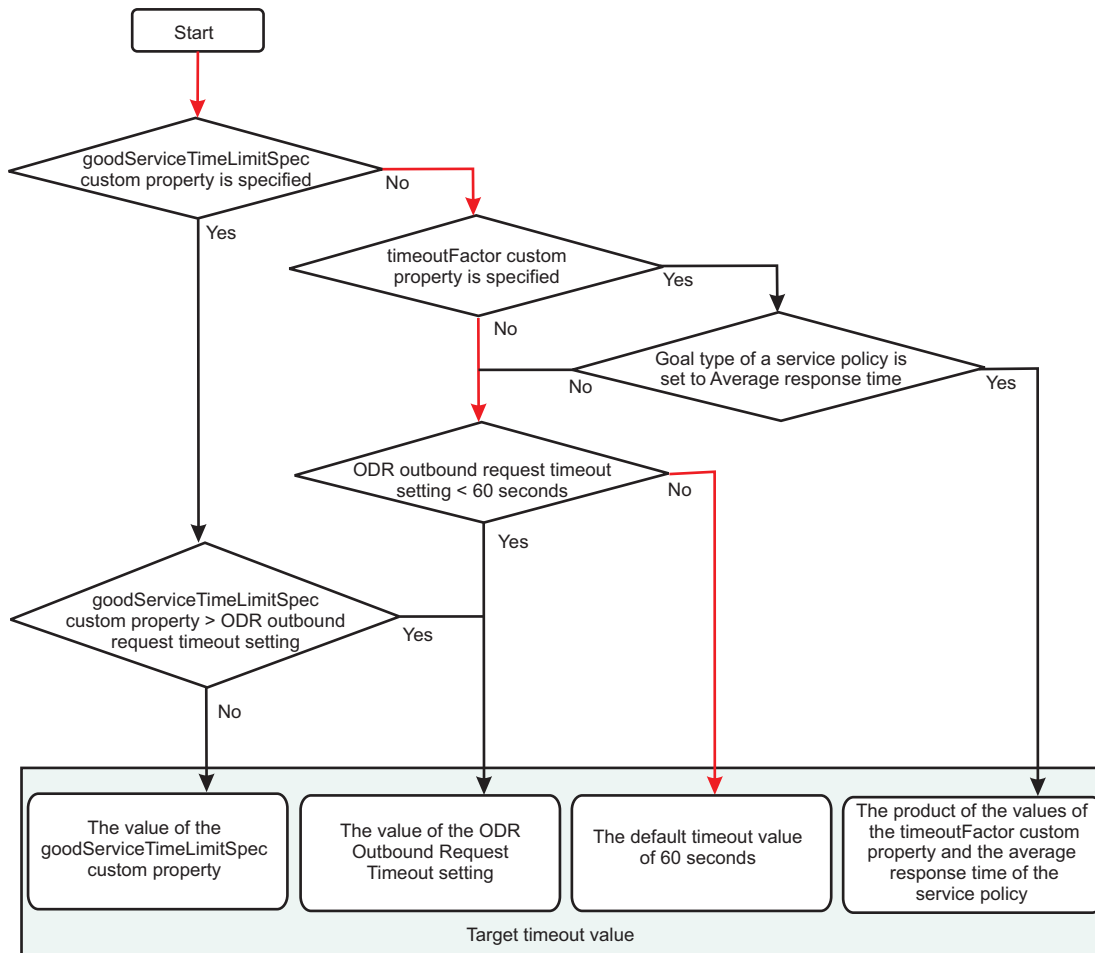


Figure 22. Target timeout value decision tree.

The settings that affect the target timeout value combine together to indicate the target timeout value.

- The value of the `goodServiceTimeLimitSpec` custom property is used as the target timeout value if the `goodServiceTimeLimitSpec` custom property is specified and is larger than the **ODR outbound request timeout** value.
- The value of the **ODR outbound request timeout** setting is used as the target timeout value if the `goodServiceTimeLimitSpec` custom property is not specified, the `timeoutFactor` custom property is not specified, and the **ODR outbound request timeout** setting is set to less than 60 seconds.
- The default value of 60 seconds is used as the target timeout value if the `goodServiceTimeLimitSpec` custom property is not specified, the `timeoutFactor` custom property is not specified, and the **ODR outbound request timeout** setting is set to greater than 60 seconds.
- The timeout value is set to the product of the values of the `timeoutFactor` custom property and the average response time of the service policy if the `goodServiceTimeLimitSpec` custom property is not specified, the `timeoutFactor` custom property is specified, and the goal type of the service policy is set to average response time.

Creating health policies:

A health policy is the definition of specific health criteria. Intelligent Management protects your environment against these criteria. The health management function uses defined policy to identify software malfunctions in the environment.

Before you begin

- To create a health policy, you require configurator or administrator administrative privilege. The health controller must be enabled.

- If you want a *custom action* to run on the target server when the health condition breaches, define the custom action before you create the health policy. For more information, read about creating health policy custom actions

About this task

Health policies work with the health controller to monitor the operation of the servers in your environment. When the health controller detects that your servers are not meeting a defined health policy, you can take action to fix the problem. You can notify the administrator of problems, or Intelligent Management can fix the problems automatically.

Procedure

1. In the administrative console, click **Operational policies > Health policies > New**.
2. Define health condition properties for the health policy.

Remember: The excessive request timeout and storm drain conditions do not apply to Java Message Service (JMS) and Internet Inter-ORB Protocol (IIOP) traffic.

Health policy conditions include the following properties:

- Set properties that pertain to the health condition that you selected. If you chose to create a custom health condition, specify a subexpression that represents the metrics that you are evaluating in your custom condition. For more information about the conditions that you can set, click **Syntax help**.

As a best practice, consider the cost of collecting the data, analyzing the data, and if needed, enforcing the health policy when you define a custom condition. Consider the amount of traffic in your network, especially when you scale out the number of servers that produce data. Before introducing new health policies into the production environment, analyze these aspects of your custom health conditions.

You can further configure your custom health conditions that leverage PMI modules, notably the `webAppModule`s, at finer granularities than the server granularity. For example, you can use the subexpression builder to create a `webAppModule` policy as a starting point, then edit the expression to define a finer granularity:

```
PMIMetric_FromServerStart$webAppModule$SlamSess\#SlamSess.war\webAppModule.servlets\SlamSess\responseTime > 100L
```

In this example, the application name is displayed as `SlamSess` when you list the applications in the administrative console. If you are using an EAR file, specify the Web archive (WAR) file name after the EAR file name. If the WAR is not embedded in an EAR file, specify only the WAR file name. The `SlamSess` value is the servlet name that is listed in the `web.xml` file. The `responseTime` value is the statistic that is listed in the Performance Monitoring Infrastructure (PMI) module definition.

- Choose a reaction mode. **Supervise** mode allows the administrator to approve or reject actions before they are taken.
 - Select the actions to take when the health policy conditions are not met. The available actions depend on the health condition type. These actions can be the existing default actions, or you can define custom actions to run an executable file. A list of actions are displayed in the sequence in which they run when the health condition breaches. You can add and remove steps from this list.
3. Select the members to monitor. Layers of logic can apply to monitored members. For example, you might want to apply a specific health policy to each member of a cluster and to an application server outside of the cluster.
 4. Review and save your health policy.

Results

You created a health policy and applied that policy to a target environment. The health controller monitors the conditions that you defined for the health policy members, and takes the defined actions on the members when the conditions in the health policy breach.

What to do next

If you chose the **Supervise** reaction mode, then you receive recommendations to improve your health conditions. These recommendations display as runtime tasks that you can accept, deny, or close. To manage runtime tasks, click **System administration > Task management > Runtime tasks** in the administrative console. If you chose the **Automatic** reaction mode, actions to improve the health of your environment occur automatically.

For supervised reaction mode runtime tasks, you can set the Java virtual machine (JVM) `com.ibm.ws.xd.hmm.controller.ControlConfig.approvalTimeoutMinutes` custom property, which specifies the number of minutes that can pass before a runtime task for the health controller expires. If you set the value to 5 minutes or less, the default value of 30 minutes is automatically used instead. If you do not take any actions on the runtime task, the task expires in the number of minutes that is specified in this property. If the runtime task expires when the health condition still exists, a new runtime task is generated.

If you configure your health policies often, consider using **AdminTask** commands to automate the process.

Custom health condition subexpression builder:

Use the custom health condition subexpression builder to define a custom health condition for your health policy. Use the build subexpression utility to build complex rule conditions from subexpressions by using AND, OR, NOT and parenthetical grouping. The subexpression builder validates the rule when you apply the changes, and alerts you to mismatched parentheses and unsupported logic operators.

To view this administrative console page, click **Operational polices > Health policies > New**. If you choose a custom health condition, the **Run reaction plan when** field is displayed. Click **Subexpression builder** to build the custom health condition.

Select the properties that you want to include in your custom health condition, and click **Generate subexpression**. The subexpression value displays. To append the subexpression to your custom health condition, click **Append**.

Logical operator:

Specifies the operator that is used to append this subexpression to the previous subexpression in the custom health condition.

- and** Both subexpressions that are around the and operator must be true for actions to be taken on the health policy.
- or** To select a node, one of the two subexpressions that are around the or operator must be true for actions to be taken on the health policy.

Operand:

- **PMI metric: From server start** and **PMI metric: From last interval**

The **PMI metric: From server start operand** uses an average number of the reported values from the time that the server started.

The **PMI metric: From last interval** operand uses an average of the reported values in the last interval. The interval is the length of the health controller cycle.

Both PMI operands have the following PMI modules:

- **Connection pool module (JDBC):** These metrics can be used only on servers that are running WebSphere Application Server. The connection pool module has the following metrics:
 - **Faults:** Specifies the number of connection timeouts in the pool.
 - **Number of creates:** Specifies the total number of connections that are created.

- **Percent used:** Specifies the average percent of the pool that is in use. The value is based on the total number of configured connections in the connection pool, not the current number of connections.
- **Prepared statement cache discards:** Specifies the number of statements that are discarded because the cache is full.
- **Number of destroys:** Specifies the total number of connections that are closed.
- **Pool size:** Specifies the size of the connection pool.
- **Number of connection handles:** Specifies the number of connection objects that are discarded because the cache is full.
- **Concurrent waiters:** Specifies the average number of threads that are waiting for a connection at the same time.
- **Number of managed connections:** Specifies the number of ManagedConnection objects that are in use for a particular connection pool. This metric applies to Version 5.0 data sources only.
- **Percent maxed:** Specifies the average percent of the time that all connections are in use.
- **JDBC time (milliseconds):** Specifies the average time, in milliseconds, that is spent running Java Database Connectivity (JDBC) calls. This time includes time that is spent in the JDBC driver, network, and database. This metric applies to Version 5.0 data sources only.
- **Average use time (milliseconds):** Specifies the average time in milliseconds that a connection is used. The value is the difference between the time at which the connection is allocated and returned. This value includes the JDBC operation time.
- **Number of returns:** Specifies the total number of connections that are returned to the pool.
- **Free pool size:** Specifies the number of free connections in the pool.
- **Number of allocates:** Specifies the number of connections that are allocated.
- **Average wait time (milliseconds):** Specifies the average waiting time in milliseconds
- **System module:** These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers. The system module has the following metrics:
 - **CPU utilization: Since server start:** Specifies the average CPU utilization since the server started.
 - **CPU utilization: Last interval:** Specifies the average CPU utilization since the last query.
 - **Free memory (KB):** Specifies a snapshot of free memory, in kilobytes.
- **Process module (for other servers):** These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers. The process module has the following metrics:
 - **Process resident memory (KB):** Specifies the process resident memory, in kilobytes.
 - **Process CPU utilization: Since server start:** Specifies the process CPU utilization since the server start.
 - **Process CPU utilization: Last interval:** Specifies the process CPU utilization in the last interval.
 - **Process total memory (KB):** Specifies the process total memory, in kilobytes.
- **EJB module:** These metrics can be used only on servers that are running WebSphere Application Server. The EJB module has the following metrics:
 - **Average concurrent active methods:** Specifies the average number methods that are active at the same time.
 - **Total method calls:** Specifies the number of calls to the remote methods of the bean.
 - **Methods submodule: Method loads:** Specifies the method load in the methods submodule.
 - **Stores:** Specifies the amount of time that the bean data was stored in persistent storage.
 - **Message count:** Specifies the number of messages that were delivered to the onMessage method of the bean. This message count applies to message-driven beans.
 - **Average concurrent live beans:** Specifies the average beans that are alive at the same time.

- **Removes:** Specifies the number of times that beans were removed.
- **Returns to pool:** Specifies the number of calls that are returning an object to the pool.
- **Passivates:** Specifies the number of times that beans were moved to passivated state.
- **Gets from pool:** Specifies the number of calls that are retrieving an object from the pool.
- **Drains from pool:** Specifies the number of times that the daemon found the pool idle and attempted to clean the pool.
- **Ready count:** Specifies the number of bean instances that are in ready state.
- **Average create time (milliseconds):** Specifies the average time, in milliseconds, to run a bean create call. This time includes the time to load the bean.
- **Returns discarded** Specifies the number of times that the returning object was discarded because the pool was full.
- **Activates:** Specifies the number of times that beans were activated.
- **Server session usage (percentage):** Specifies the percentage of the ServerSession pool that is being used. This metric applies to: message-driven beans.
- **Loads:** Specifies the number of times that bean data was loaded from persistent storage.
- **Message backout count:** Specifies the number of backed out messages that failed to be delivered to the onMessage method of the bean. This metric applies to: message-driven beans.
- **Methods submodule: Method response time (milliseconds):** Specifies the method response time, in milliseconds.
- **Passivation count:** Specifies the number of beans that are in a passivated state.
- **Pool size:** Specifies the average number of objects that are in the pool.
- **Load time (milliseconds):** Specifies the average time in milliseconds for loading the bean data from persistent storage.
- **Average remove time (milliseconds):** Specifies the average time, in milliseconds, to run a beanRemove call. This time includes the time at the database.
- **Gets found:** Specifies the number of times that a retrieve call found an available object in the pool.
- **Activation time:** Specifies the average time, in milliseconds, for activating a bean object
- **Average drain size:** Specifies the average number of objects that are discarded in each drain.
- **Methods submodule: Method calls:** Specifies the number of method calls.
- **Destroys:** Specifies the number of times that beans were destroyed.
- **Average server session wait time (milliseconds):** Specifies the average time, in milliseconds, that is required to obtain a server session from the pool. This metric applies to message-driven beans.
- **Creates:** Specifies the number of times that beans were created.
- **Average method response time (milliseconds):** Specifies the average response time, in milliseconds, that elapses for remote method calls on the bean.
- **Instantiates:** Specifies the number of times that beans were instantiated.
- **Store time (milliseconds):** Specifies the average time, in milliseconds, for storing the bean data to persistent storage.
- **Passivation time (milliseconds):** Specifies the average time, in milliseconds, for bean object passivation to occur.
- **Web application module:** These metrics can be used only on servers that are running WebSphere Application Server. Web application modules have the following metrics:
 - **Number of errors:** Specifies the total number of times that an error was received from the servlet or JavaServer Pages (JSP) files.
 - **Total requests:** Specifies the total number of requests that a servlet processed.

- **Response time (milliseconds):** Specifies the average response time, in milliseconds, in which servlet requests finish.
- **Concurrent requests:** Specifies the number of requests that are currently processing.
- **Number of reloads:** Specifies the number of servlets that are reloaded.
- **Number of loaded servlets:** Specifies the number of servlets that are loaded.
- **JVM runtime module:** These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers. Java Virtual Machine (JVM) runtime modules have the following metrics:
 - **Free memory (KB):** Specifies the free memory, in kilobytes, in the JVM runtime.
 - **Up time (seconds):** Specifies the amount of time, in seconds, that the JVM has been running.
 - **Total memory (KB):** Specifies the total memory, in kilobytes, in the JVM runtime.
 - **Used memory (KB):** Specifies the amount of used memory, in kilobytes, in the JVM runtime.
- **Thread pool module:** These metrics can be used only on servers that are running WebSphere Application Server. Thread pool modules have the following metrics:
 - **Number of thread stops:** Specifies the number of threads that are declared as stopped.
 - **Percent maxed:** Specifies the average percent of the time that all threads are in use.
 - **Average active time (milliseconds):** Specifies the average time in milliseconds that the threads are in active state.
 - **Thread destroys:** Specifies the total number of threads that are destroyed.
 - **Pool size:** Specifies the average number of threads in a pool.
 - **Thread creates:** Specifies the total number of threads that are created.
 - **Concurrently hung threads:** Specifies the number of threads that are concurrently stopped.
 - **Number of cleared thread stops:** Specifies the number of thread stops that are cleared.
 - **Active threads:** Specifies the number of threads that are concurrently active.

Subexpression format for PMI Metric: From server start:

PMIMetric_FromServerStart\$moduleName\$metricName operator LongValueL (with "L" suffix)

Example:

PMIMetric_FromServerStart\$systemModule\$cpuUtilization > 90L

Subexpression format for PMI Metric: Last reported interval:

PMIMetric_FromLastInterval\$moduleName\$metricName operator LongValueL (with "L" suffix)

Example:

PMIMetric_FromLastInterval\$webAppModule\$responseTime > 200L

• **ODR server level metric: From server start**

You can use a subset of server level metrics that the on demand router (ODR) publishes. These metrics are cumulative and reported since the server start.

Metric name

You can use the following server level metrics:

- **Departs:** Specifies the number of requests that are dispatched from the queue to the server during the reported interval. A request is considered to be dispatched just once, even if it fails at the first server and is retried at another time. The next event after dispatch is return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Response time (milliseconds):** Specifies an average response time for requests. To calculate this average, add the sum of the response times for requests that returned from the server to the client during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value to get the average response time. The response time of a request

is the sum of the request's waiting time and service time. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

- **Currently executing requests:** Specifies the number of requests that are running at the end of the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Service time (milliseconds)** Specifies the average service time for requests. To calculate this average, add the sum of each request service time for requests that returned to the server during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value in the same interval to get the average. The service time of a request is the time from dispatch to return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Wait time (milliseconds):** Specifies the average wait time for requests. To calculate this average, add the sum of the time that each request spent waiting in the queue over the reported interval. The sum is in units of milliseconds. Divide by the number of departs to get the average wait time. Dropped requests do not contribute to this sum. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Errors:** Specifies the number of requests that returned from the server with an error indicator during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Serviced:** Specifies the number of requests that returned from server to client during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Timeouts:** Specifies the number of requests that returned due to service timeout during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

Subexpression format:

`ODRServerMetric_FromServerStart$metricName operator LongValueL (with "L" suffix)`

Example:

`ODRServerMetric_FromServerStart$errors > 100L`

• ODR server level metric: Last reported interval

You can use the same set of metrics as the ODR server level metric: From server start operand. This operand uses an average of the reported values in the last interval. The interval is the length of the health controller cycle.

- **Departs:** Specifies the number of requests that are dispatched from the queue to the server during the reported interval. A request is considered to be dispatched just once, even if it fails at the first server and is retried at another time. The next event after dispatch is return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Response time (milliseconds):** Specifies an average response time for requests. To calculate this average, add the sum of the response times for requests that returned from the server to the client during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value to get the average response time. The response time of a request is the sum of the request's waiting time and service time. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Currently executing requests:** Specifies the number of requests that are running at the end of the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Service time (milliseconds)** Specifies the average service time for requests. To calculate this average, add the sum of each request service time for requests that returned to the server during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value in the same

interval to get the average. The service time of a request is the time from dispatch to return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

- **Wait time (milliseconds):** Specifies the average wait time for requests. To calculate this average, add the sum of the time that each request spent waiting in the queue over the reported interval. The sum is in units of milliseconds. Divide by the number of departs to get the average wait time. Dropped requests do not contribute to this sum. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Errors:** Specifies the number of requests that returned from the server with an error indicator during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Serviced:** Specifies the number of requests that returned from server to client during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Timeouts:** Specifies the number of requests that returned due to service timeout during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

Subexpression format:

`ODRServerMetric_FromLastInterval$metricName operator LongValue (with "L" suffix)`

Example:

`ODRServerMetric_FromLastInterval$serviced > 10000L`

• **ODR cell level metric: From ODR start**

You can use a subset of cell level metrics that the ODR publishes. These metrics are cumulative and reported since the server start. You can use the following set of metrics:

- **Departs:** Specifies the number of requests that are dispatched from the queue to the server during the reported interval. A request is considered to be dispatched just once, even if it fails at the first server and is retried at another time. The next event after dispatch is return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Response time (milliseconds):** Specifies an average response time for requests. To calculate this average, add the sum of the response times for requests that returned from the server to the client during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value to get the average response time. The response time of a request is the sum of the request's waiting time and service time. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Current[®] queue length:** Specifies the length of the queue at the end of the reported interval.
- **Service time (milliseconds)**
- **Errors:** Specifies the number of requests that returned from the server with an error indicator during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Average queue length:** Specifies the average length of the queue. To calculate this average, add the sum of the queue lengths that are reported at each request arrival before insertion, and divide the sum by the number of arrivals. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Serviced:** Specifies the number of requests that returned from server to client during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Timeouts:** Specifies the number of requests that returned due to service timeout during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

- **Currently executing requests:** Specifies the number of requests that are running at the end of the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Arrivals:** Specifies the number of requests that arrived during the reported interval. The next event, if any, after arrival is either dispatch or drop. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Queue overflow drops:** Specifies the number of requests that were initially accepted into the queue at some time and then, at some time during the reported interval, were ejected from the queue. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Queue drops:** Specifies the number of requests that were initially accepted into the queue at some time and then were ejected from the queue at some time during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Delayed:** Specifies the number of requests that arrived during the reported interval and were not immediately dispatched or dropped. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Wait time (milliseconds):** Specifies the average wait time for requests. To calculate this average, add the sum of the time that each request spent waiting in the queue over the reported interval. The sum is in units of milliseconds. Divide by the number of departs to get the average wait time. Dropped requests do not contribute to this sum. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

Subexpression format:

`ODRCellMetric_FromServerStart$metricName operator LongValue (with "L" suffix)`

Example:

`ODRCellMetric_FromServerStart$arrivals > 10000L`

• **ODR cell level metric: Last reported interval**

You can use the same set of metrics as the ODR cell level metric: From server start operand. This operand uses an average of the reported values in the last interval. The interval is the length of the health controller cycle.

- **Departs:** Specifies the number of requests that are dispatched from the queue to the server during the reported interval. A request is considered to be dispatched just once, even if it fails at the first server and is retried at another time. The next event after dispatch is return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Response time (milliseconds):** Specifies an average response time for requests. To calculate this average, add the sum of the response times for requests that returned from the server to the client during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value to get the average response time. The response time of a request is the sum of the request's waiting time and service time. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Current queue length:** Specifies the length of the queue at the end of the reported interval.
- **Service time (milliseconds)** Specifies the average service time for requests. To calculate this average, add the sum of each request service time for requests that returned to the server during the reported interval. The sum is in units of milliseconds. Divide by the serviced metric value in the same interval to get the average. The service time of a request is the time from dispatch to return. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Errors:** Specifies the number of requests that returned from the server with an error indicator during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

- **Average queue length:** Specifies the average length of the queue. To calculate this average, add the sum of the queue lengths that are reported at each request arrival before insertion, and divide the sum by the number of arrivals. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Serviced:** Specifies the number of requests that returned from server to client during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Timeouts:** Specifies the number of requests that returned due to service timeout during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Currently executing requests:** Specifies the number of requests that are running at the end of the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Arrivals:** Specifies the number of requests that arrived during the reported interval. The next event, if any, after arrival is either dispatch or drop. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Queue overflow drops:** Specifies the number of requests that were initially accepted into the queue at some time and then, at some time during the reported interval, were ejected from the queue. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Queue drops:** Specifies the number of requests that were initially accepted into the queue at some time and then were ejected from the queue at some time during the reported interval. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Delayed:** Specifies the number of requests that arrived during the reported interval and were not immediately dispatched or dropped. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.
- **Wait time (milliseconds):** Specifies the average wait time for requests. To calculate this average, add the sum of the time that each request spent waiting in the queue over the reported interval. The sum is in units of milliseconds. Divide by the number of departs to get the average wait time. Dropped requests do not contribute to this sum. These metrics can be used on servers that are running WebSphere Application Server or that are running other middleware servers.

Subexpression format:

`ODRCellMetric_FromLastInterval$metricName operator LongValue (with "L" suffix)`

Example:

`ODRCellMetric_FromLastInterval$timeouts > 100L`

• **MBean operation metric: Long return type and MBean operation metric: String return type**

For Managed Bean (Mbean) operation metric operands, you must specify the Object name query string and the MBean method name. These metrics can be used only on servers that are running WebSphere Application Server.

Object name query string

When you create the Object name query string, escape all special characters with a backslash character.

The value that you enter for the object name query string must have both the `process=process_name>`, and `node=<node_name>` strings specified, or none specified. If you specify both `process=process_name>`, and `node=node_name>`, the backend creates a singleton MBeanSensor sensor that senses the particular MBean on a server and node. If you specify none, the backend appends the name of the current server as the process name and the name of the current node as the node name, creating a MBeanSensor sensor for each server to which the health policy applies. If you specify only one of the two `process=<process_name>` or `node=node_name>`, an error results.

MBean method name

Specifies the name of the MBean method to invoke.

Subexpression format long metrics:

```
MBeanOperationMetric_TypeLong$ObjectNameQueryString$methodName operator LongValueL (with "L" suffix)
```

Example for long metrics:

```
MBeanOperationMetric_TypeLong$WebSphere\:*\,type\  
=HealthConditionLanguageInitializer\,node\=hipods3\,process\=nodeagent$getNumberOfOperands > 10L
```

Subexpression format for string metrics:

```
MBeanOperationMetric_TypeString$ObjectNameQueryString$methodName operator StringValue
```

Example for string metrics:

```
MBeanOperationMetric_TypeString$WebSphere\:*\,type\  
=HealthConditionLanguageInitializer\,node\=hipods3\,process\=nodeagent$getOperands = 't'
```

- **MBean attribute metric: Long return type and MBean attribute metric: String return type**

The MBean attribute metrics are used for querying an attribute of a MBean rather than invoking a method on the MBean. The operand takes the Object name query string and the attribute name as inputs. These metrics can be used only on servers that are running WebSphere Application Server.

Object name query string

When you create the Object name query string, escape all special characters with a backslash character.

The value that you enter for the object name query string must have both the process=process_name>, and node=<node_name> strings specified, or none specified. If you specify both process=process_name>, and node=node_name>, the backend creates a singleton MBeanSensor sensor that senses the particular MBean on a server and node. If you specify none, the backend appends the name of the current server as the process name and the name of the current node as the node name, creating a MBeanSensor sensor for each server to which the health policy applies. If you specify only one of the two process=<process_name> or node=node_name>, an error results.

Attribute name

Specifies the attribute that is queried on the MBean.

Subexpression format for long metrics:

```
MBeanAttributeMetric_TypeLong$ObjectNameQueryString$attributeName operator LongValue
```

Example for long metrics:

```
MBeanAttributeMetric_TypeLong$WebSphere\:*\,type\  
=HealthConditionLanguageInitializer\,node\=hipods3\,process\=nodeagent$NumberOfOperands > 10L
```

Subexpression format for string metrics:

```
MBeanAttributeMetric_TypeString$ObjectNameQueryString$attributeName operator StringValue
```

Example for string metrics:

```
MBeanAttributeMetric_TypeString$WebSphere\:*\,type\  
=HealthConditionLanguageInitializer\,node\=hipods3\,process\=nodeagent$OperatorList = 'test'
```

- **URL return code metric**

With this operand, you can ping any relative path (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.

URL port number

Specifies the port number to ping.

URL relative path

Specifies the URL to ping. Any special characters in the string must be escaped with a backslash (\) character.

Value Specifies an integer that is the expected return code of the ping.

Use this operand to ping any general purpose URL by selecting the on demand router (ODR) as the target of the health policy and by setting the appropriate routing rules in the ODR.

You can use this operand to select members that are running WebSphere Application Server, or that are running other middleware servers.

Subexpression format:

URLReturnCodeMetric\$portNumber\$relativePath operator IntValue

Example:

URLReturnCodeMetric\$9060\$ibm\console\login\do = 200

The URL sensor returns 0 if the Web site cannot be reached:

URLReturnCodeMetric\$9060\$ibm\console\login\do = 0

- **External URL return code metric**

With this operand, you can enter an absolute URL instead of a relative URL. By doing so, you can periodically send a ping request to other targets than application servers or on demand routers.

Always create a custom health action without specifying a target server.

Example:

ExternalURLReturnCodeMetric\$http://foo.bar.com <> 200

If http://foo.bar.com returns a different response code than 200, the health policy is triggered. Take this custom action to run a custom script, for example a script to recycle a web server.

Operator:

- **Equals (=):** The equality operator expresses a case-sensitive match.
- **Not Equals (<>):** The not equal operator expresses that the operand value is not equal to the value you enter.
- **Greater Than (>):** The greater-than operator is for use with numbers.
- **Greater Than or Equals (>=):** The greater-than or equal to operator is for use with numbers.
- **Less Than (<):** The less-than operator is for use with numbers.
- **Less Than or Equals (<=):** The less-than or equal to operator is for use with numbers.
- **Between (BETWEEN):** The value must be between a **Lower bound** and **Upper bound** that you specify.
- **In (IN):** The value must be in a list of values. You can type in values and add them to a list.

Value:

Depending on the operator that you choose, type in a value for the subexpression that you want to create.

Subexpression:

After you click **Generate subexpression**, this field displays the generated subexpression fragment based on the options that you selected. To add this subexpression to your custom health condition, click **Append**.

Creating health policy custom actions:

Use custom actions to define custom corrective tasks that you can use when a health condition broken. You can also use custom actions to invoke **wasadmin** scripts.

Before you begin

Create a Java or non-Java executable file to run when the health condition breaches. You can use environment variables in your executable file to display the cell, node, and server that is affected by the health action, for example:

```
#!/bin/sh
echo "The sick server is "
echo $server
echo " which is on node "
echo $node
echo " which is a part of cell "
echo $cell
```

About this task

By default, you can define the following actions to occur when a health condition is broken: restart the server, take thread dumps, or take heap dumps. If you have more specific needs for actions that occur when a health condition breaches, then create a *custom action*. Custom actions consist of Java or non-Java process definitions, and 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.

When a health policy contains multiple custom actions, updating the custom actions can cause the health controller runtime to be out-of-sync with the administrative console configuration. If you add, delete or modify a custom action for a policy that contains multiple custom actions, start, and stop the health controller after saving your changes.

If you use a custom action to invoke a `wsadmin` script, create a custom script on your target endpoint, that contains the `wsadmin` invocation. For example, create a custom script `test.sh`, which uses a Python script to achieve its goal:

```
test.sh:
#!/bin/sh
/opt/IBM/WAS/bin/wsadmin.sh -lang jython -f /opt/IBM/WAS/wsadmin_test.py param1 param2 param3

wsadmin_test.py:
for arg in sys.argv:
#do something with args
```

In the administrative console, click **Operational policies > Custom action > New**. Choose the action type and define the custom action properties:

- **Name:**
- **Executable:**
- **Executable arguments:**
- **Operating system:**
- **Working directory:**

Procedure

1. Select a Java or non-Java action type. In the administrative console, click **Operational policies > Custom action > New**. Choose the type of action that you want to create.
2. Define the custom action properties.

For health actions that run non-Java and Java executable files:

- Specify a name for the action, the fully qualified path and name of the executable file, and any arguments to pass to the executable file.
- If required, specify variables for the user name and password, and specify the actual values to pass for the user name and password at the time the executable file runs.
- Specify the operating systems on which the executable files can run.

- Specify a fully qualified path for a working directory, which is where to run the executable file.

For health actions that run Java executable files, you must also specify the following properties:

- Specify the location of the Java executable directory.
- Specify the type of target: a Java class, or an executable Java Archive (JAR) file.
- Specify the environment variable that stores the process ID (PID) file name. The PID file name is stored in a WebSphere variable. Remember the name of the variable so that you can create the variable in a later step.

For example, if you had an executable file to collect logs called `logCollector.sh`, you enter the following values for the custom action:

- **Name:** `LogCollect`
- **Executable:** `/opt/mws/bin/logCollector.sh`
- **Executable arguments:** `None`
- **Operating system:** `UNIX`
- **Working directory:** `/opt/mws/bin/`

3. Save the custom action.
4. If your health action is running a Java executable file, create the PID file name environment variable. The PID file name specifies the location of the PID file where the server operation runs. To define the PID file name variable, click **Servers > All servers > *middleware_server* > Variables > New** or **Environment > WebSphere variables > New**. Enter the same name for the variable that you entered in the health action panel, and for the value, enter the fully qualified or variable qualified location of the PID file.

Results

You created a custom action that you can add to the action plans of your health policies.

What to do next

Create a health policy. In the administrative console, click **Operational policies > Health policies > New**.

Modifying the JVM heap size:

The Java virtual machine (JVM) heap size setting directly relates to how many server instances can be started within a dynamic cluster on a specific node. You might need to modify the JVM heap size setting based on your environment configuration. The default value is 256 MB. Modifying the heap size can affect application placement and the health management function.

Before you begin

Before modifying the JVM heap size, read about tuning Java virtual machines.

About this task

The total value of all server JVM heap sizes within the dynamic cluster for a specific node must be less than half of the total RAM of that computer. To determine the maximum heap size setting for a single server instance, use the following equation:

$$total_RAM / 2 / number_of_servers = maximum_heap_size$$

For example, to support three servers on a machine with 1.5 GB of RAM:

$$1.5 \text{ GB} / 2 = 750 \text{ MB}$$

$$750 \text{ MB} / 3 = 250 \text{ MB}$$

The maximum heap size is 250 MB for each server instance.

The following table contains some maximum heap size samples.

Table 33. Maximum heap size samples

Intelligent Management server total RAM in GB	Number of supported servers	Dynamic cluster size in MB	Approximate maximum heap size in MB
1.5	1	Less than 750	750
1.5	2	Less than 750	375
1.5	3	Less than 750	250
2	1	Less than 1000	1000
2	2	Less than 1000	500
2	3	Less than 1000	333

When a dynamic cluster creates server instances, a server template is used to set initial values for the newly created server instance. The following steps outline the procedure for modifying JVM heap size within the server template. Newly created server instances can then use the maximum heap size you specified.

Procedure

1. In the administrative console click **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name* > Server template > Java and process management > Process definition > Java virtual machine.**
2. Modify the heap size accordingly. Typically, the total value of all server instance JVM heap sizes on a specific node must be less than half of the total RAM of that computer. The default maximum heap size value is 256 MB.
3. Click **OK**.

Results

The heap size that you define in the dynamic cluster template is configured on each server in the dynamic cluster. If the server and application placement controller are on the same node, the application placement controller can retrieve the heap size from this configured value.

If the server is not on the same node as the application placement controller, the application placement controller obtains the heap size from the configuration repository in the deployment manager. If the deployment manager is not available or the heap size is not defined on the deployment manager, the application placement controller tries to obtain the heap size by contacting the server. If the server is not available, the application placement controller uses the heap size value in the `<os>.systemlaunch.properties` file.

Monitoring and tuning health management:

Health management offers default settings that suit most environments. However, if you discover that your health controller is not working as expected, tune the default parameters.

Before you begin

Verify that you have proper security authorization in the console to modify these settings. Privileges for health policies differ, depending on the administrative role of the user. Roles include monitor, operator, configurator, and administrator. If you are a user with either a monitor or an operator role, you can only view health policy information. If you are a user with either a configurator or an administrator role, you have all configuration privileges for health policies.

About this task

Use the following steps to modify the health controller parameters. Tune these parameters when the health management infrastructure is not working the way that you want.

Procedure

1. In the administrative console click **Operational policies > Autonomic managers > Health controller**.
2. Determine whether you want your changes to be persistent or applied to the current runtime for testing purposes. On the **Configuration** tab you can view the fields that are previously configured, and in some cases, you can edit these fields. On the **Runtime** tab, you can view the fields that are currently used by the health controller, and in some cases, make changes to these values. The values changed on the **Runtime** tab are sent directly to the health controller, and the controller parameters are modified. Because these changes are not stored in the repository by default, you can make temporary parameter changes.

Tip: Enter your changes on the **Runtime** tab and test the changes before committing them. Select **Save to configuration** on the **Runtime** tab, to make configuration changes and test them in the runtime. If you want to commit your changes, click **Save to configuration**.

3. Modify and test your settings.

Setting	Description
Control cycle length	Specifies the time between consecutive health checks. The value is specified in minutes and ranges from 1 to 60 minutes. Longer control cycles reduce the health monitoring load. The disadvantage is that health conditions that occur during that period are not detected until the next control cycle. For example, if you have a health policy with a workload condition of 10,000 requests associated to an application server and the value is specified as 60 minutes, the health controller checks every 60 minutes to determine if the application server has served 10,000 requests. If 9,999 requests are detected during a health check, and a new health check occurs after another 60 minutes (the control cycle length), the server actually serves more than 10,000 requests prior to a restart.
Maximum consecutive restarts	Specifies the number of attempts to revive an application server after a restart decision is made. If this number is exceeded, the assumption is that the operation failed and restarts are disabled for the server. The value must be a whole number between 1 and 5, inclusive.
Minimum restart interval	Controls the minimum amount of time that must elapse between consecutive restarts of an application server instance. If a health condition for an application server is breached during that time, the restart is set to a pending state. When the minimum restart interval passes, the restart occurs. The value can range from 15 minutes to 365 days, inclusive. A value of 0 disables the minimum restart value.
Restart timeout	Consists of the sequence of stop and start server actions. The restart timeout specifies how long to wait for a server to stop before explicitly checking its state and attempting startup. If the length of time to start and stop an application server is unusually high, set this value so that the restart action does not time out. Always specify the value in minutes. The value can range from 1 minute to 60 minutes, specified as a whole number.
Enable health monitoring	Enables or disables the operation of the health controller. When enabled, the health controller continuously monitors the health policies in the system. You can disable the health controller without removing the health policies from the system.

Setting	Description
Prohibited restart times	<p>Specifies the times and days of the week when a restart of an application server instance is prohibited. Specify the start and end times by selecting the hour and minute using a 24 hour clock, and by selecting the days of the week.</p> <p>You can specify multiple time blocks, if needed. If you specify a start time and end time, you must also specify at least one day of the week when these intervals are prohibited. The block between the start time and end time cannot cross the midnight boundary. If you need to specify a time block of, for example, 10:00 PM to 1:00 AM, you need to specify two time blocks, one from 22:00 to 23:59 and one from 00:00 to 01:00. Click Add to add additional time constraints.</p> <p>To remove an existing constraint, select the check box next to the constraint and click Remove. If the restart time breaches a health condition, the restart is delayed until the prohibited time interval passes.</p>

Results

You have modified the health management configuration settings to tune your system.

What to do next

For more information about modifying the health management settings when they are not working as expected, read the troubleshooting information.

Tuning the Intelligent Management cell:

Start here for a summary of key performance tuning parameters for your Intelligent Management configuration.

About this task

You can complete any of these steps to improve the performance of your Intelligent Management configuration.

Procedure

- Assign all URL patterns from an application to transaction classes. Any URL that does not map to a transaction class is classified as *best effort*, also called discretionary. Transaction classes are part of service policies. To change transaction classes from the administrative console, select **Operational policies > Service policies > *service_policy_name*Transaction classes**.
- Check service policy goals against the performance of the service class and adjust the goals if necessary. For example, your service policy has an average response goal of 200 milliseconds and the measured average service time is 300 milliseconds. Consider adjusting your goal to 500 milliseconds.
- To change the application placement controller speed for loading changes, adjust the following properties:
 - Configure the minimum time between placement changes in the administrative console. Select **Operational policies > Autonomic managers > Autonomic controllers > Application placement controller**. Set the minimum time between placement changes to a suitable value.
 - Configure the application placement controller cycle from the administrative console. Select **Operational policies > Autonomic managers > Autonomic controllers > Application placement controller > Custom properties**. Add a custom property with the name **minControlCycleLength** and a suitable value in minutes.

- Adjust the control cycle length. The control cycle length minimum property determines how quickly the autonomic request flow manager (ARFM) reacts to load changes. You can change the ARFM control cycle length. From the administrative console select **Operational policies > Autonomic managers > Autonomic request flow manager**. Set the control cycle length minimum property to a suitable value.
- Adjust ARFM to limit the maximum use of the nodes.
 1. In the administrative console, select **Operational policies > Autonomic managers > Autonomic request flow manager**.
 2. Set the Maximum CPU Utilization property. Higher values allow better resource use, lower values ensure more robust operation.

Troubleshooting health management:

You can look for the following problems when health management is not working, or not working the way you expect.

Finding the correct logs

The health controller is a distributed resource that is managed by the high availability (HA) manager. It exists within all node agent and deployment manager processes and is active within one of these processes. If a process fails, the controller becomes active on another node agent or deployment manager process.

To determine where the health controller is running, click **Runtime operations > Component stability > Core components** in the administrative console. The location and stability status of the health controller displays.

The performance advisor is enabled with the default memory leak health policy

The default memory leak health policy uses the performance advisor functionality, so the performance advisor is enabled when this policy has members assigned. To disable the performance advisor, remove this health policy or narrow the membership of the health policy. To preserve the health policy for future use, keep the default memory leak policy, but remove all members. To change the members, click **Operational policies > Health policies > default_memory_leak**. You can edit the health policy memberships by adding and removing specific members.

Health controller settings

The following list contains issues that are encountered as a result of the health controller settings:

Health controller is disabled

To verify the setting in the administrative console, click **Operational policies > Autonomic managers > Health controller** and select both the **Configuration** and **Runtime** tabs. The health controller is enabled by default.

Restarts are prohibited at this time

To verify the prohibited restart times in the administrative console, click **Operational policies > Autonomic managers > Health controller** and select the **Prohibited restart** field. By default, no time values are prohibited.

Restarting too soon after the previous restart

To check the minimum restart interval in the administrative console, click **Operational policies > Autonomic managers > Health controller** and modify the **Minimum restart interval** field. No minimum interval is defined by default.

Control cycle is too long

To check the control cycle length in the administrative console, click **Operational policies >**

Autonomic managers > Health controller and adjust the value if necessary. The health controller checks for policy violations periodically. If its control cycle length is too long, it might not restart servers quickly enough.

The server was restarted X times consecutively, and the health condition continues to be violated

In this case, *X* indicates the maximum consecutive restart parameter of the health controller. The health controller concludes that restarts are not fixing the problem, and disables the restarts for the server. The following message displays in the log:

```
WXDH0011W: Server servername exceeded maximum verification failures: disabling restarts.
```

The health controller continues to monitor the server and displays messages in the log if the health policy is violated:

```
WXDH0012W: Server servername with restarts disabled failed health check.
```

You can enable restarts for the server by performing any of the following actions:

- Disable and then enable the health controller.
- Adjust the **Maximum consecutive restarts** controller setting.
- Run the following command from the prompt:

```
wsadmin -profile HmmControllerProcs.jacl enableServer servername
```

This script is available in the `<app_server_root>\bin` directory on the node agent or deployment manager nodes. This script requires a running deployment manager.

Health policy settings

The following issues are encountered as a result of the health policy settings:

The server is not part of a health policy

To verify that the health policy memberships apply to your server in the administrative console, click **Operational policies > Health policies**.

The reaction mode of a policy containing the server is supervised

To check the administrative console, click **System administration > Task management > Runtime tasks**. Find approval requests for a restart action for a policy in **Supervised** mode. Servers are restarted automatically when you set the reaction mode to **Automatic**. The following message is written to the log for the supervised condition:

```
WXDH0024I: Server server name has violated the health policy health condition, reaction mode is supervised.
```

The server is a member of a static cluster and is the only cluster member running

The health policy does not bring down all members of a cluster at the same time. If a cluster has one cluster member, or one cluster member is running, then the cluster is not restarted.

The server is a member of a dynamic cluster, the number of running instances does not exceed the minimum value, and the placement controller is disabled

To check the **minimum number of instances required** for the dynamic cluster, click **Servers > Clusters > Dynamic clusters** in the administrative console. In this case, health management treats the dynamic cluster like a static cluster, using the minimum number of instances parameter.

The health controller has not received the policy

The health controller does not run on the deployment manager where the health policies are created. If the deployment manager is restarted after the health controller started, the health controller might not have the new policy.

To resolve this problem, perform the following steps:

1. Disable the health controller. In the administrative console, click **Operational policies > Autonomic managers > Health controller**.

2. Synchronize the configuration repositories with the back-end nodes. In the administrative console, click **System administration > Nodes**. Select the nodes to synchronize, and click **Synchronize**.
3. Restart the health controller. In the administrative console, click **Operational policies > Autonomic managers > Health controller**.
4. Synchronize the configuration repositories with the back-end nodes. In the administrative console, click **System administration > Nodes**. Select the nodes to synchronize, and click **Synchronize**.

Application placement controller interactions

The following list contains issues resulting from health management and application placement controller interactions:

The server is a member of a dynamic cluster, but the placement controller cannot be contacted

For dynamic cluster members, health monitoring checks with the application placement controller to determine whether a server can be restarted. If the application placement controller is enabled, but cannot be contacted, the following message displays in the log:

```
WXDH1018E: Could not contact the placement controller
```

Verify that the placement controller is running. To determine where the health controller is running, click **Runtime operations > Component stability > Core components** in the administrative console. The location and stability status of the health controller displays. The health controller logs messages to the particular node agent or deployment manager indicated by the current location.

The server is stopped, but not started.

In a dynamic cluster, a restart can take one of several forms:

- Restart in place (stop server, start server).

Note: Always occurs when a dynamic cluster is in a manual mode.

- Start a server instance on another node, and stop the failing one.
- Stop the failing server only, assuming that the remaining application instances can satisfy demand.

Sensor problems

The following list contains issues related to health management and node group membership settings:

No sensor data is received for the server.

Health management cannot detect a policy violation if it receives no data from the sensors that are required by the policy. If sensor data is not received during the control cycle, health management prints the following log message:

```
WXDH3001E: No sensor data received during control cycle from server server_name for health class healthpolicy.
```

For response time conditions, health management receives data from the on demand router (ODR). No data is generated for these conditions until requests are sent through the ODR.

Routing requests to nodes that are not running Intelligent Management

Use this task for configuring multiple tiers.

Before you begin

Plan your topology.

About this task

To make the autonomic request flow manager aware of nodes that are not running Intelligent Management, you must use one of the following options to configure the node in the Intelligent Management configuration.

Procedure

- The most common scenario is to install the node agent on each node that is running other middleware software. For more information about running middleware software, read about adding middleware servers to configurations .
- Set the cell to run in Magic N mode. Magic N is the total number of requests that can run concurrently on the node group. To configure Magic N mode, specify the `magicNMode` and `magicN` custom properties. For more information about these custom properties, read about autonomic request flow manager custom properties. Do not use this approach if your different service classes have very different computational demands.
- Specify work factors or speed factors and the computing power of each node. Speed factors are computed automatically by the work profiler for target tiers. For external nodes, you can manually configure speed factors for transaction class modules, which are a transaction class and web module pair. For more information about work and speed factors, read about configuring work factors in multiple tier configurations.
- Set the autonomic request flow manager in manual mode and specify the various settings. Let the autonomic request flow manager estimate speed factors based only on observations from an on demand router (ODR).

In manual mode, you must supply seat counts, or concurrency allocations, for each combination of ODR, deployment target, and service class. The dequeuing weights and gateway concurrency limits are derived from this information.

To allocate a certain number of seats to each combination of ODR, deployment target, and service class, use the `ProfileRoot/properties/arfm/manual/cell_name.xml` file. In WebSphere Extended Deployment Version 5.1, the file was `was_root/properties/arfm/manual/cell_name/node_group_name.xml`. Use this file on the machine where the autonomic request flow manager (ARFM) controller is running. The `manual/` directory contains an example file that describes the format.

For more information about the `manual/` directory, read about configuring the autonomic request flow manager.

Routing requests to external nodes with generic server clusters

Use this task to route requests from the on demand router (ODR) to external nodes.

Before you begin

Note: To configure dynamic operations support for other middleware server types, you can define your other middleware servers as members of a dynamic cluster. For more information about defining middleware servers, read about adding middleware servers to configurations.

You must have one or more nodes that are not running Intelligent Management. When you perform this task, you add the nodes that are not running Intelligent Management as generic server cluster ports.

To create and edit generic server clusters, you need administrator or configurator administrative privileges.

The following tasks must be completed before you configure the ODR to send requests to external nodes:

1. Create an ODR.
2. Configure service classes and transaction classes for the ODR.

About this task

The ODR supports automatic routing to nodes that are in remote cells if the ODR's local cell is connected to the remote cell by a core group bridge. To route to nodes that do not have Intelligent Management installed, configure a generic server cluster. A generic server cluster is a collection of transport end points that can be used as a target for an ODR routing rule. Servers and nodes are inferred from the generic server cluster end points and properties. The inferred servers can be foreign or generic servers. Generic servers can be stopped and started by a WebSphere Application Server node agent. Foreign servers can also be stopped and started by the WebSphere Application Server node agent. Some examples of foreign servers include Apache Tomcat or JBoss Application Server.

Procedure

1. Optional: Create one or more unmanaged nodes. You need to create unmanaged nodes in any of the following configuration situations:
 - If you want the remote agent to listen for requests from a port other than the default port, 9980.
 - If you are using a custom statistics collector plug-in to obtain node speed and average CPU utilization for the work profiler.
 - If your nodes are multi-homed, that is, they have more than one network interface card or multiple host names.

In the administrative console, click **System administration > Nodes > Add node** to create an unmanaged node.

2. Optional: If the remote agent listens for incoming requests from a port other than the default port, 9980, specify the `stats.collector.remote.agent.port` custom property on the unmanaged node.
 - a. Click **System administration > Nodes > *node_name* > Custom properties > New**.
 - b. For the **Name**, enter `stats.collector.remote.agent.port`.
 - c. For the **Value**, enter the port number where the remote agent listens for requests.

Remember: Write down the value of this port. You use the port value when you start the remote agent on the unmanaged node.

3. Optional: If you are using a custom statistics collector plug-in to obtain node speed and average CPU utilization for the work profiler, define the name of the class as a custom property on the unmanaged node. In this case, the plug-in might or might not use a remote agent. Before you configure this custom property, you must have a custom statistics collector plug-in coded and installed in the `lib` directory.
 - a. Click **System administration > Nodes > *node_name* > Custom properties > New**.
 - b. For the **Name**, enter `stats.collector.implementation.class`.
 - c. For the **Value**, enter the name of the custom statistics collector plug-in class. The default is the `com.ibm.ws.externalnode.plugins.DefaultStatsCollector` class.
 - d. You can specify additional configuration parameters by adding the `stats.collector.config` custom property on the unmanaged nodes. The provider of the custom plug-in must provide you with the information. The `stats.collector.config` is not required if you are using the default `com.ibm.ws.externalnode.plugins.DefaultStatsCollector` class.
4. Create a generic server cluster.
 - a. In the administrative console, click **Servers > Clusters > Generic server clusters > New**.
 - b. Type a **Name** and select a **Protocol** for your generic server cluster.
 - c. Click **Apply**.
 - d. Specify the generic cluster server end points, or ports. On the generic cluster server settings page, click **Ports**.
 - e. Click **New** to add end points to the generic server cluster.

5. Optional: If you configured the remote agent to listen on a non-default port in a previous step, if you are using a custom statistics collector plug-in, or if your nodes are multi-homed, that is, they have more than one network interface card or multiple host names, you must define the node custom property on the generic server cluster end points.
 - a. In the administrative console, click **Servers > Clusters > Generic server clusters > *cluster_name* > Additional properties > Ports > *host_name* > Additional properties > Custom properties > New**.
 - b. Define the custom property. Type the name of the custom property as **node**. The **value** of the custom property should be the name of the unmanaged node that is associated with the generic server cluster port.
 - c. Click **Apply**.
6. Install your application on the external nodes using your defined port numbers.
7. Enable external monitoring by enabling the remote agent on all of your external nodes.
8. Define routing policies for generic server clusters.

Before creating routing rules, you must create the appropriate virtual hosts. The virtual hosts can be defined in the console by clicking **Environment > Virtual hosts**.
9. Define service policies for generic server clusters.

Results

The work profiler obtains average CPU utilization and node speed from external nodes and can route traffic to the external nodes.

What to do next

Continue configuring the on demand router and the autonomic request flow manager.

Enabling external monitoring with the remote agent:

Enable the remote agent on external nodes.

Before you begin

Set up generic server clusters with ports for all of your external nodes.

About this task

By installing the mixed server environment, a lightweight Java process called the remote agent runs on the node that is being monitored. There is also a plug-in that polls the remote agent for CPU information. The plug-in runs in the on demand router that has the work profiler running.

Procedure

1. Start the remote agent on nodes that you want to monitor. On a command line for each node, start the remote agent.
 - a. Navigate to the directory where the remote agent is installed. For example, type `cd <installed_dir>/bin`.
 - b. Start the remote agent. Run `./startAgent.sh` or `startagent.bat`.

Note: If you start the remote agent using the `startAgent.sh` or `startAgent.bat` script from the `bin` directory, the Java Runtime Environment (JRE that is bundled with the installer is used. The JRE is installed in the `_jvm` subdirectory of the `_uninstXD` directory.

You can also supply the port number as a parameter to the script if the default port, 9980, is used by another process that is running on the node. By default, the remote agent can be run by any user. You can limit access to the directory that contains the remote agent for increased security.

2. Optional: Stop the remote agent. To stop the remote agent, use the command window where you started the remote agent. Type the command **quit**. You can also stop the remote agent by pressing Control-C or by using the UNIX **kill** command if you are running AIX, HP-UX, Linux, or Solaris.

Results

The remote agent sends average CPU utilization, node speed, number of CPUs, and several other attributes and values to the plug-in that polls the remote agent for CPU information.

Example

You can start the remote agent in any way that a standalone Java program can be started. You can run the remote agent in the background.

On the Windows platform, you can start the remote agent as a scheduled task during system startup.

1. From Windows, click **Start > Settings > Control panel > Scheduled tasks > Add scheduled task**.
2. From the Scheduled Task Wizard, go to `WAS_HOME/bin` and select **startAgent.bat**.
3. Specify when you want to perform the task. For example, choose **When my computer starts**.
4. Specify the user ID and password for the user that you want to use to run **startAgent.bat**.

On the Linux or UNIX operating system, add an entry to the `/etc/inittab` file of the operating system. If you installed the feature in the `/opt/WAS60/MixedServer` directory, the new entry for the file is:

```
was:2:once:/opt/WAS60/MixedServer/bin/startAgent.sh >/dev/console 2>&1
```

For more information, see the UNIX manual page on `inittab` by typing `man inittabb`.

What to do next

Enabling the remote agent is a requirement when routing requests to nodes that are not running Intelligent Management. For more information about routing requests, read about routing requests to external nodes with generic server clusters.

Multiple tiers of processing

The autonomic request flow manager (ARFM) can process requests in multiple tiers. The tiers might include both WebSphere Application Server software and other software.

In a configuration that has multiple tiers, it is assumed that all of the software is organized into nodes, server processes or application servers, server process clusters, deployment targets, which can include both clusters and server processes that are not clustered, and processing tiers. For more information about servers and clusters that are not running WebSphere software, read about Intelligent Management topology terminology.

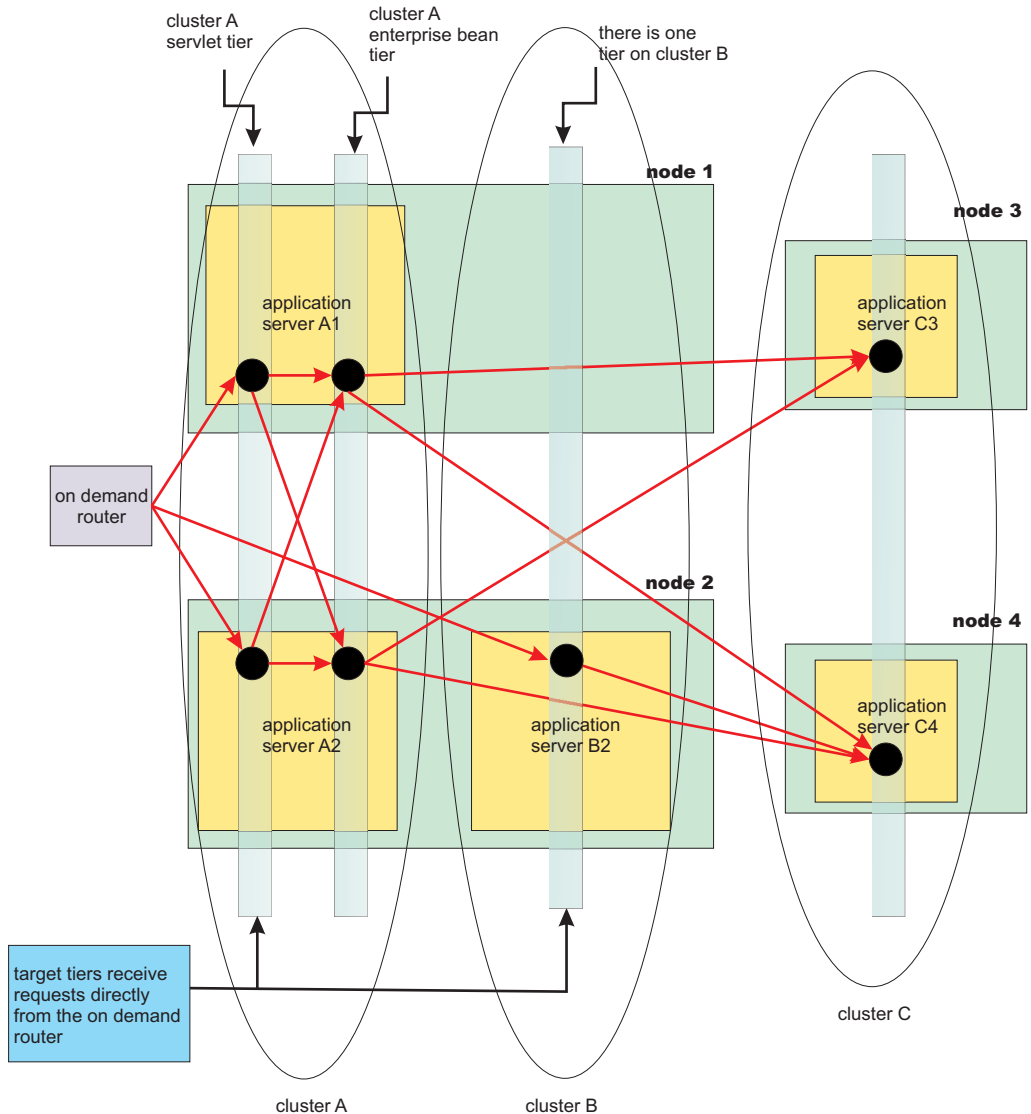


Figure 23. Multiple tiers of processing

In the preceding diagram, there are four nodes that have application servers that are members of a cluster. Each application server has one or more areas of processing, represented by black circles in the diagram.

Cluster A is made of WebSphere Application Server application servers. Server A1 and server A2 are both members of the cluster. Servlets and enterprise beans are deployed on the cluster. The servlets send work to the enterprise beans.

Each tier is specific to one deployment target. There are two tiers in cluster A. Cluster B and cluster C have one tier each. The servlet tier in cluster A and the only tier in cluster B are *target tiers*. A target server, target node, target cluster, target cell, or target tier receives requests directly from the on demand router (ODR).

The ARFM applies control at the entry to the system. In this case, the entry is the ODR. The ARFM does not measure traffic to or from tiers that are any deeper than the first tier. You must explicitly configure non-target tiers.

Restriction: The work profiler cannot determine good work factors when a node hosts both target and non-target tiers. The ARFM sends the application placement controller speed requests for the target clusters only. Each target cluster should have only one processing tier and should run on nodes that host target clusters only. Other nodes in your configuration can host non-target clusters that each contain multiple tiers.

For more information about configuring multiple tier configurations, read about routing requests to nodes that are not running Intelligent Management.

Configuring work factors in multiple tier configurations

Use this task to compute and configure work factors for your multiple tier configuration.

Before you begin

- Install the product.
- Install applications that are operational under a workload.

About this task

A work factor exists for every combination of transaction class, target web module, and processing tier. The work factor describes how heavily a request of the given transaction class loads the processing tier. You can define work factors at varying levels of granularity. The autonomic request flow manager (ARFM) uses work factors at the level of service class, target deployment target, and processing tier. You can define work factors at a variety of levels for any processing tier that is not a target tier or is not the one and only processing tier in the target module.

In a configuration that has multiple tiers, the work profiler automatically computes work factors for the target tier, which communicates directly with the on demand router (ODR). For any tiers that are deeper than the target tier, you must define work factors. If your deployment target contains both a target tier and a non-target tier, you must configure the work factors for both tiers because the work profiler cannot automatically compute work factors in that situation. You can compute the work factor by dividing the average processor utilization by the average number of executing requests per second. This task describes how to find these values and configure the work factor for your multiple tier configuration.

Procedure

1. Generate traffic for a transaction class and module pair. You can generate traffic by using an application client or a stress tool.
2. Monitor processor utilization in your configuration. Determine an average processor utilization. You need the processor utilization of all the machines that are involved in serving your traffic, and all machines that have performance interactions with them, to be at the configured limit that is defined with the **Maximum CPU utilization** property on the **Operational policies > Autonomic managers > Autonomic request flow manager** panel. Disable all the autonomic managers so that you can ensure the system does not make changes while you take the processor utilization measurement.
 - The application placement controller: Disable the application placement controller by putting it in manual mode. Click **Operational policies > Autonomic managers > Application placement controller**. Select the **Enable** check box to disable the application placement controller.
 - The autonomic request flow manager: You can set the `arfmManageCpu` custom property to false at the cell level to disable the ARFM.
 - Dynamic workload management: Disable dynamic workload management for each dynamic cluster. Click **Servers > Dynamic clusters > *dynamic_cluster_name* > Dynamic WLM**. Select the **Dynamic WLM** check box to disable dynamic workload management.

If you disable the autonomic managers, you can add processor load through background tasks. Use an external monitoring tool for your hardware.

3. Using the runtime charting in the administrative console, monitor the number of requests per second (throughput). Click **Runtime operations > Runtime topology**. You can view the number of requests per second.
4. Compute the work factor for the deployment target. Use the following equation to calculate the work factor:

$$\text{work factor} = (\text{normalized CPU speed}) * (\text{CPU utilization}) / (\text{number of requests per second, measured at entry and exit of the target tier})$$
5. Configure the work factor in the administrative console. You set the custom property on the deployment target, for example, a cluster of servers, or a stand-alone application server. For more information about the overrides that you can create with the `workFactorOverrideSpec` custom property, read about *autonomic request flow manager advanced custom properties*.
 - a. Define a case for each tier in the deployment target. Each case is separated by a comma, and contains a pattern that is set to a value that is equal to the work factor that you calculated. The pattern defines the set of service classes, transaction classes, applications, or modules that you can override for the particular tier, as shown in the following example:


```
service-class:transaction-class:application:module:[tier, optional]=value
```

You can specify a wild card for any of the service class, transaction class, application, or module by entering a * symbol. Each pattern can include at most one application, at most one module, at most one service class, and at most one transaction class. The tier is optional, and represents the deployment target name and relative tier name. Set the value to a work factor override number or to none to define no override.

