IBM WebSphere Test 000-377 Study Guide
WebSphere Application Server Network Deployment V7.0
Core Administration

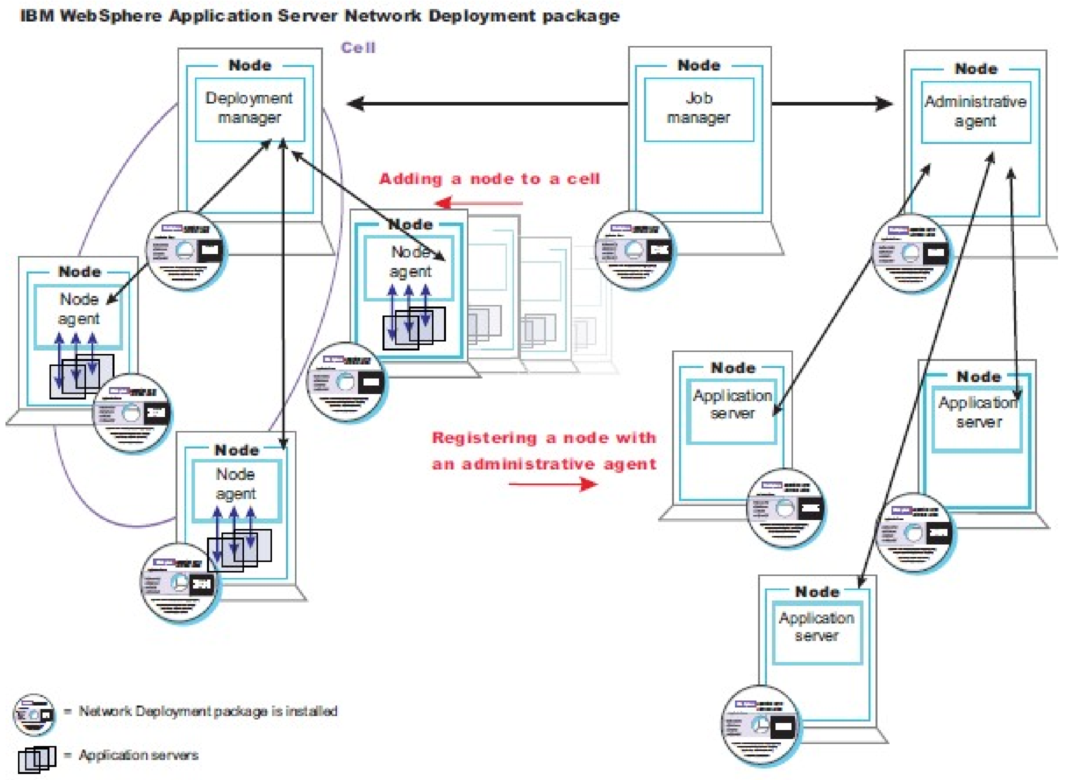
Kevin J. Ackerman

Wednesday February 13, 2010

Table of Contents

Section 1 - Architecture (13%)

A) Illustrate the relationships between IBM WebSphere Application Server, Network Deployment, V7.0 and the application components (e.g., browser, HTTP server, proxy server, plug-in, firewall, database servers, WebSphere MQ, load balancing, ip spraying, and Tivoli Performance Viewer.)



Java EE 5, the latest release of the Java EE platform, represents a significant evolution in the Java enterprise programming model. It improves the application developer experience and productivity with following new features:

Latest specifications

- Enterprise JavaBeans™ (EJB™) 3.0
- JAX-WS 2.0
- JSP™ 2.1
- Servlet 2.5
- JSF 1.2
- Java Development Kit (JDK™) 6.0

Feature Pack for EJB 3.0: The Feature Pack for EJB 3.0 available for WebSphere Application Server V6.1 is embedded into WebSphere Application Server V7.0.

If you have used EJB 3.0 Feature Pack functionality in your application in a WebSphere Application Server V6.1 environment before, there is no need to install additional WebSphere Application Server packages to use these applications with V7.0.

Job manager - Transfer management jobs like deploying, starting and stopping applications, and distributing files. You can include stand-alone and Network Deployment servers in such topologies.

Fine-grained administration security can now be enforced through the administration console. You can restrict access based on the administrators' role at the cell, node, cluster, or application level, offering fine-grained control over administrator scope. This capability is valuable in large-cell implementations where multiple administrators are responsible for subsets of the application portfolio running on the cell.

Ability to create multiple security domains within a single WebSphere Application Server cell. Each security domain can have its own user population (and underlying repository). Additionally, the application domain can be separated from the administrative domain.

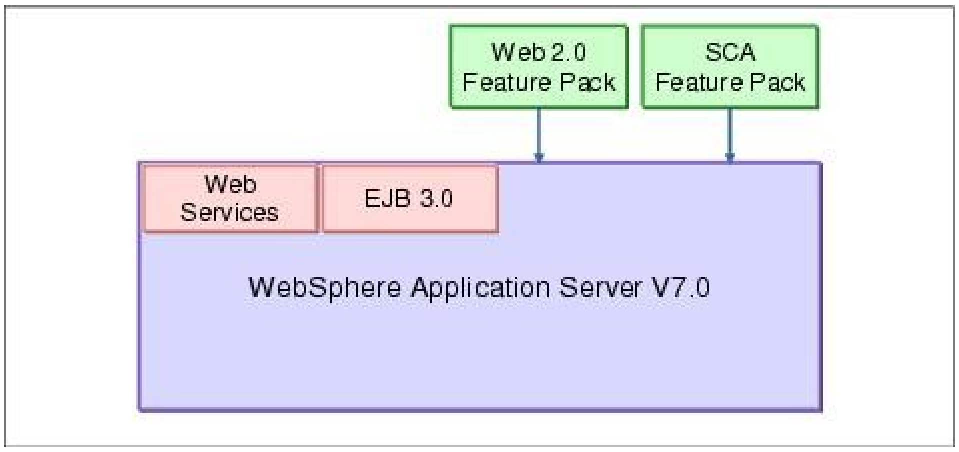
Web services

WebSphere Application Server V7.0 includes support for the following Web services and Web services security standards:

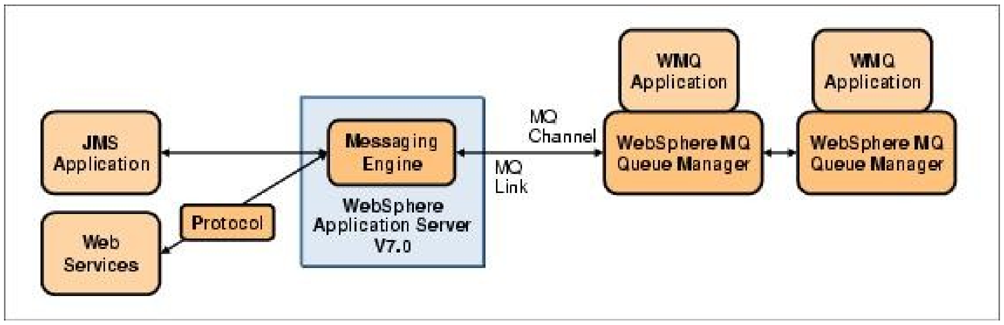
- Web Services Interoperability Organization (WS-I) Basic Profile 1.2 and 2.0
- WS-I Reliable Secure Profile
- Java API for XML Web Services (JAX-WS)
- SOAP 1.2
- SOAP Message Transmission Optimization Mechanism (MTOM)
- XML-binary Optimized Packaging (XOP)
- WS-ReliableMessaging
- WS-Trust
- WS-SecureConversation
- WS-Policy

Installation Factory - WebSphere Application Server includes the Installation Factory tool for creating customized install packages (CIPs). CIPs are used for packaging installations, updates, fixes, and applications to create custom and all-in-one installation solutions. Installation Factory is convenient for ISVs that want to create customized and bundled application solutions with WebSphere Application Server, as well as for organizations who prefer to have ready-to-deploy installation packages containing all the pre-installed artifacts. Consider using Installation Factory to create one or more CIPs and use those CIPs to deploy or update WebSphere throughout your organization

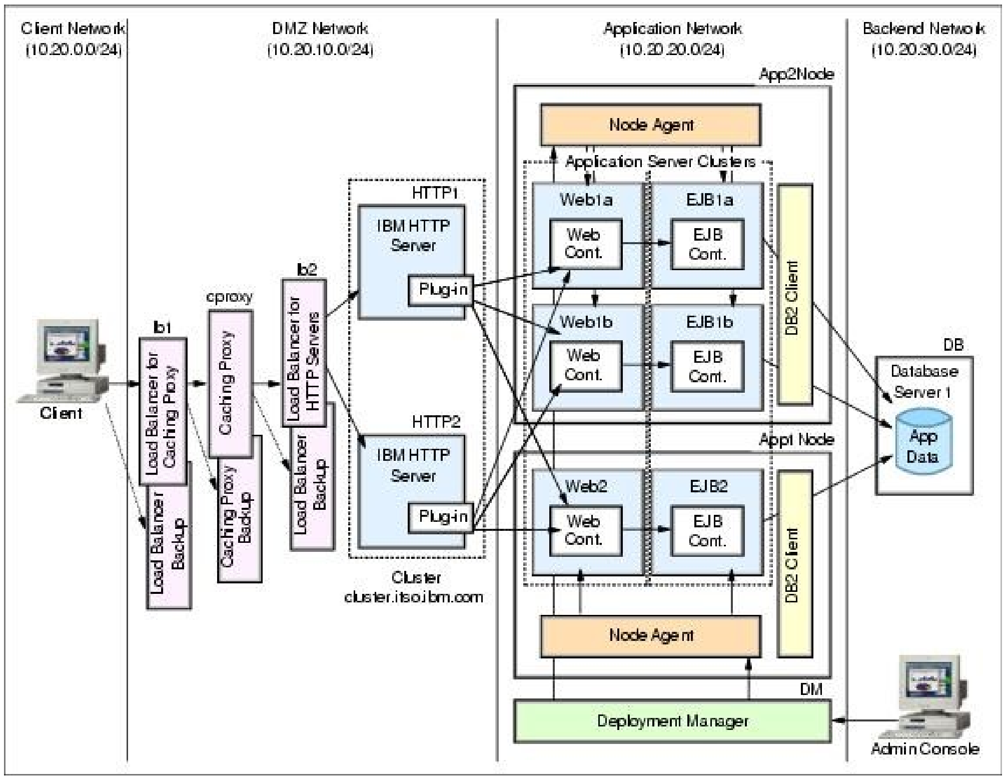
WebSphere Application Server Feature Packs



A sample integration for WebSphere Application Server and WebSphere MQ.



A complex sample topology diagram



The topology includes the following elements:

Two IBM HTTP Server Web servers configured in a cluster

Incoming requests for static content are served by the Web server. Requests for dynamic content is forwarded to the appropriate application server by the Web server plug-in.

A Caching Proxy that keeps a local cache of recently accessed pages

Caching Proxy is included in WebSphere Application Server Edge Components. Cacheable content includes static Web pages and JSPs with dynamically generated but infrequently changed fragments. The Caching Proxy can satisfy subsequent requests for the same content by delivering it directly from the local cache, which is much quicker than retrieving it again from the content host.

A backup server is configured for high availability.

A Load Balancer to direct incoming requests to the Caching Proxy and a second Load Balancer to manage the workload across the HTTP servers

Load Balancer is included in WebSphere Application Server Edge Components. The Load Balancers distribute incoming client requests across servers, balancing workload and providing high availability by routing around unavailable servers.

A backup server is configured for each primary Load Balancer to provide high availability.

A dedicated server to host the deployment manager

The deployment manager is required for administration but is not critical to the runtime execution of applications. It has a master copy of the configuration that should be backed up on a regular basis.

Two clusters consisting of three application servers

Each cluster spans two machines. In this topology, one cluster contains application servers that provide the Web container functionality of the applications (servlets, JSPs), and the second cluster contains the EJB container functionality. Whether you choose to do this or not is a matter of careful consideration. Although it provides failover and workload management capabilities for both Web and EJB containers, it can also affect performance.

A dedicated database server running IBM DB2 V9

Data flow from web browser to web container is via HTTPS. From inside the web container, servlet/JSP's access the EJB container is done via RMI/IIOP.

Data flow from Java client to the EJB container is done via RMI/IIOP

JMS Receiver request flow: From message sender to message topic/queue via JMS, MQ. From the topic/queue to the MDB.

JMS Sender Request flow: From the topic/queue to the message receiver via JMS, MQ.

Web Service Provider SOAP/HTTP(s) flow: From Web Service Client to Web Container done via SOAP/HTTP(s) to the web services engine to the EJB session bean.

Web Service Provider SOAP/JMS flow: From the Web service client to the message topic/queue via SOAP/JMS, then to the MDB and onto the EJB session bean or the web services engine.

Web Service Client SOAP/HTTP(s) request flow: Application artifacts (servlets, EJB's nad JavaBeans) within the application server act as Web Service clients. Flow goes from the artifacts to the web services engine and outbound to the Web Service Provider or Gateway via SOAP/HTTP(s)

Web Service Client SOAP/JMS request flow: Flow goes from the artifacts on the application server to the web services engine on to the message topic/queue and onto the web service provider via SOAP/JMS.

The web container executes servlets and JSP's, both of which are java classes that generate markup to be viewed by a web browser. Traffic into and out of the web container travels through the embedded HTTP server. While servlets and JSP's can act independently, they most commonly make calls to EJB's to execute business logic or access data. EJB's, which run in the EJB container, are easily reusable Java classes. They most commonly communicate with a relational database or other external source of application data, either returning that data to the web container or making changes to the data on behalf of the servlets/JSP.

In order to access a database from an application server, a JDBC provider is necessary.

The JDBC provider object supplies the specific JDBC driver implementation class for access to a specific vendor database. To create a pool of connections to that database, you associate a data source with the JDBC provider. Together, the JDBC provider and the data source objects are functionally equivalent to the Java™ Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection factory, which provides connectivity with a non-relational database.

Rather than having the JDBC Drivers directly communicate with the database, the communication is abstracted into a data source. Data sources can improve performance and portability for database access. Can use standard and XA datasource.

Connection pooling is provided by two parts, a J2C connection manager or a Relational Resource adapter.

B) Apply design considerations of IBM WebSphere Application Server, V7.0 when installing an enterprise environment (e.g., LDAP, database servers, WebSphere messaging, etc.)

The following topics are important to consider when designing a WebSphere Application Server deployment. They will most likely impact your design significantly:

<u>Scalability</u> - Scalability, as used in this book, means the ability of the environment to grow as the infrastructure load grows. Scalability depends on many different factors such as hardware, operating system, middleware, and so on.

<u>Caching</u> - WebSphere Application Server provides many different caching features at different locations in the architecture. WebSphere Application Network Deployment provides caching features at each possible layer of the infrastructure:

- Infrastructure edge:
  - Caching Proxy provided by the Edge Components
  - WebSphere Proxy Server
- HTTP server layer:
  - Edge Side Include (ESI) fragment caching capabilities provided by the WebSphere plug-in
  - Caching capabilities of the HTTP server itself (like the Fast Response Cache Accelerator (FRCA) as provided by most implementations of IBM HTTP Server)
- Application server layer:
  - Dynamic caching service inside the application server's JVM
  - WebSphere Proxy Server

<u>High availability</u> – Designing an infrastructure for high availability means to design an infrastructure in a way that your environment can survive the failure of one or multiple components. High availability implies redundancy by avoiding any single point of failure on any layer (network, hardware, processes, and so forth). The number of failing components your environment has to survive without losing service depends on the requirements for the specific environment.

<u>Load-balancing and fail-over</u> - As a result of the design considerations for high availability you will most likely identify a number of components that need redundancy. Having redundant systems in an architecture requires you to think about how to implement this redundancy to ensure getting the most benefit from the systems during normal operations, and how you will manage a seamless fail-over in case a component fails. In a typical WebSphere Application Server environment, there are a variety of components that need to be considered when implementing load-balancing and fail-over capabilities:

- Caching proxy servers
- HTTP servers
- Containers (such as the Web, SIP, and Portlet containers)
- Enterprise JavaBeans (EJB) containers
- Messaging engines
- Back-end serves (database, enterprise information systems, and so forth)
- User registries

<u>Disaster recovery</u> - Disaster recovery concentrates on the actions, processes, and preparations

to recover from a disaster that has struck your infrastructure. Important points when planning for disaster recovery are as follows:

Recovery Time Objective (RTO) - How much time can pass before the failed component must be up and running?

Recovery Point Objective (RPO) - How much data loss is affordable? The RPO sets the interval for which no data backup can be provided. If no data loss can be afforded, the RPO would be zero.

http://www.ibm.com/developerworks/websphere/techjournal/0707_col_alcott/0707_col_alcott.html

Security - Consider how security will affect your infrastructure:

- Understand the security policy and requirements for your future environment.
- Work with a security subject matter expert to develop a security infrastructure that adheres to the requirements and integrates in the existing infrastructure.
- Make sure that sufficient physical security is in place.
- Make sure the application developers understand the security requirements and code the application accordingly.
- Consider the user registry (or registries) you plan to use. WebSphere Application Server V7.0 supports multiple user registries and multiple security domains.
- Make sure that the user registries are not breaking the high availability requirements. Even if the user registries you are using are out of scope of the WebSphere Application Server project, considerations for high availability need to be taken and requested. For example, make sure that your LDAP user registries are made highly available and are not a single point of failure.
- Define the trust domain for your environment. All computers in the same WebSphere security domain trust each other. This trust domain can be extended, and when using SPNEGO / Kerberos, even out to the Windows desktop of the users in your enterprise.
- Assess your current implementation design and ensure that every possible access to your systems is secured.
- Consider the level of auditing required and how to implement it.
- Consider how you will secure stored data. Think of operating system security and encryption of stored data.
- Define a password policy, including considerations for handling password expirations for individual users.
- Consider encryption requirements for network traffic. Encryption introduces overhead and increased resource costs, so use encryption only where appropriate.
- Define the encryption (SSL) endpoints in your communications.
- Plan for certificates and their expiration:
- Decide which traffic requires certificates from a trusted certificate authority and for which traffic a self-signed certificate is sufficient. Secured connections to the outside world usually use a trusted certificate, but for connections inside the enterprise, self-signed certificates are usually enough.
- Develop a management strategy for the certificates. Many trusted certificate authorities provide online tools which support certificate management. But what about self-signed certificates?
- How are you going to back up your certificates? Always keep in mind that your

certificates are the key to your data. Your encrypted data is useless if you lose your certificates.

- Plan how you will secure your certificates. Certificates are the key to your data, therefore make sure that they are secured and backed up properly.
- http://www.ibm.com/developerworks/websphere/techjournal//0512_botzum/0512_botzum1.html

Application deployment - When planning for WebSphere Application Server infrastructure, do not delay application deployment thoughts until the application is ready to deploy. Start this planning task when you start designing your infrastructure. The way you deploy your application affects your infrastructure design in various ways:

- Performance :

  The way you deploy your application can affect performance significantly. Tests showed significant performance differences when deploying Enterprise Java Bean (EJB) modules to the same application server as the EJB client components, compared to deploying EJB modules to a separate application server than the EJB client components. Deploying EJB modules to separate application servers, however, gives a lot more flexibility and avoids duplication of code.

- Number of application servers :

  When you plan the deployment of multiple applications in your environment you can either deploy multiple applications to the same application server or create one server for each application. Depending on that decision you might end up with a higher number of application servers in your cell. This means that you will have a larger cell, which implies increased server startup times and complexity for your environment due to larger high availability views.

- Cluster communication :

  The higher the number of application servers, the higher is the communications and systems management overhead in your environment.

- Platforms :

  WebSphere Application Server supports building cross-platform cells. This means you can create one cell containing servers on multiple platforms. While this option increases the complexity of the environment and makes the administration more complicated, it gives a lot of flexibility when it comes to integration of existing software components.

Servicability - Plan for servicability and problem determination tasks for your environment. Planning for servicability includes planning for tools, education, disk requirements for debug data, communications requirements, required agents for your tools, and so on. WebSphere Application Server V7.0 comes with IBM Support Assistant V4.0 which provides many tools for problem analysis and data collection.

C) Relate the various components of the IBM WebSphere Application Server Network Deployment V7.0 runtime architecture.

WebSphere Application Server Network Deployment (ND) provides the capabilities to develop more enhanced server infrastructures. It extends the WebSphere Application Server base package and includes the following features:

- Clustering capabilities
- Edge components

- Dynamic scalability
- High availability
- Advanced management features for distributed configurations

These features become more important at larger enterprises, where applications tend to service a larger client base, and more elaborate performance and high availability requirements are in place.

WebSphere Application Server Network Deployment Edge Components provide high performance and high availability features. For example, the Load Balancer (a software load balancer) provides horizontal scalability by dispatching HTTP requests among several Web server or application server nodes supporting various dispatching options and algorithms to assure high availability in high volume environments. The usage of Edge Component Load Balancer can reduce Web server congestion, increase content availability, and provide scaling ability for the Web server.

WebSphere Application Server Network Deployment also includes a dynamic cache service, which improves performance by caching the output of servlets, commands, Web services, and JSP files. This cache can be replicated to the other servers. The state of dynamic cache can be monitored with the cache monitor application. For more information about WebSphere Application Server Network Deployment V7.0, see the following Web page:
http://www.ibm.com/software/webservers/appserv/was/network/

WebSphere Extended Deployment

WebSphere Extended Deployment (XD) is a suite of application infrastructure products that can be installed and used separately or as a package:

WebSphere Virtual Enterprise

This product, formerly Operations Optimization, provides application infrastructure virtualization capabilities. Virtualization can lower costs required to create, manage, and run enterprise applications by using existing hardware resources. This package also contains additional management features like application edition and dynamic application management, management of heterogeneous application servers, and service policy management, which provide an enhanced management environment for existing infrastructure.

WebSphere eXtreme Scale

This product, formerly WebSphere Extended Deployment Data Grid, allows business applications to process large volumes of transactions with efficiency and linear scalability. WebSphere eXtreme Scale operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers. It provides transactional integrity and transparent failover to ensure high availability, high reliability, and constant response times. WebSphere eXtreme Scale is a software technology for conducting extreme transaction processing.

Compute Grid

This product enables the scheduling, execution, and monitoring of batch type jobs and compute-intensive tasks with service policy and workload management. This composition of packages delivers enhanced qualities of service with features for optimizing IT resources and can be used in collaboration with WebSphere Application Server packages.

Some of the benefits of WebSphere Extended Deployment are as follows:

Provides virtualization capabilities that can dynamically match available resources to changing workload demands. It enables more efficient resource use as well as improved application performance and scalability.

Features virtualization, and workload and health management for the following application environments:

PHP

BEA WebLogic

JBoss

Apache Tomcat

WebSphere Application Server Community Edition

Offers enhanced capabilities in job scheduling, monitoring, and management for batch-type workloads.

Delivers customizable health policies and actions to help enhance manageability.

Supports innovative and highly scalable data fabrics to accelerate data-intensive application performance.

For more information about WebSphere Extended Deployment, see the following Web page: http://www.ibm.com/software/webservers/appserv/extend/

The node agent must be running before any application server on that node. The reason is that the node agent hosts the Location Service Daemon that is needed by the application server and application clients. File Synchronization occurs either automatically or can happen manually via syncNode.sh script. File synchronization is a one-way process from the DM to the node agents and utilizes the HTTP(s) protocol. At synchronization time, the node agent send message digests to the DM, where they are compared against the digests in the master repository, consisting of:

Configuration files for all processes.

J2EE Application (Deployment descriptors files and binaries).

Workspace is a temporary space given to the user making Administrative Configuration changes – not needed for operational changes. Configuration XML files are copied from the master repository and cached into workspace. Different temporary space is maintained for each user. Workspaces are retained persistently and can be reused in the next login.

Vertical scaling

Vertical scaling refers to configuring multiple application servers on a single machine and creating a cluster of associated application servers all hosting the same J2EE application(s). In this case, the Web server plug-in routes the requests according to the application servers availability. Load balancing is performed at the Web server plug-in level based on a round-robin algorithm and with consideration of session state. Failover is also possible as long as there are active application servers (JVMs) on the system. Vertical scaling can be combined with other topologies to boost performance, throughput and availability.

Vertical scaling – many processes on one machine; single point of failure

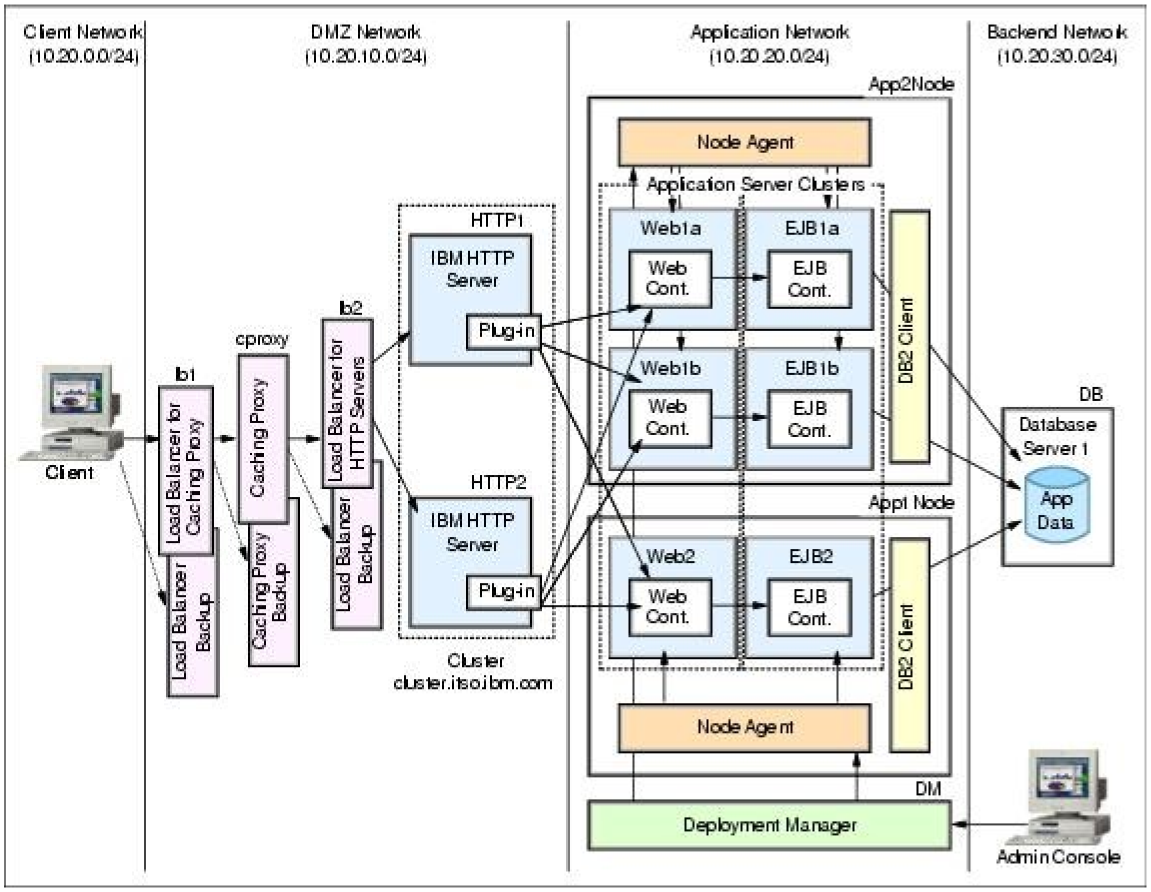
Horizontal Scaling

Horizontal scaling exists when the cluster members are located across multiple machines. This lets a single application span over several machines, yet still presenting the application as a single logical image. The Web server plug-in distributes requests to the cluster members on each node and performs load balancing and failover. If the Web server (Server A) goes down, then the WebContainer Inbound Chain of Server B or C could be utilized (limited throughput) meanwhile Server A or the Web server on Server A is repaired. Be aware that this configuration introduces a single point of failure; when the HTTP server is out of service, your entire application is inaccessible from the outside network (internal users could still access the application server(s) using the WebContainer Inbound Chain). You can omit this SPOF by adding a backup Web server. A Load Balancer, part of the WebSphere Edge Components, can be configured to create a cluster of Web servers and add it to a cluster of application servers. Load balancing products can be used to distribute HTTP requests among Web servers that are running on multiple physical machines. The Dispatcher component of Load Balancer, which is part of the WebSphere Edge Components, is an IP sprayer that performs intelligent load balancing among Web servers based on server availability and workload capacity as the main selection criteria to distribute the requests The Load Balancer Node sprays Web client requests to the Web servers. The Load Balancer is configured in cascade. The primary Load Balancer communicates to his backup through a heartbeat to perform failover, if needed, and thus eliminates the Load Balancer Node as a single point of failure. Both Web servers perform load balancing and failover between the application servers (cluster members) through the Web server plug-in.

The main theme with network deployment is distributed applications.

Horizontal scaling uses clustering and eliminates single points of process and hardware failure.

D) Illustrate workload management and failover strategies using IBM WebSphere Application Server Network Deployment V7.0.



This configuration provides both the greatest resiliency of a site, and the greatest administrative complexity. The topology includes the following elements:

Two IBM HTTP Server Web servers configured in a cluster

Incoming requests for static content are served by the Web server. Requests for dynamic content is forwarded to the appropriate application server by the Web server plug-in.

A Caching Proxy that keeps a local cache of recently accessed pages

Caching Proxy is included in WebSphere Application Server Edge Components. Cacheable content includes static Web pages and JSPs with dynamically generated but infrequently changed fragments. The Caching Proxy can satisfy subsequent requests for the same content by delivering it directly from the local cache, which is much quicker than retrieving it again from the content host.

A backup server is configured for high availability.

A Load Balancer to direct incoming requests to the Caching Proxy and a second Load Balancer to manage the workload across the HTTP servers

Load Balancer is included in WebSphere Application Server Edge Components. The Load Balancers distribute incoming client requests across servers, balancing workload and providing high availability by routing around unavailable servers.

A backup server is configured for each primary Load Balancer to provide high availability.

A dedicated server to host the deployment manager

The deployment manager is required for administration but is not critical to the runtime execution of applications. It has a master copy of the configuration that should be backed up on a regular basis.

Two clusters consisting of three application servers

Each cluster spans two machines. In this topology, one cluster contains application servers that provide the Web container functionality of the applications (servlets, JSPs), and the second cluster contains the EJB container functionality. Whether you choose to do this or not is a matter of careful consideration. Although it provides failover and workload management capabilities for both Web and EJB containers, it can also affect performance.

A dedicated database server running IBM DB2 V9.

Core group collection

A core group is a component of the high availability manager function. A default core group, called DefaultCoreGroup, is created for each cell in the product environment. A core group can contain standalone servers, cluster members, node agents and the deployment manager. A core group must contain at least one node agent or the deployment manager.

A cell must contain at least one core group, although multiple core groups are supported. Each core group contains a core group coordinator to manage its high availability relationships, and a set of high availability policies that are used to manage the highly available components within that core group.

Peer core groups are core groups that reside in different cell. The local core group bridge attempts to establish communication between peer core groups in the order in which they appear in the list of peer core groups.

The peer bridge endpoint is used as a bridge interface to another peer core group. This bridge interface enables members of peer core groups to use the core group bridge service to communicate with other even though they reside in different cells. One or more peer bridge endpoints can be specified for a peer core group.

Core group members

For transitioning users: The requirement that every core group must contain at least one node agent or the deployment manager, that exists in Versions 6.0.x and 6.1.x, does not apply to for this version of the product. However, if you are running in a mixed cell environment, every core group that contains any Version 6.x members must also contain at least one node agent or the deployment manager. trns

Every deployment manager, node agent, application server, and proxy server is a member of a core group. When a process is created it is automatically added to a core group. The core group membership is stored in a product configuration document. You can move processes from one core group to another. The following rules govern the core group membership:

* Every process is a member of exactly one core group.

* All members of a cluster must be members of the same core group.

A core group member has a well-defined life cycle. When the first core group member starts, the transport that is dedicated to that core group automatically starts. The Discovery Protocol, View Synchrony Protocol, and Failure Detection Protocol for that core group member also start and run for the entire lifetime of the core group member:

* The Discovery Protocol is responsible for discovering when other core group processes start, and for opening network connections to these other members.

* The View Synchrony Protocol is responsible for establishing reliable messaging with other core group members after the connections are opened.

* The Failure Detection Protocol is responsible for detecting when other core group members stop or become unreachable because of a network partition.

Core group coordinator

The core group coordinator is responsible for coordinating high availability activities between the core group members for which View Synchrony Protocol is established.

Core group transport

Network communication between all the members of a core group is essential. The network environment must consist of a fast local area network (LAN) with full Internet Protocol (IP) visibility and bidirectional communication between all core group members. Each core group member must be able to receive communications from any of the other core group members.

Multiple core groups

A cell, by default, contains a single core group, called DefaultCoreGroup. All processes in the cell are initially members of this core group. A single core group is usually sufficient. However, some topologies or special circumstances require multiple core groups. There are also topologies that do not require multiple core groups but having them is a good practice. For example, you might want to define multiple core groups if :

* A large number of processes in the cell and the core group protocols, such as the View Synchrony Protocol, consume correspondingly large amounts of resources such as CPU.

* Core group protocols, such as the Failure Detection Protocol, need tuning or configuring to use values that work best with smaller numbers of core group members.