In the following examples, work factor override values are set for two-tier configurations:

- Set an override value to 100 for the one and only processing tier in the target cluster:


```
*:*:*:*100
```
 - Set an override value to none for the first tier of the `MyDynamicCluster` cluster. Set an override value to 100 for the second tier of the `MyDynamicCluster` cluster in the default cell:


```
*:*:**=none,*:*:*:*MyDynamicCluster+2=100
```
 - Set an override value to none for the first tier. Set an override value to 0.7 for the CICS+1 tier in the DbCe1 cell:


```
*:*:**=none,*:*:*:*../DbCe1/CICS=0.7
```
- b. Create the custom property in the administrative console. Click **Servers > Dynamic clusters > dynamic_cluster_name > Custom properties > New**. Set the name of the property to `workFactorOverrideSpec`. Set the value of the property to the string you created in the previous step.
 - c. Save your configuration.

Results

Work factors are configured to override the work factor values that are created by the work profiler and support performance management of more than one tier.

What to do next

Repeat these steps for each transaction class module and non-target tier node pair. You also must configure the node speed for each external node. For more information about configuring the node, see *configuring node computing power*.

Overriding work factor estimates:

Use this task to override the values that are computed by the work profiler.

Before you begin

If you have already specified a speed factor override, you do not need to configure a work factor override. For more information about speed factors, read about configuring work factors in multiple tier configurations.

About this task

The work profiler computes a work factor for each transaction class module (TCM). Specifying the override depends on the version of the product that you are using. You can use work classes to classify work into transaction classes.

When you are using work classes, an XML element does not exist for each TCM. In this case, the work factor overrides related to a given deployment target (cluster, or unclustered server) can be specified in a custom property that is attached to that deployment target.

Use the `workFactorOverrideSpec` custom property. The value for this custom property is a string that contains a set of matching rules, or cases, that define the work factor override, either a value or none, for every TCM of that deployment target. The syntax of that string is the same as the syntax of the speed factor overrides, except that you cannot specify a tier for a work factor override case. For more information about speed factors, read about configuring work factors in multiple tier configurations. A work factor is in units of millions of CPU clock cycles on a standard kind of machine, and describes the average amount of work that must be done on the managed tier for a single request of that TCM.

Procedure

1. Reduce the number of back end nodes in the node group to one back end node in the node group panel in the administrative console.
2. Using a workload generator, send traffic to one TCM only. Generate enough traffic to load the back end node to above 70% utilization.
3. Record the average values for the steady state throughput and corresponding CPU utilization. You can get those values from the runtime topology view of the admin console. To view the runtime topology, click **Runtime operations > Runtime topology**.

4. Compute the work factor value for the transaction class module. Use the value from the following equation:

$$\text{work_factor} = (\text{CPU_Utilization} * \text{node_speed}) / \text{Throughput}$$

In this case, `Node_speed` equals the value of the `node.speed` node custom property. For more information about setting the `node.speed` custom property, read about configuring node computing power. It is assumed that all the nodes in the node group are homogeneous, with equal node speeds.

5. Compose a string to override the work factor for the deployment target. The pattern defines the set of service classes, transaction classes, applications, or modules that you can override. The pattern is :

```
service-class:transaction-class:application:module = value
```

You can separate each work factor with a comma, or specify a wild card for any of the variables by using an asterisk. Set the value variable to the computed work factor value. For more information about the grammar for work factor override specifications, read about autonomic request flow manager advanced custom properties.

6. Create the custom property.
 - a. Click **Custom properties > New** in the deployment target.
 - b. Enter the **Name** of the custom property as `workFactorOverrideSpec`.
 - c. The **Value** of the custom property is equal to the string composed in the previous step to override the work factor.

What to do next

Configure the speed of the external nodes. For more information about setting the `node.speed` custom property, read about configuring node computing power.

Configuring node computing power:

You must configure the node computing power if you are using speed factor or work factor overrides to configure your multiple tier configuration.

Before you begin

Configure work factor or speed factor overrides. For more information, read about configuring work factors in multiple tier configurations and overriding work factor estimates. You must know which kind of node you are configuring:

- A *managed node* is a node with an application server and a node agent that belongs to a cell. Managed nodes can have custom properties defined.
- An *unmanaged node* is a node that is defined in the cell topology that does not have a node agent that manages the process. An unmanaged node is typically used to manage Web servers. Unmanaged nodes are defined in the administrative console and can have custom properties defined.
- A node that is *not managed and not unmanaged* does not have any information stored in the configuration files. You cannot define custom properties for nodes that are not managed and not unmanaged.

About this task

Define node computing power when you are using speed factor or work factor overrides to define your multiple tier configuration.

Procedure

- If your node is managed or unmanaged, specify the node speed using the `node.speed` custom property.
 1. In the administrative console, click **System administration > Nodes > *node_name* > Custom Properties > New**.
 2. Enter the **Name** as `node.speed`.
 3. Enter the **Value**. The value equals the node speed in MHz. In a multi-processor node, multiply the processor speed by the number of processors on a multi-processor node.
- If your node is not managed and not unmanaged, configure the `external.placement` custom property. For more information about the `external.placement` custom property, read about autonomic request flow manager advanced custom properties.
 1. Compose a string that defines a case for each external node that is not managed and not unmanaged. Each case is separated by a semicolon. Each case consists of information about the node that is defined in the following pattern:

```
nodeName:nodeSpeed:plmtlist
```

The `nodeName` variable is the name of the node. The `nodeSpeed` variable is the speed of the node. The `plmtlist` is a placement list of deployment targets. Separate the deployment targets with commas. For example, you have the DBA and DBB unmonitored nodes in the SysX cell. The DBA node has a power of 4.7. The DBB node has a power of 2.7. There are two external clusters in the SysX cell that are named DB1 and DB2. The DB1 cluster is placed on the DBA node only. The DB2 cluster is placed on both the DBA node and the DBB node. Use the following string to define this configuration:

```
../SysX/DBA:4.7:DB1,DB2; ../SysX/DBB:2.7:DB2
```

2. Using the string that describes the external nodes, create the external.placement custom property. In the administrative console, click **System administration > Cell > Custom properties > New**. Enter the name of the custom property as external.placement. The **Value** equals the string that you created to define the external nodes.

Configuring a high availability deployment manager environment

The high availability (HA) deployment manager function can be configured to eliminate single point of failure for administrative functions in a WebSphere Application Server Network Deployment cell.

Before you begin

This feature is supported only on distributed platforms and is not available on z/OS.

Privileges for the high availability (HA) deployment manager differ, depending on the various roles. Roles include monitor, operator, configurator, and administrator. If you are a user with either a monitor or a configurator role, you can only view the HA deployment manager information. If you have the role of operator or administrator, you have all the privileges for the HA deployment manager.

Before you follow these steps ensure that the system environment has a suitable shared file system available, such as IBM SAN FS or NFS Version 4. To verify that your file system is suitable for use with the high availability deployment manager, use the file system locking protocol test for WebSphere Application Server. For more information, read about the IBM File System Locking Protocol Test for WebSphere Application Server. The shared file system requirements are the same for both the failover of transaction logs and for the high availability deployment manager.

Note: You must stop all deployment managers that are running in your environment before you can perform maintenance on the NFS drive. Use the extended repository service in conjunction with the HA deployment manager feature. In the event of a NFS failure, you can recover the latest configuration changes by using the extended repository service.

Each deployment manager must share the same master configuration repository and workspace area, which must be on the shared file system. Each deployment manager must also have read and write permissions to the shared master configuration repository and workspace area. The master configuration repository and workspace area default locations are *install_root/profiles/deployment_manager/config* and *install_root/profiles/deployment_manager/wstemp/* directories.

File sharing is typically accomplished by installing the deployment manager on the shared file system. Sharing the entire installation is not mandatory, and an alternative is to locate only the deployment manager profiles on the shared file system. In the scenarios that are described in this topic, Intelligent Management is installed on the shared file system. For scenarios in which only the deployment manager configuration repository and workspace reside on the Network Attached Storage (NAS), read about the `thexd_hadmgrAdd` command.

About this task

Set up a high availability deployment manager environment to eliminate the deployment manager as a single point of failure. When one deployment manager fails, a configured deployment manager can resume the work. For more information, read about the high availability deployment manager.

Procedure

1. Choose a topology. The HA deployment manager function can be configured in a variety of topologies. In a production environment, it is recommended that each deployment manager runs on its own computer to increase availability. If a single on demand router (ODR) is in the topology, it must run on

its own computer. To eliminate the ODR as a single point of failure, you can use multiple on demand routers in conjunction with an IP sprayer. A typical HA deployment manager environment requires at least three computers:

Computer name	Processes
A	Deployment manager
B	Standby deployment manager
C	Node agent and on demand router

2. You can either create a new cell with an HA deployment manager configuration, or convert an existing cell to an HA deployment manager configuration.

To create a new cell with an HA deployment manager configuration:

- a. Install WebSphere Application Server on the shared file system.
- b. Install Intelligent Management on the shared file system.
- c. Create a deployment manager profile using the IP address for computer A.
- d. Start the deployment manager on computer A.
- e. Create a custom profile using the IP address for computer C. Federate computer C with the deployment manager.
- f. Create an ODR on the federated custom profile.
- g. Create a deployment manager profile using the IP address for computer B.
- h. Run the `xd_hadmgrAdd` command within the deployment manager profile for computer B to convert it into a standby deployment manager. For more information, read about the `xd_hadmgrAdd` command.

To convert an existing cell to an HA deployment manager configuration:

- a. Install WebSphere Application Server on the shared file system.
 - b. Install Intelligent Management on the shared file system.
 - c. Create a deployment manager profile using the IP address for computer A and with the same cell and node name as the existing deployment manager.
 - d. Use the `backupConfig` and `restoreConfig` command line utilities to relocate the existing cell configuration to the shared file system.
 - e. Copy the `tmsStorage` folder and its contents from the existing deployment manager profile to the new deployment manager profile on the shared file system.
 - f. Ensure that the existing cell configuration contains an ODR. Create an ODR if an ODR does not exist.
 - g. Create a deployment manager profile using the IP address for computer B.
 - h. Run the `xd_hadmgrAdd` command within the deployment manager profile for computer B to convert it into a standby deployment manager. For more information, read about the `xd_hadmgrAdd` command.
3. Restart the deployment managers and on demand routers. To avoid errors, use the following start sequence:
 - a. Start the node agent for the on demand router node.
 - b. Start the on demand router.
 - c. Start the deployment managers. The first deployment manager that starts is the active deployment manager, and the other is the standby deployment manager.

If you do not follow this start sequence, you might encounter an error:

```
XHAD0005E: A Deployment Manager process running in active mode cannot be found.
```

This error indicates that the deployment manager started before the on demand router, so the environment does not know which deployment manager is active. When the on demand router is started, the primary deployment manager can be assigned.

Results

Multiple deployment managers are configured. The deployment manager is eliminated as a single point of failure because a primary deployment manager hosts the administrative function for the cell, and has backup deployment managers that are in standby node. If the primary deployment manager fails, the standby deployment managers can resume without losing work.

What to do next

After your high availability deployment manager environment is set up, you can manage the deployment managers in the administrative console. Click **System administration > Deployment manager > All deployment managers** to view and manage the configured deployment managers in your environment.

If you need to remove a high availability deployment manager from the cell, use the command line utility to remove the function. For more information about removing the high availability deployment manager from the cell, read about the `xd_hadmgrRemove` command.

Adding the signer certificate from the secondary deployment manager to the local trust store

To enable Secure Sockets Layer (SSL) in your high availability deployment manager environment, the local trust store must contain the signer certificate from the secondary deployment manager. If the trust store does not contain the signer certificate, add the certificate to the trust store to prevent errors and enable secure communication among the core group members.

About this task

To elect the secondary deployment manager to take over as the primary deployment manager when SSL is enabled in your environment, the signer certificate of the secondary deployment manager must exist in the local trust store. Specifically, the `com.ibm.ssl.trustStore` value must be set to the cell-level default trust store in the `deployment_manager_profile/properties/ssl.client.props` file. If the certificate cannot be located in the local trust store, the SSL handshake fails and you might receive the following error message:

```
CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN
"CN=xdblade36b07.rtp.raleigh.ibm.com, O=IBM, C=US"
was sent from target host:port "*:9043".
The extended error message from the SSL handshake exception is:
"No trusted certificate found".
```

Add the signer certificate from the secondary deployment manager to the local trust store to enable secure communication in your high availability deployment manager environment.

Procedure

1. In the administrative console, click **Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates > Retrieve from port**.
2. Define the following general properties to retrieve the signer certificate from the remote SSL port, and click **Retrieve signer information**:

Host Specifies the host name that you connect to when you retrieve the signer certificate from the SSL port

Port Specifies the SSL port that you connect to when you retrieve the signer certificate

SSL configuration for outbound connection

Specifies the configuration that is used to connect to the SSL port

This configuration is the SSL configuration that contains the signer certificate after you add the certificate to the trust store.

Alias Specifies the certificate alias that is used in the SSL configuration

Results

The configuration can connect to and accurately check the status of the secondary deployment manager.

The high availability deployment manager

The high availability (HA) deployment manager function is configured using a shared file system. When this configuration option is chosen, multiple deployment managers are configured. The benefit of the HA deployment manager function is that the deployment manager is no longer the single point of failure for cell administration. This is important in environments relying on automated operations, including application deployment and server monitoring.

Deployment manager overview

The deployment managers exist as peers. One is considered active, also known as primary, and hosts the administrative function of the cell, while the others are backups in standby mode. If the active manager fails, a standby takes over and is designated the new active deployment manager. A command line utility is provided to clone the original cell deployment manager into additional deployment managers. Each deployment manager is installed and configured to run on a different physical or logical computer. The deployment managers need not be hosted on homogenous operating platforms, although like platforms are recommended. Each deployment manager shares the same instance of the master configuration repository and workspace area. These must be located on a shared file system.

The file system must support fast lock recovery. The IBM® General Parallel File System™ (GPFS™) is recommended, and the Network File System Version 4 (NFS) is also an option. If you use the high availability deployment manager on AIX Version 5.3 and are using NFS Version 4, you must have `bos.net.nfs.client` Version 5.3.0.60 or later.

Note: You must stop all deployment managers that are running in your environment before you can perform maintenance on the NFS drive. Use the extended repository service in conjunction with the HA deployment manager feature. In the event of a NFS failure, you can recover the latest configuration changes by using the extended repository service.

Normal operation includes starting at least two deployment managers. A new 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 on demand router (ODR) is configured with the communication endpoints for the administrative console, the `wsadmin` tool, and scripting. The ODR recognizes which deployment manager instance is active and routes all administrative communication to that instance. The HA deployment manager function supports only use of the JMX SOAP connector. The JMX RMI connector is not supported in this configuration.

Configuration

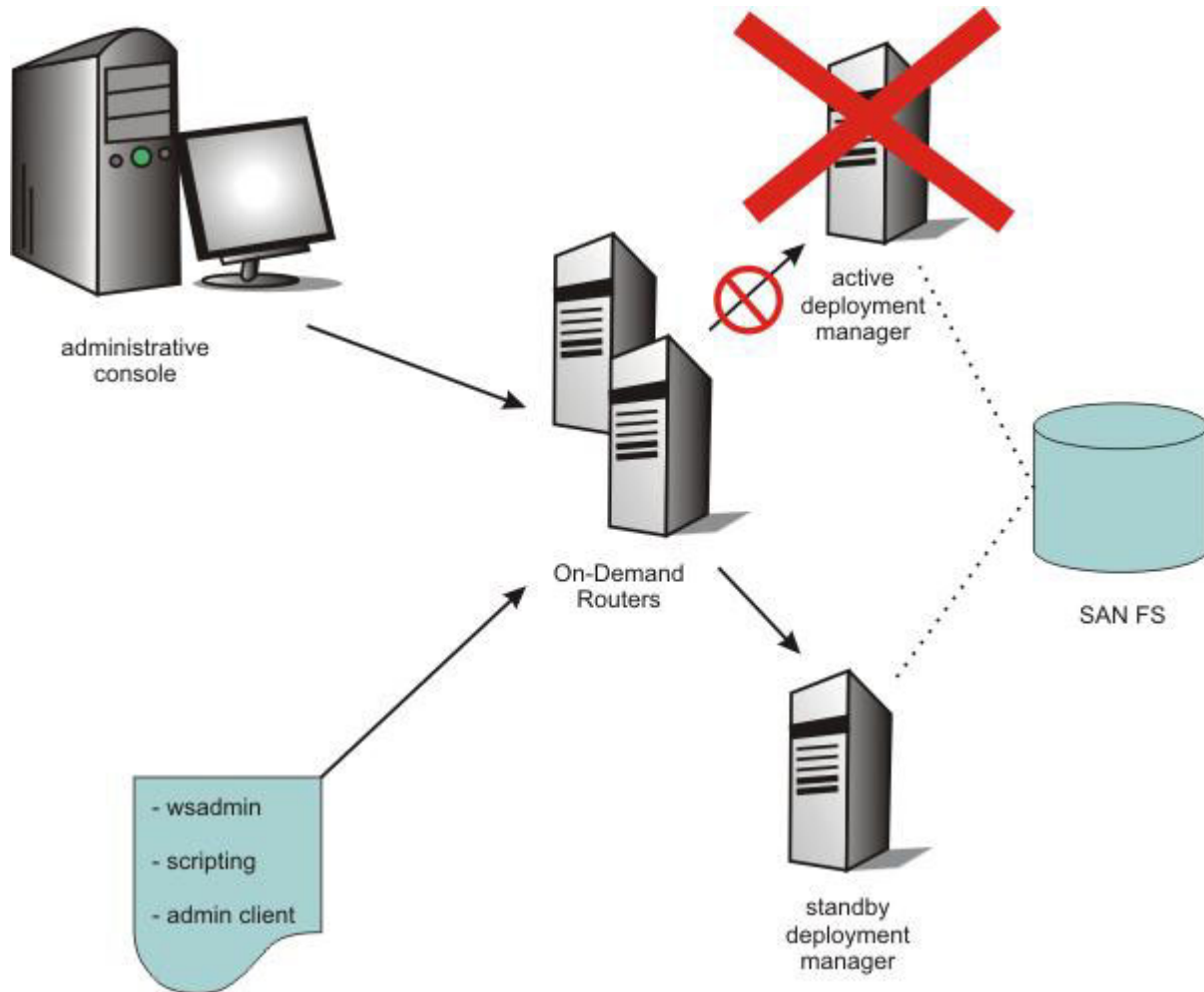
The deployment managers are initially configured into the same core group. Configuring the deployment managers in the same core group is important so that the routing information that is exposed to the ODR is consistent across all the deployment managers. If the deployment managers are placed into separate core groups, the core groups must be connected with a core group bridge.

A typical HA deployment manager configuration consists of two deployment managers that are located on separate workstations. The deployment managers share a master repository that is located on a SAN FS. 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 highly available 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, because the ODR automatically recognizes the election of the new active deployment manager and reroutes administrative requests to the new active deployment manager. Note that there is a sub 1 minute period of time where the deployment manager will not be available until failover to the secondary is complete.

Failover to the new active deployment manager is depicted in the following diagram:



While the HA deployment manager component is able to detect deployment manager failure and initiate takeover, there are edge conditions where each deployment manager could temporarily believe it is the active deployment manager. To prevent this situation from occurring, 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 will take 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.

hadmgrAdd command

The **hadmgrAdd** command incorporates a new deployment manager profile into a cell. The new deployment manager profile is converted to a highly available (HA) deployment manager peer in the same cell as the existing deployment manager profile.

Run the **hadmgrAdd** command within the new deployment manager profile that is being added. The HA deployment manager function supports use of the JMX SOAP connector only. The JMX RMI connector is not supported in this configuration.

trns: The Intelligent Management command that equates to **hadmgrAdd** is **xd_hadmgrAdd**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **xd_hadmgrAdd** command, which operates the same as the **hadmgrAdd** command.

Usage

Distributed operating systems

```
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_port  
-proxyServerHttpPort HTTP_portHTTP_secure_port  
[-uniquePort] [-user user_name] [-password password]  
[-quiet] [-logfile file_name] [-replaceLog] [-trace] [-help]
```

The **proxyServerJmxSoapAddress** and **proxyServerHttpPort** parameters are required only when you are creating the first standby deployment manager.

Parameters

The following options are available for the **hadmgrAdd** command:

- hostname <host name>**
Specifies the host name that is used to connect to the existing deployment manager.
- port <port>**
Specifies the default SOAP port that is used to connect to the existing deployment manager.
- configRoot <fully qualified path>**
Specifies the fully-qualified configuration path.
- workspaceRoot <fully qualified path>**
Specifies the fully-qualified workspace path.
- proxyServerJmxSoapAddress <host:port>**
Specifies the host name and IP address of the on demand router (ODR) and the SOAP_CONNECTOR_ADDRESS port for the ODR. Use this port to run wsadmin scripts.
- proxyServerHttpPort <port> <secure_port>**
Specifies the HTTP ports for the proxy server. Select two free ports on the ODR and IP sprayer host. These ports should be used to access the administrative console by way of the ODR.
- user <user name>**
Specifies the user name that is used to connect to the existing deployment manager.
- password <password>**
Specifies the password that is used to connect to the existing deployment manager.
- quiet**
Suppresses the progress information that the **hadmgrAdd** command prints in normal mode.
- logfile <filename>**
Specifies the location of the log file to which information gets written. By default, the **hadmgrAdd.1** log file is created in the **logs** directory of the profile for the node that is being added.

-replaceLog

Replaces the log file instead of appending to the current log. By default, the **hadmgrAdd** command appends to the existing trace file. This option causes the **hadmgrAdd** command to overwrite the trace file.

-trace

Generates additional trace information in the log file for debugging purposes.

-uniquePort

Checks for port conflict. If the new port is conflict with the existing ports, the new port increments by one until a free port is found. This process is not necessary if unique ports were assigned to the deployment manager profile when it was created.

-help

Displays syntax help.

Example

1. Start with an existing WebSphere Application Server Network Deployment cell that is installed and configured with the following directory structure:

- WebSphere Application Server home directory: `app_server_root`
- Existing deployment manager profile: `/shared/profiles/PrimaryManagerNode`

where the `/shared mountpoint` is on a SAN FS device.

2. Run the Profile Management Tool plug-in or the **wasprofile** command line utility to create another deployment manager profile associated with this same Network Deployment installation. Create the profile on the shared disk, `/shared mountpoint` for this example.

You must specify the correct host name for this deployment manager to listen on. You can specify the explicit port numbers for your standby deployment manager when you create the profile or you can specify the **-uniquePort** parameter when you run the **hadmgrAdd** command, as shown in this example. You can specify any value for the cell name, because it is changed to match the cell name to which this HA deployment manager peer is added during the conversion process. You must specify a node name that is unique in the cell to which this HA deployment manager peer is added.

You have the following directory structure after you create the profile:

- WebSphere Application Server home directory: `app_server_root`
- Existing deployment manager profile: `/shared/profiles/PrimaryManagerNode`
- New deployment manager profile: `/shared/profiles/StandbyManagerNode`

3. Run the **hadmgrAdd** command to convert the new deployment manager profile into an HA deployment manager peer in the same cell in which the existing deployment manager profile exists.

```
hadmgrAdd.sh|bat -hostname a.a.a.a -port 8879 -configRoot /shared/profiles/PrimaryManagerNode/config
-workspaceRoot /shared/profiles/PrimaryManagerNode -proxyServerJmxSoapAddress x.x.x.x:8880
```

```
-proxyServerHttpPort 9060 9043
-user wsadmin
-password *****
-uniquePort
```

If you install the centralized installation manager repository, and after you create the second deployment manager profile, change the value of the `CENTRALIZED_INSTALL_REPOSITORY_ROOT` property in the `app_server_root\properties\cimgr.props` file from `c:\ProgramFiles\IBM\WebSphere\AppServer\repository_folder` to `${app_server_root}/repository_folder`.

When you use the **hadmgrAdd** command, the following changes are made to your cell configuration:

- 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.
- 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.

- The ODR is configured with the `wc_admin` and `ws_admin_secure` HTTP transport channels and the `JMX_SOAP_PROXY_ADDRESS` endpoint.
- The `JMX_SOAP_CONNECTOR` addresses of the deployment managers point to the `JMX_SOAP_PROXY_ADDRESS` of the ODR.
- The deployment managers have an added `JMX_SOAP_PROXY_ADDRESS` that holds the same configured host or port as their original `JMX_SOAP_ADDRESS`.

As a result of the configuration change, the ODR hosts the logical endpoints for the deployment manager and proxies communication requests for the deployment manager to the active deployment manager.

For example in a non-HA ODR configuration, the deployment manager configured host name points to the ODR.

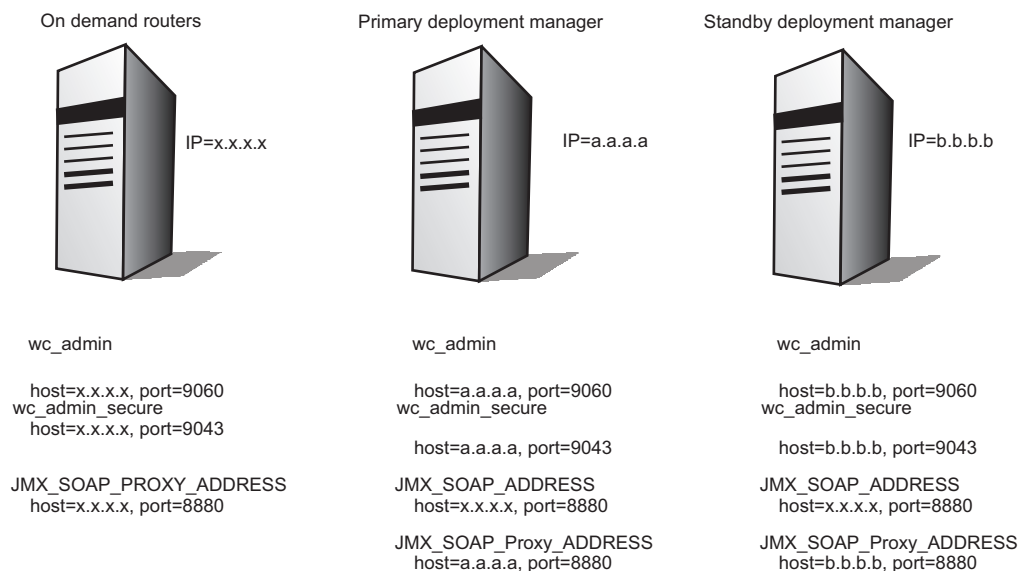


Figure 24. ODR configuration without high availability

To eliminate the ODR as a single point of failure, set up a HA ODR configuration that has at least two on demand routers receiving traffic from an internet protocol spraying device. In this case, the deployment managers `JMX_SOAP_ADDRESS` configured host name points to the device.

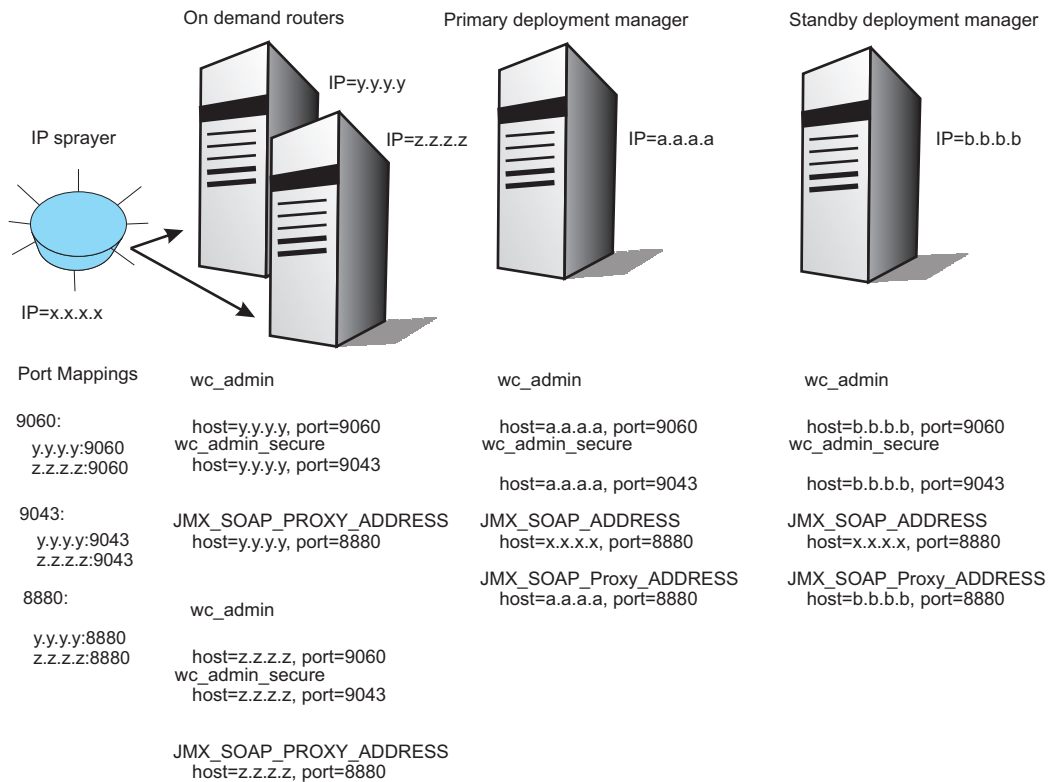


Figure 25. High availability ODR configuration with two ODRs that includes an IP sprayer

hadmgrRemove command

The **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.

trns: The Intelligent Management command that equates to **hadmgrRemove** is **xd_hadmgrRemove**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **xd_hadmgrRemove** command, which operates the same as the **hadmgrRemove** command.

Syntax

Distributed operating systems **z/OS** The command syntax is as follows:

```
hadmgrRemove -hostname (primary_dmgr_host) [-port (primary_dmgr_port)]
[-user (uid)] [-password (pwd)] [-quiet] [-logfile (filename)]
[-replaceolog] [-trace] [-help]
```

Parameters

The following options are available for the **hadmgrRemove** command:

-hostname <host name>

Specifies the host name that is used to connect to the existing deployment manager.

-port <port>

Specifies the port that is used to connect to the existing deployment manager.

-user <user>

Specifies the user name that is used to connect to the existing deployment manager.

-password <password>

Specifies the password that is used to connect to the existing deployment manager.

-quiet

Suppresses the progress information that the **hadmgrRemove** command prints in normal mode.

-logfile <filename>

Specifies the location of the log file to which information gets written. By default, the log file is called **hadmgrRemove.log** and is created in the **logs** directory of the profile for the node being added.

-replacelog

Replaces the log file instead of appending to the current log. By default, the **hadmgrRemove** command appends to the existing trace file. This option causes the **hadmgrRemove** command to overwrite the trace file.

-trace

Generates additional trace information in the log file for debugging purposes.

-help

Displays syntax help.

Using centralized logging to diagnose problems

You can use centralized logging to easily enable mustGather traces, perform a per-request trace analysis, and follow the flow of a request through both the ODR and application server tiers.

Before you begin

- Centralized logging is a tool to help you collect the data necessary for troubleshooting and diagnosing a problem. However, for performance reasons, the features of centralized logging are not enabled all the time. Therefore, you must enable the centralized logging features when you anticipate capturing a particular problem.
- When you run the **mustgather.py** script, trace specifications are overwritten. If you want to restore the trace specifications, save the trace strings before you run the script.
- If you use the remote log collection feature of the **mustgather.py** script, ensure that your target file system has the appropriate amount of space necessary to contain the gathered log files.

About this task

Using centralized logging, you can enable tracing based on the type of problem that you experience, for example, 503 HTTP response codes. With centralized logging, specific types of mustGather documents can be identified by predefined strings. Some examples of predefined mustGather documents include application editioning, 404 error and 503 response codes, health monitoring, and more.

You can also enable rule-based tracing which allows you to trace requests based on specific rules.

Procedure

- Run the **mustgather.py** script to enable tracing, collect mustGather documents, and disable tracing. All the necessary logs within the cell are compiled on the system where the script runs. The supported commands are included in the following list:

enable [mustgatherType]

Enables tracing

collect [mustgatherType] [destination]

Collects a specific type of mustGather document

disable [mustgatherType]

Disables tracing

1. Determine the type of mustGather document that you need. The supported types of mustGather documents are included in the following list:

404
404 HTTP response code

503
503 HTTP response code

504
504 HTTP response code

agent
node agent

appedition
Application edition manager

arfm
Autonomic request flow manager

dc Dynamic cluster

hadmgr
High availability deployment manager

hmm
Health monitoring

odr
On demand router

operations
Visualization issues with the Extended Deployment and Operations tabs

reports
Visualization issues with the Reports tab

reportsPerf
Visualization issues with performance data that is displayed on the Reports tab

repository
Extended Repository Service

sip
SIP request routing

2. Run the **enable** command with the appropriate mustGather type specified to set the appropriate tracing on all relevant servers in the cell.

```
wsadmin -lang jython -f c:\WebSphere\AppServer\bin\mustgather.py enable 404
```

3. Recreate the desired scenario.

4. Run the **collect** command with the appropriate mustGather type and a destination file specified to collect the local and remote mustGather documents.

```
wsadmin -lang jython -f c:\WebSphere\AppServer\bin\mustgather.py collect 404  
"c:\mydocs\collection.zip"
```

5. Run the **disable** command with the appropriate mustGather type specified. Running this command sets the trace to *=info on all relevant servers in the cell.

```
wsadmin -lang jython -f c:\WebSphere\AppServer\bin\mustgather.py disable 404
```

- Run the setupReqBasedTracing.py script to enable or disable request-based tracing rules. The supported commands are included in the following list:

enableReqBasedTracing

Sets up a request-based tracing rule. A rule consists of an expression and identifier called a rule ID. Optionally, a rule can also contain ODR trace and application server trace strings. Start and end markers are placed in the log files for requests that match one or more rules.

The ODR logs a start marker when a request that matches one or more rules arrives and logs an end marker before the request is dispatched to the backend application server. The ODR also logs a start marker when a response is received from the application server and an end marker after response is sent back to the user. The application server logs a start marker when a matched request arrives from the ODR and an end marker before a response is sent to the ODR. These markers make it possible to find a particular request, or set of requests, and to correlate them with processing of that request on the application server.

listRuleIDs

Lists all the rules. This command will output all the rules that are set on all the ODRs. After an ODR is restarted, the rules must be recreated.

disableReqBasedTracing

Disables a request-based tracing rule.

1. Enable request-based tracing.

```
./wsadmin.sh -lang jython -f setupReqBasedTracing.py enableReqBasedTracing  
-ruleExpression:<expression> -odrTraceSpec:<trace strings>  
-appServerTraceSpec:<trace string> -ruleID:<rule ID>
```

where

-ruleExpression:<expression>

Specifies an expression that is used to match the requests. (Required)

-odrTraceSpec:<trace string>

Specifies an ODR trace string that is set at run time for the requests that match the specified expression. If the parameter is not specified, the trace specification is not dynamically set. (Optional)

-appServerTraceSpec:<trace string>

Specifies an application server trace string that is set at run time for the requests that match the specified expression. If the parameter is not specified, the trace specification is not dynamically set. (Optional)

-ruleID:<rule ID>

Specifies the ID for the request-based tracing rule. If the parameter is not specified, a rule ID is generated by the script in the form of ruleID-<time stamp>. (Optional)

2. List all the rules.

```
./wsadmin.sh -lang jython -f setupReqBasedTracing.py listRuleIDs
```

3. Disable request-based tracing.

```
./wsadmin.sh -lang jython -f setupReqBasedTracing.py disableReqBasedTracing  
-ruleIDs:<rule ID1>,<rule ID2>...,<ruleIDn>
```

where

-ruleIDs:<rule ID1>,<rule ID2>...,<ruleIDn>

Specifies a list of rule IDs to disable. (Required)

For more information on rule expressions, read about HTTP operands.

Monitoring operations

You can easily monitor the status of your environment. With the operational summaries, you can identify where applications, servers, or autonomic managers are running in your environment, the health of your environment, and if your environment is performing according to your service-level agreement.

Before you begin

For more information about configuring your cell to autonomously manage your environment, read about preparing the hosting environment for dynamic operations.

Procedure

- Create and manage reports.
- Log historic data by configuring the visualization data service.
- Manage tasks.

Runtime operations overview

With Intelligent Management, you can manage complex system operations with real time, meaningful visualization tools. Gradually controlled implementation of autonomic capabilities helps you reduce your resource management cost.

Operational alerts

Operational alerts inform you of the current state of your environment, including any issues, so that you can take action if necessary. For example, you might see an operational alert when a service policy is breached. For notification about issues that are continuing to occur over multiple intervals, runtime tasks are generated.

Reports

You can use customized charting to show if goals are being met. To benefit from this virtualized environment, you must know how your applications are performing. Dynamic charting provides a visual perspective of application performance. Specifically, statistics such as availability, response time, traffic, and throughput are supported. A wide range of options from which you can create various charts is also provided.

You can organize your charts into *chart groups* to easily access charts from any reports view.

You can move charts to a new window to continue viewing the chart while performing other tasks in the administrative console. You can also move the chart back to the original chart group.

You can also view reports in the **Reports** tab of the following settings panels:

- **Applications**
- **Dynamic clusters**
- **Clusters**
- **Servers**
- **Service policies**

Runtime operations reports use Scalable Vector Graphics (SVG) to display the data. Microsoft Internet Explorer does not have an SVG viewer installed by default. You can install a SVG plug-in or configure runtime operations to display JPEG images. For more information, read about creating and managing reports.

The runtime operations reports are blank until you begin running an active load against your cell.

Dashboard

The dashboard displays a high level summary of your overall environment. You can use the dashboard preferences to configure the information that displays in the dashboard, such as the default reports or the data from a specified chart group.

Runtime summaries

You can view the runtime information for the on demand routers, nodes, core groups, applications, deployment targets, service policies, and core components in your environment. Core components include a variety of autonomic controllers and managers. The runtime summaries include a list of instances for the particular resource type, status, stability and other information.

The charts that are included in the applications, deployment targets, and service policies summary views do not automatically refresh. To update the data, you can click the refresh icon on the chart. These charts display the average response time at a certain point in time. The default number of data sets that are displayed is 5. You can, however, configure custom properties to specify that a unique number of data sets be displayed instead. For more information, read about runtime operations custom properties.

If you are not running any work load requests through the on demand router (ODR) to your applications, no data is displayed in the chart. Also, if you are charting service policy related metrics, no goal lines for the data set are displayed in the key. After the ODR sends the work load requests, the chart is automatically updated and the statistics are displayed. However, the key table is not automatically updated. Refresh the panel to view the goal lines that are added to the data set.

To access the runtime summaries, click **Runtime operations** in the administrative console.

Operations tabs

The operations tabs show the overall status of the selected item in the following settings panels:

- **Applications**
- **Dynamic clusters**
- **Clusters**
- **Servers**
- **Core groups**

Creating and managing reports

With reports, you can view the performance of your virtualized environment. You can view statistics on availability, response time, traffic, throughput.

Before you begin

- Your servers, clusters, and applications must be running to display data in the charts.
- **Windows** By default, runtime operations reports use Scalable Vector Graphics (SVG) charts. Microsoft Internet Explorer does not have an SVG viewer installed by default. You can install the SVG viewer plug-in or update the reports preferences to display JPEG images. To update the reports preferences, click **Runtime operations > Reports > Reports preferences**. Edit the **Default chart format** field.

About this task

Reports are charts that show runtime data. You can use this data to monitor your environment, and take correctional actions when necessary.

You can view reports which display live data, or you can use charts displaying historic data logged over a period of days, weeks, months, years. In order to use historic charting, configure and enable the visualization data service. For more information, read about configuring the visualization data service. For historic charting, use the sliding bar to adjust the percentage of time to view. You can focus on a specific time interval and include more data in the chart.

Note: To observe historic data over a long period of time, increase the maximum heap size of the deployment manager. The default maximum heap size is 256 MB. To increase the performance,

and to prevent heap dumps, go to **System administration > Deployment manager > Java and process management > Process definition > Java Virtual Machine** and set the maximum heap size to a higher value, for example 2048 MB.

Procedure

1. Open the **Reports** tab. In the administrative console, you can access the **Reports** tab from:
 - **Runtime operations > Reports**
 - **Servers > All servers > *server_name* > Reports**

Restriction: The **Reports** tab is not available for on demand router (ODR) servers.

- **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name* > Reports**
 - **Servers > WebSphere Application server clusters > *cluster_name* > Reports**
 - **Applications > All application > *application_name* > Reports**
 - **Operational policies > Service policies > *service_policy_name* > Reports**
2. Act on operational alerts. Operational alerts display the status of resources in your environment. The name of the specific resource is highlighted. Go to the configuration panel or view the chart for that resource to take corrective actions.
 3. Configure report and chart preferences. Expand **Reports preferences**. The preferences are global and apply to any new charts you create. However, the preference change does not apply to existing charts. To change the preferences for an existing chart, click **Preferences** on the chart.
 4. Add a chart. Click **Open a new chart tab**. A new tab opens with a blank chart. Click **Add data...** to specify the data set and metrics that you want to monitor. If you accessed the **Reports** tab through the **Runtime operations > Reports** panel, you can specify the scope of the chart. Click **Change scope...**
 5. Create, access, and remove chart groups. Chart groups are global and can be accessed from any **Reports** tab.
 - **Create:** If you have a specific set of charts in the tabs that you want to save, then type a name in **Save current group of chart tabs configuration as a chart group**, and click **Save**.
 - **Access:** To access the chart group from any reports panel later, go to **Saved chart groups**, and click the name of the chart group.

You can also open a chart in a new window, so that you can continue doing other tasks in the administrative console while still monitoring your chart. Click **View chart in new window**. To return the chart from the new window to the chart group, click **View chart in chart group**.
 - **Remove:** To remove a chart group, select the chart group from **Saved chart groups**, and click **Remove chart group**.

What to do next

If your operational alerts are taking a long time to load, you can configure custom properties to change how the alerts are loaded while the cache initializes.

You can configure custom properties to specify the maximum number of data sets to display, or to limit what cell information is displayed.

Creating and managing reports

With reports, you can view the performance of your virtualized environment. You can view statistics on availability, response time, traffic, throughput.

Before you begin

- Your servers, clusters, and applications must be running to display data in the charts.

- **Windows** By default, runtime operations reports use Scalable Vector Graphics (SVG) charts. Microsoft Internet Explorer does not have an SVG viewer installed by default. You can install the SVG viewer plug-in or update the reports preferences to display JPEG images. To update the reports preferences, click **Runtime operations > Reports > Reports preferences**. Edit the **Default chart format** field.

About this task

Reports are charts that show runtime data. You can use this data to monitor your environment, and take correctional actions when necessary.

You can view reports which display live data, or you can use charts displaying historic data logged over a period of days, weeks, months, years. In order to use historic charting, configure and enable the visualization data service. For more information, read about configuring the visualization data service. For historic charting, use the sliding bar to adjust the percentage of time to view. You can focus on a specific time interval and include more data in the chart.

Note: To observe historic data over a long period of time, increase the maximum heap size of the deployment manager. The default maximum heap size is 256 MB. To increase the performance, and to prevent heap dumps, go to **System administration > Deployment manager > Java and process management > Process definition > Java Virtual Machine** and set the maximum heap size to a higher value, for example 2048 MB.

Procedure

1. Open the **Reports** tab. In the administrative console, you can access the **Reports** tab from:
 - **Runtime operations > Reports**
 - **Servers > All servers > *server_name* > Reports**

Restriction: The **Reports** tab is not available for on demand router (ODR) servers.

 - **Servers > Clusters > Dynamic clusters > *dynamic_cluster_name* > Reports**
 - **Servers > WebSphere Application server clusters > *cluster_name* > Reports**
 - **Applications > All application > *application_name* > Reports**
 - **Operational policies > Service policies > *service_policy_name* > Reports**
2. Act on operational alerts. Operational alerts display the status of resources in your environment. The name of the specific resource is highlighted. Go to the configuration panel or view the chart for that resource to take corrective actions.
3. Configure report and chart preferences. Expand **Reports preferences**. The preferences are global and apply to any new charts you create. However, the preference change does not apply to existing charts. To change the preferences for an existing chart, click **Preferences** on the chart.
4. Add a chart. Click **Open a new chart tab**. A new tab opens with a blank chart. Click **Add data...** to specify the data set and metrics that you want to monitor. If you accessed the **Reports** tab through the **Runtime operations > Reports** panel, you can specify the scope of the chart. Click **Change scope...**
5. Create, access, and remove chart groups. Chart groups are global and can be accessed from any **Reports** tab.
 - **Create:** If you have a specific set of charts in the tabs that you want to save, then type a name in **Save current group of chart tabs configuration as a chart group**, and click **Save**.
 - **Access:** To access the chart group from any reports panel later, go to **Saved chart groups**, and click the name of the chart group.

You can also open a chart in a new window, so that you can continue doing other tasks in the administrative console while still monitoring your chart. Click **View chart in new window**. To return the chart from the new window to the chart group, click **View chart in chart group**.

- **Remove:** To remove a chart group, select the chart group from **Saved chart groups**, and click **Remove chart group**.

What to do next

If your operational alerts are taking a long time to load, you can configure custom properties to change how the alerts are loaded while the cache initializes.

You can configure custom properties to specify the maximum number of data sets to display, or to limit what cell information is displayed.

Configuring the 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. By using the visualization data service, you can log historical data, calculate charge back values, or perform capacity planning.

Before you begin

You must use the deployment manager to implement this feature. Ensure that if you are using multiple core groups, they are correctly bridged.

If you are a user with either a monitor or an operator administrative role, then you can only view the visualization data service information. If you have a configurator administrative role, then you can change the configuration. If you have an administrator role, then you have all the privileges for the visualization data service.

About this task

Attention: You must configure the visualization data service before you enable logging. If you need to make changes to the configuration after you have enabled logging, you must restart the deployment manager after you make the configuration changes.

Procedure

1. In the administrative console, click **System administration > Visualization data service**.
2. 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 timestamp, 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'
```

You can also specify the time stamp format with the `wsadmin` tool:

```
wsadmin.sh -lang jython  
wsadmin>> vds = AdminConfig.getid("/Cell:OpsManTestCell/VisualizationDataService:/")  
wsadmin>> vd1 = AdminConfig.showAttribute(vds,"visualizationDataLog")  
wsadmin>> AdminConfig.modify(vd1,[["timestampFormat","MM.dd.yyyy hh:mm:ss:SSS aaa z"]])  
wsadmin>> print AdminConfig.show(vd1)  
wsadmin>> AdminConfig.save()
```

3. In the **Maximum file size** field, type a whole integer for the maximum file size for logs.

4. 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.
5. 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`.
6. 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. If you plan to log data for several metrics over a period longer than 1 week, increase the **Data log write interval** for better performance.
7. 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.
8. Select **Enable log** to start logging historic data.
9. If logging was enabled before you configured the visualization data service, restart your deployment manager.

Results

Operational data is exported to the file name that you specified.

What to do next

Now that you have configured the visualization data service, you can import data into an external charting program.

Task management

You can be notified about decisions that are made by autonomic managers. Notifications can represent either planned or unplanned events.

Planned events

Planned events are events for which the runtime environment has an action plan. For example, you might have a health policy defined that has an average response time that is breaching its configured limit, which might trigger an increased footprint of a dynamic cluster. If the product is operating in automatic mode, then the action plan runs, and you can view a notification of the action that was taken. In supervised mode, you can view and approve the action plan. The interaction modes can vary in your configuration. A mix of automatic and supervised mode activities by dynamic clusters and health policies can exist, for example.

Unplanned events

If an event occurs and it is not assigned to an action plan, then you can be notified that something unexpected has happened. After you receive the notification, develop a plan to correct the situation, if it is a valid issue.

Runtime tasks

A runtime task is generated when an event occurs. Runtime tasks provide information from which you can accept, deny or close the recommended action plan. Tasks that have taken action or expired tasks stay in the runtime task list for 24 hours by default. You can change this default by setting a cell level custom property. For more information on how to set the custom property, read about runtime task custom properties.

Runtime tasks reside in the deployment manager memory, and are logged in `<was_root>/profiles/<Dmgr_profile_name>/tmsStorage` as well. When the deployment manager is restarted, the deployment manager reads the runtime task entries in the `tmsStorage` log.

Event logging

You can enable logging for events. For more information, read about task management service event logger.

Notifications

You can have email notifications sent to specified users when runtime tasks occur. For more information, read about defining email notifications.

Managing runtime tasks

A runtime task is generated by a runtime component within the product. Runtime tasks provide information from which you can accept, deny or close the recommended action plan.

Before you begin

Configure your hosting environment. For more information, read about preparing the hosting environment for dynamic operations

About this task

Use runtime tasks to view and manage the recommendations that the autonomic controllers are providing to improve the health and performance of your environment. If you are running in supervised mode, you must accept, deny, or close each runtime task. In automatic mode, the autonomic controllers automatically take these actions.

A task is lost when it is sent from a node when the deployment manager is not running. The same holds true in a high availability deployment manager environment. Even though the down time is minimal between the time when an active highly available deployment manager shuts down and when the standby deployment manager becomes active, tasks are still lost. After a failover of a highly available deployment manager occurs, the executing tasks at the original active deployment manager are changed to the unknown state.

Procedure

1. From the administrative console, click **System administration > Task management > Runtime tasks**.
2. Click the task ID link to view the task target objects and corresponding monitors of a specific task. A task target object is a server, cluster, service policy, node, health policy, or application. One of these objects might be linked to configuration, performance or log monitor views in the administrative console.

With a configuration monitor, the placement controller might recommend that an instance of the TestCluster dynamic cluster start on the test4 node. The recommendation states that this is action is required for the dynamic cluster to meet its configured minimum number of running instances. You can click the link to the configuration panel for the TestCluster dynamic cluster to view its settings and verify or change the minimum number of running cluster instances.

For a performance monitor, you can click a link to a chart with data that is specified in the monitor and specific to the target object. For a log monitor, you can click the **Runtime** tab of the target object Java virtual machine logs to investigate the displayed logs.
3. To view the action plan for the task, click the task ID.
4. To act on a specific task, select the corresponding task, and from the action list select **Accept**, **Deny** or **Close**. Accepting the task commits the previewed action plan. Closing the task means that the task is successfully handled by the task management service or is manually closed. Denying the task places the task in inactive status if the task is not submitted again in the next batch of task submissions by the originating component. You can also select multiple tasks with the same actions. After you act on a task, the action list is unavailable for that task.
5. Click **Submit**.

Results

The **Runtime tasks** panel refreshes, and performs the actions that you selected to perform on each task.

Task management service event logger

You can enable an event logger that logs all Java Management Extensions (JMX) notifications from the TaskManagement MBean into its own log file.

The event logger is a valuable addition to the task management service. It logs all JMX notifications: tasknew, taskstatechange, taskserveritychange, taskstatuschange. The log for every task event contains the following fields:

Table 34. Event log fields

Field	Description
timestamp	The time when the event was received and logged.
taskID	The global ID of the task that the event concerns.
originatedTime:	The time the task was originated.
cell	The cell that the task event was received in.
notification	The type of JMX event / notification that was received.
submitter	The originator of the event.
reasonMsg	The task's reason message.
taskType	The type of task.
severity	The global severity of the task.
actions	Any actions associated with the task (if applicable).
state	The task's lifecycle state.
status:	The status of a terminated task.
statusMsg	The status message related to the status of a terminated task.

Enabling and configuring the event logger

The event logger must be enabled and can be configured through cell level custom properties. To access the event logger, go to **System Administration > Cell > Custom properties**. The following table describes the name and default values for the event logger.

Table 35. Names and default values for the event logger

Property Name	Default value	Description
tmslog.delim		The delimiter to use between fields in the log file.
tmslog.enable	false	Whether or not to enable logging of task management service events.
tmslog.fileLocation	\${LOG_ROOT}	The location of where the log file will be written out to.
tmslog.maxFiles	1	The maximum number of historic files to keep.
tmslog.maxSize	20	The maximum log file size to allow before rolling log files (in MB)

Table 35. Names and default values for the event logger (continued)

Property Name	Default value	Description
tmslog.timeFormat	empty string: ""	The format to use when logging timestamp and originatedTime. Reference the javadoc for SimpleDateFormat for information on how to construct a timestamp pattern. If you want the time to span fields, such as to separate date and time into different columns, then use the delimiter in the pattern. If you do not specify a value for the custom property, the epoch time format is used.

The custom properties are not recognized during runtime. Restart the deployment manager to pick up any changes to the custom properties.

Defining email notification

In addition to actively monitoring tasks on the **Runtime tasks** panel, you can have email notifications sent to specified users when runtime tasks occur.

Before you begin

When security is enabled, some fields are not available without proper authorization. You require *configurator* administrative permission to define email notification.

About this task

By defining email notification, you can specify a set of email addresses to be notified when runtime tasks are generated.

The sender user ID is preset to wasxd. Register the sender user ID in your Simple Mail Transfer Protocol (SMTP) registry before the cell can successfully send email notifications.

Procedure

- Optional: Set the user ID from which the emails are sent. By default, the value is preset to wasxd. To change this default, perform the following steps:
 - In the administrative console, click **System administration > Cell > Custom properties > New**.
 - Enter the name of the custom property as `task.email.global.sender.id`, and set the value to the specific email user ID that you want to use.
 - Click **Apply**. The change becomes effective when you save the configuration.
- Configure email notification in the administrative console. Select **System administration > Task management > Notifications**. Select **Enable notifications**. When runtime tasks are generated, an email notification is sent to each of the specified email addresses.
- Verify that email is sent for a particular task. When all of the changes are made, click **Test email** to verify that the email is sent for a particular task. If the email server uses spam-blocking software, this test can fail and prevent the email from being displayed in the test inbox. To save your changes, click **Apply** or **OK**.

Results

When notification is enabled, an email is sent to each of the specified addresses when a task is generated. When notification is enabled and no email addresses are specified, no emails are sent.

What to do next

- You can configure custom properties to get more detailed task information in the emails, or to customize the subject line in the emails.
- Manage the runtime tasks in the administrative console.

Troubleshooting extended administration

Occasionally, when you are using the extended manageability features, you might encounter behavior that is not expected. For example, you can correct Web browser configurations so that your Web browser can support the visualization function.

Linux

Visualization is not working using Mozilla software on Linux systems

For visualization to function correctly, some Linux libraries are required to view the runtime chart. Without these libraries, your environment might not have the visual enhancements that are available in the administrative console.

If you are not able to view the runtime charts, verify that you have the following shared library `/usr/lib/libstdc++-libc6.2-2.so.3`. This library is not installed by default for Red Hat Enterprise Linux 3.0. The following package contains `compat-libstdc++-7.3-2.96.122.i386.rpm`.

Charting not reflecting expectations

Check whether the service policy goals are being met, and that the relative importance of different service policies is respected. Most likely, the system is working as designed. However, to look at or change a service policy, click **Operational policies** > **Service policies** > *service_policy_name* in the administrative console.

Runtime operations reports are empty

To see data in the runtime operations reports, you must run an active load against your cell.

Configuring Intelligent Management to work with other IBM products

Configure Intelligent Management to work with other IBM products.

Procedure

Configure the on demand router (ODR) to make Application Response Measurement (ARM) calls to report response times to IBM Enterprise Workload Manager™. IBM Enterprise Workload Manager can use the response time information to monitor the environment both inside and outside of the Intelligent Management domain. For more information, read about enabling the on demand router to work with IBM Enterprise Workload Manager.

Enabling the on demand router to work with IBM Enterprise Workload Manager

Use this task to enable IBM Enterprise Workload Manager to monitor Intelligent Management.

Before you begin

To perform this task, you must install IBM Enterprise Workload Manager. See the IBM Enterprise Workload Manager information center for installation instructions.

About this task

By performing this task, the on demand router (ODR) makes Application Response Measurement (ARM) calls to report response times to IBM Enterprise Workload Manager. IBM Enterprise Workload Manager can use the response time information to monitor the environment both inside and outside of the Intelligent Management domain. For more information, read about Intelligent Management and IBM Enterprise Workload Manager.

Procedure

1. Configure service and transaction classes in IBM Enterprise Workload Manager. The service and transaction classes that you configure in IBM Enterprise Workload Manager override any service classes that are configured in Intelligent Management. See the Enterprise Workload Manager information center for more information.
2. Configure IBM Enterprise Workload Manager to manage Intelligent Management. After you complete this step, any service policies that are configured in Intelligent Management are not valid. IBM Enterprise Workload Manager has its own configured service and transaction classes.
3. Copy the ARM libraries from IBM Enterprise Workload Manager to the Intelligent Management configuration. Include the ARM libraries from IBM Enterprise Workload Manager in the *install_root/bin* directory. Remember the name of this class so that you can supply the information in the administrative console. Alternatively, you can add two custom properties to the application server or ODR Java virtual machine (JVM). First, define the `java.library.path` custom property and set its value to the IBM Enterprise Workload Manager library path, for example `c:\Program Files\IBM\VE2\EWLMMS\classes\ms`. Next, create the `ws.ext.dirs` custom property and set its value to the folder that contains the ARM libraries that are provided by IBM Enterprise Workload Manager, for example, `c:\Program Files\IBM\VE2\EWLMMS\classes\ARM`.
4. Enable ARM reporting. On the request metrics administrative console page, use the following settings:
 - Select the appropriate application component, such as servlets or Web services, that needs to report ARM statistics.
 - Defining the trace level is optional.
 - Enter the name of the IBM Enterprise Workload Manager ARM factory in the ARM transaction factory implementation class name. The transaction class factory name for IBM Enterprise Workload Manager is `com.ibm.wlm.arm40SDK.transaction.Arm40TransactionFactory`.

For more information, read about getting performance data from request metrics.

5. Enable ARM reporting on your other tiers and applications, such as IBM DB2, IBM HTTP Server, and also for the various components running on WebSphere Application Server. You can enable ARM instrumentation in specific WebSphere Application Server components by selecting the appropriate components. For more information, read about getting performance data from request metrics. See the product documentation for your specific tier applications regarding more information on enabling ARM reporting. Any tiers that send requests to the Intelligent Management tier must be ARM enabled to facilitate the interaction between IBM Enterprise Workload Manager and Intelligent Management. Any tiers that receive requests from the Intelligent Management tier do not need to have ARM enabled, however, enabling ARM on these tiers is strongly encouraged.
6. To enable WebSphere Application Server and Intelligent Management with IBM Enterprise Workload Manager, you must manually update the WebSphere Application Server `server.policy` files. You must update the `server.policy` file for all application servers and on demand routers. For more information, read about the `server.policy` file permission.
7. Restart all applications and servers that IBM Enterprise Workload Manager is now monitoring.

Results

The on demand routers in the Intelligent Management infrastructure will begin to make ARM calls to report response times on requests. The IBM Enterprise Workload Manager can monitor the Intelligent Management environment.

Intelligent Management and IBM Enterprise Workload Manager

You can enable Intelligent Management to work with other autonomic managers, such as IBM Enterprise Workload Manager.

If you enable the on demand router (ODR) to report request metrics, Application Response Measurement (ARM) -enabled autonomic managers like Enterprise Workload Manager can influence how requests are processed in Intelligent Management. Enterprise Workload Manager has knowledge and control outside of the Intelligent Management domain. Enterprise Workload Manager can use the request metrics to determine if service class policies and transaction class policies that are defined in the Enterprise Workload Manager configuration are met. If you need to make changes to meet the requirements, Enterprise Workload Manager uses its knowledge and control outside of Intelligent Management to assist Intelligent Management in meeting the broader response time goals for Enterprise Workload Manager.

Support of ARM instrumentation in the ODR enables the reporting of response time metrics on a per transaction basis. ARM is an industry standard for accurate measurements of request and response transactions as they move through various tiers of applications. For more information about ARM and request metrics, see the WebSphere Application Server Network Deployment Information Center.

Any tiers that send requests to Intelligent Management must be ARM-enabled to facilitate the interaction between Enterprise Workload Manager and Intelligent Management. Any tiers that receive requests from Intelligent Management do not need to have ARM enabled, however, enabling ARM on these tiers is encouraged. For example, enable ARM on all application servers in software that works in cooperation with Intelligent Management, such as IBM DB2 and so on. Enterprise Workload Manager can make resource level measurements for tiers that do not have ARM enabled. You can also enable ARM reporting on any Web servers that forward requests to the ODR.

For more information about the IBM Enterprise Workload Manager, see the IBM Enterprise Workload Manager information center.

Reference

Reference information is organized to help you locate particular facts quickly.

Command-line utilities

Look up a command by its name to find detailed syntax and usage of the command.

To open the information center table of contents to the location of this reference information, click the **Refresh/Show Current Topic** button (👉) on your information center border.

Commands (wsadmin scripting)

Look up a scripting object or command class to find details about its command syntax.

To open the information center table of contents to the location of this reference information, click the **Show in Table of Contents** button (👉) on your information center border.

Settings

This reference information describes settings that you can view and configure on the pages of the product administrative console and elsewhere. Custom properties are documented separately. They are name-value pairs that you can enter on specific console pages if you know what to specify.

To open the Table of Contents to the location of this reference information, click the **Refresh/Show Current Topic** button, 👉, located on the information center border. For example, if you found this page from the information center search or from an internet search engine, such as Google, click the button to

make the information center show you the location of this page in the information center navigation tree. You will be able to browse the contents indented under this navigation entry.

Custom properties

This reference information lists the custom properties that are available with the product. Custom properties are unique settings that you specify as name-value pairs on specific administrative console pages.

For example, the Session Initiation Protocol (SIP) container custom property, `immediate.replication`, which is used to control whether each change is immediately sent to the Data Replication Service, is set on the SIP container administrative console page.

Additional custom properties might be documented on the Support site as resolutions to specific customer problem reports.

When you click on the name of one of the custom properties in the following list, you are automatically taken to the topic that fully describes that custom property. That topic also describes how to navigate to the administrative console page where you can specify that custom property.

Administration services custom properties

- `com.ibm.websphere.mbeans.disableRouting`

Cell custom properties

- `com.ibm.websphere.management.launcher.options`
- `com.ibm.websphere.process.terminator.deletepid`
- `enableadminauthorizationcache`
- `IBM_CLUSTER_CALLBACK_TIMEOUT`
- `IBM_CLUSTER_CUSTOM_ADVISOR_THREAD_POOL_SIZE`
- `IBM_CLUSTER_ENABLE_ACS_DELAY_POSTING`
- `IBM_CLUSTER_ENABLE_CAR_DELAY_POSTING`
- `IBM_CLUSTER_ENABLE_PRELOAD`
- `IBM_CLUSTER_ENABLE_NON_DEFAULT_COOKIE_NAMES`
- `IBM_CLUSTER_ENABLE_SERVLET30_NON_DEFAULT_COOKIE_NAMES`
- `IBM_CLUSTER_PURGE_NOTIFICATIONS`
- `IBM_CLUSTER_RIPPLESTART_NOTIFICATION_TIMEOUT`
- `IBM_CLUSTER_USE_LEGACY_COMPRESSOR`
- `IBM_CLUSTER_WBI_SUPPORT`

Compensation service custom properties

- Suppressing the compensation service

Core group bridge custom properties

- `CGB_ENABLE_602_FEATURES`
- `cgb.allowUndefinedBridges`
- `cgb.rebuild.waitTime`
- `FW_PASSIVE_MEMBER`
- `IBM_CS_LS_DATASTACK_MEG`

Core group custom properties

- `IBM_CS_DATASTACK_MEG`
- `IBM_CS_FD_CONSECUTIVE_MISSED`
- `IBM_CS_FD_PERIOD_SECS`

- IBM_CS_HAM_PROTOCOL_VERSION
- IBM_CS_IP_REFRESH_MINUTES
- IBM_CS_SOCKET_BUFFER_SIZE
- IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS
- IBM_CS_WIRE_FORMAT_VERSION

EJB container system properties

- com.ibm.websphere.ejbcontainer.allowEarlyInsert
- com.ibm.websphere.ejbcontainer.checkEJBApplicationConfiguration
- com.ibm.websphere.ejbcontainer.declaredUncheckedAreSystemExceptions
- com.ibm.websphere.ejbcontainer.defaultSessionAccessTimeout
- com.ibm.websphere.ejbcontainer.defaultStatefulSessionTimeout
- com.ibm.websphere.ejbcontainer.EE5Compatibility
- com.ibm.websphere.ejbcontainer.limitSetRollbackOnlyBehaviorToInstanceFor
- com.ibm.websphere.ejbcontainer.poolSize

HTTP proxy server custom properties

- cache.ignore.header.Authorization
- cache.ignore.header.Cookie
- cache.ignore.header.Proxy-Authorization
- cache.query.string
- com.ibm.ws.webcontainer.returnDefaultContextPath
- http.auto.redirect.correction
- http.cache.nocache.headers
- http.clientInfoFromTrustedIntermediary
- http.connectionPoolUseForPOST
- http.disable.retry.on.503.uriprefix
- http.disable.response.buffering.urls
- http.isDisable10ResponseCaching
- http.log.history
- http.maxCachedPayload
- http.odcUpdateTimeout
- http.pmiTimerInterval
- http.routing.sendReverseProxyNameInHost
- http.virtual.port.map
- HTTPProxyAdvisorMethodName
- HTTPProxyAdvisorStartupDelay
- HTTPProxyAdvisorURI
- HTTPProxyAdvisorUserAgent
- LBIPAddr
- localOutboundTCPAddress

HTTP transport channel custom properties

- com.ibm.ws.webcontainer.returnDefaultContextPath
- ConnectionResponseTimeout
- CookiesConfigureNoCache
- EnableBuildBackupList

- HonorTransferEncoding
- limitFieldSize
- limitNumHeaders
- localLogFilenamePrefix
- RemoveServerHeader
- ServerHeaderValue

HTTP transport custom properties

- ConnectionIOTimeout
- ConnectionKeepAliveTimeout
- **z/OS** ConnectionResponseTimeout
- **Distributed operating systems** KeepAliveEnabled
- **Distributed operating systems** MaxConnectBacklog
- **Distributed operating systems** MaxKeepAliveConnections
- MaxKeepAliveRequests
- **z/OS** MutualAuthCBindCheck
- RemoveServerHeader
- ResponseBufferSize
- **z/OS** ServerHeader
- ServerHeaderValue
- SoLingerValue
- TcpNoDelay
- Trusted
- UseSoLinger

HTTP transport custom properties for Web services applications

- com.ibm.websphere.webservices.http.connectionIdleTimeout
- com.ibm.websphere.webservices.http.connectionKeepAlive
- com.ibm.websphere.webservices.http.connectionPoolCleanUpTime
- com.ibm.websphere.webservices.http.connectionTimeout
- com.ibm.websphere.webservices.http.maxConnection
- com.ibm.websphere.webservices.http.responseContentEncoding
- com.ibm.websphere.webservices.http.requestContentEncoding
- com.ibm.websphere.webservices.http.requestResendEnabled
- com.ibm.websphere.webservices.http.SocketTimeout
- com.ibm.ws.webservices.enableHTTPPrefix
- enableMultiThreadedSession
- HttpInboundPersistReadTimeout
- http.nonProxyHosts
- http.proxyHost
- http.proxyPort
- https.proxyHost
- https.proxyPort
- timeout
- write_timeout

Intelligent Management custom properties

- Application edition manager custom properties
- Administering applications and their environment
- Autonomic controllers custom properties
- Autonomic request flow manager advanced custom properties
- Autonomic request flow manager custom properties
- Binary trace facility custom properties
- Controlling the generation of the plugin-cfg.xml file
- Custom properties for Intelligent Management
- Dynamic cluster custom properties
- Health controller custom properties
- Middleware server custom properties and variables
- Node agent custom properties for Intelligent Management
- On demand router system and custom properties
- Operational alerts custom properties
- Runtime operations custom properties
- Runtime task custom properties
- Service policy custom properties
- VMware custom properties

Java Management Extensions connector properties

SOAP connector properties

- Configuration URL
- Secure Sockets Layer (SSL) security
- Security context provider
- SOAP request timeout
- SSL alias

IPC connector properties

- Configuration URL
- IPC request timeout
- Secure Sockets Layer (SSL) security
- Security context provider
- SSL alias

SOAP, RMI, JSR160RMI, and IPC connector properties

- Connector type
- Disabling a connector
- Host
- Password
- Port
- User name

RMI connector properties

- Disabling the JSR 160 RMI connector

JavaServer Pages specific web container custom properties

- com.ibm.ws.jsp.getParameterreturnemptystring
- com.ibm.ws.jsp.jdksourcelevel
- com.ibm.ws.jstl.allowLenientDateParsing

- com.ibm.wsspi.jsp.allowjspoutpulelementmismatch
- com.ibm.wsspi.jsp.allowtaglibprefixusebeforedefinition
- com.ibm.wsspi.jsp.allowtaglibprefixredefinition
- com.ibm.wsspi.jsp.allowunmatchedendtag
- com.ibm.wsspi.jsp.evalquotedandescapedexpression
- com.ibm.wsspi.jsp.modifyPageContextVariable
- com.ibm.wsspi.jsp.recompilejuponrestart
- com.ibm.wsspi.jsp.usecdatatrims
- com.ibm.wsspi.jsp.usescriptvardupinit
- com.ibm.wsspi.jsp.usestringcast
- com.ibm.wsspi.jsp.reusepropertygroupconfigoninclude

Java virtual machine cache settings

- com.ibm.ws.cache.CacheConfig.explicitBufferLimitOnStop
- com.ibm.ws.cache.CacheConfig.htodCleanupFrequency
- com.ibm.ws.cache.CacheConfig.htodDelayOffloadEntriesLimit
- com.ibm.ws.cache.CacheConfig.lruToDiskTriggerPercent
- com.ibm.ws.cache.CacheConfig.lruToDiskTriggerTime

Java virtual machine custom properties

- allowDeployerRoleGenPluginCfg
- com.ibm.cacheLocalHost
- com.ibm.config.eclipse.wtp.enablejemtrim
- com.ibm.config.eclipse.wtp.enablexmltrim
- com.ibm.config.eclipse.wtp.jem=finer
- com.ibm.config.eclipse.wtp.xmltrim=finer
- AIX HP-UX Linux Solaris z/OS
com.ibm.eclipse.wtp.allowRootedEntries
- com.ibm.ejs.ras.writeSystemStreamsDirectlyToFile
- com.ibm.ejs.sm.server.quiesceInactiveRequestTime
- com.ibm.ejs.sm.server.quiesceTimeout
- com.ibm.websphere.application.updateapponcluster.waitforappsav
- z/OS com.ibm.websphere.bean.delete.sleep.time
- com.ibm.websphere.deletejspclasses
- com.ibm.websphere.deletejspclasses.delete
- com.ibm.websphere.deletejspclasses.update
- com.ibm.websphere.ejb.UseEJB61FEPScanPolicy
- com.ibm.websphere.ejbcontainer.expandCMPCFJNDIName
- com.ibm.websphere.ejbcontainer.includeRootExceptionOnRollback
- com.ibm.websphere.jaxrpc.stub.typemapping.per.thread
- com.ibm.websphere.jaxrs.server.DisableIBMJAXRSEngine
- com.ibm.websphere.management.application.client.EnvEntry_processBindings
- com.ibm.websphere.management.application.fullupdate
- com.ibm.websphere.management.application.fullupdate.application_name
- com.ibm.websphere.management.application.keepExistingSharedLibraries
- com.ibm.websphere.management.application.persistWebContext
- com.ibm.websphere.management.application.sync.recycleappasv5

- com.ibm.websphere.management.application.sync.recycleappasv5
- com.ibm.websphere.management.application.updateSync.appExpansionTimeout
- com.ibm.websphere.management.configservice.getServerLogRootFromTemplate
- com.ibm.websphere.management.configservice.sessionIdUniqueness
- com.ibm.websphere.management.configservice.validatePropNames
- **z/OS** com.ibm.websphere.management.jmx.random
- com.ibm.websphere.management.nodesync.skipWebServerTarget
- com.ibm.websphere.management.processEmbeddedConfigGlobal
- **z/OS** com.ibm.websphere.management.registerServerIORWithLSD
- com.ibm.websphere.metadata.ignoreDuplicateRefBindingsInWebModule
- com.ibm.websphere.network.useMultiHome
- com.ibm.websphere.sib.webservices.useTypeSoapArray
- com.ibm.websphere.webservices.attachment.tempfile.expiration
- com.ibm.websphere.webservices.attachments.maxMemCacheSize
- com.ibm.websphere.webservices.DisableIBMJAXWSEngine
- com.ibm.websphere.webservices.http.OneWayConnectionRecycleTime
- com.ibm.websphere.webservices.http.waitingThreadsThreshold
- com.ibm.websphere.webservices.jaxrpc.client.publishwsdl
- com.ibm.websphere.webservices.soap.enable.legacy.get.behavior
- com.ibm.websphere.webservices.tempAttachDir
- com.ibm.websphere.webservices.transport.jms.messageType
- com.ibm.websphere.webservices.transport.OPTIMIZE_HTTP_HEADERS
- com.ibm.websphere.webservices.transport.ssl.loadFromPolicyBinding
- com.ibm.websphere.webservices.UseWSFEP61ScanPolicy
- com.ibm.websphere.webservices.WSDL_Generation_Extra_Classpath
- com.ibm.ws.amm.scan.context.filter.archives
- com.ibm.ws.amm.scan.context.filter.packages.
- com.ibm.ws.application.enhancedScanning
- com.ibm.ws.cache.CacheConfig.alwaysSetSurrogateControlHdr
- com.ibm.ws.cache.CacheConfig.cascadeCachespecProperties
- com.ibm.ws.cache.CacheConfig.filteredStatusCodes
- com.ibm.ws.CacheConfig.alwaysTriggerCommandInvalidations
- com.ibm.ws.classloader.allowDisposedClassLoad
- com.ibm.ws.classloader.encodeResourceURLs
- com.ibm.ws.classloader.strict
- com.ibm.ws.classloader.zipFileCacheSize
- com.ibm.ws.el.reuseEvaluationContext
- **z/OS** com.ibm.ws.iiop.channel.disableOnewayLocateRequiredMessage
- com.ibm.ws.management.connector.soap.logClientInfo
- **Windows** com.ibm.ws.management.connector.soap.waitBeforeCloseTime
- com.ibm.ws.management.event.max_polling_interval
- com.ibm.ws.management.event.pull_notification_timeout
- com.ibm.ws.management.repository.tempFileKeepTimeMinutes
- com.ibm.ws.management.repository.tempFileSweepIntervalMinutes
- com.ibm.ws.odr.plugincfg.config.ASDisableNagle

- com.ibm.ws.odr.plugincfg.config.AcceptAllContent
- com.ibm.ws.odr.plugincfg.config.AppServerPortPreference
- com.ibm.ws.odr.plugincfg.config.ChunkedResponse
- com.ibm.ws.odr.plugincfg.config.IISDisableNagle
- com.ibm.ws.odr.plugincfg.config.IISPluginPriority
- com.ibm.ws.odr.plugincfg.config.IgnoreDNSFailures
- com.ibm.ws.odr.plugincfg.config.RefreshInterval
- com.ibm.ws.odr.plugincfg.config.ResponseChunkSize
- com.ibm.ws.odr.plugincfg.config.VHostMatchingCompat
- com.ibm.ws.odr.plugincfg.config.TrustedProxyEnable
- com.ibm.ws.odr.plugincfg.log.Name
- com.ibm.ws.odr.plugincfg.log.LogLevel
- com.ibm.ws.odr.plugincfg.cluster.CloneSeparatorChange
- com.ibm.ws.odr.plugincfg.cluster.LoadBalance
- com.ibm.ws.odr.plugincfg.cluster.PostSizeLimit
- com.ibm.ws.odr.plugincfg.cluster.RemoveSpecialHeaders
- com.ibm.ws.odr.plugincfg.cluster.RetryInterval
- com.ibm.ws.odr.plugincfg.odrIncludeStopped
- com.ibm.ws.odr.plugincfg.server.ConnectTimeout
- com.ibm.ws.odr.plugincfg.server.ExtendedHandShake
- com.ibm.ws.odr.plugincfg.server.MaxConnections
- com.ibm.ws.odr.plugincfg.cluster.WaitForContinue
- com.ibm.ws.odr.plugincfg.property.ESIEnable
- com.ibm.ws.odr.plugincfg.property.ESIMaxCacheSize
- com.ibm.ws.odr.plugincfg.property.ESIInvalidationMonitor
- com.ibm.ws.odr.plugincfg.property.https.keyring
- com.ibm.ws.odr.plugincfg.property.https.stashfile
- com.ibm.ws.odr.plugincfg.property.PluginInstallRoot
- com.ibm.ws.pm.checkingDBconnection
- com.ibm.ws.runtime.component.ResourceMgr.postBindNotify
- com.ibm.ws.runtime.dumpShutdown
- **Distributed operating systems** com.ibm.ws.runtime.logThreadPoolGrowth
- com.ibm.ws.scripting.apptimeout
- com.ibm.ws.sib.webservices.useSOAPJMSTextMessages
- com.ibm.ws.use602RequiredAttrCompatibility
- com.ibm.ws.webservices.allowNoSOAPActionHeader
- com.ibm.ws.webservices.allowStatusCode202OneWay
- com.ibm.ws.webservices.appendRootCauseToWSF
- com.ibm.ws.webservices.contentTransferEncoding
- com.ibm.ws.webservices.disableSOAPElementLazyParse
- com.ibm.ws.webservices.engine.transport.jms.propagateOneWaySystemExceptions
- com.ibm.ws.webservices.forceLegacyDispatchFromSOAPConnection
- com.ibm.ws.webservices.HttpRedirectWithProxy
- com.ibm.ws.webservices.ignoreUnknownElements
- com.ibm.ws.webservices.jaxrpc.parse.tolerate.invalid.namespace

- com.ibm.ws.webservices.resolveXMLSchemaDTD
- com.ibm.ws.webservices.searchForAppServer
- com.ibm.ws.webservices.serialize.2DimArray.asArrays
- com.ibm.ws.webservices.serializeDetailElementUsingDefaultNamespace
- com.ibm.ws.webservices.suppressHTTPRequestPortSuffix
- com.ibm.ws.websvcs.attachments.sizethreshold
- com.ibm.ws.websvcs.getJAXBContext.cacheClassList
- com.ibm.ws.websvcs.relaxClientPolsetMatching
- com.ibm.ws.websvcs.suppressHTTPRequestPortSuffix
- com.ibm.ws.websvcs.transport.enableProxyTunnel
- com.ibm.ws.websvcs.transport.jms.enableBasicAuthOnResponse
- com.ibm.ws.websvcs.unmanaged.client.dontUseOverriddenEndpointUri
- com.ibm.ws.ws.wsba.protocolmessages.twoway
- **z/OS** com.ibm.ws390.SystemOutErrCodepage
- com.ibm.wsspi.amm.merge.ignoreValidationExceptions
- com.ibm.wsspi.wssecurity.dsig.enableEnvelopedSignatureProperty
- com.ibm.xml.xlsp.jaxb.opti.level
- config_consistency_check
- deactivateWildCardURIMapping
- DISABLE_LOCAL_COMM_WHEN_SSL_REQUIRED
- disableWSAddressCaching
- DRS_BATCH_INTERVAL_SIZE
- DRS_THREADPOOL_ISGROWABLE
- DRS_THREADPOOL_MINSIZE
- DRS_THREADPOOL_MAXSIZE
- dynacache.jms.cacheInstance
- dynacache.jms.connRetryInterval
- dynacache.jms.invProcessingDelay
- dynacache.jms.numStoredInvalidations
- **z/OS** invocationCacheSize
- **Distributed operating systems** java.net.preferIPv4Stack
- **Distributed operating systems** java.net.preferIPv6Addresses
- java.util.logging.configureByLoggingPropertiesFile
- jaxws.asyncClient.maxThreadPoolSize
- jaxws.payload.highFidelity
- jaxws.provider.interpretNullAsOneway
- #jaxws.runtime.inheritedimplmethodsaccessible
- jaxws.runtime.restrictStaticWebmethod
- jaxws.soapfault.local.exceptions.disable
- ODCClearMessageAge
- ODCInit.disabled
- #org.apache.axiom.attachments.tempfile.expiration
- org.eclipse.jst.j2ee.commonarchivecore.disableZip
- org.eclipse.jst.j2ee.commonarchivecore.FILTERBINARIES

- plugin.syncdisabled
- sizeThreshold
- threadpool.maxsize
- **z/OS** was.xcfmonitor.enabled
- webservices.unify.faults
- **z/OS** wink.client.readTimeout
- **z/OS** wink.client.connectTimeout

Message listener service custom properties

- DYNAMIC.CONFIGURATION.ENABLED
- **z/OS** ENABLE.ZOS.LP.RECOVERY
- MAX.RECOVERY.RETRIES
- MQJMS.POOLING.THRESHOLD
- MQJMS.POOLING.TIMEOUT
- NON.ASF.RECEIVE.TIMEOUT
- NON.ASF.BMT.ROLLBACK.ENABLED
- RECOVERY.RETRY.INTERVAL
- SERVER.SESSION.POOL.REAP.TIME
- SERVER.SESSION.POOL.UNUSED.TIMEOUT
- SERVER.SESSION.POOL.UNUSED.TIMEOUT.lpname

Object Request Broker custom properties

- com.ibm.CORBA.BootstrapHost
- com.ibm.CORBA.BootstrapPort
- com.ibm.CORBA.ConnectTimeout
- com.ibm.CORBA.ConnectionInterceptorName
- com.ibm.CORBA.enableLocateRequest
- **Distributed operating systems** com.ibm.CORBA.FragmentSize
- com.ibm.CORBA.ListenerPort
- com.ibm.CORBA.LocalHost
- **Distributed operating systems** com.ibm.CORBA.numJNIReaders
- **Distributed operating systems** com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.JNIReaderPoolImpl
- com.ibm.CORBA.RasManager
- **Distributed operating systems** com.ibm.CORBA.SendConnectionContexts
- com.ibm.CORBA.ServerSocketQueueDepth
- com.ibm.CORBA.ShortExceptionDetails
- com.ibm.CORBA.WSSSLClientSocketFactoryName
- com.ibm.CORBA.WSSSLServerSocketFactoryName
- com.ibm.websphere.ObjectIDVersionCompatibility
- com.ibm.websphere.orb.threadPoolTimeout
- **Distributed operating systems** com.ibm.websphere.threadpool.strategy.implementation
- **Distributed operating systems** com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.calcinterval
- **Distributed operating systems** com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.lruinterval
- **Distributed operating systems** com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.outqueues

- **Distributed operating systems** com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.statsinterval
- **Distributed operating systems** com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.workqueue
- com.ibm.ws.orb.services.lsd.EnableSecurityServiceCheck
- com.ibm.ws.orb.services.lsd.SecurityServiceCheckInterval
- com.ibm.ws.orb.services.lsd.SecurityServiceCheckRetry
- com.ibm.ws.orb.services.redirector.MaxOpenSocketsPerEndpoint
- com.ibm.ws.orb.services.redirector.RequestTimeout
- com.ibm.ws.orb.transport.SSLHandshakeTimeout
- com.ibm.ws.orb.transport.useMultiHome
- javax.rmi.CORBA.UtilClass

Portlet container custom properties

- useShortMBeanNames

Distributed operating systems

z/OS

Repository service custom properties

- recoveryNode

Service integration custom properties

- sib.msgstore.cachedDataBufferSize
- sib.msgstore.discardableDataBufferSize
- sib.msgstore.jdbcFailoverOnDBConnectionLoss
- sib.msgstore.jdbcInitialDatasourceWaitTimeout
- sib.msgstore.jdbcResAuthForConnections
- sib.msgstore.jdbcStaleConnectionRetryDelay
- sib.meEnableInstanceOnFailure
-
- sib.msgstore.storeFullWaitForCheckPoint
- sib.msgstore.transactionSendLimit
- **z/OS** sib.ra.zosMessageLockTimeout
- sib.trm.retry
- sib.wsrn.tokenLockTimeout

Session management custom properties

- AlwaysEncodeURL
- CloneSeparator
- CloneSeparatorChange
- DebugSessionCrossover
- DelayAfterDuplicateIdException
- ForceSessionInvalidationMultiple
- HideSessionValues
- HttpSessionCloneId
- **z/OS** HttpSessionEnableUnmanagedServerReplication
- HttpSessionIdLength
- HttpSessionIdReuse
- HttpSessionReaperPollInterval
- InvalidateOnUnauthorizedSessionRequestException
- NoAdditionalSessionInfo

- NoAffinitySwitchBack
- OptimizeCacheIdIncrements
- SecurityUserIgnoreCase
- Servlet21SessionCompatibility
- SessionIdentifierMaxLength
- SessionRewriteIdentifier
- SessionTableName
- SessionTableSkipIndexCreation
- UseInvalidatedId
- UseOracleBLOB
- UsingApplicationSessionsAndInvalidateAll
- UsingCustomSchemaName

SIP container custom properties

- auth.int.enable
- com.ibm.sip.sm.lnm.size
- com.ibm.websphere.sip.security.digest.ldap.cachecleanperiod
- com.ibm.websphere.sip.security.tai.usercachecleanperiod
- com.ibm.ws.sip.key.set
- com.ibm.ws.sip.tai.DisableSIPBasicAuth
- DigestPasswordServerClass
- enable.system.headers.modify
- end.of.service.replication
- immediate.replication
- javax.servlet.sip.ar.dar.configuration
- javax.servlet.sip.ar.spi.SipApplicationRouterProvider
- javax.sip.bind.retries
- javax.sip.bind.retry.delay
- javax.sip.detect.pre.escaped.params
- javax.sip.force.connection.reuse
- javax.sip.hide.message.body
- javax.sip.hide.message.headers
- javax.sip.hide.request.uri
- javax.sip.OUTBOUND_PROXY
- javax.sip.PATH_MTU
- javax.sip.stat.report.interval
- javax.sip.trace.msg.in
- javax.sip.trace.msg.out
- javax.sip.transaction.invite.auto100
- javax.sip.transaction.timer.a
- javax.sip.transaction.timer.b
- javax.sip.transaction.timer.cancel
- javax.sip.transaction.timer.d
- javax.sip.transaction.timer.e
- javax.sip.transaction.timer.f
- javax.sip.transaction.timer.g

- javax.sip.transaction.timer.h
- javax.sip.transaction.timer.i
- javax.sip.transaction.timer.invite.server
- javax.sip.transaction.timer.j
- javax.sip.transaction.timer.k
- javax.sip.transaction.timer.non.invite.server
- javax.sip.transaction.timer.t1
- javax.sip.transaction.timer.t2
- javax.sip.transaction.timer.t4
- on.outgoing.message.replication
- pws_atr_name
- replicate.with.confirmed.dialog.only
- sip.container.heartbeat.enabled
- sip.jsr289.parse.address
- SIP_RFC3263_nameserver
- thread.message.queue.max.size
- weight.overload.watermark

SIP Proxy custom properties

- Overload custom properties
- burstResetFactor
- deflatorRatio
- dropOverloadPackets
- inDialogAveragingPeriod
- maxThroughputFactor
- numFailuresSipAdvisorRequests
- outDialogAveragingPeriod
- overloadResponseCode
- overloadResponseReasonPhrase
- perSecondBurstFactor
- proxyTransitionPeriod
- sipProxyStartupDelay

SIP general custom properties

The following SIP general custom properties allow you to apply a variety of settings.

- defaultTCPChainName
- defaultTLSChainName
- defaultUDPChainName
- disable.failover.suicide
- http.connectRetryLimit
- identityAssertionHeaderRemovalEnabled
- ignore.ucf.messages.from.proxy
- **Fix Pack 1** ipForwardingLBEnabled
- IsnLookupFailureResponseCode
- IsnLookupFailureReasonPhrase
- isSipComplianceEnabled

- keepAliveFailures
- keepAliveInterval
- LBIPAddr
- localOutboundTCPAddress
- maddrParameterEnabled
- maxBackupLogFiles
- maxForwardsHeaderRequired
- maxViaHeaderPortNumber
- maxWriteQueueEntries
- numFailuresSipAdvisorRequests
- receiveBufferSizeChannel
- receiveBufferSizeSocket
- retryAfterValue
- sendBufferSizeSocket
- serverUDPInterface
- serverUDPPort
- SIPAdvisorMethodName
- sipAdvisorRequestTimeout
- sipClusterCellName
- tcp.IPSprayer.host
- tcp.IPSprayer.port
- tls.IPSprayer.host
- tls.IPSprayer.port
- trustedIPAddressList
- udp.IPSprayer.host
- udp.IPSprayer.port
- udpLSNFailoverTimeout

Distributed operating systems SIP UDP transport channel custom properties

- autoCreateConnLink
- hideMessageBody
- hungThreadTimeout
- hideMessageHeaders
- receiveBufferSizeChannel
- receiveBufferSizeSocket
- sendBufferSizeSocket

SPNEGO TAI custom properties configuration (deprecated)

- com.ibm.ws.security.spnego.SPN<id>.enableCredDelegate
- com.ibm.ws.security.spnego.SPN<id>.filter
- com.ibm.ws.security.spnego.SPN<id>.filterClass
- com.ibm.ws.security.spnego.SPN<id>.hostName
- com.ibm.ws.security.spnego.SPN<id>.NTLMTokenReceivedPage
- com.ibm.ws.security.spnego.SPN<id>.spnegoNotSupportedPage
- com.ibm.ws.security.spnego.SPN<id>.trimUserName

SPNEGO TAI JVM configuration custom properties (deprecated)

- com.ibm.ws.security.spnego.isEnabled
- com.ibm.ws.security.spnego.propertyReloadFile
- com.ibm.ws.security.spnego.propertyReloadTimeout
- com.ibm.security.jgss.debug
- com.ibm.security.krb5.Krb5Debug
- java.security.properties
- javax.security.auth.useSubjectCredsOnly

TCP transport channel custom properties

- listenBacklog
- **z/OS** zaioFreeInitialBuffers

Transaction service custom properties

- DELAY_CANCELLING_ALARMS
- **z/OS** DISABLE_OUTBOUND_CASCADE_SUPPORT
- DISABLE_RECOVERY_AUDIT_LOGGING
- **z/OS** DISABLE_TRANSACTION_TIMEOUT_GRACE_PERIOD
- **z/OS** DISABLE_WSTX_RMFAIL_LOGGING
- REMOVE_PARTNER_LOG_ENTRY
- **z/OS** RLS_LOGSTREAM_COMPRESS_INTERVAL

Web container custom properties

- BodyContentBufferSize
- com.ibm.ws.jsf.disablealternatefacesconfigsearch
- com.ibm.ws.jsp.enableDefaultIsELIgnoredInTag
- com.ibm.ws.jsp.expressionreturnemptystring
- com.ibm.ws.jsp.getWriterOnEmptyBuffer
- com.ibm.ws.jsp.limitBuffer
- com.ibm.ws.jsp.throwExceptionForAddELResolver
- **z/OS** com.ibm.ws.jsp.zosFileLockRetrying
- **z/OS** com.ibm.ws.jsp.zosReCompile
- com.ibm.ws.webcontainer.assumeFiltersSuccessOnSecurityError
- com.ibm.ws.webcontainer.channelWritetype
- com.ibm.ws.webcontainer.checkEDRinGetRealPath
- com.ibm.ws.webcontainer.copyAttributesKeyset
- com.ibm.ws.webcontainer.disableSetCharacterEncodingAfterParametersRead
- com.ibm.ws.webcontainer.disableSystemAppGlobalListenerLoading
- com.ibm.ws.webcontainer.disablePoweredBy
- com.ibm.ws.webcontainer.disallowAllFileServing
- com.ibm.ws.webcontainer.disallowServeservletsbyclassname
- com.ibm.ws.webcontainer.discernUnavailableServlet
- com.ibm.ws.webcontainer.dispatcherRethrowSER
- com.ibm.ws.webcontainer.dispatcherRethrowSEError
- **z/OS** com.ibm.ws.webcontainer.divertRecursiveExceptionToErrorLog
- com.ibm.ws.webcontainer.doNotServebyclassname
- com.ibm.ws.webcontainer.enableDefaultServletRequestPathElements
- com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst

- com.ibm.ws.webcontainer.enableExactMatchJSecurityCheck
- com.ibm.ws.webcontainer.enableJspMappingOverride
- com.ibm.ws.webcontainer.enableMultiReadOfPostData
- com.ibm.ws.webcontainer.extractHostHeaderPort and trusthostheaderport
- **z/OS** com.ibm.ws.webcontainer.fileWrapperEvents
- **z/OS** com.ibm.ws.webcontainer.FileWrapperEventsLessDetail
- com.ibm.ws.webcontainer.finishresponseonclose
- com.ibm.ws.webcontainer.ForceDifferentCookiePaths
- com.ibm.ws.webcontainer.HTTPOnlyCookies
- com.ibm.ws.webcontainer.ignoreinjectionfailure
- com.ibm.ws.webcontainer.ignoreInvalidQueryString
- com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequest
- com.ibm.ws.webcontainer.invokeFilterInitAtStartup
- com.ibm.ws.webcontainer.invokeFiltersCompatibility
- com.ibm.ws.webcontainer.invokerequestlistenerforfilter
- com.ibm.ws.webcontainer.KeepUnreadPostDataAfterResponseSentToClient
- com.ibm.ws.webcontainer.logServletContainerInitializerClassloadingErrors
- com.ibm.ws.webcontainer.mapFiltersToAsterisk
- com.ibm.ws.webcontainer.maxParamPerRequest
- com.ibm.ws.webcontainer.modifiedFileNotFoundExceptionBehavior
- com.ibm.ws.webcontainer.normalizerequesturi
- com.ibm.ws.webcontainer.parseUTF8PostData
- com.ibm.ws.webcontainer.provideQStringToWelcomeFile
- com.ibm.ws.webcontainer.SendResponseToClientAsPartOfSendRedirect
- com.ibm.ws.webcontainer.SendResponseToClientWhenResponselsComplete
- com.ibm.ws.webcontainer.setcontenttypebysetheader
- com.ibm.ws.webcontainer.ServeWelcomeFileFromExtendedDocumentRoot
- com.ibm.ws.webcontainer.ServletDestroyWaitTime
- com.ibm.ws.webcontainer.setUnencodedHTMLinsenderror
- com.ibm.ws.webcontainer.suppressheadersinrequest
- com.ibm.ws.webcontainer.suppressHtmlRecursiveErrorOutput
- com.ibm.ws.webcontainer.suppressLastZeroBytePackage
- com.ibm.ws.webcontainer.suppressServletExceptionLogging
- com.ibm.ws.webcontainer.throwMissingJspException
- **AIX** com.ibm.ws.webcontainer.tolerateLocaleMismatchForServingFiles
- com.ibm.ws.webcontainer.webgroupvhostnotfound
- com.ibm.ws.webcontainer.xPoweredBy
- com.ibm.wsspi.jsp.convertAttrValueToString
- com.ibm.wsspi.jsp.disableEICache
- com.ibm.wsspi.jsp.disableResourceInjection
- com.ibm.wsspi.jsp.disableTldSearch
- com.ibm.wsspi.jsp.enabledoublequotesdecoding
- com.ibm.wsspi.jsp.removexmlnsfromoutput
- **Distributed operating systems** DebugSessionCrossover
- DecodeUrlAsUTF8

- enableInProcessConnections
- fileServingEnabled, directoryBrowsingEnabled, and serveServletsByClassnameEnabled
- ForceSessionIdLengthCheck
- ForceSessionInvalidationMultiple
- httpsIndicatorHeader
- HttpSessionIdReuse
- listeners
- **z/OS** MutualAuthCBindCheck
- prependSlashToResource
- trusted
- UseOracleBLOB

z/OS Web container transport channel custom properties

- disableRequestMessageChunking
- maxRequestMessageBodySize
- sslCustomApplicationBufferSize
- useStrictSSLConnectTimeout

Web services security custom properties

- com.ibm.ws.wssecurity.createSTR
- com.ibm.ws.wssecurity.sc.FaultCode
- com.ibm.wsspi.wssecurity.Caller.assertionLoginConfig
- com.ibm.wsspi.wssecurity.config.disableWSSIfApplicationSecurityDisabled
- com.ibm.wsspi.wssecurity.config.gen.checkCacheUsernameTokens
- com.ibm.wsspi.wssecurity.config.request.setMustUnderstand and com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne
- com.ibm.wsspi.wssecurity.config.token.inbound.retryOnceAfterTrustFailure
- com.ibm.wsspi.wssecurity.consumer.timestampRequired
- com.ibm.wsspi.wssecurity.dsig.inclusiveNamespaces
- com.ibm.wsspi.wssecurity.dsig.oldEnvelopedSignature
- com.ibm.wsspi.wssecurity.generator.usewssobject
- com.ibm.wsspi.wssecurity.login.useSoap12FaultCodes
- com.ibm.wsspi.wssecurity.token.forwardable
- com.ibm.wsspi.wssecurity.token.username.addNonce and com.ibm.wsspi.wssecurity.token.username.addTimestamp
- com.ibm.wsspi.wssecurity.token.username.password.forwardable
- com.ibm.wsspi.wssecurity.token.username.verifyNonce and com.ibm.wsspi.wssecurity.token.username.verifyTimestamp
- com.ibm.wsspi.wssecurity.token.UsernameToken.disableUserRegistryCheck
- com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7

Web services security generic security token login module custom properties

- Callback handler custom properties for both token generator and token consumer bindings
- Callback handler custom properties for token generator bindings
- Callback handler custom properties for token consumer bindings

Web services security SAML token custom properties

- SAML token generator custom properties
- SAML token consumer custom properties

- SAML token custom properties for both token generator and token consumer
- Trust client custom properties

WebSphere MQ messaging provider custom properties

- WAS_EndpointInitialState

Log and trace file descriptions

This reference information describes the location, syntax, and usage of log and trace files generated by the product.

To open the information center table of contents to the location of this reference information, click the **Show in Table of Contents** button (🔗) on your information center border.

Administrator best practices

This reference information describes best practices and other considerations for administrators.

To open the Table of Contents to the location of this reference information, click the **Show in Table of Contents** button, 🔗, on your information center border.

Note: Do not manually edit configuration XML files or the `systemlaunch.properties` files. The `systemlaunch.properties` files contain default values to make servers behave as expected. If changes are made to these files, unexpected behavior might occur.

Supported configurations and limitations

This reference information discusses what the product does and does not support.

To open the information center table of contents to the location of this reference information, click the **Show in Table of Contents** button (🔗) on your information center border.

Troubleshooting tips

This reference information helps you prevent and recover from problems.

To open the information center table of contents to the location of this reference information, click the **Show in Table of Contents** button (🔗) on your information center border.

Intelligent Management reference

Reference information is organized to help you locate particular facts quickly.

Port number settings

Identify the default port numbers used in the various configuration processes to avoid port conflicts.

Open certain ports in any firewalls that are running between the deployment manager and node agent server processes to avoid conflicts with other assigned ports when configuring Version 6.1 resources or assigning port numbers to other applications. Additionally, when you configure a firewall, enable access to specific port numbers.

If you modify the ports, or if you want to confirm the assigned port, check the port assignments in the `app_server_root/profiles/myprofile/config/cells/mycell/nodes/mynode/serverindex.xml` file. If more than one node exists, you must check the port assignment for each node.

The following table is a list of port assignments that the node agent server uses by default. When you federate an application server node into a deployment manager cell, the deployment manager instantiates the node agent server process on the application server node. The node agent server uses these port assignments by default. When multiple processes share a port on the same node, the configuration uses the next number in the sequence for the additional processes. For example, if three processes are running, the BOOTSTRAP_ADDRESS port for each process is 2809, 2810, and 2811.

For a complete list of default port definitions, read about port settings. For information on security ports, read about configuring inbound transports.

Table 36. Default port definitions for the node agent server process

Port Name	Description	Default Value (increments for multiple processes)
BOOTSTRAP_ADDRESS	The TCP/IP port on which the name service listens. This port is also the Remote Method Invocation (RMI) connector port. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	2809
ORB_LISTENER_ADDRESS	The TCP/IP port on which the application server Object Request Broker (ORB) listens for requests. This also the port on which the location service daemon for the node listens. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	9100
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	The TCP/IP port on which the Secure Association Services (SAS) listen for inbound authentication requests. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	9901
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	The TCP/IP port on which the Common Secure Interoperability Verison 2 (CSIV2) Service listens for inbound client authentication requests. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	9202
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	The TCP/IP port on which the Common Secure Interoperability Verison 2 (CSIV2) Service listens for inbound server authentication requests. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	9201

Table 36. Default port definitions for the node agent server process (continued)

Port Name	Description	Default Value (increments for multiple processes)
NODE_DISCOVERY_ADDRESS	The TCP/IP port on which the node discovery service for the node agent listens. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	7272
NODE_MULTICAST_DISCOVERY_ADDRESS	The TCP/IP port for the multicast discovery service on which the node agent listens. Specify this port with the administrative console or with the chgwassvr script. For more information, read about the chgwassvr script.	5000
SOAP_CONNECTOR_ADDRESS	This port is required by every WebSphere process to enable SOAP connectivity for JMX calls when using wsadmin.	8879
OVERLAY_UDP_LISTENER_ADDRESS	Used for peer-to-peer (P2P) communication. The ODC (On Demand Configuration) and asynchronous PMI components use P2P as their transport. This port is required by every WebSphere Extended Deployment process.	11001
OVERLAY_TCP_LISTENER_ADDRESS	Used for P2P communication. The ODC (On Demand Configuration) and asynchronous PMI components use P2P as their transport. This port is required by every WebSphere Extended Deployment process.	11002
XD_AGENT_PORT	The deployment manager, the node agents, and the middleware agents each have one XD_AGENT_PORT. Note that, unlike the OVERLAY ports, the application servers are not configured with XD_AGENT_PORTS. Used to enable communication between the deployment manager, the node agents, and the middleware agents. The ODR uses this port to collect information from other servers, including node agents. This port should be available to all servers that the Intelligent Management ODR is managing.	7061
DRS_CLIENT_ADDRESS	Deprecation: This port is deprecated and is no longer used in the current version of the product.	7873

During the **addNode** command operation, the **filetransfer** application uses port 9090 by default. The **filetransfer** application uses the same HTTP transport port that is used by the administrative console. If security is enabled, the default secured port 9043 must be opened in the firewall. If you modify the ports,

or if you want to confirm the assigned port, check the port assignments in the `app_server_root\config\cells\cellname\nodes\nodename\servers\dmgr\server.xml` file.

Table 37. Default port definitions for the fileTransfer application

Port Name	Default Value
Default fileTransfer application Port	9090
Secured - Default fileTransfer application port	9043

When you federate an application server node with the embedded messaging server feature into a deployment manager cell, the deployment manager instantiates a Java Message Service (JMS) server process, `jmsserver`, on the application server node. The following table lists the port assignments that the JMS server uses by default:

Table 38. Default port definitions for the JMS server

Port Name	Default Value
JMSSERVER_DIRECT_ADDRESS	5559
JMSSERVER_QUEUED_ADDRESS	5558
SOAP_CONNECTOR_ADDRESS	8879
JMSSERVER SECURITY PORT	5557

Intelligent Management scripts

You can use the Intelligent Management scripts after you install the product. The scripts are in the `app_server_root/bin` directory. In addition, you can use a new series of scripting objects. To access the new scripting objects, invoke the `$AdminConfig types` command from the `wsadmin` tool.

Scripts

appEditionRename.py script

You can use the `appEditionRename.py` script to change the edition of all applications installed on a server or a cluster, excluding system applications.

Location

The `appEditionRename.py` script is in the `app_server_root/bin` directory.

Usage

The following syntax describes the default script usage:

```
./wsadmin.sh -lang jython -f appEditionRename.py
```

Parameters

-renameForCluster

Renames all non-system applications installed on a specified cluster.

Table 39. renameForCluster arguments. The following table lists each argument and its description.

Argument	Description
<code>newEdition</code>	Specifies the new edition name
<code>cluster</code>	Specifies the name of the target cluster

-renameForServer

Renames all non-system applications installed on a specified server.

Table 40. *renameForServer* arguments. The following table lists each argument and its description.

Argument	Description
node	Specifies the name of the node
server	Specifies the name of the server on the node

Options

-test

Allows users to preview the applications that will be renamed. All applications that will be renamed are listed.

-edition

Specifies the edition of an application.

-excludedApps=application_name

Allows users to exclude applications that are not to be modified. For example, if a user wants to exclude the applications of app1 and app1 that are part of edition two, then the user would set `-excludedApps=app1,app1-editiontwo`.

Usage

1. Identify the target cluster with applications to be renamed.

Run the script with the `-test` option to identify the applications that will be renamed.

```
-renameForCluster cluster1 2 -test
```

2. Ensure that only expected applications are renamed.

If applications need to be excluded, then use the `-excludedApps` option to prevent the applications from being renamed.

```
-renameForCluster cluster1 2 -test -excludedApps=AppB,AppC,AppD-edition2
```

3. Run the script without the `-test` option to rename the applications.

```
-renameForCluster cluster1 2 -excludedApps=AppB,AppC,AppD-edition2
```

arfmController.py script

You can use the `arfmController.py` script to force the autonomic request flow manager (ARFM) to forget all of its historical data.

Purpose

Use the `arfmController.py` script to cause the ARFM to forget all of its historical data. This is useful if the application characteristics suddenly change.

Location

The `arfmController.py` script is in the `app_server_root/bin` directory.

Usage

Run the `arfmController.py` script from a command line:

```
./wsadmin.sh -f arfmController.py -lang jython reset
```

checkHmLocation.jacl script

You can use the `checkHmLocation.jacl` script to locate the health controller. Locating the health controller is useful when you are troubleshooting your configuration.

Purpose

With the **checkHmmLocation.jacl** script, you can get the location of the health controller. The health controller can run in the node agent on any node in the cell that is not also running the deployment manager.

Location

The **checkHmmLocation.jacl** script is in the *app_server_root/bin* directory.

Usage

Before you run this script, verify that the deployment manager for the cell is running.

To run the **checkHmmLocation.jacl** script with the wsadmin tool, use the following command:

```
wsadmin.sh -f checkHmmLocation.jacl
```

Example

Sample output from running this script is as follows:

```
# WASX7209I: Connected to process "dmgr" on node eutil33Manager
using SOAP connector; The type of process is: DeploymentManager
looking for group name: HAMItemGroup:com.ibm.ws.xd.hmm.controller.HmmControllerImpl_com.ibm.ws.xd.hmm.
controller.HmmController_Default
```

Results:

```
{SERVER_NAME nodeagent}
{MEMBER_NAME HAMItemMember:com.ibm.ws.xd.hmm.controller.HmmControllerImpl_com.ibm.ws.xd.hmm.controller.
HmmController_Default_eutil33Network
\df245\nodeagent_1}
{NODE_NAME df245}
{GROUP_ID HAMItemGroup:com.ibm.ws.xd.hmm.controller.HmmControllerImpl_com.ibm.ws.xd.hmm.controller.
HmmController_Default}
```

The `NODE_NAME df245` indicates the specific location information for the health controller. In this case, the controller is running on the `df245` node.

checkPlacementLocation.jacl script

You can use the **checkPlacementLocation.jacl** script to locate the application placement controller. Locating the application placement controller is useful when you are troubleshooting your configuration.

Purpose

The **checkPlacementLocation.jacl** script returns the location information for the application placement controller. You can use this script for debugging the application placement feature. Run the script and then enable tracing to verify any problems that might occur.

Location

The **checkPlacementLocation.jacl** script is in the *app_server_root/bin* directory.

Usage

Before you run this script, verify that the deployment manager for the cell is running.

To run the **checkPlacementLocation.jacl** script with the wsadmin tool, use the following command:

```
wsadmin.sh -f checkPlacementLocation.jacl
```


Example

Sample output from running this script is as follows:

```
# WASX7209I: Connected to process "dmgr" on node eutil33Manager
using SOAP connector; The type of process is: DeploymentManager
Looking for group name: HAMItemGroup:com.ibm.ws.xd.placement.controller.impl.
PlacementControllerImpl_com.ibm.ws.xd.placement.controller.PlacementController_Default

Results:
{SERVER_NAME nodeagent}
{NODE_NAME xddesktop}
{MEMBER_NAME HAMItemMember:com.ibm.ws.xd.placement.controller.impl.PlacementControllerImpl_com.ibm.ws.xd.
placement.controller.PlacementController_Default}
{NODE_NAME df245}
{GROUP_ID HAMItemGroup:com.ibm.ws.xd.placement.controller.
PlacementControllerImpl_com.ibm.ws.xd.placement.controller.PlacementController_Default}
```

NODE_NAME df245 indicates the specific location information for the application placement controller. In this case, the controller is running on the df245 node.

createDynamicCluster.jacl script

You can use the **createDynamicCluster.jacl** script to create a dynamic cluster.

Purpose

The **createDynamicCluster.jacl** script creates a dynamic cluster and associates it with a node group.

You can also use the Jacl scripts to create and delete dynamic clusters. For more information about available options for configuring dynamic clusters, read about dynamic cluster administrative tasks.

Location

The **createDynamicCluster.jacl** script is in the *app_server_root/bin* directory.

Usage

To run the **createDynamicCluster.jacl** script with the wsadmin utility, use the following command:

```
wsadmin -profile createDynamicCluster.jacl node_group_name dynamic_cluster_name
```

In place of *node_group_name* and *dynamic_cluster_name*, use the name of the node group that you want to associate with your new dynamic cluster. Modify the wsadmin command to wsadmin.sh or wsadmin.bat, depending on your operating environment.

The code in the runtime creates a membership policy when you create your dynamic cluster. For example, if you indicate the node_group_1 node group, the membership policy of node_nodegroup='node_group_1' is created.

createodr.jacl script

You can use the **createodr.jacl** script to create an on demand router (ODR) from the command line.

Purpose

You can use the **createodr.jacl** script to create an ODR server.

To create an ODR, you must have administrator or configurator privileges.

Location

The `createodr.jacl` script is in the `app_server_root/bin` directory.

Usage

The default script usage follows:

```
./wsadmin.sh -f createodr.jacl
```

Procedures

Specify the following parameters with the `createodr.jacl` script:

- *nodename*: The name of the node on which you want to create your ODR.
 - *odr_template*: In Version 6.1, a template parameter is required. The following list contains the options for *odr_template*:
 - odr
 - sip_odr_server
 - http_sip_odr_server
- z/OS** The following list contains the options for *odr_template* when using the z/OS operating system:
- odr_zos
 - sip_odr_server_zos
 - http_sip_odr_server_zos
- *odr_name*: The name of the ODR you want to create.

Alternatively, you can first launch the scripting client. In wsadmin:

```
wsadmin> source createodr.jacl  
wsadmin> createodr nodename
```

In place of *nodename*, specify the name of the node on which you want to create your ODR.

Example

Create an ODR.

```
./wsadmin.sh -f createodr.jacl node01 odr
```

z/OS

```
./wsadmin.sh -f createodr.jacl node01 odr_zos
```

Create an ODR that is named `great_ODR`.

```
./wsadmin.sh -f createodr.jacl node01 http_sip_odr_server great_ODR
```

coregroupsplit.py script

You can use the `coregroupsplit.py` script to split your existing cell into multiple core groups. Consider running this script if you have more than 40 WebSphere Application Server related processes such as application servers, node agents, and on demand routers (ODRs) defined in your core group.

Purpose

The `coregroupsplit.py` script divides your existing cell into multiple core groups. If a server changes its core group membership, then you must restart the entire cell to prevent partitions from forming. For this reason, the default options used by this script do not change the core group membership of servers that are members of any core groups other than `DefaultCoreGroup`.

Running the script attempts to satisfy the following best practices for core groups:

- Each core group must have fewer than 40 servers. This best practice is possible only when you have an adequate ratio of nodes to servers in your cell.
- Each core group must be equipped with at least three core group bridges. This best practice is possible only when you have an adequate ratio of nodes to servers in your cell.
- Each cluster must remain fully mapped to a single core group.
- If you use the **-createbridges** parameter, the core group bridges and coordinators are preferably hosted on otherwise idle nodes.

This script also tunes the high availability manager for optimum performance. By default, the script configures a core group bridge on each node agent in the cell. The script increases the number of core groups until it reaches the optimal level, depending on the number of nodes and servers in the cell. The core group bridge node agents are configured as a part of the `DefaultAccessPointGroup` access point group in a mesh topology. In the preferred mesh topology, all of the access points are collected into a single access point group. As a result, all bridge interfaces can directly intercommunicate.

Remember that you must give core group bridges at least 512 MB of Java virtual machine (JVM) space.

Location

The `coregroupsplit.py` script is in the `app_server_root/bin` directory.

Usage

The default script usage follows:

```
./wsadmin.sh -lang jython -f coregroupsplit.py
```

Running this script might result in unbalanced core groups, in which some core groups are loaded more or less than other core groups. You can rerun this script to rebalance core group membership, but in this case you must restart the entire cell for the changes to take effect. To rerun the script, use the following command:

```
./wsadmin.sh -lang jython -f coregroupsplit.py -reconfig
```

Parameters

-reconfig

Performs a full reconfiguration to rebalance the distribution of servers among the core groups.

-linked

Creates a ring topology of core group bridges.

-createbridges

Creates separate core group bridge processes instead of creating the bridge in the node agent.

-numcoregroups

Specifies the number of core groups to create.

-datastacksize

Specifies a number of megabytes that overrides the default data stack size.

-proxycoregroup

Places the on demand routers (ODR) and proxy servers in a separate core group.

-odrcoregroup

Places the on demand routers (ODR) and proxy servers in a separate core group.

-nosave

Does not save any changes made to the core group. You can use this option to test setting parameters and running the script.

-debug

Prints troubleshooting information.

-nodesPerCG: *number*

Specifies the number of node agents that are required for each core group.

-numberOfServersPerCG: *number*

Specifies the maximum number of servers for each core group.

-bridgeHeapSize: *number*

Specifies the server heap size of the core group bridge in megabytes.

Examples

The following example results in a linked topology where the core group bridges are connected in a ring:

```
./wsadmin.sh -lang jython coregroupsplit.py -linked
```

You can also use this script to create static cluster that are dedicated as core group bridges for communication within the core group. Use the following example:

```
./wsadmin.sh -lang jython coregroupsplit.py -createbridges
```

DataReaderWrapper class

You can use a command-line Java class to configure the runtime operations reports to log data for reuse in external programs.

Purpose

You can read and filter the data logs into your own files by invoking this Java class from a command line.

Location

The **DataReaderWrapper** class is located in the *app_server_root/plugins/com.ibm.ws.ve.runtime.was.jar* file.

The full name of the class is:

```
com.ibm.ws.xd.visualizationengine.cacheservice.DataReaderWrapper
```

The full class name is required when running the Java command with that class:

```
java com.ibm.ws.xd.visualizationengine.cacheservice.DataReaderWrapper ServerStatsCache log
data.out 1113189469300 1113189472500 name server1 node dabtcNode01
```

Usage

To run the **DataReaderWrapper** class from a command line, use the following command:

```
java DataReaderWrapper
```

Parameters

The following parameters are available:

- *log_type*

Type of log. You can use the *log_type* parameter alone to retrieve a help list of properties and their descriptions for the specified log. These properties can be used for the *property_filter* parameter.

Example:

```
java DataReaderWrapper ServerStatsCache
```

- *log_directory*

Directory of the log file.

Example:

```
java DataReaderWrapper ServerStatsCache log
```

- *output_file_name*

Specifies the name of the output file.

Example:

```
java DataReaderWrapper ServerStatsCache log data.log
```

- *interval_start_time*

The start time of the log interval in long or string form.

Examples:

- Long form:

```
java DataReaderWrapper ServerStatsCache log data.log 1113189469300
```

- String form (yy.MM.dd_HH.mm.ss):

```
java DataReaderWrapper ServerStatsCache log data.log 05.05.18_12.02.45
```

- *interval_end_time*

The end time of the log interval in long or string form.

Examples:

- Long form:

```
java DataReaderWrapper ServerStatsCache log data.log 1113189469300 1113189472500
```

- String form (yy.MM.dd_HH.mm.ss):

```
java DataReaderWrapper ServerStatsCache log data.log 05.05.18_12.02.45 05.05.18_01.02.00
```

- *property_filter*

Property value pairs.

Example:

```
java DataReaderWrapper ServerStatsCache log data.log 1113189469300 1113189472500 name server1 node dabtcNode01
```

- ?

Displays syntax help.

Example: `java DataReaderWrapper ?`

In the following example, the `DataReaderWrapper` reads the `ServerStatsCache` log files in the `log` directory from the time 1113189469300 to 1113189472500 on the `server1` machine and the `dabtcNode01` node:

```
java DataReaderWrapper ServerStatsCache log data.out 1113189469300 1113189472500 name server1 node dabtcNode01
```

deleteDynamicCluster.jacl script

You can use the `deleteDynamicCluster.jacl` script to delete dynamic clusters.

Purpose

The `deleteDynamicCluster.jacl` script deletes a dynamic cluster.

Although you can still use the Jacl scripts to create and delete dynamic clusters, the administrative tasks provide all of the options that are available for configuring dynamic clusters in Version 6.1 and later. See For more information about creating and deleting dynamic clusters, read about dynamic cluster administrative tasks.

Location

The `deleteDynamicCluster.jacl` script is in the `app_server_root/bin` directory.

Usage

To run the **deleteDynamicCluster.jacl** script with the `wsadmin` utility, use the following command:

```
wsadmin -f deleteDynamicCluster.jacl node_group_name dynamic_cluster_name
```

In place of *node_group_name* and *dynamic_cluster_name*, use the name of the node group that you want to disassociate with your deleted dynamic cluster. You might need to modify the `wsadmin` command to **wsadmin.sh** or **wsadmin.bat**, depending upon your operating environment.

deleteodr.jacl script

You can use the **deleteodr.jacl** script to delete an on demand router (ODR) from the command line.

Purpose

You can use the **deleteodr.jacl** script to delete an ODR.

To delete an ODR, you must have administrator or configurator privileges.

Location

The **deleteodr.jacl** script is located in the *app_server_root/bin* directory.

Usage

The default script usage follows:

```
./wsadmin.sh -f deleteodr.jacl
```

Procedure

Specify the following parameter with the **deleteodr.jacl** script:

nodename

The name of the node on which an ODR with the name *odr* exists.

Alternatively, you can first launch the scripting client. In `wsadmin`:

```
wsadmin> source deleteodr.jacl  
wsadmin> deleteodr nodename
```

In place of *nodename*, specify the name of the node on which your ODR exists.

Example

Delete an ODR.

```
./wsadmin.sh -f deleteodr.jacl node01
```

dumpOdrState.jacl script

You can use the **dumpOdrState.jacl** script to dump on demand router (ODR) states, and for problem diagnosis and debugging purposes. The detailed state of the queues and throttles of each gateway running in the ODR, as well as other global counters, are returned.

Purpose

You can use the **dumpOdrState.jacl** script to dump ODR states.

Location

The `dumpOdrState.jacl` script is in the `app_server_root/bin` directory.

Usage

The default script usage follows:

```
./wsadmin.sh -profile dumpOdrState.jacl
```

Procedures

Specify the following parameter with the `dumpOdrState.jacl` script:

odr_name

Specifies the name of the ODR.

Example

```
./wsadmin.sh -profile dumpOdrState.jacl myODR
```

Results

The `dumpOdrState.jacl` script dumps the ODR state of the ODR that you specified by invoking the `WsmmProxyMBean` managed bean (MBean).

This script returns one string that contains a description of the state of every Autonomic Request Flow Manager (ARFM) gateway in the ODR. The description lists the queued and running requests, and the throttling state of each queue. The description also contains debug-level information.

dumpIMPState.py script

Use the `dumpIMPState.py` script to discard of various Intelligent Management processes. You can use this script for problem diagnosis and debugging purposes when requested by IBM support. The detailed state of the on demand router (ODR), the deployment manager, and application server processes are displayed when using this script.

Purpose

You can dump the state of active Intelligent Management processes with this script.

trns: The WebSphere Virtual Enterprise command that equates to `dumpIMPState.py` is `dumpXdState.py`. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the `dumpXdState.py` command, which operates the same as the `dumpIMPState.py` command.

Location

The `dumpIMPState.py` script is in the `app_server_root\bin` directory.

Usage

The `dumpIMPState.py` script must be run from the `app_server_root\bin` directory using `wsadmin.sh`. To automatically detect active Intelligent Management processes and write their state information to a file, use the following command:

```
wsadmin.sh -lang jython -f dumpIMPState.py --auto > output_file_name
```

Where `output_file_name` is the name of a file to save the output.

Options

--auto

Exports information for all of the Intelligent Management processes into the specified text file. This option exports a large amount of data, for example, a small topology might result in a text file that is over 2500 kilobytes.

--dom

Specifies a domain. The default is *, or all domains.

--dmgr=*deployment_manager_name*

Exports information for the deployment manager that you specify in the option.

--help

Displays the help for this script.

--odr=*odr_name*

Exports information for the ODR that you specify in the option. The output for this option is similar to output that you get from the `dumpOdrState.jac1` script, but includes additional information from internal processes.

--svr=*server_name*

Exports information for the server that you specify in the option.

The following example automatically detects Intelligent Management processes and writes their state information to the file `file1`:

```
wsadmin.sh -lang jython -f dumpIMPState.py --auto > file1
```

Results

The `dumpIMPState.py` script dumps the state of various Intelligent Management processes by invoking the managed beans (MBean) of those processes. The dumped information is written to the output file name specified on the command line. Provide this file to IBM support for diagnosis and debugging of problems.

Health controller commands with the AdminConfig object

You can use the AdminConfig object to modify the health controller settings. You can change the controller cycle, prohibit server restarts at certain times of the day, and so on.

Purpose

Health management comes equipped with smart defaults that accommodate most environments. However, if you discover that your health controller is not working the way you want, then tune the default parameters. You can change these settings in the administrative console or with the wsadmin tool.

Usage

To change configuration settings for the health controller, you must have configurator or administrator privileges. If you have operator privileges, you can only change the runtime configuration.

To edit the health controller properties, run the following commands:

```
.\wsadmin.sh -lang jython
hcid = AdminConfig.getid("/HealthController:/")
AdminConfig.modify(hcid, [{"attribute_name", value}])
AdminConfig.save()
```

Attributes

You can edit the following attributes:

controlCycleLength

Specifies the time between consecutive health checks to determine if a health policy condition is breached.

Default: 5

enable

Specifies if health monitoring is enabled.

Default: true

maxConsecutiveRestarts

Specifies the number of attempts to revive a server after a restart decision is made. If this number is exceeded, a failed operation is assumed and restarts are disabled for the server.

Valid values: whole numbers between 1 and 5

Default: 3

minRestartInterval

Controls the minimum amount of time that must pass between consecutive restarts of an server instance.

Valid values: The value can range from 15 minutes to 365 days, inclusive. Indicate the units with the **minRestartIntervalUnits** attribute. A value of 0 disables the minimum restart value.

Default: 0 (disabled)

minRestartIntervalUnits

Indicates the units to use with the **minRestartInterval** attribute

Valid values: 2 (minutes), 3 (hours), or 4 (days)

Default: 2

prohibitedRestartTimes

Specifies the times and days of the week during which a restart of an application server instance is prohibited. You cannot change this attribute with the AdminConfig object. To update the prohibited restart times in the administrative console, click **Operational policies > Autonomic managers > Health controller**. Edit the **Prohibited restart times** field.

properties

Specifies a custom property on the health controller.

restartTimeout

Specifies a number of minutes to wait for a server to stop before explicitly checking its state and attempting another start.

Valid values 1 to 60 minutes, specified as a whole number

Default: 5

Example

The following command example changes the restart timeout setting:

```
hcid = AdminConfig.getid("/HealthController:/")
AdminConfig.modify(hcid, [["restartTimeout", 6]])
AdminConfig.save()
```

The following command sets the number of minutes for the approval timeout of runtime tasks for the health controller by specifying the `com.ibm.ws.xd.hmm.controller.ControlConfig.approvalTimeOutMinutes` Java virtual machine (JVM) custom property. In this example specifically, the value of the approval timeout is set to 40 minutes:

```
.\wsadmin.sh -lang jython
hcid = AdminConfig.getid("/HealthController:/")
AdminConfig.create('Property', hcid, [['name', 'com.ibm.ws.xd.hmm.controller.ControlConfig.
approvalTimeOutMinutes'], [['value', '40']]) AdminConfig.save()
```

HmmControllerProcs.jacl script

Using the **HmmControllerProcs.jacl** script, you can modify your health management runtime configuration to enable or disable the health controller and automated restarts for the server, and set or retrieve values for other health management settings.

Purpose

To change runtime settings, you must have operator or administrator administrative privileges.

You can use the **HmmControllerProcs.jacl** file to complete the following actions.

- Enable or disable the health controller.
- Enable or disable automated restarts for the server.
- Set or retrieve values for other health management settings.

The **HmmControllerProcs.jacl** script only modifies the runtime configuration.

Location

The **HmmControllerProcs.jacl** script is in the *app_server_root/bin* directory.

Usage

Run the following command.

```
wsadmin -profile HmmControllerProcs.jacl -c "insert_procedure_parameters"
```

Replace the *insert_procedure_parameters* variable with the name of the procedure and the proper variable values to complete your changes. You might have to modify the wsadmin command to wsadmin.sh or wsadmin.bat, depending on your operating environment.

Procedures

To see a list of all available procedures, use the following command.

```
wsadmin -profile HmmControllerProcs.jacl -c "help"
```

enable

Enables the health controller.

disable

Disables the health controller.

isEnabled

Verifies that the health controller is enabled.

enableServer *node_name server_name*

Enables automated restarts for the server on the specified node.

disableServer *node_name server_name*

Disables automated restarts for the server on the specified node.

isServerEnabled *node_name server_name*

Verifies if automated restarts are enabled for the server on the specified node.

isNodeMaintenance *node_name*

Verifies if the health controller acknowledges that the specified node is in maintenance mode.

getMaxConsecutiveRestarts

Returns the number of attempts to revive an application server after a restart is performed. If the maximum value is exceeded, the server is declared failed and server restarts are disabled.

setMaxConsecutiveRestarts *number_of_consecutive_restarts*

Sets the number of attempts to revive an application server after a restart is performed.

getControlCycleLength

Returns the length of time, in minutes, between health policy checks on the application server instances to determine if breaches occurred.

setControlCycleLength *time_in_minutes*

Sets the length of time, in minutes, between health policy checks on the application server instances to determine if breaches occurred.

getMinRestartInterval

Returns the length of time that must pass between application server instance restarts.

setMinRestartInterval *time_in_minutes*

Sets the length of time that must pass between application server instance restarts.

getRestartTimeout

Returns the length of time that the controller uses to wait for start and stop events during a restart before polling the server status.

setRestartTimeout *time_in_minutes*

Sets the length of time that the controller uses to wait for start and stop events during a restart before polling the server status.

Example

The following command enables automatic restarts on the `server_1` server, which runs on the `node_1` node:

```
wsadmin.sh -profile HmmControllerProcs.jacl -c "enableServer node_1 server_1"
```

importOverlayConfig.py script

When setting up a star topology, you can use the **importOverlayConfig.py** script to configure the overlay communication between multiple cells.

Purpose

Use the **importOverlayConfig.py** script to enable communication between Intelligent Management cells. Each cell contains a `overlaynodes.config` file that must be copied between the cells to enable multi-cell performance management.

Location

The **importOverlayConfig.py** script is in the `app_server_root/bin` directory.

The `overlaynodes.config` file is in the `app_server_root/profiles/deployment_manager_profile_name/config/cells/cell_name` directory of each cell.

Example

See the following examples to configure the overlay communication between multiple cells. Consider a scenario in which there are three cells: `CellA` is the center cell, and `CellB` and `CellC` are the point cells.

Run the **importOverlayConfig.py** script from the deployment manager on `CellA` to link the overlay anchor transports of `CellA` to `CellB`.