If members of different core groups need to share workload management or on-demand configuration routing information, use the core group bridge service to connect these core groups. The core group bridge service uses access point groups to connect the core groups. A core group access point defines a set of bridge interfaces that resolve to IP addresses and ports. The core group bridge service uses this set of bridge interfaces to enable members of one core group to communicate with members of another core group.

High availability manager

The product includes a high availability manager component. The services that the high availability manager provides are only available to product components.

A high availability manager provides several features that allow other product components to make themselves highly available. A high availability manager provides:

* A framework that allows singleton services to make themselves highly available. Examples of singleton services that use this framework include the transaction managers for cluster members, and the default messaging provider, also known as the service integration bus.

* A mechanism that allows servers to easily exchange state data. This mechanism is commonly referred to as the bulletin board.

* A specialized framework for high speed and reliable messaging between processes. This framework is used by the data replication service when the product is configured for memory-to-memory replication.

A high availability manager instance runs on every application server, proxy server, node agent and deployment manager in a cell. A cell can be divided into multiple high availability domains known as core groups. Each high availability manager instance establishes network connectivity with all other high availability manager instances in the same core group, using a specialized, dedicated, and configurable transport channel. The transport channel provides mechanisms which allow the high availability manager instance to detect when other members of the core group start, stop, or fail.

Within a core group, high availability manager instances are elected to coordinate high availability activities. An instance that is elected is known as a core group coordinator. The coordinator is highly available, such that if a process that is serving as a coordinator stops or fails, another instance is elected to assume the coordinator role, without loss of continuity.

Highly available components

A highly available component is a component for which a high availability group is defined on the processes where that component can run. The coordinator tracks high availability group membership, and knows on which processes each highly available component can run.

The coordinator also associates a high availability policy with each high availability group. A high availability policy is a set of directives that aid the coordinator in managing highly available components. For example, a directive might specify that a component runs on a specific process, if that process is available. Directives are configurable, which makes it possible for you to tailor policies to your installation.

The coordinator is notified as core group processes start, stop or fail and knows which processes are available at any given time. The coordinator uses this information, in conjunction with the high availability group and policy information, to ensure that the component keeps functioning. The coordinator uses the policy directives to determine on which process it starts and runs each component. If the chosen process fails, the coordinator restarts the component on another eligible process. This reduces the recovery time, automates failover, and eliminates the need to start a replacement process.

State data exchange

The high availability manager provides a specialized messaging mechanism that enables processes to exchange information about their current state. Each process sends or posts information related to its current state, and can register to be notified when the state of the other processes changes. The workload management (WLM) component uses this mechanism to build and maintain routing table information. Routing tables built and maintained using this mechanism are highly available.

Replication

The data replication service (DRS) that is provided with the product, is used to replicate HTTP session data, stateful EJB sessions, and dynamic cache information among cluster members. When DRS is configured for memory-to-memory replication, the transport channels defined for the high availability managers are used to pass this data among the cluster members.

When to use a high availability manager

A high availability manager consumes valuable system resources, such as CPU cycles, heap memory, and sockets. These resources are consumed both by the high availability manager and by product components that use the services that the high availability manager provides. The amount of resources that both the high availability manager and these product components consume increases nonlinearly as the size of a core group increases.

For large core groups, the amount of resources that the high availability manager consumes can become significant. Disabling the high availability manager frees these resources. However, before you disable the high availability manager, you should thoroughly investigate the current and future needs of your system to ensure that disabling the high availability manager does not also disable other functions that you use that require the high availability manager. For example, both memory to memory session replication, and remote request dispatcher (RRD) require the high availability manager to be enabled.

The capability to disable the high availability manager is most useful for topologies where none of the high availability manager provided services are used. In certain topologies, only some of the processes use the services that the high availability manager provides. In these topologies, you can disable the high availability manager on a per-process basis, which optimizes the amount of resources that the high availability manager uses.

Do not disable the high availability manager on administrative processes, such as node agents and the deployment manager, unless the high availability manager is disabled on all application server processes in that core group.

Some of the services that the high availability manager provides are cluster based. Therefore, because cluster members must be homogeneous, if you disable the high availability manager on one member of a cluster, you must disable it on all of the other members of that cluster.

When determining if you must leave the high availability manager enabled on a given application server process, consider if the process requires any of the following high availability manager services:

* Memory-to-memory replication
* Singleton failover
* Workload management routing
* On-demand configuration routing

Memory-to-memory replication

Memory-to-memory replication is a cluster-based service that you configure or enable at the application server level. If memory-to-memory replication is enabled on any cluster member, then the high availability manager must be enabled on all of the members of that cluster. Memory-to-memory replication is automatically enabled if:

* Memory-to-memory replication is enabled for Web container HTTP sessions.
* Cache replication is enabled for the dynamic cache service.
* EJB stateful session bean failover is enabled for an application server.

Singleton failover

Singleton failover is a cluster-based service. The high availability manager must be enabled on all members of a cluster if:

* The cluster is configured to use the high availability manager to manage the recovery of transaction logs.
* One or more instances of the default messaging provider are configured to run in the cluster. The default messaging provider that is provided with the product is also referred to as the service integration bus.

Workload management routing

Workload management (WLM) propagates the following classes or types of routing information:

* Routing information for enterprise bean IIOP traffic.
* Routing information for the default messaging engine, which is also referred to as the service integration bus.
* Routing HTTP requests through the IBM® WebSphere® Application Server proxy server.
* Routing Web Services Addressing requests through the IBM WebSphere Application Server proxy server.
* Routing SIP (Session Initiation Protocol) requests.

WLM uses the high availability manager to both propagate the routing information and make it highly available. Although WLM routing information typically applies to clustered resources, it can also apply to non-clustered resources, such as standalone messaging engines. During typical circumstances, you must leave the high availability manager enabled on any application server that produces or consumes either IIOP or messaging engine routing information.

For example, if:

* The routing information producer is an enterprise bean application that resides in cluster 1.

* The routing information consumer is a servlet that resides in cluster 2.

When the servlet in cluster 2 calls the enterprise bean application in cluster 1, the high availability manager must be enabled on all servers in both clusters.

Workload management provides an option to statically build and export route tables to the file system. Use this option to eliminate the dependency on the high availability manager.

<u>On-demand configuration routing</u>

In a Network Deployment system, the on-demand configuration is used for IBM WebSphere Application Server proxy server routing. If you want to use on-demand configuration routing in conjunction with your Web services, you must verify that the high availability manager is enabled on the proxy server and on all of the servers to which the proxy server routes work.

<u>Resource adapter management</u>

New feature: In a high availability environment, you can configure your resource adapters for high availability. After a resource adapter is configured for high availability, the high availability manager assigns the resource adapter to a high availability group, the name for which is derived from the resource adapter key. The resource adapter key is the cell-scoped configuration ID of the resource adapter, and must be identical for all of the servers in a cluster that use the same resource adapter. The high availability manager then controls when each resource adapter is started. newfeat

When you configure a resource adapter for high availability, select one of the following types of failover:

* Message endpoint (MEP) failover. When a resource adapter is configured for MEP failover, that resource adapter can be active within any member of a high availability group, but only supports inbound communication within one high availability group member.

* Resource adapter (RA) instance failover. When a resource adapter is configured for RA instance failover, that resource adapter can support either outbound or inbound communication within one high availability group member at a time.

When a resource adapter that is configured for high availability starts, the Java™ EE Connector Architecture (JCA) container joins the resource adapter into the high availability group. This high availability group is configured to run under the one of n policy with quorum disabled. When MEP failover is selected, the container starts the adapter with outbound communication enabled, but disables inbound communication because the high availability manager controls inbound communication for that resource adapter. When RA instance failover is selected, the container starts the adapter and disables both outbound and inbound communication because the high availability manager controls both inbound and outbound communication for that resource adapter.

When the run time stops a resource adapter that is configured for high availability, the JCA container removes the resource adapter from the high availability group to which it was assigned.

Disabling or enabling a high availability manager

A unique HAManagerService configuration object exists for every core group member. The enable attribute in this configuration object determines if the high availability manager is enabled or disabled for the corresponding process. When the enable attribute is set to true, the high availability manager is enabled. When the enable attribute is set to false, the high availability manager is disabled. By default, the high availability manager is enabled. If the setting for the enable attribute is changed, the corresponding process must be restarted before the change goes into effect. You must use the wsadmin tool to disable or enable a high availability manager.

Before you begin

Determine if you need to use a high availability manager to manage members of a core group.

About this task

You might want to disable a high availability manager if you are trying to reduce the amount of resources, such as CPU and memory, that the product uses and have determined that the high availability manager is not required on some or all of the processes in a core group.

You might need to enable a high availability manager that you previously disabled because you are installing applications on core group members that must be highly available.

Complete the following steps if you need to disable a high availability manager or to enable a high availability manager that you previously disabled.

Procedure

1. In the administrative console, navigate to the Core group service page for the process.

 * For a deployment manager, click System Administration > Deployment manager > Core group service.

 * For a node agent, click System Administration > Node agent > node_agent > Core group service.

 * For an application server, click Servers > Server Types > WebSphere application servers > server_name > Core group service.

2. If you want to disable the high availability manager for this process, deselect the Enable service at server startup option.

3. If you want to enable the high availability manager for this process, select the Enable service at server startup option.

4. Click OK and then click Review.

5. Select Synchronize changes with nodes, and then click Save.

6. Restart all of the processes for which you changed the Enable service at server startup property setting.

To verify that the high availability manager is in the proper state, check the log file for one of the following messages:

HMGR0005I: The Single Server DCS Core Stack transport has been started for core group DefaultCoreGroup.

This message indicates that the high availability manager is disabled because the high availability manager communication transport can only establish communication with a single server process.

HMGR0001I: The DCS Core Stack transport has been started for core group DefaultCoreGroup. There are x members.

This message indicates that the high availability manager is enabled because the high availability manager communication transport can establish communication with multiple server processes. x indicates the number of server processes with which communication is established.

Workload management optimizes the distribution of client processing tasks. Incoming work requests are distributed to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests.

Workload management provides the following benefits to applications that are installed on the product:

 * It balances client workloads, allowing processing tasks to be distributed according to the capacities of the different machines in the system.

 * [AIX Solaris HP-UX Linux Windows] It provides failover capability by redirecting client requests if one or more servers is unable to process them. This improves the availability

of applications and administrative services.

* It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With clustering, additional instances of servers, servlets, and other objects can easily be added to the configuration.

* It enables servers to be transparently maintained and upgraded while applications remain available for users.

* It centralizes the administration of servers and other objects.

In the product environment, you use clusters, transports, and replication domains to implement workload management.

Clusters and workload management

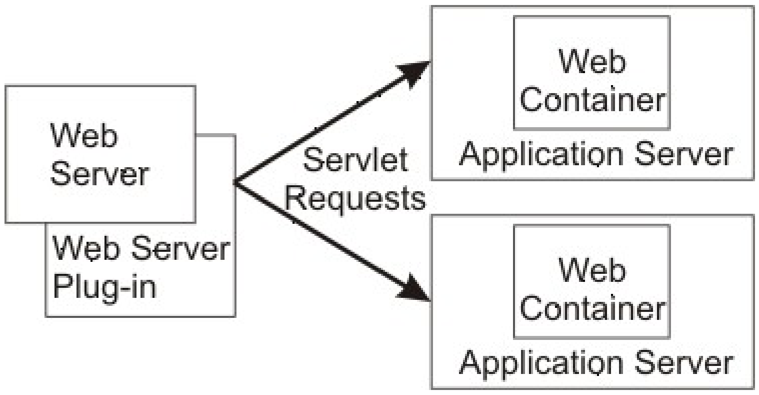
Clusters are sets of servers that are managed together and participate in workload management. Clusters enable enterprise applications to scale beyond the amount of throughput capable of being achieved with a single application server. Clusters also enable enterprise applications to be highly available because requests are automatically routed to the running servers in the event of a failure. The servers that are members of a cluster can be on different host machines. In contrast, servers that are part of the same node must be located on the same host machine. A cell can include no clusters, one cluster, or multiple clusters.

Servers that belong to a cluster are members of that cluster set and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data. One cluster member might be running on a huge multi-processor enterprise server system, while another member of that same cluster might be running on a smaller system. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical. This allows client work to be distributed across all the members of a cluster instead of all workload being handled by a single application server.

When you create a cluster, you make copies of an existing application server template. The template is most likely an application server that you have previously configured. You are offered the option of making that server a member of the cluster. However, it is recommended that you keep the server available only as a template, because the only way to remove a cluster member is to delete the application server. When you delete a cluster, you also delete any application servers that were members of that cluster. There is no way to preserve any member of a cluster. Keeping the original template intact allows you to reuse the template if you need to rebuild the configuration.

A vertical cluster has cluster members on the same node, or physical machine. A horizontal cluster has cluster members on multiple nodes across many machines in a cell. You can configure either type of cluster, or have a combination of vertical and horizontal clusters.

[AIX Solaris HP-UX Linux Windows] [iSeries] Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. The routing of servlet requests occurs between the Web server plug-in and clustered application servers using HTTP transports, or HTTP transport channels.



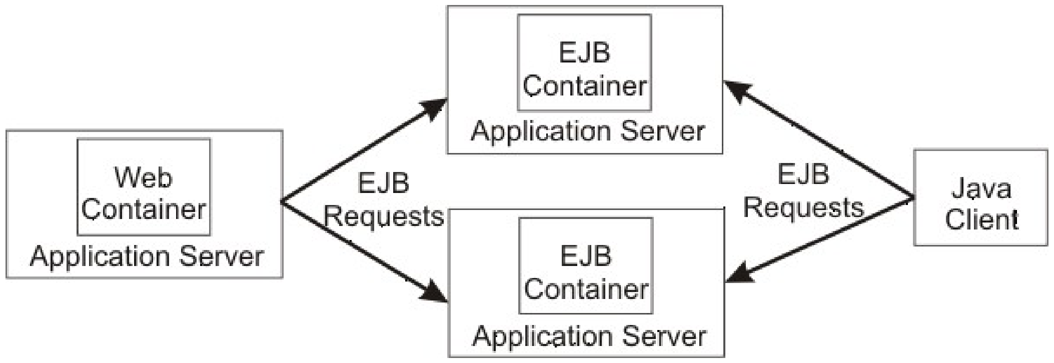
This routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster, assuming there are no strong affinity configurations. If the weights are scaled in the range from zero to twenty, the plug-in usually routes requests to those cluster members with the higher weight values.

You can use the administrative console to specify a weight for a cluster member. The weight you assign to a cluster member should be based on its approximate, proportional ability to do work. The weight value specified for a specific member is only meaningful in the context of the weights you specify for the other members within a cluster. The weight values do not indicate absolute capability. If a cluster member is unavailable, the Web server plug-in temporarily routes requests around that cluster member.

For example, if you have a cluster that consists of two members, assigning weights of 1 and 2 causes the first member to get approximately 1/3 of the workload and the second member to get approximately 2/3 of the workload. However, if you add a third member to the cluster, and assign the new member a weight of 1, approximately 1/4 of the workload now goes to the first member, approximately 1/2 of the workload goes to the second member, and approximately 1/4 of the workload goes to the third member. If the first cluster member becomes unavailable, the second member gets approximately 2/3 of the workload and third member gets approximately 1/3 of the workload.

The weight values only approximate your load balance objectives. There are other application dependencies, such as thread concurrency, local setting preferences, affinity, and resource availability that are also factors in determining where a specific request is sent. Therefore, do not use the exact pattern of requests to determine the weight assignment for specific cluster members.

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers can be clustered with the EJB containers, enabling the distribution of enterprise bean requests between EJB containers on different application servers.



In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java™ programs using RMI/IIOP, or other EJBs.

The server weighted round robin routing policy ensures a balanced routing distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster is that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism sends more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members.

Node Agents, Cluster Members, and the DM can be members of a Core Group. A process can be a member of one and only one Core Group. Additionally, all members of a cluster must be members of the same Core Group. Members of a Core Group share information among one another, so that they are aware of the status of other Core Group members. Singleton services, such as WLM routing, running on one member of a Core Group can be failed over to any other member of the Core Group.

When the DM is installed, a default Core Group named "DefaultCoreGorup" is created. It has HA policies for the Transaction Manager and Messaging Engines predefined. As managed processes are added to the cell, they are automatically added to the DefaultCoreGroup.

Each Core Group has an HA Coordinator that manages the HA related work for the Core Group. It keeps track of what services are running in what processes and, in the case of failure, decides which process should restart that service. In cases w here you have several clusters within your cell, it can be useful to configure more than one HA Coordinator in your cell, so that a single process does not get overloaded.

The Transaction Log Hot Standby allows hot failover of in-transit two-phase commit transactions. To set this, you need to check the box labeled, "Enable high availability for persistent services" when configuring your cluster.

An HA policy defines how failover occurs and which servers to use for failover. A Core Group can have different HA policies for different services. WLM clustering for transaction can use one HA policy, while messaging Engines can use another HA policy. By default, the Core Group has a clustered transaction manager policy and a messaging policy for Service Integration Buses.

In an HA policy, singleton services can be configured to run in three basic modes:

Static

1 of N

No Operation

In a Static policy, singleton services run on a single server, and the failure of that server will not result in another server recovering that service. The service will only be reactivated when the failed server is restarted.

In the "1 of N" policy the HA coordinator is enabled to manager failover. The coordinator determines which processes should run which services and assigns them accordingly. The coordinator will ensure that the singleton service is running on exactly one server at any given time. This is the suggested HA policy for most users. When configuring a "1 of N" policy you can choose "Preferred servers only", "Fail back" or "Quorum".

With Preferred servers only, singleton services will run only on servers in the preferred list, and if none of them are available, the singleton will fail.

With Fail back policy, the HA Manager that after a process has failed over, the process should be moved back to a preferred server when that server becomes available.

The "Quorum" policy specificies that singleton services should only be started if at least half of the servers in the Core Group are running. This means that if more than half of your servers fail, your singleton services will be automatically stopped.

The "No Operation" policy specificies that WAS will never activate singletons on its own. Instead, it will be told what to do using JMX commands. The value of this policy is that allows you to make the HA Coordinator aware of the actions taken by external clustering software, so that applications will always have access to the necessary resources.

Any of these options can be changed dynamically using wsadmin.

The HA Manager uses two parallel methods to monitor processes: TCP Keep-Alive sockets, and an active hearbeat. Both of these methods are used across the Core Group in a peer-to-peer fashion. You should tune your OS's TCP KEEP_ALIVE value to ensure that failures are detected in a reasonable time.

The second method used to determine server status is an active heartbeat. Each process sends a heart beat message to every other process once per 'N' seconds. If a peer fails to respond to 'M' consecutive heart beats, then services from that process will be failed over to other processes.

HA policies decide how and where services will be failed over to.

WLM is implemented by using Clusters of application servers.

What can be workload managed?

- HTTP requests - Spread across multiple web servers by an edge product.
- Servlet requests - Spread across multiple web containers by the web server plugin
- EJB requests - Spread across multiple EJB containers by the WLM service.
- Partitioned Messaging destinations - Spread across multiple Messaging Engines by the WLM survive
- Web services outbound requests - Routed directly by WLM service, no longer require external routing by the web server plugin

The Location Service Daemon process is responsible for the EJB routing table, which can have entries for servers in other clusters.

HA does not depend on the DM.

For failover of stateful session EJBs:

The DRS is used, which is similar to HTTP session failover.

This can be enabled on a per-Application basis.

WLM will fail beans over to a server that already has a copy of the session data in memory if possible.

Ability to collocate stateful session bean replicas with http session replicas with hot failover.

By definition, clusters are a set of application servers having the same applications installed, and grouped logically for WLM.

Cluster configuration is as follows:

Prefer local - Routes EJB client requests to local EJB, if possible.

Enable HA for persistent services(for transaction log recovery)

Backup Cluster - If the entire cluster fails, the backup cluster supplies failover routing information

Cluster member weights.

To install applications to a cluster select a cluster, rather than selecting a server. Required resources, such as databases, must be available to all cluster members. New in there is a rollout update option available on clusters.

Basic WLM Request Routing

Load Balancer - The load balancer is an IP sprayer that makes intelligent load balancing decisions. Using the NDAdmin tool, you can set it up to route your HTTP servers based on round robin, statistical round robin, best, custom advisor, or content based routing. Once the request arrives at an HTTP server, the routing is weighted round robin - the only configuration option is how much 'weight' to give eachserver. The routing information, the list of available servers and their weights, is ultimately stored by the DM. When the HTTP server plugin is generated, servlet request routing weights are written into the plugin.xml file, which the HTTP server will reload at configurable intervals. When a client requests an IOR for an EJB, the LSD returns the IOR and a copy of the routing table. The client uses two in-memory copies of the table - one static, one dynamic. The static copy is only to cache a local copy of the weights.

If an HTTP server fails, the load balancer will simply route around it. The plugin reads the cloneID from the session key, and can route the request to its originating server. If a server process fails, the HTTP server notes the failure and marks that application server as unavailable, then routes the request to the next cluster member. Sessions already in progress will have a server ID for that failed server; the HTTP server routes them to the next server. Session data can be handled in two ways. Session Persistence to a database, or internal messaging of session information. The scope of the replication domain is the core group; DRS will coordinate with the WLM component to determine which cluster members should hold backup session information for a specific cluster member. Other than the fact that Node Agents will not be notified of failed servers on other nodes, the impact of the DM going down is minimal. Singleton services, like the WLM routing component, are 'portable' in , they do not run in the DM.

There are two kinds of high availability: process high availability and data high availability.

Redundant hardware and clustering software are approaches to high availability.

We can divide availability into the following levels:

1. Basic systems. Basic systems do not employ any special measures to protect data and services, although backups are taken regularly. When an outage occurs, the support personnel restores the system from backup (usually tape).

2. Redundant data. Disk redundancy and/or disk mirroring are used to protect the data against the loss of a disk. Full disk mirroring provides more data protection than RAID-5.

3. Component failover. For an e-business infrastructure like WebSphere, there are many components. As we discussed above, an outage in any component may result in service interruption. Multiple threads or multiple instances can be employed for availability purposes. For example, if we do not make the firewall component highly available, it may cause the whole system to go down (worse than that, it may expose your system to hackers) even though the servers are highly available.

For WebSphere, we have process high availability (vertical scaling) and process and node high availability (horizontal scaling). Entity EJBs are persisted into the database. Highly available data management is critical for a highly available transactional system. Therefore, it is very important to balance the availability of all components in the WebSphere production system. Do not overspend on any particular component, and do not underspend on other components, either.

4. System failover. A standby or backup system is used to take over for the primary system if the primary system fails. In principle, any kind of service can become highly available by employing system failover techniques. However, this will not work if the software is hard-coded to physical host-dependent variables. We can configure the systems as active/active mutual takeover or active/standby takeover. Although the active/active mutual takeover configuration increases the usage of hardware, it also increases the possibility of interruption, and hence reduces the availability. In addition, it is not efficient to include all components into a single cluster system. We have a firewall cluster, LDAP cluster, WebSphere server cluster, and database cluster. In system failover, clustering software monitors the health of the network, hardware, and software process, detects and communicates any fault, and automatically fails over the service and associated resources to a healthy host. Therefore, you can continue the service before you repair the failed system. As we discussed before, as MTTR approaches zero, A increases toward 100%. System failover can also be used for planned software and hardware maintenance and upgrades.

5. Disaster recovery. This applies to maintaining systems in different sites. When the primary site becomes unavailable due to disasters, the backup site can become operational within a reasonable time. This can be done manually through regular data backups, or automatically by geographical clustering software.

E) Describe WebSphere dynamic caching features.

Dynamic cache works within an application server Java virtual machine (JVM), intercepting calls to cacheable objects. For example, it intercepts calls through a servlet service method, or a command execute method, and either stores the output of the object to the cache or serves the content of the object from the dynamic cache.

Key concepts pertaining to the dynamic cache service

Explore the key concepts pertaining to the dynamic cache service, which improves performance by caching the output of servlets, commands, Web services, and JavaServer Pages (JSP) files.

Cache instances

An application uses a cache instance to store, retrieve, and share data objects within thedynamic cache.

Using the dynamic cache service to improve performance

Caching the output of servlets, commands, and JavaServer Pages (JSP) improves application performance. WebSphere Application Server consolidates several caching activities including servlets, Web services, and WebSphere commands into one service called the dynamic cache. These caching activities work together to improve application performance, and share many configuration parameters that are set in the dynamic cache service of an application server.

Configuring dynamic cache to use the WebSphere eXtreme Scale dynamic cache provider [Fix Pack 5 or later]

Configuring the dynamic cache service to use WebSphere eXtreme Scale lets you leverage transactional support, improved scalability, high availability, and other WebSphere eXtreme Scale features without changing your existing dynamic cache caching code.

Configuring servlet caching

After a servlet is invoked and completes generating the output to cache, a cache entry is created containing the output and the side effects of the servlet. These side effects can include calls to other servlets or JavaServer Pages (JSP) files or metadata about the entry, including timeout and entry priority information.

Configuring portlet fragment caching

After a portlet is invoked and completes generating the output to cache, a cache entry is created containing the output and the side effects of the portlet. These side effects can include calls to other portlets or metadata about the entry, including timeout and entry priority information.

Eviction policies using the disk cache garbage collector

The disk cache garbage collector is responsible for evicting objects out of the disk cache, based on a specified eviction policy.

Configuring the JAX-RPC Web services client cache

The Web services client cache is a part of the dynamic cache service that is used to increase the performance of Web services clients by caching responses from remote Web services.

Cache monitor

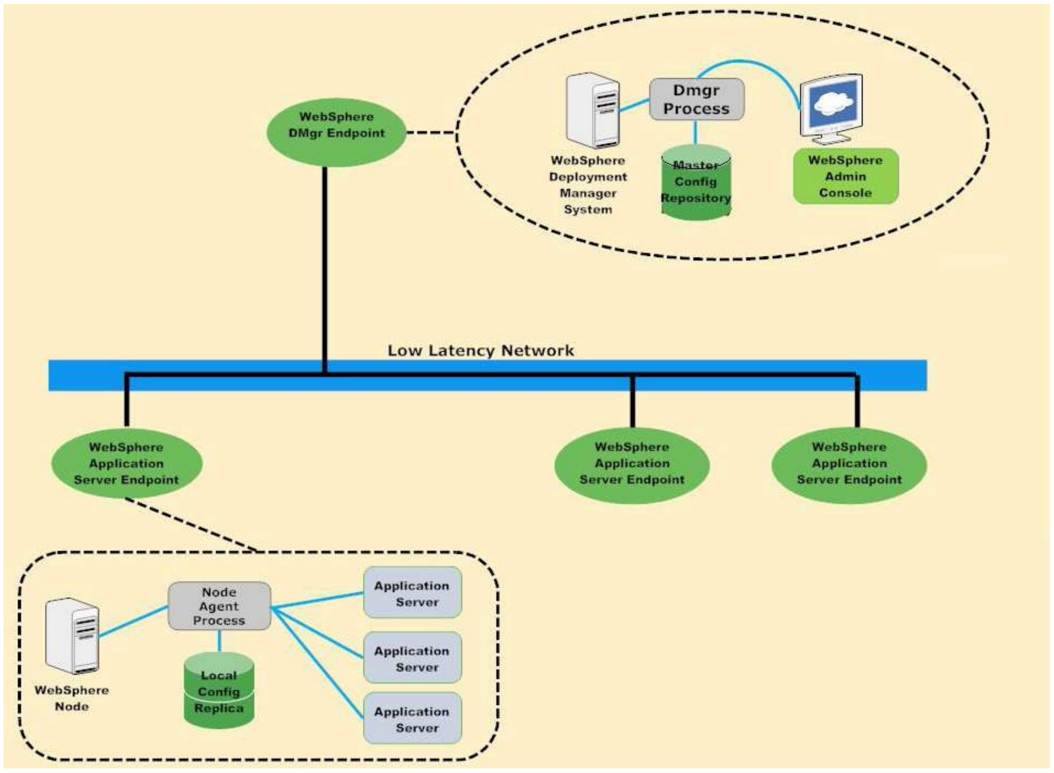
Cache monitor is an installable Web application that provides a real-time view of the current state of dynamic cache. You use it to help verify that dynamic cache is operating as expected. The only way to manipulate the data in the cache is by using the cache monitor. It provides a GUI interface to manually change data.

Invalidation listeners

Invalidation listener mechanism uses Java events for alerting applications when contents are removed from the cache.

F) Compare the Network Deployment (ND) cell model with the flexible management model.

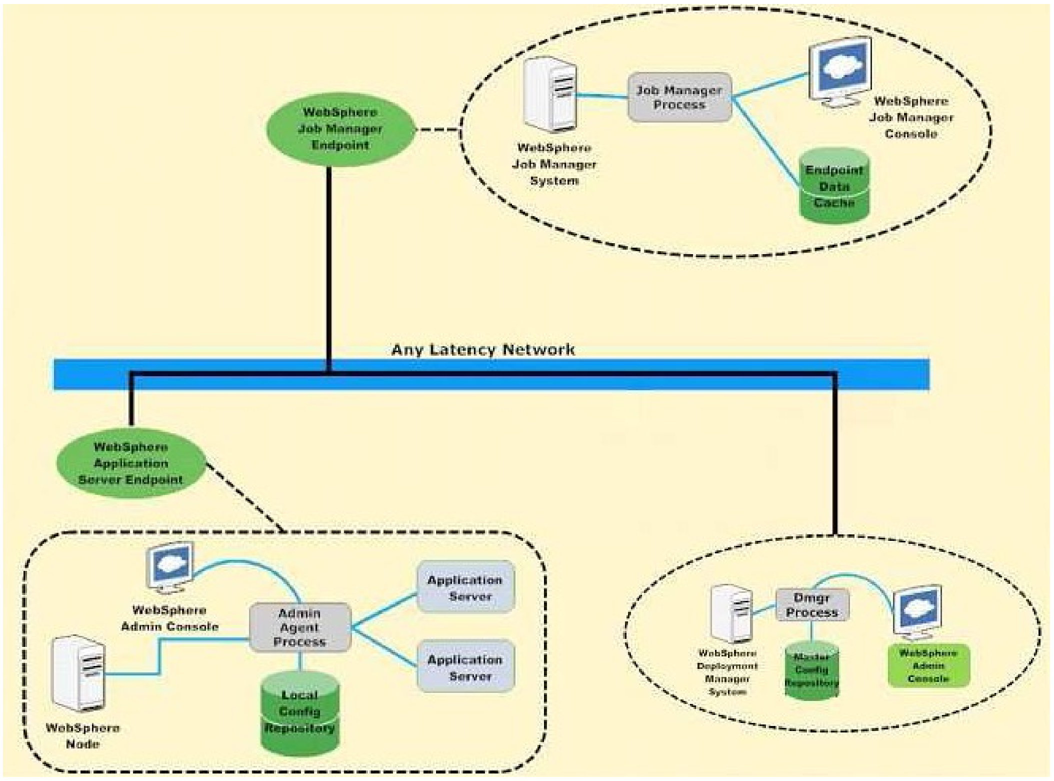
Network Deployment Model



ND Model Limitations

- Model expects tight coupling, highly synchronous environment
- Management is at the individual server level - Does not support management at the node level
- Scalability - Problems managing very large number of base servers.
- Synchronous model has problems with high latency remote branch servers.

Flexibile Management Topology



Flexible management characteristics

- Loose coupling of various pieces
- Offers asynchronous job queuing mechanism for administration purposes
- An alternative to the cell model
- Offers administrators additional management options not previously available.
- Management of multiple base servers, up to a server farm containing hundreds of base servers.
- Coordinate management actions across multiple deployment managers with job agents.

Flexible management environments rely on asynchronous processing of work units (known as jobs) from the job manager. This lends itself to large scaling and can support many application servers without degrading performance. It also reduces latency and bandwidth requirements on the network; even dialup lines to remote sites can work well without slowing down the overall system. Additionally, configuration information does not exist beyond the node level so there is no bottleneck associated with accessing a master configuration repository.

Flexible management is not a replacement for the network deployment model but can be used as an alternative to it, and the two can be combined by having a job manager coordinate management actions across multiple deployment managers.

While flexible management enables some management scenarios, including very large server farms and remote branch operations, there are also some capabilities that are not available outside of a cell environment, including clustering, memory-to-memory data replication, and performance monitoring from the administrative console.

Flexible Management Components

The administrative agent is a new profile type to support flexible management.

Register base application server profiles with the administrative agent.

reduces application server footprint

migrates administrative functions to administrative agent

most administrative function is removed from base application

maintains isolation between base nodes; no master repository or synchronization

Register base application server profiles with the job manager through the administrative agent

Administrative agent polls job manager for jobs on behalf of application servers

Job manager

- New profile type to support flexible management
- Use administrative agent to register base server profiles with a job manager
- To manage multiple cells, register deployment managers with job manager directly
- Use job manager to queue job for registered profiles
- Registered profiles retain autonomy and can be managed without the job manager
- Scales to support large number of registered profiles

The job manager is a new server type that was added to support flexible management. It is a new profile type, and the various tools that can create profiles have been modified to support creation and maintenance of this profile. The job manager is absolutely central to flexible management.

To participate in flexible management, a base server first registers itself with the administrative agent. The base server must then register with the job manager. If a deployment manager wants to participate in an environment controlled by a job manager, the deployment manager registers directly with the job manager; no administrative agent is involved in this case.

The main use of the job manager is to queue jobs to application servers in a flexible management environment. These queued jobs are pulled from the job manager by the administrative agent and distributed to the appropriate application server or servers.