```
./wsadmin.sh -profileName profile_name -f importOverlayConfig.py link path_to_CellB_overlaynodes.config_file
```

Run the **importOverlayConfig.py** script from the deployment manager on CellA to link the overlay anchor transports of CellA to CellC.

```
./wsadmin.sh -profileName profile_name -f importOverlayConfig.py link path_to_CellC_overlaynodes.config_file
```

Run the **importOverlayConfig.py** script from the deployment manager on CellB to link the overlay anchor transports of CellB to CellA.

```
./wsadmin.sh -profileName profile_name -f importOverlayConfig.py link path_to_CellA_overlaynodes.config_file
```

Run the **importOverlayConfig.py** script from the deployment manager on CellC to link the overlay anchor transports of CellC to CellA.

```
./wsadmin.sh -profileName profile_name -f importOverlayConfig.py link path_to_CellA_overlaynodes.config_file
```

Example

See the following example to unlink two cells.

```
./wsadmin.sh -profileName profile_name -f importOverlayConfig.py unlink path_to_target_cell_overlaynodes.config_file
```

linkCells script

When you set up a star topology, you can use the **linkCells** script to configure the overlay communication between multiple cells.

Purpose

Use the **linkCells** script to enable communication between a Intelligent Management cell containing servers that are enabled with an on demand router (ODR) that routes work requests to other administrative cells.

Location

The **linkCells** script is in the `app_server_root/bin` directory.

Usage

Run the **linkCells** script from the center cell to link the center cell with a point cell:

```
./linkCells.sh centerHost:center_cell_soap_port:user_id:password pointHost:point_cell_soap_port:user_id:password
```

Example

Consider a scenario in which there are two cells, center and point1, with security enabled in both. For the center cell, the host name of the deployment manager is centerHost, the SOAP port is 8879, the user name is centerUID, and the password is centerPWD. For the point cell, the host name of the deployment manager is point1Host, the SOAP port is 8880, the user name is point1UID, and the password is point1PWD. The following example illustrates how to link the center and point1 cells together as is needed to support a star topology.

```
./linkCells.sh centerHost:8879:centerUID:centerPWD point1Host:8880:point1UID:point1PWD
```

Troubleshooting

When you run the **linkCells** script, the following error messages might be displayed. To resolve the errors, verify that the `com.ibm.ssl.enableSignerExchangePrompt` property in the `profile_home/properties/ssl.client.props` file is set to `gui`, `true`, or `stdin`. By setting this property, clients can obtain a signer certificate from the server, and thus communicate with Intelligent Management.

When the `com.ibm.ssl.enableSignerExchangePrompt` property is set to `gui` or `true`, a signer-exchange window is displayed, and you are asked to accept or reject the certificate. If you accept the certificate, it is installed in the trust store automatically and the handshake succeeds. If you reject the certificate, it is not installed in the trust store and the handshake fails since the certificate is not trusted.

When the `com.ibm.ssl.enableSignerExchangePrompt` property is set to `stdin`, a signer-exchange ASCII prompt is displayed, and you are asked to accept or reject the certificate. If you accept the certificate, it is installed in the trust store automatically and the handshake succeeds. If you reject the certificate, it is not installed in the trust store and the handshake fails since the certificate is not trusted.

```
$ ./linkCells.sh centerHost:center_cell_soap_port:user_id:password pointHost:point_cell_soap_port:user_id:password
```

```
"Begin linking cells..."
```

```
WASX7209I: Connected to process "dmgr" on node dmgr using SOAP connector. The type of process is: DeploymentManager
```

```
CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN "CN=edgeaphid16.rtp.raleigh.ibm.com, OU=e16VEcell, OU=edgeaphid16CellManager02, O=IBM, C=US" was sent from target host:port "9.42.96.77:8915". The signer may need to be added to local trust store "c:/AutoWAS2/09072011/WAS/profiles/node1/etc/trust.p12" located in SSL configuration alias "DefaultSSLSettings" loaded from SSL configuration file "file:c:\AutoWAS2\09072011\WAS\profiles\node1\properties\ssl.client.props". The extended error message from the SSL handshake exception is: "PKIX path building failed: java.security.cert.CertPathBuilderException: PKIXCertPathBuilderImpl could not build a valid CertPath.; internal cause is: java.security.cert.CertPathValidatorException: The certificate issued by CN=edgeaphid16.rtp.raleigh.ibm.com, OU=Root Certificate, OU=e16VEcell, OU=edgeaphid16CellManager02, O=IBM, C=US is not trusted; internal cause is: java.security.cert.CertPathValidatorException: Certificate chaining error".
```

```
CWPKI0040I: An SSL handshake failure occurred from a secure client. The server's SSL signer has to be added to the client's trust store. A retrieveSigners utility is provided to download signers from the server but requires administrative permission. Check with your administrator to have this utility run to setup the secure environment before running the client. Alternatively, the com.ibm.ssl.enableSignerExchangePrompt can be enabled in ssl.client.props for "DefaultSSLSettings" in order to allow acceptance of the signer during the connection attempt.
```

```
WASX7023E: Error creating "SOAP" connection to host "edgeaphid16.rtp.raleigh.ibm.com"; exception information: com.ibm.websphere.management.exception.ConnectorNotAvailableException: [SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket: javax.net.ssl.SSLHandshakeException: com.ibm.jsse2.util.g: PKIX path building failed: java.security.cert.CertPathBuilderException: PKIXCertPathBuilderImpl could not build a valid CertPath.; internal cause is: java.security.cert.CertPathValidatorException: The certificate issued by CN=edgeaphid16.rtp.raleigh.ibm.com, OU=Root Certificate, OU=e16VEcell, OU=edgeaphid16CellManager02, O=IBM, C=US is not trusted; internal cause is: java.security.cert.CertPathValidatorException: Certificate chaining error; targetException=java.lang.IllegalArgumentException: Error opening socket: javax.net.ssl.SSLHandshakeException: com.ibm.jsse2.util.g: PKIX path building failed: java.security.cert.CertPathBuilderException: PKIXCertPathBuilderImpl could not build a valid CertPath.; internal cause is: java.security.cert.CertPathValidatorException: The certificate issued by CN=edgeaphid16.rtp.raleigh.ibm.com, OU=Root Certificate, OU=e16VEcell, OU=edgeaphid16CellManager02, O=IBM, C=US is not trusted; internal cause is: java.security.cert.CertPathValidatorException: Certificate chaining error]
```

```
WASX7213I: This scripting client is not connected to a server process; please refer to the log file c:\AutoWAS2\09072011\WAS\profiles\node1\logs\wsadmin.traceout for additional information.
```

manageODC.py script

`manageODC.py` permits the interaction with the On Demand Configuration (ODC) tree. The ODC tree is an in-memory representation of the state of a WebSphere Application Server cell.

Purpose

The `manageODC.py` script helps with troubleshooting On Demand Router (ODR) issues. You can also use the script to alter the ODC tree.

Note: Forced alterations to the ODC tree might require a restart of the cell in order to correct inadvertent ODC tree corruption.

trns: The WebSphere Virtual Enterprise command that equates to `manageODC.py` is `ve_manageODC.py`. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the `ve_manageODC.py` command, which operates the same as the `manageODC.py` command.

Location

The `manageODC.py` script is in the `app_server_root/bin` directory.

Usage

To obtain the usage information for `manageODC.py`, run:

```
./wsadmin.sh -lang jython -f manageODC.py
```

or

```
./wsadmin.sh -lang jython -f manageODC.py operation --help
```

Generate a `target.xml` file to determine the ODC names to plug into the script.

Operations

You can perform the following operations with the `manageODC.py` script:

- **getTargetTree:** Retrieves the target tree.
 - *nodeName:* Specifies the name of the WebSphere node that contains the server from which the tree is retrieved.
 - *serverName:* Specifies the name of the server from which the tree is retrieved.
- **getP2PMemberData:** Retrieves data about peer-to-peer (P2P) members.
 - *nodeName:* Specifies the name of the WebSphere node that contains the server from which the P2P member data is retrieved.
 - *serverName:* Specifies the name of the server from which the P2P member data is retrieved.
- **generateHAPPluginCfgs:** Generates the `plugin-cfg.xml` file.
 - *generationDefinitionNames:* Is a comma-separated list of generation names that configured via cell custom properties of the form `ODCPluginCfg`. For example: `ODCPluginCfg_1, ODCPluginCfg_2`
 - *nodeName:* Specifies the name of the WebSphere node that contains the server that generates `plugin-cfg.xml`.
 - *serverName:* Specifies the name of the server to that generates `plugin-cfg.xml` file.

Example

Generate a `target.xml` file to determine the ODC names to include in the script. The following code example shows a shortened version of a `target.xml` file, where parameters for the cell, node, and server that you want to use in the script are located.

```
cellGroup name="target">
  !-- cell section -->
  <cell name="Cell1">
    !-- node section -->
    <node name="metis07">
      !-- server section -->
      <server name="odr">
        <property name="state" priority="1" value="STOPPED" />
      </server>
    </node>
  </cell>
</cellGroup>
```

To delete the server named `odr` from the ODC tree, type the following command:

```
./wsadmin.sh -lang jython -f manageODC.py removeODCNode /cell/Cell1/node/metis07/server/odr <myNode> <myServer>
```

Note that `/cellGroup/target` is never specified as part of the path.

In this example, the property ODC object is named `state`, with a value of `STOPPED`, and a priority of 1. To change the property to `STARTED`, use the following command:

```
./wsadmin.sh -lang jython -f manageODC.py modifyODCProperty /cell/Cell11/node/metis07/server/odr state
1::STARTED <mynode> <myserver>
```

manageODR.py script

You can use the `manageODR.py` script to manage custom logging and create a cluster of on demand routers (ODR).

Location

The `manageODR.py` script is located in the `app_server_root/bin` directory. Before running this script, ensure that you have the environment variable `WAS_HOME` configured to point to the directory of your WebSphere installation.

Operations

You can perform the following operations with the `manageODR.py` script:

- **convertToCluster** `<node:odr> <cluster>`: Convert an existing ODR on node `node` with name `odr` to an ODR cluster named `cluster`
- **addToCluster** `<node:odr> <cluster>`: Create a new ODR on node `node` with name `odr` and add it to ODR cluster `cluster`.
- **insertCustomLogRule** `<node:odr> <cluster><odrServerOrCluster> <ruleNumber> <condition> <logFileFormat>`: Insert a custom log rule into the list at position `<ruleNumber>`.
- **RemoveCustomLogRule** `<odrServerOrCluster> <ruleNumber>`: Remove the custom log rule number `<ruleNumber>`.
- **listCustomLogRules** `<odrServerOrCluster>`: List the custom log rules associated with an ODR or ODR cluster.
- **insertCustomLogRule** `<node:odr> <cluster><odrServerOrCluster> <ruleNumber> <condition> <logFileFormat>`: Insert a custom log rule into the list at position `<ruleNumber>`. If `<ruleNumber>` is larger than the number of rules in the list, the new rule is appended to the list with the next available rule number.

where:

Table 41. Arguments

Argument	Description
<code><node:odr></code>	Name of the node and ODR. For example, <code>mynode:myodr</code> .
<code><cluster></code>	Name of an ODR cluster.
<code><odrServerOrCluster></code>	Is either <code><node:odr></code> or <code><cluster></code>
<code><ruleNumber></code>	Number of the rule, from 1 to the total number of rules (or larger for <code>insertCustomLogRule</code>).
<code><condition></code>	Condition (that is, boolean expression) which must evaluate to true in order to trigger the associated custom logging.
<code><logFileFormat></code>	Specification denoting the file name and format of the log entry. For more information, read about custom log file format.

The following example shows how to create an HTTP ODR named `odr`, on `node 1`:

```
wsadmin.sh -f createodr.jacl node1 odr odr
```

The following example shows how to convert an ODR named *odr* on node *node1* to an ODR cluster named *ODRCluster*:

```
wsadmin.sh -f manageODR.py -lang jython convertToCluster node1:odr ODRCluster
```

The following example shows how to add a custom log rule to put all requests whose service time is longer than 2 seconds in the *slow.log* custom log, and include the application server to which the request was sent and the service time:

```
wsadmin.sh -f manageODR.py -lang jython insertCustomLogRule myNode02:odr1 1 "service.time > 2000" "slow.log %t %r %Z %T"
```

The following example shows how to add a custom log rule to put all 503 responses in *503.log*. The rule is added at position 2. If there is an existing rule at position 2, the new rule is inserted before the existing rule.

```
wsadmin.sh -f manageODR.py -lang jython insertCustomLogRule myNode02:odr1 2 "response.code = 503" "503.log %t %r %s"
```

The following example shows how to add a custom log rule at position 1 to log errors that can occur when writing a response to a client in the *response.write.error.log*:

```
wsadmin.sh -f manageODR.py -lang jython insertCustomLogRule node1:odr 1 "response.write.error" "response.write.error.log %t %r %s"
```

manageBBSON.py script

Use the **manageBBSON.py** script to gather the statistics usage of the BBSON bulletin board and periodically dump BBSON states. The statistics are available in the standard output and trace log files.

Purpose

For problem diagnosis and debugging purposes, use the **manageBBSON.py** script to collect BBSON state dumps from every process in the product cell.

Location

The **manageBBSON.py** script is in the *app_server_root/bin* directory.

trns: The WebSphere Virtual Enterprise command that equates to **manageBBSON.py** is **manageWVEBB.py**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **manageWVEBB.py** command, which operates the same as the **manageBBSON.py** command.

Usage

Run the following command on each Intelligent Management cell:

```
wsadmin -f manageBBSON.py operation [nodeName serverName]
```

You must specify the *operation* variable, which can be one of the following choices:

- **dump:** Dumps the state of the bulletin board
- **getsnapshot:** Obtains statistics about the current state of the bulletin board
- **getStatisticsPrintInterval:** Shows the current interval between printing bulletin board statistics in seconds
- **setStatisticsPrintInterval:** Sets the interval between printing bulletin board statistics in seconds

If you do not specify the `[nodeName serverName]` argument, the MBean is invoked against every process in the cell. Provide a node name and server name to narrow the invocation to the specific node and server that are listed.

Counter definitions

Name	Description
subjCreate	The number of created Subjects
subjForget	The number of closed Subjects
subscrCreate	The number of created SubjectSubscriptions
subscrClose	The number of closed SubjectSubscriptions
postCreate	The number of created SubjectPosts
postClose	The number of closed SubjectPosts
localUpdate	The number of updates that are performed by local posters
remoteUpdate	The number of updates that are performed by remote posters
localPostSizeSum	The data length in bytes that is sent by the local update calls
remotePostSizeSum	The data length in bytes that is sent by the remote update calls
localCallbackCall	The number of the update callback calls
localCallbackNumPostersSum	The number of posters that data is passed to in the local update callbacks
localCallbackPostSizeSum	The size in bytes of the data payloads that are passed in the update callbacks
localCallbackTimeSum	The time that is spent during the local update callback calls
localSubjPostCallback	The number of callbacks that are registered through SubjectPost#registerCallback
outOpenedTCP	The number of outgoing TCP connections that are open
outClosedTCP	The number of outgoing TCP connections that are closed
inOpenedTCP	The number of incoming TCP connections that are open
inClosedTCP	The number of incoming TCP connections that are closed
nothingToSend	The number of times an outgoing connection is opened with nothing to send at the moment the connection is completed

MirrorCell script

You can use the **MirrorCell** script to create representations of servers that are running previous versions of WebSphere Application Server in the Intelligent Management environment.

Purpose

The **MirrorCell** script creates a representation in the Intelligent Management administrative console of WebSphere Application Server application servers from previous versions, such as Version 5.1 or Version 6. Before you run this script you must generate the mappings file.

Location

The **MirrorCell** script is in the *app_server_root/bin* directory.

Usage

```
MirrorCell.sh [-props PROPS] [-mode MODE] [-nosec, -nosave]
```

Parameters

PROPS

Specifies the full path to the properties file.

MODE

The default mode is **ALL** if you do not specify the value or specify a value that is not valid. You can specify the following modes:

- **ALL**: Runs through every phase of the process.
- **READ**: Reads your cell configuration into an XML definition file only.
- **DIFF**: Differentiates your previously known configuration from the new configuration to detect changes or unnecessary configurations.
- **WRITE**: Creates the new configurations from the XML definitions file.
- **PROPSGEN**: Generates the properties file only. One useful application of the **PROPSGEN** mode is to run this mode as a cell administrator, set the user name and password while in the default encryption mode, and then give the properties file to a non-administrator to use when running the scripts. This action ensures that only the initial administrator that generates the file knows the values for the user name and password.

Options

-nosec

Specifies that your properties file does not encrypt the user name and password; however you must then use this option every time you run the scripts. Otherwise, the logic assumes that you are using encrypted data.

-nosave

Specifies that the script performs trial run and does not persist any of the changes that were made. To ensure that the script runs correctly, use this option when you first run the script.

The following example creates the new configurations only and disables the persistence of changes:

```
./MirrorCell.sh -mode WRITE -nosave
```

odrDebug.py script

You can use the **odrDebug.py** script to diagnose errors that an on demand router (ODR) returns.

Purpose

You can use the **odrDebug.py** script to diagnose errors that an ODR returns, such as 404 errors and 503 errors.

Location

The **odrDebug.py** script is located in the *app_server_root/bin* directory.

Usage

The default script usage follows:

```
./wsadmin.sh -lang jython -f odrDebug.py
```

Operations

Specify the following parameters with the `odrDebug.py` script:

setHttpDebug:

- **nodeName:** Specifies the name of the node that you want to debug.
- **odrName:** Specifies the name of the ODR that you want to debug.
- **errorCode:** Specifies the HTTP error code. You can specify 404, 503, and so on.
- **expression:** Specifies if the error code is being debugged. Specify `false` to disable debugging when the error occurs. Specify `true` to enable debugging when the error occurs.
- **debugLevel:** Specifies how much information is provided for debugging. You can use one of the following debug levels:
 - 0: prints a concise description on a single line
 - 1: prints a subset of `target.xml` file information
 - 2: prints the entire `target.xml` file

Example

```
./wsadmin.sh -lang jython -f odrDebug.py setHttpDebug my_node my_ODR 503 true 1
```

Result

If a 503 error occurs because the servers are not started, the `SystemOut.log` file contains the following message:

```
[11/3/07 19:20:00:427 EDT] 00000034 HttpDebugResp I
URL: http://draco02.rtp.raleigh.ibm.com:80/A/CpuAndSleepBound,
statusCode: 503, reason: no servers in cluster xd61/DC1 are running web module xd61/A/microwebapp.war
webModule=xd61/A/microwebapp.war
cluster=xd61/DC1
server=xd61/draco03/DC1_draco03
state=STOPPED
weight=0
reachable=true
server=xd61/draco04/DC1_draco04
state=STOPPED
weight=0
reachable=true
server=xd61/draco05/DC1_draco05
state=STOPPED
weight=0
reachable=true
server=xd61/draco06/DC1_draco06
state=STOPPED
weight=0
reachable=true
server=xd61/draco07/DC1_draco07
state=STOPPED
weight=0
reachable=true
```

PlacementControllerProcs.jacl script

You can use the `PlacementControllerProcs.jacl` script to modify the application placement runtime configuration.

Purpose

Depending on your administrative role, you are allowed specific privileges when configuring the autonomic managers. To use this script to configure the autonomic managers, you must have administrator privileges.

Use the **PlacementControllerProcs.jacl** script to modify an entire series of application placement controller configuration options. You can complete the following operations on the runtime configuration.

- Enable or disable the application placement controller
- Make the specified node managed or unmanaged by the application placement controller
- Enable or disable a dynamic cluster
- Retrieve or set the values of additional settings

This script modifies only the runtime configuration of the application placement controller. To make persistent changes to the configuration, use the **APCconfig.jacl** script.

trns: The WebSphere Virtual Enterprise command that equates to **APCconfig.jacl** is **xd_APCconfig.jacl**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **xd_APCconfig.jacl** command, which operates the same as the **APCconfig.jacl** command.

Location

The **PlacementControllerProcs.jacl** script is in the *app_server_root/bin* directory.

Usage

To run the **PlacementControllerProcs.jacl** script with the **wsadmin** utility, use the following command.

```
wsadmin -profile PlacementControllerProcs.jacl -c "insert_proc_parameters"
```

In place of the *insert_proc_parameters* variable, use the name of the procedure and the proper variable values to complete your changes. You must enclose the procedures in quotation marks ("). You might have to modify the **wsadmin** command to **wsadmin.sh** or **wsadmin.bat**, depending on your operating environment.

Procedures

You can use the following command to see a list of all the available procedures.

```
wsadmin -profile PlacementControllerProcs.jacl -c "help"
```

The following procedures are available.

anyFailedServerOperations

Displays a list of failed starts by servers that the application placement controller tried to start.

closeLockMultiAPCGate removeAPCID

Closes and locks a gate for a multiple application placement controller setup. The APCGate mechanism can optionally be used to ensure that multiple placement controllers in different Intelligent Management systems do not run their decision-making cycles at the same time. Enabling the APCGate mechanism is important when multiple separate Intelligent Management cells are sharing the same physical hardware.

deferFailedServerOperations

Disables the enablement of maintenance mode on any servers that failed to start by the application placement controller.

disable

Disables the application placement controller.

disableDynamicCluster dynamic_cluster_name

Sets the specified dynamic cluster to run manually.

disableNode node_name

Makes the specified node unmanaged by the application placement controller.

enable

Enables the application placement controller.

enableDynamicCluster dynamic_cluster_name

Sets the specified dynamic cluster to run automatically.

enableNode node_name

Enables the application placement controller for the specified node.

findBestLocation cell_name node_name server_name

Returns one of three values that indicates an action to take when you attempt to stop a server that is a member of a dynamic cluster. This procedure returns one of the following values:

- null: You can stop the server that was passed into the procedure without problems.
- server name: You should not stop the server name that you passed into the procedure.
- alternate server name: If the name of another server is returned, you should stop that server first.

getApprovalTimeout

When running in supervised mode, returns the amount of time to wait for an administrator to approve a task before the task times out.

getMinTimeBetweenPlacementChange

Returns the minimum time between two consecutive placement changes.

getNodeName

Displays the node on which the application placement controller is running.

getServerOperationTimeout

Returns the amount of time after which the start and stop operation that is performed on the server is considered a failure, if not completed.

handleFailedServerOperations

Places any servers into maintenance mode that did not start when the application placement controller tried to start the server.

isNodeInUse node_name

Checks if a node is still in use after it is in maintenance mode.

isPrimary

Returns if this node is the primary node where the application placement controller is running.

isEnabled

Returns if the application placement controller is enabled.

openMultiAPCGate remoteAPCID

Opens the gate for a multiple application placement controller setup.

recomputePlacement

Triggers the application placement controller to compute its optimization and perform any necessary placement changes.

setMinTimeBetweenPlacementChange time

Sets the minimum time, in minutes, between two consecutive placement changes.

setServerOperationTimeout timeout

Sets the timeout, in minutes, after which the start or stop operation that is performed on the server is considered a failure, if not completed.

setApprovalTimeout approval_timeout

Sets the amount of time, in minutes, to wait for the administrator approval when operating in supervised mode before considering the task denied.

unlockMultiAPCGate remoteAPCID

Unlocks the gate for a multiple application placement controller setup.

enableElasticity

Enables elasticity mode in the application placement controller. Valid arguments are true or false.

isElasticityModeEnabled

Specifies whether elasticity mode is enabled. The return value is either true or false.

elasticityMode

Sets the runtime tasks for application placement elasticity as supervised or automatic. Valid arguments are 0 for automatic or 1 for supervised.

getElasticityMode

Specifies if application placement elasticity is set to supervised or automatic. The return value is either 0 or 1.

elasticityModeTimeout

Specifies the amount of time, in minutes, that an elasticity action is allowed to complete before it is considered a failure. For example, if you want to set the operation timeout value to 30 minutes, specify the procedure as `elasticityModeTimeout 30 0`.

getElasticityTimeoutMins

Returns the amount of time, in minutes, that an elasticity action is allowed to complete before it is considered a failure.

pluginMerge script

You can use the `pluginMerge` script to merge the `plugin-cfg.xml` files from two or more unbridged cells in order to allow an IBM HTTP server plug-in to route to all cells according to the defined mode.

Purpose

The `pluginMerge.bat | .sh` script combines the `plugin-cfg.xml` files from two or more unbridged cells so that an IBM HTTP server plug-in can load-balance across the cells, fail over to defined (backup) cells, or route to all cells, according to precedence.

Location

The `pluginMerge` script is in the `app_server_root/bin` directory. Before running this script, ensure that you have the environment variable configured to point to the directory of your WebSphere installation.

Usage

The default script usage follows:

```
pluginMerge.sh [MODE] plugin-cfg1.xml plugin-cfg2.xml [...] outputfile.xml
```

To display help, run the script without passing any arguments.

Modes

- l Load-balance merge. Merged output allows for load-balancing across the cells.
- f Failover merge. The merged output allows for failover to backup servers. The sequence of the input `plugin-cfg.xml` files determines which servers are considered primary or backup for a specific URI. If a shared URI is discovered, the primary server is the one corresponding to the URI in the leftmost matched `plugin-cfg.xml` file. All other servers are listed as backup. If a URI is unique to a specific `plugin-cfg.xml` file, the corresponding servers are marked as primary for the unique URI.
- p Precedence merge. The precedence ranking is determined by the sequence of the input `plugin-cfg.xml` files. The merged output limits routing requests for shared URIs to only the servers contained in the leftmost matched input `plugin-cfg.xml` file. If a URI is unique across the input `plugin-cfg.xml` files, then the servers corresponding to that URI are the ones routed to.

Note: A URI is considered to be shared between two unbridged cells if the URI and the corresponding virtual host definition are identical between `plugin-cfg.xml` files.

In the following example, the merged output contained in `plugin-cfg-merged.xml` allows an IBM HTTP server plug-in to load balance requests between all servers contained in the input `plugin-cfg.xml` files (`plugin-cfg-cell11.xml`, `plugin-cfg-cell12.xml`, `plugin-cfg-cell13.xml`):


```
app_server_root/bin/pluginMerge.sh -l /tmp/plugin-cfg-cell11.xml /tmp/plugin-cfg-cell12.xml  
/tmp/plugin-cfg-cell13.xml /tmp/plugin-cfg-merged.xml
```

serverQuiesce.py script

You can use the `serverQuiesce.py` script to decrease workload to a Session Initiation Protocol (SIP) container over an interval, and optionally stop the server. You can also cancel the quiesce during a set time interval if certain conditions are met.

Purpose

With this script, you can gradually decrease workload to a SIP container over an interval. You can also optionally stop the server.

Restriction:  Intelligent Management does not support SIP features on the z/OS operating system.

Location

The `serverQuiesce.py` script is located in the `app_server_root/bin` directory.

Usage

Run the following command.

```
./wsadmin.sh -f serverQuiesce.py <operation> [options]
```

Replace the `<operation>` and `[options]` variables with the appropriate operation and options for your task.

Operations

Use the following operations with the `serverQuiesce.py` script:

quiesceServer

Gradually decreases the workload to a SIP container.

Table 42. `quiesceServer` options

Option	Description
<code>nodeName</code>	Specifies the name of the node.
<code>serverName</code>	Specifies the name of the server.
<code>phaseOneInterval</code>	Specifies the time in milliseconds for the interval.
<code>performPhaseTwo</code>	Specifies a boolean that indicates whether to stop the server.

cancelQuiesce

Cancels the quiesce during the time interval that you set. To successfully cancel the quiesce, ensure that the time interval has not expired and the optional server stop occurs at the end of the interval.

Table 43. *cancelQuiesce* options

Option	Description
nodeName	Specifies the name of the node.
serverName	Specifies the name of the server.

Example

```
./wsadmin.sh -f serverQuiesce.py quiesceServer node1 server1 300000 false
```

servicepolicy.py script

You can use the **servicepolicy.py** script to perform operations on service policies from the command line, such as creating service policies, removing service policies, and editing transaction classes.

Purpose

You can complete the following actions with the **servicepolicy.py** script.

- Create a service policy
- Remove a service policy
- Create a transaction class
- Remove a transaction class

To create, modify, and remove service policies and transaction classes, you must have configurator or administrator administrative privileges.

Location

The **servicepolicy.py** script is located in the *app_server_root/bin* directory.

Usage

The script usage for general help follows:

```
./wsadmin.sh -lang jython -f servicepolicy.py
```

The script usage for operation-specific help follows:

```
./wsadmin.sh -lang jython -f servicepolicy.py operation --help
```

Operations

createServicePolicy

Creates a service policy with the specified options. You must create and associate transaction classes separately.

- **--spname**: Specifies a name for the service policy that is unique in the cell.
- **--spgt**: Specifies an integer that represents one of the following service policy goal types:
 - 0: discretionary
 - 1: average response time
 - 2: percentile response time
 - 4: completion time
- **--spgv**: Specifies a service policy goal value for non-discretionary goals. This value is assumed to be in milliseconds if you do not specify the units.
- **--spgvu**: Specifies an integer that represents a service policy goal value for non-discretionary goals. This value is assumed to be in milliseconds if you do not specify the units.
 - 0: milliseconds

- 1: seconds
- 2 : minutes
- **--sppgv**: Specifies an integer that represents a percentile value for a service policy with percentile response time goal between 1 and 100.
- **--spi**: Specifies an integer that represents one of the following service policy goal types:
 - 1: highest
 - 2: higher
 - 3: high
 - 4: medium
 - 5: low
 - 6: lower
 - 7: lowest
- **--spd**: Specifies a service policy description.

removeServicePolicy

Deletes an existing service policy with the specified option.

- **--spname**: Specifies the unique name for the service policy that you want to remove.

createTransactionClass

Creates a transaction class with the specified options.

- **--spname**: Specifies a name for the service policy that is unique in the cell.
- **--tcname**: Specifies a name for the transaction class that you want to create that is unique in the cell.
- **--tcd**: Specifies a transaction class description.

removeTransactionClass

Removes a transaction class with the specified option. All Uniform Resource Identifiers (URIs) in the transaction class are no longer associated with the parent service policy. If a request comes in for these URIs and they are not associated with a new service policy and transaction class, they are classified to the default service policy with a discretionary goal.

- **--tcname**: Specifies the cell-unique name for the transaction class that you want to remove.
-

Example

Create a service policy:

```
./wsadmin.sh -lang jython -f servicepolicy.py createServicePolicy --spname
Platinum --splt 2 --spltv 3000 --spltvu 0 --sppgv 80 --spi 5
```

Remove an existing service policy:

```
./wsadmin.sh -lang jython -f servicepolicy.py removeServicePolicy --spname Bronze
```

Create a new transaction class:

```
./wsadmin.sh -lang jython -f servicepolicy.py createTransactionClass --spname
Platinum --tcname PlatinumWorkload --tcd 'my platinum workload'
```

Remove an existing transaction class:

```
./wsadmin.sh -lang jython -f servicepolicy.py removeTransactionClass --tcname
PlatinumWorkload
```

unlinkCells script

You can use **unlinkCells** script to disable the overlay communication between multiple cells.

Purpose

Use the **unlinkCells** script to disable communication between Intelligent Management cells that were previously linked by using the **linkCells.shl.bat** script. After you run the **unlinkCells** script, the on demand router (ODR) no longer routes work requests to the unlinked cell.

Location

The **unlinkCells** script is located in the *app_server_root/bin* directory.

Usage

Run the **unlinkCells** script from the center cell to unlink the center cell from a point cell.

```
./unlinkCells.sh center_cell_deployment_manager:center_cell_soap_port:user_id:password point_cell_deployment_manager:point_cell_soap_port:user_id:password
```

Example

Consider a scenario in which there are two security-enable cells, **center** and **point1**, that were previously linked together by using the **linkCells.shl.bat** script. For the center cell, the host name of the deployment manager is **centerHost**, the SOAP port is **8879**, the user name is **centerUID**, and the password is **centerPWD**. For the point cell, the host name of the deployment manager is **point1Host**, the SOAP port is **8880**, the user name is **point1UID**, and the password is **point1PWD**. The following example illustrates how to unlink the center and point1 cells.

```
./unlinkCells.sh centerHost:8879:centerUID:centerPWD point1Host:8880:point1UID:point1PWD
```

useBBSON.py script

Intelligent Management uses the bulletin board service overly network (BBSON) by default, which alleviates the dependency on the WebSphere Application Server Network Deployment (ND) high availability manager and its required core group configuration. Run **useBBSON.py** to disable the usage of BBSON, enabling the ND high availability manager instead. You can also use the script to enable BBSON if it was previously disabled.

Purpose

Use the **useBBSON.py** script to disable the usage of BBSON, enabling the ND high availability manager instead. You can also use the script to enable BBSON if it was previously disabled. For restrictions and considerations, see the BBSON bulletin board.

trns: The Intelligent Management command that equates to **useBBSON.py** is **useWVEBB.py**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **useWVEBB.py** command, which operates the same as the **useBBSON.py** command.

Location

The **useBBSON.py** script is located in the *app_server_root/bin* directory.

Usage

To enable BBSON, run the following command:

```
./wsadmin.sh -lang jython -f useBBSON.py
```

To disable BBSON, and to use the ND high availability manager instead, run the following command:

```
./wsadmin.sh -lang jython -f useBBSON.py -useHAMBB
```

Note: After running the **useBBSON.py** script, synchronize your change, stop and start the cell again.

wve_encodePassword script

You can use the **wve_encodePassword** script to encode a password value that you enter in the administrative console. By encoding your password, you can prevent the password from being shared.

Purpose

You can use the **wve_encodePassword** script to encode the passwords that you enter as custom property values into the administrative console.

Location

The **wve_encodePassword** script is located in the `app_server_root/bin` directory.

Usage

The default script usage follows:

```
./wve_encodePassword.sh
```

Parameters

Use the following parameter with the **wve_encodePassword** script:

password

Specifies the password that you want to encode.

```
./wve_encodePassword.sh password01
```

Result

The script returns an encoded password that you can copy and use for your custom property value.

workclassoperations.py script

You can use the **workclassoperations.py** script to create, update, and manage work classes.

Purpose

You can complete the following actions with the **workclassoperations.py** script:

- Create and update work classes for enterprise applications and generic server clusters.
- List the rules in work classes.
- List the members of work classes.
- Delete work classes for the various policies and communication protocols.
- Set default actions.
- Modify default actions.
- Fetch default actions.
- Add rules.
- Delete rules.
- Delete members.

Location

The **workclassoperations.py** script is in the `app_server_root/bin` directory.

Usage

The script usage for general help follows:

```
./wsadmin.sh -lang jython -f workclassoperations.py
```

The script usage for operation-specific help follows:

```
./wsadmin.sh -lang jython -f workclassoperations.py operation --help
```

Depending on where you are issuing the command, you might have to specify the path to the **workclassoperation.py** script. For example, you create a profile named Dmgr01 located in the `c:\app_server_root\profiles\Dmgr01\bin` directory.

```
./wsadmin.sh -lang jython -f ../../bin/workclassoperations.py
```

Note that if the deployment manager is not started, you must include the additional parameter **-conntype NONE**:

```
./wsadmin.sh -lang jython -conntype NONE workclassoperations.py
```

Operations

listWorkClasses

Lists work classes by type. The format of listing is *workclassname type appname/odname deploymentname*

- **--type**: Specifies the type of work class to list:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
- **--appname**: Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
- **--odname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.

createWorkClass

Creates a work class. The new service policy does not contain any application modules or classification rules. You must create and associate application modules and classification rules separately.

- **--type**: Specifies the type of work class to list:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
- **--appname**: Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
- **--odname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
- **--nodename**: Specifies the name of the node. Required if you specify the **--type** parameter as GSP or GRP.
- **--wcname**: Specifies the unique name for the work class within the specified type.
- **--protocol**: Specifies the communications protocol that transmits the request. Currently supported protocols are HTTP, IIOP, SOAP, and JMS.

- **--wcaction:** Specifies the default action to take when a request is matched to a member for the specified work class.
- **--module:** Specifies the application module to associate members. Required if you specify the **--appname** and **--members** parameters.
- **--members:** Specifies the protocol-specific pattern:
 - HTTP: /test1?/test2/*
 - IIOP: ejbName:ejbMethod?ejbName:ejbMethod
 - SOAP: webService:operationName?webService:operationName
 - JMS: bus:destination?bus:destination
- **--rule:** Specifies a classification rule in the format of "priority?rule?action"
- **--virtualhost:** Specifies the virtual host. Required if you specify the **--type** parameter as GSP or GRP.
- **removeWorkClass:** Removes a work class.
 - **--type:** Specifies the type of work class to remove:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname:** Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odname:** Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname:** Specifies the unique name for the work class within the specified type.
- **addMembers:** Adds members to a work class. The ODR uses the new member to match incoming requests.
 - **--type:** Specifies the type of work class to remove:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname:** Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odname:** Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname:** Specifies the unique name for the work class within the specified type.
 - **--protocol:** Specifies the communications protocol that transmits the request. Currently supported protocols are HTTP, IIOP, SOAP, and JMS.
 - **--module:** Specifies the application module to associate members. Required if you specify the **--appname** and **--members** parameters.
 - **--members:** Specifies the protocol-specific pattern:
 - HTTP: /test1?/test2/*
 - IIOP: ejbName:ejbMethod?ejbName:ejbMethod
 - SOAP: webService:operationName?webService:operationName
 - JMS: bus:destination?bus:destination
 - **--ejbname:** Specifies the name of the EJB. Required only when you specify the **--protocol** parameter as JMS and the **--module** parameter as IIOP.
- **removeMembers:** Removes members from a work class. The member is no longer used by the ODR.

- **--type**: Specifies the type of work class to modify:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
- **--appname**: Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
- **--odname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
- **--wcname**: Specifies the unique name for the work class within the specified type.
- **--protocol**: Specifies the communications protocol that transmits the request. Currently supported protocols are HTTP, IIOP, SOAP, and JMS.
- **--module**: Specifies the application module from which to disassociate members. Required if you specify the **--appname** and **--members** parameters.
- **--members**: Specifies the protocol-specific pattern:
 - HTTP: /test1?/test2/*
 - IIOP: ejbName:ejbMethod?ejbName:ejbMethod
 - SOAP: webService:operationName?webService:operationName
 - JMS: bus:destination?bus:destination
- **ListMembers**: Lists members of a work class.
 - **--type**: Specifies the type of work class to list:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname**: Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--nodename**: Specifies the name of the node. Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname**: Specifies the unique name for the work class within the specified type.
 - **--protocol**: Specifies the communications protocol that transmits the request. Currently supported protocols are HTTP, IIOP, SOAP, and JMS.
- **addRules**: Adds a classification rule that is used by the ODR to a work class. If you issue the command with the like wildcard '%' on UNIX operating systems, replace the escape sequence (\%%) with (\%).
 - **--type**: Specifies the type of work class to modify:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname**: Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.

- **--nodename:** Specifies the name of the node. Required if you specify the **--type** parameter as GSP or GRP.
- **--wcname:** Specifies the unique name for the work class within the specified type.
- **--rule:** Specifies a classification rule in the format of "priority?rule?action"
- **removeRule:** Removes a classification rule from a work class. The rule is no longer used by the ODR.
 - **--type:** Specifies the type of work class to modify:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname:** Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odrname:** Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname:** Specifies the unique name for the work class within the specified type.
 - **--expression:** Specifies the rule expression.
 - **--priority:** Specifies the priority of the rule to match. The rule with the lowest priority is matched first.
- **ListRules:** Lists classification rules.
 - **--type:** Specifies the type of work class to modify:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname:** Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odrname:** Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname:** Specifies the unique name for the work class within the specified type.
- **modifyDefaultAction:** Edits the default action for a work class.
 - **--type:** Specifies the type of work class to modify:
 - ASP: application service policy
 - ARP: application routing policy
 - GSP: generic server service policy
 - GRP: generic server routing policy
 - **--appname:** Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
 - **--odrname:** Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname:** Specifies the unique name for the work class within the specified type.
 - **--wcaction:** Specifies the default action to take when a request is matched to a member for the specified work class.
- **getDefaultAction:** Specifies the default action for a work class.
 - **--type:** Specifies the type of work class to list:
 - ASP: application service policy
 - ARP: application routing policy

- GSP: generic server service policy
- GRP: generic server routing policy
- **--appname**: Specifies the name of the application. Required if you specify the **--type** parameter as ASP or ARP.
- **--odrname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
- **--wcname**: Specifies the unique name for the work class within the specified type.
- **modifyVirtualHost**: Edits the virtual host for a work class.
 - **--type**: Specifies the type of work class to modify. The valid types are GSP and GRP.
 - **--odrname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname**: Specifies the unique name for the work class within the specified type.
 - **--virtualhost**: Specifies the virtual host. Required if you specify the **--type** parameter as GSP or GRP.
 -
- **getVirtualHost**: Lists the virtual host for a work class.
 - **--type**: Specifies the type of work class to list. The valid types are GSP and GRP.
 - **--odrname**: Specifies the name of the on demand router (ODR). Required if you specify the **--type** parameter as GSP or GRP.
 - **--wcname**: Specifies the unique name for the work class within the specified type.

Example

List all work classes:

```
./wsadmin.sh -lang jython -f workclassoperations.py listWorkClasses
```

List all work classes for application Trade:

```
./wsadmin.sh -lang jython -f workclassoperations.py listWorkClasses
--appname Trade
```

Create a HTTP application routing policy work class for application Trade with a default action of reject and reject code of 404:

```
./wsadmin.sh -lang jython -f workclassoperations.py createWorkClass
--type ARP --wcname CustomWorkClass --protocol HTTP --waction "reject?404" --appname Trade
--module myModule.war --members "/test1?/test2"
```

List all routing policy work classes for application Trade:

```
./wsadmin.sh -lang jython -f workclassoperations.py listWorkClasses
--type ARP --appname Trade
```

View the default action of a HTTP application routing policy work class for application Trade:

```
./wsadmin.sh -lang jython -f workclassoperations.py getDefaultAction
--type ARP --wcname CustomWorkClass --appname Trade
```

Modify the default action to Default_TC in a HTTP application service policy work class for application Trade-edition1.0:

```
./wsadmin.sh -lang jython -f workclassoperations.py modifyDefaultAction --type ASP --wcname Default_HTTP_WC
--waction Default_TC --appname Trade-edition1.0
```

Map the default HTTP work class of edition 1.2 of the DayTrader application to the default transaction class for the service policy GOLD:


```

import java.lang.System as System

wasinstallroot=System.getProperty("was.install.root")
execfile(wasinstallroot+"/bin/IMPPYModules.py")
import java.lang.System as System

wasinstallroot=System.getProperty("was.install.root")
execfile(wasinstallroot+"/bin/IMPPYModules.py")

#application with name DayTrader installed with edition 1.2
appName="DayTrader-edition1.2"
#Default_TC_GOLD is the default transaction class mapped to service policy GOLD
modifyAppSLADefaultAction("Default_HTTP_WC", appName, tcName, 0)

AdminConfig.save()

```

trns: The WebSphere Virtual Enterprise command that equates to **IMPPYModules.py** is **XDPYModules.py**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **XDPYModules.py** command, which operates the same as the **IMPPYModules.py** command.

Add a classification rule to the application service policy work class for application Trade:

```

./wsadmin.sh -lang jython -f workclassoperations.py addRules --wcname CustomWorkClass
--appname Trade --type ARP --rule "I?clienthost='localhost\' and serverhost like '\\\%.ibm.com\'?permit?Trade"

```

Remove a classification rule by priority from the application service policy work class for application Trade:

```

./wsadmin.sh -lang jython -f workclassoperations.py removeRule --wcname
CustomWorkClass --appname Trade --type ARP --priority 1

```

Remove a HTTP application routing policy work class for application Trade:

```

./wsadmin.sh -lang jython -f workclassoperations.py removeWorkClass --type ARP
--wcname CustomWorkClass --appname Trade

```

APCconfig.jacl script

Use the **APCconfig.jacl** script to make persistent changes to the application placement controller configuration.

Purpose

You can use the **APCconfig.jacl** script to modify attributes and custom properties of the application placement controller and dynamic clusters.

trns: The WebSphere Virtual Enterprise command that equates to **APCconfig.jacl** is **xd_APCconfig.jacl**. If you are making the transition from WebSphere Virtual Enterprise, you can continue to use the **xd_APCconfig.jacl** command, which operates the same as the **APCconfig.jacl** command.

Location

The **APCconfig.jacl** script is in the *app_server_root/bin* directory.

Usage

To change the configuration, you must have configurator or administrator administrative privileges.

To run the **APCconfig.jacl** script with the **wsadmin** tool, use the following command:

```
wsadmin.sh -profile APCconfig.jacl -c "insert_proc"
```

In place of the *insert_proc* variable, use the name of the procedure and the proper variable values to complete your changes. Enclose the procedures in quotation marks. Modify the **wsadmin** command to

wsadmin.sh or **wsadmin.bat**, depending on your operating system.

Procedures

To receive a list of the valid attributes that you can set on the application placement controller or dynamic cluster, run the following **wsadmin** command:

```
$AdminConfig show [$AdminConfig getid "/DynamicCluster:clustername/"]
```

The following procedures are available:

setAPCAtribute *attribute_value*

Sets the defined attribute and value on the application placement controller. For a list of the valid attributes for the application placement controller, run the following **wsadmin** command:

```
$AdminConfig show [$AdminConfig getid "/AppPlacementController:/"]
```

getAPCAtribute *attribute_name*

Displays the specified application placement controller attribute and value.

setAPCCustomProperty *property value description*

Sets the specified application placement controller custom property name, value, and description. For more information about custom properties that you can set on the application placement controller, read about application placement custom properties . For example, the following command sets the `numVerticalInstances` custom property:

```
wsadmin.sh -profile APCconfig.jacl -c "setAPCCustomProperty numVerticalInstances.myNode 3 verticalstacking"
```

getAPCCustomProperty *custom_property_name*

Gets the value and description for the specified application placement controller custom property. For example, the following command gets the value of the `numVerticalInstances` custom property:

```
wsadmin.sh -profile APCconfig.jacl -c "getAPCCustomProperty numVerticalInstances.myNode"
```

setDynamicClusterAttribute *dynamic_cluster_name attribute_value*

Sets the defined attribute and value for the specified dynamic cluster. For a list of the valid attributes for dynamic clusters, run the following **wsadmin** command:

```
$AdminConfig show [$AdminConfig getid "/DynamicCluster:clustername/"]
```

getDynamicClusterAttribute *dynamic_cluster_name attribute*

Displays the value for the specified dynamic cluster attribute.

setDynamicClusterCustomProperty *dynamic_cluster_name custom_property_name value description*

Sets a custom property on the specified dynamic cluster. For more information about custom properties that you can set on dynamic clusters, read about dynamic cluster custom properties. For example, the following command sets the `HTTPSessionRebalanceOff` custom property:

```
wsadmin.sh -profile APCconfig.jacl -c "setDynamicClusterCustomProperty my_dyncluster HTTPSessionRebalanceOff true disablerebalancing"
```

getDynamicClusterCustomProperty *dynamic_cluster_name property*

Gets the value of the specified custom property for a dynamic cluster. For example, the following command gets the value of the `HTTPSessionRebalanceOff` custom property:

```
wsadmin.sh -profile APCconfig.jacl -c "getDynamicClusterCustomProperty my_dyncluster HTTPSessionRebalanceOff"
```

Administrative roles and privileges

Roles and privileges vary depending on your administrative role and the component.

Administrative roles and privileges

For complete definitions of administrative roles in WebSphere Application Server and how to assign them, read about authorizing access to administrative roles.

Table 44. Administrative roles and privileges. The following table outlines the roles and privileges for your administrative role and component.

GUI	Monitor privileges	Operator privileges	Configurator privileges	Administrator privileges
Dynamic clusters	View the configuration.	View the configuration. Can change the operating mode of the dynamic cluster. Can start/stop cluster members.	Can alter the configuration of the dynamic cluster, including server template for the dynamic cluster. Can create new and delete dynamic clusters. Cannot change the operating mode of the dynamic cluster. Cannot start/stop cluster members.	Has all privileges.
Task management notification configuration	View the information.	View the information.	Can change the configuration.	Has all privileges.
Task management runtime tasks	View the information	Can run tasks.	View information.	Has all privileges.
Runtime operations / runtime topology	View the information	View the information. Can start/stop servers. Can set maintenance mode on nodes.	View the information.	Has all privileges.
Runtime operations / runtime charting	Has all privileges.	Has all privileges.	Has all privileges.	Has all privileges.
Runtime operations / runtime map	Has all privileges.	Has all privileges.	Has all privileges.	Has all privileges.
Runtime operations / job management	View the jobs.	Has all privileges.	View the jobs.	Has all privileges.
Service policies	View the information.	View the information.	Can create and delete service policies and transaction classes.	Has all privileges.
On demand router (ODR) configuration	View the information.	View information. Can start and stop ODRs.	Can edit create, and delete ODRs.	Has all privileges.
ODR configuration / generic server cluster / service and routing policies	View the information.	View the information.	Has all privileges except modifying rules through the rule builder	Has all privileges.
Application routing policies	View the information.	View the information.	Has all privileges.	Has all privileges.
Application edition control center	View the information.	View the information.	Has all privileges.	Has all privileges.
Job scheduler (batch only)	View the information.	View the information.	Has all privileges.	Has all privileges.
Visualization data service	View the information.	View the information.	Can change the configuration.	Has all privileges.
High Availability (HA) deployment manager	View the information.	Has all privileges.	View the information.	Has all privileges.

Table 44. Administrative roles and privileges (continued). The following table outlines the roles and privileges for your administrative role and component.

GUI	Monitor privileges	Operator privileges	Configurator privileges	Administrator privileges
Autonomic manager configuration	View the information.	View the information on the configuration tab. Can change the settings on the runtime tab.	Can change the configuration but not the runtime settings.	Has all privileges.
Health policies	View the information.	View the information.	Has all privileges.	Has all privileges.
Node maintenance mode	View the information.	Can: <ul style="list-style-type: none"> • set and unset maintenance nodes • synchronize nodes • stop nodes 	Can: <ul style="list-style-type: none"> • view the configuration. • add and remove nodes. 	Has all privileges.
Extended repository service	View the information.	View the information.	Has all privileges.	Has all privileges.
Middleware applications	View the information.	View the information.	Has all privileges.	Has all privileges.

Administrator scripting interfaces

This reference information provides examples of how to complete various tasks using the administrative scripting interfaces.

Scripting interfaces

Use the following scripting interfaces to automate tasks that you might normally accomplish with the administrative console.

- “Application edition management administrative tasks”
- “Dynamic cluster administrative tasks” on page 342
- “Default work class administrative tasks” on page 357
- “Health policy administrative tasks” on page 361
- “Maintenance mode administrative tasks” on page 369
- “Middleware application administrative tasks” on page 371
- “Middleware server creation administrative tasks” on page 393
- “Middleware server management administrative tasks” on page 398
- “Middleware server template administrative tasks” on page 404
- “PHP server administrative tasks” on page 409
- “Rules for ODR routing policy administrative tasks” on page 413
- “Rules for ODR service policy administrative tasks” on page 422
- “Ruleset administrative tasks” on page 428
- “Runtime operations user preferences administrative tasks” on page 436

Application edition management administrative tasks

You can use the Jacl or Jython scripting languages to manage application editions. You can also use the AdminApp object to manage applications with specific editions.

Use the following commands to activate, deactivate, perform a rollout on, validate, list, and verify an application edition. You can also clone an existing dynamic cluster to use a different node group, and you can cancel the validation mode on an application edition:

- `activateEdition`
- `deactivateEdition`
- `rolloutEdition`
- `validateEdition`
- `listEditions`
- `isEditionExists`
- `cloneDynamicCluster`
- “cancelValidation” on page 340
- “getEditionState” on page 341

You can perform the following operations using the `AdminApp` object in `wsadmin` to manage applications with editions:

- “Install an application with an edition” on page 341
- “Update an application with an edition” on page 342
- “Edit an application with an edition” on page 342
- “Uninstall an application with an edition” on page 342

For a complete list of operations that you can perform using the `AdminApp` object, read about `Commands` for the `AdminApp` object using `wsadmin` scripting.

activateEdition

The **`activateEdition`** command activates an edition.

Target object

None.

Parameters

- **`-appName`**
Specifies the name of the application. (String)
- **`-edition`**
Specifies the name of the application edition. (String)

Return value

The command returns:

- `true`: success
- `false`: failure

Batch mode example usage

- Using Jacl:
`$AdminTask activateEdition {-appName BeenThere -edition 1.0}`
- Using Jython string:
`AdminTask.activateEdition ('[-appName BeenThere -edition 1.0]')`
- Using Jython list:
`AdminTask.activateEdition (['-appName', 'BeenThere', '-edition', '1.0'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask activateEdition {-interactive}`
- Using Jython string:
`AdminTask.activateEdition ('[-interactive]')`
- Using Jython list:
`AdminTask.activateEdition (['-interactive'])`

deactivateEdition

The **deactivateEdition** command deactivates an edition.

Target object

None.

Parameters

- appName**
Specifies the name of the application. (String)
- edition**
Specifies the name of the application edition. (String)

Return value

The command returns:

- true: success
- false: failure

Batch mode example usage

- Using Jacl:
`$AdminTask deactivateEdition {-appName BeenThere -edition 1.0}`
- Using Jython string:
`AdminTask.deactivateEdition ('[-appName BeenThere -edition 1.0]')`
- Using Jython list:
`AdminTask.deactivateEdition (['-appName', 'BeenThere', '-edition', '1.0'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask deactivateEdition {-interactive}`
- Using Jython string:
`AdminTask.deactivateEdition ('[-interactive]')`
- Using Jython list:
`AdminTask.deactivateEdition (['-interactive'])`

rolloutEdition

The **rolloutEdition** command rolls out an edition and specifies the group size.

Target object

None.

Parameters

-appName

Specifies the name of the application. (String)

-edition

Specifies the value of the custom property. (String)

-params

Specifies configuration values for the rollout. (String)

rollout strategy

- **grouped**: Activates the specified application in place of the current edition of the same application across the cluster on which the application cluster deploys, N servers at a time, as specified by the **groupSize** keyword. The default group size is 1. During rollout, both editions of the application can serve requests. Always set the group number to be at least one less than the cluster size so that at least one cluster member serves the old edition while the rollout of the new edition starts.
- **atomic**: Activates the specified application in place of the current edition of the same application across the cluster on which the application cluster deploys, one half of the cluster at a time, such that only one edition of the application serves requests at any given time.

reset strategy

Specifies how to start an application edition instance during the rollout operation.

- **hard**: Stops or restarts the application server.
- **soft**: Stops or restarts the application instance, while leaving the application server running.

group size for group rollout

Specifies the number of servers to update when **rolloutStrategy** is grouped.

- integer

drainage interval

Specifies the number of seconds to wait before stopping an application edition instance during the rollout such that sessions can complete. The default is 30 seconds.

- integer

Specify the following properties for Session Initiation Protocol (SIP) application rollout:

quiesce strategy

Specifies if cluster members and servers quiesce after all active dialogs and sessions are complete, or if the cluster members and servers quiesce after a specified interval.

DEFAULT: Cluster members and servers quiesce when active dialogs and sessions complete.

INTERVAL: Cluster members and servers quiesce after a specified interval. You must also specify the **quiesceInterval** as an integer value in number of seconds.

Return value

The command returns:

- true: success
- false: failure

Batch mode example usage

- Using Jacl:

```
$AdminTask rolloutEdition {-appName BeenThere -edition 1.0 -params  
"{quiesceStrategy INTERVAL}{quiesceInterval 30}"}
```

Using Jython string:

```
AdminTask.rolloutEdition ('[-appName BeenThere -edition 1.0 -params  
"{quiesceStrategy INTERVAL}{quiesceInterval 30}"']')
```

Note: The following example applies only to Version 6.1.0.5 running with WebSphere Application Server Network Deployment Version 7.0.0.x.

```
AdminTask.rolloutEdition ('[-appName BeenThere -edition 1.0 -params  
[[rolloutStrategy grouped][resetStrategy soft]  
[groupSize 1][drainageInterval 30]]')
```

- Using Jython list:

```
AdminTask.rolloutEdition (['-appName', 'BeenThere', '-edition', '1.0', '-params',  
'{quiesceStrategy INTERVAL}{quiesceInterval 30}'])
```

Note: The following example applies only to Version 6.1.0.5 running with WebSphere Application Server Network Deployment Version 7.0.0.x.

```
AdminTask.rolloutEdition (['-appName', 'BeenThere', '-edition', '1.0', '-params',  
'[[rolloutStrategy grouped][resetStrategy soft]  
[groupSize 1][drainageInterval 30]]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask rolloutEdition {-interactive}
```

Using Jython string:

```
AdminTask.rolloutEdition ('[-interactive]')
```

- Using Jython list:

```
AdminTask.rolloutEdition (['-interactive'])
```

validateEdition

The **validateEdition** command validates an edition.

Target object

None.

Parameters

-appName

Specifies the name of the application. (String)

-edition

Specifies the name of the application edition. (String)

-params

Specifies settings for the cloned cluster created during validation.

Note: For dynamic clusters, the size of the original cluster is used for the cloned cluster.

- Dynamic cluster:

dynClusterMaxSize

Specifies the maximum number of dynamic cluster instances that will start.

dynClusterMinSize

Specifies the minimum number of dynamic cluster instances that will start.

- Static cluster:

staticClusterSize

Specifies the size of the static cluster. The value of **staticClusterSize** cannot exceed the size of the static cluster that is being cloned.

Return value

The command returns:

- true: success
- false: failure

Batch mode example usage

- Using Jacl:

```
$AdminTask validateEdition {-appName BeenThere -edition 1.0 -params  
"{dynClusterMaxSize 2}{dynClusterMinSize 1}"}
```

Using Jython string:

```
AdminTask.validateEdition (['-appName BeenThere -edition 1.0 -params  
"{dynClusterMaxSize 2}{dynClusterMinSize 1}"]')
```

Note: The following example applies only to Version 6.1.0.5 running with WebSphere Application Server Network Deployment Version 7.0.0.x.

```
AdminTask.validateEdition (['-appName BeenThere -edition 1.0 -params  
[[dynClusterMaxSize 2][dynClusterMinSize 1]]']')
```

- Using Jython list:

```
AdminTask.validateEdition (['-appName', 'BeenThere', '-edition', '1.0', '-params',  
'{dynClusterMaxSize 2}{dynClusterMinSize 1}']')
```

Note: The following example applies only to Version 6.1.0.5 running with WebSphere Application Server Network Deployment Version 7.0.0.x.

```
AdminTask.validateEdition (['-appName', 'BeenThere', '-edition', '1.0', '-params',  
'[[dynClusterMaxSize 2][dynClusterMinSize 1]]']')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask validateEdition {-interactive}
```

Using Jython string:

```
AdminTask.validateEdition (['-interactive']')
```

- Using Jython list:

```
AdminTask.validateEdition (['-interactive'])
```

listEditions

The **listEditions** command provides a list of all the installed editions of the application.

Target object

None.

Parameters

-appName

Specifies the name of the application. (String)

Return value

The command returns a string array of all the installed editions of the application.

Note: The return value for base editions is displayed as "".

Batch mode example usage

- Using Jacl:

```
$AdminTask listEditions {-appName HelloWorld}
```

Using Jython string:

```
AdminTask.listEditions ('[-appName HelloWorld]')
```

- Using Jython list:

```
AdminTask.listEditions (['-appName', 'HelloWorld'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask listEditions {-interactive}
```

Using Jython string:

```
AdminTask.listEditions ('[-interactive]')
```

- Using Jython list:

```
AdminTask.listEditions (['-interactive'])
```

isEditionExists

The **isEditionExists** command verifies that the specified edition exists for the particular application.

Target object

None.

Parameters

-appName

Specifies the name of the application. (String)

-edition

Specifies the name of the application edition. (String)

Return value

The command returns:

- true: success
- false: failure

Batch mode example usage

- Using Jacl:

```
$AdminTask isEditionExists {-appName HelloWorld -edition 2.0}
```

Using Jython string:

```
AdminTask.isEditionExists ('[-appName HelloWorld -edition 2.0]')
```

- Using Jython list:

```
AdminTask.isEditionExists (['-appName', 'HelloWorld', '-edition', '2.0'])
```

Interactive mode example usage

- Using Jacl:
`$AdminTask.isEditionExists {-interactive}`

Using Jython string:

```
AdminTask.isEditionExists ('[-interactive]')
```

- Using Jython list:
`AdminTask.isEditionExists (['-interactive'])`

cloneDynamicCluster

The `cloneDynamicCluster` command clones a dynamic cluster to use a different node group.

Target object

None.

Parameters

-clusterName

Specifies the name of the original dynamic cluster. (String)

-newClusterName

Specifies the name of the cloned dynamic cluster. (String)

-nodeGroupName

Specifies the name of the node group in which the cloned dynamic cluster is located. By default, the name is set to the current node group if this parameter is not specified. (String)

-opMode

Specifies the operational mode of the cloned dynamic cluster. By default, the mode of the original dynamic cluster is used if this parameter is not specified. Set the value to `automatic`, `manual`, or `supervised`. (String)

For more information about operational modes, read about dynamic clusters..

-templateName

Specifies the name of an existing template in the format of `cell_name/node_name/server_name`. The template is used to create the server instances of the cloned cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask.cloneDynamicCluster {-clusterName SourceCluster -newClusterName newCluster -nodeGroupName CloneNodeGroup -opMode supervised}
$AdminTask.cloneDynamicCluster {-clusterName SourceCluster -newClusterName newCluster -nodeGroupName CloneNodeGroup -opMode supervised -templateName xdCell/xdNode/SourceCluster_member1}
```

Using Jython string:

```
AdminTask.cloneDynamicCluster (['-clusterName SourceCluster -newClusterName newCluster -nodeGroupName CloneNodeGroup -opMode supervised'])
```

```
AdminTask.cloneDynamicCluster (['-clusterName SourceCluster -newClusterName newCluster -nodeGroupName CloneNodeGroup -opMode supervised -templateName xdCell/xdNode/SourceCluster_member1'])
```

- Using Jython list:

```
AdminTask.cloneDynamicCluster (['-clusterName', 'SourceCluster', '-newClusterName', 'newCluster', '-nodeGroupName', 'CloneNodeGroup', '-opMode', 'supervised'])
```

```
AdminTask.cloneDynamicCluster (['-clusterName', 'SourceCluster', '-newClusterName', 'newCluster', '-nodeGroupName', 'CloneNodeGroup', '-opMode', 'supervised', '-templateName', 'xdCell/xdNode/SourceCluster_member1'])
```

Interactive mode example usage

- Using Jacl:
`$AdminTask cloneDynamicCluster {-interactive}`

Using Jython string:

```
AdminTask.cloneDynamicCluster ('[-interactive]')
```

- Using Jython list:
`AdminTask.cloneDynamicCluster (['-interactive'])`

cancelValidation

The `cancelValidation` command cancels the validation mode on an application edition.

The `getEditionState` command returns the state of the application edition. The state of an existing edition can be either `ACTIVE`, `INACTIVE` or `VALIDATE`. For a non-existent application or edition the method returns null.

Target object

None.

Parameters

- appName**
Specifies the name of the application. (String)
- edition**
Specifies the name of the application edition. (String)

Return value

The command returns:

- **true**: The cancellation of the validation succeeded.
- **false**: The cancelation of the validation failed.

Batch mode example usage

- Using Jacl:
`$AdminTask cancelValidation {-appName BeenThere -edition 1.0}`
- Using Jython string:
`AdminTask.cancelValidation ('[-appName BeenThere -edition 1.0]')`
- Using Jython list:
`AdminTask.cancelValidation (['-appName', 'BeenThere', '-edition', '1.0'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask cancelValidation {-interactive}`
- Using Jython string:
`AdminTask.cancelValidation ('[-interactive]')`
- Using Jython list:
`AdminTask.cancelValidation (['-interactive'])`

getEditionState

The **getEditionState** command returns the state of the application edition. The state of an existing edition can be either ACTIVE, INACTIVE or VALIDATE. For a non-existent application or edition, the method returns null.

Target object

None.

Parameters

-appName

Specifies the name of the application. (String)

-edition

Specifies the name of the application edition. (String)

Return value

The command returns:

- **ACTIVE**
- **INACTIVE**
- **VALIDATE**

Batch mode example usage

- Using Jacl:

```
$AdminTask getEditionState {-appName BeenThere -edition 1.0}
```

Using Jython string:

```
AdminTask.getEditionState ('[-appName BeenThere -edition 1.0]')
```

- Using Jython list:

```
AdminTask.getEditionState (['-appName', 'BeenThere', '-edition', '1.0'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask getEditionState {-interactive}
```

Using Jython string:

```
AdminTask.getEditionState ('[-interactive]')
```

- Using Jython list:

```
AdminTask.getEditionState (['-interactive'])
```

Install an application with an edition

Use the AdminApp **install** command to install an application with an edition. You must pass the **-edition** parameter to specify the edition identifier.

You can use the following example in wsadmin to install the application BeenThere with edition 1.0.

- Using Jacl:

```
$AdminApp install /tmp/BeenThere1.0.ear {-appname BeenThere -edition 1.0 -nopreCompileJSPs -distributeApp  
-noseMetaDataFromBinary -nodeployejb -createMBeansForResources -noreloadEnabled -nodeployws -validateinstall  
warn -noprocessEmbeddedConfig -filepermission *.*.dll=755#.*\.*.so=755#.*\.*.a=755#.*\.*.sl=755 -noallowDispatchRemoteInclude  
-noallowServiceRemoteInclude -MapModulesToServers {"BeenThere WAR" BeenThere.war,WEB-INF/web.xml WebSphere:  
cell=TestCell,cluster=TestClusterB} {"BeenThere EJB" BeenThere.jar,META-INF/ejb-jar.xml WebSphere:  
cell=TestCell,cluster=TestClusterB}}
```

- Using Jython:

```
AdminApp.install("/tmp/BeenThere1.0.ear", ['-appName BeenThere -edition 1.0 -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -processEmbeddedConfig -filepermission .*.dll=755#.*.so=755#.*.a=755#.*.sl=755 -buildVersion Unknown
-noallowDispatchRemoteInclude -noallowServiceRemoteInclude -MapModulesToServers [{"BeenThere WAR" BeenThere.war,WEB-INF/web.xml
WebSphere:cell=TestCell,cluster=TestClusterB} [{"BeenThere EJB" BeenThere.jar,META-INF/ejb-jar.xml WebSphere:
cell=TestCell,cluster=TestClusterB}]]')
```

Update an application with an edition

Use the AdminApp **update** command to update an application with an edition. Pass the full application name, which is the name of the application and the edition, to specify the edition.

You can use the following example in wsadmin to update edition 1.0 of the application BeenThere with the BeenThereE1-update.ear file.

- Using Jacl:

```
$AdminApp update BeenThere-edition1.0 app {-operation update -contents /tmp/BeenThereE1-update.ear -nopreCompileJSPs
-distributeApp -nouseMetaDataFromBinary -nodeployejb -createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
warn -noprocessEmbeddedConfig -filepermission .*.dll=755#.*.so=755#.*.a=755#.*.sl=755 -noallowDispatchRemoteInclude
-noallowServiceRemoteInclude}
```

- Using Jython:

```
AdminApp.update('BeenThere-edition1.0', 'app', ['-operation update -contents /tmp/BeenThereE1-update.ear -nopreCompileJSPs
-distributeApp -nouseMetaDataFromBinary -nodeployejb -createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
warn -noprocessEmbeddedConfig -filepermission .*.dll=755#.*.so=755#.*.a=755#.*.sl=755 -noallowDispatchRemoteInclude
-noallowServiceRemoteInclude]')
```

Edit an application with an edition

Use the AdminApp **edit** command to edit an application with an edition. Pass the full application name, which is the name of the application and the edition, to specify the edition.

You can use the following example in wsadmin to edit edition 1.0 of the application BeenThere by changing the context root of the Web module to /beenthere.

- Using Jacl:

```
$AdminApp edit BeenThere-edition1.0 {-CtxRootForWebMod [{"BeenThere WAR" BeenThere.war,WEB-INF/web.xml /beenthere}]}
```

- Using Jython:

```
AdminApp.edit('BeenThere-edition1.0', ['-CtxRootForWebMod [{"BeenThere WAR" BeenThere.war,WEB-INF/web.xml /beenthere}]]')
```

Uninstall an application with an edition

Use the AdminApp **uninstall** command to uninstall an application with an edition. Pass the full application name, which is the name of the application and the edition, to specify the edition.

You can use the following example in wsadmin to uninstall edition 1.0 of the application BeenThere.

- Using Jacl:

```
$AdminApp uninstall BeenThere-edition1.0
```

- Using Jython:

```
AdminApp.uninstall('BeenThere-edition1.0')
```

Dynamic cluster administrative tasks

Use the dynamic cluster commands to view or edit the dynamic clusters, without using the administrative console.

Use the following commands to create dynamic clusters, add or remove servers from clusters, list dynamic clusters, or modify dynamic clusters:

- “createDynamicCluster” on page 343
- “createPHPDynamicCluster” on page 345

- “createDynamicClusterFromStaticCluster” on page 346
- “createDynamicClusterFromForeignServers” on page 346
- “addForeignServersToDynamicCluster” on page 347
- “removeForeignServersFromDynamicCluster” on page 348
- “deleteDynamicCluster” on page 348
- “getDynamicClusterMembers” on page 348
- “getDynamicClusterOperationalMode” on page 349
- “getDynamicClusterMembershipPolicy” on page 349
- “getDynamicClusterServerType” on page 350
- “getDynamicClusterMinInstances” on page 350
- “getDynamicClusterMaxInstances” on page 350
- “getDynamicClusterVerticalInstances” on page 351
- “getDynamicClusterIsolationProperties” on page 351
- “listDynamicClusters” on page 352
- “listDynamicClusterIsolationGroups” on page 352
- “listDynamicClusterIsolationGroupMembers” on page 352
- “modifyDynamicClusterIsolationProperties” on page 353
- “setDynamicClusterOperationalMode” on page 353
- “setDynamicClusterMembershipPolicy” on page 354
- “setDynamicClusterMinInstances” on page 354
- “setDynamicClusterMaxInstances” on page 355
- “setDynamicClusterVerticalInstances” on page 355
- “testDynamicClusterMembershipPolicy” on page 356
- “createNonWASDynamicCluster ” on page 356
- “Edit the dynamic cluster server template” on page 357

createDynamicCluster

The `createDynamicCluster` command creates a new dynamic cluster.

Required parameters

- **-membershipPolicy**: Specifies the membership policy. (String, required)

Optional parameters

- **-dynamicClusterProperties**: Specifies the dynamic cluster properties. (String, optional) You can specify the following options:
 - **operationalMode**: Specifies the operational mode. The valid values are: manual, supervised, and automatic.
 - **minInstances**: Specifies the minimum number of cluster instances as an integer.
 - **serverInactivityTime**: Specifies the time to wait before stopping instances, in minutes. This value is valid only when the minInstances parameter value is set to 0.
 - **maxInstances**: Specifies the maximum number of cluster instances as an integer. If you do not want a limit to the number of cluster instances that can start, specify -1.
 - **numVerticalInstances**: Specifies the number of instances that can start on the same node by creating vertical instances. Specify an integer value.
- **-clusterProperties**: Specifies the cluster properties. (String, optional) You can specify the following options:

- **preferLocal**: Specifies whether enterprise bean requests are routed to the node on which the client resides when possible. The valid values are true or false.
- **createDomain**: Specifies that a replication domain is created for this cluster. Replication domains can be created only for WebSphere Application Server application servers. The valid values are true or false.
- **coreGroup**: Specifies the name of the core group for the dynamic cluster. You only need to specify the core group for WebSphere Application Server application servers. The value can be DefaultCoreGroup or the core group name.
- **templateName**: Specifies the name of the server template. The valid input includes the following values:
 - **For WebSphere Application Server dynamic clusters:** default, *cell_name/node_name/was_server_name*

Note: The defaultXD and defaultXDZOS server templates that are used when creating a dynamic cluster are deprecated. Use the default server template.

- **z/OS clusterShortName**: For z/OS platforms, specifies the short name of the cluster. After the dynamic cluster has been created, you can use the AdminTask changeClusterShortName command to update this field.
- **z/OS serverSpecificShortNames**: For z/OS platforms, specifies the specific short name of cluster member in a comma-separated list format, for example: SSN1,SSN2. Use commas to separate multiple short names. If you do not provide enough short names to be used for all of the cluster members, then the remaining cluster members are assigned generated generic short names, such as BBOS001, BBOS002, and so on. After the dynamic cluster has been created, you can update this list with the serverSpecificShortNames custom property. This property is set on the dynamic cluster. If you want to change this property with scripting after the dynamic cluster has been created, you can use the AdminTask changeServerSpecificShortName command.

Return value

The command returns javax.management.ObjectName value of the dynamic cluster that you created.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDynamicCluster dcName {-membershipPolicy node_nodegroup='ngName'}
```

or

```
$AdminTask createDynamicCluster dcName {-membershipPolicy "node_nodegroup = 'ngName'" -dynamicClusterProperties
"{operationalMode automatic}{minInstances 0}{maxInstances -1}{numVerticalInstances 1}{serverInactivityTime 1440}"
-clusterProperties "{preferLocal false}{createDomain false}{templateName default}{coreGroup DefaultCoreGroup}
{clusterShortName BB0C001}{serverSpecificShortNames BBOS001, BBOS002}"}
```

- Using Jython string:

```
AdminTask.createDynamicCluster(dcName, '-membershipPolicy "node_nodegroup = \'ngName\'" -dynamicClusterProperties
"{operationalMode automatic}{minInstances 0}{maxInstances -1}{numVerticalInstances 1}{serverInactivityTime 1440}"
-clusterProperties "{preferLocal false}{createDomain false}{templateName default}{coreGroup DefaultCoreGroup}
{clusterShortName BB0C001}{serverSpecificShortNames BBOS001, BBOS002}"')
```

or, if you are running on WebSphere Application Server Version 7.0 or later:

```
AdminTask.createDynamicCluster('dcName', '[-membershipPolicy "node_nodegroup = \'ngName\'" -dynamicClusterProperties
"[operationalMode automatic][minInstances 0][maxInstances -1][numVerticalInstances 1][serverInactivityTime 1440]"
-clusterProperties "[preferLocal false][createDomain false][templateName default][coreGroup DefaultCoreGroup]
[clusterShortName BB0C001][serverSpecificShortNames BBOS001, BBOS002]"')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDynamicCluster {-interactive}
```

- Using Jython string:

```
AdminTask.createDynamicCluster ('[-interactive]')
```


createPHPDynamicCluster

The `createPHPDynamicCluster` command creates a new PHP dynamic cluster.

Required parameters

- **-membershipPolicy**: Specifies the membership policy subexpression. (String, required)

Optional parameters

- **-dynamicClusterProperties**: Specifies the dynamic cluster properties. (String, optional) You can specify the following options:
 - **operationalMode**: Specifies the operational mode. The valid values are: manual, supervised, and automatic.
 - **minInstances**: Specifies the minimum number of cluster instances as an integer.
 - **serverInactivityTime**: Specifies the time to wait before stopping instances, in minutes. This value is valid only when the minInstances parameter value is set to 0.
 - **maxInstances**: Specifies the maximum number of cluster instances as an integer. If you do not want a limit to the number of cluster instances that can start, specify -1.
 - **numVerticalInstances**: Specifies the number of instances that can start on the same node by creating vertical instances. Specify an integer value.
- **-clusterProperties**: Specifies the name of the server template. The valid input includes the **templateName** value:
 - **For PHP dynamic clusters**: APACHE13_PHP4, APACHE13_PHP5, APACHE20_PHP4, APACHE20_PHP5, APACHE22_PHP4, APACHE22_PHP5, cellName/nodeName/php_serverName.

Return value

The command returns `javax.management.ObjectName` value of the dynamic cluster that you created.

Batch mode example usage

- Using Jacl:

```
$AdminTask createPHPDynamicCluster dcName {-membershipPolicy "node_property$APACHE_2_0 IS NOT NULL AND node_property$PHP_5 IS NOT NULL"}
```

or

```
$AdminTask createPHPDynamicCluster dcName {-membershipPolicy "node_property$APACHE_2_0 IS NOT NULL AND node_property$PHP_5 IS NOT NULL" -dynamicClusterProperties "{{operationalMode automatic}{minInstances 1}{maxInstances -1}{numVerticalInstances 1}}" -clusterProperties "{{templateName APACHE20_PHP5}}"}'
```

- Using Jython string:

```
AdminTask.createPHPDynamicCluster(dcName,'[-membershipPolicy "node_property$APACHE_2_0 IS NOT NULL AND node_property$PHP_5 IS NOT NULL" -dynamicClusterProperties "{{operationalMode automatic}{minInstances 1}{maxInstances -1}{numVerticalInstances 1}}" -clusterProperties "{{templateName APACHE20_PHP5}}"]')
```

or, if you are running on WebSphere Application Server Version 7.0 or later:

```
AdminTask.createPHPDynamicCluster('dcName','[-membershipPolicy "node_property$APACHE_2_0 IS NOT NULL AND node_property$PHP_5 IS NOT NULL" -dynamicClusterProperties "[[operationalMode automatic][minInstances 1][maxInstances -1][numVerticalInstances 1]]" -clusterProperties "[[templateName APACHE20_PHP5]]"]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createPHPDynamicCluster {-interactive}
```

- Using Jython string:

```
AdminTask.createPHPDynamicCluster ('[-interactive]')
```

createDynamicClusterFromStaticCluster

The `createDynamicClusterFromStaticCluster` command creates a new dynamic cluster from an existing static cluster.

Optional parameters

- **-templateName:** Specifies the name of the existing static cluster member template. The format of the name must be in the `cell_name/node_name/server_name` format. If the template is not specified, an existing static cluster member template is randomly chosen. If no members exist, the default server template is used. (String, optional)
- **-dynamicClusterProperties:** Specifies the dynamic cluster properties. (String, optional) You can specify the following options:
 - **operationalMode:** Specifies the operational mode. The valid values are: `manual`, `supervised`, and `automatic`.
 - **minInstances:** Specifies the minimum number of cluster instances as an integer.
 - **serverInactivityTime:** Specifies the time to wait before stopping instances, in minutes. This value is valid only when the `minInstances` parameter value is set to 0.
 - **maxInstances:** Specifies the maximum number of cluster instances as an integer. If you do not want a limit to the number of cluster instances that can start, specify `-1`.

Return value

The command returns the `javax.management.ObjectName` value of the dynamic cluster that you created.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDynamicClusterFromStaticCluster clusterName {-templateName cellName/nodeName/serverName}
```
- Using Jython string:

```
AdminTask.createDynamicClusterFromStaticCluster('clusterName')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDynamicClusterFromStaticCluster {-interactive}
```
- Using Jython string:

```
AdminTask.createDynamicClusterFromStaticCluster ('[-interactive]')
```

createDynamicClusterFromForeignServers

The `createDynamicClusterFromForeignServers` command creates a new dynamic cluster of assisted life-cycle servers. Create representations of these servers with an administrative task or in the administrative console before you create a dynamic cluster.

Required parameters

- **-foreignServers [[node_name1 server_name1][node_name2 server_name2] ...]:** Specifies the node and server names of the existing other middleware servers. (String, required)

Optional parameters

- **-dynamicClusterProperties:** Specifies the dynamic cluster properties. (String, optional) You can specify the following options:
 - **operationalMode:** Specifies the operational mode. The valid values are: `manual`, `supervised`, and `automatic`.
 - **minInstances:** Specifies the minimum number of cluster instances as an integer.

- **serverInactivityTime**: Specifies the time to wait before stopping instances, in minutes. This value is valid only when the `minInstances` parameter value is set to 0.
- **maxInstances**: Specifies the maximum number of cluster instances as an integer. If you do not want a limit to the number of cluster instances that can start, specify -1.

Return value

The command returns the `javax.management.ObjectName` identification of the server type that you updated.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDynamicClusterFromForeignServers dcName {-foreignServers {{nodeName1 serverName1}
{nodeName2 serverName2}}}
```

- Using Jython string:

```
AdminTask.createDynamicClusterFromForeignServers('dcName', ['-foreignServers [[nodeName1 serverName1]
[nodeName2 serverName2]]'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDynamicClusterFromForeignServers {-interactive}
```

- Using Jython string:

```
AdminTask.createDynamicClusterFromForeignServers ('[-interactive]')
```

addForeignServersToDynamicCluster

The `addForeignServersToDynamicCluster` command adds foreign servers to an existing dynamic cluster.

Required parameters

- **-foreignServers [[node_name1 server_name1] [node_name2 server_name1] ...]**: Specifies the node and server names of the existing other middleware servers. (String, required)

Return value

The command returns the `javax.management.ObjectName` identification of the server type that you updated.

Batch mode example usage

- Using Jacl:

```
$AdminTask addForeignServersToDynamicCluster dcName {-foreignServers {{nodeName1 serverName1}
{nodeName2 serverName2}}}
```

- Using Jython string:

```
AdminTask.addForeignServersToDynamicCluster('dcName', ['-foreignServers [[nodeName1 serverName1]
[nodeName2 serverName2]]'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask addForeignServersToDynamicCluster {-interactive}
```

- Using Jython string:

```
AdminTask.addForeignServersToDynamicCluster ('[-interactive]')
```

removeForeignServersFromDynamicCluster

The `removeForeignServersFromDynamicCluster` command removes foreign servers from the dynamic cluster

Required parameters

- `-foreignServers [[node_name1 server_name1][node_name2 server_name1] ...]`: Specifies the node and server names of the existing other middleware servers. (String, required)

Return value

The command returns the `javax.management.ObjectName` identification of the server type that you updated.

Batch mode example usage

- Using Jacl:

```
$AdminTask removeForeignServersFromDynamicCluster dcName {-foreignServers {{nodeName1 serverName1}{nodeName2 serverName2}}}
```
- Using Jython string:

```
AdminTask.removeForeignServersFromDynamicCluster('dcName', ['-foreignServers [[nodeName1 serverName1] [nodeName2 serverName2]]'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask removeForeignServersFromDynamicCluster {-interactive}
```
- Using Jython string:

```
AdminTask.removeForeignServersFromDynamicCluster ('[-interactive]')
```

deleteDynamicCluster

The `deleteDynamicCluster` deletes a dynamic cluster from the configuration.

Return value

The command returns the `javax.management.ObjectName` identification of the dynamic cluster that you deleted.

Batch mode example usage

- Using Jacl:

```
$AdminTask deleteDynamicCluster dcName
```
- Using Jython string:

```
AdminTask.deleteDynamicCluster('dcName')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask deleteDynamicCluster {-interactive}
```
- Using Jython string:

```
AdminTask.deleteDynamicCluster ('[-interactive]')
```

getDynamicClusterMembers

The `getDynamicClusterMembers` command displays the members of the specified dynamic cluster and node name. If the node name is not specified, then all of the members of the dynamic cluster are displayed.

Optional parameters

- **-nodeName:** Specifies the name of a node. (String, optional)

Return value

The command returns a list of `ClusterMember` objects.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMembers dcName`
- Using Jython string:
`AdminTask.getDynamicClusterMembers('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMembers {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterMembers ('[-interactive]')`

getDynamicClusterOperationalMode

The `getDynamicClusterOperationalMode` command displays the operational mode of the dynamic cluster.

Return value

The command returns the value of `operationalMode`. String.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterOperationalMode dcName`
- Using Jython string:
`AdminTask.getDynamicClusterOperationalMode('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterOperationalMode {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterOperationalMode ('[-interactive]')`

getDynamicClusterMembershipPolicy

The `getDynamicClusterMembershipPolicy` command displays the dynamic cluster membership policy.

Return value

The command returns the value of `membershipPolicy`. String.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMembershipPolicy dcName`
- Using Jython string:
`AdminTask.getDynamicClusterMembershipPolicy('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMembershipPolicy {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterMembershipPolicy ('[-interactive]')`

getDynamicClusterServerType

The **getDynamicClusterServerType** command displays the dynamic cluster server type.

Return value

The command returns the value of `serverType`. String.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterServerType dcName`
- Using Jython string:
`AdminTask.getDynamicClusterServerType('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterServerType {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterServerType ('[-interactive]')`

getDynamicClusterMinInstances

The **getDynamicClusterMinInstances** command displays the minimum number of cluster instances for the specified dynamic cluster.

Return value

The command returns the value of `minInstances`. Integer.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMinInstances dcName`
- Using Jython string:
`AdminTask.getDynamicClusterMinInstances('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMinInstances {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterMinInstances ('[-interactive]')`

getDynamicClusterMaxInstances

The **getDynamicClusterMaxInstances** command displays the maximum number of cluster instances for the specified dynamic cluster.

Return value

The command returns the value of maxInstances. Integer.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMaxInstances dcName`
- Using Jython string:
`AdminTask.getDynamicClusterMaxInstances('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterMaxInstances {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterMaxInstances ('[-interactive]')`

getDynamicClusterVerticalInstances

The **getDynamicClusterVerticalInstances** command displays the number of dynamic cluster vertical stacking of instances on the node.

Return value

The command returns the value of numVerticalInstances. Integer.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterVerticalInstances dcName`
- Using Jython string:
`AdminTask.getDynamicClusterVerticalInstances('dcName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterVerticalInstances {-interactive}`
- Using Jython string:
`AdminTask.getDynamicClusterVerticalInstances ('[-interactive]')`

getDynamicClusterIsolationProperties

The **getDynamicClusterVerticalInstances** command displays the dynamic cluster isolation properties.

Return value

The command returns the value of isolationGroup and strictIsolationEnabled. String.

Batch mode example usage

- Using Jacl:
`$AdminTask getDynamicClusterIsolationProperties dcName`
- Using Jython string:
`AdminTask.getDynamicClusterIsolationProperties('dcName')`

Interactive mode example usage

- Using Jacl:
\$AdminTask getDynamicClusterIsolationProperties {-interactive}
- Using Jython string:
AdminTask.getDynamicClusterIsolationProperties ('[-interactive]')

listDynamicClusters

The **listDynamicClusters** command displays all the dynamic clusters in the cell.

Return value

The command returns each of the dynamic cluster names. String.

Batch mode example usage

- Using Jacl:
\$AdminTask listDynamicClusters
- Using Jython string:
AdminTask.listDynamicClusters()

Interactive mode example usage

- Using Jacl:
\$AdminTask listDynamicClusters {-interactive}
- Using Jython string:
AdminTask.listDynamicClusters ('[-interactive]')

listDynamicClusterIsolationGroups

The **listDynamicClusterIsolationGroups** command displays all the dynamic cluster isolation groups in the cell.

Return value

The command returns a list of the dynamic cluster isolation groups. String.

Batch mode example usage

- Using Jacl:
\$AdminTask listDynamicClusterIsolationGroups
- Using Jython string:
AdminTask.listDynamicClusterIsolationGroups()

Interactive mode example usage

- Using Jacl:
\$AdminTask listDynamicClusterIsolationGroups {-interactive}
- Using Jython string:
AdminTask.listDynamicClusterIsolationGroups ('[-interactive]')

listDynamicClusterIsolationGroupMembers

The **listDynamicClusterIsolationGroupMembers** command displays the dynamic cluster isolation group members for the specified isolation group.

Return value

The command returns a list of the dynamic cluster isolation group members. String.

Batch mode example usage

- Using Jacl:
`$AdminTask listDynamicClusterIsolationGroupMembers isolationGroupName`
- Using Jython string:
`AdminTask.listDynamicClusterIsolationGroupMembers('isolationGroupName')`

Interactive mode example usage

- Using Jacl:
`$AdminTask listDynamicClusterIsolationGroupMembers {-interactive}`
- Using Jython string:
`AdminTask.listDynamicClusterIsolationGroupMembers ('[-interactive]')`

modifyDynamicClusterIsolationProperties

The **modifyDynamicClusterIsolationProperties** command modifies dynamic cluster isolation properties.

Optional parameters

- **-isolationGroup**: Specifies the name of the isolation group. (String, optional)
- **-strictIsolationEnabled**: Indicates if strict isolation is enabled. (String, optional)

Return value

The return value is void.

Batch mode example usage

- Using Jacl:
`$AdminTask modifyDynamicClusterIsolationProperties dcName {-isolationGroup isolationGroupName -strictIsolationEnabled true}`
- Using Jython string:
`AdminTask.modifyDynamicClusterIsolationProperties('dcName', '[-isolationGroup isolationGroupName -strictIsolationEnabled true]')`

Interactive mode example usage

- Using Jacl:
`$AdminTask modifyDynamicClusterIsolationProperties {-interactive}`
- Using Jython string:
`AdminTask.modifyDynamicClusterIsolationProperties ('[-interactive]')`

setDynamicClusterOperationalMode

The **setDynamicClusterOperationalMode** command sets the operational mode for the dynamic cluster.

Required parameters

- **-operationalMode**: Specifies the operational mode of the dynamic cluster. Set the value to `automatic`, `manual`, or `supervised`. (String, required)

Return value

The return value is void.

Batch mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterOperationalMode dcName {-operationalMode manual}
```

- Using Jython string:

```
AdminTask.setDynamicClusterOperationalMode('dcName', '[-operationalMode manual]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterOperationalMode {-interactive}
```

- Using Jython string:

```
AdminTask.setDynamicClusterOperationalMode ('[-interactive]')
```

setDynamicClusterMembershipPolicy

The **setDynamicClusterMembershipPolicy** command sets the membership policy for the dynamic cluster.

Required parameters

-membershipPolicy

Specifies the membership policy subexpression. (String, required)

Return value

The return value is void.

Batch mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterMembershipPolicy dcName {-membershipPolicy node_nodegroup='ngName'}
```

- Using Jython string:

```
AdminTask.setDynamicClusterMembershipPolicy('dcName', '[-membershipPolicy node_nodegroup=\'ngName\']')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterMembershipPolicy {-interactive}
```

- Using Jython string:

```
AdminTask.setDynamicClusterMembershipPolicy ('[-interactive]')
```

setDynamicClusterMinInstances

The **setDynamicClusterMinInstances** command sets the minimum number of cluster instances for the dynamic cluster.

Required parameters

-minInstances

Specifies the minimum number of cluster instances. (Integer, required)

Return value

The return value is void.

Batch mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterMinInstances dcName {-minInstances 1}
```

- Using Jython string:

```
AdminTask.setDynamicClusterMinInstances('dcName', '[-minInstances 1]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterMinInstances {-interactive}
```
- Using Jython string:

```
AdminTask.setDynamicClusterMinInstances ('[-interactive]')
```

setDynamicClusterMaxInstances

The **setDynamicClusterMaxInstances** command sets the maximum number of cluster instances for the dynamic cluster.

Required parameters

-maxInstances

Specifies the minimum number of cluster instances. (Integer, required)

Return value

The return value is void.

Batch mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterMaxInstances dcName {-maxInstances -1}
```
- Using Jython string:

```
AdminTask.setDynamicClusterMaxInstances('dcName', '[-maxInstances 1]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterMaxInstances {-interactive}
```
- Using Jython string:

```
AdminTask.setDynamicClusterMaxInstances ('[-interactive]')
```

setDynamicClusterVerticalInstances

The **setDynamicClusterVerticalInstances** command sets the number of dynamic cluster vertical stacking instances on the node.

Required parameters

-numVerticalInstances

Specifies the number of vertical stacking of instances on a node.(Integer, required)

Return value

The return value is void.

Batch mode example usage

- Using Jacl:

```
$AdminTask setDynamicClusterVerticalInstances dcName {-numVerticalInstances 2}
```
- Using Jython string:

```
AdminTask.setDynamicClusterVerticalInstances('dcName', '[-numVerticalInstances 2]')
```

Interactive mode example usage

- Using Jacl:
`$AdminTask setDynamicClusterVerticalInstances {-interactive}`
- Using Jython string:
`AdminTask.setDynamicClusterVerticalInstances ('[-interactive]')`

testDynamicClusterMembershipPolicy

The **testDynamicClusterMembershipPolicy** command tests the dynamic cluster membership policy to see which nodes are returned.

Required parameters

- **-membershipPolicy**
Specifies the membership policy subexpression. (String, required)

Return value

The command returns a String [] list of node names.

Batch mode example usage

- Using Jacl:
`$AdminTask testDynamicClusterMembershipPolicy {-membershipPolicy node_nodegroup='ngName'}`
- Using Jython string:
`AdminTask.testDynamicClusterMembershipPolicy('[-membershipPolicy node_nodegroup=\'ngName\']')`

Interactive mode example usage

- Using Jacl:
`$AdminTask testDynamicClusterMembershipPolicy {-interactive}`
- Using Jython string:
`AdminTask.testDynamicClusterMembershipPolicy ('[-interactive]')`

createNonWASDynamicCluster

The **createNonWASDynamicCluster** command creates a new dynamic cluster of complete life-cycle management servers.

Required parameters

- **-serverType**: Specifies the server type. (String, required)
To see the server types, view the `install_root\profiles\dmgr_profile\config\templates\servertypes` directory.
- **-membershipPolicy**: Specifies the membership policy subexpression. (String, required)

Optional parameters

- **-dynamicClusterProperties**: Specifies the dynamic cluster properties. (String, optional) You can specify the following options:
 - **operationalMode**: Specifies the operational mode. The valid values are: manual, supervised, and automatic.
 - **minInstances**: Specifies the minimum number of cluster instances as an integer.
 - **serverInactivityTime**: Specifies the time to wait before stopping instances, in minutes. This value is valid only when the minInstances parameter value is set to 0.

- **maxInstances:** Specifies the maximum number of cluster instances as an integer. If you do not want a limit to the number of cluster instances that can start, specify -1.
- **numVerticalInstances:** Specifies the number of instances that can start on the same node by creating vertical instances. Specify an integer value.
- **-clusterProperties:** Specifies the cluster properties (String, optional)
 - **templateName:** Specifies the name of the server template. The valid input includes the following values:
 - **For PHP dynamic clusters:** APACHE13_PHP4, APACHE13_PHP5, APACHE20_PHP4, APACHE20_PHP5, APACHE22_PHP4, APACHE22_PHP5, cellName/nodeName/php_serverName.
 - **For WebSphere Application Server Community Edition dynamic clusters:** wasce20, cell_name/node_name/wasce_server_name

Return value

The command returns the `javax.management.ObjectName` value of the dynamic cluster that you created.

Batch mode example usage

- Using Jacl:

```
$AdminTask createNonWASDynamicCluster dcName {-serverType WASCE_SERVER -membershipPolicy
"node_property$com.ibm.websphere.wasceProductShortName = 'WASCE' OR node_property$WASCE_2.0.0.0
IS NOT NULL" -dynamicClusterProperties "{operationalMode automatic}{minInstances 1}{maxInstances -1}
{numVerticalInstances 1}" -clusterProperties "{templateName wasce20}"}
```

- Using Jython string:

```
AdminTask.createNonWASDynamicCluster(dcName,['-serverType WASCE_SERVER -membershipPolicy
"node_property$com.ibm.websphere.wasceProductShortName = \'WASCE\' OR node_property$WASCE_2.0.0.0
IS NOT NULL" -dynamicClusterProperties "{operationalMode automatic}{minInstances 1}{maxInstances -1}
{numVerticalInstances 1}]" -clusterProperties "{templateName wasce20}"])
```

or, if you are running on WebSphere Application Server Version 7.0 or later:

```
AdminTask.createNonWASDynamicCluster('dcName',['-serverType WASCE_SERVER -membershipPolicy
"node_property$com.ibm.websphere.wasceProductShortName = \'WASCE\' OR node_property$WASCE_2.0.0.0
IS NOT NULL" -dynamicClusterProperties "[operationalMode automatic][minInstances 1][maxInstances -1]
[numVerticalInstances 1]" -clusterProperties "[templateName wasce20]"'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createNonWASDynamicCluster {-interactive}
```

- Using Jython string:

```
AdminTask.createNonWASDynamicCluster(['-interactive'])
```

Edit the dynamic cluster server template

Identify the dynamic cluster server template and assign the template to the `serverid` variable.

Batch mode example usage

- Using Jacl:

```
set serverid [$AdminConfig getid /Cell:mycell/DynamicCluster:mydynamiccluster/Server:mydynamiccluster/]
```

- Using Jython string:

```
serverid = AdminConfig.getid('/Cell:mycell/DynamicCluster:mydynamiccluster/Server:mydynamiccluster/')
```

You can use the `serverid` variable to modify, create, or delete configuration objects in the dynamic cluster server template.

Default work class administrative tasks

You can use the default work class administrative tasks to create work classes.

The administrative tasks for default work classes include the following commands:

- “createDefaultASPWorkClass”
- “createDefaultARPWorkClass”
- “createDefaultGSPWorkClass” on page 359
- “createDefaultGRPWorkClass” on page 359
- “createDefaultSystemSPWorkClass” on page 360
- “createDefaultSystemRPWorkClass” on page 360

createDefaultASPWorkClass

The `createDefaultASPWorkClass` command creates default application service policy work classes.

Required parameters

-apptime

Specifies the name of the application deployment. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:


```
$AdminTask createDefaultASPWorkClass {-apptime A-edition1.0}
```
- Using Jython:


```
AdminTask.createDefaultASPWorkClass ('[-apptime A-edition1.0]')
```

Interactive mode example usage

- Using Jacl:


```
$AdminTask createDefaultASPWorkClass {-interactive}
```
- Using Jython:


```
AdminTask.createDefaultASPWorkClass ('[-interactive]')
```

createDefaultARPWorkClass

The `createDefaultARPWorkClass` command creates default application routing policy work classes.

Required parameters

-apptime

Specifies the name of the application deployment. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:


```
$AdminTask createDefaultARPWorkClass {-apptime A-edition1.0}
```
- Using Jython:


```
AdminTask.createDefaultARPWorkClass ('[-apptime A-edition1.0]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDefaultARPWorkClass {-interactive}
```

- Using Jython:

```
AdminTask.createDefaultARPWorkClass ('[-interactive]')
```

createDefaultGSPWorkClass

The **createDefaultGSPWorkClass** command creates default generic server service policy work classes.

Required parameters

-odrname

Specifies the name of the on demand router (ODR). (String, required)

-odrnnode

Specifies the name of the node where the ODR runs. (String, required)

-vhost

Specifies the name of the virtual host. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDefaultGSPWorkClass {-odrname myodr -odrnnode mynode -vhost default_host}
```

- Using Jython:

```
AdminTask.createDefaultGSPWorkClass ('[-odrname myodr -odrnnode mynode -vhost default_host]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDefaultGSPWorkClass {-interactive}
```

- Using Jython:

```
AdminTask.createDefaultGSPWorkClass ('[-interactive]')
```

createDefaultGRPWorkClass

The **createDefaultGRPWorkClass** command creates default generic server routing policy work classes.

Required parameters

-odrname

Specifies the name of the on demand router (ODR). (String, required)

-odrnnode

Specifies the name of the node where the ODR runs. (String, required)

-vhost

Specifies the name of the virtual host. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDefaultGRPWorkClass {-odrname myodr -odrnnode mynode -vhost default_host}
```

- Using Jython:

```
AdminTask.createDefaultGRPWorkClass ('[-odurname myodr -odrnnode mynode -vhost default_host]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDefaultGRPWorkClass {-interactive}
```

- Using Jython:

```
AdminTask.createDefaultGRPWorkClass ('[-interactive]')
```

createDefaultSystemSPWorkClass

The **createDefaultSystemSPWorkClass** command creates default system application service policy work classes.

Required parameters

-appname

Specifies the name of the application deployment. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDefaultSystemSPWorkClass {-appname A-edition1.0}
```

- Using Jython:

```
AdminTask.createDefaultSystemSPWorkClass ('[-appname A-edition1.0]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDefaultSystemSPWorkClass {-interactive}
```

- Using Jython:

```
AdminTask.createDefaultSystemSPWorkClass ('[-interactive]')
```

createDefaultSystemRPWorkClass

The **createDefaultSystemRPWorkClass** command creates default system application routing policy work classes.

Required parameters

-appname

Specifies the name of the application deployment. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask createDefaultSystemRPWorkClass {-appname A-edition1.0}
```

- Using Jython:


```
AdminTask.createDefaultSystemRPWorkClass ('[-appname A-edition1.0]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createDefaultSystemRPWorkClass {-interactive}
```
- Using Jython:

```
AdminTask.createDefaultSystemRPWorkClass ('[-interactive]')
```

Health policy administrative tasks

You can use the following administrative tasks to create, delete, and manage health policies.

- “createHealthPolicy command”
- “modifyHealthPolicy command” on page 363
- “deleteHealthPolicy command” on page 365
- “listHealthPolicies command” on page 366
- “createHealthAction command” on page 366
- “modifyHealthAction command” on page 367
- “deleteHealthAction command” on page 368
- “listHealthActions command” on page 368
- “executeHealthAction command” on page 369

createHealthPolicy command

The `createHealthPolicy` command creates a new health policy.

Table 45. `createHealthPolicy` command arguments

Argument	Description
name (required)	Specifies a name for the health policy that is unique in the cell.
description	Specifies a description for the health policy.
reactionMode	Specifies if runtime tasks are created before taking actions on a health policy, or if actions are automatic. The value can be SUPERVISE or AUTOMATIC.

Table 46. `addCondition` step arguments

Argument	Description
type (required)	Specifies the condition type. The value can be CUSTOM, AGE, WORKLOAD, MEMORY_LEAK, EXCESSIVE_MEMORY, EXCESSIVE_RESPONSE_TIME, EXCESSIVE_REQUEST_TIMEOUT, or STORM_DRAIN. Each value has different parameters that you must specify.
expression	If you are using a custom condition, then specify a subexpression to evaluate.

Table 46. `addCondition` step arguments (continued)

Argument	Description
params	<ul style="list-style-type: none"> • AGE <ul style="list-style-type: none"> – maxAge: Specifies maximum age. – ageUnits: Specifies the type of units for the age parameter. The value can be HOURS or MINUTES. The default is HOURS. • WORKLOAD <ul style="list-style-type: none"> – totalRequests: Specifies total requests to service before restarting the members. The number must be greater than or equal to 1000. • MEMORY_LEAK <ul style="list-style-type: none"> – level: Specifies the detection level for the memory leak. The value can be FAST, NORMAL, or CONSERVATIVE. The default is FAST. • EXCESSIVE_MEMORY <ul style="list-style-type: none"> – timeOverThreshold: Specifies the offending time period for the excessive memory condition. The default is 1 MINUTE. – timeUnits: Specifies the units for the timeOverThreshold value. The value can be MINUTES or SECONDS. The default is MINUTES. – memoryUsed: Specifies a percentage that represents the maximum Java virtual machine (JVM) heap size to use for the JVM process. The value ranges from 1 to 99. • EXCESSIVE_RESPONSE_TIME <ul style="list-style-type: none"> – responseTime: Specifies the average response time that is considered to be excessive. When the average reaches this value, members restart. – responseTimeUnits: Specifies the time units for the response time. The value can be MINUTES, SECONDS, or MILLISECONDS. • EXCESSIVE_REQUEST_TIMEOUT <ul style="list-style-type: none"> – timeoutPercent: Specifies the threshold value for the percentage of the timed out requests to cause a breach of condition. This value ranges from 1 to 99. • STORM_DRAIN <ul style="list-style-type: none"> – level: Specifies the detection level. The value can be NORMAL or CONSERVATIVE. NORMAL is less accurate, but can detect a storm drain condition quickly. CONSERVATIVE is the most accurate, but takes a longer time to detect a storm drain condition.

Table 47. `addAction` step arguments.

Use the `addAction` step arguments in table format. With table format, you specify the parameters in order without the parameter name.

Argument	Description
type (required)	Specifies the action type. The value can be CUSTOM, HEAPDUMP, THREADDUMP, RESTART_SERVER, SET_MAINT_MODE, MAINT_MODE_BREAK, or MAINT_MODE_STOP.
stepNum (required)	Specifies the order in which to take the actions. Step 1 is completed first, and so on.
action	Specifies the name of a custom action. Use the <code>createHealthAction</code> command to create a custom action.
node	Specifies the node on which to take the action.
server	Specifies the server on which to take the action.

Table 48. addMember step arguments.

Use the addMember step arguments in table format. With table format, you specify the parameters in order without the parameter name.

Argument	Description
type (required)	Specifies the type of member. The value can be CELL, CLUSTER, DYNAMIC_CLUSTER, SERVER, or ODR.
name (required)	Specifies the name of the member.
node	Specifies the name of the node. The node name is required if you select the SERVER type.

createHealthPolicy command examples

Interactive mode example usage:

- Using Jacl:


```
$AdminTask createHealthPolicy {-interactive}
```
- Using Jython:


```
AdminTask.createHealthPolicy ('[-interactive]')
```

Batch mode example usage:

The following command creates a health policy with one of the predefined health conditions:

```
$AdminTask createHealthPolicy { -name myHealthPolicy -description "My Health Policy" -reactionMode SUPERVISE
-addCondition { -type AGE -params {{maxAge 12} {ageUnits HOURS}} } -addAction {{HEAPDUMP 1}{CUSTOM 2 myAction
myNode myServer} } -addMember {SERVER myServer myNode} }
```

The following command creates a health policy with a custom condition:

```
$AdminTask createHealthPolicy { -name myHealthPolicy -description "My Health Policy" -reactionMode SUPERVISE
-addCondition { -type CUSTOM -expression "ODRServerMetric_FromServerStart$errors > 100L" }
-addAction {{HEAPDUMP 1}{CUSTOM 2 myAction myNode myServer} } -addMember {SERVER myServer myNode} }
```

Put the expression for your custom condition in quotes.

modifyHealthPolicy command

The modifyHealthPolicy command changes an existing health policy.

Table 49. Initial arguments

Argument	Description
name (required)	Specifies the name of the health policy that you want to change.
reactionMode	Specifies if runtime tasks are created before taking actions on a health policy, or if actions are automatic. The value can be SUPERVISE or AUTOMATIC.

Table 50. modifyCondition step arguments

Argument	Description
type	Specifies the condition type. The value can be CUSTOM, AGE, WORKLOAD, MEMORY_LEAK, EXCESSIVE_MEMORY, EXCESSIVE_RESPONSE_TIME, EXCESSIVE_REQUEST_TIMEOUT, or STORM_DRAIN. Each value has different parameters that you must specify.
expression	If you are using a custom condition, then specify a subexpression to evaluate.

Table 50. `modifyCondition` step arguments (continued)

Argument	Description
params	<ul style="list-style-type: none"> • AGE <ul style="list-style-type: none"> – maxAge: Specifies maximum age. – ageUnits: Specifies the type of units for the age parameter. The value can be HOURS or MINUTES. The default is HOURS. • WORKLOAD <ul style="list-style-type: none"> – totalRequests: Specifies total requests to service before restarting the members. The number must be greater than or equal to 1000. • MEMORY_LEAK <ul style="list-style-type: none"> – level: Specifies the detection level for the memory leak. The value can be FAST, NORMAL, or CONSERVATIVE. The default is FAST. • EXCESSIVE_MEMORY <ul style="list-style-type: none"> – timeOverThreshold: Specifies the offending time period for the excessive memory condition. The default is 1 MINUTE. – timeUnits: Specifies the units for the timeOverThreshold value. The value can be MINUTES or SECONDS. The default is MINUTES. – memoryUsed: Specifies a percentage that represents the maximum Java virtual machine (JVM) heap size to use for the JVM process. The value ranges from 1 to 99. • EXCESSIVE_RESPONSE_TIME <ul style="list-style-type: none"> – responseTime: Specifies the average response time that is considered to be excessive. When the average reaches this value, members restart. – responseTimeUnits: Specifies the time units for the response time. The value can be MINUTES, SECONDS, or MILLISECONDS. • EXCESSIVE_REQUEST_TIMEOUT <ul style="list-style-type: none"> – timeoutPercent: Specifies the threshold value for the percentage of the timed out requests to cause a breach of condition. This value ranges from 1 to 99. • STORM_DRAIN <ul style="list-style-type: none"> – level: Specifies the detection level. The value can be NORMAL or CONSERVATIVE. NORMAL is less accurate, but can detect a storm drain condition quickly. CONSERVATIVE is the most accurate, but takes a longer time to detect a storm drain condition.

Table 51. `removeAction` step arguments

Argument	Description
name	Specifies the name of the health action to remove. The name parameter is only used for custom actions.
type (required)	Specifies the action type. The value can be CUSTOM, HEAPDUMP, THREADDUMP, RESTART_SERVER, SET_MAINT_MODE, MAINT_MODE_BREAK, or MAINT_MODE_STOP.

Table 52. `addAction` step arguments

Argument	Description
type (required)	Specifies the action type. The value can be CUSTOM, HEAPDUMP, THREADDUMP, RESTART_SERVER, SET_MAINT_MODE, MAINT_MODE_BREAK, or MAINT_MODE_STOP.
name	Specifies the name of the health action. The name parameter is only used for custom actions.
node	Specifies the node on which to take the action.
server	Specifies the server on which to take the action.

Table 52. `addAction` step arguments (continued)

Argument	Description
stepNum (required)	Specifies the order in which to take the actions. Step 1 is completed first, and so on.

Table 53. `orderAction` step arguments

Argument	Description
name	Specifies the name of the health action. The name parameter is only used for custom actions.
type	Specifies the action type. The value can be CUSTOM, HEAPDUMP, THREADDUMP, RESTART_SERVER, SET_MAINT_MODE, MAINT_MODE_BREAK, or MAINT_MODE_STOP.
stepNum (required)	Specifies the order in which to take the actions. Step 1 is completed first, and so on.

Table 54. `addMember` step arguments

Argument	Description
type (required)	Specifies the type of member. The value can be CELL, CLUSTER, DYNAMIC_CLUSTER, SERVER, or ODR.
name (required)	Specifies the name of the member.
node	Specifies the name of the node. The node name is required if you select the SERVER type.

modifyHealthPolicy command examples

Interactive mode example usage:

- Using Jacl:


```
$AdminTask modifyHealthPolicy {-interactive}
```
- Using Jython:


```
AdminTask.modifyHealthPolicy ('[-interactive]')
```

Batch mode example usage:

```
$AdminTask modifyHealthPolicy { -name myHealthPolicy -removeAction { -type HEAPDUMP } }
```

deleteHealthPolicy command

The `deleteHealthPolicy` command removes the named health policy.

Table 55. `deleteHealthPolicy` command arguments

Argument	Description
name (required)	Specifies the health policy name to delete.

deleteHealthPolicy command examples

Batch mode example usage:

- Using Jacl:


```
$AdminTask deleteHealthPolicy {-name MyHealthPolicy}
```
- Using Jython:


```
AdminTask.deleteHealthPolicy('MyHealthPolicy')
```

Interactive mode example usage:

- Using Jacl:
\$AdminTask deleteHealthPolicy {-interactive}
- Using Jython:
AdminTask.deleteHealthPolicy ('[-interactive]')

listHealthPolicies command

The **listHealthPolicies** command lists each configured health policy in the cell.

listHealthPolicies command examples

- Using Jacl:
\$AdminTask listHealthPolicies
- Using Jython:
AdminTask.listHealthPolicies

createHealthAction command

The **createHealthAction** command creates a new health action that you can use when you define your health policies.

Table 56. createHealthAction command arguments

Argument	Description
name (required)	Specifies a name for the health action.

You can create a Java action or an action that is any other type of executable file.

Table 57. non-Java step arguments

Argument	Description
executable	Specifies the executable file path. This argument is required if you specify a nonJava step argument.
executableArgs	Specifies a string array of arguments to pass to the executable file.
osNames	Specifies a comma-delimited list of operating systems. Specify null to have the action apply for each operating system. The valid operating system names include: windows, unix, and zos.
workingDir	Specifies the location in the file system from where the command is invoked. This argument is required if you specify a nonJava step argument.
usernameVar	Specifies a variable for the user name.
userNameVal	Specifies the value of the user name variable.
passwordVar	Specifies a variable for the password.
passwordVal	Specifies the value of the password variable.

Table 58. Java step arguments

Argument	Description
javaDir	Specifies the location of your Java executable file, for example, \${JAVA_HOME}. You can specify a variable or a path.
executableTargetType	Specifies the file type that you are using for your Java executable file. The value can be EXECUTABLE_JAR or JAVA_CLASS.
executable	Specifies the executable file path. This argument is required if you specify a Java step argument.

Table 58. Java step arguments (continued)

Argument	Description
executableArgs	Specifies a string array of arguments to pass to the executable file.
osNames	Specifies a comma-delimited list of operating systems. Specify null to have the action apply for each operating system. The valid operating system names include: windows, unix, and zos.
pidVarName	Specifies the Java process id variable name.
workingDir	Specifies the location in the file system from where the command is invoked. This argument is required if you specify a Java step argument.
usernameVar	Specifies a variable for the user name.
usernameVal	Specifies the value of the user name variable.
passwordVar	Specifies a variable for the password.
passwordVal	Specifies the value of the password variable.

createHealthAction command examples

Batch mode example usage:

- Using Jacl:

```
$AdminTask createHealthAction { -name myAction -java { -javaDir c:\java\bin -executableTargetType EXECUTABLE_JAR
  -executable test.jar -executableArgs "{arg1}{arg2a arg2b}" -osNames "windows, unix" -pidVarName pvn
  -workingDir c:\workingDir } }
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask createHealthAction {-interactive}
```

- Using Jython

```
AdminTask.createHealthAction(['interactive'])
```

modifyHealthAction command

Use the **modifyHealthAction** command to modify an existing health action.

Table 59. modifyHealthAction command arguments

Argument	Description
name (required)	Specifies a name for the health action that you want to modify.

Table 60. non-Java step arguments

Argument	Description
executable	Specifies the executable file path.
executableArgs	Specifies a string array of arguments to pass to the executable file.
workingDir	Specifies the location in the file system from where the command is invoked.
usernameVar	Specifies a variable for the user name.
usernameVal	Specifies the value of the user name variable.
passwordVar	Specifies a variable for the password.
passwordVal	Specifies the value of the password variable.

Table 61. Java step arguments

Argument	Description
javaDir	Specifies the location of your Java executable file, for example, <code>\${JAVA_HOME}</code> . You can specify a variable or a path.
executableTargetType	Specifies the file type that you are using for your Java executable file. The value can be <code>EXECUTABLE_JAR</code> or <code>JAVA_CLASS</code> .
executable	Specifies the executable file path.
executableArgs	Specifies a string array of arguments to pass to the executable file.
pidVarName	Specifies the Java process id variable name.
usernameVar	Specifies a variable for the user name.
usernameVal	Specifies the value of the user name variable.
passwordVar	Specifies a variable for the password.
passwordVal	Specifies the value of the password variable.
workingDir	Specifies the location in the file system from where the command is invoked.

modifyHealthAction command examples

Batch mode example usage:

```
$AdminTask modifyHealthAction { -name myAction -java { -javaDir c:\java\bin -executableTargetType EXECUTABLE_JAR
-executable test.jar -executableArgs "{arg1}{arg2a arg2b}" -pidVarName pvn -workingDir c:\workingDir } }
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask modifyHealthAction {-interactive}
```
- Using Jython

```
AdminTask.modifyHealthAction(['interactive'])
```

deleteHealthAction command

The `deleteHealthAction` command removes a health action.

Table 62. `deleteHealthAction` command arguments

Argument	Description
name (required)	Specifies a name for the health action.

deleteHealthAction command examples

Batch mode example usage:

- Using Jacl:

```
$AdminTask deleteHealthAction {-name myHealthAction}
```
- Using Jython:

```
AdminTask.deleteHealthAction('myHealthAction')
```

listHealthActions command

The `listHealthActions` command lists each of the defined health actions in the cell.

listHealthActions command examples

Batch mode example usage:

- Using Jacl:
`$AdminTask listHealthActions`
- Using Jython:
`AdminTask.listHealthActions`

executeHealthAction command

The `executeHealthAction` command runs the specified health action.

Table 63. `executeHealthAction` command arguments

Argument	Description
name (required)	Specifies the name of the health action that you want to run.
node (required)	Specifies the node on which to take the action.
server (required)	Specifies the server on which to take the action.
cell	Specifies the cell on which to run the health action.
timeout	Specifies a timeout value for running the health action.
variables	Specifies <code>java.util.Properties</code> values of variables.

executeHealthAction command examples

Batch mode example usage:

- Using Jacl:
`$AdminTask executeHealthAction {-name myHealthAction -node myNode -server myServer}`
- Using Jython:
`AdminTask.executeHealthAction ('[-name myHealthAction -node myNode -server myServer]')`

Interactive mode example usage:

- Using Jacl:
`$AdminTask executeHealthAction {-interactive}`
- Using Jython:
`AdminTask.executeHealthAction ('[interactive]')`

Maintenance mode administrative tasks

You can use the server maintenance mode commands to view or edit the maintenance mode of your servers without using the administrative console. To set maintenance mode on a node, you can use the `NodeGroupManager MBean` .

Use the following commands to set, disable, and check maintenance mode for your servers:

- “setMaintenanceMode” on page 370
- “unsetMaintenanceMode” on page 370
- “checkMode” on page 371

To set, disable, and check maintenance mode for a node:

- “Node maintenance mode” on page 371

setMaintenanceMode

The **setMaintenanceMode** command puts a server into maintenance mode.

Required parameters

-name

Specifies the name of the server that you want to put into maintenance mode.

Optional parameters

-mode

Mode is an optional parameter. If you do not specify a value, then the default is used.

Possible values:

break, affinity, or stop.

Specify **break** to break Hypertext Transfer Protocol (HTTP) or Session Initiation Protocol (SIP) affinity and keep the server running. Specify **affinity** to keep HTTP or SIP affinity active and keep the server running. Specify **stop** to stop the server immediately and put the server in affinity mode.

Default: affinity

Return value

The command returns void.

Batch mode example usage

- Using Jacl:

```
$AdminTask setMaintenanceMode xdnodel {-name test1 -mode break}
```
- Using Jython string:

```
AdminTask.setMaintenanceMode ('xdnodel','[-name test1 -mode affinity]')
```

Interactive mode example usage

- Using Jacl:

```
AdminTask setMaintenanceMode {-interactive}
```
- Using Jython string:

```
AdminTask.setMaintenanceMode ('[-interactive]')
```

unsetMaintenanceMode

The **unsetMaintenanceMode** command takes the server out of maintenance mode.

Required parameters

-name

Specifies the name of the server that you want to take out of maintenance mode.

Return value

The command returns void.

Batch mode example usage

- Using Jacl:

```
$AdminTask unsetMaintenanceMode xdnodel {-name test1}
```
- Using Jython string:

```
AdminTask.unsetMaintenanceMode ('xdnode1','[-name test1]')
```

Interactive mode example usage

- Using Jacl:

```
AdminTask unsetMaintenanceMode {-interactive}
```

- Using Jython string:

```
AdminTask.unsetMaintenanceMode ('[-interactive]')
```

checkMode

The **checkMode** command checks whether a server is in maintenance mode.

Required parameters

-name

Specifies the name of the server that you want to check.

Return value

The command returns a string value that indicates the server mode.

Batch mode example usage

- Using Jacl:

```
AdminTask checkMode xdnodel {-name test1}
```

- Using Jython string:

```
AdminTask.checkMode ('xdnode1','[-name test1]')
```

Interactive mode example usage

- Using Jacl:

```
AdminTask checkMode {-interactive}
```

- Using Jython string:

```
AdminTask.checkMode ('[-interactive]')
```

Node maintenance mode

You can use the **NodeGroupManager MBean** to enable or disable maintenance mode for a node. For more information, read about **NodeGroupManager MBean**.

You can use the following example in **wsadmin** to set maintenance mode for a node:

```
set ngmMbean [$AdminControl queryNames WebSphere:*,type=NodeGroupManager,process=dmgr]
$AdminControl invoke $ngmMbean setMaintenanceMode {nodeName true true}
```

Middleware application administrative tasks

You can use the Jacl or Jython scripting languages to create and manage middleware applications with the **wsadmin** tool.

The administrative tasks for middleware applications include the following commands:

- addMiddlewareAppWebModule
- addMiddlewareTarget
- deployWasCEApp
- exportMiddlewareApp
- exportMiddlewareAppScript
- listMiddlewareAppEditions

- listMiddlewareApps
- listMiddlewareAppWebModules
- listMiddlewareTargets
- modifyMiddlewareAppWebModule
- modifyPHPApp
- modifyUnmanagedWebApp
- modifyWasCEApp
- removeMiddlewareAppWebModule
- removeMiddlewareTarget
- showMiddlewareApp
- installWasCEApp
- startWasCEApp
- stopWasCEApp
- undeployWasCEApp
- uninstallMiddlewareApp
- unregisterApp

addMiddlewareAppWebModule

The **addMiddlewareAppWebModule** command adds a Web module to a middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

-moduleName

Specifies the name of a module. (String, required)

-virtualHost

Specifies the name of the virtual host, which enables a single host machine to resemble multiple host machines. (String, required)

-contextRoot

Specifies the path prefix that is associated with the application. Use this parameter to compose the URL that is needed to access the application files. (String, required)

Optional parameters

-edition

Specifies the name of the edition. This parameter is required if the specified application has more than one edition. (String, optional)

-cluster

Specifies the name of the cluster. This parameter adds a dynamic cluster as a deployment target. (String, optional)

-node

Specifies the name of the node. (String, optional)

-server

Specifies the name of the server. Issue both the `server` and `node` parameters to add a standalone server as a deployment target. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask addMiddlewareAppWebModule {-app myJavaEEApplication -moduleName myModule -contextRoot /MyJ2EEApp
-virtualHost default_host -node AppServerNode1 -server tomcatServer1}
```

- Using Jython string:

```
AdminTask.addMiddlewareAppWebModule ('[-app myJavaEEApplication -moduleName myModule -contextRoot /MyJ2EEApp
-virtualHost default_host -node AppServerNode1 -server tomcatServer1]')
```

- Using Jython list:

```
AdminTask.addMiddlewareAppWebModule (['-app', 'myJavaEEApplication', '-moduleName', 'myModule', '-contextRoot',
'/MyJ2EEApp', '-virtualHost', 'default_host', '-node', 'AppServerNode1', '-server', 'tomcatServer1'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask addMiddlewareAppWebModule {-interactive}
```

- Using Jython string:

```
AdminTask.addMiddlewareAppWebModule ('[-interactive]')
```

- Using Jython list:

```
AdminTask.addMiddlewareAppWebModule (['-interactive'])
```

addMiddlewareTarget

The `addMiddlewareTarget` command adds a deployment target to a middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

Optional parameters

-edition

Specifies the name of the edition. This parameter is required if the specified application has more than one edition. (String, optional)

-module

Specifies the name of the module. Use this parameter to narrow the scope of a command to a single module. (String, optional)

-cluster

Specifies the name of the cluster. Use this parameter to add a dynamic cluster as a deployment target. (String, optional)

-node

Specifies the name of the node. (String, optional)

-server

Specifies the name of the server. Issue both the server and node parameters to add a standalone server as a deployment target. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask addMiddlewareTarget {-app myJavaEEApplication -node AppServerNode1 -server tomcatServer1}
```

- Using Jython string:

```
AdminTask.addMiddlewareTarget ('[-app myJavaEEApplication -node AppServerNode1 -server tomcatServer1]')
```

- Using Jython list:

```
AdminTask.addMiddlewareTarget (['-app', 'myJavaEEApplication', '-node', 'AppServerNode1', '-server', 'tomcatServer1'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask addMiddlewareTarget {-interactive}
```

- Using Jython string:

```
AdminTask.addMiddlewareTarget ('[-interactive]')
```

- Using Jython list:

```
AdminTask.addMiddlewareTarget (['-interactive'])
```

deployWASCEApp

The **deployWasCEApp** command adds a representation of a WebSphere Application Server Community Edition application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

-node

Specifies the name of the node. (String, required)

-server

Specifies the name of the server. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask deployWasCEApp {-app newSample -edition 2 -node nodeName -server WasCEServerRep}
```

- Using Jython string:

```
AdminTask.deployWasCEApp ('[-app newSample -edition 2 -node nodeName -server WasCEServerRep]')
```

- Using Jython list:

```
AdminTask.deployWasCEApp (['-app', 'newSample', '-edition', '2', '-node', 'nodeName',  
'-server', 'WasCEServerRep'])
```

Interactive mode example usage

- Using Jacl:
`$AdminTask deployWasCEApp {-interactive}`
- Using Jython string:
`AdminTask.deployWasCEApp ('[-interactive]')`
- Using Jython list:
`AdminTask.deployWasCEApp (['-interactive'])`

exportMiddlewareApp

The **exportMiddlewareApp** command exports a middleware application to a directory.

Target object

None.