Characteristics of flexible management jobs

- Have semantics tied to the underlying node - For example, application management jobs
- Have few parameters - Most application installation bindings are stored with the application.
- As asynchronous, with activation and expiration times, and can recur
- Can send email notification upon completion
- Have status that shows # of nodes succeeded, failed, or pending.

The units of work that are handled by the flexible management environment are known as jobs. The semantics of these jobs are typically straightforward, and the jobs require few parameters. The jobs are processed asynchronously and can have an activation time, expiration time, and a recurrence indicator. You can specify that an e-mail notification be sent upon completion of a job. Additionally, you can view the current status of a job by issuing a status command.

Examples of flexible management jobs

- Examples
- download application binaries
- install applications
- create/start/stop application servers
- wsadmin scripts

Summary

Flexible management is an asynchronous, scalable deployment architecture

Flexible management components include the administrative agent and job manager

Section 2 - Installation/Configuration of WebSphere Application Server (9%)

A) Identify installation options and determine the desired configuration (e.g., silent install, etc.)

Before You Begin Steps

- Planning the WebSphere Application Server product installation (Disk Space, Supported OS, Language)
- Preparing the operating system for product installation
- Installing the product and additional software
- Installing maintenance packages, interim fixes, fix packs, and refresh packs
- Configuring the product after installation

WebSphere Application Server can interoperate with your other e-business systems, including other versions of WebSphere Application Server. Interoperability provides a communication mechanism for WebSphere Application Server nodes that are at different versions, running on separate machines. Coexistence describes multiple versions or instances running on the same machine at the same time.

Interoperability support enhances migration scenarios with more configuration options. Interoperating is often more convenient or practical during the migration of a configuration from an earlier WebSphere Application Server version to a later one. Some machines can have the earlier product version and other machines can have the later version. An environment of machines and application components at different software version levels can involve both interoperability and coexistence.

Coexistence

Coexistence is a state in which nodes from different versions of WebSphere Application Server can start and run in the same environment at the same time. Coexistence, as it applies to WebSphere Application Server products, is the ability of multiple installations of WebSphere Application Server to run on the same machine at the same time. Multiple installations include multiple versions and multiple instances of one version. Coexistence also implies various combinations of Web server interaction.

It is often impractical, or even physically impossible, to migrate all of the machines and applications within an enterprise at the same time. Understanding multiversion interoperability and coexistence is therefore an essential part of a migration between version levels. See the migration documentation for more information.

Workload Partition (WPAR): If you are going to install the application server product on a WPAR on AIX 6.1, then you must make sure that the WPAR has private and writable versions of the /usr and /opt file systems. If you do not have this type of WPAR, then create a new WPAR

A major enhancement to non-root support for the application server is the ability to verify and set file permissions. Users can now verify adequate file permissions before installing the product, and can use a utility to change ownership to another user for the file system after installation for future operations on that product.

IBM® HTTP Server and the Web server plug-ins installers now install a private copy of IBM Global Security Kit (GSKit) which allows both root and non-root users to enable SSL support.

Install WebSphere Application Server Network Deployment from the command line

```
install -options /tmp/was/my_response_file.txt -silent
```

Contents of /tmp/was/my_response_file.txt

```
-OPT allowNonRootSilentInstall="false"
-OPT feature="samplesSelected"
-OPT installLocation="app_server_root"
-OPT installType="installNew"
-OPT silentInstallLicenseAcceptance="true"
-OPT traceFormat=ALL
-OPT traceLevel=INFO
```

Create a cell profile from the command line

```
install -options /tmp/was/my_response_file2.txt -silent
```

Contents of /tmp/was/my_response_file2.txt

```
-OPT allowNonRootSilentInstall="false"
-OPT installLocation="app_server_root"
-OPT installType="installNew"
-OPT profileType="cell"
-OPT silentInstallLicenseAcceptance="true"
-OPT traceFormat=ALL
-OPT traceLevel=INFO
-OPT PROF_appServerNodeName="cellappnode01"
-OPT PROF_appServerProfileName="cellapp01"
-OPT PROF_cellName="dmgrcell01"
-OPT PROF_defaultPorts=
-OPT PROF_dmgrProfileName="dmgr01"
-OPT PROF_enableAdminSecurity="false"
-OPT PROF_hostName="5.55.555.555"
-OPT PROF_isDefault=
-OPT PROF_nodeName="dmgrnode"
-OPT PROF_profilePath="profile_root"
-OPT PROF_serverName="server01"
-OPT PROF_validatePorts=
-OPT PROF_webServerCheck="true"
-OPT PROF_webServerHostname="5.55.555.555"
-OPT PROF_webServerInstallPath="Web_server_root"
-OPT PROF_webServerName="IHS01"
-OPT PROF_webServerOS="linux"
-OPT PROF_webServerPluginPath="plugins_root"
-OPT PROF_webServerPort="80"
-OPT PROF_webServerType="IHS"
```

B) Install WebSphere Application Server Network Deployment V7.0 and verify the installation (e.g., Installation Verification Tool (IVT), default application (i.e., snoop and/or hitcount.))

Windows - The installation procedure requires the installer ID to have the following advanced user rights, Act as part of the operating system and Log on as a service to install Windows services.

If the Installation wizard detects a previous installation, the product-detection panel is displayed.

Start the Snoop servlet to verify the ability of the Web server to retrieve an application from the Application Server. Test your environment by starting your Application Server, your Web server, and using the snoop servlet with an IP address.

To start apache ${IHSROOT}/bin/apachectl start

Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://Host_name_of_Web_server_machine/snoop to test the Web server plug-in.

The CIM capability is automatically installed with the Network Deployment product.

Verify the success of the installer program by examining the completion panel and the app_server_root/logs/install/log.txt file to verify that there were no file system or other unusual errors while installing. If there are problems, correct them, and reinstall the product. Important information about the profile you created is also available in profile_root/logs/AboutThisProfile.txt.

Use the First steps console to verify the installation, start the Profile Management Tool, start the application server, or open the administrative console, where you can deploy sample applications.

The installer program records the following indicators of success in the logs:

* INSTCONFSUCCESS - The operation was a success.

* INSTCONFPARTIALSUCCESS - The operation was partially successful. Refer to the log for more details.

* INSTCONFFAILED - The operation failed. Refer to the log for more details.

After installing the product or after installing maintenance packages, you can use the installation verification utility (IVU) to compute checksums of the installed file set to verify the checksum against the checksum in the product bill of materials. Use the installver command to compute a checksum on the installed files and compare the checksum to the product bill of materials.

The IVU tool is installed during the installation of the following product component

* WebSphere® Application Server Network Deployment

* Application Client

* IBM® HTTP Server

* Web server plug-ins

* Update Installer for WebSphere Application Server

You can also use the IVU to compute a new checksum for a system after you make significant configuration changes. The installver tool computes a new baseline checksum for each file in the inventory of a configured system to use to identify file changes in the later comparisons. Such a comparison is useful for detecting file tampering on the configured system, for example.

The level tag is an early indicator of event status:

- INFO - Indicates a normal event.
- WARNING - Indicates an event that occurred with errors that do not prevent the creation of the profile.
- ERROR - Indicates an event that prevents the creation of the profile.

Cell - Create a cell with two profiles: a deployment manager and an application server node that is already federated into the deployment manager cell. This is useful for development environments.

Management - Create a management profile that provides the servers and services necessary to manage your WebSphere environment. You can select one of the following management profile types on the following panel:

* Deployment manager - The basic function of the deployment manager is to deploy applications to a cell of application servers, which it manages. Each application server t hat belongs to the cell is a managed node.

* Job manager - The basic function of the job manager is to provides a single console to administer multiple base servers, multiple deployment managers, and do asynchronous job submission.

* Administrative agent - The basic function of the administrative agent is to provide a single interface to administer multiple unfederated application servers.

Application server - Create a standalone application server profile.

Custom - Create a custom profile which belongs to a deployment manager cell, to make applications available to the Internet or to an intranet under the management of the deployment manager. You must federate this node to use it.

Secure proxy - Create a secure proxy server to take requests from the internet and forward them to application servers. The secure proxy server resides in the DMZ.

None - Do not create a profile during installation. However, if you do not create a profile during installation, then you must create a profile after installation to have an operational product.

You can also start the ivt command directly from the bin directory of the profile: ${profile_root}/bin/ivt.sh

The log file for installation verification is the ${profile_root}/logs/ivtClient.log.

The IVT provides the following useful information about the application server:

* The application server name

* The name of the profile

* The profile file path

* The type of profile

* The node name

* The current encoding

* The port number for the administrative console

* Various informational messages that include the location of the SystemOut.log file and how many errors are listed within the file

* A completion message

The installver tool validates the integrity of all installed files.

The IVT tool starts the server process of a profile automatically if the server is not running. Once the server initializes, the IVT runs a series of verification tests. The tool displays pass or fail status in a console window. The tool also logs results to the profile_root/logs/ivtClient.log file. As the IVT verifies your system, the tool reports any detectable errors in the SystemOut.log file.

The following examples test the server1 process in the profile01 profile on the myhost machine using the default_host on port 9081

```
ivt.sh server1 profile01 -p 9081 -host myhost
```

The ivt command logs results to the profile_root/logs/ivtClient.log file.

C) Create profiles.

You can create profiles, which define runtime environments, using the Profile Management Tool.

Using profiles instead of multiple product installations saves disk space and simplifies updating the product because a single set of core product files is maintained.

The Profile Management Tool is the graphical user interface for the manageprofiles command. See the description of the manageprofiles command for more information.

If you define coexisting nodes on the same computer with unique IP addresses, then define each IP address in a domain name server (DNS) look-up table. Configuration files for standalone application servers do not provide domain name resolution for multiple IP addresses on a machine with a single network address.

If you create the certificates, you can use the default values or modify them to create new certificates. The default personal certificate is valid for one year by default and is signed by the root signing certificate. The root signing certificate is a self-signed certificate that is valid for 15 years by default. The default keystore password for the root signing certificate is WebAS.

The keystore types that are supported depend on the providers in the java.security file.

When you create either or both certificates, or import either or both certificates, the keystore files that are created are key.p12, trust.p12, root-key.p12, default-signers.p12, deleted.p12, and ltpa.jceks. These files all have the same password when you create or import the certificates, which is either the default password, or a password that you specify. The key.p12 file contains the default personal certificate. The trust.p12 file contains the signer certificate from the default root certificate. The root-key.p12 file contains the root signing certificate. The default-signer.p12 file contains signer certificates that are added to any new keystore file that you create after the server is installed and running.

The ltpa.jceks file contains server default Lightweight Third-Party Authentication (LTPA) keys that the servers in your environment use to communicate with each other.

Refer to the description of the manageprofiles command to learn about creating a profile using a command instead of the Profile Management Tool.

manageprofiles command

Use the manageprofiles command to create, delete, augment, back up, and restore profiles, which define runtime environments. Using profiles instead of multiple product installations saves disk space and simplifies updating the product because a single set of core product files is maintained.

The manageprofiles command and its graphical user interface, the Profile Management tool, are the only ways to create runtime environments.

The command file is located in the app_server_root/bin directory. The command file is a script named manageprofiles.

Syntax

The manageprofiles command is used to perform the following tasks:

* create a profile (-create)

* delete a profile (-delete)

* augment a profile (-augment)

* unaugment a profile (-unaugment)

* unaugment all profiles that have been augmented with a specific augmentation template (-unaugmentAll)

* delete all profiles (-deleteAll)

* list all profiles (-listProfiles)

* list augments for a profile (-listAugments)

* get a profile name (-getName)

* get a profile path (-getPath)

* validate a profile registry (-validateRegistry)

* validate and update a profile registry (-validateAndUpdateRegistry)

* get the default profile name (-getDefaultName)

* set the default profile name (-setDefaultName)

* back up a profile (-backupProfile)

* restore a profile (-restoreProfile)

* perform manageprofiles command tasks that are contained in a response file (-response)

Example

```
app_server_root/bin/manageprofiles.sh -create \
-profileName Dmgr001 \
-profilePath profile_root \
-templatePath app_server_root/profileTemplates/management \
-serverType DEPLOYMENT_MANAGER \
-nodeName Dmgr001Node \
-cellName Dmgr001NodeCell \
-hostName localhost \
-isDefault \
-startingPort 20000 \
```

The manageprofiles command creates a log for every profile that it creates.

* The logs are in the app_server_root/logs/manageprofiles directory. The files are named in this pattern: profile_name_create.log.

* The command also creates a log for every profile that it deletes. The logs are in the app_server_root/logs/manageprofiles directory. The files are named in this pattern: profile_name_delete.log.

D) Troubleshoot the installation (e.g., identify and analyze log files.)

PrereqChecker errors do not prevent you from installing. WebSphere Application Server prevents users from installing to a non-empty directory.

Verify the success of the installer program by examining the completion panel and the app_server_root/logs/install/log.txt file to verify that there were no file system or other unusual errors while installing. If there are problems, correct them, and reinstall the product. Important information about the profile you created is also available in profile_root/logs/AboutThisProfile.txt.

Use the First steps console to verify the installation, start the Profile Management Tool, start the application server, or open the administrative console, where you can deploy sample applications.

The installer program records the following indicators of success in the logs:

* INSTCONFSUCCESS - The operation was a success.

* INSTCONFPARTIALSUCCESS - The operation was partially successful. Refer to the log for more details.

* INSTCONFFAILED - The operation failed. Refer to the log for more details.

AboutThisProfile.txt General information about the profile

activty.log - Compiled activity log from various installation activities

amjrte_config.log - Tivoli® Access Manager configuration log for its Java™ Runtime Environment

createDefaultServer.log - A log from wsadmin recording the creation of the server1 process in the default profile

createshortcutforprofile.log - Windows tool log for creating menu entries and shortcuts

defaultapp_config.log - JACL script log from configuring default application resources

defaultapp_deploy.log - Application DefaultApplication installation log

node_name Service.log - Start and stop events for server1

filetransfer_config.log - Application filetransfer installation log

hamanager_config.log - Configuration log for the high availability application

ivt_config.log - Application ivtApp installation log

mejb_config.log - Application ManagementEJB installation log

query_config.log - Application Query installation log

samples_config.log - Configuration log for the PlantsByWebSphere Samples application

samples_install.log - Installation log for the SamplesGallery and PlantsByWebSphere Samples applications

scheduler.cal_config.log - Application SchedulerCalendars installation log

SIBDefineChains.log - Creation log for service integration bus endpoints, inbound channels and channel chains, outbound thread pool, and outbound channel and channel chains

SIBDeployRA.log - Deployment log for the service integration bus function

webui_config.log - Application administrative console installation log

winservice_config.log - Service log for the Windows service created for server1

The app_server_root/logs/install/log.txt file and the app_server_root/logs/manageprofiles/profile_name_create.log file record installation and profile creation status.

If the error happens early in the installation, look for the log.txt file in the system temporary area.

app_server_root /logs/install/log.txt

0	Success
1	Failure
2	Partial Success

user_data_root/profileRegistry/logs/manageprofiles/create.log

* Traces all events that occur during the creation of the named profile

* Created when using the Profile Management Tool or the manageprofiles command

INSTCONFFAILED - Total profile creation failure.

INSTCONFSUCCESS - Successful profile creation.

INSTCONFPARTIALSUCCESS - Profile creation errors occurred but the profile is still functional. Additional information identifies the errors.

IP address caching and WebSphere Application Server process discovery

If you change the IP address of a federated WebSphere Application Server node, processes running in other nodes cannot contact the changed node until you stop and restart them.

If a deployment manager process starts on a disconnected node, it cannot communicate with cell member processes until you stop and restart the deployment manager process. For example, plugging in an unplugged network cable does not restore proper addresses in the IP cache until the deployment manager process is restarted.

<u>vpd.properties file</u>

The installer program for WebSphere® Application Server Version 7.0 uses the Install Shield for Multiplatforms (ISMP) program to install the product. The vpd.properties file lists application server program components, and products which extend the application server, that are currently installed on the system.

Note: There may be some future WebSphere installation programs that do not use ISMP and subsequently would not create or update the vpd.properties file. The information in the vpd.properties file is specifically there to assist you in troubleshooting problems, and it is not intended to be used or relied on as a supported way to determine which WebSphere products are installed.

Location of the vpd.properties file

The location of the vpd.properties file varies per operating platform:

* [AIX] The root directory or the /usr/lib/objrepos directory

* [HP-UX] [Solaris] The user_home directory when installed as a non-root user.

* [Linux] The root or user_home directories.

* [Windows] Installation directory of the operating system, such as the C:\WINNT directory or the C:\windows directory, or the user_home directory.

[HP-UX] [Solaris] ISMP creates or updates the vpd.properties file on all platforms except Solaris and HP-UX. ISMP uses native operating system registration on these platforms when installing as root, and does not create a vpd.properties file.

Non-root installation

When installing as a non-root installer, the installer programs create a vpd.properties file on all platforms, including Solaris and HP-UX.

<u>IP address caching and WebSphere Application Server process discovery</u>

If you change the IP address of a federated WebSphere Application Server node, processes running in other nodes cannot contact the changed node until you stop and restart them.

If a deployment manager process starts on a disconnected node, it cannot communicate with cell member processes until you stop and restart the deployment manager process. For example, plugging in an unplugged network cable does not restore proper addresses in the IP cache until the deployment manager process is restarted.

<u>Using the IP address cache setting</u>

You can always stop and restart a deployment manager process to refresh its IP address cache. However, this process might be expensive or inappropriate.

The networkaddress.cache.ttl (public, JDK1.4) and sun.net.inetaddr.ttl (private, JDK1.3) parameters control IP caching. The value is an integer that specifies the number of seconds to cache IP addresses. The default value, -1, specifies to cache forever. A value of zero (0) is a specification to never cache.

Using a zero (0) value is not recommended for normal operation. If you do not anticipate network outages or changes in IP addresses, use the cache forever setting. The never caching setting introduces the potential for DNS spoofing attacks.

Errors while installing the product

If the WebSphere® Application Server installation program indicates that errors were encountered while installing the product:

> * Browse the log files in the app_server_root/logs directory and in the app_server_root/profiles/profile_name/logs directory for clues. Pay particular attention to the main installation log file, log.txt.
>
> * Check the command prompt from which the installation panel was launched for error messages.
>
> * Look up any error or warning messages in the message reference table by selecting the "Reference" view in the information center navigation and expanding the "Messages" heading.

E) Utilize installation factory and Centralized Installation Manager (CIM).

The IBM® WebSphere® Installation Factory is an Eclipse-based tool which creates installation packages for installing WebSphere Application Server in a reliable and repeatable way, tailored to your specific needs.

The product can produce two kinds of packages, a customized installation package (CIP) and an integrated installation package (IIP). A CIP is used to install a WebSphere Application Server product in addition to maintenance, profile customization, and user-defined scripting. An IIP is a larger package which can install an entire WebSphere software stack, such as an application server, a feature pack, and other user files. An IIP can even contain several CIPs.

Customized installation packages

The Installation Factory combines the installation image for a version or release of a WebSphere software product with applicable maintenance packages, a configuration archive, one or more enterprise archive files, customization scripts, and other user files to create a customized installation package. You can even create a CIP containing a WebSphere feature pack and any associated feature pack fixes. You can significantly reduce the time and complexity of a standard installation process by creating and installing a CIP

CIP build definition file - A build definition file is an XML file that identifies components and characteristics for a customized installation package (CIP).

Integrated installation packages

Customers who need to install multiple installation packages in an automated and highly repeatable manner can create an IIP which aggregates those packages into a single installable package. As an example, you can have multiple servers on which you need to deploy WebSphere Application Server and some number of feature packs. Instead of having to install each of these products as an independent step on each server, you can create an IIP that will install all of them at once.

You specify which installation packages to include in the IIP, the order in which they should be installed, and various other details about the desired behavior of the IIP and each of its contained installation packages.

Each product you include in the IIP can be customized separately for greater flexibility. For example, you could run the WebSphere Application Server product install interactively and then run one or more feature pack installs silently to obtain a seamless install of the entire set of packages. There is also flexibility as to which contained installation packages actually get installed on any given invocation of the IIP; in other words you can choose not to install certain packages in the IIP.

To set up Installation Factory, you can either use the product code copied from the WebSphere Application Server product Supplements disc or use the code downloaded from the IBM WebSphere Installation Factory Web site. The advantage to downloading the code from the Web site is that you will be downloading the latest version of the product.

The ifcli command accepts a build definition XML file as input and invokes the IBM® WebSphere® Installation Factory processing engine. The processing engine interprets the XML file, locates the product source files and maintenance packages, and then creates a customized installation package (CIP) or integrated installation package (IIP). The command runs on any supported operating system.

Create an installation package:

```
./ifcli.sh -buildDef build_definition_file \
  -silent \
  -logLevel log_level \
  -logFile fully_qualified_log_file_path_name \
  -traceLevel trace level \
  -traceFile fully_qualified_trace_file_path_name \
  -wasPath app_server_root \
  -repositoryPath cim_repository_path \
  -installationPackagePath installation_package_path \
  -overwrite
```

Centralized installation manager

The CIM capability is automatically installed with the Network Deployment product.

In V7.0, the Network Deployment packaging adds the capability to perform centralized installations from the deployment manager to remote nodes through its centralized installation manager component.

Centralized installation manager enables a single installation to be pushed out as an installation package from the deployment manager to a series of endpoints. The endpoint can either be a node that is not part of a Network Deployment cell or an existing Network Deployment node that might need a fix pack.

With centralized installation manager, an administrator can remotely install or uninstall product components or maintenance to specific nodes directly from the Integrated Solutions Console without having to log in and repetitively complete these tasks for each node. Using centralized installation manager is a good way to shorten the number of steps that are required to create and manage your environment, and for an easier installation and patch management.

Installable packages and products

The centralized installation manager does not replace the Installation wizard or the IBM® Update Installer for WebSphere® Software. Instead, the centralized installation manager starts the Installation wizard for the product component or the Update Installer to install or uninstall the components or maintenance.

The various product components and maintenance files that you can install or uninstall by using the centralized installation manager are included in the following list:

* WebSphere Application Server Network Deployment Version 7.0
* WebSphere Application Server Version 7.0 refresh packs, fix packs, and interim fixes
* WebSphere Application Server Version 6.1 refresh packs, fix packs, interim fixes, and feature pack update
* Update Installer for WebSphere Application Server Version 7.0

<u>Update Installer for WebSphere Software</u>

The centralized installation manager installs an appropriate level of the Update Installer on the target systems that it uses to install fix packs and other maintenance. If you had previously installed the Update Installer tool on any of the target hosts in a directory location other than app_server_root/UpdateInstaller, then you may want to consider uninstalling the Update Installer by using its uninstallation process because that copy would not be used by the centralized installation manager. But it is not mandatory to uninstall that copy for CIM to work properly.

The centralized installation manager will automatically install the Update Installer tool (if it is not already installed in app_server_root/UpdateInstaller) when you install fix packs or other maintenance on the target systems. If the version of the Update Installer tool found in app_server_root/UpdateInstaller does not meet the minimum version required by the interim fix or fix pack, the centralized installation manager automatically installs a newer version on the targets, if you have downloaded the newer version of the Update Installer tool to your repository.

The CIM is capable of creating nodes on remote hosts by installing WebSphere Application Server Network Deployment and federating them to the existing deployment manager.

Prior to CIM, you had to log in to every machine in the potential cell, install the servers manually, create a profile for each node, and federate the nodes to the deployment manager. Now, these steps are all done for you by the CIM. You only select the machine host name, and provide the login credentials.

Procedure

1. On the deployment manager machine, install WebSphere Application Server Network Deployment with management profile and deployment manager server type.

    1. Launch the WebSphere Application Server Network Deployment Installation Wizard.
    2. From the Welcome Panel, click Next.
    3. From the License Panel, read and agree to the terms of the license, click Next.
    4. From the System Prerequisite Check panel, click Next.
    5. From the Optional Feature Installation panel, select all features, click Next.
    6. From the Installation Directory panel, select a writable folder as the WebSphere installation root, click Next.
    7. From the Environment Selection panel, select Management, click Next.
    8. From the Server Type Panel, select Deployment Manager, click Next.
    9. From the Administrative Security panel, specify a user name and password for logging in to the administrative console. This need not be the same user name and password for logging in to the deployment manager host. Click Next.

    10. From the Repository for centralized installation manager panel, select the option to create a repository and specify the repository location. Select to populate the repository with the installation package. Click Next.

    11. From the Installation Summary panel, click Next.

    12. After the installation is done, click Next.

2. Start the deployment manager. This can be done from the command line. From app_server_root/profiles/Dmgr01/bin, enter the following command:

3. Log in to the administrative console.

4. Add other platform images for WebSphere Application Server Network Deployment to the CIM repository. Refer to "Using the Installation Factory to add installation packages" for more details.

5. Launch installations of WebSphere Application Server Network Deployment on the remote machines. Refer to "Installing packages" for more details.

6. You can monitor the status of the installations using CIM. Refer to "Monitoring requests" for more details.

Results

The installation requests are sent via the centralized installation manager to install WebSphere application servers on the remote machines to create the cell.

Complete the following steps to update all the nodes within the cell to the new maintenance level. You do not need to access the managed nodes directly while using CIM. With the node agent running on the targets, CIM will be able to stop all the running servers on the target node, update the remote node, and then restart the node agent.

Procedure

1. Update the deployment manager node to the new maintenance level.

    1. Stop the deployment manager server.

    2. Update the deployment manager node to the maintenance level needed using the Update Installer tool.

    3. Restart the deployment manager server.

2. Log in to the administrative console.

3. Download the fix pack binary files and Update Installer tool for the platforms that you need into the centralized installation manager repository. You need the fix packs and Update Installers for all the platforms in the cell. Refer to the "Downloading the IBM Update Installer for WebSphere Software" and "Downloading package descriptors and the associated binary files to the repository" topics for more information.

4. Using the administrative console, install the new fix pack on all the nodes. You do not need to install the Update Installer tool directly on each node. CIM installs UPDI automatically if needed. Refer to the "Installing refresh packs or fix packs" topic for more details on this step.

5. Monitor the installation requests of the maintenance packages. Refer to the "Monitoring requests" topic for more details on this step.

Results

The installation requests are sent via the centralized installation manager to install WebSphere application servers on the remote machines to create the cell.

Section 3 - Application Assembly/Deployment and Cell Configuration/Resource Allocation (21%)

A) Describe the name service management of WebSphere Application Server Network Deployment V7.0 (JNDI).

Using naming

Naming is used by clients of WebSphere® Application Server applications most commonly to obtain references to objects related to those applications, such as Enterprise JavaBeans™ (EJB) homes. The Naming service is based on the Java™ Naming and Directory Interface (JNDI) Specification and the Object Management Group (OMG) Interoperable Naming (CosNaming) specifications Naming Service Specification, Interoperable Naming Service revised chapters and Common Object Request Broker: Architecture and Specification (CORBA).

Procedure

1. Develop your application using either JNDI or CORBA CosNaming interfaces. Use these interfaces to look up server application objects that are bound into the namespace and obtain references to them. Most Java developers use the JNDI interface. However, the CORBA CosNaming interface is also available for performing Naming operations on WebSphere Application Server name servers or other CosNaming name servers.

2. Assemble your application using an assembly tool. Application assembly is a packaging and configuration step that is a prerequisite to application deployment. If the application you are assembling is a client to an application running in another process, you should qualify the jndiName values in the deployment descriptors for the objects related to the other application. Otherwise, you might need to override the names with qualified names during application deployment. If the objects have fixed qualified names configured for them, you should use them so that the jndiName values do not depend on the other application's location within the topology of the cell.

3. Optional: Verify that your application is assigned the appropriate security role if administrative security is enabled. For more information on the security roles, see Naming roles.

4. Deploy your application. Put your assembled application onto the application server. If the application you are assembling is a client to an application running in another server process, be sure to qualify the jndiName values for the other application's server objects if they are not already qualified. For more information on qualified names, refer to Lookup names support in deployment descriptors and thin clients.

5. Optional: If your application must access applications in other cells, configure foreign cell bindings for the other cells.

6. Configure namespace bindings. This step is necessary in these cases:

* Your deployed application is to be accessed by legacy client applications running on previous versions of the product. In this case, you must configure additional name bindings for application objects relative to the default initial context for legacy clients. (Version 5 clients have a different initial context from legacy clients.)

* The application requires qualified name bindings for such reasons as:

  It will be accessed by Java Platform, Enterprise Edition (Java EE) client applications or server applications running in another server process.

  It will be accessed by thin client applications.

In this case, you can configure name bindings as additional bindings for application objects. The qualified names for the configured bindings are fixed, meaning they do not contain elements of the cell topology that can change if the application is moved to another server. Objects as bound into the namespace by the system can always be qualified with a topology-based name. You must explicitly configure a name binding to use as a fixed qualified name.

For more information on qualified names, refer to Lookup names support in deployment descriptors and thin clients. For more information on configured name bindings, refer to Configured name bindings.

7. Troubleshoot any problems that develop. If a Naming operation is failing and you need to verify whether certain name bindings exist, use the dumpNameSpace tool to generate a dump of the namespace.

Running dumpNameSpace

You can run the tool from a command line or using its program interface. This topic describes command-line invocations. To access the dumpNameSpace tool through its program interface, refer to the class com.ibm.websphere.naming.DumpNameSpace in the WebSphere® Application Server API documentation.

If you run the dumpNameSpace tool with security enabled and the com.ibm.CORBA.loginSource property is set in the profile_root/properties/sas.client.props file, a login prompt is displayed.

- dumpNameSpace -host myhost.mycompany.com -port 901
- dumpNameSpace -url corbaloc:iiop:myhost.mycompany.com:901

If the object a client needs to access does not appear, use the administrative console to verify that:

* The server hosting the target resource is started.

* The Web module or EJB container, if applicable, hosting the target resource is running.

* The Java Naming and Directory Interface (JNDI) name of the target resource is correct and updated.

* If the problem resource is remote, that is, not on the same node as the Name Server node, that the JNDI name is fully qualified, including the host name.

Connection factory JNDI name practices

Observe the conventions of the Java™ Naming and Directory Interface (JNDI) service in WebSphere® Application Server when you create connection factory JNDI names.

Distributed computing environments often employ naming and directory services to obtain shared components and resources. Naming and directory services use name-to-object mappings to associate names with objects such locations, services, information, and resources. The Java Naming and Directory Interface (JNDI) provides a common interface that is used to access the various naming and directory services.

Naming your resources indirectly

When creating a connection factory or data source, a JNDI name is given by which the connection factory or data source can be looked up by a component. WebSphere Application Server uses an indirect name with the java:comp/env prefix:

* When you create a WebSphere Application Server data source, the default JNDI

name is set to jdbc/data_source_name.

* When you create a connection factory, its default name is eis/j2c_connection_factory_name.

If you override these values by specifying your own, retain the java:comp/env prefix. An indirect name makes any resource-reference data associated with the application available to the connection management runtime, to better manage resources based on the res-auth, res-isolation-level, res-sharing-scope, and res-resolution-control settings.

Naming your resources for use with CMP

In addition, if you click the checkbox for the Use this data source for container managed persistence (CMP) option when you create the data source, another reference is created with the name of eis/jndi_name_of_datasource_CMP. For example, if a data source has a JNDI name of jdbc/myDatasource, the CMP JNDI name is eis/jdbc/myDatasource_CMP. This name is used internally by CMP and is provided simply for informational purposes.

Naming service authorization

CosNaming security offers increased granularity of security control over CosNaming functions. CosNaming functions are available on CosNaming servers such as the WebSphere Application Server. These functions affect the content of the WebSphere Application Server name space. Generally, you have two ways in which client programs result in CosNaming calls. The first is through the Java Naming and Directory Interface (JNDI) call. The second is with common object request broker architecture (CORBA) clients invoking CosNaming methods directly.

Four security roles are introduced :

* CosNamingRead
* CosNamingWrite
* CosNamingCreate
* CosNamingDelete

The roles have authority levels from low to high:

CosNamingRead

You can query the WebSphere Application Server name space, using, for example, the JNDI lookup method. The special-subject, Everyone, is the default policy for this role.

CosNamingWrite

You can perform write operations such as JNDI bind, rebind, or unbind, and CosNamingRead operations. As a default policy, Subjects are not assigned this role.

CosNamingCreate

You can create new objects in the name space through such operations as JNDI createSubcontext and CosNamingWrite operations. As a default policy, Subjects are not assigned this role.

CosNamingDelete

You can destroy objects in the name space, for example using the JNDI destroySubcontext method and CosNamingCreate operations. As a default policy, Subjects are not assigned this role.

A Server special-subject is assigned to all of the four CosNaming roles by default. The Server special-subject provides a WebSphere Application Server process, which runs under the server identity, to access all the CosNaming operations. The Server special-subject does not display and cannot be modified through the administrative console or other administrative tools.

Special configuration is not required to enable the server identity as specified when enabling administrative security for administrative use because the server identity is automatically mapped to the administrator role.

Users, groups, or the special subjects AllAuthenticated and Everyone can be added or removed to or from the naming roles from the WebSphere Application Server administrative console at any time. However, a server restart is required for the changes to take effect.

A best practice is to map groups or one of the special-subjects, rather than specific users, to naming roles because it is more flexible and easier to administer in the long run. By mapping a group to a naming role, adding or removing users to or from the group occurs outside of WebSphere Application Server and does not require a server restart for the change to take effect.