Required parameters

- app**
Specifies the name of the middleware application. (String, required)
- dir**
Specifies the directory in which the application is located. (String, required)

Optional parameters

- edition**
Specifies the name of the edition. This parameter is required if the specified application has more than one edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask exportMiddlewareApp {-app sample -dir /home/user1}`
- Using Jython string:
`AdminTask.exportMiddlewareApp ('[-app sample -dir /home/user1]')`
- Using Jython list:
`AdminTask.exportMiddlewareApp (['-app', 'sample', '-dir', '/home/user1'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask exportMiddlewareApp {-interactive}`
- Using Jython string:
`AdminTask.exportMiddlewareApp ('[-interactive]')`
- Using Jython list:
`AdminTask.exportMiddlewareApp (['-interactive'])`

exportMiddlewareAppScript

The `exportMiddlewareAppScript` command exports the scripts of a middleware application to a directory.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

-dir

Specifies the name of the directory in which the middleware application script is located. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

-type

Specifies the type of script. The valid types are SETUP and CLEANUP. Specify the type to narrow the scope of the command to a single script. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask exportMiddlewareAppScript {-app sample -dir /home/user1}
```
- Using Jython string:

```
AdminTask.exportMiddlewareAppScript ('[-app sample -dir /home/user1]')
```
- Using Jython list:

```
AdminTask.exportMiddlewareAppScript (['-app', 'sample', '-dir', '/home/user1'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask exportMiddlewareAppScript {-interactive}
```
- Using Jython string:

```
AdminTask.exportMiddlewareAppScript (['-interactive'])
```
- Using Jython list:

```
AdminTask.exportMiddlewareAppScript (['-interactive'])
```

listMiddlewareAppEditions

The `listMiddlewareAppEditions` command lists all the editions of a middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

Optional parameters

None.

Return value

The command returns a list of edition names.

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareAppEditions {-app sample}`
- Using Jython string:
`AdminTask.listMiddlewareAppEditions ('[-app sample]')`
- Using Jython list:
`AdminTask.listMiddlewareAppEditions (['-app', 'sample'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask listMiddlewareAppEditions {-interactive}`
- Using Jython string:
`AdminTask.listMiddlewareAppEditions ('[-interactive]')`
- Using Jython list:
`AdminTask.listMiddlewareAppEditions (['-interactive'])`

listMiddlewareApps

The **listMiddlewareApps** command list all of the middleware applications.

Target object

None.

Required parameters

- **-app**
Specifies the name of the middleware application. (String, required)

Optional parameters

None.

Return value

The command returns a list of the middleware applications.

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareApps {-app sample}`
- Using Jython string:
`AdminTask.listMiddlewareApps ('[-app sample]')`
- Using Jython list:
`AdminTask.listMiddlewareApps (['-app', 'sample'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask listMiddlewareApps {-interactive}`
- Using Jython string:
`AdminTask.listMiddlewareApps ('[-interactive]')`
- Using Jython list:
`AdminTask.listMiddlewareApps (['-interactive'])`

listMiddlewareAppWebModules

The `listMiddlewareAppWebModules` command lists the Web modules of a middleware application.

Target object

None.

Required parameters

- app**
Specifies the name of the middleware application. (String, required)
- edition**
Specifies the name of the edition. (String, required)

Optional parameters

None.

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareAppWebModules {-app sample -edition 1}`
- Using Jython string:
`AdminTask.listMiddlewareAppWebModules ('[-app sample -edition 1]')`
- Using Jython list:
`AdminTask.listMiddlewareAppWebModules (['-app', 'sample', '-edition', '1'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask listMiddlewareAppWebModules {-interactive}`
- Using Jython string:
`AdminTask.listMiddlewareAppWebModules ('[-interactive]')`
- Using Jython list:
`AdminTask.listMiddlewareAppWebModules (['-interactive'])`

listMiddlewareTargets

The `listMiddlewareTargets` command lists the deployment targets for a middleware application.

Target object

None.

Required parameters

-app
Specifies the name of the middleware application. (String, required)

Optional parameters

-edition
Specifies the name of the edition. (String, optional)

-module
Specifies the name of the module. (String, optional)

Return value

The command returns a list of deployment target objects.

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareTargets {-app sample}`
- Using Jython string:
`AdminTask.listMiddlewareTargets ('[-app sample]')`
- Using Jython list:
`AdminTask.listMiddlewareTargets (['-app', 'sample'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask listMiddlewareTargets {-interactive}`
- Using Jython string:
`AdminTask.listMiddlewareTargets (['-interactive'])`
- Using Jython list:
`AdminTask.listMiddlewareTargets (['-interactive'])`

modifyMiddlewareAppWebModule

The **modifyMiddlewareAppWebModule** command modifies the virtual hosts of an application.

Target object

None.

Required parameters

-app
Specifies the name of the application. (String, required)

Optional parameters

-edition
Specifies the name of the edition. (String, optional)

-renameEdition
Specifies the new name of the edition. (String, optional)

-description
Specifies how the edition is modified. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask modifyMiddlewareAppWebModule {-app sample -edition 1 -renameEdition 2 -description Update Edition}
```
- Using Jython string:

```
AdminTask.modifyMiddlewareAppWebModule ('[-app sample -edition 1 -renameEdition 2 -description Update Edition]')
```
- Using Jython list:

```
AdminTask.modifyMiddlewareAppWebModule (['-app', 'sample', '-edition', '1', '-renameEdition', '2', '-description', 'Update Edition'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask modifyMiddlewareAppWebModule {-interactive}
```
- Using Jython string:

```
AdminTask.modifyMiddlewareAppWebModule ('[-interactive]')
```
- Using Jython list:

```
AdminTask.modifyMiddlewareAppWebModule (['-interactive'])
```

modifyPHPApp

The **modifyPHPApp** command updates a PHP application.

Target object

None.

Required parameters

- app**
Specifies the name of the PHP application. (String, required)

Optional parameters

- edition**
Specifies the edition of the PHP application. (String, optional)
- renameEdition**
Specifies the new name of the edition. (String, optional)
- description**
Specifies the description of the modified edition. (String, optional)
- archive**
Specifies the path to the application archive file. (String, optional)
- setupScript**
Specifies the path to the setup script file. (String, optional)
- cleanupScript**
Specifies the path to the cleanup script file. (String, optional)
- contextRoot**
Specifies the path prefix that is associated with the application. (String, optional)
- virtualHost**
Specifies the name of the virtual host. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask modifyPHPApp {-app myPHPApplication -edition 1 -renameEdition 1.0.0}`
- Using Jython string:
`AdminTask.modifyPHPApp ('[-app myPHPApplication -edition 1 -renameEdition 1.0.0]')`
- Using Jython list:
`AdminTask.modifyPHPApp (['-app', 'myPHPApplication', '-edition', '1', '-renameEdition', '1.0.0'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask modifyPHPApp {-interactive}`
- Using Jython string:
`AdminTask.modifyPHPApp (['-interactive'])`
- Using Jython list:
`AdminTask.modifyPHPApp (['-interactive'])`

modifyUnmanagedWebApp

The `modifyUnmanagedWebApp` command updates an unmanaged Web application.

Target object

None.

Required parameters

- app**
Specifies the name of the application. (String, required)

Optional parameters

- edition**
Specifies the name of the edition. (String, optional)
- renameEdition**
Specifies the new name of the edition. (String, optional)
- description**
Specifies the description of the modified edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask modifyUnmanagedWebApp {-app sample -edition 1 -renameEdition 2 -description Update Edition}`
- Using Jython string:
`AdminTask.modifyUnmanagedWebApp ('[-app sample -edition 1 -renameEdition 2 -description Update Edition]')`
- Using Jython list:
`AdminTask.modifyUnmanagedWebApp (['-app', 'sample', '-edition', '1', '-renameEdition', '2', '-description', 'Update Edition'])`

Interactive mode example usage

- Using Jacl:

```
$AdminTask modifyUnmanagedWebApp {-interactive}
```

- Using Jython string:

```
AdminTask.modifyUnmanagedWebApp ('[-interactive]')
```

- Using Jython list:

```
AdminTask.modifyUnmanagedWebApp (['-interactive'])
```

modifyWasCEApp

The **modifyWasCEApp** command updates a WebSphere Application Server Community Edition application.

Target object

None.

Required parameters

-app

Specifies the name of the application. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

-renameEdition

Specifies the new name of the edition. (String, optional)

-description

Specifies the description of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask installWasCEApp {-app sample -edition 1 -renameEdition 2 -description Update Edition}
```

- Using Jython string:

```
AdminTask.modifyWasCEApp ('[-app sample-edition 1 -renameEdition 2 -description Update Edition]')
```

- Using Jython list:

```
AdminTask.modifyWasCEApp (['-app', 'sample', '-edition', '1', 'renameEdition', '2', '-description', 'Update Edition'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask modifyWasCEApp {-interactive}
```

- Using Jython string:

```
AdminTask.modifyWasCEApp ('[-interactive]')
```

- Using Jython list:

```
AdminTask.modifyWasCEApp (['-interactive'])
```

removeMiddlewareAppWebModule

The **removeMiddlewareAppWebModule** command removes the Web module from the middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

-moduleName

Specifies the name of the Web module. (String, optional)

-cluster

Specifies the name of the cluster. (String, optional)

-node

Specifies the name of the node. (String, optional)

-server

Specifies the name of the server. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask removeMiddlewareAppWebModule {-app sample -edition 1 -moduleName myModule}
```

- Using Jython string:

```
AdminTask.removeMiddlewareAppWebModule ('[-app sample -edition 1 -moduleName myModule]')
```

- Using Jython list:

```
AdminTask.removeMiddlewareAppWebModule (['-app', 'sample', '-edition', '1', '-moduleName', 'myModule'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask removeMiddlewareAppWebModule {-interactive}
```

- Using Jython string:

```
AdminTask.removeMiddlewareAppWebModule ('[-interactive]')
```

- Using Jython list:

```
AdminTask.removeMiddlewareAppWebModule (['-interactive'])
```

removeMiddlewareTarget

The **removeMiddlewareTarget** command removes the deployment targets from the middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

Optional parameters

- edition**
Specifies the name of the edition. (String, optional)
- module**
Specifies the name of the Web module. (String, optional)
- cluster**
Specifies the name of the cluster. (String, optional)
- node**
Specifies the name of the node. (String, optional)
- server**
Specifies the name of the server. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask removeMiddlewareTarget {-app sample -cluster cluster1}`
- Using Jython string:
`AdminTask.removeMiddlewareTarget ('[-app sample -cluster cluster1]')`
- Using Jython list:
`AdminTask.removeMiddlewareTarget (['-app', 'sample', '-cluster', 'cluster1'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask removeMiddlewareTarget {-interactive}`
- Using Jython string:
`AdminTask.removeMiddlewareTarget ('[-interactive]')`
- Using Jython list:
`AdminTask.removeMiddlewareTarget (['-interactive'])`

showMiddlewareApp

The **showMiddlewareApp** command displays the attributes of the middleware application.

Target object

None.

Required parameters

- app**
Specifies the name of the application. (String, required)

Optional parameters

- edition**
Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask showMiddlewareApp {-app sample}`

- Using Jython string:
AdminTask.showMiddlewareApp ('[-app sample]')
- Using Jython list:
AdminTask.showMiddlewareApp (['-app', 'sample'])

Interactive mode example usage

- Using Jacl:
\$AdminTask showMiddlewareApp {-interactive}
- Using Jython string:
AdminTask.showMiddlewareApp ('[-interactive]')
- Using Jython list:
AdminTask.showMiddlewareApp (['-interactive'])

installWasCEApp

The **installWasCEApp** command installs a WebSphere Application Server Community Edition application.

Target object

None.

Required parameters

- app**
Specifies the name of the application. (String, required)
- archive**
Specifies the location of the archive file. (String, required)

Optional parameters

- edition**
Specifies the name of the edition. (String, optional)
- description**
Specifies the description of the edition.
- plan**
Specifies the location of the external deployment plan.
- webModules: [[moduleName1 contextRoot1 virtualHost1][moduleName2 contextRoot2 virtualHost2]...]**
Specifies the Web modules. Each **moduleName** value must match the name that is contained in the deployment descriptor file.
- clusterTargets**
Specifies the cluster targets. Apply this parameter to the entire application.
- serverTargets**
Specifies the server targets. Apply this parameter to the entire application.

Return value

Batch mode example usage

- Using Jacl:
\$AdminTask installWasCEApp {-app sample -archive /tmp/sample.ear -edition 1 -serverTargets {{myserver01 WASCE_2.0.0.2_myserver01_DS_1}} -webModules {{samplemicrowebapp.war /B default_host}}
- Using Jython string:

```
AdminTask.installWasCEApp ('[-app sample -archive /tmp/sample.ear -edition 1 -serverTargets
[[myserver01 WASCE_2.0.0.2_myserver01_DS_1]] -webModules [[samplemicrowebapp.war /B default_host}}]')
```

- **Using Jython list:**

```
AdminTask.installWasCEApp (['-app', 'sample', '-archive', '/tmp/sample.ear', '-edition', '1',
'-serverTargets', '[[myserver01 WASCE_2.0.0.2_myserver01_DS_1]]', '-webModules',
'[[samplemicrowebapp.war /B default_host]]'])
```

Interactive mode example usage

- **Using Jacl:**

```
$AdminTask installWasCEApp {-interactive}
```

- **Using Jython string:**

```
AdminTask.installWasCEApp ('[-interactive]')
```

- **Using Jython list:**

```
AdminTask.installWasCEApp (['-interactive'])
```

startWasCEApp

The **startWasCEApp** command starts a WebSphere Application Server Community Edition application.

Target object

None.

Required parameters

-app

Specifies the name of the application. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- **Using Jacl:**

```
$AdminTask startWasCEApp {-app sample -edition 1}
```

- **Using Jython string:**

```
AdminTask.startWasCEApp ('[-app sample -edition 1]')
```

- **Using Jython list:**

```
AdminTask.startWasCEApp (['-app', 'sample', '-edition', '1'])
```

Interactive mode example usage

- **Using Jacl:**

```
$AdminTask startWasCEApp {-interactive}
```

- **Using Jython string:**

```
AdminTask.startWasCEApp ('[-interactive]')
```

- **Using Jython list:**

```
AdminTask.startWasCEApp (['-interactive'])
```

stopWasCEApp

The **stopWasCEApp** command stops a WebSphere Application Server Community Edition application.

Target object

None.

Required parameters

-app
Specifies the name of the application. (String, required)

Optional parameters

-edition
Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask stopWasCEApp {-app sample -edition 1}`
- Using Jython string:
`AdminTask.stopWasCEApp ('[-app sample -edition 1]')`
- Using Jython list:
`AdminTask.stopWasCEApp (['-app', 'sample', '-edition', '1'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask stopWasCEApp {-interactive}`
- Using Jython string:
`AdminTask.stopWasCEApp ('[-interactive]')`
- Using Jython list:
`AdminTask.stopWasCEApp (['-interactive'])`

undeployWasCEApp

The **undeployWasCEApp** command removes a representation of a WebSphere Application Server Community Edition application.

Target object

None.

Required parameters

-app
Specifies the name of the application. (String, required)

-node
Specifies the name of the node. (String, required)

-server
Specifies the name of the server. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask undeployWasCEApp {-app sample -edition 2 -node nodeName -server WasCEServerRep}`
- Using Jython string:
`AdminTask.undeployWasCEApp ('[-app newSample -edition 2 -node nodeName -server WasCEServerRep]')`
- Using Jython list:
`AdminTask.undeployWasCEApp (['-app', 'newSample', '-edition', '2', '-node', 'nodeName', '-server', 'WasCEServerRep'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask undeployWasCEApp {-interactive}`
- Using Jython string:
`AdminTask.undeployWasCEApp ('[-interactive]')`
- Using Jython list:
`AdminTask.undeployWasCEApp (['-interactive'])`

uninstallMiddlewareApp

The **uninstallMiddlewareApp** command uninstalls the middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the application. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask uninstallMiddlewareApp {-app sample -edition 1}`
- Using Jython string:
`AdminTask.uninstallMiddlewareApp ('[-app sample -edition 1]')`
- Using Jython list:
`AdminTask.uninstallMiddlewareApp (['-app', 'sample', '-edition', '1'])`

Interactive mode example usage

- Using Jacl:

```
$AdminTask uninstallMiddlewareApp {-interactive}
```

- Using Jython string:

```
AdminTask.uninstallMiddlewareApp ('[-interactive]')
```

- Using Jython list:

```
AdminTask.uninstallMiddlewareApp (['-interactive'])
```

unregisterApp

The **unregisterApp** command removes an unmanaged middleware application.

Target object

None.

Required parameters

-app

Specifies the name of the middleware application. (String, required)

Optional parameters

-edition

Specifies the name of the edition. (String, optional)

Return value

Batch mode example usage

- Using Jacl:

```
$AdminTask unregisterApp {-app sample -edition 1}
```

- Using Jython string:

```
AdminTask.unregisterApp ('-app sample -edition 1')
```

- Using Jython list:

```
AdminTask.unregisterApp (['-app', 'sample', '-edition', '1'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask unregisterApp {-interactive}
```

- Using Jython string:

```
AdminTask.unregisterApp ('[-interactive]')
```

- Using Jython list:

```
AdminTask.unregisterApp (['-interactive'])
```

Middleware descriptor administrative tasks

You can use the middleware descriptor administrative tasks to list the middleware descriptors and versions, display or change middleware descriptor information, and modify the middleware descriptor discovery interval.

Use the following commands to manage the middleware descriptors in your environment:

- “listMiddlewareDescriptors” on page 390
- “listMiddlewareDescriptorVersions” on page 390
- “showMiddlewareDescriptorInformation” on page 391
- “modifyMiddlewareDescriptorProperty ” on page 391
- “modifyMiddlewareDescriptorDiscoveryInterval” on page 392

listMiddlewareDescriptors

The `listMiddlewareDescriptors` command lists the installed middleware descriptors in your cell.

Required parameters

None

Return value

The command returns a list of the middleware descriptors, for example:

```
jboss_server
application_server
phpRuntime
customhttp_server
wasceRuntime
apacheWebServerRuntime
apache_server
weblogic_server
tomcat_server
```

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareDescriptors`
- Using Jython:
`AdminTask.listMiddlewareDescriptors()`

listMiddlewareDescriptorVersions

The `listMiddlewareDescriptorVersions` command lists which versions have specific information provided in the middleware descriptors.

Required parameters

- **-name**: Specifies the name of the middleware descriptor. (String, required)

Return value

A list of versions that have specific information provided in the middleware descriptor. For example:

```
default
```

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareDescriptorVersions {-name wasceRuntime}`
- Using Jython:
`AdminTask.listMiddlewareDescriptorVersions ('[-name wasceRuntime]')`

Interactive mode example usage

- Using Jacl:
`$AdminTask listMiddlewareDescriptorVersions {-interactive}`
- Using Jython:
`AdminTask.listMiddlewareDescriptorVersions ([-interactive]')`

showMiddlewareDescriptorInformation

The **showMiddlewareDescriptorInformation** command displays the contents of the specified middleware descriptor.

Required parameters

- **-name**: Specifies the name of the middleware descriptor. (String, required)

Return value

An example of the command output follows:

```
wasceRuntime:
  Discovery Interval: 4 min

  Version: default
    install.locations:win.install.loc="C:\\Program
Files\\IBM\\WebSphere\\AppServerCommunityEdition;C:\\Program Files
(x86)\\WebSphere\\AppServerCommunityEdition"

install.locations:unix.install.loc="/opt/IBM/WebSphere/AppServerCommunityEdition"

install.locations:foreign.discovery.class="com.ibm.ws.xd.agent.discovery.wasce.WASCEDiscoveryPlugin"
  timeOutGroup:startTimeoutValue="300000"
  timeOutGroup:stopTimeoutValue="300000"
```

Batch mode example usage

- Using Jacl:
\$AdminTask showMiddlewareDescriptorInformation {-name wasceRuntime}
- Using Jython:
AdminTask.showMiddlewareDescriptorInformation ('[-name wasceRuntime]')

Interactive mode example usage

- Using Jacl:
\$AdminTask showMiddlewareDescriptorInformation {-interactive}
- Using Jython:
AdminTask.showMiddlewareDescriptorInformation ([-interactive]')

modifyMiddlewareDescriptorProperty

You can use the **modifyMiddlewareDescriptorProperty** command to edit the middleware descriptor properties.

Required parameters

- **-name**: Specifies the name of the middleware descriptor.
- **-version**: Specifies the middleware descriptor version.
- **-propName**: Specifies a property for the middleware descriptor. This property name can be any property that displays when you run the **showMiddlewareDescriptorInformation** command for the middleware descriptor.
- **-propValue**: Specifies a value for the middleware descriptor property.

Return value

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask modifyMiddlewareDescriptorProperty {-name wasceRuntime
-version default -propName install.locations:win.install.loc -propValue
c:\blade\server1 }
```

- Using Jython:

```
AdminTask.modifyMiddlewareDescriptorProperty ('[-name wasceRuntime -version default -propName
install.locations:win.install.loc -propValue
c:\blade\server1]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask modifyMiddlewareDescriptorProperty {-interactive}
```

- Using Jython:

```
AdminTask.modifyMiddlewareDescriptorProperty ([-interactive]')
```

modifyMiddlewareDescriptorDiscoveryInterval

You can use the **modifyMiddlewareDescriptorDiscoveryInterval** command to modify the discovery interval of the selected middleware descriptor.

Required parameters

- **-name**: Specifies the name of the middleware descriptor.
- **-interval**: Specifies the integer value of the discovery interval. You can use one of the following values:
 - -1: Disables automatic discovery.
 - 0 : Performs middleware discovery when the node agent starts.
 - value greater than 0: Specifies that middleware discovery runs when the node agent starts and at the integer value interval.
- **-units**: Specifies the units for the discovery interval. You can use one of the following values:
 - seconds
 - minutes
 - hours
 - days

Return value

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask modifyMiddlewareDescriptorDiscoveryInterval {-name wasceRuntime
-interval 5 -units seconds}
```

- Using Jython:

```
AdminTask.modifyMiddlewareDescriptorDiscoveryInterval ('[-name wasceRuntime -interval 5 -units seconds]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask modifyMiddlewareDescriptorProperty {-interactive}
```

- Using Jython:

```
AdminTask.modifyMiddlewareDescriptorProperty ([-interactive]')
```


Middleware server creation administrative tasks

You can use administrative tasks to write a script that can recreate your middleware server configuration.

Use the following commands to create middleware servers:

- “createTomCatServer”
- “createWebLogicServer” on page 394
- “createJBossServer” on page 395
- “createForeignServer” on page 396
- “createWasCEServer” on page 397

createTomCatServer

The `createTomCatServer` command creates a representation of an Apache Tomcat server.

Target object

The node name.

Required parameters

-name

Specifies the name of the server to create. (String, required)

Optional parameters

-templateName

Specifies the name of the template that is used to create the server.

-genUniquePorts

Specifies a parameter to generate unique HTTP ports for a server. (Boolean)

-templateLocation

Specifies the location where the template is stored. Use the system defined location if it is not specified. Using the system defined location is recommended.

z/OS -specificShortName

Specifies the specific short name of the server. All servers should have unique specific short name. This parameter is optional and when it is not specified a unique specific short name is automatically assigned. The value must be eight characters or less and all upper case.

z/OS -genericShortName

Specifies the generic short name of the server. All members of a cluster must have the same generic short name. Individual servers must have a unique generic short name. This parameter is optional. When this parameter is not specified, a unique generic short name is automatically assigned. The value must be eight characters or less and all upper case.

-clusterName

Specifies the name of the cluster for this server.

Return value

The command returns a list of the middleware servers for the specified type.

Batch mode example usage

- Using Jacl:
\$AdminTask createTomCatServer nodename {-name ServerName}
- Using Jython:

```
AdminTask.createTomCatServer ('nodename', ['-name ServerName'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createTomCatServer {-interactive}
```
- Using Jython:

```
AdminTask.createTomCatServer ([-interactive])
```

createWebLogicServer

The **createWebLogicServer** command creates a representation of a BEA WebLogic Server.

Target object

The node name.

Required parameters

- name**
Specifies the name of the server to create. (String, required)

Optional parameters

- templateName**
Specifies the name of the template that is used to create the server.
- genUniquePorts**
Specifies a parameter to generate unique HTTP ports for a server. (Boolean)
- templateLocation**
Specifies the location where the template is stored. Use the system defined location if it is not specified. Using the system defined location is recommended.
- z/OS -specificShortName**
Specifies the specific short name of the server. All servers should have unique specific short name. This parameter is optional and when it is not specified a unique specific short name is automatically assigned. The value must be eight characters or less and all upper case.
- z/OS -genericShortName**
Specifies the generic short name of the server. All members of a cluster must have the same generic short name. Individual servers must have a unique generic short name. This parameter is optional. When this parameter is not specified, a unique generic short name is automatically assigned. The value must be eight characters or less and all upper case.
- clusterName**
Specifies the name of the cluster for this server.

Return value

The command returns a list of the middleware servers for the specified type.

Batch mode example usage

- Using Jacl:

```
$AdminTask createWebLogicServer nodename {-name ServerName}
```
- Using Jython:

```
AdminTask.createWebLogicServer ('nodename', ['-name ServerName'])
```

Interactive mode example usage

- Using Jacl:
\$AdminTask createWebLogicServer {-interactive}
- Using Jython:
AdminTask.createWebLogicServer ([-interactive]')

createJBossServer

The **createJBossServer** command creates a representation of a JBoss server.

Target object

The node name.

Required parameters

- name**
Specifies the name of the server to create. (String, required)

Optional parameters

- templateName**
Specifies the name of the template that is used to create the server.
- genUniquePorts**
Specifies a parameter to generate unique HTTP ports for a server. (Boolean)
- templateLocation**
Specifies the location where the template is stored. Use the system defined location if it is not specified. Using the system defined location is recommended.

z/OS -specificShortName

Specifies the specific short name of the server. All servers should have unique specific short name. This parameter is optional and when it is not specified a unique specific short name is automatically assigned. The value must be eight characters or less and all upper case.

z/OS -genericShortName

Specifies the generic short name of the server. All members of a cluster must have the same generic short name. Individual servers must have a unique generic short name. This parameter is optional. When this parameter is not specified, a unique generic short name is automatically assigned. The value must be eight characters or less and all upper case.

- clusterName**
Specifies the name of the cluster for this server.

Return value

Batch mode example usage

- Using Jacl:
\$AdminTask createJBossServer nodename {-name ServerName}
- Using Jython:
AdminTask.createJBossServer ('nodename', '[-name ServerName]')

Interactive mode example usage

- Using Jacl:
\$AdminTask createJBossServer (-interactive)
- Using Jython:
AdminTask.createJBossServer ([-interactive]')

createForeignServer

The **createForeignServer** command creates a new foreign server.

Target object

The node name.

Required parameters

-name

Specifies the name of the server to create. (String, required)

Optional parameters

-templateName

Specifies the name of the template that is used to create the server.

-genUniquePorts

Specifies a parameter to generate unique HTTP ports for a server. (Boolean)

-templateLocation

Specifies the location where the template is stored. Use the system defined location if it is not specified. Using the system defined location is recommended.

z/OS -specificShortName

Specifies the specific short name of the server. All servers should have unique specific short name. This parameter is optional and when it is not specified a unique specific short name is automatically assigned. The value must be eight characters or less and all upper case.

z/OS -genericShortName

Specifies the generic short name of the server. All members of a cluster must have the same generic short name. Individual servers must have a unique generic short name. This parameter is optional. When this parameter is not specified, a unique generic short name is automatically assigned. The value must be eight characters or less and all upper case.

-clusterName

Specifies the name of the cluster for this server.

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask createForeignServer nodename {-name ServerName}`
- Using Jython:
`AdminTask.createForeignServer ('nodename', '[-name ServerName]')`

Interactive mode example usage

- Using Jacl:
`$AdminTask createForeignServer {-interactive}`
- Using Jython:
`AdminTask.createForeignServer ([-interactive]')`

createPHPServer

The **createPHPServer** command creates a new PHP server.

Required parameters

-name
Specifies the name of the server. (String, required)

Optional parameters

-templateName
Specifies the name of the server template to use. You can specify `templateServerNode` and `templateServerName` to use an existing server as a virtual template instead of a defined template. (String, optional)

-genUniquePorts
Specifies a boolean to generate unique ports for the server. (String, optional)

-templateLocation
Specifies the location in which the template is stored. Use the system defined location if the location is not specified. (String, optional)

-serverConfig
Specifies the configuration of the server definition properties. Specify `phpServerRoot` or `apacheServerRoot` for the runtime path. Specify `phpVersion` or `apacheVersion` for the version number. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask createPHPServer workstation_nameNode01 {-name myphpserver -serverConfig {-phpPath C:/PHP5 -apachePath C:/apache2}}
```
- Using Jython:

```
AdminTask.createPHPServer workstation_nameNode01 ('[-name myphpserver -serverConfig [-phpPath C:/PHP5 -apachePath C:/apache2]]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createPHPServer workstation_nameNode01 {-interactive}
```
- Using Jython:

```
AdminTask.createPHPServer workstation_nameNode01 ('[-interactive]')
```

createWasCEServer

The **createWasCEServer** command creates a representation of a WebSphere Application Server Community Edition server.

Target object

None.

Required parameters

-name
Specifies the name of the server to create. (String, required)

Optional parameters

-templateName
Specifies the name of the template that is used to create the server.

-genUniquePorts
Specifies a parameter to generate unique HTTP ports for a server. (Boolean)

-templateLocation
Specifies the location where the template is stored. Use the system defined location if it is not specified. Using the system defined location is recommended.

z/OS -specificShortName

Specifies the specific short name of the server. All servers should have unique specific short name. This parameter is optional and when it is not specified a unique specific short name is automatically assigned. The value must be eight characters or less and all upper case.

z/OS -genericShortName

Specifies the generic short name of the server. All members of a cluster must have the same generic short name. Individual servers must have a unique generic short name. This parameter is optional. When this parameter is not specified, a unique generic short name is automatically assigned. The value must be eight characters or less and all upper case.

-clusterName

Specifies the name of the cluster for this server.

-assistedServer

Specifies if the server that you create is an assisted life cycle server that is a representation of an existing server. The default is `false`. If you specify `false`, a complete life cycle server is created. If you specify `true`, an assisted life cycle server is created.

-templateServerNode

Specifies the name of the node that contains the server that you want to use as a template.

-templateServerName

Specifies the name of the existing server that you want to use as a template.

Return value

Batch mode example usage

- Using Jacl:
`$AdminTask createWasCEServer nodename {-name ServerName -templateName wasce21}`
- Using Jython:
`AdminTask.createWasCEServer ('nodename', ['-name ServerName -templateName wasce21'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask createWasCEServer {-interactive}`
- Using Jython:
`AdminTask.createWasCEServer ([-interactive]')`

Middleware server management administrative tasks

You can use the Jython and Jacl scripting languages to manage middleware servers with the `wsadmin` tool. Use the commands and parameters in the **MiddlewareServerManagement** group.

Use the following commands to work with middleware servers:

- `getMiddlewareServerType`
- `listForeignServerTypes`
- `listMiddlewareServers`
- `listMiddlewareServerTypes`
- `listWASServerTypes`
- `modifyForeignServerProperty`
- `showMiddlewareServerInfo`
- `startMiddlewareServer`
- `stopMiddlewareServer`
- `executeMiddlewareServerOperation`

getMiddlewareServerType

The `getMiddlewareServerType` command lists the middleware server types.

Target object

None.

Required parameters

-serverName

Specifies the name of the server for which you want to display the type. (String, required)

-nodeName

Specifies the node on which the server runs. (String, required)

Return value

The command returns a list of the middleware servers for the specified type.

Batch mode example usage

- Using Jacl:
`$AdminTask getMiddlewareServerType {-serverName myserver -nodeName mynode}`
- Using Jython string:
`AdminTask.getMiddlewareServerType('[-serverName myserver -nodeName mynode]')`

Interactive mode example usage

- Using Jacl:
`$AdminTask stopMiddlewareServer {-interactive}`
- Using Jython string:
`AdminTask.stopMiddlewareServer ('[-interactive]')`

listForeignServerTypes

The `listForeignServerTypes` command lists all of the supported middleware server types, not including any middleware servers associated with Intelligent Management.

Target object

Return value

The command returns a list of middleware server types, for example:

```
PHP_SERVER
WASCE_SERVER
CUSTOMHTTP_SERVER
APACHE_SERVER
TOMCAT_SERVER
WEBLOGIC_SERVER
JBOSS_SERVER
WASAPP_SERVER
```

Batch mode example usage

- Using Jacl:
`$AdminTask listForeignServerTypes`
- Using Jython string:

```
AdminTask.listForeignServerTypes()
```

listMiddlewareServers

The `listMiddlewareServers` command lists the middleware servers and the path to the configuration file for the servers. If you do not pass node or middleware server type parameters, then all of the servers are displayed.

Target object

None.

Optional parameters

-middlewareServerType

Specifies the type of middleware server to list. The middleware server types include: TOMCAT_SERVER, PHP_SERVER, WEBLOGIC_SERVER, JBOSS_SERVER, WASCE_SERVER, APACHE_SERVER, and CUSTOMHTTP_SERVER. (String, optional)

-nodeName

Specifies the node for which to list the servers. (String, optional)

Return value

The command returns the list of the middleware servers for the type that you indicated. For example, you might see the following output:

```
nodeagent(cells/xdblade02b07/nodes/xdblade01b08/servers/nodeagent|server.xml)
WASCE_2.0.0.0_xdblade01b08_DS_2(cells/xdblade02b07/nodes/xdblade01b08/servers/WASCE_2.0.0.0_xdblade01b08_DS_2|server.xml)
WASCE_2.0.0.0_xdblade01b08_DS_1(cells/xdblade02b07/nodes/xdblade01b08/servers/WASCE_2.0.0.0_xdblade01b08_DS_1|server.xml)
WASCE_2.0.0.0_xdblade01b08_DS_4(cells/xdblade02b07/nodes/xdblade01b08/servers/WASCE_2.0.0.0_xdblade01b08_DS_4|server.xml)
frucg(cells/xdblade02b07/nodes/xdblade01b08/servers/frucg|server.xml)
WASCE_2.0.0.0_xdblade01b08_DD_1(cells/xdblade02b07/nodes/xdblade01b08/servers/WASCE_2.0.0.0_xdblade01b08_DD_1|server.xml)
dmgr(cells/xdblade02b07/nodes/xdblade02b07/servers/dmgr|server.xml)
middlewareagent(cells/xdblade02b07/nodes/thebe14/servers/middlewareagent|server.xml)
tttomcat_server(cells/xdblade02b07/nodes/thebe14/servers/tttomcat_server|server.xml)
```

Batch mode example usage

- Using Jacl:

```
$AdminTask listMiddlewareServers {-middlewareServerType TOMCAT_SERVER}
```
- Using Jython string:

```
AdminTask.listMiddlewareServers('[-middlewareServerType TOMCAT_SERVER]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask listMiddlewareServers
```
- Using Jython string:

```
AdminTask.listMiddlewareServers()
```

listMiddlewareServerTypes

The `listMiddlewareServerTypes` command lists all of the server types.

Target object

Return value

The command returns a list of all of the server types, for example:

APPLICATION_SERVER
PHP_SERVER
WASCE_SERVER
CUSTOMHTTP_SERVER
APACHE_SERVER
TOMCAT_SERVER
WEBLOGIC_SERVER
ONDEMAND_ROUTER
PROXY_SERVER
WEB_SERVER
JBOSS_SERVER
WASAPP_SERVER
GENERIC_SERVER

Batch mode example usage

- Using Jacl:
`$AdminTask listMiddlewareServerTypes`
- Using Jython string:
`AdminTask.listMiddlewareServerTypes`

Interactive mode example usage

listWASServerTypes

The **listWASServerTypes** command lists only the server types that are associated with WebSphere Application Server.

Target object

Return value

The command returns a list of server types, for example:

APPLICATION_SERVER
ONDEMAND_ROUTER
PROXY_SERVER
WEB_SERVER
GENERIC_SERVER

Batch mode example usage

- Using Jacl:
`$AdminTask listWASServerTypes`
- Using Jython string:
`AdminTask.listWASServerTypes()`

modifyForeignServerProperty

The **modifyForeignServerProperty** command modifies a property on a middleware server.

Target object

None.

Required parameters

- **-serverName**
Specifies the name of the server. (String, required)

-nodeName

Specifies the name of the node. (String, required)

-propKey

Specifies the property key that is associated with the server.xml file. (String, required)

-propValue

Specifies the property value that you want to set in the server.xml file. (String, required)

Return value

The command returns

Batch mode example usage

- Using Jacl:

```
$AdminTask modifyForeignServerProperty {-serverName WASCE_2.0.0.0_xdblade01b08_DS_4 -nodeName xdblade01b08
-propKey port -propValue 9090 }
```

- Using Jython string:

```
AdminTask.modifyForeignServerProperty(['-serverName WASCE_2.0.0.0_xdblade01b08_DS_4 -nodeName xdblade01b08
-propKey port -propValue 9090'])
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask modifyForeignServerProperty {-interactive}
```

- Using Jython string:

```
AdminTask.modifyForeignServerProperty(['-interactive'])
```

showMiddlewareServerInfo

The **showMiddlewareServerInfo** command displays a list of properties for the middleware server.

Return value

The command returns

Required parameters

server_name(path_to_configuration_file|configuration_file_name)

This parameter determines the server for which you want to display the information. For example, you might use the following value for this parameter:

```
odr2(cells/SVT61/nodes/oberon04/servers/odr2|server.xml)
```

You can determine the value of the *path_to_configuration_file* variable and the *configuration_file_name* value by running the **listMiddlewareServers** command. You can use the output of that command as the parameter value.

Batch mode example usage

- Using Jacl:

```
$AdminTask showMiddlewareServerInfo odr2(cells/SVT61/nodes/oberon04/servers/odr2|server.xml)
```

- Using Jython string:

```
AdminTask.showMiddlewareServerInfo('odr2(cells/SVT61/nodes/oberon04/servers/odr2|server.xml)')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask showMiddlewareServerInfo {-interactive}
```

- Using Jython string:

```
AdminTask.showMiddlewareServerInfo('[-interactive]')
```

Example output

```
{cell SVT61}
{serverMiddlewareType ONDEMAND_ROUTER}
{com.ibm.websphere.baseProductVersion 6.1.0.11}
{node oberon04}
{server odr2}
```

startMiddlewareServer

The **startMiddlewareServer** command starts a specified middleware server.

Target object

None.

Required parameters

-serverName

Specifies the name of the server to start. (String, required)

-nodeName

Specifies the node on which the server that you want to start runs. (String, required)

Return value

The command returns

Batch mode example usage

- Using Jacl:

```
$AdminTask startMiddlewareServer {-serverName myserver -nodeName mynode}
```

- Using Jython string:

```
AdminTask.startMiddlewareServer('[-name myserver -node mynode]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask startMiddlewareServer{-interactive}
```

- Using Jython string:

```
AdminTask.startMiddlewareServer('[-interactive]')
```

stopMiddlewareServer

The **stopMiddlewareServer** command stops the specified middleware server.

Target object

None.

Required parameters

-serverName

Specifies the name of the middleware server to stop. (String, required)

-nodeName

Specifies the node on which the server that you want to stop runs. (String, required)

Batch mode example usage

- Using Jacl:
`$AdminTask stopMiddlewareServer {-serverName myserver -nodeName mynode}`
- Using Jython string:
`AdminTask.stopMiddlewareServer(['-name myserver -node mynode'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask stopMiddlewareServer {-interactive}`
- Using Jython string:
`AdminTask.stopMiddlewareServer (['-interactive'])`

executeMiddlewareServerOperation

The **executeMiddlewareServerOperation** command runs an existing server operation on a specified server.

Target object

None.

Required parameters

- serverName**
Specifies the name of the server on which to run the server operation. (String, required)
- nodeName**
Specifies the name of the node on which the specified server runs. (String, required)
- operation**
Specifies the name of the server operation that you want to run on the server. (String, required)

Return value

The command returns a status message for the server operation, for example:

```
CWMWS0420I: Start completed for middleware server "WASCE_2.0.0.0_xdblade01b08_DS_4"
on node "xdblade01b08"
```

Batch mode example usage

- Using Jacl:
`$AdminTask executeMiddlewareServerOperation {-serverName WASCE_2.0.0.0_xdblade01b08_DS_4 -nodeName xdblade01b08 -operation start }`
- Using Jython string:
`AdminTask.executeMiddlewareServerOperation(['-serverName WASCE_2.0.0.0_xdblade01b08_DS_4 -nodeName xdblade01b08 -operation start'])`

Interactive mode example usage

- Using Jacl:
`$AdminTask executeMiddlewareServerOperation {-interactive}`
- Using Jython string:
`AdminTask.executeMiddlewareServerOperation(['-interactive'])`

Middleware server template administrative tasks

Use middleware server template administrative tasks to create server templates for middleware servers that you have created.

Use the following commands to create middleware server templates:

- “createApacheServerTemplate command”
- “createForeignServerTemplate command”
- “createJBossServerTemplate command” on page 406
- “createTomCatServerTemplate command” on page 407
- “createWasCEServerTemplate command” on page 407
- “createWebLogicServerTemplate command” on page 408

createApacheServerTemplate command

The **createApacheServerTemplate** command creates a new server template that is based on an existing Apache server representation.

Target object

Required parameters

-templateName

Specifies the name of the server template that you want to create. (String, required)

-serverName

Specifies the server from which to base the template. (String, required)

-nodeName

Specifies the node on which the specified server runs. (String, required)

-description

Specifies a description for the template.

-templateLocation

Specifies the location where you want to place the template.

Return value

This command returns the configuration ID of a new template.

Batch mode example usage

- Using Jacl:

```
$AdminTask createApacheServerTemplate{-templateName weblogic_new -serverName xdtest1 -nodeName xdnodel1 -description "My new template"}
```

- Using Jython String:

```
AdminTask.createApacheServerTemplate(['-templateName weblogic_new -serverName xdtest1 -nodeName xdnodel1 -description "My new template"'])
```

- Using Jython List:

```
AdminTask.createApacheServerTemplate(['-templateName','weblogic_new','-serverName','xdtest1','-nodeName','xdnodel1','-description','My new template'])
```

createForeignServerTemplate command

The **createForeignServerTemplate** command creates a new server template that is based on an existing custom HTTP server representation.

Target object

Required parameters

- templateName**
Specifies the name of the server template that you want to create. (String, required)
- serverName**
Specifies the server from which to base the template. (String, required)
- nodeName**
Specifies the node on which the specified server runs. (String, required)
- description**
Specifies a description for the template.
- templateLocation**
Specifies the location where you want to place the template.

Return value

This command returns the configuration ID of a new template.

Batch mode example usage

- Using Jacl:


```
$AdminTask createForeignServerTemplate{-templateName weblogic_new -serverName xdtest1 -nodeName xdnodel
-description "My new template"}
```
- Using Jython String:


```
AdminTask.createForeignServerTemplate(['-templateName weblogic_new -serverName xdtest1 -nodeName xdnodel
-description "My new template"]')
```
- Using Jython List:


```
AdminTask.createForeignServerTemplate(['-templateName','weblogic_new','-serverName','xdtest1','-nodeName',
'xdnodel','-description','My new template'])
```

createJBossServerTemplate command

The **createJBossServerTemplate** command creates a new server template that is based on an existing JBoss server representation.

Target object

Required parameters

- templateName**
Specifies the name of the server template that you want to create. (String, required)
- serverName**
Specifies the server from which to base the template. (String, required)
- nodeName**
Specifies the node on which the specified server runs. (String, required)
- description**
Specifies a description for the template.
- templateLocation**
Specifies the location where you want to place the template.

Return value

This command returns the configuration ID of a new template.

Batch mode example usage

- Using Jacl:

```
$AdminTask createJBossServerTemplate{-templateName jboss_new -serverName xdtest1 -nodeName xdnodel
-description "My new template"}
```

- Using Jython String:

```
AdminTask.createJBossServerTemplate(['-templateName jboss_new -serverName xdtest1 -nodeName xdnodel
-description "My new template"'])
```

- Using Jython List:

```
AdminTask.createJBossServerTemplate(['-templateName','jboss_new','-serverName','xdtest1','-nodeName',
'xdnodel','-description','My new template'])
```

createTomCatServerTemplate command

The **createTomCatServerTemplate** command creates a new server template that is based on an existing Apache Tomcat server representation.

Target object

Required parameters

-templateName

Specifies the name of the server template that you want to create. (String, required)

-serverName

Specifies the server from which to base the template. (String, required)

-nodeName

Specifies the node on which the specified server runs. (String, required)

-description

Specifies a description for the template.

-templateLocation

Specifies the location where you want to place the template.

Return value

This command returns the configuration ID of a new template.

Batch mode example usage

- Using Jacl:

```
$AdminTask createTomCatServerTemplate{-templateName tomcat_new -serverName xdtest1 -nodeName
xdnodel -description "My new template"}
```

- Using Jython String:

```
AdminTask.createTomCatServerTemplate(['-templateName tomcat_new -serverName xdtest1 -nodeName
xdnodel -description "My new template"'])
```

- Using Jython List:

```
AdminTask.createTomCatServerTemplate(['-templateName','tomcat_new','-serverName','xdtest1','-nodeName',
'xdnodel','-description','My new template'])
```

createWasCEServerTemplate command

The **createWasCEServerTemplate** command creates a new server template that is based on an existing WebSphere Application Server Community Edition server representation.

Target object

Required parameters

-templateName

Specifies the name of the server template that you want to create. (String, required)

- serverName**
Specifies the server from which to base the template. (String, required)
- nodeName**
Specifies the node on which the specified server runs. (String, required)
- description**
Specifies a description for the template.
- templateLocation**
Specifies the location where you want to place the template.

Return value

This command returns the configuration ID of a new template.

Batch mode example usage

- Using Jacl:


```
$AdminTask createWasCEServerTemplate{-templateName wasce_new -serverName xdtest1 -nodeName xdnodel -description "My new template"}
```
- Using Jython String:


```
AdminTask.createWasCEServerTemplate(['-templateName wasce_new -serverName xdtest1 -nodeName xdnodel -description "My new template"'])
```
- Using Jython List:


```
AdminTask.createWasCEServerTemplate(['-templateName','wasce_new','-serverName','xdtest1', '-nodeName','xdnodel','-description','My new template'])
```

createWebLogicServerTemplate command

The **createWebLogicServerTemplate** command creates a new server template that is based on an existing BEA WebLogic server representation.

Target object

Required parameters

- templateName**
Specifies the name of the server template that you want to create. (String, required)
- serverName**
Specifies the server from which to base the template. (String, required)
- nodeName**
Specifies the node on which the specified server runs. (String, required)
- description**
Specifies a description for the template.
- templateLocation**
Specifies the location where you want to place the template.

Return value

This command returns the configuration ID of a new template.

Batch mode example usage

- Using Jacl:


```
$AdminTask createWebLogicServerTemplate{-templateName weblogic_new -serverName xdtest1 -nodeName xdnodel -description "My new template"}
```


- Using Jython String:

```
AdminTask.createWebLogicServerTemplate(['-templateName weblogic_new -serverName xdtest1
-nodeName xdnodel -description "My new template"'])
```

- Using Jython List:

```
AdminTask.createWebLogicServerTemplate(['-templateName','weblogic_new','-serverName','xdtest1',
'-nodeName','xdnodel','-description','My new template'])
```

PHP server administrative tasks

Use the following commands to create and configure PHP servers and server templates.

The administrative tasks for creating and configuring PHP servers include the following commands:

- “createPHPServer”
- “listPHPServers” on page 410
- “startServer” on page 410
- “stopServer” on page 411
- “deleteServer” on page 411
- “createPHPServerTemplate” on page 412
- “deleteServerTemplate” on page 412

createPHPServer

The **createPHPServer** command creates a new PHP server.

Required parameters

-name

Specifies the name of the server. (String, required)

Optional parameters

-templateName

Specifies the name of the server template to use. You can specify `templateServerNode` and `templateServerName` to use an existing server as a virtual template instead of a defined template. (String, optional)

-genUniquePorts

Specifies a boolean to generate unique ports for the server. (String, optional)

-templateLocation

Specifies the location in which the template is stored. Use the system defined location if the location is not specified. (String, optional)

-serverConfig

Specifies the configuration of the server definition properties. Specify `phpServerRoot` or `apacheServerRoot` for the runtime path. Specify `phpVersion` or `apacheVersion` for the version number. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask createPHPServer workstation_nameNode01 {-name myphpserver -serverConfig {-phpPath C:/PHP5 -apachePath C:/apache2}}
```

- Using Jython:

```
AdminTask.createPHPServer workstation_nameNode01 ('[-name myphpserver -serverConfig [-phpPath C:/PHP5 -apachePath C:/apache2]]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createPHPServer workstation_nameNode01 {-interactive}
```

- Using Jython:

```
AdminTask.createPHPServer workstation_nameNode01 ('[-interactive]')
```

listPHPServers

The **listPHPServers** command lists all of the PHP servers.

Required parameters

None.

Optional parameters

-nodeName

Specifies the name of the node. Specify this parameter to narrowly scope the command. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask listPHPServers {}
```
- Using Jython:

```
AdminTask.listPHPServers ('[]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask listPHPServers {-interactive}
```
- Using Jython:

```
AdminTask.listPHPServers ('[-interactive]')
```

startServer

The **startServer** command starts a PHP server.

Required parameters

-serverName

Specifies the name of the server to start. (String, required)

-nodeName

Specifies the name of the node on which the server is located. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask startServer {-nodeName workstation_nameNode01 -serverName phpserver}
```
- Using Jython:

```
AdminTask.startServer ('[-nodeName workstation_nameNode01 -serverName phpserver]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask startServer {-interactive}
```

- Using Jython:

```
AdminTask.startServer ('[-interactive]')
```

stopServer

The **stopServer** command stops a PHP server.

Required parameters

-serverName

Specifies the name of the server to stop. (String, required)

-nodeName

Specifies the name of the node on which the server is located. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask stopServer {-nodeName workstation_nameNode01 -serverName phpserver}
```

- Using Jython:

```
AdminTask.stopServer ('[-nodeName workstation_nameNode01 -serverName phpserver]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask stopServer {-interactive}
```

- Using Jython:

```
AdminTask.stopServer ('[-interactive]')
```

deleteServer

The **deleteServer** command deletes a PHP server.

Required parameters

-serverName

Specifies the name of the server to delete. (String, required)

-nodeName

Specifies the name of the node on which the server is located. (String, required)

Optional parameters

None.

Batch mode example usage

- Using Jacl:

```
$AdminTask deleteServer {-nodeName workstation_nameNode01 -serverName phpserver}
```

- Using Jython:

```
AdminTask.deleteServer ('[-nodeName workstation_nameNode01 -serverName phpserver]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask deleteServer {-interactive}
```

- Using Jython:

```
AdminTask.deleteServer ('[-interactive]')
```

createPHPServerTemplate

The `createPHPServerTemplate` command creates a PHP server template.

Required parameters

-templateName

Specifies the name of the template to create. (String, required)

-serverName

Specifies the name of the server to use as a template. (String, required)

-nodeName

Specifies the name of the node on which the server is located. (String, required)

Optional parameters

-description

Specifies a description for the server template. (String, optional)

-templateLocation

Specifies the location in which the template is stored. Use the system defined location if the location is not specified. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask createPHPServerTemplate {-templateName myphptemplate -nodeName workstation_nameNode01 -serverName phpserver}
```

- Using Jython:

```
AdminTask.createPHPServerTemplate ('[-templateName myphptemplate -nodeName workstation_nameNode01 -serverName phpserver]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createPHPServerTemplate {-interactive}
```

- Using Jython:

```
AdminTask.createPHPServerTemplate ('[-interactive]')
```

deleteServerTemplate

The `deleteServerTemplate` command deletes a PHP server template.

Required parameters

-templateName

Specifies the name of the template to delete. (String, required)

-serverName

Specifies the name of the server that uses the template. (String, required)

-nodeName

Specifies the name of the node on which the server is located. (String, required)

Optional parameters

-description

Specifies a description for the server template. (String, optional)

-templateLocation

Specifies the location in which the template is stored. Use the system defined location if the location is not specified. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask deleteServerTemplate {-templateName myphptemplate -nodeName workstation_nameNode01 -serverName phpserver}
```

- Using Jython:

```
AdminTask.deleteServerTemplate ('[-templateName myphptemplate -nodeName workstation_nameNode01 -serverName phpserver]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask deleteServerTemplate {-interactive}
```

- Using Jython:

```
AdminTask.deleteServerTemplate ('[-interactive]')
```

Rules for ODR routing policy administrative tasks

You can use administrative tasks to configure HTTP or Session Initiation Protocol (SIP) rules for the on demand router (ODR) routing policy.

The following rules are the preferred way to configure routing policies, but using the multi-cluster routing property (MCRP) custom property is another option. See the custom properties topic provided in the related links at the end of this topic. The advantage of using the following rules is that an expression can be used to determine which requests are affected by the policy, whereas MCRP custom properties only allow filtering by an application or an application's webmodule. The other advantage of these rules is that you can select the target (routingLocations) by cluster, server or webmodule, as opposed to just cluster.

You can specify SIP or HTTP protocols in the commands.

- “addRoutingRule”
- “changeRoutingDefaultRulesAction” on page 415
- “changeRoutingRuleAction” on page 417
- “changeRoutingRuleExpression” on page 419
- “changeRoutingRulePriority” on page 419
- “createRoutingRules” on page 420
- “listRoutingRules” on page 421
- “removeRoutingRule” on page 421

addRoutingRule

The **addRoutingRule** command adds a routing policy rule.

Note: If you have an existing application edition with a defined multi-cluster routing policy and you install a new edition, you must create a new multi-cluster routing policy for the new edition.

Required parameters

- **-protocol:** Specifies the name of the protocol to associate with a rule. (String, required)
- **-priority:** Positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)
- **-expression:** Specifies the rule expression. The expression must be enclosed in double quotes. For more information about specifying the parameters for the rule expression, refer to the SIP operands topic and the HTTP operands topic. (String, required)
- **-actionType:** Specifies the type of action to associate with a rule. (String, required)

The following list contains the types of actions to associate with HTTP rules:

- `permit`: Permit routing to servers not in a maintenance mode.
- `redirect`: Redirect the request to the URL specified by the `redirectURL` option.
- `reject`: Reject routing with return code specified by the `errorcode` option.
- `permitsticky`: Permit routing to servers not in maintenance mode and perform active affinity; that is, affinity is always preserved even when not requested by the application.
- `permitMM`: Permit routing to servers in maintenance mode.
- `permitstickyMM`: Permit routing to servers in maintenance mode and perform active affinity.

The following list contains the types of actions to associate with SIP rules:

- `permit`: Permit routing to servers not in maintenance mode.
- `reject`: Reject routing with return code specified by the `errorcode` option.

Optional parameters

- **-odrname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odrname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername** : Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.
- **-multiclusterAction**: Specifies the method to route requests if multiple routing location clusters are matched. The **-multiclusterAction** parameter applies to any of the `permit` actionTypes, and is required only if `actionType` is equal to `permit`, `permitsticky`, `permitMM`, or `permitstickyMM`.
 - `Failover`: Find the first cluster with an available server and load balance across that cluster. The order of a dynamically generated list of clusters is undefined.
 - `WRR`: Weighted Round Robin load balance. For UDP retransmission, maintain affinity.
 - `WLOR`: Weighted least outstanding request.

Note: The recommendation is to use the `WLOR` value instead of the `WRR` value.

The following list contains the possible values for SIP rules:

- `Failover`: Find the first cluster with an available server and load balance across that cluster. The order of a dynamically generated list of clusters is undefined.
- `WRR`: Weighted Round Robin load balance. For UDP retransmission, maintain affinity.
- `Error`: If there are multiple clusters, to select from it throws an error. it expects one and only one cluster.
- **-routingLocations**: Specifies a list of target locations to route requests. The **-routingLocations** parameter is required only if `actionType` is equal to any of the `permit` actionTypes.

Each operand in the list follows one of three formats, and can contain a wildcard `*` value, which matches any value:

- `cluster=cellName/clusterName`
- `server=cellName/nodeName/serverName`
- `module=cellName/applicationName/applicationVersion/moduleName`

With SIP routing rules only, you can alternatively define target clusters through a rule expression. The valid operators are `AND`, `OR`, `NOT` and parenthetical grouping. Format according to the following list:

- `cluster=cellName/clusterName`
- `server=cellName/nodeName/serverName`
- `modules=cellName/applicationName/applicationVersion/moduleName`
- `server maintenance mode=true` or `false`
- `node maintenance mode=true` or `false`

- protocol=PROTO_VALUE:
 - PROTO_SIP = sip**
SIP over TCP
 - PROTO_SIPS = sips**
SIP over SSL and TCP
 - PROTO_SIPU = sipu**
SIP over UDP
 - PROTO_SIPX = sipx**
SIP over XMEM

Note: For applications that have no applicationVersion value, leave the applicationVersion value blank:
module=cellName/application/moduleName.

- **-errorcode** : Integer error code to reject request. The **-errorcode** parameter is required only if **actionType** is equal to reject.

Batch mode example usage

The following example shows a failover of all applications in a cell to a generic server cluster that points to another cell:

- Using Jacl:

```
$AdminTask addRoutingRule {-odrname odr -nodename node1 -protocol SIP -priority 0 -expression
"request.method = 'getOperation'" -actionType permit -multiclusterAction Failover
-routingLocations cluster=**/*}
```

- Using Jython string:

```
AdminTask.addRoutingRule('-odrname odr -nodename node1 -protocol HTTP -priority 0 -expression
"queryparm$userid = \'123\'" -actionType permit -multiclusterAction Failover -routingLocations
"module=**/*/*,cluster=myCell/myFailoverGSCThatPointsToAnotherCell"')
```

The following example creates a multi-cluster routing policy for a new application edition:

- Using Jacl:

```
$AdminTask addRoutingRule {-odrname odr -nodename node1 -protocol
HTTP -priority 0 -expression "uri LIKEIN {'/contextRoot','/contextRoot/%'}"
-actionType permit -multiclusterAction Failover -routingLocations
cluster=cellName/clusterName}
```

- Using Jython string:

```
AdminTask.addRoutingRule('-odrname odr -nodename node1 -protocol
HTTP -priority 0 -expression "uri LIKEIN (\'/contextRoot\',\'/contextRoot/%\')"
-actionType permit -multiclusterAction Failover -routingLocations
cluster=cellName/clusterName')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask addRoutingRule {-interactive}
```

- Using Jython string:

```
AdminTask.addRoutingRule ('[-interactive]')
```

changeRoutingDefaultRulesAction

The **changeRoutingDefaultRulesAction** command changes the routing policy default action for a rule.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)

- **-actionType:** Specifies the type of action to associate with a rule. (String, required)
The following list contains the types of actions to associate with HTTP rules:
 - permit: Permit routing to servers not in a maintenance mode.
 - redirect: Redirect the request to the URL specified by the *redirectURL* option.
 - reject: Reject routing with return code specified by the *errorcode* option.
 - permitsticky: Permit routing to servers not in maintenance mode and perform active affinity; that is, affinity is always preserved even when not requested by the application.
 - permitMM: Permit routing to servers in maintenance mode.
 - permitstickyMM: Permit routing to servers in maintenance mode and perform active affinity.
 The following list contains the types of actions to associate with SIP rules:
 - permit: Permit routing to servers not in maintenance mode.
 - reject: Reject routing with return code specified by the *errorcode* option.

Optional parameters

- **-odrname:** Specifies the name of the ODR to which the routing policy work class applies. The **-odrname** parameter is required only if you modify an ODR.
- **-nodename:** Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername :** Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.
- **-multiclusterAction:** Specifies the method to route requests if multiple routing location clusters are matched. The **-multiclusterAction** parameter applies to any of the permit actionTypes, and is required only if **actionType** is equal to permit, permitsticky, permitMM, or permitstickyMM.
 - Failover: Find the first cluster with an available server and load balance across that cluster. The order of a dynamically generated list of clusters is undefined.
 - WRR: Weighted Round Robin load balance. For UDP retransmission, maintain affinity.
 - WLOR: Weighted least outstanding request.

Note: The recommendation is to use the WLOR value instead of the WRR value.

The following list contains the possible values for SIP rules:

- Failover: Find the first cluster with an available server and load balance across that cluster. The order of a dynamically generated list of clusters is undefined.
- WRR: Weighted Round Robin load balance. For UDP retransmission, maintain affinity.
- Error: If there are multiple clusters, to select from it throws an error. it expects one and only one cluster.
- **-routingLocations:** Specifies a list of target locations to route requests. The **-routingLocations** parameter is required only if **actionType** is equal to any of the permit actionTypes.

Each operand in the list follows one of three formats, and can contain a wildcard * value, which matches any value:

- cluster=cellName/clusterName
- server=cellName/nodeName/serverName
- module=cellName/applicationName/applicationVersion/moduleName

With SIP routing rules only, you can alternatively define target clusters through a rule expression. The valid operators are AND, OR, NOT and parenthetical grouping. Format according to the following list:

- cluster=cellName/clusterName
- server=cellName/nodeName/serverName
- modules=cellName/applicationName/applicationVersion/moduleName
- server maintenance mode=true or false

- node maintenance mode=true or false
- protocol=PROTO_VALUE:

PROTO_SIP = sip
SIP over TCP

PROTO_SIPS = sips
SIP over SSL and TCP

PROTO_SIPU = sipu
SIP over UDP

PROTO_SIPX = sipx
SIP over XMEM

Note: For applications that have no applicationVersion value, leave the applicationVersion value blank:
module=cellName/application//moduleName.

- **-errorcode** : Integer error code to reject request. The **-errorcode** parameter is required only if **actionType** is equal to reject.

Batch mode example usage

- Using Jacl:

The following examples shows a failover of a single cluster to the failover generic server cluster:

```
$AdminTask changeRoutingDefaultRulesAction {-odrname odr -nodename node1 -protocol
SIP -actionType permit -multiclusterAction Failover -routingLocations cluster=**/*}
```

- Using Jython string:

```
AdminTask.changeRoutingDefaultRulesAction('[-odrname odr -nodename node1 -protocol
HTTP -actionType permit -multiclusterAction Failover -routingLocations
"cluster=myCell/myPrimaryCluster,cluster=myCell/myFailoverCluster"]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeRoutingDefaultRulesAction {-interactive}
```

- Using Jython string:

```
AdminTask.changeRoutingDefaultRulesAction ('[-interactive]')
```

changeRoutingRuleAction

The **changeRoutingRuleAction** command changes a routing policy action for a rule.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)
- **-priority**: Positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)

Optional parameters

- **-odrname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odrname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername** : Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.
- **-multiclusterAction**: Specifies the method to route requests if multiple routing location clusters are matched. The **-multiclusterAction** parameter applies to any of the permit **actionTypes**, and is required only if **actionType** is equal to permit, permitsticky, permitMM, or permitstickyMM.

- Failover: Find the first cluster with an available server and load balance across that cluster. The order of a dynamically generated list of clusters is undefined.
- WRR: Weighted Round Robin load balance. For UDP retransmission, maintain affinity.
- WLOR: Weighted least outstanding request.

Note: The recommendation is to use the WLOR value instead of the WRR value.

The following list contains the possible values for SIP rules:

- Failover: Find the first cluster with an available server and load balance across that cluster. The order of a dynamically generated list of clusters is undefined.
- WRR: Weighted Round Robin load balance. For UDP retransmission, maintain affinity.
- Error: If there are multiple clusters, to select from it throws an error. it expects one and only one cluster.
- **-routingLocations:** Specifies a list of target locations to route requests. The **-routingLocations** parameter is required only if **actionType** is equal to any of the permit actionTypes.

Each operand in the list follows one of three formats, and can contain a wildcard * value, which matches any value:

- cluster=cellName/clusterName
- server=cellName/nodeName/serverName
- module=cellName/applicationName/applicationVersion/moduleName

With SIP routing rules only, you can alternatively define target clusters through a rule expression. The valid operators are AND, OR, NOT and parenthetical grouping. Format according to the following list:

- cluster=cellName/clusterName
- server=cellName/nodeName/serverName
- modules=cellName/applicationName/applicationVersion/moduleName
- server maintenance mode=true or false
- node maintenance mode=true or false
- protocol=PROTO_VALUE:

PROTO_SIP = sip
SIP over TCP

PROTO_SIPS = sips
SIP over SSL and TCP

PROTO_SIPU = sipu
SIP over UDP

PROTO_SIPX = sipx
SIP over XMEM

Note: For applications that have no applicationVersion value, leave the applicationVersion value blank:
module=cellName/application//moduleName.

- **-errorcode** : Integer error code to reject request. The **-errorcode** parameter is required only if **actionType** is equal to reject.

Batch mode example usage

- Using Jacl:

```
$AdminTask changeRoutingRuleAction {-odname odr -nodename node1 -protocol
SIP -priority 0 -multiclusterAction Failover -routingLocations cluster=*/*}
```

- Using Jython string:

```
AdminTask.changeRoutingRuleAction('[-odname odr -nodename node1 -protocol
HTTP -priority 0 -multiclusterAction WRR -routingLocations "cluster=myCell/*"]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeRoutingRuleAction {-interactive}
```
- Using Jython string:

```
AdminTask.changeRoutingRuleAction ('[-interactive]')
```

changeRoutingRuleExpression

The **changeRoutingRuleExpression** command changes a routing policy rule expression.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)
- **-priority**: Positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)
- **-expression**: Specifies the rule expression. The expression must be enclosed in double quotes. For more information about specifying the parameters for the rule expression, refer to the SIP operands topic and the HTTP operands topic. (String, required)

Optional parameters

- **-odurname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odurname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername**: Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask changeRoutingRuleExpression {-odurname odr -nodename node1 -protocol SIP -priority 0 -expression "request.method = 'getOperation0'"} 
```
- Using Jython string:

```
AdminTask.changeRoutingRuleExpression('[-odurname odr -nodename node1 -protocol HTTP -priority 0 -expression "queryparm$userid = \'123\'"]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeRoutingRuleExpression {-interactive}
```
- Using Jython string:

```
AdminTask.changeRoutingRuleExpression ('[-interactive]')
```

changeRoutingRulePriority

The **changeRoutingRulePriority** command changes a routing policy rule priority.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)
- **-priority**: Positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)
- **-expression**: Specifies the rule expression. The expression must be enclosed in double quotes. For more information about specifying the parameters for the rule expression, refer to the SIP operands topic and the HTTP operands topic. (String, required)

Optional parameters

- **-odrname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odrname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername** : Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask changeRoutingRulePriority {-odrname odr -nodename node1 -protocol SIP -priority 0 -expression "request.method = 'getOperation0'"}

```
- Using Jython string:

```
AdminTask.changeRoutingRulePriority('[-odrname odr -nodename node1 -protocol HTTP -priority 1 -expression "queryparam$userid = \'123\'"]')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeRoutingRulePriority {-interactive}

```
- Using Jython string:

```
AdminTask.changeRoutingRulePriority ('[-interactive]')
```

createRoutingRules

The **createRoutingRules** command creates a routing policy rule list.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)

Optional parameters

- **-odrname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odrname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername** : Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask createRoutingRules {-odrname odr -nodename node1 -protocol SIP}

```
- Using Jython string:

```
AdminTask.createRoutingRules('[-odrname odr -nodename node1 -protocol SIP']')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createRoutingRules {-interactive}

```
- Using Jython string:

```
AdminTask.createRoutingRules ('[-interactive]')
```

listRoutingRules

The **listRoutingRules** deletes a dynamic cluster from the configuration.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)

Optional parameters

- **-odurname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odurname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername** : Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask listRoutingRules {-odurname odr -nodename node1 -protocol SIP}
```
- Using Jython string:

```
AdminTask.listRoutingRules('-odurname odr -nodename node1 -protocol SIP')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask listRoutingRules {-interactive}
```
- Using Jython string:

```
AdminTask.listRoutingRules ('[-interactive]')
```

removeRoutingRule

The **removeRoutingRule** command removes a routing policy rule.

Required parameters

- **-protocol**: Specifies the name of the protocol to associate with a rule. (String, required)
- **-priority**: Positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)
- **-expression**: Specifies the rule expression. The expression must be enclosed in double quotes. For more information about specifying the parameters for the rule expression, refer to the SIP operands topic and the HTTP operands topic. (String, required)

Optional parameters

- **-odurname**: Specifies the name of the ODR to which the routing policy work class applies. The **-odurname** parameter is required only if you modify an ODR.
- **-nodename**: Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- **-clustername** : Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask removeRoutingRule {-odurname odr -nodename node1 -protocol SIP -expression  
"request.method = 'getOperation'"} 
```

- Using Jython string:

```
AdminTask.removeRoutingRule(['-odrname odr -nodename node1 -protocol HTTP -expression
"queryparm$userid = \'123\'"])')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask removeRoutingRule {-interactive}
```

- Using Jython string:

```
AdminTask.removeRoutingRule ('[-interactive]')
```

Rules for ODR service policy administrative tasks

You can use administrative tasks to configure Session Initiation Protocol (SIP) or HTTP rules for an on demand router (ODR) service policy.