The CosNaming authorization policy is only enforced when administrative security is enabled. When administrative security is enabled, attempts to do CosNaming operations without the proper role assignment result in an org.omg.CORBA.NO_PERMISSION exception from the CosNaming server.

Each CosNaming function is assigned to only one role. Therefore, users who are assigned the CosNamingCreate role cannot query the name space unless they have also been assigned CosNamingRead. And in most cases a creator needs to be assigned three roles: CosNamingRead, CosNamingWrite, and CosNamingCreate. The CosNamingRead and CosNamingWrite roles assignment for the creator example are included in the CosNamingCreate role. In most of the cases, WebSphere Application Server administrators do not have to change the roles assignment for every user or group when they move to this release from a previous one.

Although the ability exists to greatly restrict access to the name space by changing the default policy, unexpected org.omg.CORBA.NO_PERMISSION exceptions can occur at runtime. Typically, Java EE applications access the name space and the identity they use is that of the user that authenticated to WebSphere Application Server when accessing the Java EE application. Unless the Java EE application provider clearly communicates the expected naming roles, use caution when changing the default naming authorization policy.

B) Package Java enterprise applications, including enhanced ear files using the Rational Assembly and Deployment Tool.

Rational Assembly and Deployment Tool

Rational Application Developer for Assembly and Deploy provides a small subset of the development and test function available in Rational Application Developer for WebSphere .

Application assembly consists of creating Java™ Platform, Enterprise Edition (Java EE) modules that can be deployed onto application servers. The modules are created from code artifacts such as Web application archive (WAR) files, resource adapter archive (RAR) files, enterprise bean (EJB) JAR files, and application client archive (JAR) files. This packaging and configuring of code artifacts into enterprise archive (EAR) modules or stand-alone Web modules is necessary for deploying the modules onto an application server.

Application assembly is the process of creating an enterprise archive (EAR) file containing all files related to an application. This configuration and packaging prepares the application for deployment onto an application server.

EAR files are comprised of the following archives:

* Enterprise bean JAR files (known as EJB modules)

* Web archive (WAR) files (known as Web modules)

* Application client JAR files (known as client modules)

* Resource adapter archive (RAR) files (known as resource adapter modules)

* SAR files (known as Session Initiation Protocol (SIP) modules)

Ensure that modules are contained in an EAR file so that they can be deployed onto the server. The exceptions are WAR modules, which you can deploy individually. Although WAR modules can contain regular Java™ archive (JAR) files, they cannot contain the other module types described previously.

The assembly process includes the following actions:

* Selecting all of the files to include in the module.

* Creating an annotation or deployment descriptor containing instructions for module deployment on the application server.

You can use the graphical interface of Rational® Application Developer assembly tools to generate the annotation or deployment descriptor. You can also edit annotations or descriptors directly in your favorite XML editor.

* Packaging modules into a single EAR file, which contains one or more files in a compressed format.

As part of the assembly process, you might also set environment-specific binding information. These bindings are defaults for an administrator to use when installing the application through the administrative console. Further, you might define IBM® extensions to the Java Platform, Enterprise Edition (Java EE) specifications, such as to allow servlets to be served by class name. To ensure portability to other application servers, these extensions are saved in an XML file that is separate from the standard annotation or deployment descriptor.

wsadmin scripts - Invoke AdminApp object install commands in a script or at a command prompt.

Job manager runs wsadmin scripts - Invoke AdminTask.submitJob -jobType installApplication command in a script or at a command prompt.

For transitioning users: In the Version 6.1 Feature Pack for Web services and Feature Pack for EJB 3.0, the default is to scan pre-Java EE 5 Web application modules to identify JAX-WS services and to scan pre-Java EE 5 Web application modules and EJB modules for service clients during application installation. For Version 7.0, the default is not to scan pre-Java EE 5 modules for annotations during application installation or server startup. To preserve backward compatibility with either or both feature packs, you can define Java virtual machine custom properties on servers to request scanning during application installation and server startup.

* You can define these custom properties using the console. Click Servers > Server Types > WebSphere application servers > server name > Java and Process Management > Process definition > Java virtual machine > Custom properties. To request scanning for Feature Pack for Web services modules, set the com.ibm.websphere.webservices.UseWSFEP61ScanPolicy custom property to true. To request scanning for Feature Pack for EJB 3.0 modules, set the

com.ibm.websphere.ejb.UseEJB61FEPScanPolicy custom property to true. The default value for each of these custom properties is false. You must change the setting on each server that requires a change in the default.

You can specify values for these custom properties in the META-INF/MANIFEST.MF file of a module. Values specified in the META-INF/MANIFEST.MF file always take precedence over a server-level setting.

Enhanced EAR file

Exporting applications enables you to back up your applications and preserve binding information for the applications. You might export your applications before updating installed applications or migrating to a later version of the product.

To export applications, use the Export button on the Enterprise applications page. Using Export produces an enhanced enterprise archive (EAR) file that contains the application as well as the deployment configuration. The deployment configuration consists of the deployment.xml and other configuration files that control the application behavior on a deployment target.

You can edit your exported enhanced EAR file and then reinstall it. By default, installation expands an EAR file in the profile_root/installedApps/cell_name directory. If you specified the $(CELL) variable for Directory to install application on the Select installation options panel of the application installation wizard when you first installed the application, the cell_name directory is the current cell name.

To reinstall the enhanced EAR file, do either of the following:

* Use the Update operation available from the Enterprise applications page to upgrade the existing application installation.

The Update operation adds the application files to the profile_root/installedApps/cell_name directory, where cell_name is the current cell name or the name of the cell that you specified for Directory to install application when you first installed the application on a deployment target. The Directory to install application setting is on the Select installation options panel of the application installation wizard. If you specified the $(CELL) variable for Directory to install application when you first installed the application, the cell_name directory is the current cell name.

* Use the Applications > New application > New Enterprise Application operation to install the exported EAR file.

If you specified the $(CELL) variable for Directory to install application when you first installed the application, the cell_name directory is the current cell name. That is, if the file is originally installed on Cell1 with $(CELL) variable in the destination directory and you reinstall the enhanced EAR file on Cell2, the cell_name directory is Cell2, the current cell name.

If the $(CELL) variable was not specified for the first installation, using New Enterprise Application to reinstall an enhanced EAR file installs the application in the cell_name directory of the exported application. That is, if the application is originally installed on and exported from Cell1 and you reinstall the enhanced EAR file on Cell2, the cell_name directory is Cell1. The enhanced EAR file expands in the Cell1 directory even though the current cell name is Cell2. By default, the application destination directory contains Cell1 in its path because the deployment.xml file in the exported application has Cell1 in it.

If you exported the application from Cell1 and did not specify the $(CELL) variable when first installing the application, and you want to install the enhanced EAR file on a different cell, deselect Process embedded configuration on the Select installation options panel of the application installation wizard to expand the enhanced EAR file in the current cell name directory, which is not Cell1.

C) Define and map security roles (e.g., Java enterprise security).

You can use three types of Web login authentication mechanisms to configure a Web application: basic authentication, form-based authentication and client certificate-based authentication. Protect Web resources in a Web application by assigning security roles to those resources.

Configure the login mechanism for the Web module. This configured login mechanism applies to all the servlets, JavaServer Pages (JSP) files and HTML resources in the Web module.

1. Click the Pages tab of a Web deployment descriptor editor and click Login. Select the required authentication method. Available method values include: Unspecified, Basic, Digest, Form, and Client-Cert.
2. Specify a realm name.
3. If you select the Form authentication method, select a login page and an error page Web address. For example, you might use /login.jsp or /error.jsp. The specified login and error pages are present in the .war file.
4. Install the client certificate on the browser or Web client and place the client certificate in the server trust keyring file, if ClientCert is selected.

After securing a Web application, the resulting Web archive (WAR) file contains security information in its deployment descriptor. The Web module security information is stored in the web.xml file. When you work in the Web deployment descriptor editor, you also can edit other deployment descriptors in the Web project, including information on bindings and IBM® extensions in the ibm-web-bnd.xmi and ibm-web-ext.xmi files

After securing an EJB application, the resulting .jar file contains security information in its deployment descriptor. The security information of the EJB modules is stored in the ejb-jar.xml file. Deploying applications that have security constraints (secured applications) is not much different than deploying applications that do not contain any security constraints. The only difference is that you might need to assign users and groups to roles for a secured application. The secured application requires that you have the correct active user registry.

To deploy a newly secured application click Applications > Install New Application and follow the prompts to complete the installation steps. One of the required steps to deploy secured applications is to assign users and groups to roles that are defined in the application.

* If you are installing a secured application, roles will be defined in the application.

* If delegation is required in the application, you will be defining RunAs roles also.

After creating new roles and assigning them to enterprise bean and Web resources, add users and groups to roles with an assembly tool.

The ibm-application-bnd.xmi or ibm-application-bnd.xml file in the application contains the users and groups-to-roles mapping table, which is the authorization table. For Java™ EE Version 5 applications, the ibm-application-bnd.xml file contains the authorization table.

D) Configure application resources (e.g., JCA resource adapters, connection factories, resource scoping, MDB activation specification, JDBC providers, data sources)

Relational resource adapters and JCA

A resource adapter is a system-level software driver that a Java™ application uses to connect to an enterprise information system (EIS). A resource adapter plugs into an application server and provides connectivity between the EIS, the application server, and the enterprise application.

WebSphere® Application Server supports JCA versions 1.0 and 1.5 including additional, configurable features for JCA 1.5 resource adapters with activation specifications that handle inbound requests.

Data access for container-managed persistence (CMP) beans is indirectly managed by the WebSphere Persistence Manager. The JCA specification supports persistence manager delegation of the data access to the JCA resource adapter without knowing the specific backend store. For the relational database access, the persistence manager uses the relational resource adapter to access the data from the database.

WebSphere Relational Resource Adapter

WebSphere Application Server provides the WebSphere Relational Resource Adapter implementation. This resource adapter provides data access through JDBC calls to access the database dynamically. The connection management is based on the JCA connection management architecture and provides connection pooling, transaction, and security support. The WebSphere RRA is installed and runs as part of WebSphere Application Server, and needs no further administration.

The RRA supports both the configuration and use of JDBC data sources and Java EE Connection Architecture (JCA) connection factories. The RRA supports the configuration and use of data sources implemented as either JDBC data sources or Java EE Connector Architecture connection factories. Data sources can be used directly by applications, or they can be configured for use by container-managed persistence (CMP) entity beans.

For more information about the WebSphere Relational Resource Adapter, see the following topics:

* For information about resource adapters and data access, see the topic Data access portability features.

* For RRA settings, see the topic WebSphere relational resource adapter settings.

* For information about enterprise beans, see the topic EJB applications.

To access data from a Java™ EE Connector Architecture (JCA) compliant application in WebSphere® Application Server, you configure and use resource adapters and connection factories

A resource adapter is an implementation of the J2EE Connector Architecture (JCA) Specification that provides access for applications to resources outside of the server or provides access for an enterprise information system (EIS) to applications on the server. It can provide application access to resources such as DB2, CICS, SAP and PeopleSoft. It can provide an EIS with the ability to communicate with message-driven beans that are configured on the server. Some resource adapters are provided by IBM; however, third party vendors can provide their own resource adapters. A resource adapter implementation is provided in a resource adapter archive file; this file has an extension of .rar. A resource adapter can be provided as a standalone adapter or as part of an application, in which case it is referred to as an embedded adapter.

Procedure

1. Open the administrative console.
2. Select Resources > Resource adapters > resource_adapter.
3. Set the scope setting. This field specifies the level to which this resource definition is visible. For general information, see Administrative console scope settings in the Related Reference section. The Scope field is a read only string field that shows where the particular definition for a resource adapter is located. This is set either when the resource adapter is installed (which can only be at the node level) or when a new

resource adapter definition is added.

4. Configure the description. This field specifies a text description of the resource adapter. Use a free-form text string to describe the resource adapter and its purpose.
5. Set the archive path. Use this field to specify the path to the RAR file containing the module for this resource adapter. This property is required.
6. Set the classpath. The list of paths or JAR file names that together form the location for the resource adapter classes is set here. This includes any additional libraries needed by the resource adapter. The resource adapter code base itself is automatically added to the class path, but if anything outside the RAR is needed it can be specified here.
7. Set the native path. The list of paths which forms the location for the resource adapter native libraries is set here. The resource adapter code base itself is automatically added to the class path, but if anything outside the RAR is needed it can be specified here.
8. Set the ThreadPool alias. The name of a thread pool that is configured in the server that is used by the resource adapter's Work Manager is specified in this field. If there is no thread pool configured in the server with this name, the default configured thread pool instance, named Default, is used. This property is only necessary if this resource adapter uses Work Manager. This field does not apply for the z/OS platform.

When you use the Install RAR dialog to installing a RAR file, the scope you define on the Resource Adapters page has no effect on where the RAR file is installed. You can install RAR files only at the node level, which you specify on the Install RAR page. To set the scope of an RAR file to a specific cluster, or server, after you install the RAR file at each node level, create a new copy of the RAR file with the appropriate cluster or server scope.

<u>Connection Factories</u>

A JMS connection factory is used to create connections to the associated messaging provider of JMS destinations, for both point-to-point and publish/subscribe messaging. Use connection factory administrative objects to manage JMS connection factories for the default messaging provider, the WebSphere® MQ messaging provider or a third-party messaging provider.

Resources > JMS > Connection factories

Name

The display name of each connection factory instance.

JNDI name

The Java™ Naming and Directory Interface (JNDI) name of each connection factory instance.

Provider

The messaging provider that supports each connection factory instance. This is the default messaging provider (service integration), the WebSphere MQ messaging provider or a third-party messaging provider.

Description

An optional description of each connection factory instance.

Scope

The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

<u>J2C connection factories settings</u>

* Resources > Resource Adapters > J2C connection factories > J2C_connection_factory

* Resources > Resource Adapters > Resource adapters > resource_adapter > J2C connection factories > J2C_connection_factory

CMP connection factory settings

Use this page to view the settings of a connection factory that is used by a container-managed persistence (CMP) bean to access any database server. This connection factory is only in "read" mode. It cannot be modified or deleted.

To view this administrative console page:

1. Click Resources > Resource Adapters > Resource adapters.
2. Click Preferences, and select Show built-in resources.
3. Click WebSphere Relational Resource Adapter > CMP Connection Factories > connection_factory .

Queue connection factory collection

A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

In the administrative console, to view this page click Resources > JMS > Queue connection factories

Topic connection factory collection

A JMS topic connection factory is used to create connections to the associated messaging provider of JMS topic destinations, for publish and subscribe messaging.

In the administrative console, to view this page click Resources > JMS > Topic connection factories

Resource Scoping

You always create resources at the current scope that is selected in the administrative console panel, even though the resources might be visible at more than one scope.

Resources such as Java Database Connectivity (JDBC) providers, namespace bindings, or shared libraries can be defined at multiple scopes. Resources that are defined at more specific scopes override duplicate resources that are defined at more general scopes:

- The application scope has precedence over all the scopes.
- For Network Deployment, the server scope has precedence over the node, cell, and cluster scopes.
- For Network Deployment, the cluster scope has precedence over the node and cell scopes.
- The node scope has precedence over the cell scope.

Despite the scope of a defined resource, the resource properties apply at an individual server level only. For example, if you define the scope of a data source at the cell level, all the users in that cell can look up and use that data source, which is unique within that cell. However, resource property settings are local to each server in the cell. For example, if you define the maximum connections as 10, then each server in that cell can have 10 connections.

The cell scope is the most general scope and does not override any other scope. The recommendation is that you generally specify a more specific scope than the cell scope. When you define a resource at a more specific scope, you provide greater isolation for the resource. When you define a resource at a more general scope, you provide less isolation. Greater exposure to cross-application conflicts occur for a resource that you define at a more general scope.

You can configure the following resources for message-driven beans:

 * Activation specifications for message-driven beans that comply with Java™ EE Connector Architecture (JCA) Version 1.5.

 * The message listener service, listener ports, and listeners for any message-driven beans that you want to deploy against listener ports.

For WebSphere Application Server Version 7 and later, listener ports are deprecated. Therefore you should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications.

WebSphere Application Server Version 7 continues to support the same options for message-driven bean deployment that WebSphere Application Server Version 6 supports, and adds a new option for WebSphere MQ message-driven beans. This gives you the following options for deploying message-driven beans on WebSphere Application Server Version 7:

 * You must deploy default messaging (service integration bus) message-driven beans to use activation specifications.

 * You can deploy new and existing WebSphere MQ message-driven beans to use listener ports (as on WebSphere Application Server Version 6) or to use activation specifications.

 * You can deploy third-party messaging message-driven beans to use either listener ports or activation specifications, depending on the facilities available from your third-party messaging provider.

Configure a JMS activation specification to enable a message-driven bean to communicate with the default messaging provider.

You create a JMS activation specification if you want to use a message-driven bean to communicate with the default messaging provider through Java™ EE Connector Architecture (JCA) 1.5. JCA provides Java connectivity between application servers such as WebSphere® Application Server, and enterprise information systems. It provides a standardized way of integrating JMS providers with Java EE application servers, and provides a framework for exchanging data with enterprise systems, where data is transferred in the form of messages.

One or more message-driven beans can share a single JMS activation specification.

Because a JMS activation specification is a group of messaging configuration properties not a component, it cannot be manually started and stopped. For this reason, to prevent a message-driven bean from processing messages you must complete the following tasks:

 * Stop the application that contains the message-driven bean.

 * Stop the messaging engine.

All the activation specification configuration properties apart from Name, JNDI name, Destination JNDI name, and Authentication alias are overridden by appropriately named activation-configuration properties in the deployment descriptor of an associated EJB 2.1 or later message-driven bean. For an EJB 2.0 message-driven bean, the Destination type, Subscription durability, Acknowledge mode and Message selector properties are overridden by the corresponding elements in the deployment descriptor. For either type of bean the Destination JNDI name property can be overridden by a value specified in the message-driven bean bindings.

Procedure

1. Start the administrative console.
2. Display the default messaging provider. In the navigation pane, expand Resources > JMS > JMS providers.
3. Select the default provider for which you want to configure an activation specification.
4. Optional: Change the Scope check box to the scope level at which the activation specification is to be visible to applications, according to your needs.
5. In the content pane, under the Additional properties heading, click Activation specifications. This lists any existing JMS activation specifications for the default messaging provider in the content pane.
6. Display the properties of the JMS activation specification. If you want to display an existing activation specification, click one of the names listed.

   Alternatively, if you want to create a new activation specification, click New, then specify the following required properties:

   Name

   Type the name by which the activation specification is known for administrative purposes.

   JNDI name

   Type the JNDI name that is used to bind the activation specification into the JNDI namespace.

   Destination type

   Whether the message-driven bean uses a queue or topic destination.

   Destination JNDI name

   Type the JNDI name that the message-driven bean uses to look up the JMS destination in the JNDI namespace.

   Select the type of destination on the Destination type property.

   Bus name

   The name of the bus to connect to.

   Specify the name of the service integration bus to which connections are made. This must be the name of the bus on which the bus destination identified by the Destination JNDI name property is defined.

   You can either select an existing bus or type the name of another bus. If you type the name of a bus that does not exist, you must create and configure that bus before the activation specification can be used.
7. Specify properties for the JMS activation specification, according to your needs.
8. Optional: Specify the JMS activation specification connection properties that influence how the default messaging provider chooses the messaging engine to which your message-driven bean application connects. By default, the environment automatically connects applications to an available messaging engine on the bus. However you can specify extra configuration details to influence the connection process; for example to identify special bootstrap servers, or to limit connection to a subgroup of available messaging engines, or to improve availability or

performance, or to ensure sequential processing of messages received. For information about why and how to do this, see How JMS applications connect to a messaging engine on a bus.

9. Click OK.
10. Save your changes to the master configuration.

Use the createSIBJMSActivationSpec command to create a new JMS activation specification for the default messaging provider at a specific scope.

Example

* Using Jython:

```
 wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'
wsadmin>AdminTask.createSIBJMSActivationSpec("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml)", ["-name", "myjmsas", "-jndiName", "jms/myjmsas",
 "-destinationJndiName", "jms/mqueue", "-busName", "abus"])
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)'
wsadmin>AdminTask.listSIBJMSActivationSpecs("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)'
```

JDBC providers

Installed applications use JDBC providers to interact with relational databases.

The JDBC provider object supplies the specific JDBC driver implementation class for access to a specific vendor database. To create a pool of connections to that database, you associate a data source with the JDBC provider. Together, the JDBC provider and the data source objects are functionally equivalent to the Java™ Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection factory, which provides connectivity with a non-relational database.

Procedure

1. Verify that all of the necessary JDBC driver files are installed on your application server. Consult the article, Data source minimum required settings, by vendor for that information. If you opt to configure a user-defined JDBC provider, check your database documentation for information about the driver files.
2. Create a JDBC provider.

From the administrative console, see the topic, Creating a JDBC provider using the administrative console.

OR

Using the wsadmin scripting client, see the topic, Configuring a JDBC provider using the scripting.

OR

Using the Java Management Extensions (JMX) API, see the topic, Creating a JDBC provider and data source using the JavaManagement Extensions API.

3. Create a data source.

From the administrative console, see the topic, Creating a data source using the administrative console.

OR

Using the wsadmin scripting client, see the topic, Configuring new data sources using scripting. For V4 data sources, see the topic, Configuring new WAS40 data sources using scripting.

OR

Using the JMX API, see the topic, Creating a JDBC provider and data source using the JavaManagement Extensions API.

Required properties: Different database vendors require different properties for implementations of their JDBC drivers. Set these properties on the WebSphere Application Server data source. Because Application Server contains templates for many vendor JDBC implementations, the administrative console surfaces the required properties and prompts you for them as you create a data source. However, if you script your data access configurations, you must consult the article Data source minimum required settings, by vendor, for the required properties and settings options.

4. Optional: Configure custom properties. Like the required properties, custom properties for specific vendor JDBC drivers must be set on the Application Server data source. Consult your database documentation for information about available custom properties. To configure a custom class to facilitate the handling of database properties that are not recognized natively by the Application Server, refer to the topic, Developing a custom DataStoreHelper class.

You can also learn about optional data source properties in the Application Programming Guide and Reference for Java for your version of DB2 for z/OS® if you use the DB2 Universal JDBC Driver provider.

5. Bind resource references to the data source. See the article, Data source lookups for enterprise beans and Web modules.

6. Test the connection (for non-container-managed persistence usage). See the topic, Test connection service.

Data sources

Installed applications use a data source to obtain connections to a relational database. A data source is analogous to the Java™ Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection factory, which provides connectivity to other types of enterprise information systems (EIS).

A data source is associated with a JDBC provider, which supplies the driver implementation classes that are required for JDBC connectivity with your specific vendor database. Application components transact directly with the data source to obtain connection instances to your database. The connection pool that corresponds to each data source provides connection management.

You can create multiple data sources with different settings, and associate them with the same JDBC provider. For example, you might use multiple data sources to access different databases within the same vendor database application. WebSphere® Application Server requires JDBC providers to implement one or both of the following data source interfaces, which are defined by Sun Microsystems. These interfaces enable the application to run in a single-phase or two-phase transaction protocol.

ConnectionPoolDataSource - a data source that supports application participation in local and global transactions, excepting two-phase commit transactions. When a connection pool data source is involved in a global transaction, transaction recovery is not provided by the transaction manager. The application is responsible for providing the backup recovery process if multiple resource managers are involved.

XADataSource - a data source that supports application participation in any single-phase or two-phase transaction environment. When this data source is involved in a global transaction, the product transaction manager provides transaction recovery.

These architectures are represented by two types of data sources. To choose the right data source, administrators must understand the nature of their applications, EJB modules, and enterprise beans.

- Data source (WebSphere Application Server V4) - This data source runs under the original CM architecture. Applications using this data source behave as if they were running in Version 4.0.
- Data source - This data source uses the JCA standard architecture to provide support for J2EE version 1.3 and 1.4, as well as Java EE applications. It runs under the JCA connection manager and the relational resource adapter.

E) Configure WebSphere messaging (e.g., SIBus bus members and destinations).

Asynchronous messaging

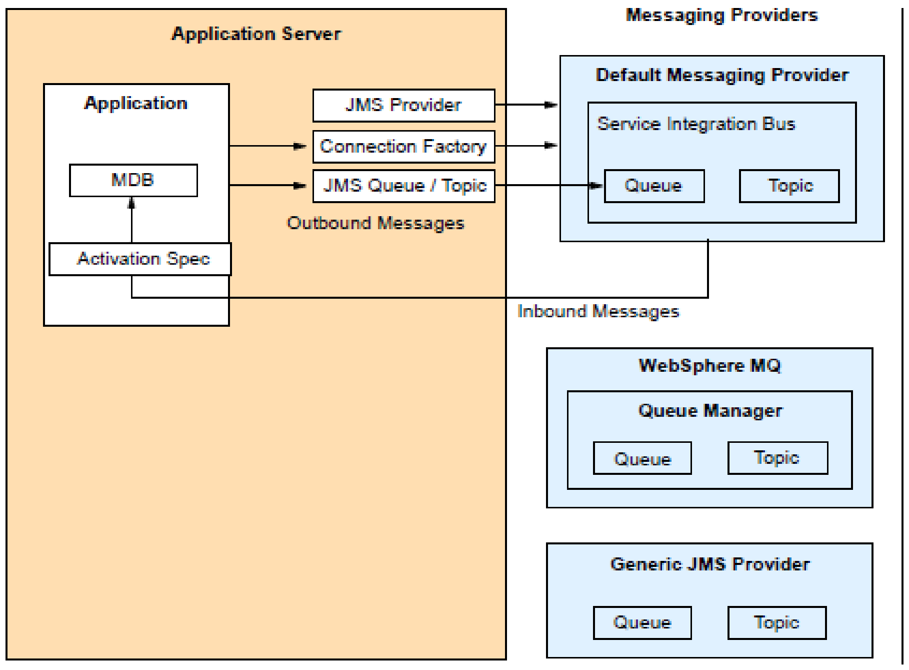
Asynchronous messaging involves loosely coupled processes, where the sending and receiving applications communicate through a messaging provider. The sending application is able to pass the data to the messaging provider and then continue with its processing. The receiving application is able to connect to the messaging provider, possibly at some later point in time, to retrieve the data.

WebSphere Application Server supports asynchronous messaging through the use of the Java Message Service (JMS). The JMS API is the standard Java API for accessing enterprise messaging systems from Java programs. In other words, it is a standard API that sending and receiving applications written in Java can use to access a messaging provider to create, send, receive, and read messages.

The JMS API was first included in Version 1.2 of the Java EE specification. This specification required that the JMS API definitions be included in a Java EE product, but that the platform was not required to include an implementation of the JMS ConnectionFactory and Destination objects. Subsequent versions of the Java EE specification have placed further requirements on application server vendors. WebSphere Application Server V7 is fully compliant with the Java EE 5 specification.

Messaging applications require runtime resources in order to deliver messages. These resources consist of the messaging provider implementation that holds the messages on queue and topic destinations for delivery and the JMS configuration objects that the application uses to access the queue and topic destinations.

WebSphere Application Server provides a default messaging provider that uses the service integration bus as the messaging system. In addition, WebSphere supports WebSphere MQ as a messaging provider and third-party messaging providers that implement the ASF component of the JMS 1.0.2 specification. WebSphere supports JCA 1.5 compliant messaging providers through the installation of resource adapters that allow applications to connect to third-party provided external providers.



The JMS configuration objects used by the application to connect to a provider and to access queues and topics on the provider are specific to the JMS provider.

In order for a JMS provider to be used by a messaging application, the following items must be configured using the WebSphere administrative tools:

<u>A JMS provider</u>

A JMS provider is configured in WebSphere to manage resources specific to a messaging provider implementation. The JMS administered objects required to connect to the provider and the destinations (queues or topics) on the provider are associated with the JMS provider definition.

<u>A JMS connection factory</u>

The connection factory is used by an application to connect to the JMS provider. The connection factory configuration includes the JNDI name that binds it to the WebSphere name space and the information required to connect to the JMS provider.

For example, a connection factory for a WebSphere MQ provider would include the queue manager name and the information required to connect to that queue manager. A connection factory for the default messaging provider would include the bus name.

<u>JMS queues and JMS topics</u>

These resources define the destination for messages sent to the provider. Applications attach to these resources as producers, consumers, or both to exchange messages. Queue destinations are used for point-to-point messaging, while topic destinations are used for publish/subscribe messaging.

The corresponding destinations on the provider must be created through administrative facilities provided by the implementation. For example, the corresponding queues and topics must be defined to WebSphere MQ using the WebSphere MQ Explorer. Queues and topics for the default messaging provider can be configured on the service integration bus using the

WebSphere administrative tools.

Activation specifications

An activation specification is created and associated with a message-driven bean in order for the beans to receive messages. Note that if you are using third-party JMS providers that implement ASF, you would need to configure a message listener port instead an activation spec.

The underlying queues and topics on the messaging systems The JMS destinations are representations of a real endpoint provided by the JMS provider implementation.

WebSphere Application Server V7 supplies a pre-installed resource adapter for communicating with installations of the following products:

WebSphere MQ

WebSphere Event Broker

WebSphere Message Broker

New in V7: The WebSphere MQ messaging provider is now a J2EE™ Connector Architecture Version 1.5 compliant resource adapter. In WebSphere Application Server V7, it is now possible to create MQ messaging provider activation specifications to manage the relationship between an MDB running in WebSphere Application Server and a destination in WebSphere MQ.

A JMS connection factory is used by JMS clients to create connections to a messaging provider. To be compatible with JMS specification Version 1.0, there are two specific types of connection factories (queue connection factories and topic connection factories) and a more general type of connection factory. All three are configured in exactly the same way with minor exceptions.

In the administrative console, you will see three connection factory types:

Connection factories

Queue connection factories

Topic connection factories

Message-driven beans and activation specifications

WebSphere Application Server supports the use of message-driven beans as asynchronous message consumers. Clients send messages to a destination that is associated with a listener for a message-driven bean. When a message arrives at the destination, the EJB™ container invokes the message-driven bean, which processes the incoming message.

When messages are received using a JMS provider implemented with a JCA 1.5 resource adapter, such as the default messaging provider or the WebSphere MQ messaging provider, the message-driven beans use a J2C activation specification to listen for incoming messages. If the JMS provider does not have a JCA 1.5 resource adapter (for example, the V5 default messaging provider), you must configure JMS message-driven beans against a listener port.

Configuring JMS activation specifications

Listener ports are used instead of activation specifications in certain situations, such as when the messaging provider does not have a JCA 1.5 resource adapter, for compatibility with existing message-driven bean applications, or because you are using an EJB 2.0 message-driven bean and you do not want to upgrade the application.

A JMS activation specification is associated with a message-driven bean during application installation. The JMS activation specification object defines all of the properties that the J2EE

Connector Architecture requires or recommends an ActivationSpec JavaBean to support. It also defines other properties specific to using it in conjunction with a service integration bus.

Configuring the WebSphere MQ provider

The WebSphere MQ messaging provider can be configured to communicate with WebSphere MQ using a bindings or client connection. These two connectivity options are described below:

Bindings connection

When used in bindings mode, the WebSphere MQ messaging provider uses the Java Native Interface (JNI™) to call directly into the existing queue manager API, rather than communicating through a network. This provides better performance when connecting to WebSphere MQ than using a client connection.

However, to use a bindings connection, WebSphere MQ and WebSphere Application Server must be installed on the same machine.

Client connection

If it is not possible to collocate WebSphere Application Server and WebSphere MQ on the same machine, the WebSphere MQ messaging provider must be configured to connect to WebSphere MQ using TCP/IP. Using a client connection also allows you to perform authorization checks.

Additional considerations must be taken into account when configuring the WebSphere MQ messaging provider to use a client connection, for example:

– Whether the connection must be secured by encrypting the data that flows over the connection

– Whether the connection will go through a firewall

The sections that follow describe the properties exposed by WebSphere MQ connection factories and destinations, and also how to configure connection factories and destinations for the WebSphere MQ messaging provider.

The actual WebSphere MQ resources, such as queue managers, channels, and queues, must be created using the tools provided with WebSphere MQ.

Configuring activation specifications

MQ messaging provider activation specifications are now the preferred mechanism for delivering messages from a WebSphere MQ destination to a message-driven bean running in WebSphere Application Server. Activation specifications provide the following advantages over a message listener port:

Activation specifications are part of the JCA 1.5 specification.

Activation specifications are simple to configure.

Activation specifications can be defined at any scope and not restricted to server scope.

Thread pool for WebSphere MQ JMS provider

WebSphere MQ messaging provider uses threads from the WMQCommonServices thread pool to accomplish most of its tasks. Properties for this thread pool can be seen by opening the application server configuration page and selecting Thread pools in the Additional Properties section.

Default Messaging Concepts

The service integration bus (or just bus) provides a managed communications framework that supports a variety of message distribution models, reliability options, and network topologies. It provides support for traditional messaging applications, as well as enabling the implementation of service-oriented architectures (SOAs) within the WebSphere Application Server environment.

The service integration bus is the underlying messaging provider for the default messaging provider. The sections that follow discuss each of the concepts in more detail.

A bus consists of a group of interconnected application servers or clusters of application servers that have been added as members of the bus. Each member has a messaging engine that performs the message processing.