Command list

You can specify SIP or HTTP protocols in the commands. The following examples use SIP as the protocol:

- “addServiceRule”
- “removeServiceRule” on page 423
- “createServiceRules” on page 424
- “listServiceRules” on page 424
- “changeServiceRuleAction” on page 425
- “changeServiceDefaultRulesAction” on page 426
- “changeServiceRuleExpression” on page 426
- “changeServiceRulePriority” on page 427

addServiceRule

The **addServiceRule** command adds a service policy rule.

Required parameters

-protocol

Specifies the name of the protocol to associate with a rule. (String, required)

-priority

Specifies a positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)

-expression

Specifies the rule expression. The expression must be enclosed in double quotes. (String, required)

-transactionClass

Specifies the transaction class to associate with a rule. (String, required)

Optional parameters

-odrname

Specifies the name of the ODR to which the service policy work class applies. The **-odrname** parameter is required only if you modify an ODR.

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername

Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask addServiceRule {-odrname odr -nodename node1 -protocol SIP -priority 0 -expression "request.method = 'getOperation'" -transactionClass Default_TC}
```

- Using Jython string:

```
AdminTask.addServiceRule('-odrname odr -nodename node1 -protocol SIP -priority 0 -expression "request.method = \'getOperation\'" -transactionClass Default_TC')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask addServiceRule {-interactive}
```

- Using Jython string:

```
AdminTask.addServiceRule ('[-interactive]')
```

removeServiceRule

The **removeServiceRule** command removes a service policy rule.

Required parameters

-protocol

Specifies the name of the protocol to associate with a rule. (String, required)

-expression

Specifies the rule expression. The expression must be enclosed in double quotes. (String, required)

Optional parameters

-odrname

Specifies the name of the ODR to which the service policy work class applies. The **-odrname** parameter is required only if you modify an ODR.

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername

Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask removeServiceRule {-odrname odr -nodename node1 -protocol SIP -expression "request.method = 'getOperation'"}
```

- Using Jython string:

```
AdminTask.removeServiceRule('-odrname odr -nodename node1 -protocol SIP -expression "request.method = \'getOperation\'"')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask removeServiceRule {-interactive}
```

- Using Jython string:

```
AdminTask.removeServiceRule ('[-interactive]')
```

createServiceRules

The **createServiceRules** command creates a rule list for the service policy.

Required parameters

-protocol

Specifies the name of the protocol to associate with a rule. (String, required)

Optional parameters

-odrname

Specifies the name of the ODR to which the service policy work class applies. The **-odrname** parameter is required only if you modify an ODR.

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername

Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask createServiceRules {-odrname odr -nodename node1 -protocol SIP}
```
- Using Jython string:

```
AdminTask.createServiceRules('-odrname odr -nodename node1 -protocol SIP')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createServiceRules {-interactive}
```
- Using Jython string:

```
AdminTask.createServiceRules ('[-interactive]')
```

listServiceRules

The **listServiceRules** lists service policy rules.

Required parameters

-protocol

Specifies the name of the protocol to associate with a rule. (String, required)

Optional parameters

-odrname

Specifies the name of the ODR to which the service policy work class applies. The **-odrname** parameter is required only if you modify an ODR.

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername

Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask listServiceRules {-odname odr -nodename node1 -protocol SIP}
```

- Using Jython string:

```
AdminTask.listServiceRules('-odname odr -nodename node1 -protocol SIP')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask listServiceRules {-interactive}
```

- Using Jython string:

```
AdminTask.listServiceRules ('[-interactive]')
```

changeServiceRuleAction

The **changeServiceRuleAction** command changes the service policy action for a rule.

Restriction: The use of the **changeServiceRuleAction** command with HTTP is not supported.

Required parameters

-protocol

Specifies the name of the protocol to associate with a rule. (String, required)

-priority

Specifies a positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)

-transactionClass

Specifies the transaction class to associate with a rule. (String, required)

Optional parameters

-odname

Specifies the name of the ODR to which the service policy work class applies. The **-odname** parameter is required only if you modify an ODR.

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername

Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask changeServiceRuleAction {-odname odr -nodename node1 -protocol SIP -priority 0  
-transactionClass Default_TC}
```

- Using Jython string:

```
AdminTask.changeServiceRuleAction('-odname odr -nodename node1 -protocol SIP -priority 0  
-transactionClass Default_TC')
```

Interactive mode example usage

- Using Jacl:


```
$AdminTask changeServiceRuleAction {-interactive}
```
- Using Jython string:


```
AdminTask.changeServiceRuleAction ('[-interactive]')
```

changeServiceDefaultRulesAction

The **changeServiceDefaultRulesAction** command changes the default action of a service policy rule.

Restriction: The use of the **changeServiceDefaultRulesAction** command with HTTP is not supported.

Required parameters

- protocol**
Specifies the name of the protocol to associate with a rule. (String, required)
- transactionClass**
Specifies the transaction class to associate with a rule. (String, required)

Optional parameters

- odurname**
Specifies the name of the ODR to which the service policy work class applies. The **-odurname** parameter is required only if you modify an ODR.
- nodename**
Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.
- clustername**
Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:


```
$AdminTask changeServiceDefaultRulesAction {-odurname odr -nodename node1 -protocol SIP  
-transactionClass Default_TC}
```
- Using Jython string:


```
AdminTask.changeServiceDefaultRulesAction('-odurname odr -nodename node1 -protocol SIP  
-transactionClass Default_TC')
```

Interactive mode example usage

- Using Jacl:


```
$AdminTask changeServiceDefaultRulesAction {-interactive}
```
- Using Jython string:


```
AdminTask.changeServiceDefaultRulesAction ('[-interactive]')
```

changeServiceRuleExpression

The **changeServiceRuleExpression** command changes a rule expression of a service policy.

Required parameters

- protocol**
Specifies the name of the protocol to associate with a rule. (String, required)

-priority
Specifies a positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)

-expression
Specifies the rule expression. The expression must be enclosed in double quotes. (String, required)

-transactionClass
Specifies the transaction class to associate with a rule. (String, required)

Optional parameters

-odurname
Specifies the name of the ODR to which the service policy work class applies. The **-odurname** parameter is required only if you modify an ODR.

-nodename
Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername
Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

- Using Jacl:

```
$AdminTask changeServiceRuleExpression {-odurname odr -nodename node1 -protocol SIP -priority 0  
-expression "request.method = 'getOperation0'"}  
}
```

- Using Jython string:

```
AdminTask.changeServiceRuleExpression('-odurname odr -nodename node1 -protocol SIP -priority 0  
-expression "request.method = \'getOperation0\'")  
}
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeServiceRuleExpression {-interactive}
```

- Using Jython string:

```
AdminTask.changeServiceRuleExpression ('[-interactive]')
```

changeServiceRulePriority

The **changeServiceRulePriority** command changes the priority of a service policy rule.

Required parameters

-protocol
Specifies the name of the protocol to associate with a rule. (String, required)

-priority
Specifies a positive integer value representing the priority of a rule. Zero is the highest priority. (String, required)

-expression
Specifies the rule expression. The expression must be enclosed in double quotes. (String, required)

Optional parameters

-odurname
Specifies the name of the ODR to which the service policy work class applies. The **-odurname** parameter is required only if you modify an ODR.

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR.

-clustername

Specifies the name of the cluster to which the rule applies. The **-clustername** parameter is required only if you modify an ODR cluster.

Batch mode example usage

• Using Jacl:

```
$AdminTask changeServiceRulePriority {-odname odr -nodename node1 -protocol SIP -priority 0
-expression "request.method = 'getOperation0'"}

```

• Using Jython string:

```
AdminTask.changeServiceRulePriority('-odname odr -nodename node1 -protocol SIP -priority 0
-expression "request.method = \'getOperation0\'")

```

Interactive mode example usage

• Using Jacl:

```
$AdminTask changeServiceRulePriority {-interactive}

```

• Using Jython string:

```
AdminTask.changeServiceRulePriority ('[-interactive]')

```

Ruleset administrative tasks

You can use the **ruleset** commands to add, remove, or modify rules and actions of the `ruleset.xml` file without using the administrative console.

Use the following commands to create ruleset lists, and add or remove rules and actions from the ruleset:

- “createRuleset”
- “listRuleset” on page 429
- “addRuleToRuleset” on page 430
- “removeRuleFromRuleset” on page 430
- “changeRulePriority” on page 431
- “changeRuleExpression” on page 432
- “addActionToRule” on page 433
- “removeActionFromRule” on page 434
- “addDefaultAction” on page 434
- “removeDefaultAction” on page 435

createRuleset

The **createRuleset** command creates a ruleset list.

Required parameters

-rulesetName

Specifies the name of the ruleset. (String, required)

-rulesetType

Specifies the protocol type to associate with the ruleset. The only valid types are HTTP and SOAP. (String, required)

-defaultContinue

Specifies the setting for the default continue flag value. Specify True or False. (Boolean, required)

Optional parameters

-odrname

Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask createRuleset {-odrname odr -nodename node1 -rulesetName myRuleset -rulesetType HTTP -defaultContinue True}
```

- Using Jython string:

```
AdminTask.createRuleset ('-odrname odr -nodename node1 -rulesetName myRuleset -rulesetType HTTP -defaultContinue True')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask createRuleset {-interactive}
```

- Using Jython string:

```
AdminTask.createRuleset ('[-interactive]')
```

listRuleset

The **listRuleset** command lists ruleset rules and actions.

Required parameters

-rulesetName

Specifies the name of the ruleset. (String, required)

Optional parameters

-odrname

Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask listRuleset {-odrname odr -nodename node1 -rulesetName myRuleset}
```

- Using Jython string:

```
AdminTask.listRuleset ('-odrname odr -nodename node1 -rulesetName myRuleset')
```

Interactive mode example usage

- Using Jacl:


```
$AdminTask listRuleset {-interactive}
```
- Using Jython string:


```
AdminTask.listRuleset ('[-interactive]')
```

addRuleToRuleset

The **addRuleToRuleset** command adds a rule to the ruleset.

Required parameters

- rulesetName**
Specifies the name of the ruleset. (String, required)
- ruleName**
Specifies the name of the rule. The **-ruleName** parameter must be unique within the ruleset. (String, required)
- rulePriority**
Specifies a positive integer value representing the priority of the rule. Zero is the highest priority. (String, required)
- expression**
Specifies the rule expression. You must enclose the value in double quotes. (String, required)

Optional parameters

- odrname**
Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)
- nodename**
Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)
- clustername**
Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:


```
$AdminTask addRuleToRuleset {-odrname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -rulePriority 0 -expression "vhost = vhostA"}
```
- Using Jython string:


```
AdminTask.addRuleToRuleset('-odrname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -rulePriority 0 -expression "vhost = \'vhostA\'")
```

Interactive mode example usage

- Using Jacl:


```
$AdminTask addRuleToRuleset {-interactive}
```
- Using Jython string:


```
AdminTask.addRuleToRuleset ('[-interactive]')
```

removeRuleFromRuleset

The **removeRuleFromRuleset** command removes a rule from the ruleset.

Required parameters

-rulesetName
Specifies the name of the ruleset. (String, required)

-ruleName
Specifies the name of the rule to remove. (String, required)

Optional parameters

-odurname
Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odurname** parameter is required only if you modify an ODR. (String, optional)

-nodename
Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername
Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:
`$AdminTask removeRuleFromRuleset {-odurname odr -nodename node1 -rulesetName myRuleset -ruleName myRule}`
- Using Jython string:
`AdminTask.removeRuleFromRuleset('-odurname odr -nodename node1 -rulesetName myRuleset -ruleName myRule')`

Interactive mode example usage

- Using Jacl:
`$AdminTask removeRulefromRuleset {-interactive}`
- Using Jython string:
`AdminTask.removeRulefromRuleset ('[-interactive]')`

changeRulePriority

The **changeRulePriority** command modifies the priority of a rule.

Required parameters

-rulesetName
Specifies the name of the ruleset. (String, required)

-ruleName
Specifies the name of the rule to modify. (String, required)

-rulePriority
Specifies a positive integer value representing the priority of the rule. Zero is the highest priority. (String, required)

Optional parameters

-odurname
Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odurname** parameter is required only if you modify an ODR. (String, optional)

-nodename
Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask changeRulePriority {-odurname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -rulePriority 1}
```

- Using Jython string:

```
AdminTask.changeRulePriority('-odurname odr-nodename node1 -rulesetName myRuleset -ruleName myRule -rulePriority 1')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeRulePriority {-interactive}
```

- Using Jython string:

```
AdminTask.changeRulePriority ('[-interactive]')
```

changeRuleExpression

The **changeRuleExpression** command modifies a rule expression.

Required parameters

-rulesetName

Specifies the name of the ruleset. (String, required)

-ruleName

Specifies the name of the rule to modify. (String, required)

-expression

Specifies the new rule expression. (String, required)

Optional parameters

-odurname

Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odurname** parameter is required only if you modify an ODR. (String, optional)

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask changeRuleExpression {-odurname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -expression "vhost = vhostB"}
```

- Using Jython string:

```
AdminTask.changeRuleExpression('-odurname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -expression "vhost = \'vhostB\'"')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask changeRuleExpression {-interactive}
```

- Using Jython string:


```
AdminTask.changeRuleExpression ('[-interactive]')
```

addActionToRule

The **addActionToRule** command adds an action to a rule.

Required parameters

-rulesetName

Specifies the name of the ruleset. (String, required)

-ruleName

Specifies the name of the rule. The **-ruleName** parameter must be unique within the ruleset. (String, required)

-actionName

Specifies the name of the action. The **-actionName** parameter must be unique within the ruleset. (String, required)

-actionType

Specifies the type of action to associate with the rule. (String, required)

-actionValue

Specifies the action value, such as the format of the log file, to use. (String, required)

-actionContinue

Specifies the setting for the action continue flag value. Specify True or False. (Boolean, required)

Optional parameters

-odrname

Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask addActionToRule {-odrname odr -nodename node1 -rulesetName myRuleset -ruleName myRule  
-actionName myAction -actionType log -actionValue "MyCustom.log %r %T %Z %Z" -actionContinue true}
```

- Using Jython string:

```
AdminTask.addActionToRule('-odrname odr -nodename node1 -rulesetName myRuleset -ruleName myRule  
-actionName myAction -actionType log -actionValue "MyCustom.log %r %T %Z %Z" -actionContinue true')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask addActionToRule {-interactive}
```

- Using Jython string:

```
AdminTask.addActionToRule ('[-interactive]')
```

removeActionFromRule

The **removeActionFromRule** command removes an action from a rule.

Required parameters

- rulesetName**
Specifies the name of the ruleset. (String, required)
- ruleName**
Specifies the name of the rule that contains the action. (String, required)
- actionName**
Specifies the name of the action to remove. (String, required)

Optional parameters

- odrname**
Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)
- nodename**
Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)
- clustername**
Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:

```
$AdminTask removeActionFromRule {-odrname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -actionName myAction}
```
- Using Jython string:

```
AdminTask.removeActionFromRule('-odrname odr -nodename node1 -rulesetName myRuleset -ruleName myRule -actionName myAction')
```

Interactive mode example usage

- Using Jacl:

```
$AdminTask removeActionFromRule {-interactive}
```
- Using Jython string:

```
AdminTask.removeActionFromRule ('[-interactive]')
```

addDefaultAction

The **addDefaultAction** command adds a default action to a ruleset.

Required parameters

- rulesetName**
Specifies the name of the ruleset. (String, required)
- actionName**
Specifies the name of the action. The **-actionName** parameter must be unique within the ruleset. (String, required)
- actionType**
Specifies the type of action to associate with the rule. (String, required)

-actionValue

Specifies the action value, such as the format of the log file, to use. (String, required)

-actionContinue

Specifies the setting for the action continue flag value. Specify True or False. (Boolean, required)

Optional parameters

-odrname

Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

• Using Jacl:

```
$AdminTask addDefaultAction {-odrname odr -nodename node1 -rulesetName myRuleset -actionName defAction
-actionType log -actionValue "default.log %a %d %D" -actionContinue True}
```

• Using Jython string:

```
AdminTask.addDefaultAction('-odrname odr -nodename node1 -rulesetName myRuleset -actionName defAction
-actionType log -actionValue "default.log %a %d %D" -actionContinue True')
```

Interactive mode example usage

• Using Jacl:

```
$AdminTask addDefaultAction {-interactive}
```

• Using Jython string:

```
AdminTask.addDefaultAction ('[-interactive]')
```

removeDefaultAction

The **removeDefaultAction** command removes a default action from a ruleset.

Required parameters

-rulesetName

Specifies the name of the ruleset. (String, required)

-actionName

Specifies the name of the action to remove. (String, required)

Optional parameters

-odrname

Specifies the name of the on demand router (ODR) to which the ruleset applies. The **-odrname** parameter is required only if you modify an ODR. (String, optional)

-nodename

Specifies the name of the node on which the ODR resides. The **-nodename** parameter is required only if you modify an ODR. (String, optional)

-clustername

Specifies the name of the cluster to which the ruleset applies. The **-clusterName** parameter is required only if you modify an ODR cluster. (String, optional)

Batch mode example usage

- Using Jacl:
`$AdminTask removeDefaultAction {-odrname odr -nodename node1 -rulesetName myRuleset -actionName defAction}`
- Using Jython string:
`AdminTask.removeDefaultAction('-odrname odr -nodename node1 -rulesetName myRuleset -actionName defAction')`

Interactive mode example usage

- Using Jacl:
`$AdminTask removeDefaultAction {-interactive}`
- Using Jython string:
`AdminTask.removeDefaultAction ('[-interactive]')`

Runtime operations user preferences administrative tasks

You can use the Jacl and Jython scripting languages to clone, set, and get preferences with the `wsadmin` tool. Use the commands and parameters in the `UserPreferences` group.

User preferences are stored under the cell context or in the context for each resource type. The following list gives examples at the cell level and each resource level:

- `cells/myCell/preferences/User_1/preferenceset.xml`
- `cells/myCell/applications/TestAppA.ear/preferences/User_1/preferenceset.xml`
- `cells/myCell/nodes/myNode/servers/myServer/preferences/User_1/preferenceset.xml`
- `cells/myCell/clusters/myCluster/preferences/User_1/preferenceset.xml`
- `cells/myCell/serviceclasses/Platinum/preferences/User_1/preferenceset.xml`

Use the following commands to work with the preferences for runtime operations:

- “`clonePreference`”
- “`getPreferences`” on page 437
- “`setPreference`” on page 438

For examples using Jython to interact with preferences objects, see “Jython preferences object commands” on page 439.

clonePreference

The `clonePreference` command copies a preference from one user or role to another user or role.

Target object

The `configID` of the parent.

Required parameters

-name Specifies the name of the preference to clone. (String, required)

Optional parameters

- **-scope**: Specifies the scope of the preference to save when the preferences are cloned. If you do not specify a scope, the scope of the `configID` is used. (String, optional) The scope can be one of the following values:
 - Applications
 - Servers
 - Clusters

- ServiceClasses
- **-touser:** Specifies the user for which to retrieve preferences. If you do not indicate this value, the default uses the user ID for the session in which the command is running. (String, optional)
- **-torole:** Specifies the role for which to retrieve preferences. This value is ignored if you specified the **-touser** parameter. (String, optional)
- **-fromuser:** Specifies the user that gets a copy the preferences. You can indicate multiple users by putting the users in a list, for example: user1,user2,user3. Do not put spaces in the list. (String, optional)
- **-fromrole:** Specifies the role that gets a copy of the preferences. (String, optional)

Batch mode example usage

- Using Jython:


```
cellid= AdminConfig.getid("/Cell:myCell")
AdminTask.clonePreference(cellid, '[-name ChartGroups -fromuser user1 -touser user2,user3,user4,user5]')
AdminConfig.save()
```
- Using Jacl:


```
$AdminTask clonePreference cells/dabtcCell02|cell.xml#Cell_1 {-name ChartGroups -fromuser user1
-touser user2,user3,user4,user5}
```

Interactive mode example usage

- Using Jython:


```
AdminTask.clonePreference('[-interactive]')
```
- Using Jacl:


```
$AdminTask clonePreference {-interactive}
```

getPreferences

The **getPreferences** command returns a collection of preferences based on the preference document you specify by passing in a role or user name. You must have administrator privileges to get preferences for a certain user or user role.

Target object

The configID of the parent.

Required parameters

- **-name:** Specifies the name of the preference. (String, required)

Optional parameters

- **-scope:** Specifies a scope for the preferences. You can choose from one of the following valid values:
 - Applications
 - Servers
 - Clusters
 - ServiceClasses

The default scope is empty, or undefined. If the scope is empty, the scope is derived from where the document is being read. For example, if the document is in the cell scope, then the scope is cell scoped. If the document is in a specific server context, then the scope is assumed to be applicable to that server. (String, optional)

- **-user:** Specifies the user name for which to retrieve preferences. If you do not indicate a user, the default action is to use the user name that is running the script. (String, optional)
- **-role:** Specifies the user role for which to retrieve preferences. (String, optional)

Return value

The command returns a `DescriptivePropertyGroup` object that contains the preferences, or null if no preferences are found.

Batch mode example usage

- Using Jython string:

```
cellid= AdminConfig.getid("/Cell:myCell")
AdminTask.getPreferences(cellid, '[-name ChartGroups -user user1]')
```

- Using Jacl:

```
$AdminTask getPreferences cells/dabtcCell102|cell.xml#Cell_1 {-name Reports -user user1}
```

Interactive mode example usage

- Using Jython string:

```
AdminTask.getPreferences('[-interactive]')
```

- Using Jacl:

```
$AdminTask getPreferences {-interactive}
```

setPreference

The **setPreference** command sets a specific preference in a preference document. You must have administrator privileges to set preferences for a certain user or user role.

Target object

The configID of the parent.

Required parameters

- **-name**: Specifies the name of the preference. (String, required)
- **-propertyLongName**: Specifies the fully-qualified path to the property in the property group, delimited by a forward slash (/) character. If the property is directly contained in the root property group, then the short name of the property can be used because it would be exactly equal to the long name. (String, required)
- **-propertyValue**: Specifies the new value to set for the preference. (String, required)

Optional parameters

- **-scope**: Specifies the scope of the preference to save when the preferences are cloned. If you do not specify a scope, the scope of the configID is used. (String, optional) The scope can be one of the following values:
 - Applications
 - Servers
 - Clusters
 - ServiceClasses
- **-user**: Specifies the user name for which to save preferences. If you do not indicate a user, the default action is to use the user name that is running the script. (String, optional)
- **-role**: Specifies the user role for which to save preferences. (String, optional)

Batch mode example usage

- Using Jython string:

```
AdminTask.setPreference(cellid, '[-name myChartGroup -propertyLongName defaultChartSize -propertyValue large]')
```

- Using Jacl:

```
$AdminTask setPreference cells/dabtcCell102|cell.xml#Cell_1 {-name Reports -user user1 -propertyLongName
defaultChartSize -propertyValue large}
```

Interactive mode example usage

- Using Jython string:

```
AdminTask.setPreference(['-interactive'])
```

- Using Jacl:

```
$AdminTask setPreference {-interactive}
```

Jython preferences object commands

Use the following examples to interact with the preferences object with Jython commands:

To list preferences per user

```
prefSets = AdminConfig.list("PreferenceSet")
print prefSets
```

To remove preferences for a user

```
prefid = AdminConfig.getid("/PreferenceSet:User_1")
AdminConfig.remove(prefid)
AdminConfig.save()
```

To list preferences for all users, grouped by user

```
prefs = AdminConfig.list("Preferences")
print prefs
```

Custom properties for Intelligent Management

Custom properties are unique settings in the administrative console. They are name-value pairs that you can enter on specific administrative console pages. Use the documentation for the specific custom property to determine where the property must be set.

Setting cell-wide custom properties

You can configure a cell-wide custom property by completing the following steps:

1. In the administrative console, select **System administration > Cell > Custom properties > New**.
2. Provide a name and value for the cell property.
3. Click **OK**.

Setting custom properties at other levels

Some custom properties can be set at other levels than the cell level. If the custom property documentation specifies another scope at which the custom property must be set, the steps are similar, but you first navigate to the indicated level before you set the custom property.

1. Navigate to the indicated configuration object. For example, if the scope indicated is the dynamic cluster, click **Servers > Dynamic clusters > *dynamic_cluster_name***.
2. Click **Custom properties > New**.
3. Provide a name and value for the cell property.
4. Click **OK**.

Application edition manager custom properties

By configuring custom properties for application edition manager, you can change the settings for your validation cluster. For example, you can set the operating mode of the validation cluster or create a custom property to keep the validation cluster after the rollout completes.

appedition.rollout.softreset.fulldrainageinterval:

You can set the `appedition.rollout.softreset.fulldrainageinterval` system property on the deployment manager to force a soft reset to wait the entire drainage interval. Starting in Version 7.0.0.1, the application edition quiesce manager might not wait the full length of the drainage interval if all requests are quiesced before the interval times out. Performance Monitoring Infrastructure (PMI) statistics are available for the quiesce manager to determine if all active requests on a server have been quiesced.

Table 64. appedition.rollout.softreset.fulldrainageinterval custom property values

Value	Description
Scope	Deployment manager
Valid values	No value: Deactivated. The quiesce manager might not wait the full length of the drainage interval. true: Enabled. The quiesce manager is forced to wait the entire drainage interval.

deactivate.checkRoutingRules:

You can enable the `deactivate.checkRoutingRules` custom property if you do not want an application edition deactivation to occur when routing policy work classes are still routing workload to the application edition. If you do not set the property, the deactivation still occurs, even if routing policy work class actions are found that reference the edition.

Table 65. deactivate.checkRoutingRules custom property values

Value	Description
Scope	Cell
Valid values	No value: Deactivated, application editions can be deactivated even if routing policy work classes reference the edition. true: Enabled. If routing policy work classes reference an edition that is being deactivated, the deactivation does not occur until the actions are removed or changed.

saveClonedCluster:

You can set the `saveClonedCluster` custom property to maintain the validation cluster after you perform rollout to a new edition. If you do not set this custom property, the validation cluster is deleted after the rollout or after the validation is canceled. The `saveClonedCluster` custom property applies only to dynamic clusters.

Set this custom property on the validation cluster. To save the validation cluster, set the value to `true`.

Table 66. saveClonedCluster custom property values

Value	Description
Scope	Dynamic cluster
Valid values	No value: Deactivated. The validation cluster is deleted after a rollout to a new edition. true: Enabled. The validation cluster is saved when the rollout to a new edition occurs.

uninstall.checkRoutingRules:

You can enable the `uninstall.checkRoutingRules` custom property if you do not want the uninstallation of an application edition to occur when routing policy work classes are still routing workload to the application edition. If you do not set the property, the uninstallation still occurs, even if routing policy work class actions are found that reference the edition.

Table 67. uninstall.checkRoutingRules custom property values

Value	Description
Scope	Cell
Valid values	No value: Deactivated. Application editions can be uninstalled even if routing policy work classes reference the edition. true: Enabled. If routing policy work classes reference an edition that is being uninstalled, the uninstallation does not occur until the actions are removed or changed.

VALIDATION_OPERATIONALMODE:

To set the operational mode of the cloned validation cluster to a different mode from the production cluster, create the `VALIDATION_OPERATIONALMODE` custom property.

You can set this custom property on the production cluster before you start the validation process.

Table 68. VALIDATION_OPERATIONALMODE custom property values

Value	Description
Scope	Dynamic cluster
Valid values	automatic, manual, or supervised

If the value is unspecified or not valid, the validation cluster operational mode is set to the same operational mode as the production cluster. If you specify any other value or you do not specify a value, then the validation dynamic cluster is set to manual mode.

Application placement custom properties

Custom properties modify the application placement controller configuration. You can use these settings to tune the application placement controller behavior beyond the settings that are in the administrative console.

To set the application placement custom properties, expand **Operational policies > Autonomic controllers > Autonomic managers > Autonomic request flow manager**.

APC.predictor custom property:

You can use the `APC.predictor` custom property to enable elasticity mode for a dynamic cluster of on demand routers (ODRs).

By setting the custom property, elasticity mode is enabled on all ODR dynamic clusters, and the application placement controller starts and stops servers based on CPU usage alone. Furthermore, the controller no longer retrieves data from the autonomic request flow manager (ARFM) regarding what servers to start and stop.

Table 69. APC.predictor custom property values

Value	Description
Scope	Cell
Valid value	CPU

APC.XD.memoryProfiler.totalMemoryWeight and APC.XD.memoryProfiler.residentMemoryWeight:

Custom properties used for specifying the importance of total memory and available RAM when calculating available memory.

You can set the custom property to calculate available memory using the following code (on one line):

```
available memory = APC.XD.memoryProfiler.totalMemoryWeight * totalMemory +
  APC.XD.memoryProfiler.residentMemoryWeight * residentMemory.
```

The default value is 33.33333 for APC.XD.memoryProfiler.totalMemoryWeight.

The default value is 66.66666 for APC.XD.memoryProfiler.residentMemoryWeight.

On some operating systems, particularly Linux, the virtual size can be extremely high, causing the memory on a system is to appear consumed. This prevents APC from starting instances on nodes that actually have plenty of free memory. To resolve this issue, set APC.XD.memoryProfiler.totalMemoryWeight to 12 and the APC.XD.memoryProfiler.residentMemoryWeight to 88.

CenterCell custom property:

When you are configuring multi-cell performance management in your environment, you can use the CenterCell custom property to designate one cell as the center cell. You also set the CenterCell custom property individually for each cell that you want to designate as a point cell.

Note: One and only one custom property should be set to true.

Table 70. CenterCell custom property values

Value	Description
Scope	Cell
Valid values	true: Designates one cell as the center cell false: Designates one cell as a point cell

apcConcurrentStartSize:

You can use the apcConcurrentStartSize custom property to define the maximum number of server instances the application placement controller should attempt to start concurrently on a particular node.

The default number of server instances that the application placement controller attempts to start on a node is one instance. Set this property to a value greater than 1 if you want the application placement controller to start more than one server instance at the same time on a particular node.

Table 71. apcConCurrentStartSize custom property values

Value	Description
Scope	Application placement controller
Valid values	An integer value that specifies the maximum number of instances to be started concurrently on a specific node.

Table 71. *apcConCurrentStartSize* custom property values (continued)

Value	Description
Default	1

lazyStartMinInstances:

You can use the `lazyStartMinInstances` custom property to configure multiple server instances to start when the on demand router (ODR) detects activity.

Prior to Version 6.1.1.2, only one server instance was started when a dynamic cluster was configured for application lazy start and the ODR received requests for the inactive dynamic cluster. By setting the `lazyStartMinInstances` custom property, however, multiple instances can be started.

When you set the custom property at the application placement level, the property applies to all of your dynamic clusters. Alternatively, you can set the custom property on a specific dynamic cluster so that multiple instances are started by that particular dynamic cluster. If other dynamic clusters configured for application lazy start exist, those clusters start only one instance each. The custom property value set at the dynamic cluster level overrides the custom property value set at the application placement level.

Table 72. *lazyStartMinInstances* custom property values

Value	Description
Scope	Dynamic cluster
Valid values	An integer value that specifies the minimum number of instances to be lazily started.
Default	1

apc.log.enablePlacementLog:

You can use the `apc.log.enablePlacementLog` custom property to enable or disable logs from being saved to the `apc.log` log file. The `apc.log` log file contains information about placement decisions. IBM support can use this log file to help understand the placement decisions that are being made by the application placement controller.

Table 73. *apc.log.enablePlacementLog* custom property values

Value	Description
Scope	Application placement controller
Valid values	No value: The placement log is enabled. false : The placement log is disabled.

apc.log.LogFileSize:

You can use the `apc.log.LogFileSize` custom property to set the size of the application placement controller log file, the `apc.log` file, to a size in megabytes. The default size is 100 megabytes, but you can change the default value.

Table 74. *apc.log.LogFileSize* custom property values

Value	Description
Scope	Application placement controller

Table 74. *apc.log.LogFileSize* custom property values (continued)

Value	Description
Valid values	<p>An integer value of the number of megabytes for the log file. For example, if you enter 200, each log file will be about 200 megabytes.</p> <p>The Java virtual machine (JVM) attempts to limit the log file to a byte size of 200 * 1024 * 1024 or 209,715,200 bytes, which is a smaller limit used by the JVM and is an approximate value. The actual size will likely be larger and varies by each JVM vendor. The actual byte size on the disk depends on the sector and cluster size of the physical disk, which will likely be larger than the byte size of the file.</p>

apc.log.numHistoricalFiles:

You can use the *apc.log.numHistoricalFiles* custom property to set the number of log files that are saved for the application placement controller. When the application placement controller log file reaches its maximum size, the historical file is saved if this value is greater than 1. If the value is equal to 1, the historical data appends to the existing file.

Table 75. *apc.log.numHistoricalFiles* custom property values

Value	Description
Scope	Application placement controller
Valid values	An integer value that specifies the number of historical files you want to save, for example, 2.

cpuUtilizationThreshold:

You can use the *cpuUtilizationThreshold* custom property to define the percentage of processor power that you can use on each node in a cell.

The *cpuUtilizationThreshold* custom property defines the percentage of processor power that is used on the nodes in your environment. The default value is 100, which means that 100% of the processor can be used on each node in the cell. For configurations that involve z/OS nodes, you must set this value to less than 100.

The application placement controller attempts to compute placements so that the utilization of any node does not exceed the defined threshold. The application placement controller considers all processes, including processes that are not related to WebSphere Application Server or Intelligent Management when making these calculations.


Important:  This property must be defined and must be set to a value less than 100 for configurations that involve z/OS nodes.

Table 76. *cpuUtilizationThreshold* custom property values

Value	Description
Scope	<p>Application placement controller: applies to all nodes in the cell.</p> <p>Node: applies to a specific node.</p>
Valid values	An integer value that specifies the percentage of processor power to use on each node in the cell.

GenerateUniquePorts:

You can use the GenerateUniquePorts custom property to generate unique ports for each dynamic cluster member.

z/OS By default, dynamic cluster members on distributed nodes are created with unique ports, while z/OS nodes do not have unique ports. On the z/OS platform, port numbers are usually set symmetrically according to specific range of numbers. The first dynamic cluster member always has unique ports, and these port numbers are applied to the other dynamic cluster members. You can set this property on the node group or on an individual node.

Table 77. GenerateUniquePorts custom property values

Value	Description
Scope	Node group, node
Valid values	true: Unique ports are generated for each dynamic cluster member. false: Unique ports are created for the first dynamic cluster member, and subsequent dynamic cluster members use the same ports.
Default	z/OS platforms: false Attention: If vertical stacking is enabled for a z/OS dynamic cluster, the port allocation strategy is forced to true, to generate unique ports for each dynamic cluster member. Distributed platforms: true

maintenanceModeOnOperationFail:

You can use the maintenanceModeOnOperationFail custom property to move a server into maintenance mode when the server start fails. The default value is false, which, instead of placing the server into maintenance mode when a server start fails, generates a runtime task. This runtime task contains a notification that the application placement controller did not receive notification about the completion of the server start.

Table 78. maintenanceModeOnOperationFail custom property values

Value	Description
Scope	Application placement controller
Valid values	true: Places the server into maintenance mode when the server start fails. false: Generates a runtime task when a server start fails.
Default	false

memoryProfiler.isDisabled:

You can use the memoryProfile.isDisabled custom property to disable the memory profiler.

Under the default operation, the application placement controller profiles dynamic clusters online to determine their memory usage. You can disable this default operation by setting the memoryProfiler.isDisabled custom property to true. When the memory profiler is disabled, the application

placement controller assumes a default memory usage value. This memory usage value is computed as $64k + 1.2 \text{ maxHeapSize}$, where `maxHeapSize` is the maximum heap size setting in the server template of the dynamic cluster.

Table 79. `memoryProfiler.isDisabled` custom property values

Value	Description
Scope	Application placement controller
Valid values	true: Disables the memory profiler. false: Enables the memory profiler.
Default	false

OverlappingCells:

You can use the `OverlappingCells` custom property when two Intelligent Management cells share the same physical host. Configuring this property ensures that the application placement controller does not overuse resources on the shared host.

For more information about setting custom properties, read about configuring application placement for cells that share the same nodes.

Table 80. `OverlappingCells` custom property values

Value	Description
Scope	Application placement controller
Valid values	Enter the custom property value in the following format: <code>user_name:password@deployment_manager_host:port.SOAP</code> If you are connecting three or more cells, separate the custom property values for each cell with a comma. <ul style="list-style-type: none"> • <code>user_name</code>: Specifies the name of the user that can log into the other cell. • <code>password</code>: Specifies the password for that user. You can encode the password so that it does not display in plain text in the administrative console. To encode a password, use the <code>encodePassword</code> script. • <code>deployment_manager_host</code>: Specifies the name of the deployment manager host for the other cell. • <code>port</code>: Specifies the SOAP port that is associated with the deployment manager, for example, the <code>SOAP_CONNECTOR_ADDRESS</code> port.

reservedMemoryFixed and reservedMemoryPercent:

You can use these custom properties to define how much memory on each node to reserve for processes that are not related to WebSphere Application Server or Intelligent Management.

When these properties are defined, the total memory on a node for processes that are not related to WebSphere Application Server or Intelligent Management can be calculated using the following formula:

$$\text{reservedMemoryFixed} + \text{reservedMemoryPercent} * (\text{totalNodeMemory} - \text{reservedMemoryFixed})$$

The `reservedMemoryFixed` custom property specifies a fixed amount of memory to reserve. The `reservedMemoryPercent` custom property specifies a percentage of memory to reserve.

Table 81. reservedMemoryFixed and reservedMemoryPercent custom property values

Value	Description
Scope	Application placement controller
Valid values	To set the value for the entire cell: Type the name of the custom property as reservedMemoryFixed or reservedMemoryPercent. To set the value for a particular node: Type the name of the custom property, either reservedMemoryFixed. <i>node_name</i> or reservedMemoryPercent. <i>node_name</i> , where <i>node_name</i> is the name of your node. For example, if your node is named node1, the names of your custom properties are reservedMemoryFixed.node1 and reservedMemoryPercent.node1.
Default	reservedMemoryFixed: 64 MB reservedMemoryPercent: 5%

unsetMaintenanceModeAfterStart:

Use this custom property to take a server out of maintenance mode after the server starts. If the maintenanceModeOnOperationFail custom property is enabled, the server is placed into maintenance mode after the server start fails. When the server starts successfully, you can use this property to move the started server out of maintenance mode automatically.

Table 82. unsetMaintenanceModeAfterStart custom property values

Value	Description
Scope	Application placement controller
Valid values	true: Moves the started server out of maintenance mode automatically. false: Leaves the server in maintenance mode after it is started.
Default	false

useNodeFreeMemory:

Use this custom property to define a replacement to the reservedMemory default custom property. This custom property uses the node free memory statistic, plus the memory calculated for any dynamic cluster instances that are already on the node, to calculate the memory available for launching additional dynamic cluster instances.

Table 83. useNodeFreeMemory custom property values

Value	Description
Scope	Application placement controller
Valid values	true: Automatically calculates available memory. false: Use reservedMemory default custom properties to define available memory.

Autonomic controllers custom properties

You can use custom properties to configure the autonomic controllers to always start on a particular node or deployment manager process.

To force a controller to always start on a deployment manager or a node agent process, set the `HAManagedItemPreferred_component_variable` custom property on the specific deployment manager or node.

Complete the following steps to set the custom property on a node agent process.

1. In the administrative console, click **System administration > Nodes > *node_name* > Node agent > Java and process management > Process definition > Java virtual machine > Custom properties**.
2. Enter the name of the custom property.
3. Set the value of the custom property to `true`.
4. Click **Apply**, and save your changes.

Complete the following steps to set the custom property on a deployment manager process.

1. In the administrative console, click **System administration > Deployment manager > Java and process management > Process definition > Java virtual machine > Custom properties**.
2. Enter the name of the custom property.
3. Set the value of the custom property to `true`.
4. Click **Apply**, and save your changes.

After you set each custom property, perform these steps:

- Restart the current process in which the controller is running. Skip this step if the custom property is used in the environment for the first time.
- Restart the Java virtual machine where the custom property was set.

Examples

To configure the application placement controller to start on a specific deployment manager or a node agent process, set the following custom property:

```
HAManagedItemPreferred_apc=true
```

If the cell is started after the custom property is set, the application placement controller starts at the specified deployment manager or node agent process. If the cell is already active when the custom property is set, the controller starts with the specified process only after the cell is stopped and started again.

When you use elasticity mode in an environment in which multi-cell performance management is configured, you must configure certain controllers to start on the deployment managers of the center cell and the point cells.

Note: When you are using elasticity mode, you must set the `HAManagedItemPreferred_apc` custom property on the deployment manager or node that will not be removed. Otherwise, the application placement controller does not operate.

To configure the autonomic request flow manager to start on a deployment manager or a node agent process, set the following custom property:

```
HAManagedItemPreferred_arfm=true
```

To configure the health management controller to start on a deployment manager or a node agent process, set the following custom property:

HAManagedItemPreferred_hmm=true

To configure the high availability plugin-cfg generation service to start on a deployment manager or a node agent process, set the following custom property:

HAManagedItemPreferred_PluginCfgGenerator=true

To configure the VMware activity publisher to start on a deployment manager or a node agent process, set the following custom property:

HAManagedItemPreferred_vmshimactivitypublisher=true

To configure the cell agent to start on the deployment managers of the point cells, set the following custom property:

HAManagedItemPreferred_cellagent=true

Autonomic request flow manager custom properties

You can use the following custom properties to change the behavior of the autonomic request flow manager (ARFM). Some custom properties are set on deployment targets.

CenterCell custom property:

When you are configuring multi-cell performance management in your environment, you can use the CenterCell custom property to designate one cell as the center cell. You also set the CenterCell custom property individually for each cell that you want to designate as a point cell.

Note: One and only one custom property should be set to true.

Table 84. CenterCell custom property values

Value	Description
Scope	Cell
Valid values	true: Designates one cell as the center cell false: Designates one cell as a point cell

enableRateBasedARFM:

ARFM uses a rate-based algorithm that results in a more consistent loading and protecting of application server resources. The rate-based ARFM feature is enabled by default. To disable the feature, set the enableRateBasedARFM custom property to false.

Table 85. AC5.BurstFactor custom property

Value	Description
Scope	Cell
Valid values	Boolean
Default	False

AC5.BurstFactor:

You can set this value to change how tolerant memory overload protection (MOP) is to bursts of traffic. If the maximum rate is one unit per second and the burst factor is one, then one unit of non-affinity traffic is admitted if it did not admit one in the last second. If the burst factor is three however, it admits three units every three seconds.

Table 86. AC5.BurstFactor custom property

Value	Description
Scope	On demand router
Valid values	Integer
Default	3

arfmIgnoreHttpSessionsForCpu:

You can set this custom property to true so that the ARFM ignores the HTTP dialog structure for the purposes of CPU overload protection. Every incoming HTTP message is subject to admission control.

Table 87. arfmIgnoreHttpSessionsForCpu custom property

Value	Description
Scope	Cell
Valid values	Boolean
Default	False

arfmIgnoreSipDialogsForCpu:

You can set this custom property to true so that the ARFM ignores the Session Initiation Protocol (SIP) dialog structure for the purposes of processor overload protection. Every incoming SIP message is subject to admission control.

Table 88. arfmIgnoreSipDialogsForCpu custom property

Value	Description
Scope	Cell
Valid values	Boolean
Default	False

arfmManageCpu:

CPU overload protection is on by default. Set the value of this custom property to false to disable CPU overload protection and request prioritization.

Table 89. arfmManageCpu custom property

Value	Description
Scope	Cell
Valid values	Boolean (true or false)
Default	True

arfmManualAllocation:

You can specify this custom property on a deployment target to put ARFM into manual mode when the custom property arfmMode is not set to manual. The value of the property is a string in a syntax that allows expression of allocations.

Table 90. *arfmManualAllocation* custom property

Value	Description
Scope	Deployment target (server, dynamic cluster, and so on)
Valid values	String
Default	No default

arfmMode:

You can use this custom property to specify the operating mode of the ARFM. In automatic mode, the ARFM has an autonomic controller that adjusts the dispatching priority of requests from the on demand routers to the servers and sends the placement controller information about the computational needs for the various dynamic clusters. In manual mode, you can override the autonomic controller with administrative settings.

Table 91. *arfmMode* custom property

Value	Description
Scope	Cell
Valid values	Manual, Automatic
Default	Automatic

CPUAdjustment:

You can set this custom property on nodes that are on hyperthreaded computers. When the CPUAdjustment custom property is specified on a node, a correction function to apply to processor utilization readings is specified for that node. This function is applied to the processor utilization that is reported by the operating system, and yields a more accurate utilization measure.

Table 92. *CPUAdjustment* custom property

Value	Description
Scope	Node
Valid values	String
Default	No default

goodServiceTimeLimitSpec:

Use a goodServiceTimeLimitSpec custom property when configuring the ARFM. The default value of one minute for determining timeouts is used if goodServiceTimeLimitSpec or timeoutFactor is not set. To override the default, set this custom property to customize the value for different levels of granularity. For example, service class, application, or module. To create a general rule of 5 minutes, create the custom property and set the value to `"*:*:*:*:=300"` where the last part is the timeout value of 300 seconds. The timeout value is a decimal number, not necessarily a whole number, and is in units of seconds.

Table 93. *goodServiceTimeLimitSpec* custom property

Value	Description
Scope	Cell
Valid values	Decimal number in unit of seconds
Default	60 if timeoutFactor is not set. Otherwise, no default.

magicNMode:

You can set this custom property when ARFM is in automatic mode. When in automatic mode, concurrency limiting is performed according to one of two criteria, depending if the magicNMode custom property is enabled. When the magicNMode custom property is enabled, the total number of requests that are running concurrently at any given time is limited to a certain total. In normal mode, the total number of requests depends on the use of processor power on the nodes.

Table 94. magicNMode custom property

Value	Description
Scope	Cell
Valid values	Boolean
Default	False

magicN:

You can use this custom property when ARFM has the magicNMode custom property is enabled, to the limit on the total number of concurrent requests. If this property is not defined, a reasonable limit is estimated from other data.

If you are using the reasonable limit estimation, the limit on the total number of concurrent requests for a cell is the sum of a contribution from each node in the cell. The contribution from a node is the average, over the running application servers on that node, of the maximum size of the thread pool of the Web container on that application server.

Table 95. magicN custom property

Value	Description
Scope	Cell
Valid values	A positive integer defines the number of concurrent requests. To use the reasonable estimate, set the property value to a negative number or delete the custom property.
Default	-1

maxHttpLiveRequests:

You can use this custom property to define the maximum number of requests that are targeted for a particular cell and could be anywhere in the cell. That is, either queued in an ODR or running on an application server at a given time.

Table 96. maxHttpLiveRequests custom property

Value	Description
Scope	Cell
Valid values	Integer
Default	No default

MOP.AveragingWindowInMS:

The averaging window in milliseconds that is used by MOP. This time interval should be at least as long as the lifetime of 95 percent of the HTTP sessions, SIP dialogs, and application sessions. If the time interval is too short, MOP might not protect against memory overload. If the time interval is much longer than the lifetime of these objects, MOP still protects against memory overload but MOP takes a longer amount of time to ramp up to learn the maximum rate.

Table 97. MOP.AveragingWindowInMS custom property

Value	Description
Scope	Cell
Units	milliseconds
Default	180000 ms

MOP.InitialMaxRatePerSec:

If no persisted maximum rate value exists, the value of this custom property is used as the initial maximum rate.

Table 98. MOP.AveragingWindowInMS custom property

Value	Description
Scope	Cell
Default	1

node.speed:

You can use this property to define the node speed in MHz. Set the value to the processor speed, multiplied by the number of processors on a multi-processor node. Set this custom property on the node.

Table 99. node.speed custom property

Value	Description
Scope	Node
Valid values	A floating point number that represents the number of MHz of the reference instruction set.
Default	No default

node.memory:

You can use this property to specify the available RAM defined in megabyte units on a node. Set this custom property on the node. Although this value is determined automatically, you can override the value.

Table 100. node.memory custom property

Value	Description
Scope	Node
Valid values	A number of megabytes
Default	No default

node.numCPUs:

You can use this custom property to specify the number of processors on a multi-processor node. This value is automatically determined, but you can override the value. Set this custom property on the node.

Table 101. node.numCPUs custom property

Value	Description
Scope	Node
Valid values	Integer that represents the number of processors

Table 101. *node.numCPUs* custom property (continued)

Value	Description
Default	No default

timeoutFactor:

Specify the *timeoutFactor* custom property to indicate a value that is multiplied by the response time threshold to determine the fine-grained timeout threshold. Use a *timeoutFactor* custom property when configuring the ARFM. The *timeoutFactor* value is multiplied by the response time threshold to yield the fine-grained timeout threshold.

This fine-grained timeout threshold is one of three different factors that the ARFM uses to determine that a running request has timed out. The running request has timed out when one of the following factors is true:

- The proxy framework indicates the request timed out.
- The request reply includes a special HTTP header that indicates that the internal application timed out.
- The service time exceeds the fine-grained timeout value.

Table 102. *timeoutFactor* custom property

Value	Description
Scope	Cell
Valid values	Decimal representation of a floating point number
Default	No default

useODRs:

Specifies if the work profiler and placement controller rely on the services of the ODR servers.

Table 103. *useODRs* custom property

Value	Description
Scope	Cell
Valid values	Boolean
DefaultRT	True

Autonomic request flow manager advanced custom properties

You can use these properties to configure the autonomic request flow manager (ARFM).

Work profiler properties:

profilerPeriod:

Specifies a number of milliseconds, specified for each cell that specifies the length of the work profiler cycle.

Table 104. *profilerPeriod* custom property values

Value	Description
Scope	Cell
Valid values	Integer value for number of milliseconds
Default	60000 (1 minute)

profilerHalfLife:

Specifies a number of minutes, specified for each cell. The work profiler discounts observations by an exponential function of time. The half life is the amount of time over which the discount changes by a factor of 2.

Table 105. profilerHalfLife custom property values

Value	Description
Scope	Cell
Valid values	Integer value for number of milliseconds
Default	600000 (10 minutes)

publishedAlphasPrintFrequency:

Specifies the frequency at which work factors are printed to the SystemOut.log file. These work factors are printed to the log file during the work profiler cycle, which is a length of time that is specified with the profilerPeriod custom property. If you want the work profiler to print work factors to the log file during each of these cycles, then you can specify the value as 1. However, if you want to decrease the amount of text being printed to the log file, you can increase this value. For example, if you want the work factors to be printed after every 5 work profiler cycles, then you can specify the value as 5.

Table 106. publishedAlphasPrintFrequency custom property values

Value	Description
Scope	Cell
Valid values	Integer value greater than or equal to zero
Default	0 (Work factors are not printed to the SystemOut.log file.)

Work profiler output decay half life and smoothing weight function parameters: The work profiler works in two passes: first it fits the observations to a simple model to extract preliminary work factors, then it smooths the work factors by taking a weighted average. Each weight is the product of two factors, one of which diminishes the preliminary work factor importance with age and the other that varies with the goodness of the first pass fit. The age factor is an exponential decay; the half-life is the amount of time over which that factor drops by a factor of 2. This parameter is given in the cell with the profilerAlphaSmoothingHalfLife custom property, with a value that is the decimal notation for an integer, a number of milliseconds. The default is 10 minutes. To adjust the goodness level, two parameters are used, a threshold and a factor. The threshold is defined with the goodnessWeightThresholdcell custom property. The factor is given in the cell with the goodnessWeightFactor custom property.

Table 107. Work profiler output decay half life and smoothing weight function parameters

Property name	Value	Default
profilerAlphaSmoothingHalfLife	a decimal notation for an integer in a number of milliseconds	600000 (10 minutes)
goodnessWeightThreshold	a non-negative floating point number	20
goodnessWeightFactor	a non-negative floating point number	20

Work factor overrides: You can override the values that are computed by the work profiler. The work profiler computes a work factor for each transaction class and deployed Java Platform, Enterprise Edition (Java EE) module pair (TCM). The work factor is a floating point number that represents the number of megacycles of the reference instruction set.

You can override work factors by adding the custom property to the dynamic cluster.

Use the following grammar for defining work factor override specifications:

```
spec ::= case ( " , " case ) *
case ::= pattern "=" value
pattern ::= service-class ":" txn-class ":" application ":" module
service-class ::= step
txn-class ::= step
application ::= step
module ::= step
step ::= name | "*"
value ::= number | "none"
```

Examples are provided in the following table:

Table 108. Work factor override specs

Example Spec	Description
::* = none	Specifies that every transaction class module (TCM) in the deployment target has no override. The deployment target has only one tier, and the value is calculated in the normal way for each case.
::* = 42	The deployment target has one tier. Every TCM in the deployment target has a work factor override for the tier, equal to 42 megacycles per request.
Platinum:*:* = 42, *.*:* = none	The deployment target has one tier. There is an override of 42 megacycles per request for transaction class modules that have a Platinum service class, and no overrides for transaction class modules that are assigned to any other service class in the deployment target.
:tc_A::=42, *:tc_B:AccountManagement:MicroWebApp.war=17, *:tc_B:*:=none	There is an override of 42 megacycles per request for TCMs that have the tc_A transaction class. For any TCMs that have the tc_B transaction class, a deployed Java EE application named AccountManagement, and a Java EE module named MicroWebApp.war, there is an override of 17 megacycles per request. There is no override for any other TCMs that have the tc_B transaction class. This example does not consider transaction classes other than the tc_A or tc_B transaction classes and if it encounters another transaction class, an error message is displayed.
::* = none, *.*:*:../DbCel/CICS = 0.7	There is no override for the first tier. For the tier that is named CICS+1, a work factor override of 0.7 exists. The CICS+1 tier is the first tier in the CICS® deployment target, in the DbCel cell, regardless of the target TCM. The transaction class does not change from tier to tier, but the module might change.

Use per-process CPU readings: Set the useProcessCPU custom property to true to enable the ARFM controller and application placement controller to consider background work when computing the required resources and enable per-process CPU utilization statistics. When this property is set to false, the work profiler cannot estimate work factors as well because it uses CPU utilization readings for the whole node.

If you configure this property a cell restart is required.

Table 109. useProcessCPU custom property

Name	Property Setting	Default	Valid Values
useProcessCPU	Set this custom property on the cell.	true	true or false

MustGather documents

Use the Intelligent Management mustGather documents to troubleshoot the autonomic request flow manager and application placement. For more information, read about the mustGather support documents for each version of Intelligent Management.

Binary trace facility custom properties

You can modify trace dynamically and persistently by setting a cell custom property.

Set cell custom property with the following form for name and value:

Table 110. trace custom property

Value	Description
Scope	Cell
Name	trace.<scopeName>
Valid values	[<condition>;;<traceSpec>

Examples of changing trace dynamically

You can modify trace dynamically and persistently by setting a cell custom property. For example, to trace HTTP affinity processing and load balancing, use the following settings:

Table 111. trace HTTP affinity processing and load balancing

Value	Description
Scope	Cell
Name	trace.http
Valid values	http.request.affinity,http.request.loadBalance

Or, to trace server weight changes (which are traced by default), specify the following settings:

Table 112. trace HTTP affinity processing and load balancing

Value	Description
Scope	Cell
Name	trace.ODCEvent
Valid values	propertyType = 'server:weight';;;ODC.event

The ODC language supports a `nodeType` operand, `propertyType` operand which returns `<nodeName>:<propertyName>`, and `edgeType` operand which returns `<parentNodeType>:<childNodeType>`. All ODC events are in the group `ODC.event`, so as long as they pass the condition test, they will be traced.

To enable all HTTP trace except load balancing, turn on all HTTP trace, then turn off `http.request.loadBalance` trace with a trace level of 0, as shown in the following table:

Table 113. trace HTTP without load balancing

Value	Description
Scope	Cell
Name	trace.http
Valid values	http,http.request.loadBalance=0

To trace all SIP messages that take longer than 100 milliseconds to be processed by the on demand router (ODR), specify the following settings:

Table 114. trace HTTP without load balancing

Value	Description
Scope	Cell
Name	trace.sip
Valid values	sip.message.slow=100

Specifying these settings traces all SIP messages that take longer than 100 milliseconds to be processed by the ODR. Note that in this case, the trace level field is used as a time threshold field that controls when the message is traced. The record description describes how the level value is used in this case: A SIP message took longer than the level (in milliseconds) to be processed by the ODR. The default level is 300. Record descriptions should be clear in describing how the level is used and what each level means for the record.

Dynamic cluster custom properties

You can use dynamic cluster custom properties to change the behavior of your dynamic clusters and application placement.

APC.predictor custom property:

- | Use the APC.predictor custom property to enable the application placement controller function.
- | By setting the custom property, the application placement controller starts and stops servers based on CPU usage alone. The controller no longer retrieves data from the autonomic request flow manager (ARFM) regarding what servers to start and stop.

Table 115. APC.predictor custom property values

Value	Description
Scope	Cell
Valid value	CPU

quiesceTimeOutMS custom property:

You can use the quiesceTimeOutMS custom property to set the quiesce timeout value for dynamic cluster instances.

Set the value of the custom property to the amount of time, in milliseconds, to wait before a dynamic cluster is stopped. For example, if you want a dynamic cluster to stop after 1 minute of quiescing, set the value to 60000. If servers are being stopped by the application placement controller, the server operation timeout value is used by default. If servers are being stopped by health management monitoring, the restart timeout value is used by default.

Table 116. quiesceTimeOutMS custom property values

Value	Description
Scope	Dynamic cluster
Valid values	Integer

CenterCell custom property:

When you configure multi-cell performance management in your environment, you can use the CenterCell custom property to designate one cell as the center cell. You also set the CenterCell custom property individually for each cell that you want to designate as a point cell.

Note: One and only one custom property should be set to true.

Table 117. CenterCell custom property values

Value	Description
Scope	Cell
Valid values	true: Designates one cell as the center cell false: Designates one cell as a point cell

lazyStartMinInstances custom property:

You can use the lazyStartMinInstances custom property to configure multiple server instances to start when the ODR detects activity for an inactive dynamic cluster.

Prior to Version 6.1.1.2, only one server instance was started when a dynamic cluster was configured for application lazy start and the ODR received requests for the inactive dynamic cluster. By setting the lazyStartMinInstances custom property on a specific dynamic cluster, however, multiple instances can be started by that particular dynamic cluster. If other dynamic clusters configured for application lazy start exist, those clusters start only one instance each.

Alternatively, you can set the custom property on the application placement controller so that all of your dynamic clusters are affected. However, the custom property value set at the dynamic cluster level overrides the custom property value set at the application placement level.

Table 118. lazyStartMinInstances custom property values

Value	Description
Scope	Dynamic cluster
Valid values	An integer value that specifies the minimum number of instances to be lazily started
Default	1

equalCPUFactor custom property:

You can use the equalCPUFactor custom property to tell to the dynamic workload manager (DWLM) how to equalize the performance of the servers in a dynamic cluster.

The DWLM calculates dynamic weights for the servers in a given dynamic cluster to equalize the performance of the servers. The two common measures of server performance are:

- Average service time for a request that is sent to the server.
- The processor utilization of the node.

With the custom property, you can specify the relevance of one of these measures over the other. For example, if you want consistency in how the users of your site perceive the site performance, you might choose to optimize average service time. If you are more concerned with the usage of your hardware, you might choose processor utilization as your measure of performance.