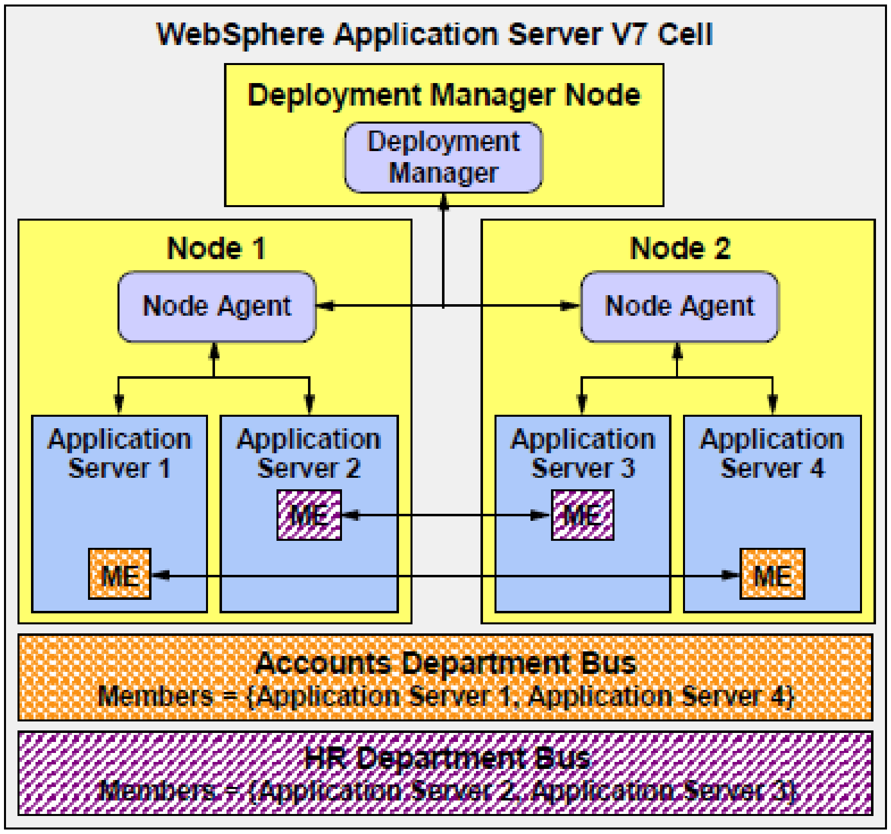
A bus is defined at the cell level. In a standard configuration, no more than one bus is normally required within a cell. However, a cell can contain any number of buses.

Bus Member

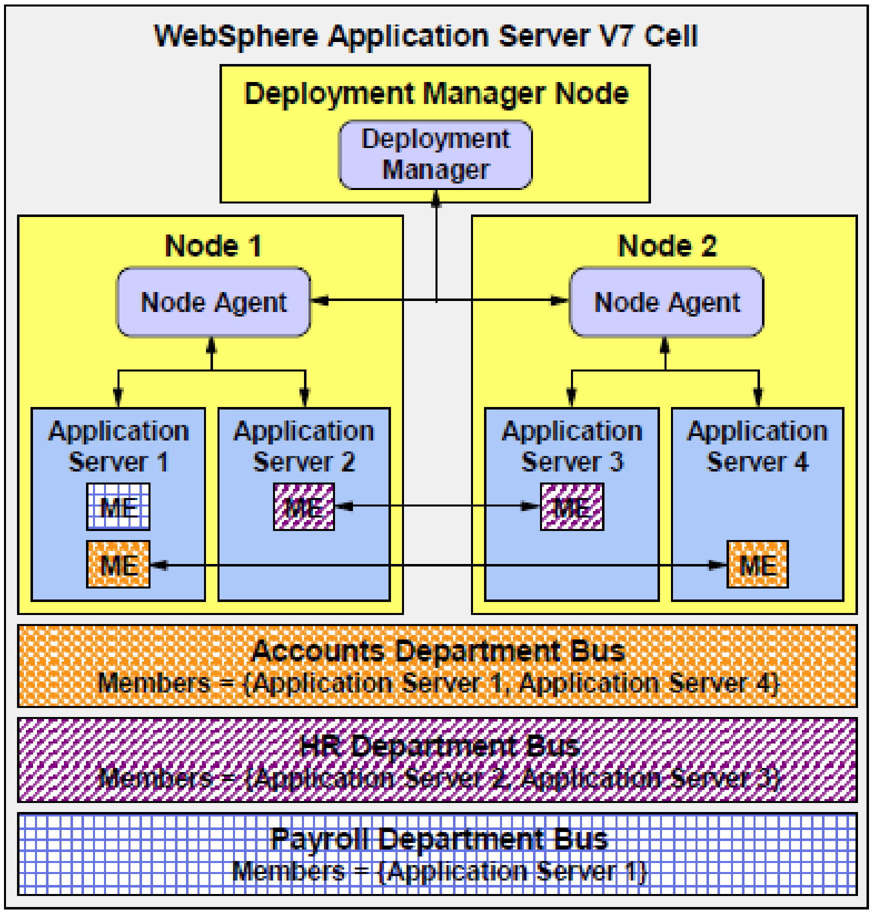
A bus member is simply an application server, a cluster of application servers, or an MQ Server that has been added as a member of a bus. Adding an application server, or cluster of application servers, as a member of a bus automatically defines a number of resources on the bus member in question. In terms of the functionality provided by a bus, the most important of the resources that are automatically defined is a messaging engine.

Messaging Engine

A messaging engine is the component within an application server that provides the core messaging functionality of a bus. At run time, it is the messaging engines within a bus that communicate and cooperate with each other to provide the messaging capabilities of the bus. A messaging engine is responsible for managing the resources of the bus and provides a connection point to which local and remote client applications can connect. A messaging engine is associated with a bus member. When an application server is added as a member of a bus, a messaging engine is automatically created and associated with this application server.



A messaging engine is a relatively lightweight runtime object. This allows a single application server to host several messaging engines. If an application server is added as a member of multiple buses, that application server is associated with multiple messaging engines, one messaging engine for each bus of which it is a member. In the following figure you can see that application server 1 is a member of two buses and has two MEs, one per bus.



<u>Message stores</u>

Every messaging engine defined within a bus has a message store associated with it. A messaging engine uses this message store to persist durable data, such as persistent messages and transaction states. Durable data written to the message store survives the orderly shutdown, or failure, of a messaging engine, regardless of the reason for the failure.

The messaging engine can also use the message store to reduce run time resource consumption. For example, the messaging engine can write non-persistent messages to the message store in order to reduce the size of the Java heap when handling high message volumes. This is known as spilling.

Message stores can be implemented as a set of database tables (known as a data store) or as flat files (known as a file store).

A data store is a message store implemented as a set of database tables within a relational database, accessed via a JDBC data source.

A file store is a message store implemented as a set of flat files within a file system that is accessed directly via the native operating system.

A data store can be used for a messaging engine hosted within an embedded Derby database. A JDBC data source to access this database is also defined on the server that has been added to the bus. These defaults allow the messaging engine to run without any further configuration.

However, while adding a bus member, it is possible to specify the JNDI name of a different data source for use by the messaging engine. The sections that follow describe the issues that must be considered when deciding which RDBMS to use as a data store.

Each messaging engine must have exclusive access to the tables defined within its data store. This can be achieved either by using a separate database from the data store for each messaging engine or by partitioning a single, shared database into multiple data stores using unique schema names for each data store.

Deciding which of these mechanisms to use depends on the capabilities of the RDBMS that will host the data store. For example, the embedded Derby database does not support concurrent access by multiple processes.

When you start a server that is part of a cluster bus member, the messaging engine will not always be started. In a high-availability topology, only one server in the cluster will have a messaging engine activated on it, and this messaging engine might already be started.

If this is the case, then you will see the messaging engine in the state Joined, but not Starting or Started. This is perfectly normal and means that the messaging engine is in a stand-by state, waiting to be activated should the currently active instance of the messaging engine become unavailable.

When you have more than one messaging engine in a bus, you will also see the messaging engines communicate with each other. Every messaging engine in the bus connects to every other messaging engine in the bus.

Example messages in SystemOut.log

CWSIT0028I: The connection for messaging engine Node1.server1-ITSOBus in bus ITSOBus to messaging engine Node2.server2-ITSOBus started.

CWSIP0382I: messaging engine B68588EF698F4527 responded to subscription request, Publish Subscribe topology now consistent.

Troubleshooting an Service Integration Bus

CWSIS1535E: Messaging engine’s unique ID does not match

The most likely cause of this is that you have deleted a messaging engine and then recreated a messaging engine by the same name using the default data store. This can happen, for example, when you add a server to a bus, then delete the bus. When you create a new bus and add the server to the new bus, the new messaging engine will use a default data source that points to the same database used by the old messaging engine, and this database will contain the ID of the old messaging engine.

This error can also be caused by configuring any messaging engine with the same message store as another messaging engine.

Solutions:

For a data store, the simplest solution is to drop the tables in the database, or delete and recreate the database and then restart the server. Another solution is to change the messaging engine’s data store by changing the schema, user, and database configured for the messaging engine. For a file store, delete the files or the directory paths.

CWSIT0019E: No suitable messaging engine

This exception can be thrown to a JMS client on a createConnection call. Causes of this exception include:

The JMS connection factory cannot contact an SIB service (for out of cell JMS clients only). Check that the provider endpoints listed in the connection factory match the host and port for the SIB services on the servers. Ensure that the SIB services are enabled and that the servers are started.

The bus name defined in the JMS connection factory does not match the name of a bus defined in WebSphere.

No messaging engines on the named bus are active.

To connect to a service integration bus, an application actually connects to a messaging engine on the bus. By default, the environment automatically connects applications to an available messaging engine on the bus. However you can specify extra configuration details to influence the connection process; for example to identify special bootstrap servers, or to limit connection to a subgroup of available messaging engines, or to improve availability or performance, or to ensure sequential processing of messages received.

Applications that are running in an application server are directed by the WebSphere® Application Server environment to an available messaging engine.

If the messaging engine is found in the same server, a connection is created that provides the application with the fastest available connection to a messaging engine. Otherwise, if a messaging engine is found in another process - on the same or a different host - a remote connection is made. If no suitable messaging engine is found the application fails to connect to the bus.

Connection to a messaging engine with applications running outside an application server.

The properties of a JMS connection factory used by a client application control the selection of a suitable messaging engine and how the client connects to the selected messaging engine. By default, a connection factory expects to use a bootstrap server that has an endpoint address of localhost:7276:BootstrapBasicMessaging. That is: the client application expects to use a bootstrap server that is on the same host as the client, and that uses port 7276 and the predefined bootstrap transport chain called BootstrapBasicMessaging.

When you create an application server, it is automatically assigned a unique non-secure bootstrap port, SIB_ENDPOINT_ADDRESS, and a secure bootstrap port, SIB_ENDPOINT_SECURE_ADDRESS. If you want to use an application server as a bootstrap server, and the server has been assigned a non-secure port other than 7276, or you want to use the secure port, then you must specify the endpoint address of the server on the Provider endpoints property of the connection factory.

The endpoint addresses for bootstrap servers must be specified in every connection factory that is used by applications outside of an application server. To avoid having to specify a long list of bootstrap servers, you can provide a few highly-available servers as dedicated bootstrap servers. Then you only have to specify a short list of bootstrap servers on each connection factory.

The selection process is used to choose a messaging engine that an application should connect to so that it can use the resources of a service integration bus. The information that controls the selection process is configured in one of the following places:

* For JMS client applications, this information is configured on the connection factory.
* For message-driven bean (MDB) applications, this information is configured on the activation specification.
* For other types of application, this information is configured programmatically by the application.

Although a connection can be made to any available messaging engine, the connection process applies a few simple rules to find the most suitable messaging engine. For an application running in an application server, the process is as follows:

1. If a messaging engine is running in the required bus within the same application server, then a connection is made from the application to the messaging engine. If there is no suitable messaging engine, the next rule is checked.
2. If a messaging engine is running on the same host as the application, then the application makes a remote connection to the selected messaging engine. If there is no suitable messaging engine, the next rule is checked.
3. If a messaging engine is running anywhere in the bus, then the application makes a remote connection to the selected messaging engine. If there is no suitable messaging engine, the connection attempt does not succeed.

For an application running outside an application server, connection requests are workload balanced across all the available messaging engines in the bus.

In both cases (that is, an application running in an application server and an application running outside an application server) you can restrict the range of messaging engines available for connection, to a subgroup of those available in the service integration bus.

For MDB applications connecting to a cluster bus member, you can also enable either of the following additional configurations:

* All servers in the cluster can receive messages from the MDB application, to make full use of the processing power in the cluster.
* Just one server at a time can receive messages from the MDB application, to ensure sequential processing of the messages.

createSIBJMSConnectionFactory command

Use the createSIBJMSConnectionFactory command to create a new JMS connection factory for the default messaging provider at a specific scope.

Example

* Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.createSIBJMSConnectionFactory("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)", ["-name", "jmscf1",
 "-jndiName", "jms/jmscf1", "-busName", "abus"])
'jmscf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733325084)'

wsadmin>AdminTask.createSIBJMSConnectionFactory("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)", ["-name", "jmsqcf2",
 "-jndiName", "jms/jmsqcf1", "-busName", "abus", "-type", "queue"])
'jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733675578)'
```

* createSIBJMSActivationSpec command

  Use the createSIBJMSActivationSpec command to create a new JMS activation specification for the default messaging provider at a specific scope.

* deleteSIBJMSActivationSpec command

  Use the deleteSIBJMSActivationSpec command to delete a JMS activation specification for the default messaging provider at a specific scope.

* listSIBJMSActivationSpecs command

  Use the listSIBJMSActivationSpecs command to list all JMS activation specifications for the default messaging provider at a specific scope.

* modifySIBJMSActivationSpec command

  Use the modifySIBJMSActivationSpec command to change properties of a JMS activation specification for the default messaging provider at a specific scope.

* showSIBJMSActivationSpec command

  Use the showSIBJMSActivationSpec command to show properties of a JMS activation specification for the default messaging provider at a specific scope.

* createSIBJMSConnectionFactory command

  Use the createSIBJMSConnectionFactory command to create a new JMS connection factory for the default messaging provider at a specific scope.

* deleteSIBJMSConnectionFactory command

  Use the deleteSIBJMSConnectionFactory command to delete a JMS connection factory for the default messaging provider at a specific scope.

* listSIBJMSConnectionFactories command

  Use the listSIBJMSConnectionFactories command to list all JMS connection factories for the default messaging provider at a specific scope.

* modifySIBJMSConnectionFactory command

  Use the modifySIBJMSConnectionFactory command to modify the properties of a JMS connection factory for the default messaging provider at a specific scope.

* showSIBJMSConnectionFactory command

  Use the showSIBJMSConnectionFactory command to show the properties of a JMS connection factory for the default messaging provider at a specific scope.

* createSIBJMSQueue command

  Use the createSIBJMSQueue command to create a new JMS queue for the default messaging provider at a specific scope.

* deleteSIBJMSQueue command

  Use the deleteSIBJMSQueue command to delete a JMS queue for the default messaging provider at a specific scope.

* listSIBJMSQueues command

  Use the listSIBJMSQueues command to list all JMS queues for the default messaging provider at a specific scope.

* modifySIBJMSQueue command

  Use the modifySIBJMSQueue command to change the properties of a JMS queue for the default messaging provider at a specific scope.

* showSIBJMSQueue command

Use the showSIBJMSQueue command to show properties of a JMS queue for the default messaging provider at a specific scope.

* createSIBJMSTopic command

  Use the createSIBJMSTopic command to create a new JMS topic for the default messaging provider at a specific scope.

* deleteSIBJMSTopic command

  Use the deleteSIBJMSTopics command to delete a JMS topic for the default messaging provider at a specific scope.

* listSIBJMSTopics command

  Use the listSIBJMSTopics command to list all JMS topics for the default messaging provider at a specific scope.

* modifySIBJMSTopic command

  Use the modifySIBJMSTopic command to change the properties of a JMS topic for the default messaging provider at a specific scope.

* showSIBJMSTopic command

  Use the showSIBJMSTopic command to show properties of a JMS topic for the default messaging provider at a specific scope.

Each service integration server or cluster bus member contains a component called a messaging engine that processes messaging send and receive requests and that can host destinations.

When you add an application server or a server cluster as a bus member, a messaging engine is automatically created for this new member. If you add the same server as a member of multiple buses, the server is associated with multiple messaging engines (one messaging engine for each bus). If the bus member is a WebSphere® MQ server, it does not have a messaging engine, but it lets you access WebSphere MQ queues directly from WebSphere MQ queue managers and (for WebSphere MQ for z/OS®) queue-sharing groups.

To host queue-type destinations, the messaging engine includes a message store where, if necessary, it can hold messages until consuming applications are ready to receive them, or preserve messages in case the messaging engine fails. There are two types of message store, file store and data store. For further information, see Administering message stores.

Messaging engines are given a name which is based on the name of the bus member. Each messaging engine also has a universal unique identifier (UUID) that provides a unique identity for the messaging engine.

Note: If you delete and recreate a messaging engine, it will have a different UUID and will not be recognized by the bus as the same engine, even though it might have the same name. For example, the recreated messaging engine will not be able to access the message store that the earlier instance used. If you accidentally delete a messaging engine configuration, and save the updated incorrect configuration, you must restore the configuration from a previous configuration backup.

Adding a messaging engine to a cluster

You can add a messaging engine to a cluster bus member to provide additional asynchronous messaging services to the servers that are members of the cluster. Typically, you do this to provide workload sharing or scalability in the cluster.

Before you begin

Ensure that you have defined a location for the message store for the messaging engine. Each messaging engine uses a message store to preserve operating and recovery information.

* To use a file store, you need a file location.

* To use a data store, you need a suitable data source, such as a relational database, that is accessed through a JDBC data source. You can use the default JDBC data source and Derby JDBC Provider for its data store. If you do not want to use the default data source configuration, you can use a different data source or you can configure the data store to use a different JDBC provider.

If you want any of the messaging engines in the cluster to fail over to another server, all servers that might host each messaging engine need access to the message store for that messaging engine.

About this task

Use this procedure when you want to add a messaging engine to an existing cluster bus member.

Alternatively, you can add one or messaging engines when you add a cluster as a member of a bus. You can use messaging engine policy assistance, which guides you through the creation and configuration of the messaging engines.

Procedure

1. In the navigation pane, click Service integration > Buses > bus_name > [Topology] Bus members.
2. In the content pane, click the name of the cluster to which you want to add a messaging engine. The Bus member detail pane is displayed.
3. In the content pane, under Additional properties, click Messaging engines. A list of messaging engines for the cluster is displayed.
4. In the content pane, click Add messaging engine.
5. Select the type of message store that you have already defined.
6. Enter details for the message store.
   * If you use a file store, specify the directory paths for the log files, the permanent file store, and the temporary file store. Do not use the default path, and ensure that you use a unique path for each messaging engine.
   * If you use a data store, specify the JNDI name of the data source that provides access to the database that holds the data store.
7. When the wizard is finished, save your changes to the master configuration.

Results

A messaging engine is added to the cluster. You can now configure the messaging engine if required.

Interconnected buses

A service integration bus topology can contain many interconnected service integration buses to form a large messaging network. The bus that an application connects to is called its local bus. There can be connections from that local bus to other service integration buses, which are called foreign buses. Buses can also be linked to WebSphere® MQ resources, for example WebSphere MQ queue managers. WebSphere MQ resources are also regarded as foreign buses.

A bus must be contained in a single cell; that is, a bus cannot span multiple cells. However, a cell can contain more than one bus. In this situation, each bus in the cell is foreign to each other bus in the cell. You can connect buses together in a cell, or between different cells.

The following scenarios are examples of situations when you might connect service integration buses in an organization:

* You can deliberately separate the messaging infrastructure to aid management.

* You can restrict access to certain messaging resources within a single WebSphere Application Server cell, because a cell can contain multiple service integration buses.

* You can span multiple administrative cells, by connecting a service integration bus in one cell to a service integration bus in another cell.

When buses are connected, applications can send messages to applications on other buses, and use resources provided on other buses. Published messages can span multiple buses where the connections between the buses are configured to allow it.

To create a connection between two buses, the administrator of the local bus configures a foreign bus connection that represents the second bus, and that is associated with the local bus. The foreign bus connection contains a routing definition, or virtual link. A physical link, called a service integration bus link, is created automatically. The link is from a messaging engine in the local bus to a messaging engine in the foreign bus, and these two messaging engines are known as gateway messaging engines. The administrator of the second bus also configures a foreign bus connection to represent the first bus, as a property of the second bus.

To create a link between a bus and a WebSphere MQ queue manager, the administrator of the local bus configures a foreign bus connection that represents the WebSphere MQ queue manager, as a property of the local bus. The foreign bus connection contains a routing definition, or virtual link. A physical link, called a WebSphere MQ link, is created automatically. The link is from a messaging engine in the local bus to a queue manager or queue sharing group in the foreign bus. The messaging engine is known as a gateway messaging engine, and the queue manager or queue sharing group is known as the gateway queue manager.

Security when connecting buses

A multiple bus topology has the following security requirements:

* You must protect the integrity and confidentiality of the data that is transported between the buses. You can protect the communications links by using a Secure Sockets Layer (SSL).

* You must establish trust between two buses. Trust between WebSphere Application Server Version 7.0 messaging engines is established by using a Lightweight Third Party Authentication (LTPA) token, and no further configuration is required.

If the bus has a WebSphere Application Server Version 6.x bus member (that is, a mixed-version bus), trust is established using an inter-engine authentication alias. The inter-engine authentication alias is configured when you add a member to a bus or with the bus security settings. The identity is passed to the remote bus where the identity is authenticated, then checked to see if it matches the configured inter-engine authentication alias on the other bus.

* You need the definition of relevant authorizations to allow messages to travel between the buses. There are two phases to authorization when communicating with a foreign bus:

1. The user that is connected to the local bus has to be explicitly granted access to send messages to the foreign destination. Failure at this level is reported back to the client.

2. The foreign bus must be configured to accept the incoming message onto the target destination

Connecting buses in different cells

To connect a local bus to a foreign bus in a different cell from the local bus, you need to provide a value for one or more bootstrap endpoints, that is, the host, port location, and transport chain for the messaging engine on the foreign bus that the local service integration bus connects to.

Connecting buses with cluster bus members

To connect a local bus to a foreign bus in a different cell from the local bus when the remote messaging engine is in a cluster, you must change the value for the bootstrap endpoints. This value must list all the bootstrap endpoints that the cluster uses to allow access to the gateway messaging engine in the cluster.

Service integration troubleshooting

Checking the communication between two messaging engines in a bus

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

Procedure

1. Check that both messaging engines are running.
2. For each messaging engine:
    1. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.
    2. Find the most recent instance (there might be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Multiple-server bus without clustering

A bus that consists of multiple servers provides advantages of scalability, the ability to handle more client connections, and greater message throughput. A multiple-server bus Includes multiple messaging engines that can share the work of distributing the messages.

Another advantage of a multiple-server bus is that you can locate the queue that an application consumes from in the same application server as that application, which might be more efficient if there are multiple application servers running applications.

You can configure a bus to have multiple server bus members, each of which runs one messaging engine. All the servers in the bus must belong to the same cell.

All the messaging engines in the bus are implicitly connected, and applications can connect to any messaging engine in the bus. All the messaging engines in the bus know about the resources that are assigned to each messaging engine in that bus.

The messaging engines do not need to all run at the same time; if one messaging engine stops, the other messaging engines in the bus continue to operate. However, if a messaging engine stops, the resources that the messaging engine owns are unavailable. Specifically, the destinations that are assigned to that messaging engine are unavailable.

Planning a bus topology

These topics contain information about planning a service integration bus topology. This information is intended primarily for use by solution architects and designers who are in the early stages of planning a bus topology.

About this task

To plan a bus topology, you must consider the following issues:

* How many buses will the topology comprise?
* How will messaging engines and bus destinations be distributed?
* What are the naming conventions for buses, messaging engines, and other service integration bus resources?

Tip: Each bus name in a cell must be unique. It is advisable to use unique names for all buses, even when they are in different cells, because you cannot connect two buses with the same name.

Remember: Your bus topology might be a combination of the topologies described in these topics. For example, a multiple-bus topology might include clustering.

When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the pass message payload by reference properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

F) Automate deployment tasks with scripting.

wsadmin scripting utility

To obtain general help, use the following examples:

Using Jacl:

```
$AdminTask help
```

Using Jython:

```
print AdminTask.help()
```

-f

Specifies a script to run.

Only one -f option can exist on the command line.

```
wsadmin -lang jython -f  test2.py
```

<u>Using Ant to automate tasks</u>

To support using Apache Ant with Java™ Platform, Enterprise Edition (Java EE) applications running on the application server, the product provides a copy of the Ant tool and a set of Ant tasks that extend the capabilities of Ant to include product-specific functions. Ant has become a very popular tool among Java programmers.

Apache Ant is a Java-based build tool. In theory, it is similar to Make, but Ant is different. Instead of a model in which it is extended with shell-based commands, Ant is extended using Java classes. Instead of writing shell commands, XML-based configuration files are used. These files reference a target tree in which various tasks are run. Each task is run by an object that implements a particular Task interface.

By combining the following tasks with those provided by Ant, you can create build scripts that compile, package, install, and test your application on the application server:

* Install and uninstall applications
* Start and stop servers in a base configuration
* Run administrative scripts or commands
* Run the Enterprise JavaBeans™ (EJB) deployment tool for EJB 1.x or 2.x modules
* Run the JavaServer Pages (JSP) file precompilation tool

# To run Ant and have it automatically see the WebSphere® classes, use the ws_ant command.

The ws_ant command is provided with the Apache Ant tool.

See the app_server_root/bin/ws_ant.bat|sh file for the Apache Ant tool.

# Use Ant tasks for deployment and server operation.

The Apache Ant tasks for the product reside in the Java package: com.ibm.websphere.ant.tasks. The API documentation for this package contains detailed information about all of the Ant tasks that are provided and how to use them.

See com.ibm.websphere.ant.tasks API documentation in the Reference section of the information center.

Automating SSL configurations using scripting

SSL configuration is needed for WebSphere® to perform SSL connections with other servers. A SSL configuration can be configured through the Admin Console. But if an automated way to create a SSL configuration is desired then AdminTask should be used.

AdminTask can be used in a interactive mode and batch mode. For automation the batch mode options should be used. AdminTask batch mode can be called in a JACL or Python script. Interactive mode will step through all the parameter the task needs, requires ones are marked with a '*'. Before the interactive task executes the task it echoes the batch mode syntax of the task to the screen. This can be helpful when writing batch mode scripts.

There attributes needed to create an ssl configurations:

* A key store
* Default client certificate alias
* Default server certificate alias
* Trust store
* The handshake protocol
* The ciphers needed during handshake
* Supporting client authentication or not

If automating the creation of a SSL Configuration it may be needed to create some of the attribute values needed like the key store, trust store, key manager, and trust managers.

```
AdminTask.createSSLConfig ('[interactive]')
```

Automating application configurations using wsadmin scripting

The scripting library provides Jython script procedures to assist in automating your environment. Use the application management scripts to install, uninstall, export, start, stop, and manage applications in your environment.

Use a text editor to combine several scripts from the Jython script library, as the following sample displays:

```
#

# My Custom Jython Script - file.py

#

AdminServerManagement.createApplicationServer("myNode", "Server1", "default")

AdminServerManagement.createApplicationServer("myNode", "Server2", "default")


# Use one of them as the first member of a cluster

AdminClusterManagement.createClusterWithFirstMember("myCluster",
"APPLICATION_SERVER", "myNode", "Server1")


# Add a second member to the cluster

AdminClusterManagement.createClusterMember("myCluster", "myNode", "Server3")


# Install an application

AdminApplication.installAppWithClusterOption("DefaultApplication",
"..\installableApps\DefaultApplication.ear", "myCluster")


# Start all servers and applications on the node

AdminServerManagement.startAllServers("myNode")
```

Save the custom script and run it from the command line, as the following syntax demonstrates:

```
bin>wsadmin -language jython -f path/to/your/jython/file.py
```

Create custom scripts to automate your environment by combining script procedures from the scripting library. Save custom scripts to a new subdirectory of the app_server_root/scriptLibraries directory.

The scripting library provides Jython script procedures to assist in automating your environment. Use the scripts in the AdminResources script library to configure mail, URL, and resource settings.

Use the Jython scripting library code as sample syntax to write custom scripts. Each script example in the script library demonstrates best practices for writing wsadmin scripts. The script library code is located in the app_server_root/scriptLibraries directory. Within this directory, the scripts are organized into subdirectories according to functionality, and further organized by version. For example, the app_server_root/scriptLibraries/application/V70 subdirectory contains procedures that perform application management tasks that are applicable to Version 7.0 and later of the product.

Application deployment configuration scripts

The scripting library provides multiple script procedures to automate your application configurations. This topic provides usage information for scripts that deploy applications. You can run each script individually or combine procedures to create custom automation scripts for your environment.

Each application management script procedure is located in the app_server_root/scriptLibraries/application/V70 directory. The application deployment script procedures contain multiple arguments. If you do not want to specify an argument with the script, specify the value of the argument as an empty string, as the following syntax demonstrates: "".

ejbdeploy command

The EJB deployment tool provides a command-line interface that you can use to generate enterprise bean deployment code. Before you can successfully run your enterprise beans on either a test or production server, you need to generate deployment code for the enterprise beans.

You generate EJB deployment code by running the ejbdeploy command.

When ejbdeploy runs, it creates an Eclipse workspace in the directory that you specify as the working directory: d:\deploydir. When it has completed running, it deletes this workspace. However, the -keep option causes ejbdeploy to end without deleting the workspace.

By default, before EJB deployment code is generated when you use the ejbdeploy command from the command line, validation is automatically run on the application.

When the EJB validator issues an error, warning, or information message, the location and the source of the message is displayed in the following format (an error message is used as an example):

[Error:]JAR_name(location):message_text

addNode command

The addNode command incorporates an application server installation into a cell.

The node agent server is automatically started as part of the addNode command unless you specify the -noagent option. If you recycle the system that hosts an application server node, and did not set up the node agent to act as an operating system daemon, you must issue a startNode command to start the node agent before starting any application servers.

When you run the addNode command, the command stops the running application server that it is incorporating into a cell. You can optionally stop the application server prior to running the addNode command.

The dmgr_host argument is required. All of the other arguments are optional. The default port number is 8879 for the default SOAP port of the deployment manager. SOAP is the default Java™ Management Extensions (JMX) connector type for the command.

When the -includeapps parameter is specified, an OutOfMemoryError might occur if the Java virtual machine (JVM) heap size is too small. When this error occurs, the following error message is issued:

```
ADMU0111E: Program exiting with error: java.lang.OutOfMemoryError
```

This error can occur when large applications are processed, or when there is a large number of applications in the Base Application Server.

To recover from this error and successfully federate the application server, complete the following actions:

1. Issue the cleanupNode command on your deployment manager server. Read about the cleanupNode command for more information about this command.
2. Increase the JVM heap size for the addNode script. When you issue the addNode command, the JVM heap size is set to -Xms128m -Xmx512m. To increase these values, use the -javaoption parameter. For example, you might specify the following (all on one line):

   [Windows]

   ```
   addNode localhost 8879 -javaoption java_option "-Xmx512m"
   ```

   [Linux] [AIX]

   ```
   addNode.sh localhost 8879 -javaoption java_option "-Xmx512m"
   ```

When adding a node to the cell, you automatically inherit both the user registry and the authentication mechanism of the cell.

backupConfig command

The backupConfig command is a simple utility to back up the configuration of your node to a file.

The following example creates a file called myBackup.zip and does not stop any servers before beginning the backup process:

[AIX] [HP-UX] [Linux] [Solaris]

```
backupConfig.sh myBackup.zip -nostop
```

[Windows]

```
backupConfig.bat myBackup.zip -nostop
```

restoreConfig command

Use the restoreConfig command to restore the configuration of your node after backing up the configuration using the backupConfig command.

The following example restores the given file to the /tmp directory and does not stop any servers before beginning the restoration:

[AIX] [HP-UX] [Linux] [Solaris]

```
restoreConfig.sh WebSphereConfig_2006-04-22.zip -location /tmp -nostop
```

[Windows]

```
restoreConfig.bat WebSphereConfig_2006-04-22.zip -location /tmp -nostop
```

collector command - summary option

The collector summary option helps you communicate with WebSphere Application Server technical staff at IBM® Support. Run the collector tool with the -Summary option to produce a lightweight text file and console version of some of the information in the Java™ archive (JAR) file that the tool produces without the -Summary parameter. You can use the collector summary option to retrieve basic configuration and prerequisite software level information when starting a conversation with IBM Support.

The collector summary option produces version information for the WebSphere Application Server product and the operating system as well as other information. It stores the information in the Collector_Summary.txt file and writes it to the console. You can use the information to answer initial questions from IBM Support or you can send the Collector_Summary.txt file directly to IBM Support.

Run the collector command to create the JAR file if IBM Support needs more information to solve your problem.

To run the collector summary option, start from a temporary directory outside of the WebSphere Application Server product installation root directory and enter one of the following commands:

* [Linux] [AIX HP-UX Solaris] app_server_root/bin/collector.sh -Summary
* [Windows] app_server_root \bin\collector.bat -Summary

createCertRequest command

The createCertRequest command creates a PKCS10 certificate request and stores it in a client keystore so that it can be used to send to a certificate authority (CA) server using the requestCertificate command line utility.

Issue the command from the profile_root/bin directory.

Syntax

The command syntax is as follows:

[AIX] [HP-UX] [Linux] [Solaris]