To place most priority on equalizing average service time, set the value of the custom property to 0. To equalize processor utilization of the node, set the value to 1. To use a blend of both measures with a relative weight to each measure, set the value to a fraction value between 0 and 1, for example 0.4. By

setting the value to a fraction, you are placing a relative weight of 0.4 to the processor utilization and 1-0.4, or 0.6 relative weight to the average service time.

Equalizing both measures at the same time might not be possible at all times. For example, in an environment where servers are heterogeneous or have varying amount of unequal background load, equalizing the processor utilization might result in unequal average service time. A fast and a slow server running at the same processor utilization can result in short and long average request service time, respectively. A request that spends a significant amount of time in one of several servers in a deeper tier might result in a different average service time. This variance can occur depending on the server to which the request was sent, even if the servers in the deeper tier are homogeneous and have equal processor utilization. Other situations exist where the service time of a request depends on resources other than the processor. The value of the `equalCPUFactor` custom property helps the DWLM controller determine a weighted measure of both average service time and processor utilization to be equalized.

Even without the `equalCPUFactor` custom property, the processor utilization of the servers in a given dynamic cluster has an effect on the behavior of the DWLM controller. In general, when processor utilization is low, equal distribution of load takes precedence over equalizing the performance of service time or utilization. Gradually, as the utilization goes up, equalizing performance begins to take precedence. At very high processor utilization values, the weights tend to not change as much to avoid instability. When processor utilization is high, the sensitivity of performance on load distribution at that extreme operating point increases.

Table 119. equalCPUFactor custom property values

Value	Description
Scope	Cell, which applies to all dynamic clusters in the cell, or at the individual dynamic cluster level. If the custom property is specified at both the dynamic cluster and cell level, dynamic cluster level value overrides the value that is specified at the cell level.
Valid values	A fraction value between 0 and 1
Default	0 for non-virtualized environments and 1 for virtualized environments

HttpSessionRebalanceOff custom property:

You can use the `HttpSessionRebalanceOff` custom property to disable HTTP session rebalancing.

HTTP session rebalancing is automatically enabled. You can use HTTP session rebalancing to reassign existing session affinities to new servers that become available for processing the given Web application. For more information, read about HTTP session rebalancing

Use the `HttpSessionRebalanceOff` custom property if you want to return your configuration to the old HTTP session behavior, where session affinities are established with a particular application server and are not reassigned to any new servers that become available.

Session rebalancing is disabled by default on any dynamic clusters that consist of servers that are not running WebSphere Application Server, such as PHP or Tomcat servers, because you might have another session clustering mechanism deployed for those servers.

You might consider disabling HTTP session rebalancing if your session sizes are large. If your sessions are large, the cost of moving the sessions to a new server might be more than the benefit of taking the workload off of the original server. You can use Performance Monitoring Infrastructure (PMI) data to make the decision to turn off session rebalancing. You might see in your PMI data that response time, memory

utilization, and processor utilization increases on specific servers to transfer the session information. For more information about analyzing PMI data and best practices for using HTTP sessions, read about Performance Monitoring Infrastructure (PMI).

If you leave session rebalancing on, as sessions become more evenly distributed, and memory and processor utilization also become more evenly distributed across the servers in the cluster. If a cluster is more balanced, it is easier for Intelligent Management to make autonomic decisions.

Table 120. *HttpSessionRebalanceOff* custom property values

Value	Description
Scope	Dynamic cluster
Valid values	true: Disables HTTP session rebalancing. false: Enables HTTP session rebalancing. If you want to disable HTTP session rebalancing for WebSphere Application Server application servers, you can delete the custom property.
Default	For dynamic clusters that consist of WebSphere Application Server application servers: false (enabled) For dynamic clusters that consist of servers that are not WebSphere Application Server application servers, such as PHP or Tomcat servers: true (disabled)

***numVerticalInstances* custom property:**

Use this custom property to define the number of dynamic cluster instances on a node.

Use this custom property only if the nodes in your dynamic cluster are heterogeneous and vary in power. If the nodes in your dynamic cluster are homogenous, you can define the number of dynamic cluster instances one time in the administrative console. For more information about dynamic cluster instances, read about configuring a dynamic cluster with heterogeneous nodes to support vertical stacking.

Table 121. *numVerticalInstances* custom property values

Value	Description
Scope	Dynamic cluster
Name format	Specify the name of the custom property as numVerticalInstances.node_name, where node_name is the name of your node.
Valid values	Integer value for the stacking number

***proactiveIdleStop* custom property:**

You can use the proactiveIdleStop custom property to stop dynamic cluster instances during periods of inactivity.

This custom property adds function to the **If other dynamic clusters need resources, stop all instances of this cluster during periods of inactivity** setting in the administrative console. This setting must be enabled with this custom property. With the administrative console setting, instances stop only if other clusters in the cell need the resources that are being used by the inactive instances. You also specify an amount of time to wait before stopping instances for the cluster. By setting this custom property, inactive instances stop even if the resources are not required elsewhere in the environment. The cluster instance goes idle in the amount of time that you specified in the administrative console setting.

The application placement controller stops the instance at some point in time between the amount of time that you specified in the administrative console setting plus the value specified for **Minimum time between placement changes** setting on the Application Placement Controller configuration panel. For example, if you set the `proactiveIdleStop` custom property to `true`, the value of **Minimum time between placement changes** is 15 minutes, and the **Time to wait before stopping instances** value is 5 minutes, the dynamic cluster instance is stopped sometime between 5 minutes and 20 minutes after the last request was routed to the instance.

Table 122. `proactiveIdleStop` custom property values

Value	Description
Scope	Dynamic cluster
Valid values	Boolean
Default	<code>false</code> (disabled)

`serverSpecificShortNames` custom property: z/OS

For z/OS platforms, the `serverSpecificShortNames` custom property is specified on the dynamic cluster to indicate the specific short names of cluster members in a comma-separated list format, for example: `SSN1,SSN2`. Use commas to separate multiple short names. If you do not provide enough short names to be used for all of the cluster members, then the remaining cluster members are assigned generated generic short names, such as `BBOS001`, `BBOS002`, and so on.

Table 123. `serverSpecificShortNames` custom property values

Value	Description
Scope	Dynamic cluster
Valid values	Comma-separated list of short names for dynamic cluster members, for example <code>SSN1,SSN2</code>

`updateWLM` custom property: When this property is set to `false`, the DWLM controller will not update the calculated weights for the cluster members in the Work Load Management (WLM). The default value of this property is `true`. Cell recycle is required for this custom property to take effect. It is recommended that `updateWLM` is set to `false` when `HAManager` is turned off through out the cell or on all the dynamic cluster members.

Note: This custom property can be set at dynamic cluster level and cell level:

- If set at cell level, it applies to all dynamic clusters in the cell.
- If set at DC level, it applies to that DC only.

Table 124. `updateWLM` custom property values

Value	Description
Scope	Dynamic cluster or cell
Valid values	<code>true</code> or <code>false</code>
Default	<code>true</code>

Health controller custom properties

You can use health controller custom properties to change the behavior of your health controller. For example, you can change the amount of time that passes before a runtime task that is generated for a health issue expires.

`usexdHeapModule`:

You can set the `usexdHeapModule` custom property so that only the amount of free heap after a full garbage collection is sampled. Otherwise, the heap might include temporary Java objects.

In releases prior to Version 7.0, the excessive memory health policy calculated the amount of heap used by a Java virtual machine (JVM) by periodically sampling the amount of free heap.

Table 125. `usexdHeapModule` custom property valid value and default

Value	Description
Scope	Cell
Valid values	<p><code>true</code>: Only the amount of free heap after a full garbage collection is sampled.</p> <p><code>false</code>: The amount of free heap is periodically sampled independent of the garbage collection cycle.</p>
Default value	<code>false</code>

`_CEE_DMPTARG`:

The `_CEE_DMPTARG` custom property specifies the location that is used for thread dumps on z/OS systems.

You must set the `_CEE_DMPTARG` custom property value if you configure health policies that perform thread dumps on z/OS systems. This custom property sets the location where the thread dumps occur for a particular server.

Table 126. `_CEE_DMPTARG` custom property valid value and default

Value	Description
Scope	Application server
Valid value	A string that indicates the location where you want to save the thread dumps

`com.ibm.ws.xd.hmm.controller.ControlConfig.approvalTimeOutMinutes`:

The `com.ibm.ws.xd.hmm.controller.ControlConfig.approvalTimeOutMinutes` custom property specifies the number of minutes for the approval timeout of runtime tasks for the health controller.

Set the Java virtual machine (JVM) custom property when you are using the supervise reaction mode. This property specifies the number of minutes that can pass before a runtime task for the health controller expires. If you set the value to 5 minutes or less, the default value of 30 minutes is automatically used instead. If you do not take any actions on the runtime task, the task expires in the number of minutes that is specified in this property. If the runtime task expires when the health condition still exists, a new runtime task is generated.

Table 127. `com.ibm.ws.xd.hmm.controller.ControlConfig.approvalTimeOutMinutes` property valid value and default

Value	Description
Scope	JVM
Valid value	Minutes
Default	30 minutes

Service policy custom properties

You can use service policy custom properties to provide service policy alerting for persistent service policy violations on a transaction class basis. When a violation is reported, a runtime task is generated by a runtime component within Intelligent Management. A task provides information from which you can accept or deny the suggested action plan.

disableResponseTimeGoals:

Use the `disableResponseTimeGoals` custom property to configure service policies without response time goals when multiple service policies are configured, and the response time goals cannot be accurately determined for any of the service policies.

Providing correct response time goals requires that load testing be performed to determine a reasonable value. However, you can configure service policies without response time goals to eliminate the need for performance testing. To determine when a new dynamic cluster instance should be started when there are no response time goals, a utility value is calculated based upon the fraction of time that requests wait. The utility value is a value between -1 and +1, based on the importance of a service class. Requests with zero wait time receive a +1 utility value, and requests with almost all time spent waiting receive a -1 utility value. A utility value of zero is obtained if the fraction of wait time equals the relative importance of the request service class, where relative importance is zero for very high importance and one for very low importance.

Table 128. `disableResponseTimeGoals` custom property

Value	Description
Scope	Cell
Value	true

detectTxcViolations:

Use the `detectTxcViolations` custom property when you want persistent service policy violations to be shown at the transaction class level.

The autonomic request flow manager (ARFM) will use the properties you specify using this custom property. To enable the monitoring and reporting of persistent service policy violations by a transaction class (defined within a service class), set the `detectTxcViolations` custom property to true. For the runtime task to be generated and enabled for monitoring, this custom property must be set to true, and the service policy must be enabled through the administrative console.

Table 129. `detectTxcViolations` custom property

Value	Description
Scope	Cell
Value	true
Default	false

detectSvcViolations:

You can use the `detectSvcViolations` custom property when you want persistent service policy violations to be shown at the service class level. You can set the value of `detectSvcViolations` to false in order to disable runtime tasks for persistent service policy violations at the service class level, while enabling the runtime tasks for violations at the transaction class level by setting `detectTxcViolations` to true.

The autonomic request flow manager (ARFM) uses the properties you specify using this custom property. Otherwise, it will monitor for only service class violations. To enable the monitoring and reporting of service

policy violations within a service class, set the `detectSvcViolations` custom property to `true`, or allow it to keep its default value of `true`. For the runtime task to be generated and enabled for monitoring, this custom property must be set to `true`, and the service policy must be enabled through the administrative console

Table 130. `detectSvcViolations` custom property

Value	Description
Scope	Cell
Value	<code>true</code>
Default	<code>true</code>

Node agent custom properties

You can use custom properties to modify the node agent configuration, and to configure timeout values for the node agent.

ServerStartupTimeoutMillis:

The `ServerStartupTimeoutMillis` custom property specifies the amount of time that the node agent waits, during the middleware server startup, before the node agent times out and stops trying to start the server. If the middleware server starts before the timeout value, the agent stops waiting and continues with the startup processing.

Table 131. `ServerStartupTimeoutMillis` custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level
Default	75000 (75 seconds)
Units	milliseconds

ServerShutdownTimeoutMillis:

The `ServerShutdownTimeoutMillis` custom property specifies the amount of time that the node agent uses to determine whether a server has successfully stopped.

Table 132. `ServerShutdownTimeoutMillis` custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level
Default	75000 (75 seconds)
Units	milliseconds

MiddlewareAgentProcessScanIntervalMillis:

The `MiddlewareAgentProcessScanIntervalMillis` custom property specifies a time interval at which the node agent scans the configured servers to reestablish monitors for all the running middleware servers. If you set the value to zero, the repeated scans are disabled.

Table 133. `MiddlewareAgentProcessScanIntervalMillis` custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level
Default	60000 (60 seconds, or one minute)
Units	milliseconds

ServerInitializedTimeoutMillis:

The `ServerInitializedTimeoutMillis` custom property is the amount of time that a node agent uses to determine if a middleware server is started or not, after performing the start server operation and binding to the middleware server process. If the node agent cannot determine if the server is running, the default assumption is that the middleware server is not running.

Table 134. ServerInitializedTimeoutMillis custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level
Default	30000 (30 seconds)
Units	milliseconds

MiddlewareServerStartWaitTime:

The `MiddlewareServerStartWaitTime` custom property is used to configure the middleware server initialization time. After a middleware server starts, additional time for the server to initialize is required. If the node agent reports that the middleware server has started too soon, HTTP requests that are directed to that middleware server might fail. You can use this custom property so that the node agent does not report that the server has started for a fixed period.

Table 135. MiddlewareServerStartWaitTime custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level
Default	75000 (75 seconds)
Units	milliseconds

WASCEAppStatusRetryCount:

The `WASCEAppStatusRetryCount` custom property allows you to configure the number of retries when running the WASCE command to determine if an application is started on the WebSphere Application Server Community Edition server.

Table 136. WASCEAppStatusRetryCount custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level
Default	3
Units	retry

WASCEAppStatusRetryDelayTime:

The `WASCEAppStatusRetryDelayTime` custom property allows you to configure the delay between retries when running the WASCE command to determine if an application is started on the WebSphere Application Server Community Edition server.

Table 137. WASCEAppStatusRetryDelayTime custom property default, units, and scope

Value	Description
Scope	Cell, node, or server level

Table 137. WASCEAppStatusRetryDelayTime custom property default, units, and scope (continued)

Value	Description
Default	10000 (10 seconds)
Units	milliseconds

Middleware server custom properties and variables

Middleware server variables and custom properties define properties for middleware servers, such as the maximum heap size for assisted lifecycle servers.

SERVER_MAX_HEAP_SIZE variable:

The *SERVER_MAX_HEAP_SIZE* variable defines the maximum heap size for assisted lifecycle servers. If the variable is not defined, the default maximum heap size value for the server type is used.

You can use **wsadmin** scripting to set the *SERVER_MAX_HEAP_SIZE* variable. You must use it to set the variable for Apache Tomcat, JBoss application servers, BEA WebLogic servers, and external WebSphere Application Server. Follow the directions in the topic on modifying variables using **wsadmin** scripting.

To set the *SERVER_MAX_HEAP_SIZE* variable using the administrative console, click **Servers > All servers > server_name > Variables**. Specify the maximum heap size in megabytes.

Note:

When the excessive memory usage health policy is triggered for WebSphere Application Server Community Edition servers, the percentage of the heap size specified in the runtime task message is inaccurate.

Configure the maximum heap size for each WebSphere Application Server Community Edition server when you configure the excessive memory usage health policy for WebSphere Application Server Community Edition servers.

Set the *SERVER_MAX_HEAP_SIZE* variable at the server level. The default value is 64 MB.

In the administrative console, click **Servers > All servers > Variables**.

For more information, read the topic on middleware server custom properties and variables.

Table 138. Default values for *SERVER_MAX_HEAP_SIZE* variable

Server type	Maximum heap size
JBoss	512 MB
Apache Tomcat	64 MB
WebSphere Application Server Community Edition	64 MB
BEA WebLogic	400 MB

UseJMXSecureConnector custom property:

The *UseJMXSecureConnector* custom property specifies whether Java Management Extensions (JMX) security is enabled for a WebSphere Application Server Community Edition Version 2.1 server or later server.

To enable JMX security, additional steps are required beyond setting the custom property. Read about the additional steps in the topic on Configuring JMX security for WebSphere Application Server Community Edition.

To set the UseJMXSecureConnector custom property, click **Servers > Other middleware servers > WebSphere Application Server Community Edition servers > wasce_server_name > Custom properties > New**. Set the value to true to enable JMX security.

Table 139. UseJMXSecureConnector custom property

Value	Description
Value	True or false
Default	false

Operational alerts custom properties

You can use the operational alerts custom properties to improve the performance of the operational alerts on your runtime reporting and operational summary pages.

You can tune the performance of your operational alerts by setting any of the following custom properties on the cell. Set the properties that you want to a value that is different from the default value.

To set the operational alerts custom properties on the cell:

1. In the administrative console, click **System administration > Cell > Custom properties > New**.
2. Type the name of the custom property in the format, `opalert.property_name`, for example, `opalert.disableCache`.
3. Indicate a value for the custom property. Use the possible values in the following list of custom properties to determine the value that you use.
4. Save the custom property. The custom property is retrieved dynamically. You do not need to restart the deployment manager. The custom property is set for all the resources in the cell.

To set a custom property on a specific resource:

1. In the administrative console, click **System administration > Cell > Custom properties > New**.
2. Type the name of the custom property in the format, `opalert.resource_typeproperty_name`. The `resource_type` value is the name of the core group or core component on which you want to set the property. Use the name for the core component or core group that is listed in the administrative console. For example, you might use the name, `opalert.ARFMController.disableCache`, to set the property on the autonomic request flow manager (ARFM).
3. Indicate a value for the custom property. Use the possible values in the following list of custom properties to determine the value that you use.
4. Save the custom property. The custom property is retrieved dynamically. You do not need to restart the deployment manager. The custom property is used only for the resource that you indicated.

opalert.disableCache:

When set to true, this custom property disables caching, which also disables any performance gains. You can use this custom property when you are debugging.

Table 140. *opalert.disableCache* custom property

Value	Description
Value	True or false
Default	false

opalert.disableAlerts:

When set to true, this custom property disables the cacheable providers from adding operational alerts. To view detailed information about items such as core groups and core components, go to the operations view that is specific to the core group or core component.

Table 141. opalert.disableAlerts custom property

Value	Description
Value	True or false
Default	false

opalert.disableXDStability:

When set to true, this custom property disables the cacheable provider from contributing to the overall status. To view detailed information about items such as core groups and core components, go to the operations view that is specific to the core group or core component.

Table 142. opalert.disableXDStability custom property

Value	Description
Value	True or false
Default	false

opalert.initPolicy:

This custom property determines if the request waits for the initialization of the cache to complete before returning a status message. The first request for the information causes caching to occur. If the initialization policy is set to blocking, the initialization of the cache must complete before returning a status message. If the initialization policy is set to nonblocking, then an UNKNOWN status message is returned. You can refresh the operational alerts page or operations view when the cache has finished initializing. This property is only used when the cache initializes.

Table 143. opalert.initPolicy custom property

Value	Description
Value	blocking or nonblocking
Default	nonblocking

opalert.disableAutoRefresh:

By default, the caches are automatically refreshed on a time interval. If this property is set to true, the caches are not automatically refreshed, and do not refresh until a new request comes in after the caches have been destroyed.

Table 144. opalert.disableAutoRefresh custom property

Value	Description
Value	true or false
Default	false

opalert.initPeriod:

If the `opalert.disableAutoRefresh` property is enabled, this custom property defines how long to wait after an initialization is completed to start initialization again.

Table 145. `opalert.initPeriod` custom property

Value	Description
Value	Integer, in seconds
Default	10 seconds

`opalert.destPolicy`:

The destruction policy specifies when to destroy the provider caches.

- Specify `inactivity` to destroy the caches if the cache is not accessed for the amount of time that is specified in the `opalert.destPeriod` custom property.
- Specify `fixedTimeout` to destroy the cache after the amount of time that is specified in the `opalert.destPeriod` custom property passes after the first initialization period. This value does not consider when the last cache access time is. If you have the `opalert.disableAutoRefresh` enabled, consider setting this custom property to `fixedTimeout`.

Table 146. `opalert.destPolicy` custom property

Value	Description
Value	<code>inactivity</code> or <code>fixedTimeout</code>
Default	<code>inactivity</code>

`opalert.destPeriod`:

This custom property specifies the destruction period in seconds. The destruction period works with the `opalert.destPolicy` custom property to determine how long to wait, either after the first initialization or the last access, before destroying the cache.

Table 147. `opalert.destPeriod` custom property

Value	Description
Value	seconds
Default	600 seconds (10 minutes)

Runtime operations custom properties

You can use the runtime operations custom properties to modify runtime reports settings of your multi-cell environment.

You can modify how runtime operations reports for multiple cells are displayed by setting any of the following custom properties on the cell. If you want to use a value other than default, create a custom property.

`cellExclusion`:

The `cellExclusion` custom property specifies a list of cells to exclude from the operations dashboard view. You cannot exclude the local cell. If you do not set the `cellExclusion` custom property, the default is to include all cells.

Table 148. `cellExclusion` custom property

Value	Description
Scope	Cell

Table 148. *cellExclusion* custom property (continued)

Value	Description
Value	A list of cell names separated by commas.

maxApplicationSummaryDataSets:

The *maxApplicationSummaryDataSets* custom property specifies the maximum number of data sets that are displayed in the application summary chart. If you do not set the *maxApplicationSummaryDataSets* custom property, the default number of data sets that are displayed is 5.

Table 149. *maxApplicationSummaryDataSets*

Value	Description
Scope	Cell
Value	Integer
Default	5

maxDeploymentTargetSummaryDataSets:

The *maxDeploymentTargetSummaryDataSets* custom property specifies the maximum number of data sets that are displayed in the deployment target summary chart. If you do not set the *maxDeploymentTargetSummaryDataSets* custom property, the default number of data sets that are displayed is 5.

Table 150. *maxDeploymentTargetSummaryDataSets* custom property

Value	Description
Scope	Cell
Value	Integer
Default	5

maxServicePolicySummaryDataSets:

The *maxServicePolicySummaryDataSets* custom property specifies the maximum number of data sets that are displayed in the service policy summary chart. If you do not set the *maxServicePolicySummaryDataSets* custom property, the default number of data sets that are displayed is 5.

Table 151. *maxServicePolicySummaryDataSets* custom property

Value	Description
Scope	Cell
Value	Integer
Default	5

topDataSetInterval:

The *topDataSetInterval* custom property specifies how many milliseconds of statistics are used to calculate the top number of data sets to display in the summary charts. If you do not set the *topDataSetInterval* custom property, the last 120000 milliseconds, or 2 minutes, of statistics are used by default. If you set the custom property to 0 milliseconds, the data sets are returned in the order they are received by the on demand configuration (ODC).

Table 152. *topDataSetInterval* custom property

Value	Description
Scope	Cell
Value	Integer
Default	Last 120000 milliseconds

destroyStatisticCacheDelay:

The *destroyStatisticCacheDelay* custom property specifies the delay in milliseconds between when Intelligent Management detects zero active sessions in the administrative console and when the statistic cache is destroyed. If no custom property exists, the default delay is set 0 milliseconds (0 seconds).

Table 153. *destroyStatisticCacheDelay* custom property

Value	Description
Scope	Cell
Value	Integer
Default	0 milliseconds

On demand router system and custom properties

You can use on demand router (ODR) custom properties to change the behavior of your ODR. For example, you can change the error code that the ODR returns when messages are rejected because of processor or memory overload.

***http.log.history* custom property:**

Use the *http.log.history* custom property to specify the number of history server log files. The server log files are the ODR custom log files, as well as the *proxy.log*, *local.log*, and *cache.log* files. Without this custom property, you have one history file for the server log files. Setting this custom property does not affect the custom logs that are already initialized.

Table 154. *http.log.history* custom property values

Value	Description
Scope	ODR or ODR cluster
Value	Any positive integer greater than 0
Default	1

***http.log.maxSize* custom property:**

Use the *http.log.maxSize* custom property to specify the maximum log size in megabytes (MB). A value of UNLIMITED indicates unlimited. As with the *http.log.history* custom property, the *http.log.maxSize* custom property applies to the ODR custom log files, as well as the *proxy.log*, *local.log*, and *cache.log* files

Table 155. *http.log.maxSize* custom property values

Value	Description
Scope	ODR or ODR cluster
Value	Any positive integer greater than 0
Default	25 MB

***CenterCell* custom property:**

When you are configuring multi-cell performance management in your environment, you can use the CenterCell custom property to designate one cell as the center cell. You also set the CenterCell custom property individually for each cell that you want to designate as a point cell.

Note: One and only one custom property should be set to true.

Table 156. CenterCell custom property values

Value	Description
Scope	Cell
Valid values	true: Designates one cell as the center cell false: Designates one cell as a point cell

ODR.heapUsage.max:

Use the ODR.heapUsage.max custom property to configure a percentage value that determines at what amount of heap usage the ODR rejects requests. If heap usage exceeds 90%, the ODR rejects any incoming request and a 503 error code is returned.

Table 157. ODR.heapUsage.max custom property

Value	Description
Scope	Cell
Value	Percentage
Default	90%

ODR.heapUsage.algorithm:

Use the ODR.heapUsage.algorithm custom property to configure when heap usage is calculated. If you set the value to globalGC, the amount of heap usage is calculated only immediately after a global garbage collection cycle. If you set the value to request, the amount of heap usage is calculated with each request.

Table 158. ODR.heapUsage.algorithm custom property

Value	Description
Scope	Cell
Value	Algorithm
Default	globalGC

ODR.heapUsage.errorCode:

Use the ODR.heapUsage.errorCode custom property to configure the type of error code that is returned when a request is received and the heap usage exceeds the maximum threshold.

Table 159. ODR.heapUsage.errorCode custom property

Value	Description
Scope	Cell
Value	Integer
Default	503

http.overload.error:

Use the `http.overload.error` custom property to configure a custom error code for the autonomic request flow manager to return when an HTTP message over TCP or SSL is rejected due to processor or memory overload. If you do not configure this custom property, the default 503 error code is returned.

Table 160. `http.overload.error` custom property

Value	Description
Scope	On demand router
Value	Integer
Default	503

sip.overload.error:

Use the `sip.overload.error` custom property to configure a custom error code for the autonomic request flow manager to return when a Session Initiation Protocol (SIP) message over Transmission Control Protocol (TCP) or Secure Sockets Layer (SSL) is rejected because of processor or memory overload. If you do not configure this custom property, the default 503 error code is returned.

Table 161. `sip.overload.error` custom property

Value	Description
Scope	On demand router
Value	Integer
Default	503

sipu.overload.error:

Use the `sipu.overload.error` custom property to configure a custom error code for the autonomic request flow manager to return when a Session Initiation Protocol (SIP) message over User Datagram Protocol (UDP) is rejected due to processor or memory overload. If you do not configure this custom property, the default 503 error code is returned.

Table 162. `sipu.overload.error` custom property

Value	Description
Scope	On demand router
Value	Integer
Default	503

http.partialResponseBodyBufferSize:

Use the `http.partialResponseBodyBufferSize` custom property to disable the ODR from buffering the size of the response payload when you perform a rollout on an application edition. Set the `http.partialResponseBodyBufferSize` custom property to a value of 0.

Table 163. `http.partialResponseBodyBufferSize` custom property

Value	Description
Scope	On demand router
Value	Integer
Default	0

System properties:

Follow the instructions to set specific system properties for the ODR.

From the administrative console, select **Servers > On Demand Routers > on_demand_router > Java and Process Management > Process Definition > Java Virtual Machine.**

ODCLeftGroupDelay:

Use the `ODCLeftGroupDelay =0` system property set to allow no delay for on demand configuration detection of down servers during failover when servers become unavailable.

ODRtraceRouteHeader:

Use the `ODRtraceRouteHeader =0` system property on the ODR, and it will add a response header with the route that the request took.

For example, `-DODR.traceRouteHeader=TraceRoute` adds a response header with the name `TraceRoute` to each request. The value of this response header is the path that the request took.

updateWLM: When this property is set to `false`, the DWLM controller will not update the calculated weights for the cluster members in the Work Load Management (WLM). The default value of this property is `true`. Cell recycle is required for this custom property to take effect. It is recommended that `updateWLM` is set to `false` when `HAManager` is turned off through out the cell or on all the dynamic cluster members.

Note: This custom property can be set at dynamic cluster level and cell level:

- If set at cell level, it applies to all dynamic clusters in the cell.
- If set at DC level, it applies to that DC only.

Table 164. updateWLM system property values

Value	Description
Scope	Dynamic cluster or cell
Valid values	true or false
Default	true

Runtime task custom properties

You can use runtime task custom properties to customize the e-mails that contain runtime task information.

Requirement: For older WebSphere Virtual Enterprise versions, install the appropriate fix packs in order to use the custom properties:

terminated.task.remove.timeout:

You can use the `terminated.task.remove.timeout` custom property to change the time that expired task entries or actioned tasks stay in the runtime task list, which is 24 hours by default. The value is expressed in milliseconds. For example, 600000 milliseconds = 10 minutes.

Table 165. terminated.task.remove.timeout custom property values

Value	Description
Scope	Cell
Valid values	milliseconds
Default	24 hours

task.email.global.sender.id:

You can use the `task.email.global.sender.id` custom property to change the e-mail address from which e-mail notifications are sent. Set the value to the specific user ID that you want to use to sent e-mails.

Table 166. `task.email.global.sender.id` custom property values

Value	Description
Scope	Cell
Valid values	user ID
Default	wasxd

task.email.custom.body:

You can set the `task.email.custom.body` custom property to customize the details of task notifications in the generated runtime task e-mails.

Table 167. `task.email.custom.body` custom property values

Value	Description
Scope	Cell
Valid values	<p><code>false</code>: Deactivated. Customized information for task notifications is not provided in the generated runtime task e-mails.</p> <p><code>true</code>: Enabled. The default value is overridden by a detailed list of task attributes. The attributes and formatting cannot be configured.</p> <p><code>custom_value</code>: Customized override. The default is overridden by a customized list of task attributes that you provide. If predefined variables exist within the custom string, the variables are replaced with actual task values. The following list contains the predefined variables:</p> <ul style="list-style-type: none"> • <code>{TASK_ID}</code>: The unique ID of the task • <code>{TASK_TARGETS}</code>: The targets of the task • <code>{CELL_NAME}</code>: The cell from which the task was generated • <code>{TASK_REASON}</code>: The reason the task was generated • <code>{TASK_SUBMITTER}</code>: The name of the entity that submitted the task • <code>{TASK_SEVERITY}</code>: The severity of the task • <code>{EMPTY_STRING}</code>: An empty string used for an empty subject or body to be defined
Default	true

task.email.custom.subject:

You can use the `task.email.custom.subject` custom property to enable a custom subject line for generated runtime task e-mails.

When enabled, the e-mail subjects are in the following format:

[target name, target name, ...] - submitter - originated time - severity

Table 168. *task.email.custom.subject* custom property values

Value	Description
Scope	Cell
Valid values	<p><code>false</code>: Deactivated. A custom subject for task notifications is not provided.</p> <p><code>true</code>: Enabled. The default value is overridden by a detailed list of task attributes. The attributes and formatting cannot be configured.</p> <p><i>custom_value</i>: Customized override. The default is overridden by a customized list of task attributes that you provide. If predefined variables exist within the custom string, the variables are replaced with actual task values. The following list contains the predefined variables:</p> <ul style="list-style-type: none"> • <code>{TASK_ID}</code>: The unique ID of the task • <code>{TASK_TARGETS}</code>: The targets of the task • <code>{CELL_NAME}</code>: The cell from which the task was generated • <code>{TASK_REASON}</code>: The reason the task was generated • <code>{TASK_SUBMITTER}</code>: The name of the entity that submitted the task • <code>{TASK_SEVERITY}</code>: The severity of the task • <code>{EMPTY_STRING}</code>: An empty string used for an empty subject or body to be defined
Default	<code>false</code>

VMware custom properties

You can use custom properties to configure the product to connect to the VMware Infrastructure SDK (VI SDK) web service endpoints.

cpuLimitForCappedVMs:

The `cpuLimitForCappedVMs` custom property defines a the target processor utilization percentage for Intelligent Management processes that run on virtual machines. This custom property value is maintained for virtual machines that have a defined processor utilization limit.

You can specify the `cpuLimitForCappedVMs` custom property on the cell with a percent value between 0 and 95%. If you specify 100%, the limit is equal to the limit that is placed on the virtual machine in the VMware software.

The custom property value is the target utilization that Intelligent Management maintains for virtual machines that have processor utilization limits. This value should be the highest value that maintains virtual machine and system stability.

If your virtual machines are constantly being overloaded, lower the `cpuLimitForCappedVMs` value. To determine if a virtual machine is being overloaded, search for the following conditions in your environment:

- The virtual machine is running at 100% of its processor utilization limit for a long period of time.
- The average service time of requests is going up, indicating that requests in the application server are being queued.

vmware.service.unique_id.url:

The `vmware.service.unique_ID.url` custom property specifies the Uniform Resource Locator (URL) of the VMware web services endpoint.

For example, you might enter the following value for this property:

```
https://myserver.ibm.com/sdk
```

The `unique_id` value in the custom property name is optional. You can specify the `unique_id` value if you are specifying multiple sets of custom properties for each VMware server. For example, you might name the custom property as `vmware.service.vmwareserver1.url`. As a convention, you can specify the hostname of the vCenter or ESX server for the `unique_id` value.

vmware.service.unique_id.userid:

The `vmware.service.unique_id.userid` custom property specifies the user ID to access the vCenter or ESX server.

For the `vmware.service.unique_id.userid` custom property, the following privileges are required by Intelligent Management to read certain properties and to perform various operations:

- System.Anonymous
- System.Read
- System.View
- Sessions.TerminateSession

For example, you might enter the following value for this property:

```
root
```

The `unique_id` value in the custom property name is optional. You can specify the `unique_id` value if you are specifying multiple sets of custom properties for each vCenter or ESX server. For example, you might name the custom property as `vmware.service.vmwareserver1.userid`. As a convention, you can specify the hostname of the vCenter or ESX server for the `unique_id` value.

vmware.service.unique_id.password:

The `vmware.service.unique_id.password` custom property specifies the password to access the vCenter or ESX server.

For example, you might enter the following value for this property:

```
myPassword
```

The value can be an encoded password. To encode your password, you can use the `encodePassword.sh|bat` script. This script is in the `install_root/bin` directory.

The `unique_id` value in the custom property name is optional. You can specify the `unique_id` value if you are specifying multiple sets of custom properties for each vCenter or ESX server. For example, you might name the custom property as `vmware.service.vmwareserver1.password`. As a convention, you can specify the hostname of the vCenter or ESX server for the `unique_id` value.

Performance logs

You can use log files to help with performance monitoring, accounting and problem determination.

You can examine the following log files:

- Visualization:
 - BusinessGridStatsCache
 - DeploymentTargetStatsHistoricCache

- NodeStatsHistoricCache
- ServerStatsCache
- TCModuleInstanceStatsCache
- TCModuleStatsCache
- TierStatsCache
- FineGrainedPowerConsumptionStatsCache
- ServerPowerConsumptionStatsCache

BusinessGridStatsCache

This log file describes the business grid statistics cache.

Location

This file is in the *log_root/visualization* directory.

Usage notes

- **node**: Node name.
- **server**: Server name.
- **tcname**: Transaction class name.
- **appname**: Application name.
- **j2eemodname**: J2EE module name.
- **version**: Node version.
- **dtname**: Deployment target name.
- **scname**: Service policy name.
- **nodegroup**: Node group name
- **cell**: Cell name.
- **updateTime**: Time of update.
- **stats num_requested**: Number of jobs which arrive at the execution environment (endpoint application) for processing.
- **num_completed**: Number of jobs which run to completion at the execution environment.
- **exec_time**: Average time in milliseconds that jobs spend executing.
- **max_concurrency**: Maximum concurrency level that is attained.
- **num_queued**: Number of jobs that are queued at the scheduler.
- **num_dispatched**: Number of jobs that are dispatched at the scheduler.
- **num_failed**: Number of jobs that failed in the execution environment.
- **num_errors**: Number of dispatch errors that occurred for jobs.
- **queue_time**: Average time in milliseconds that a job spent in the queue.
- **dispatch_time**: Average time in milliseconds that a job spent being dispatched.
- **dispatch_error_time**: Average time in milliseconds for jobs spent being dispatched when a dispatch error occurred.

StrfTime format conversions

The format used when using the `%{format}t` log parameter is based on the non-extended BSD `strftime(3)` time conversion functions. The parameters that are specifically supported, and sample output, are listed in the following table.

Selected locale-specific parameters are supported as American English only. Specifically, the order in which month, day, and year appear, and perhaps the ordering of other items, remain American English in all locales. The spelling of words such as the day of the week, or the name of the month, and the timezone will be locale-correct.

Table 169. Conversion characters. Semantics of conversion characters in format string

Conversion string	Description	Example
%A	Replaced by national representation of the full weekday name.	"Thursday"
%a	Replaced by national representation of the abbreviated weekday.	"Thu"
%B	Replaced by national representation of the full month name.	"September"
%b	Replaced by national representation of the abbreviated month.	"Sep"
%C	Year, divided by 100 (i.e. the century).	"20"
%c	Replaced by national representation of time and date.	"Thu Sep 25 22:32:00 EDT"
%D	Equivalent to %m/%d/%y	"09/25/08"
%d	Replaced by the day of the month as a decimal number (01-31).	"25"
%e	Replaced by the day of the month as a decimal number (1-31).	"25"
%F	Equivalent to %Y-%m-%d.	"2008-09-25"
%G*	The ISO 8601 week-based year (see NOTES) with century as a decimal number. The 4-digit year corresponds to the ISO week number (see %V). This has the same format and value as %Y, except that if the ISO week number belongs to the previous or next year, that year is used instead.	"2008"
%g*	Replaced by the same year as in %G, but as a decimal number without century (00-99).	"08"
%H	Replaced by the hour (24-hour clock) as a decimal number (00-23).	"22"
%h	Equivalent to %b.	"Sep"
%I	Replaced by the hour (12-hour clock) as a decimal number (01-12).	"10"
%j	Replaced by the day of the year as a decimal number (001-366).	"269"
%k	Replaced by the hour (24-hour clock) as a decimal number (0-23); single digits are preceded by a blank.	"22 (or '1' for 1am)"
%l	Replaced by the hour (12-hour clock) as a decimal number (1-12); single digits are preceded by a blank.	"10"
%M	Replaced by the minute as a decimal number (00-59).	"32"
%m	Replaced by the month as a decimal number (01-12).	"09"
%n	Replaced by a newline character.	
%P	Replaced by AM or PM.	"PM"
%R	Equivalent to %H:%M.	"22:32"
%r	Equivalent to %I:%M:%S %p .	"10:32:00 pm"
%S	Replaced by the second as a decimal number (00-60).	"00"
%s	Replaced by the number of seconds since the Epoch, UTC (see mktime(3)).	"1222396320"
%T	Equivalent to %H:%M:%S	"22:32:00"
%t	Replaced by a tab character.	

Table 169. Conversion characters (continued). Semantics of conversion characters in format string

Conversion string	Description	Example
%U	Replaced by the week number of the year (Sunday as the first day of the week) as a decimal number (00-53).	"38"
%u	Replaced by the weekday (Monday as the first day of the week) as a decimal number (1-7).	"4"
%V*	Replaced by the week number of the year (Monday as the first day of the week) as a decimal number (01-53). If the week containing January 1 has four or more days in the new year, then it is week 1; otherwise it is the last week of the previous year, and the next week is week 1.	"39"
%v	Equivalent to %e-%b-%Y.	"25-Sep-2008"
%W	Replaced by the week number of the year (Monday as the first day of the week) as a decimal number (00-53).	"38"
%w	Replaced by the weekday (Sunday as the first day of the week) as a decimal number (0-6).	"4"
%X	Replaced by national representation of the time.	"22:32:00"
%x	Replaced by national representation of the date.	"09/25/08"
%Y	Replaced by the year with century as a decimal number.	"2008"
%y	Replaced by the year without century as a decimal number.	"08"
%Z	Replaced by the time zone name.	"EDT"
%z	Replaced by the time zone offset from UTC; a leading plus sign stands for east of UTC, a minus sign or west of UTC, hours and minutes follow with two digits each and no delimiter between them (common form for RFC 822 date headers).	"-0500"
%+	Replaced by national representation of the date and time (the format is similar to that produced by date(1)).	"Thu Sep 25 22:32:00 EDT"
%1	Milliseconds, rounded to three places, with leading zeros. Note: this is a departure from the strftime format, which does not represent milliseconds.	"000"
%%	Replaced by '%'	%

* %G, %g, and %V yield values [are] calculated from the week-based year defined by the ISO 8601 standard. In this system, weeks start on a Monday, and are numbered from 01, for the first week, up to 52 or 53, for the last week. Week 1 is the first week where four or more days fall within the new year. Or, synonymously, week 01 is: the first week of the year that contains a Thursday; or, the week that has 4 January in it. When three or fewer days of the first calendar week of the new year fall within that year, then the ISO 8601 week-based system counts those days as part of week 53 of the preceding year. For example, 1 January 2010 is a Friday, meaning that just three days of that calendar week fall in 2010. Thus, the ISO 8601 week-based system considers these days to be part of week 53 (%V) of the year 2009 (%G) ; week 01 of ISO 8601 year 2010 starts on Thursday, 4 January 2010.

#{The time is: %X.%1%nThe date is: %x}t

This input yields the following output: The time is: 22:32:00.000 The date is: 09/25/08

DeploymentTargetStatsHistoricCache

This log file contains historic information on the deployment target statistics historic cache.

Location

This file is located in the *log_root/visualization* directory by default. You can configure the location of the visualization data service log files.

Usage notes

- **deploymentTargetName**: Deployment target name.
- **nodeName**: Node name.
- **deploymentTargetType**: Deployment target type.
- **speedReq**: MCycles per second.
- **highMemMark**: The highest amount of memory in MBytes consumed over the life span of the deployment target.

NodeStatsHistoricCache

This log file contains historic information on the node statistics cache.

Location

This file is in the *log_root/visualization* directory by default. You can configure the location of the visualization data service log files.

Usage notes

- **nodeName**: Node name.
- **nodeCPU**: Percentage of processor utilization.
- **nodeFreeMemory**: Free memory in kilobytes.
- **usedMemory**: Memory being used in kilobytes.
- **version**: Node version.
- **nodeSpeed**: Node speed in normalized MCycles/sec.
- **background**: Background processor use in normalized MCycles/sec.
- **wasBackground**: WebSphere Application Server background processor use in normalized MCycles/sec.
- **isDBNode**: True, if this is a database node.
- **entitledCapacity**: The percentage of entitled capacity that is being used by a server partition. This metric is only available for AIX shared-uncapped micro-partitions. If you are not using this configuration, the value displays as 0.
- **maxEntitledCapacity**: The maximum of entitled capacity that can be used by a server partition. This metric is only available for AIX shared-uncapped micro-partitions. If you are not using this configuration, the value displays as 0.
- **physicalCPUsConsumed**: The number of physical processors that are being used by the shared server partition. This metric is only available for AIX shared-uncapped micro-partitions. If you are not using this configuration, the value displays as 0.
- **availablePoolCPUs**: The number of processors available in the shared processor pool to which this server partition belongs. This metric is only available for AIX shared-uncapped micro-partitions. If you are not using this configuration, the value displays as 0.

ServerStatsCache

This log file describes the server statistics cache.

Location

This file is located in the *log_root/visualization* directory by default. You can configure the location of the visualization data service log files.

Usage notes

- **name:** Specifies the server name.
- **node:** Specifies the node name.
- **version:** Specifies the version of Intelligent Management that is running on the node.
- **weight:** Specifies the dynamic workload management (dWLM) weight of the server.
- **cpu:** Specifies the percentage of the processor that is being used.
- **usedMemory:** Specifies the number of kilobytes of memory that is used in the Java virtual machine (JVM) run time. This statistic is available only for server instances containing WebSphere Application Server.
- **uptime:** Specifies the number of milliseconds that JVM has been running. This statistic is available only for server instances containing WebSphere Application Server.
- **totalRequests:** Specifies the number of total requests for the server. This statistic is available only for server instances containing WebSphere Application Server. If your environment contains other servers, including external WebSphere application servers, you can manually calculate a value that is similar to the value of the totalRequests metric. Locate all entries for a particular server in the `TModuleInstanceStatsCache.log` file. Using the beginning and end time stamps, calculate the sum of the serviced metric for the server. For more information, read about `TModuleInstanceStatsCache`.
- **updateTime:** Specifies the time that the statistics were provided.
- **db_averageResponseTime:** Specifies the average response time for the database server.
- **db_throughput:** Specifies the throughput for the database server.
- **highMemMark:** The highest amount of memory in megabytes consumed over the life span of the server.
- **residentMemory:** The amount of physical memory not shared with other processes that the JVM is using in kilobytes.
- **totalMemory:** Specifies the total amount of memory that the JVM is using in kilobytes.
- **totalMethodCalls:** Specifies the total number of enterprise Java bean (EJB) module method calls.

TModuleStatsCache

This log file contains information about the transaction class module cache.

Location

This file is located in the `log_root/visualization` directory by default. You can configure the location of the visualization data service log files.

Log information

- **timeStamp:** Time in milliseconds.
- **tcmname:** Specifies the transaction class module name.
- **dtname:** Specifies the deployment target name.
- **gwid:** Specifies the Gateway ID.
- **j2eemodname:** Specifies the Java 2 Platform, Enterprise Edition (J2EE) module name.
- **appname:** Specifies the application name.
- **tcname:** Specifies the transaction class name.
- **scname:** Specifies the service policy name.
- **nodegroup:** Specifies the node group name.
- **cell:** Specifies the cell name.
- **proxy:** Specifies the proxy or on demand router name.
- **proxycell:** Specifies the cell where the autonomic request flow manager (ARFM) gateway that submitted these statistics resides.

- **proxynode**: Specifies the node on which the ARFM gateway that submitted these statistics resides.
- **proxyserver**: Specifies the server on which the ARFM gateway that submitted these statistics resides.
- **protocol**: Specifies the protocol family of the requests for these statistics: Hypertext Transfer Protocol (HTTP), SIP, Internet Inter-ORB Protocol (IIOP), or Java Message Service (JMS).
- **arrivals**: Specifies the number of requests that arrived during the reported interval.
- **executingInt**: Specifies the execution concurrency integral. This field is the sum, over each millisecond in the reported interval, of the number of requests that were running at the start of the millisecond.
- **lengthInt**: Specifies the queue length integral. This field is the sum, over each millisecond in the reported interval, of the number of requests in the queue at the start of the millisecond.
- **currentLen**: Specifies the number of requests in the queue at the end of the reported interval.
- **departs**: Specifies the number of messages that were dispatched to server during the reported interval and initiated a logical dialog. This field counts only the first reception of a certain message.
- **dropped**: Specifies the number of requests that were dropped because of a queue overflow.
- **waittm**: Specifies the sum in milliseconds, over all of the requests that are dispatched during the reported interval, of the time that each request spends waiting in the queue. The sum is specified in number of milliseconds. Divide the waittm value by the departs value to get the average amount of time that a request spends waiting in the queue.
- **resptm**: Specifies the sum in milliseconds of all of the response times of the requests that complete running during the reported interval. Divide the resptm value by the serviced value to get the average response time for the requests. The response time of a request is the sum of the queuing time of the request plus the service time of the request. The response time, however, does not include any time before the request reached the ARFM gateway queue. To calculate the sum, add the waittm value, which excludes time before enqueueing, and the servicetm value.
- **servicetm**: Specifies the sum of all the service times of the requests that completed running during the reported interval. The sum is in units of milliseconds. Divide the servicetm value by the serviced value to get the average service time for a request during the reported interval.
- **serviced**: Specifies the number of requests that completed during the reported interval.
- **begintm**: Specifies the start time for the statistics interval.
- **endtm**: Specifies the end time for the statistics interval.
- **qlen**: Specifies the sum, over the requests that arrived during the reported interval, of the queue length, before insertion, as seen at each request arrival.
- **abvgoal**: Specifies the number of requests that both returned during the reported interval had a response time exceeding their service class threshold.
- **workFactors**: Specifies the average amount of work that is performed on a deployment target to serve an HTTP request, a Session Initiation Protocol (SIP) message, an Internet Inter-ORB Protocol (IIOP) call, or a Java Message Service (JMS) message. The work factor is represented in standard processor megacycles. You can override the work factor values by specifying the workFactorOverrideSpec custom property on the deployment target. For more information about the workFactorOverrideSpec custom property, read about autonomic request flow manager advanced custom properties.
- **dlgRtmRefusals**: Specifies the number of dialog retransmissions that are refused admission to a queue.
- **dlgRtmAdmissions**: Specifies the number of dialog retransmissions that are admitted to a queue.
- **dlg1stRefusals**: Specifies the number of dialog initiating messages that are not retransmissions, which are refused admission to a queue.
- **dlg1stAdmissions**: Specifies the number of dialog initiating messages that are not retransmissions, which are admitted to a queue.

TCModuleInstanceStatsCache

This log file describes the transaction class module instance cache.

Location

This file is located in the *log_root/visualization* directory by default. You can configure the location of the visualization data service log files.

Usage notes

- **tcmidname**: Transaction class module instance name.
- **gwid**: Gateway ID.
- **j2eemodname**: Java EE module name.
- **dtname**: Deployment target name.
- **scname**: Service policy name.
- **appname**: Application name.
- **tcname**: Transaction class name.
- **server**: Server name.
- **node**: Node name.
- **nodegroup**: Node group name cell: cell name.
- **proxy**: Name of the proxy or on demand router (ODR) hosting the ARFM gateway that submitted the statistics. Applicable if the protocol is HTTP or SIP.
- **proxycell**: The cell in which the proxy or ODR resides.
- **proxynode**: The node in which the proxy or ODR resides.
- **proxyserver**: Name of the server hosting the ARFM gateway that submits the statistics. Applicable if the protocol is JMS or IIOP.
- **protocol**: The protocol family of the requests that these statistics are for; that is, HTTP, IIOP, JMS, SIP.
- **arrivals**: No meaningful value.
- **respOutTime**: The sum in milliseconds, over the requests that completed during the reported interval due to service timeout, of the request's response time. Divide by serviced to compute average response time. Calculate the sum of the service time and queue time. The metric, however, does not include any time spent before the request reaches the ARFM gateway queues.
- **serviceOutTime**: The sum in milliseconds, over the requests that completed during the reported interval due to service timeout, of the request's service time. Divide by serviced to compute average service time.
- **numTimedOut**: The number of requests that completed, during the reported interval, due to service timeout.
- **numErrored**: The number of requests that completed, during the reported interval, with an error indicator.
- **firstErrorTime**: The time of the first error response, if any, otherwise 0.
- **lastErrorTime**: The time of the last error response, if any, otherwise 0.
- **lastTimeoutTime**: The time of the last timeout response, if any, otherwise 0.
- **currentLen**: No meaningful value.
- **lengthInt**: No meaningful value.
- **executingInt**: Integral over each millisecond in the reported interval, of the number of requests that were executing at the start of the interval.
- **execution**: Concurrency integral, in units of milliseconds * requests.
- **departs**: Number of requests dispatched to server.
- **dropped**: No meaningful value.
- **waittm**: Total wait time in queue of all requests in interval.
- **resptm**: Total response time for all requests in interval.
- **servicetm**: Total service time for all requests in interval serviced: number of requests serviced.

- **begin_{tm}**: Start time of interval.
- **end_{tm}**: End time of interval.
- **qlen**: Total queue length over in the interval.
- **abvgoal**: Number of requests that both returned during the reported interval had a response time above their service class threshold.
- **workFactors**: The amount of work in MCycles/sec that each request is estimated to consume after the request is dispatched from the ARFM gateway.
- **dlgRtmRefusals**: Specifies the number of dialog retransmissions that are refused admission to a queue.
- **dlgRtmAdmissions**: Specifies the number of dialog retransmissions that are admitted to a queue.
- **dlg1stRefusals**: Specifies the number of dialog-initiating messages that are not retransmissions, which are refused admission to a queue.
- **dlg1stAdmissions**: Specifies the number of dialog-initiating messages that are not retransmissions, which are admitted to a queue.

TierStatsCache

This log file describes the tier statistics cache.

Location

This file is in the *log_root/visualization* directory.

Usage notes

- **proxy**: Proxy name.
- **dtName**: Server name.
- **svcclsName**: Service class name.
- **power**: Power[®] coefficient.

FineGrainedPowerConsumptionStatsCache

This log file contains fine grained power and work consumption data. A record is written for every transaction class module and server instance. This action creates a record for every middleware application, module, transaction class, and server instance that has had work routed through an on demand router (ODR). Additional fields exist that hold relationship information such as the cluster to which the server belongs, the node group with which the cluster is associated, and the service policy with which the transaction class is associated.

Location

This file is in the *log_root/visualization* directory.

Usage notes

- **timeStamp**: Specifies the time in milliseconds since January 1, 1970, 00:00:00 GMT.
- **tcmodname**: Specifies the transaction class module name. This value is a concatenation of the transaction class, the application name, and the module name.
- **gwid**: Specifies the gateway ID, which is a concatenation of the cluster name, cell name, node name, and ODR name.
- **cell**: Specifies the cell name.
- **appname**: Specifies the application name
- **modulename**: Specifies the module name.
- **servicepolicy**: Specifies the service policy name.
- **tcname**: Specifies the transaction class name.

- **server**: Specifies the server name.
- **node**: Specifies the node name.
- **odr**: Specifies the ODR name.
- **cluster**: Specifies the cluster name.
- **nodegroup**: Specifies the node group name.
- **begintm**: Specifies the time that the interval began.
- **endtm**: Specifies the end time of the interval.
- **workfactor**: Specifies the estimated work factor from the work profiler for the request type.
- **numserviced**: Specifies the number of requests serviced of this type.
- **workcompleted**: Specifies the amount of work completed in the interval, calculated as $\text{numserviced} * \text{workfactor}$.
- **powerconsumed**: Specifies the power consumption, calculated as $\text{numserviced} * \text{workfactor} / (\text{endtm} - \text{begintm})$.
- **nodepower**: Specifies the total power that is available for the node.
- **nodeworkpotential**: Specifies the total amount of work that the node could accommodate over the interval ($\text{totalnodepower} * (\text{endtime} - \text{begintime})$).
- **cellpower**: Specifies the total amount of power available for consumption for the cell. This value is the sum of the **nodepower** value over all nodes in the cell.
- **cellworkpotential**: Specifies the total amount of work that the cell could accommodate over the interval ($\text{totalcellpower} * (\text{endtime} - \text{begintime})$).

ServerPowerConsumptionStatsCache

This file is a consolidation of the FineGrainedPowerConsumptionStatsCache at the server level with some additional server data.

Location

This file is in the *log_root/visualization* directory.

Usage notes

- **timeStamp**: Time in milliseconds since January 1, 1970, 00:00:00 GMT.
- **cell**: Cell name.
- **name**: Server name.
- **node**: Node name.
- **cluster**: Cluster name.
- **nodegroup**: Node group name.
- **begintimestamp**: The begin time of the interval.
- **endtimestamp**: The end time of the interval.
- **cpu**: The average server percentage of processor utilization over the *cpuetimeinterval* value.
- **workcompleted**: The amount of work completed by the server in the interval $(\text{cpu} * \text{nodepower}) * (\text{endtime} - \text{begintime in seconds}) / 100$.
- **powerconsumed**: The power consumption by the server $(\text{cpu} * \text{nodepower}) / 100$.
- **nodepower**: The total power available for the node.
- **nodeworkpotential**: The total amount of work that the node could accommodate over the interval ($\text{totalnodepower} * (\text{endtime} - \text{begintime})$).
- **cellpower**: The total amount of power available for consumption for the cell; a sum of the node power over all nodes in the cell.
- **cellworkpotential**: The total amount of work that the cell could accommodate over the interval ($\text{totalcellpower} * (\text{endtime} - \text{begintime})$).

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

APACHE INFORMATION. This information may include all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org>. You may obtain a copy of the Apache License at <http://www.apache.org/licenses/LICENSE-2.0>.

IBM may have patents or pending patent applications covering subject matter 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 Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. For a current list of IBM trademarks, visit the IBM Copyright and trademark information Web site (www.ibm.com/legal/copytrade.shtml).

Microsoft and Windows 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.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Index

A

- activate 162
- activation 155
- active 162
- administrative console 454
- administrative roles 291
- AdminTasks
 - HTTP 413
 - middleware servers 389, 393
 - ODR 413
 - PHP servers 409
 - SIP 413, 422
 - work class 357
- AIX
 - configuration 8
- application edition 168
- application edition manager 155, 162, 177, 178, 179, 440
 - application edition manager 160
 - application validation 170
- application edition tutorial 166
- application editions 155, 440
- application infrastructure virtualization
 - benefits 2, 8, 9, 11
- application lazy start
 - dynamic cluster 84, 85, 87
 - lazy start controller 87
- application performance 155
- application placement 87, 88, 89, 192
 - dynamic 194
- Application placement
 - monitoring and tuning 203
 - troubleshooting 205, 209
- application placement controller 19, 79, 201, 448
- atomic 155, 174
- autonomic controllers 448
- autonomic managers 180
- autonomic request flow manager 245, 454
- Autonomic request flow manager
 - APC 184
 - ARFM 184
 - endpoint selection 184
 - tuning 184
 - UseAPCEndpointSelection 184

B

- backward compatible 171
- base 155
- BBSON 58, 70
- bottleneck 215
- bottlenecks 214
- bulletin board
 - high availability manager 58

C

- cell 162
 - configuring 58
 - cross cell communication 58
- cell affinity
 - generic server cluster 42, 43, 44
 - ODR 40
 - odrSessionAffinityEnabled 42, 43
 - plugin-cfg.xml 44
- center cell 196, 198, 307, 308, 322
- checkpoint 180
- clone 168, 171
- clone ID 70
- compatibility 155
- compatible 163
- components of 17
- concurrent 155, 163, 166
- configuring 57
 - cross cell communication 58
- continuity 162
- custom logs 472
- custom properties 291, 440, 472
- custom property 454

D

- database tier 214
 - monitoring 215
- deactivate 162
- default work classes
 - action types 148
 - configuration placement location 150
 - new application work classes 150
 - request 149
 - routing policy 148, 149
 - service policy 149
 - work class 149, 150
- deployment manager 19, 199, 201, 308, 322
- dynamic cluster 75, 79, 199, 440
 - creating 82, 119
 - dynamic cluster server template 80
 - membership policy 80
 - operating mode 80
 - server template 80
- dynamic clusters 19, 70, 162, 201
- dynamic operations 213
 - components of 17
 - health management
 - overview 217
 - overview 12
 - request flow prioritization 210
 - overview 14
- Dynamic operations 189, 192, 216
 - advanced user tasks
 - configuring ODR to route externally 242
 - external monitoring 244
 - overview 14, 91

E

- edition 155, 162, 164
- edition manager
 - application rollout algorithm 174
- edition rollout 155
- elasticity mode 19, 79, 199, 201, 448
- email
 - configuring 271
 - notifications 271
- emergency throttle 188
- events
 - event log file 270
 - planned, unplanned 268
- existent 162
- extended manageability 21
 - features 262
 - overview 263
 - troubleshooting 272

G

- generic server clusters 70
- group 174

H

- HA manager 251
- hardware virtualization 8
 - benefits 2, 8, 11
- health controller 216
 - AdminConfig 304
 - commands 304
- health management 180, 216
 - overview 217
- Health management
 - creating health policy 222
 - disabling 216
 - enabling 216
 - troubleshooting 239
 - Tuning 236
- health policies 216
- health policy
 - excessive request timeout 221
- heap usage 472
- high availability 448
- high availability deployment manager 57, 180
 - HA manager 254
 - scale-out administration 254
- high availability manager 62
- HTTP 472
- HTTP session affinity 70

I

- importOverlayConfig.py 196, 198
- importOverlayConfig.py script 307
- inactive 162
- interruption-free 155, 160

L

- linkCells script 308, 322
- linkCells.bat 308
- linkCells.sh 196, 308
- linkCells.shl.bat 198
- log 267
- logs 291
 - NodeStatsHistoricCache 482

M

- maintenance mode 180
- manageODR.py 79
- managing
 - extended manageability 262
- MBean documentation 291
- memory overload protection 189
- middleware applications 371
- middleware server
 - BEA WebLogic 108
- middleware servers
 - Apache Tomcat 107
 - JBoss 110
- modes of operations
 - dynamic operations 16
- MOP 189
 - memory overload protection 190
- multi-cell 196, 198
- multi-cell performance 307
- multi-cell performance management 19, 199, 201, 448
- multi-cell placement 307, 472
- multicell 196, 198
- multicell performance 307
- multicell performance management 19, 199, 201
- multicell placement 307
- multiple tiers 245

N

- node 19, 199, 201
- non-existent 162
- notifications
 - email 271
 - events 268

O

- ODR 19, 61, 70, 79, 160, 162, 163, 166, 189, 201, 245, 308, 322, 422, 472
 - cell affinity 41, 44
 - rule-based request classification 153
 - Setting up a Web server front 61
- ODR (On Demand Router)
 - role in application placement 192
- on demand configuration 70
- on demand operating environment 2, 8, 9, 11
- on demand router 19, 79, 188, 201, 308, 322, 472
 - creating a cluster 78
- On demand router 245
- on demand router (ODR) 57

- operating mode
 - dynamic operations 16
- operational environment 162
- overlay communication 307, 308, 322
- overlaynodes.config file 307
- overload 472
- overview 192
 - custom log 51

P

- performance
 - monitoring
 - database tier 214
- performing a rollout 177
- plugin-cfg 448
- plugin-cfg.xml 61
- plugin-cfg.xml file 62, 70
- point cell 196, 198, 307, 308, 322
- policy
 - batch 50
- POWER 9
- proxy server
 - configuring SIP 54

R

- release notes 291
- request flow prioritization
 - overview 14
- Request prioritization
 - configuring ODR 37, 38, 56, 57, 314
 - troubleshooting 191
- role of ODR 14
- roll back 155
- roll out 155
- rollout 162, 176
- rollout update 160
- routing 70
- routing policy 155, 166, 171
- routing rules 440
- rule 166
- rules
 - ODR request classification 153
 - routing policy 146, 147
 - service policy 146
 - tuning 146
- running 162
- runtime 162
- runtime tasks 262

S

- scale-out administration
 - high availability deployment manager 254
- script 323
 - wsadmin
 - ODR 303
- scripts 291, 294

- security 308, 322
- server template 82
- server virtualization
 - benefits 9
- service 267
- service policy 18
 - defining 123
 - goal value 123
- session ID 70
- SIP 472
- speed factor 245, 454
- star topology 198, 307, 308, 322
- static cluster 78, 82
- system administration 454
- system properties 472

T

- task management services
 - events 270
- test 168
- troubleshooting 177, 308
 - visualization 272

U

- uninstall 162
- unlinkCells script 308, 322
- unlinkCells.bat 322
- unlinkCells.sh 322
- unlinkCells.shl.bat 198

V

- validate 168
- validate an edition 168
- validation 155, 162
- version 155
- vertical stacking 87, 88, 89
- visualization
 - data 267
 - extended manageability 263
- visualization data service 262

W

- Web server 70
- web server plug-in configuration 62
- WebSphere Application Server Community Edition
 - JMX security 101
- WebSphere Extended Deployment
 - Application Edition Manager
 - edition, deleting the active 176
- work class 171
- work factor 245, 454
- wsadmin 294
- wve_encodePassword 323

Z

z/OS 178