```
createCertRequest.sh -keyStoreAlias<keystoreAlias> -subjectDN<subjectDN>
-alias<certificateAlias> [options]
```

The following required parameter are used with the createCertRequest command

-keyStoreAlias keyStoreAlias

-subjectDN subjectDN

-alias certificateAlias

dumpNameSpace tool

You can use the dumpNameSpace tool to dump the contents of a namespace accessed through a name server. The dumpNameSpace tool is based on Java™ Naming and Directory Interface (JNDI).

If you run the dumpNameSpace tool with security enabled and the com.ibm.CORBA.loginSource property is set in the profile_root/properties/sas.client.props file, a login prompt is displayed.

G) Manage the plug-in configuration file (e.g., regenerate, edit, propagate, etc.).

An application-centric file has an application that is mapped to both Web server and application server definitions. Changes that you make to the plug-in using the administrative console persist to the plugin-cfg.xml file upon generation.

The design of this file and its related function enable the product to support different types of configurations. For example, Web server definitions might not exist. In addition, many Web server plug-in properties, such as RefreshInterval, LogLevel, and the Edge Side Include (ESI) processor properties, only can be updated manually. Those values must be maintained through each iteration.

ServerCluster (one or more elements for each Config)

A group of servers that are generally configured to service the same types of requests.

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

LogLevel (zero or one attribute for each Log)

The level of detail of the log messages that the plug-in should write 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.

If a LogLevel is not specified for the Log element, the default value Error is used.

Use the administrative console to change the settings in the plug-in configuration file.

You can either use the administrative console, or issue the GenPluginCfg command to regenerate your plugin-cfg.xml file.

It is recommended that you do not manually update the plugin-cfg.xml file. Any manual updates you make for a given Web server are overridden whenever the plugin-cfg.xml file for that Web server is regenerated.

GenPluginCfg command

Parameters

The following options are available for the GenPluginCfg command:

-config.root configroot_dir

Defaults to CONFIG_ROOT. The setupCmdLine command is invoked to get this environment variable.

-profileName

Defines the profile of the Application Server process in a multi-profile installation. The -profileName option is not required for running in a single profile environment. The default for this option is the default profile.

-cell.name cell

Defaults to WAS_CELL. The setupCmdLine command is invoked to get this environment variable.

-node.name node

Defaults to WAS_NODE. The setupCmdLine command is invoked to get this environment variable.

-webserver.name webserver1

Required for creating plug-in configuration file for a given Web server.

-propagate yes/no

Applicable only when the webserver.name option is specified and the Web server is local. Otherwise, you must manually copy the plugin-cfg.xml file from app_server_root/profiles/profile_name/config\cells\cell_name\nodes\node_name\servers\web_server_name to plugins_root/config/web_server_name in the remote Web server plugins directory. The default value is no.

-propagateKeyring yes/no

Applicable only when the option webserver.name is specified and the Web server is local. Defaults to no.

-cluster.name cluster1,cluster2 | ALL

Optional list of clusters. Ignored when the option webserver.name is specified.

-server.name server1,server2

Optional list of servers. Required for single server plug-in generation. Ignored when the option webserver.name is specified.

-output.file.name file_name

Defaults to the configroot_dir/plugin-cfg.xml file. Ignored when the option webserver.name is specified.

-destination.root root

Installation root of the machine configuration is used on. Ignored when the option webserver.name is specified.

-destination.operating.system windows/unix

Operating system of the machine configuration is used on. Ignored when the option webserver.name is specified.

-debug yes/no

Defaults to no.

-help

Prints a usage statement.

-?

Prints a usage statement.

If you want to use Secure-socket layer (SSL) with this configuration, use the plug-in's installation wizard to install the appropriate GSKIT installation image file on your workstation.

If you encounter problems restarting your Web server, check the http_plugin.log file for information on what portion of the plugin-cfg.xml file contains an error.

Personal certificates contain a private key and a public key. You can extract the public key, called the signer certificate, to a file, then import the certificate into another keystore. The client requires the signer portion of a personal certificate for Security Socket Layer (SSL) communication

Procedure

1. Click Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > ssl_configuration > Key stores and certificates > keystore .
2. Under Additional Properties, click Personal certificates.
3. Select a personal certificate.
4. Click Extract.
5. Type the full path for the certificate file name. The signer certificate is written to this certificate file.
6. Select a data type from the list.
7. Click Apply

LoadBalanceWeight (zero or one attribute for each Server)

Specifies the weight associated with this server when the plug-in does weighted Round Robin load balancing. The starting value for a server can be any integer between 0 and 20. However, zero should be specified only for a server that is shut down.

The LoadBalanceWeight for each server is decremented for each request processed by that server. After the weight for a particular server in a server cluster reaches zero, only requests with session affinity are routed to that server. When all servers in the cluster reach a weight of zero, the weights for all servers in the cluster are reset, and the algorithm starts over.

When a server is shut down, it is recommended that you set the weight for that server to zero. The plug-in can then reset the weights of the servers that are still running, and maintain proper load balancing.

MaxConnections (one element for each Server)

The MaxConnections attribute is used to specify the maximum number of active connections to an Application Server that can be flowing through a Web server process at any point in time.

For example, assuming that:

* The application server is fronted by 5 nodes that are running an IBM HTTP Server.
* Each node starts 2 processes.
* The MaxConnections attribute is set to 50.

In this example, the application server could potentially get up to 500 connections. (You take the number of nodes, 5, multiply it by the number of processes, 2, and then multiply that number by the number specified for the MaxConnections attribute, 50, for a total of 500 connections.)

By default, MaxConnections is set to -1. If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the Application Servers.

Gskit install images files

The Global Security Kit (GSKit) installation image files for the WebSphere® Web server plug-ins are packaged on the CD with the Web server plug-in files.

You can download the appropriate GSKIT file to the workstation on which your Web server is running.

H) Configure class loader parameters.

Class loaders find and load class files. Class loaders enable applications that are deployed on servers to access repositories of available classes and resources. Application developers and deployers must consider the location of class and resource files, and the class loaders used to access those files, to make the files available to deployed applications.

Class loaders used and the order of use

The product runtime environment uses the following class loaders to find and load new classes for an application in the following order:

1. The bootstrap, extensions, and CLASSPATH class loaders created by the Java™ virtual machine

   The bootstrap class loader uses the boot class path (typically classes in jre/lib) to find and load classes. The extensions class loader uses the system property java.ext.dirs (typically jre/lib/ext) to find and load classes. The CLASSPATH class loader uses the CLASSPATH environment variable to find and load classes.

   The CLASSPATH class loader loads the Java Platform, Enterprise Edition (Java EE) application programming interfaces (APIs) provided by the WebSphere Application Server product in the j2ee.jar file. Because this class loader loads the Java EE APIs, you can add libraries that depend on the Java EE APIs to the class path system property to extend a server class path. However, a preferred method of extending a server class path is to add a shared library.

2. A WebSphere extensions class loader

   The WebSphere extensions class loader loads the WebSphere Application Server classes that are required at run time. The extensions class loader uses a ws.ext.dirs system property to determine the path that is used to load classes. Each directory in the ws.ext.dirs class path and every Java archive (JAR) file or ZIP file in these directories is added to the class path used by this class loader.

The WebSphere extensions class loader also loads resource provider classes into a server if an application module installed on the server refers to a resource that is associated with the provider and if the provider specifies the directory name of the resource drivers.

3. One or more application module class loaders that load elements of enterprise applications running in the server

   The application elements can be Web modules, enterprise bean (EJB) modules, resource adapter archives (RAR files), and dependency JAR files. Application class loaders follow Java EE class-loading rules to load classes and JAR files from an enterprise application. The product enables you to associate shared libraries with an application.

4. Zero or more Web module class loaders

   By default, Web module class loaders load the contents of the WEB-INF/classes and WEB-INF/lib directories. Web module class loaders are children of application class loaders. You can specify that an application class loader load the contents of a Web module rather than the Web module class loader.

Each class loader is a child of the previous class loader. That is, the application module class loaders are children of the WebSphere extensions class loader, which is a child of the CLASSPATH Java class loader. Whenever a class needs to be loaded, the class loader usually delegates the request to its parent class loader. If none of the parent class loaders can find the class, the original class loader attempts to load the class. Requests can only go to a parent class loader; they cannot go to a child class loader. If the WebSphere extensions class loader is requested to find a class in a Java EE module, it cannot go to the application module class loader to find that class and a ClassNotFoundException error occurs. After a class is loaded by a class loader, any new classes that it tries to load reuse the same class loader or go up the precedence list until the class is found.

For each application server in the system, you can set the application class-loader policy to Single or Multiple. When the application class-loader policy is set to Single, then a single application class loader loads all EJB modules, dependency JAR files, and shared libraries in the system. When the application class-loader policy is set to Multiple, then each application receives its own class loader that is used for loading the EJB modules, dependency JAR files, and shared libraries for that application.

Class-loader isolation policies

The number and function of the application module class loaders depend on the class-loader policies that are specified in the server configuration. Class loaders provide multiple options for isolating applications and modules to enable different application packaging schemes to run on an application server.

Two class-loader policies control the isolation of applications and modules:

Application

Application class loaders load EJB modules, dependency JAR files, embedded resource adapters, and application-scoped shared libraries. Depending on the application class-loader policy, an application class loader can be shared by multiple applications (Single) or unique for each application (Multiple). The application class-loader policy controls the isolation of applications that are running in the system. When set to Single, applications are not isolated.

When set to Multiple, applications are isolated from each other.

WAR

By default, Web module class loaders load the contents of the WEB-INF/classes and WEB-INF/lib directories. The application class loader is the parent of the Web module class loader. You can change the default behavior by changing the Web application archive (WAR) class-loader policy of the application.

The WAR class-loader policy controls the isolation of Web modules. If this policy is set to Application, then the Web module contents also are loaded by the application class loader (in addition to the EJB files, RAR files, dependency JAR files, and shared libraries). If the policy is set to Module, then each Web module receives its own class loader whose parent is the application class loader.

Tip: The console and the underlying deployment.xml file use different names for WAR class-loader policy values. In the console, the WAR class-loader policy values are Application or Module. However, in the underlying deployment.xml file where the policy is set, the WAR class-loader policy values are Single instead of Application, or Multiple instead of Module. Application is the same as Single, and Module is the same as Multiple.

Restriction: WebSphere Application Server class loaders never load application client modules.

For each application server in the system, you can set the application class-loader policy to Single or Multiple. When the application class-loader policy is set to Single, then a single application class loader loads all EJB modules, dependency JAR files, and shared libraries in the system. When the application class-loader policy is set to Multiple, then each application receives its own class loader that is used for loading the EJB modules, dependency JAR files, and shared libraries for that application.

An application class loader loads classes from Web modules if the application's WAR class-loader policy is set to Application. If the application's WAR class-loader policy is set to Module, then each WAR module receives its own class loader.

Class-loader modes

The class-loader delegation mode, also known as the class loader order, determines whether a class loader delegates the loading of classes to the parent class loader. The following values for class-loader mode are supported:

Parent first

Also known as Classes loaded with parent class loader first.

The Parent first class-loader mode causes the class loader to delegate the loading of classes to its parent class loader before attempting to load the class from its local class path. This value is the default for the class-loader policy and for standard JVM class loaders.

Parent last

Also known as Classes loaded with local class loader first or Application first.

The Parent last class-loader mode causes the class loader to attempt to load classes from its local class path before delegating the class loading to its parent. Using this policy, an application class loader can override and provide its own version of a class that exists in the parent class loader.

The following settings determine the mode of a class loader:

* If the application class-loader policy of an application server is Single, the server-level mode value defines the mode for an application class loader.
* If the application class-loader policy of an application server is Multiple, the application-level mode value defines the mode for an application class loader.
* If the WAR class-loader policy of an application is Module, the module-level mode value defines the mode for a WAR class loader.

Section 4 - WebSphere Application Server Security (16%)

A) Implement security policies (e.g., authentication and authorization (using different security registries), global security, etc.)
$Protect WebSphere Application Server Network Deployment V7.0 resources (e.g., Java 2 security, Java EE security roles).

An application server plays an integral part in the multiple-tier enterprise computing framework. IBM WebSphere Application Server adopts the open architecture paradigm and provides many plug-in points to integrate with enterprise software components. Plug-in points are based on standard J2EE specifications wherever applicable.

When Java 2 Security is enabled and if an application requires more Java 2 security permissions than are granted in the default policy, then the application might fail to run properly until the required permissions are granted in either the app.policy file or the was.policy file of the application. AccessControl exceptions are generated by applications that do not have all the required permissions. Review the Java 2 Security and Dynamic Policy documentation if you are unfamiliar with Java 2 security.

Updates to the app.policy file only apply to the enterprise applications on the node to which the app.policy file belongs.

Select which authentication mechanism is active when security is enabled from the Authentication mechanisms and expiration menu. In this release of WebSphere Application Server, the authentication mechanism choices include LTPA and Kerberos.

Note: SWAM was deprecated in WebSphere Application Server Version Version 7.0 and will be removed in a future release.

Use the User account repository menu to specify the repository that is active when security is enabled. You can configure settings for one of the following user repositories:

Federated repositories

The federated repositories functionality enables you to use multiple registries with WebSphere Application Server. These registries, which can be file-based registries, LDAP registries, or a sub-tree of an LDAP registry, are defined and theoretically combined under a single repository.

Local operating system

The implementation is a System Authorization Facility (SAF) compliant registry such as the Resource Access Control Facility (RACF®), which is shared in an MVS™ sysplex.

Standalone LDAP registry

The standalone LDAP registry settings are used when users and groups reside in an external LDAP directory. When security is enabled and any of these properties are changed, go to the Global security panel and click OK or Apply to validate the changes.

Standalone custom registry

The standalone custom registry feature supports any user registry that is not implemented by WebSphere Application Server. You can use any user registry that is used in the product environment by implementing the UserRegistry interface.

In previous releases of WebSphere® Application Server, when a user enabled global security, both administrative and application security were enabled. In WebSphere Application Server Version 6.1, the previous notion of global security is split into administrative security and application security, each of which you can enable separately.

As a result of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. To enable application security, you must enable administrative security. Application security is in effect only when administrative security is enabled.

When the Use Java 2 security to restrict application access to local resources option is enabled and if an application requires more Java 2 security permissions than are granted in the default policy, the application might fail to run properly until the required permissions are granted in either the app.policy file or the was.policy file of the application. AccessControl exceptions are generated by applications that do not have all the required permissions.

Available realm definitions

Specifies the available user account repositories.

The selections appear in a drop-down list containing:

* Local operating system

* Standalone LDAP registry

* Standalone custom registry

The com.ibm.websphere.java2secman.norethrow property

When Java 2 security is enabled in WebSphere Application Server, the security manager component creates a java.security.AccessControl exception when a permission violation occurs. This exception, if not handled, often causes a run-time failure. This exception is also logged in the SYSOUT file.

However, when the Java virtual machine com.ibm.websphere.java2secman.norethrow property is set and has a value of true, the security manager does not create the AccessControl exception. This information is logged.

This property is intended for a sandbox or debug environment because it instructs the security manager not to create the AccessControl exception. Java 2 security is not enforced. Do not use this property in a production environment where a relaxed Java 2 security environment weakens the integrity that Java 2 security is intended to produce.

This property is valuable in a sandbox or test environment where the application can be thoroughly tested and where the system log or trace files can be inspected for AccessControl exceptions. Because this property does not create the AccessControl exception, it does not propagate the call stack and does not cause a failure. Without this property, you have to find and fix AccessControl exceptions one at a time.

Java 2 security

Java™ 2 security provides a policy-based, fine-grain access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources. Java 2 security guards access to system resources such as file I/O, sockets, and properties. Java 2 Platform, Enterprise Edition (J2EE) security guards access to Web resources such as servlets, JavaServer Pages (JSP) files and Enterprise JavaBeans™ (EJB) methods.

Because Java 2 security is relatively new, many existing or even new applications might not be prepared for the very fine-grain access control programming model that Java 2 security is capable of enforcing. Administrators need to understand the possible consequences of enabling Java 2 security if applications are not prepared for Java 2 security. Java 2 security places some new requirements on application developers and administrators.

Important: Java 2 security only restricts Java programs that run in a Java virtual machine that has Java 2 security enabled. It does not protect system resources if Java 2 Security is disabled or if system resources are accessed from other programs or commands. Therefore, if you want to protect your system resources, you need to use operating system security.

If your applications, or third-party libraries are not ready, having Java 2 security enabled causes problems. You can identify these problems as Java 2 security AccessControlExceptions in the system log or trace files.

Enabling security

By enabling security, you protect your server from unauthorized users and are then able to provide application isolation and requirements for authenticating application users.

Procedure

1. Start the WebSphere Application Server administrative console.

Start the deployment manager and, in your browser, type in the address of your WebSphere Application Server Network Deployment server. By default, the console is located at http://your_host.your_domain:9060/ibm/console.

2. Click Security > Global security.

Use the Security Configuration Wizard, or configure security manually. The configuration order is not important.

Avoid trouble: You must separately enable administrative security, and application security. Because of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. Before you attempt to enable application security on the target server, verify that administrative security is enabled on that server. Application security can be in effect only when administrative security is enabled.

3. Configure the user account repository. For more information, see Selecting a registry or repository. On the Global security panel, you can configure user account repositories such as federated repositories, local operating system, standalone Lightweight Directory Access Protocol (LDAP) registry, and standalone custom registry. The Primary administrative user name does not run WebSphere Application Server processes. Rather, the process ID runs the WebSphere Application Server processes. If you choose the local operating system registry, the process ID requires special privileges to call the operating system APIs.

4. Select the Set as current option after you configure the user account repository. When you click Apply and the Enable administrative security option is set, a verification occurs to see if an administrative user ID has been configured and is present in the active user registry. The administrative user ID can be specified at the active user registry panel or from the console users link. If you do not configure an administrative ID for the active user registry, the validation fails.

Note: When you switch user registries, the admin-authz.xml file should be cleared of existing administrative ids and application names. Exceptions will occur in the logs for ids that exist in the admin-authz.xml file but do not exist in the current user registry.

5. Configure the authentication mechanism.

Configure Lightweight Third-Party Authentication (LTPA) or Kerberos, which is new to this release of WebSphere Application Server, under Authentication mechanisms and expiration. LTPA credentials can be forwarded to other machines. For security reasons, credential expire;

however, you can configure the expiration dates on the console. LTPA credentials enable browsers to visit different product servers, which means you do not have to authenticate multiple times. For more information, see Configuring the Lightweight Third Party Authentication mechanism. Import and export the LTPA keys for cross-cell single Sign-on (SSO) between cells.

6. Configure the authentication protocol for special security requirements from Java™ clients, if needed.

You can configure Common Secure Interoperability Version 2 (CSIv2) through links on the Global security panel. The Security Authentication Service (SAS) protocol is provided for backwards compatibility with previous product releases, but is deprecated. Links to the SAS protocol panels display on the Global security panel if your environment contains servers that use previous versions of WebSphere Application Server and support the SAS protocol.

7. Secure Socket Layers (SSL) is pre-configured by default, changes are not necessary unless you have custom SSL requirements. You can modify or a create a new SSL configuration. This action protects the integrity of the messages sent across the Internet. The product provides a centralized location to configure SSL configurations that the various WebSphere Application Server features that use SSL can utilize, including the LDAP registry, Web container and the RMI/IIOP authentication protocol (CSIv2). For more information, see Creating a Secure Sockets Layer configuration. After you modify a configuration or create a new configuration, specify it on the SSL configurations panel. To get to the SSL configurations panel, complete the following steps:

    1. Click Security > SSL certificate and key management.

    2. Under Configuration settings, click Manage endpoint security configurations > configuration_name.

    3. Under Related items for each scope (for example, node, cluster, server), select one of the many configuration links that can be scoped to the resource you are visiting.

You can either edit the DefaultSSLConfig file or create a new SSL configuration with a new alias name. If you create a new alias name for your new keystore and truststore files, change every location that references the DefaultSSLConfig SSL configuration alias. The following list specifies the locations of where the SSL configuration repertoire aliases are used in the WebSphere Application Server configuration.

For any transports that use the new network input/output channel chains, including HTTP and Java Message Service (JMS), you can modify the SSL configuration repertoire aliases in the following locations for each server:

* Click Server > Application server > server_name. Under Communications, click Ports. Locate a transport chain where SSL is enabled and click View associated transports. Click transport_channel_name. Under Transport Channels, click SSL Inbound Channel (SSL_2).

* Click System administration > Deployment manager. Under Additional properties, click Ports. Locate a transport chain where SSL is enabled and click View associated transports. Click transport_channel_name. Under Transport Channels, click SSL Inbound Channel (SSL_2).

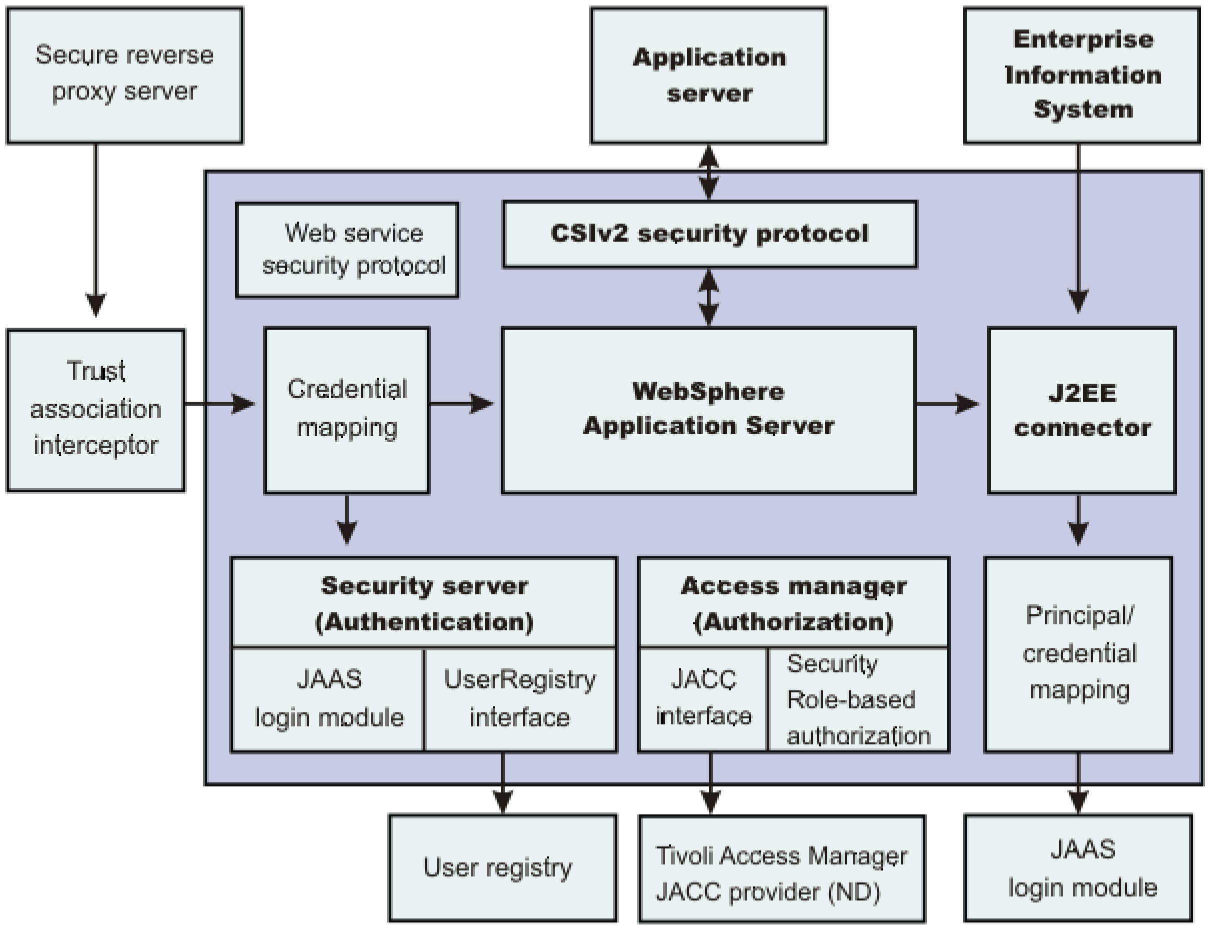
* Click System administration > Node agents > node_agent _name. Under Additional properties, click Ports. Locate a transport chain where SSL is enabled and click View associated transports. Click transport_channel_name. Under Transport Channels, click SSL Inbound Channel (SSL_2).

For the Object Request Broker (ORB) SSL transports, you can modify the SSL configuration repertoire aliases in the following locations. These configurations are for the server-level for WebSphere Application Server and WebSphere Application Server Express and the cell level for WebSphere Application Server Network Deployment.

* Click Security > Global security. Under RMI/IIOP security, click CSIv2 inbound communications.

* Click Security > Global security. Under RMI/IIOP security, click CSIv2 outbound communications.

For the Lightweight Directory Access Protocol (LDAP) SSL transport, you can modify the SSL configuration repertoire aliases by clicking Security > Global security. Under User account repository, click the Available realm definitions drop-down list, and select Standalone LDAP registry.


Secure reverse proxy server
Application server
Enterprise Information System
Web service security protocol
CSIv2 security protocol
Trust association interceptor
Credential mapping
WebSphere Application Server
J2EE connector
Security server (Authentication)
JAAS login module
UserRegistry interface
Access manager (Authorization)
JACC interface
Security Role-based authorization
Principal/ credential mapping
User registry
Tivoli Access Manager JACC provider (ND)
JAAS login module

B) Configure SIBus security.

Securing buses

Securing a service integration bus provides the bus with an authorization policy to prevent unauthorized users from gaining access. If a bus is configured to use multiple security domains, the bus also has a security domain and user realm to further enforce its authorization policy.

Before you begin

* If administrative security is not enabled for the cell that hosts the bus, you must enable it. The tasks below use an administrative console wizard that detects if administrative security is not enabled, and takes you through the steps to enable it. You must supply the type of user repository used by the server, and the administrative security username and password.

* If the bus contains a bus member at WebSphere® Application Server Version 6, you must provide an inter-engine authentication alias to establish trust between bus members, and to enable the bus to operate securely. The administrative console wizard detects whether an inter-engine authentication alias is required, and prompts you to supply one. If you want to specify a new inter-engine authentication alias, you must provide a user name and password.

About this task

When you secure a bus, consider the following points:

* If you are securing a WebSphere Application Server Version 7.0 bus that contains only Version 7.0 bus members, you can use a non-global security domain for the bus. If the bus has a WebSphere Application Server Version 6 bus member, or might have a Version 6 bus member in the future, you must assign the bus to the global security domain.

* If you want to assign the bus to a custom domain, you can select an existing security domain, or create a new one.

* If you assign the bus to a custom domain, you must specify a user realm. You can select an existing user realm, or use the global user realm.

What to do next

* The bus is secured after you restart all the servers that are members of the bus, or (for a bus that has bootstrap members) servers for which the SIB service is enabled.

* Use the administrative console to control access to the bus by administering users and groups in the bus connector role.

Securing links between messaging engines

For a mixed-version bus, when security is enabled, you must define an inter-engine authentication alias so that the messaging engines can establish trust.

Before you begin

Ensure that the user ID that you intend to use for the inter-engine authentication alias meets the following conditions:

* It exists in the user registry.

* It is used only for messaging engine to messaging engine authentication.

* It has not been added to the bus connector access role.

If you have a secure bus where all bus members are at Version 7.0, trust between Version 7.0

messaging engines is established by using a Lightweight Third Party Authentication (LTPA) token, and you do not need to perform this task.

About this task

If you have a secure, mixed-version bus, you must define an inter-engine authentication alias to prevent unauthorized messaging engines from establishing a connection. Messaging engines use the inter-engine authentication alias to establish trust in the following scenarios:

* A WebSphere® Application Server Version 6 messaging engine initiates a link with a Version 7.0 messaging engine.
* A Version 7.0 messaging engine initiates a link with a Version 6 messaging engine.

If you add a server or cluster as a bus member, if that action creates a mixed-version bus, you define an inter-engine authentication alias during that task, and you do not need to perform this task.

Procedure

1. In the navigation pane, click Service integration > Buses > security_value. The bus security configuration panel for the corresponding bus is displayed.
2. In the Inter-engine authentication alias field, select an authentication alias.
3. Click OK.
4. Save your changes to the master configuration.

Results

You have selected an inter-engine authentication alias for the bus to use in establishing trust between mixed-version messaging engines.

Service integration security planning

When you are planning the security of your messaging system, the range of options available to you can be described through a set of frequently asked questions.

These are some of the questions you might have when you start planning for messaging security:

* How do I secure the bus?
* Who has authority to access the bus?
* Who has authority to access bus destinations?
* Which connections must I secure?
* Which user IDs are stored in messages that flow between the bus and any foreign buses?
* What level of data store security do I need?

When considering these component interactions in relation to their security requirements, the following bus components can have security applied:

Client connections to the messaging engine are secured using role-based authorization.

Communication channels used by remote clients are secured using SSL.

The data store is secured using a J2C authentication alias.

Bus destinations are secured using role-based authorization.

Connections to the bus and the bus destinations are the access control points of the bus when security is enabled.

The bus uses a role-based authorization model. Users and groups can be added to the bus configuration under one of the predefined bus roles that define the access permissions being granted to the added user/group.

Tip: Remember to simplify management of security. A leading practice is to grant access to user groups rather than individual users.

The roles of the bus are summarized as the following by the information center.

Connector role: Grants the user or group permission to connect to the local bus.

Sender role: Grants the user or group the permission to send a message to a destination.

Receiver role: Grants the user or group the permission to receive a message from a destination.

Browser role: Grants the user or group the permission to browse messages on a destination.

Creator role: When temporary destinations are being used the creator role allows the user to create the temporary destination.

Pre-configuration requirements

Before securing the bus environment, note the following items:

Configuring bus security requires that administrative security is active.

If activating security on a bus that is in use, it is important to stop the bus and ensure that there are no in-doubt transactions relating to the existing message engines. Transaction recovery for these in-doubt transactions will not be able to successfully complete once security is enabled.

C) Define and implement administrative security roles.

Administrative roles

Monitor - An individual or group that uses the monitor role has the least amount of privileges. A monitor can complete the following tasks:

* View the WebSphere Application Server configuration.

* View the current state of the Application Server.

Configurator - An individual or group that uses the configurator role has the monitor privilege plus the ability to change the WebSphere Application Server configuration. The configurator can perform all the day-to-day configuration tasks. For example, a configurator can complete the following tasks:

* Create a resource.

* Map an application server.

* Install and uninstall an application.

* Deploy an application.

* Assign users and groups-to-role mapping for applications.

* Set up Java 2 security permissions for applications.

* Customize the Common Secure Interoperability Version 2 (CSIv2), Secure Authentication Service (SAS), and Secure Sockets Layer (SSL) configurations.

  Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Operator - An individual or group that uses the operator role has monitor privileges plus ability to change the runtime state. For example, an operator can complete the following tasks:

* Stop and start the server.

* Monitor the server status in the administrative console.

Administrator - An individual or group that uses the administrator role has the operator and configurator privileges plus additional privileges that are granted solely to the administrator role. For example, an administrator can complete the following tasks:

* Modify the server user ID and password.

* Configure authentication and authorization mechanisms.

* Enable or disable administrative security.

  Note: In previous releases of WebSphere Application Server, the Enable administrative security option is known as the Enable global security option.

* Enforce Java 2 security using the Use Java 2 security to restrict application access to local resources option.

* Change the Lightweight Third Party Authentication (LTPA) password and generate keys.

* Create, update, or delete users in the federated repositories configuration.

* Create, update, or delete groups in the federated repositories configuration.

Note: An administrator cannot map users and groups to the administrator roles.

Adminsecuritymanager - Only users who are granted this role can map users to administrative roles. Also, when fine-grained administrative security is used, only users who are granted this role can manage authorization groups. See Administrative roles for more information.

Deployer - Users who are granted this role can perform both configuration actions and runtime operations on applications.

Auditor - Users granted this role can view and modify the configuration settings for the security auditing subsystem. For example, a user with the auditor role can complete the following tasks:

* Enable and disable the security auditing subsystem.

* Select the event factory implementation to be used with the event factory plug-in point.

* Select and configure the service provide, or emitter. or both to be used with the service provider plug-in point.

* Set the audit policy that describes the behavior of the application server in the event of an error with the security auditing subsystem.

* Define which security events are to be audited.

The auditor role includes the monitor role. This allows the auditor to view but not change the rest of the security configuration.

This table lists an additional administrative role that is available through the administrative console.

Iscadmins - This role is only available for administrative console users and not for wsadmin users. Users who are granted this role have administrator privileges for managing users and groups in the federated respositories. For example, a user of the iscadmins role can complete the following tasks:

* Create, update, or delete users in the federated repositories configuration.
* Create, update, or delete groups in the federated repositories configuration.

This table lists an additional administrative role that is available through the administrative console.

Deployer - This role is only available for wsadmin users and not for administrative console users. Users who are granted this role can perform both configuration actions and run-time operations on applications

Internal server ID

The internal server ID enables the automatic generation of the user identity for server-to-server authentication. Automatic generation of the server identity supports improved auditability for cells only for Version 6.1 or later nodes. In the Version 6.1 release of WebSphere Application Server, you can save the internally-generated server ID because the Security WebSphere Common Configuration Model (WCCM) model contains a new tag, internalServerId. You do not need to specify a server user ID and a password during security configuration except in a mixed-cell environment. An internally-generated server ID adds a further level of protection to the server environment because the server password is not exposed as it is in releases prior to Version 6.1. However, to maintain backwards compatibility, you must specify the server user ID if you use earlier versions of WebSphere Application Server.

D) Configure SSL clients and servers.

Configuring JMS client applications to perform client SSL authentication

You can configure JMS client applications to authenticate to the bus by using client Secure Sockets Layer (SSL) authentication.

Before you begin

* You have already obtained a Secure Sockets Layer (SSL) certificate for the JMS client application.
* The JMS client application is already configured to use SSL. For more information, see SSL client properties file

This task has two objectives. First, you install the SSL certificate for the client application in the key store for the application client. Secondly, you modify the sib.client.ssl.properties file to use client SSL authentication. You use the Key Management (iKeyman) utility to work with SSL certificates. The iKeyman user interface is Java-based and uses the Java™ support that is installed with IBM® HTTP Server.

Take the following steps to configure a JMS client application to use client SSL authentication:

Procedure

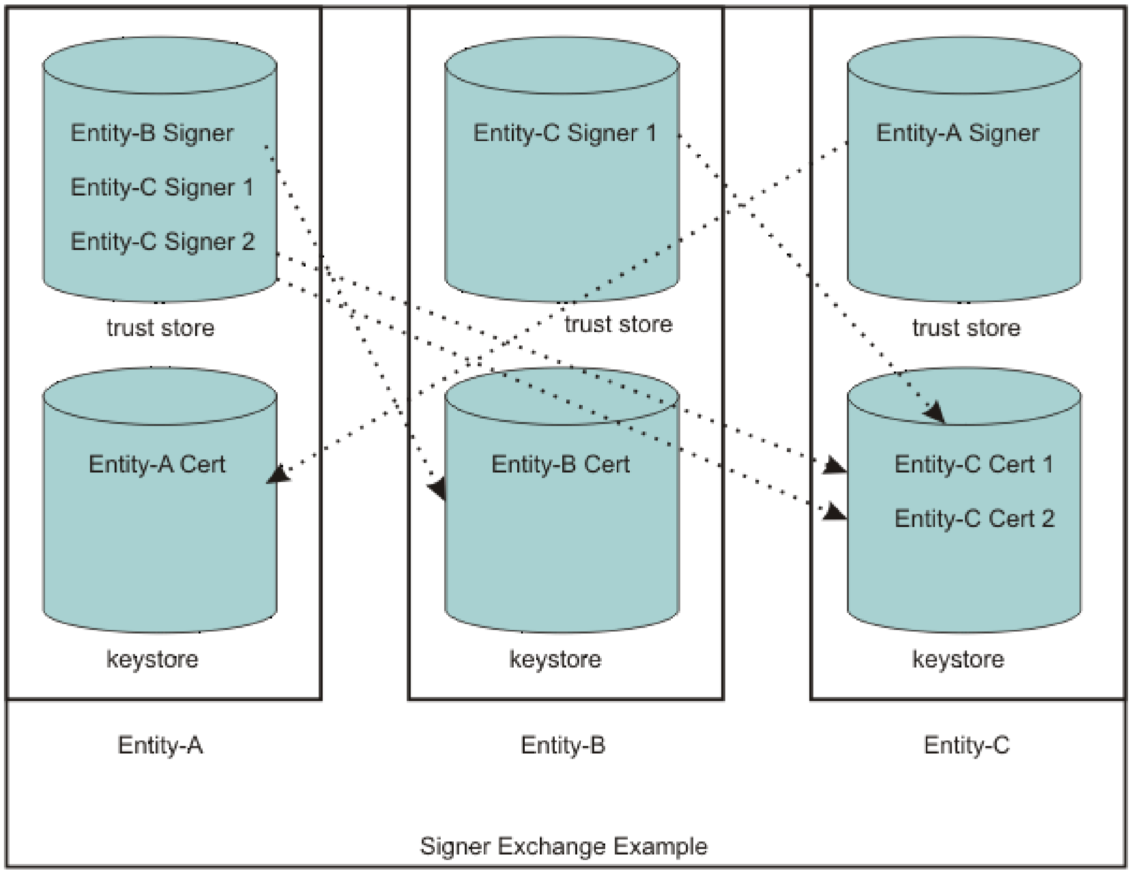
1. Start the iKeyman user interface. Refer to the iKeyman User Guide available from IBM developer kits for more information about using iKeyman.
2. When prompted, select the key store for the JMS client application.
3. When prompted for the type off certificate to work with, select the option Personal certificates. A list of personal certificates is displayed.
4. Select that you want to import a certificate to the selected key store.
5. When prompted, type the location and name for the certificate. You can provide an alias for the certificate. The certificate is installed into the keystore of the client application.
6. Close the iKeyman user interface.
7. Open a text editor to work with the sib.client.ssl.properties properties file. This file is located in the profile_root/properties directory of the application server installation, where profile_root is the directory in which profile-specific information is stored.
8. Set the value for the property com.ibm.ssl.client.clientAuthentication to True.
9. Set the value for the property com.ibm.ssl.client.keyStoreClientAlias to the alias name for the certificate in the client key store.
10. Save the sib.client.ssl.properties properties file.

Results

You have now configured a JMS client application to use client SSL authentication.

Signer exchange

When you configure an SSL connection, you can exchange signers to establish trust in a personal certificate for a specific entity. Signer exchange enables you to extract the X.509 certificate from the peer keystore and add it into the truststore of another entity so that the two peer entities can connect. The signer certificate also can originate from a CA as a root signer certificate or a chained certificate's root signer certificate or an intermediate signer certificate. You can also extract a signer certificate directly from a self-signed certificate, which is the X.509 certificate with the public key.



<u>Configuring a bus to allow client SSL authentication</u>

You can configure a service integration bus to enable connecting client JMS applications to authenticate by using Secure Sockets Layer (SSL) certificates.

Before you begin

You must ensure that the following tasks have been completed:

* Administrative security is enabled.
* A standalone Lightweight Directory Access Protocol (LDAP) user registry has been configured for storing user and group IDs. To access the user registry, you must know a valid user ID that has the administrative role, and password, the server host and port of the registry server, and the base distinguished name (DN).
* Bus security is enabled.
* JMS client applications have been configured to authenticate by using client SSL certificates.

About this task

If you want to allow connecting JMS application clients to authenticate to the bus by using client SSL certificates, define an SSL configuration. There are two parts to this task. First you use the administrative console to map SSL certificates to entries in the LDAP user registry. Secondly, you create a unique SSL configuration for each endpoint address for which you want to use client SSL authentication. Do not use the default SSL configuration for the bus.

Procedure

1. Use the administrative console to define certificate filters to map an SSL certificate to an entry in the LDAP server. The client SSL certificate is mapped to a user ID in the user registry.
2. Create a separate SSL configuration file for each endpoint address for server, bus member or cluster on the bus, and select that client authentication is required.

Results

The bus is configured to allow client SSL authentication.

What to do next

Connecting JMS client applications can now authenticate to the bus using client SSL certificates.

The attributes defining an SSL configuration repertoire entry for a specific management scope are stored in the security.xml file.

<u>ssl.client.props client configuration file</u>

Use the ssl.client.props file to configure Secure Sockets Layer (SSL) for clients. In previous releases of WebSphere® Application Server, SSL properties were specified in the sas.client.props or soap.client.props files or as system properties. By consolidating the configurations, WebSphere Application Server enables you to manage security in a manner that is comparable to server-side configuration management. You can configure the ssl.client.props file with multiple SSL configurations.

<u>SSL configurations</u>

An SSL configuration comprises a set of configuration attributes that you can associate with an endpoint or set of endpoints in the WebSphere Application Server topology. The SSL configuration enables you to create an SSLContext object, which is the fundamental JSSE object that the server uses to obtain SSL socket factories. You can manage the following configuration attributes:

* An alias for the SSLContext object
* A handshake protocol version
* A keystore reference
* A truststore reference
* A key manager
* One or more trust managers
* A security level selection of a cipher suite grouping or a specific cipher suite list
* A certificate alias choice for client and server connections

E) Implement federated repositories.

Federated repositories enable you to use multiple repositories with WebSphere® Application Server. These repositories, which can be file-based repositories, LDAP repositories, or a sub-tree of an LDAP repository, are defined and theoretically combined under a single realm. All of the user repositories that are configured under the federated repository functionality are invisible to WebSphere Application Server.

When you use the federated repositories functionality, all of the configured repositories, which you specify as part of the federated repository configuration, become active. It is required that the user ID, and the distinguished name (DN) for an LDAP repository, be unique in multiple user repositories that are configured under the same federated repository configuration. For example, there might be three different repositories that are configured for the federated repositories configuration: Repository A, Repository B, and Repository C. When user1 logs in, the federated repository adapter searches each of the repositories for all of the occurrences of that user. If multiple instances of that user are found in the combined repositories, an error message displays.

In addition, the federated repositories functionality in WebSphere Application Server supports the logical joining of entries across multiple user repositories when the Application Server searches and retrieves entries from the repositories. For example, when an application calls for a sorted list of people whose age is greater than twenty, WebSphere Application searches all of the repositories in the federated repositories configuration. The results are combined and sorted before the Application Server returns the results to the application.

Unlike the local operating system, standalone LDAP registry, or custom registry options, federated repositories provide user and group management with read and write capabilities. When you configure federated repositories, you can use one of the following methods to add, create, and delete users and groups:

Important: If you configure multiple repositories under the federated repositories realm, you must also configure supported entity types and specify a base entry for the default parent. The base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management. See Configuring supported entity types in a federated repository configuration for details.

> * Use the user management application programming interfaces (API). For more information, refer to articles under "Developing with virtual member manager" in this information center.
>
> * Use the administrative console. To manage users and groups within the administrative console, click Users and Groups > Manage Users or Users and Groups > Manage Groups. For information on user and group management, click the Help link that displays in the upper right corner of the window. From the left navigation pane, click Users and Groups.
>
> * Use the wsadmin commands. For more information, see the WIMManagementCommands command group for the AdminTask object topic.

If you do not configure the federated repositories functionality or do not enable federated repositories as the active repository, you cannot use the user management capabilities that are associated with federated repositories. You can configure an LDAP server as the active user registry and configure the same LDAP server under federated repositories, but not select federated repositories as the active user repository. With this scenario, authentication takes place using the LDAP server, and you can use the user management functionality for the LDAP server that is available for federated repositories.

Configuring federated repositories in a mixed-version environment

In a mixed-version deployment manager cell that contains both Version 6.1.x and Version 5.x or 6.0.x nodes, the following limitations apply for configuring federated repositories:

> * You can configure only one Lightweight Directory Access Protocol (LDAP) repository under federated repositories, and the repository must be supported by Version 5.x or 6.0.x.
>
> * You can specify a realm name that is compatible with prior versions only. The host name and the port number represent the realm for the LDAP server in a mixed-version nodes cell. For example, machine1.austin.ibm.com:389.

* You must configure a stand-alone LDAP registry; the LDAP information in both the stand-alone LDAP registry and the LDAP repository under the federated repositories configuration must match. During node synchronization, the LDAP information from the stand-alone LDAP registry propagates to the Version 5.x or 6.0.x nodes.

Important: Before node synchronization, verify that Federated repositories is identified in the Current® realm definition field. If Federated repositories is not identified, select Federated repositories from the Available realm definitions field and click Set as current. Do not set the stand-alone LDAP registry as the current realm definition.

* You cannot configure an entry mapping repository or a property extension repository in a mixed-version deployment manager cell.

Configuring LDAP servers in a federated repository

The LDAP connection connectTimeout default value is 20 seconds. LDAP should respond within 20 seconds for any request from WebSphere® Application Server. If you cannot connect to your LDAP within this time, make sure that your LDAP is running. A connection error displays at the top of the LDAP configuration panel when the connection timeout exceeds 20 seconds.

Coexisting with Tivoli Access Manager

For Tivoli Access Manager to coexist with a federated repositories configuration, the following limitations apply:

* You can configure only one LDAP repository under federated repositories, and that LDAP repository configuration must match the LDAP server configuration under Tivoli Access Manager.

* The distinguished name for the realm base entry must match the LDAP distinguished name (DN) of the base entry within the repository. In WebSphere Application Server, Tivoli Access Manager recognizes the LDAP user ID and LDAP DN for both authentication and authorization. The federated repositories configuration does not include additional mappings for the LDAP user ID and DN.

* The federated repositories functionality does not recognize the metadata that is specified by Tivoli Access Manager. When users and groups are created under user and group management, they are not formatted using the Tivoli Access Manager metadata. The users and groups must be manually imported into Tivoli Access Manager before you use them for authentication and authorization.

Limitation for changing the realm name for federated repositories in a multiple security domain environment

When you configure a multiple security domain for IBM® WebSphere Application Server Version 7.0, you must configure the realm name for a federated repository before you assign the federated repository to any domains.

After you assign the federated repository to a security domain, you cannot change its realm name using the administrative console because the change only reflects in the global security.xml file and not in the domain-security.xml file. This situation results in two different realm names that are used by the same registry.

If you must change the realm name for the federated repository after it has been assigned to a security domain, use the updateIdMgrRealm and configureAppWIMUserRegistry commands to change the realm name in the domain-security.xml file.

Limitation for configuring active directories with their own federated repository realms

In order to use the administrative console to perform a wildcard search for all available users on two Active Directories, and to prevent multiple entries exceptions with all built-in IDs, you must first configure each Active Directory with it's own federated repository realm.

However, you cannot use the administrative console to configure each Active Directory with it's own federated repository realm. You can instead use a wsadmin script similar to the following:

```
$AdminTask createIdMgrRealm {-name AD1realm}
$AdminTask addIdMgrRealmBaseEntry {-name AD1realm -baseEntry o=AD1}

$AdminTask createIdMgrRealm {-name AD2realm}
$AdminTask addIdMgrRealmBaseEntry {-name AD2realm -baseEntry o=AD2}

$AdminConfig save
```

Increasing the performance of the federated repository configuration

Follow this page to manage the realm in a federated repository configuration.

Before you begin

The settings that are available on the Performance panel are independent options that pertain specifically to the federated repositories functionality. These options do not affect your entire WebSphere® Application Server configuration.

Procedure

1. In the administrative console, click Security > Global security.

2. Under User account repository, select Federated repositories from the Available realm definitions field and click Configure.

3. Under Related items, click Manage repositories > repository_name.

4. Under Additional properties, click Performance.

5. Optional: Select the Limit search time option and enter the maximum number of milliseconds that the Application Server can use to search through your Lightweight Directory Access Protocol (LDAP) entries.

6. Optional: Select the Limit search returns option and enter the maximum number of entries to return that match the search criteria.

7. Optional: Select the Use connection pooling option to specify whether the Application Server can store separate connections to the LDAP server for reuse.

8. Optional: Select the Enable context pool option to specify whether multiple applications can use the same connection to the LDAP server. If you select the option, specify the initial, preferred, and maximum number of entries that can use the same connection. The Enable context pool option can be enabled either in conjunction with the Use connection pool option or separately. If this option is disabled, a new connection is created for each context. You can also select the Context pool times out option and specify the number of seconds after which the entries in the context pool expire.

9. Optional: Set the Maximum size value of the context pool to zero (0).

10. Optional: Select the Cache the attributes option and specify the maximum number of search attribute entries. This option enables WebSphere Application Server to save the LDAP entries so that it can search the entries locally rather than making multiple calls to the LDAP server. Click the Cache times out option that is associated with the Cache the attributes option to specify the maximum number of seconds that the Application Server can save these entries.

11. Optional: Select the Cache the search results option and specify the maximum number of search result entries. This option enables WebSphere Application Server to save the results of a search inquiry instead of making multiple calls to the LDAP server to search and retrieve the results of that search. Click the Cache times out option that is associated with the Cache the search results option to specify the maximum number of seconds that the Application Server can save the results.

12. Optional: Create the root DataObject object locally using the com.ibm.websphere.wim.util.SDOHelper.createRootDataObject method instead of the com.ibm.websphere.wim.ServiceProvider.createRootDataObject method.

Results

These options are available to potentially increase the performance of your federated repositories configuration. However, any increase in performance is dependent upon your specific configuration.

<u>Using custom adapters for federated repositories</u>

When the custom adapters for federated repositories are part of the default realm, the users and groups can be managed using wsadmin commands or the administrative console.

About this task

If custom adapters for federated repositories are part of the default realm, you use the administrative console to manage the users and groups in the realm.

Note: The default parent for PersonAccount and Group entities needs to be the same as the base entry of the custom adapter.

To view this administrative console page, complete the following steps:

* In the administrative console, click Security > Global security.

* Under User account repository, select Federated repositories from the Available realm definitions field and click Configure.

* Under Additional properties, click Supported entity types.

You must configure the supported entity types before you can manage this account with Users and Groups in the administrative console. The Base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management.

Procedure

1. In the administrative console, click Users and Groups to access users and groups panel.

2. Click Manage Groups to test the basic functions of the custom adapter with respect to custom adapters for federated repositories.

3. Click Manage Users to test the basic functions of the custom adapter with respect to custom adapters for federated repositories.

Results

After completing these steps, you will have ensured that the custom adapter is being used properly.

F) Configure WebSphere Application Server Network Deployment V7.0 with multiple security domains.

Multiple security domains

The WebSphere® Security Domains (WSD) provide the flexibility to use different security configurations in WebSphere Application Server. The WSD is also referred to as multiple security domains, or simply, security domains. You can configure different security attributes, such as the UserRegistry, for different applications.

New feature: Multiple security domain support is new in this release of WebSphere Application Server. You can create different security configurations and assign them to different applications in WebSphere Application Server processes. By creating multiple security domains, you can configure different security attributes for both administrative and user applications within a cell environment. You can configure different applications to use different security configurations by assigning the servers or clusters or service integration buses that host these applications to the security domains. Only users assigned to the administrator role can configure multiple security domains.

In previous versions of WebSphere Application Server, all administrative and user applications use security attributes different from those attributes that are defined in global security. All administrative and user applications in WebSphere Application Server use global security attributes by default. For example, a user registry defined in global security is used to authenticate a user for every application in the cell.

In this release of WebSphere Application Server, however, you can use multiple security attributes for user applications other than the global security attributes, create a security domain for those security attributes that must differ, and associate them with the servers and clusters that host those user applications. You also can associate a security domain with the cell. All of the user applications in the cell use this security domain if they do not have a domain previously associated with them. However, global security attributes are still required for administrative applications such as the administrative console, naming resources and MBeans.

Global Security applies to all administrative functions and the default security configuration for user applications. Security domains can be used to define a customized configuration for user applications.

You must have a global security configuration defined before you can create security domains. The global security configuration is used by all of the administrative applications such as the administrative console, naming resources, and Mbeans. If no security domains are configured, all of the applications use information from the global security configuration. User applications such as Enterprise JavaBeans™ (EJBs), servlets and administrative applications use the same security configuration.

When you create a security domain and associate it with a scope, only the user applications in that scope use the security attributes that are defined in the security domain. The administrative applications as well as the naming operations in that scope use the global security configuration. Define the security attributes at the domain level that need to be different from those at the global level. If the information is common, the security domain does not need to have the information duplicated in it. Any attributes that are missing in the domain are obtained from the global configuration. The global security configuration data is stored in the security.xml file, which is located in the $WAS_HOME/profiles/$ProfileName/cells/$CellName directory.

The following figure provides an example of a security multiple domain where the cell, a server and a cluster are associated with different security domains. As shown in the figure, the user applications in server S1.1 as well as the cluster use security attributes that are defined in Domain2 and Domain3 respectively (since these scopes are associated with these domains). Server S2.2 is not associated with a domain. As a result, the user application in S2.2 uses the domain that is associated with the cell (Domain1) by default . Security attributes that are missing from the domain level are obtained from the global configuration.

Contents of a security domain

A security domain is represented by two configuration files. One configuration file contains the list of attributes that are configured in the security domain. The other configuration file contains the scopes that use the security domain. The security domain information is stored in the $WAS_HOME/profiles/$ProfileName/config/waspolicies/default/securitydomains/ $SecurityDomainName directory. For every security domain that is configured, a $SecurityDomainName directory is created with two files in it: the security-domain.xml file contains the list of security attributes configured for the security domain, and the security-domain-map.xml file contains the scopes that use the security domain.

When you create a security domain you must supply a unique name for the domain, the security attributes you want to configure for the security domain, and the scopes that need to use the security domain. Once configured, the servers that use the security domain must be restarted. The user applications in those scopes then use the attributes that are defined in the security domain. Any attributes that are not configured at the domain level are obtained from the global security configuration. Administrative applications and naming operations in the scopes always use the security attributes from the global security configuration. You must actively manage these attributes.

Any new security domain attributes must be compatible with those global security attributes that are inherited by the user applications that are assigned to the domain.

A scope (a server, cluster, service integration bus or a cell) can be associated with only one domain. For example, you cannot define two domains that both have the cell-wide scope. Multiple scopes, however, can be defined in the same security domain. For example, a domain can be scoped to Server1 and to Server2 only within the cell.

The assigned scopes section on the security domain panel displays two views: one view that enables you to select and assign scopes to the domain, and another view that enables you to see a list of the currently assigned scopes. For convenience, you also have the flexibility to copy all of the security attributes from an existing security domain or the global configuration into a new security domain, and then modify only those attributes that must be different. You must still associate the scopes to these copied domains.

When a security domain is associated with a server that is not part of a cluster, all user applications in that server use the attributes from the security domain. Any missing security attributes are obtained from the global security configuration. If the server is part of a cluster, you can associate the security domain with the cluster but not with the individual members in that cluster. The security behavior then remains consistent across all of the cluster members

The migration tool migrates the existing server level security configuration information to the new security domain configuration when the script compatibility mode is false (-scriptCompatibility="false"). A new security domain is created for every server security configuration if it is not part of a cluster. If it is part of a cluster, a security domain is associated with the cluster instead of with all of the servers in that cluster. In both cases, all of the security attributes that were configured at the server level in previous releases are migrated to the new security domain configuration, and the appropriate scope is assigned to the security domains.

If the script compatibility mode is set to true, the server level security configuration is not migrated to the new security domains configuration. The old server security configuration is migrated without any changes. The security runtime detects that the old security configuration exists and uses that information, even if a security domain is associated either directly or indirectly to the server. If the script compatibility mode is set to true, remove the security configuration from the server level and then create a security domain with the same set of security attributes.

How domain level security attributes are used by security runtime and applications

This section describes how the individual attributes at the domain level are used by the security runtime and how that impacts the user application security. Since all of these security attributes are also defined at the global level, more information about these attributes can be obtained elsewhere. For the purposes of this section, the emphasis is on domain level behavior.

1. Application Security:

Select Enable application security to enable or disable security for user applications. When this selection is disabled, all of the EJBs and Web applications in the security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and Web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

2. Java 2 Security:

Select Use Java 2 security to enable or disable Java 2 security at the domain level or to assign or add properties related to Java 2 security. This choice enables or disables Java 2 security at the process (JVM) level so that all applications (both administrative and user) can enable or disable Java 2 security.

3. User Realm (User Registry):

This section enables you to configure the user registry for the security domain. You can separately configure any registry except the federated registry that is used at the domain level. The federated repository can only be configured at the global level but can be used at the domain level. Read about Configuring attributes for security domains for more information.

When configuring a registry at the domain level you can choose to define your own realm name for the registry. The realm name distinguishes one user registry from another. The realm name is used in multiple places – in the Java client login panel to prompt the user, in the authentication cache, and when using native authorization.

At the global configuration level, the system creates the realm for the user registry. In previous releases of WebSphere Application Server, only one user registry is configured in the system. When you have multiple security domains you can configure multiple registries in the system. For the realms to be unique in these domains, configure your own realm name for a security domain. You also can choose the system to create a unique realm name if it is certain to be unique. In the latter case, the realm name is based on the registry that is being used.

For LDAP registries, the host:port of the LDAP server is the system-generated realm name. For localOS, the name of the localOS machine is the realm name. For custom user registries, the realm is the one returned by the getRealm ( ) method of the custom registry implementation.

If the system generated realm names are unique enough, you can choose the option for the system to generate the realm name. If not, choose a unique realm name for each security domain where you have the user registry configured. If the underlying user repository is the same, use the same realm name in different domains. From a security runtime perspective, same realm names have the same set of users and groups information. For example, when users and groups information is required from a realm, the first user repository that matches the realm is used.

If a localOS registry that is not centralized is configured for any domain, and that domain is associated with servers or clusters in nodes not on the same system as the deployment manager, the realm name has to be provided. This realm name has to be the same as it would be if it were generated on the node. This realm name can be obtained by calling the getRealm() method on the SecurityAdmin MBean on that node. Typically, the realm name for localOS registries is the hostname of the machine. In this case, you should not let the system generate the realm name but rather get the realm name that is used by the processes in the node.

If you select the system to generate the realm for the localOS registry at the time of the user registry configuration, it chooses the localOS registry that is used by the deployment manager. If the realm configured does not match the realm used by the servers then there are authorization issues. Also note that in this case, the domain using this local registry can only be associated with servers and clusters that belong to nodes on the same machine.

As mentioned earlier, federated repositories user registry can only be configured at the global level but any domain can use it by configuring it as the active registry. In this case, if the default realm name needs to be changed it has to be changed at the global level.

When more than one user registry is in a process, the naming lookup that uses “UserRegistry” as the lookup name returns the user registry that is used by user applications. The user registry used by administrative applications is bound by the lookup name, “AdminUserRegistry”.

As described in Cross realm communication, when an application in one realm communicates with an application in another realm using LTPA tokens, the realms have to be trusted. The trust relationship can be established using the Trusted authentication realms – inbound link in the user registry panel or by using the addTrustedRealms command. You can establish trust between different realms. A user logged into one realm can access resources in another realm. If no trust is established between the two realms the LTPA token validation fails.

Note: The realm name used in the web.xml file is not related to the user registry realm.

4. Trust Association:

When you configure the trust association interceptor (TAI) at a domain level, the interceptors configured at the global level are copied to the domain level for convenience. You can modify the interceptor list at the domain level to fit your needs. Only configure those interceptors that are to be used at the domain level.

Tivoli Access Manager’s trust association interceptors can only be configured at the global level. The domain configuration can also use them, but cannot have a different version of the trust association interceptor. Only one instance of Tivoli Access Manager’s trust association interceptors can exist in the cell.

5. SPNEGO Web Authentication:

The SPNEGO Web authentication, which enables you to configure SPNEGO for Web resource authentication, can be configured at the domain level.

Note: In WebSphere Application Server Version 6.1, a TAI that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO Web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

6. RMI/IIOP Security (CSIv2):

The RMI/IIOP security attribute refers to the CSIv2 (Common Secure Interoperability version 2) protocol properties. When you configure these attributes at the domain level, the RMI/IIOP security configuration at the global level is copied for convenience.

You can change the attributes that need to be different at the domain level. The Transport layer settings for CSIv2 inbound communications should be the same for both the global and the domain levels. If they are different, the domain level attributes are applied to all of the application in the process.

When a process communicates with another process with a different realm, the LTPA authentication and the propagation tokens are not propagated to the downstream server unless that server is listed in the outbound trusted realms list. This can be done using the Trusted authentication realms – outbound link on the CSIv2 outbound communication panel, or by using the addTrustedRealms command task. Read about Cross realm communication for more information.

7. JAAS (Java Authentication and Authorization Service):

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. By default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

For JAAS and custom properties only, once global attributes are customized for a domain they can still be used by user applications.

8. Authentication Mechanism Attributes:

Specifies the various cache settings that must be applied at the domain level.

1. Authentication cache settings - use to specify your authentication cache settings. The configuration specified on this panel is applied only to this domain.

2. LTPA Timeout - You can configure a different LTPA timeout value at the domain level. The default timeout value is 120 minutes, which is set at the global level. If the LTPA timeout is set at the domain level, any token that is created in the security domain when accessing user applications is created with this expiration time.

3. Use realm-qualified user names - When this selection is enabled, user names returned by methods such as getUserPrincipal( ) are qualified with the security realm (user registry) used by applications in the security domain.

9. Authorization Provider:

You can configure an external third party JACC (Java Authorization Contract for Containers) provider at the domain level. Tivoli Access Manager’s JACC provider can only be configured at the global level. Security domains can still use it if they do not override the authorization provider with another JACC provider.

The JACC attributes, for example the Policy object, are based at the JVM level. This implies that there can be only be one JACC policy object in a JVM process. However, when you have multiple JACC providers configured, the deployment manager process has to handle all these providers in the same JVM because it has to propagate the authorization policy of applications to the respective provider based on the application name.

If your JACC provider can handle propagating the authorization policy to multiple providers, you can configure it at the global level. In this case, when an application is installed, this JACC provider is called in the deployment manager process and it is the responsibility of this JACC provider to propagate the information to the corresponding JACC provider based on the application name passed in the contextID.

Another way to achieve this is to set the custom property, com.ibm.websphere.security.allowMultipleJaccProviders=true, at the global security level. When this property is set, WebSphere Application Server propagates the authorization policy information to the JACC provider associated with the domain that corresponds to the target server where the application is installed. This property is only used at the deployment manager process since the managed servers do not host multiple JACC providers.

Client and application security programming model when using security domains

A Java client or an application acting as a client that accesses an EJB typically does a naming lookup first. The naming resource, which is used by both administrative and the user applications, is considered an administrative resource. It is protected by the global security configuration information. In a multiple domain setup where the global security is using one realm (the user registry) and a domain is using a different realm, the Java client must authenticate to two different realms. The first authentication is required for the realm in the global security configuration for the naming operation to succeed, and the second authentication is required to access the EJB, which uses a different realm.

The CosNamingRead role protects all naming read operations. This role is usually assigned the Everyone special subject. This implies that any user, valid or not, can look up the name space. When a multiple domain is defined, if the CosNamingRead role has the Everyone special subject the security runtime code in the client side does not prompt you to log in. It uses the UNAUTHENTICATED subject to access the naming operation instead. Once the naming lookup operation is completed, when the client attempts to access the EJB it is prompted with a login panel that indicates the realm that is currently used by that EJB application (that is, the realm used in the domain). The client then presents the appropriate user credentials for that realm, which can then access the EJB. This logic applies to all variations of login source, including properties and stdin, not just when the login source is set to prompt.

If the Everyone special subject is removed from the CosNamingRead role, you are prompted twice. If the login source is properties, you can uncomment the com.ibm.CORBA.loginRealm property in the $WAS_HOME/profiles/$ProfileName/properties/sas.client.props file and add the appropriate realms using "|" as the separator. You must also enter the corresponding users and passwords in the com.ibm.CORBA.loginUserid and com.ibm.CORBA.loginPassword properties respectively. When you are using the programmatic logon in the Java client code you must authenticate twice with different user credentials; once prior to do a naming lookup for the EJB (the user should be in the global realm), and later prior to calling any method in the EJB (the user should be in the EJB domain's realm).

In general, when a Java client needs to authenticate to multiple and different realms it has to provide the credential information for all of those realms. If the login source is prompt or stdin it is prompted to login multiple times, once for each realm. If the login source is set to properties, the appropriate properties in the sas.client.props file (or any related file) are used for authenticating to different realms.

In certain scenarios, a client might make multiple calls to the same realm. For example, the Java client can access a resource using realm1 followed by access to a resource using realm2, and then come back to access a resource in realm1 again. In this case, the client is prompted three times; first for realm1, secondly for realm2 and finally for realm1 again.

By default, the subject that is used to login at a realm is not cached by the client side code. If you have this scenario, and you want the client to cache the subject based on the realm, set the com.ibm.CSI.isRealmSubjectLookupEnabled property to true in the sas.client.props file. If the com.ibm.CSI.isRealmSubjectLookupEnabled property is set, the client code caches the subject based on the realm name. The next time the Java client needs to authenticate to this realm, the cache is located to obtain the subject and the client is not prompted. Also, when the com.ibm.CSI.isRealmSubjectLookupEnabled property is set, the same subject that was logged in the first time is used for subsequent logins. If the subject information needs to change then this property should not be set.

If the client is doing a programmatic login it can pass the realm along with the user and password that it needs to authenticate. In this case, when the com.ibm.CORBA.validateBasicAuth property is set to true (the default value) in the sas.client.props file, the registry that matches the realm name is used for login. That realm must be supported in the process where the authentication takes place.

When using the WSLogin JAAS configurations, you also must set the use_realm_callback option in the wsjaas_client.config file in $WAS_HOME/profiles/$ProfileName/properties for the realm name to be passed to the call back handler. If you want to specify a different provider URL for the name server, set the use_appcontext_callback option and pass in the provider URL properties in a hash map to WSLogin.

If you do not know the realm name, use <default> as the realm name. The authentication is performed against the application realm. If the naming read operation does not have the Everyone special subject assigned, you must provide the realm that is used by the administrative applications (the registry used in the global security configuration), as well as the appropriate user and password information in that registry for the lookup operation to succeed.

After the lookup operation succeeds, perform another programmatic login by providing the application realm (or <default>) and the user and password information for the appropriate user in the registry that is used by the application. This is similar to the case where the login source is prompt. You must authenticate twice, once for the registry used by the global security configuration (for the naming lookup operation) and again for the registry used by the application to access the EJB.

If com.ibm.CORBA.validateBasicAuth is set to false in the $WAS_HOME/profiles/$ProfileName/properties/sas.client.props file then the programmatic login can use <default> as the realm name for both the lookup and the EJB operations. The actual authentication occurs only when the resource is accessed on the server side, in which case the realm is calculated based on the resource that is accessed.

The new security domain support for WebSphere Application Version 7.0 does not change the current application security programming model. However, it provides more flexibility and capabilities such as the following:

* User applications can still find the user registry object by using the naming lookup for "UserRegistry". For the registry object used by administrative applications, the naming lookup for "AdminUserRegistry" can be used.

* The application usage of the JAAS login configuration does not change in a multiple domain setup. However, if an application must refer to the JAAS configuration that is specified at the domain level, the administrator and the deployer of that application must make sure that this domain is configured with the JAAS configurations that are required by the application.

* If an application needs to communicate with other applications using different realms, trust relationship should be established for both inbound and outbound communications when using the LTPA tokens. Read about Cross realm communication for more information.

When using programmatic login in the applications, if you want to login to the realm used by the application, use <default> as the realm name or provide the realm name that the application is using. If you need to login to the global realm, you must provide the global realm name. If you provide any other realm, only a basic authentication subject is created. When the request actually flows to the server hosting that realm, the actual authentication of the user occurs if that server hosts the realm. If the server does not host the realm, the login fails.

Application deployment in multiple domains configurations

When deploying an application in a multiple domain setup, all of the modules in the application should be installed in the servers or clusters that belong to the same security domain. If not, depending on the security attributes configured in these security domains, inconsistent behavior can result. For example, if the domains contain different user registries, the users and groups information can be different, which can cause inconsistent behavior when accessing the modules. Another example is when the JAAS data is different between the security domains. The JAAS configurations is not accessible from all of the modules in the application. The security runtime code and the command tasks rely on one domain being associated with an application when dealing with attributes such as user registry, JAAS login configurations, J2C authentication data, and authorization.

In most cases, application deployment fails when an application is deployed across different domains. However, since this was possible in earlier releases of WebSphere Application Server when only a few attributes were supported at the server level, the deployment tool first checks for attributes that are configured at the domains. If the attributes in the domain are the same as those supported in previous releases, the administrative console requests confirmation to ensure that you want to deploy application modules across multiple security domains. Unless there is an absolute requirement to deploy the applications across different domains, stop the deployment and select the servers and clusters in the same security domain.

Cross realm communication

When applications communicate using the RMI/IIOP protocol and LTPA is the authentication mechanism, the LTPA token is passed between the servers involved. The LTPA token contains the realm-qualified uniqueId, (also called the accessId), of the user who is logging into the front-end application. When this token is received by the downstream server it attempts to decrypt the token. If the LTPA keys are shared between the two servers, decryption succeeds and the accessId of the user is obtained from the token. The realm in the accessId is checked with the current realm that is used by the application. If the realms match, the LTPA token validation succeeds and it proceeds with the authorization. If the realms do not match, the token validation fails since the user from the foreign realm cannot be validated in the current realm of the application. If applications are not supposed to communicate with each other when using RMI/IIOP and the LTPA authentication mechanism, you do not to have to do anything further.

If you do want the cross realm communication to succeed when using RMI/IIOP and LTPA tokens, you must first establish trust between the realms involved, both for inbound and outbound communications.

For the server originating the request, its realm must have the realms that it can trust to send the token to. This is referred to as outboundTrustedRealms. For the server receiving the request, its realm needs to trust the realms that it can receive LTPA tokens from. This is referred to as inboundTrustedRealms.

Outbound trusted realms can be established using the addTrustedRealms command with the –communicationType option set to outbound. It can also be established in the administrative console by clicking Trusted authentication realms - outbound on theCSIv2 outbound communications panel.

Inbound trusted realms can be established using the same addTrustedRealms command task with the –communicationType option set to inbound. It can also be established by using the administrative console.

The figure below shows the communication between applications that use different user realms (registries) using RMI/IIOP. In this example, application app1 (for example, a servlet) is configured to use the realm1 user registry. The app2 application (for example, an EJB) is configured to use the realm2 user registry. The user (user1) initially logs into the servlet in app1, which then attempts to access an EJB in app2. The following must be set:

* In Domain1, realm1 should trust realm2 for the outbound communication.

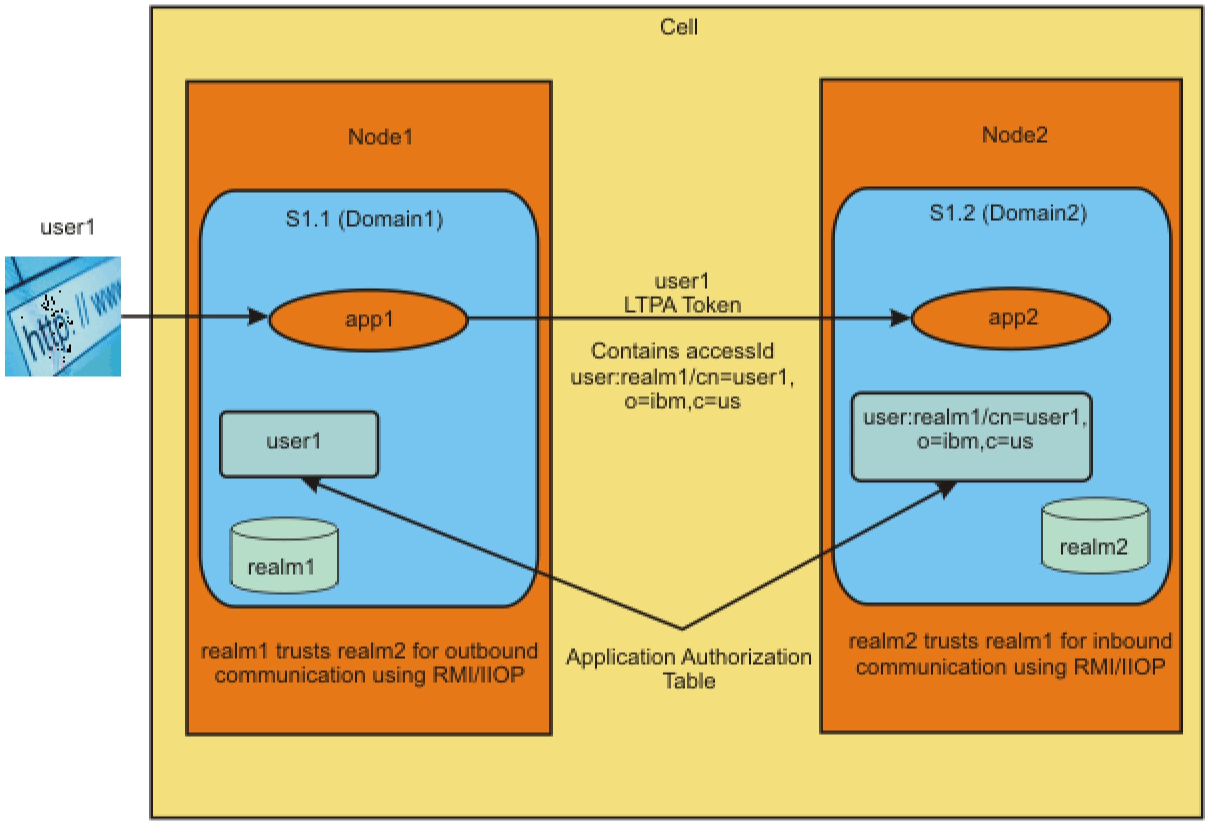
* In Domain2, realm2 should trust realm1 for the inbound communication.

* The accessId for user1 should be configured in the authorization table for app2.

When the LTPA token that contains the accessId of user1 is received by app2, it decrypts the token. Both of the servers share the same LTPA keys. The LTPA token then ensures that the foreign realm is a trusted realm, and performs the authorization based on the accessId of user1. If security attribute propagation is not disabled, then the group information of user1 is also propagated to app2. The groups can be used for the authorization check, provided that the authorization table contains the group information. You can associate a special subject, AllAuthenticatedInTrustedRealms, to the roles instead of adding individual users and groups to the authorization table.

If the applications in the above example are deployed in different cells, you must do the following:

- Share the LTPA keys between the cells.
- Update the authorization table for app2 with foreign users and groups accessIds by using the wsadmin utility. The administrative console does not have access to the realms outside of the scope of the cell.



Once trust has been established between the realms, when the server receives the LTPA token and the token is decrypted, it checks to see if the foreign realm is in its inbound trusted realms list. If it is trusted, the authentication succeeds. However, since it is a foreign realm, it does not go search the user registry to gather information about the user. Whatever information is in the LTPA token is used to authorize the user.

The only information in the LTPA token is the unique id of the user. This unique id of the user should exist in the authorization table for this application. If it does, authorization succeeds. However, if attribute propagation is enabled, additional authorization attributes (groups that this user belongs to) for the user are sent from the originating server to the receiving server. These additional attributes are used to make the access decisions. If the groups information exists in the propagation tokens it is used when making the authorization decision.

As previously mentioned, the information about the users and or the groups from the trusted realms should exist in the authorization table of the receiving application. Specifically, the accessId of the users and or groups should exist in the binding file of the application. This must be the case when the application is deployed. In the administrative console, when an application is deployed in a domain you can add the accessIds of the users and groups from any of its trusted realms to the authorization table.

You also have an option to associate a special subject, AllAuthenticatedInTrustedRealms, to the roles instead of adding individual users and groups. This is similar to the AllAuthenticated special subject that is currently supported. The difference is that the AllAuthenticated special subject refers to users in the same realm as the application while the AllAuthenticatedInTrustedRealms special subject applies to all of the users in the trusted realms and in the realm of the application.

You can associate the accessId by using the $AdminApp install script. Because the accessId takes a unique format, use the command task listRegistryUsers with displayAccessIds set to true. If an invalid name or format is entered in this field, the authorization fails.

User and group information from the trusted realms is obtained by the deployment manager since it has access to all of the user registry configurations in all domains. However, in certain situations it is not possible to obtain the users and group information.

Federating a node with security domains

When a security domain is configured in the base version and is federated to a cell, the security domain configured at the base version is also configured for that server in the cell. The same domain security configuration can be used by the server before and after the federation. If a base server is to be federated to a cell, the resource assigned to the security domain should be the server scope instead of the cell scope.

If the base server is expected to be registered with an Administrative Agent process, use the cell scope as the resource if the intention is to have all of the servers in the base profile use this security domain.

If during federation the security domain at the base already exists at the cell level, the addNode command fails. You can use the –excludesecuritydomains option not to include the security domain during federation.

When the federated node is removed from a cell, the resources in that node should be removed from the security domains. If security domains have clusters associated with them that span nodes, the nodes are not removed. You can always remove resources from the security domains or any domains that are not used by using scripting commands or the administrative console.

Security domains in a mixed-version environment

You should create security domains once all of the nodes have been migrated to the latest version. This is especially true if there is a need to associate the cell with a domain. However, if you want to create security domains in a mixed- version environment, be aware of the following:

* If a cell-wide domain is created in a mixed version setup, a domain called PassThroughToGlobalSecurity is created automatically. All mixed clusters are assigned to this domain at the time of the creation of the cell-wide domain. This PassThroughToGlobalSecurity domain is special in the sense that attributes cannot be added to it, only resources can be assigned to it.

All resources assigned to the PassThroughToGlobalSecurity domain use the global security configuration information. Whenever a node in the mixed version setup is migrated to the latest version, the servers and clusters in these nodes are added to this domain. Applications in all of the servers and clusters in these nodes do not use the cell-wide domain; they instead use the global security configuration before and after migration.

If any of these servers need to use the cell-wide domain, you must remove these resources from this PassThroughToGlobalSecurity domain. New servers and clusters that are created in the migrated node use the cell-wide domain, not the PassThroughToGlobalSecurity domain. As a result, you have a mix of servers and clusters, some of them using global security configuration and some using the cell-wide domain.

* Once a cell-wide domain is created, adding any old version cluster members to a WebSphere Application Server Version 7.0 cluster is restricted since this action makes it a mixed cluster. This restriction also holds true when a WebSphere Application Server Version 7.0 cluster is associated with a domain. and a previous version cluster member is added to this cluster. This

restriction is needed to avoid associating a security domain to a mixed cluster.

* If possible, you should create a cell-wide domain after all of the nodes have been migrated. In this case, the cell-wide domain is applicable to the entire cell and not just to parts of it. This also eliminates the need to create the PassThroughToGlobalSecurity domain and the mixed cluster scenario with security domains.

G) Configure auditing.

Security auditing will not be performed unless the audit security subsystem has been enabled. Global security must be enabled for the security audit subsystem to function, as no security auditing occurs if global security is not also enabled.

Before you begin

Before enabling security auditing subsystem, enable global security in your environment.

About this task

The recording of auditable security events is achieved by enabled the security auditing subsystem. Follow these steps to enable the security auditing subsystem.

Procedure

1. Click Security > Security auditing.
2. Select Enable security auditing. The Enable security auditing check box is not selected by default. This check box must be selected to allow security auditing to be performed with the configurations that have been specified in the audit.xml file.

   Note: The audit.xml file is used to store the audit subsystem configurations. Changes to the security auditing subsystem should be made with the user interface or the wsadmin utility. This file should not be edited manually.
3. Select the action from the Audit subsystem failure action dropdown menu to be perform when an audit subsystem failure occurs. Notifications configured to warn of a security auditing subsystem failure will not be posted if the No Warning option is selected for this field. If you select either the Log warning or the Terminate server option, then you must also configure a notification for the action to be performed.
4. Select the Auditor ID from the dropdown menu. The auditor role is needed to make changed to the security auditing configurations. By default, when auditing is first enabled, the primary administrator is also given the auditor role. The primary administrator can then add the auditor role to other users. After the auditor role is added to other users, the auditor role can be removed from the administrator to create a separation of authority between the auditor and the administrator. The Auditor ID is the user considered to be the primary auditor.
5. Optional: Select Enable verbose auditing. When an auditable event is recorded, a default set of audit data is included in the audit data object and recorded to the repository. An additional set of audit data is made available by enabling verbose auditing.
6. Click Apply.
7. Restart the application server. The application server must be restarted before the changes go into effect.

Results

The successful competition of these steps results in the security auditing subsystem being enabled.

What to do next

After enabling the security auditing subsystem, refinements can be made to the configuration. You might want to modify the access control of the audit subsystem to separate the authority of the administrator from the authority of the auditor. If no changes to your access control are needed, then you can configure the types of auditable security events should be recorded. To configure the types of events that are recorded, click Event type filters.

Security Auditing detail

The Security auditing subsystem can be enabled and configured from this panel, by users assigned the auditor role.

To view this administrative console page, click Security > Security Auditing. If Enable security auditing is not selected, then all of the other fields on this panel will be disabled.

Enable security auditing

The Enable security auditing check box allows users to enable or disable Security Auditing. By default, Security Auditing will not be enabled. This field corresponds with the auditEnabled field in the audit.xml file.

Audit subsystem failure action

The Audit subsystem failure action setting describes the behavior of the application server in the event of a failure in the auditing subsystem. Audit Notifications must be configured in order for notifications of a failure in the audit subsystem to be logged. If security auditing is not enabled, then these actions will not be performed. Failures can include an error in the interface or in the event processing. By default, the audit subsystem failure action setting is set to No warning.

The Audit subsystem failure action dropdown menu has the following options:

* No warning

  The No warning action specifies that the auditor will not be notified of a failure in the audit subsystem. The product will continue processing but audit reporting will be disabled.

* Log warning

  The Log warning action specifies that the auditor will be notified of a failure in the audit subsystem. The product will continue processing but audit reporting will be disabled.

* Terminate server

The Terminate server action specifies the application server to gracefully quiesce when an unrecoverable error occurs in the auditing subsystem. If e-mail notifications are configured, the auditor will be sent a notification that an error has occurred. If logging to the system log is configured, the notification of the failure will be logged to the system file.

Primary auditor user name

The Primary auditor user name dropdown menu defines a valid user which exists in the current user registry and for whom the auditor role has been given. By default, this field is blank and is a required field.

Enable verbose auditing

The Enable verbose auditing option determines the amount of audit data that is reported in an audit record. Verbose mode captures all the auditable data points, whereas not enabling verbose mode captures only a subset of the available data. This option is disabled by default.

The Event type filters panel displays a listing of all configured audit specifications with their unique names, the state of their enablement, and the event types and event outcomes that are specified for each configuration.

To view this administrative console page, click Security > Security Auditing > Event type filters.

Creating security auditing event type filters

Event type filters are used to specify the types of auditable security events that are audited. Default event type filters are included with the product, but you can also configure new event type filters to specify a subset of auditable event types to be recorded by the security auditing subsystem.

Before you begin

Before configuring security auditing filters and the rest of the security auditing subsystem, enable global security in your environment. You must be assigned the auditor role to complete this task. Event type filters are used to specify what events are audited. The amount of data that is recorded for each event is specified with the Enable verbose auditing check box on the same panel used to enable the auditing subsystem. Navigate to Security > Security auditing to enable security auditing and determine the data recorded for each event.

The application server provides the following commonly used event type filters by default in the audit.xml template file:

DefaultAuditSpecification_1	SECURITY_AUTHN	SUCCESS
DefaultAuditSpecification_2	SECURITY_AUTHN	DENIED
DefaultAuditSpecification_3	SECURITY_RESOURCE_ACCESS	SUCCESS
DefaultAuditSpecification_4	SECURITY_AUTHN	REDIRECT

New event type filters can be created, or the existing default filters can be extended, to capture more event types and outcomes. Use this task to create new event type filters.

Procedure

1. Click Security > Security Auditing > Event type filters> New.
2. Enter the unique name that should be associated with this event type filter configuration in the Name field.
3. Specify the events that should be recorded when this filter is applied:
    1. Select the events that you want to be audited from the Selectable events list.
    2. Click Add >> to add the selected events to the Enabled events list.
    3. Select the outcomes that you want to be audited from the Selectable event outcomes list.
    4. Click Add >> to add the selected outcomes to the Enabled event outcomes lists.
4. Click OK.

Results

The successful completion of this task results in the creation of an event type filter than can be selected by the audit service providers and audit event factories to gather and record a specific set of auditable security events.

What to do next

After creating an event type filter, the filter must be specified in the audit service provider and the audit event factory to be used to gather or report audit data. The next step in configuring the security auditing subsystem is you should configure an audit service provider to define where the audit data will be archived.

Configuring the default audit service providers for security auditing

The audit service provider is used to format the audit data object that was sent by the audit event factory. After being formatted, the audit data is recorded to the repository defined in the audit service provider configuration.

Before you begin

Before configuring the audit service provider, enable global security in your environment.

About this task

This task configures the audit service provider used to record generated audit records.

Procedure

1. Click Security > Security Auditing > Audit service provider .
2. Click New and then select Binary file based emitter.
3. Enter the unique name that should be associated with this audit service provider in the Name field.
4. Enter the file location of the binary log file in the Audit log file location field.
5. Optional: Enter the maximum size allowed for a single binary log file in the Audit log file size field.

   This field is specified in megabytes. After the maximum audit file size is reached, a new audit file will be created or an existing audit file will be overwritten. If the maximum number of audit log files has not been set, the default maximum file value used is 10 megabytes. There is no audit archiving utility included with the product. You are responsible for the archiving of your audit data.
6. Optional: In the Maximum number of audit log files field, enter the maximum number of audit logs to be stored before the oldest is overwritten.

   The default value for this field is 100. The value of 100 is also used if the field is empty.

   Note: The maximum number of logs does not include the current binary log that is being written to. It is a reference to the maximum number of archived (timestamped) logs. The total number of binary logs that can exist for a server process is the maximum number of archived logs plus the current log.
7. Select the filters to be used by this audit service provider. The Selectable filter list consists of a list of the configured filters that have been configured and are currently enabled.
   1. Select the filters that should be audited from the Selectable filter list.
   2. Click Add >> to add the selected filters to the Enabled filter list.
8. Click Apply.

Results

After completing these steps, your audit data will be sent to the specified repository in the format required by that repository.

What to do next

After creating an audit service provider, the audit service provider must be associated with an audit event factory provide the audit data objects to the audit service provider. Next you should configure an audit event factory.

Audit event factory configuration collection

The Audit event factory configuration panel displays a list of all currently configured audit event factory implementations. This panel allows a user with the auditor role to manage their configured audit event factories. This includes the ability to configure a new implementation, which is done using the New button on this panel.

To view this administrative console page, click Security > Security Auditing > Audit event factory configuration.

Name

The Name field specifies the unique name associated with the audit event factory configuration.

Type

The Type field specifies this audit event factory configuration as either an IBM® audit event factory or a Third party audit event factory.

Class name

The Class name field specifies the class that is being implemented in an audit event factory configuration.

The class name is com.ibm.ws.security.audit.AuditEventFactoryImpl for an IBM event factory. For a Third party audit event factory, the class name is the class specified in the Third party audit event factory class name field.

Audit event factory settings

The Audit event factory settings panel displays the details of a specific audit event factory. The auditor uses this panel to manage and create audit event factory configurations.

To view this administrative console page, click on of the following paths:

* Security > Security Auditing > Audit event factory configuration > audit_event_factory_configuration_name.

* Security > Security Auditing > Audit event factory configuration > New.

Name

Specifies the unique name associated with the audit event factory configuration.

Type

Specifies this audit event factory configuration as either an IBM® audit event factory or a Third party audit event factory. This field does not appear on the panel during the creation of a new audit event factory. It is included when viewing or modifying an existing audit event factory.

IBM audit event factory

Specifies that the Type field of this audit event factory is IBM audit event factory. This check box only appears on the panel during the creation of a new audit event factory. This check box is selected by default when creating a new audit event factory.

Third party audit event factory

Specifies that the Type field of this audit event factory is Third party audit event factory. This check box only appears on the panel during the creation of a new audit event factory. This check box is not selected by default when creating a new audit event factory.

* The Third party audit event factory class name field is active when the Third party audit event factory check box is selected. This field represents the class name of the third-party implementation of the Audit Event Factory interface

Class name

Specifies the class that is being implemented in a audit event factory configuration.

Although not specified during creation, the class name is com.ibm.ws.security.audit.AuditEventFactoryImp for an IBM event factory.

Audit service provider

Specifies where the audit data objects gathered by this audit event factory will be sent.

Selectable filters

Specifies the filters that are currently available to be used for an implementation. To enable a filter for an implementation, select the filter from the Selectable filter list and then click >.

Enabled filters

Specifies the filters that are currently enabled for an implementation. To disable a filter for an implementation, select the filter from the Enabled filter list and then click <.

Custom properties

Specifies properties that the auditor can define to configure the Audit Event Factory implementation. This might be used by third party implementation of the audit event factory interface. Custom properties are not used for the IBM audit event factory implementation.

Each custom property has the following fields:

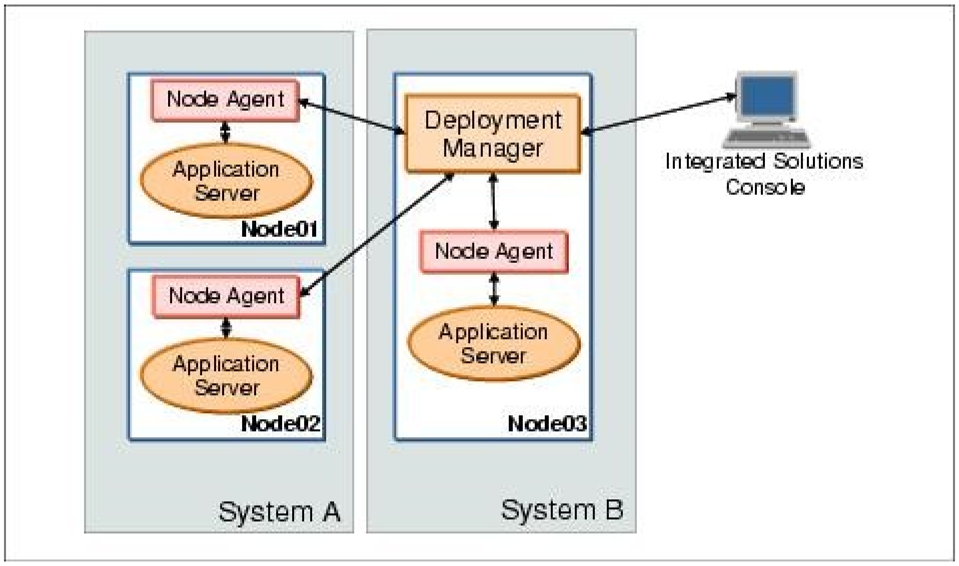
* Name

* Value

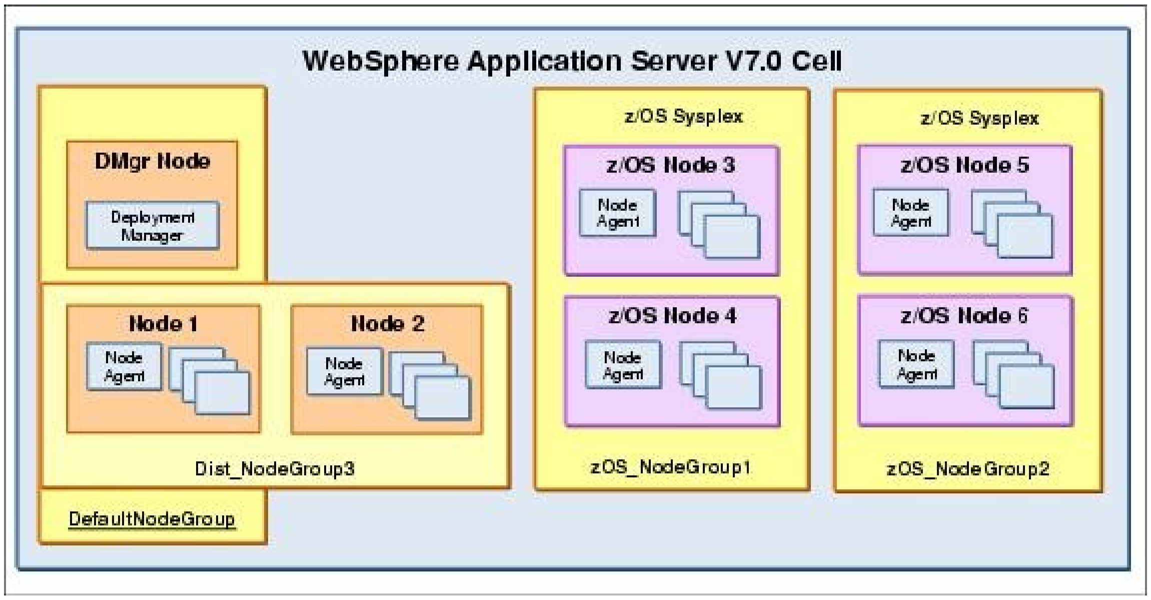
Section 5 - Workload Management, Scalability, High Availability Failover (14%)

A) Federate nodes (including custom profiles).

Node concept in a WebSphere Application Server network deployment configuration.



Node groups - A node group is a grouping of nodes within a cell that have similar capabilities. A node group validates that the node is capable of performing certain functions before allowing them. For example, a cluster cannot contain both z/OS nodes and nodes that are not z/OS-based. In this case, you can define multiple node groups, one for the z/OS nodes and one for nodes other than z/OS. A DefaultNodeGroup is automatically created. This node group contains the deployment manager and any new nodes with the same platform type. A node can be a member of more than one node group.

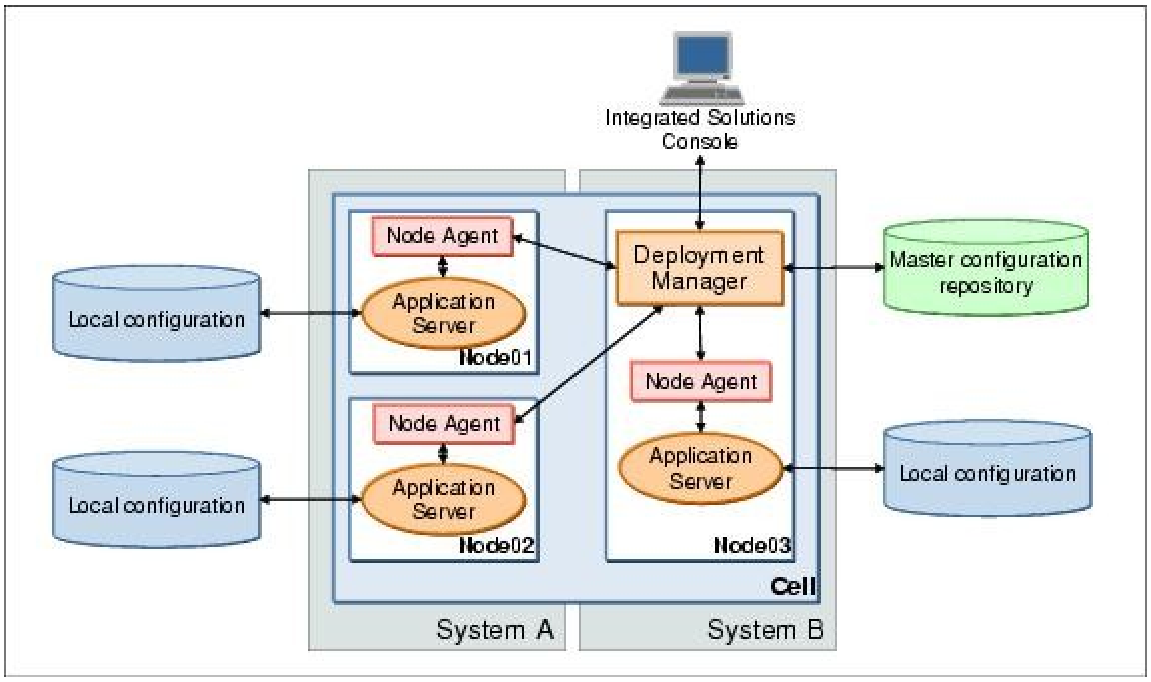


Deployment manager

The deployment manager is the central administration point of a cell that consists of multiple nodes and node groups in a distributed server configuration. The deployment manager uses the node agent to manage the applications servers within one node.

A deployment manager provides management capability for multiple federated nodes and can manage nodes that span multiple systems and platforms. A node can only be managed by a single deployment manager and must be federated to the cell of that deployment manager.

The configuration and application files for all nodes in the cell are centralized into a master configuration repository. This centralized repository is managed by the deployment manager and synchronized with local copies that are held on each of the nodes.



After you set up the Network Deployment product, you mainly need to monitor and control incorporated nodes and the resources on those nodes by using the administrative console or other administrative tools.

A node is a grouping of managed or unmanaged servers. You can add both managed and unmanaged nodes to the WebSphere® Application Server topology. If you add a new node for an existing WebSphere Application Server to the Network Deployment cell, you add a managed node. If you create a new node in the topology for managing Web servers or servers other than WebSphere Application Servers, you add an unmanaged node.

For a managed node, verify that an application server is running on the remote host for the node that you are adding. On the Add Node page, specify a host name, connector type, and port for the application server at the node you are adding.

When nodes are added while LDAP security is enabled, the following exception is generated in the deployment manager System.out log under certain circumstances. If this happens, restart the deployment manager to resolve the problem.

```
0000004d ORBRas E com.ibm.ws.security.orbssl.WSSSLClientSocketFactoryImpl

createSSLSocket ProcessDiscovery : 0 JSSL0080E:
javax.net.ssl.SSLHandshakeException -

The client and server could not negotiate the desired level of security.

Reason?com.ibm.jsse2.util.h: No trusted certificate found
```

Select the discovery protocol.

If the discovery protocol that a node uses is not appropriate for the node, select the appropriate protocol. On the Nodes page, click the node to access the Settings for the node. Select a value for Discovery protocol. User Datagram Protocol (UDP) is faster than Transmission Control Protocol (TCP). However, TCP is more reliable than UDP because UDP does not guarantee the delivery of datagrams to the destination. The default of TCP is the recommended value.

For a node agent or deployment manager, use TCP or UDP.

Synchronize the node configuration.

If you add a managed node or change a managed node configuration, synchronize the node configuration. On the Node Agents page, ensure that the node agent for the node is running. Then, on the Nodes page, select the check box beside the node whose configuration files you want to synchronize and click Synchronize or Full Resynchronize.

Clicking either option sends a request to the node agent for that node to perform a configuration synchronization immediately, instead of waiting for the periodic synchronization to occur. This action is important if automatic configuration synchronization is disabled, or if the synchronization interval is set to a long time, and a configuration change is made to the cell repository that needs to replicate to that node. Settings for automatic synchronization are on the File synchronization service page.

<u>Synchronize requests</u> that a node synchronization operation be performed using the normal synchronization optimization algorithm. This operation is fast, but might not fix problems from manual file edits that occur on the node. It is still possible for the node and cell configuration to be out of synchronization after this operation is performed.

<u>Full Resynchronize</u> clears all synchronization optimization settings and performs configuration synchronization anew, so there is no mismatch between node and cell configuration after this operation is performed. This operation can take longer than the Synchronize operation.

<u>Migration considerations</u>

If you plan to migrate an installation of Network Deployment Version 5 or Version 6 to Version 7, then use the same cell name for the Version 7 deployment manager that you used for the Version 5 or Version 6 cell. A cell name must be unique in any circumstance in which the product is running on the same physical machine or cluster of machines, such as a sysplex. Additionally, a cell name must be unique in any circumstance in which network connectivity between entities is required either between the cells or from a client that must communicate with each of the cells. Cell names must also be unique if their namespaces are federated. Otherwise, you might encounter symptoms such as a javax.naming.NameNotFoundException error, in which case, create uniquely named cells.

After migrating the cell, the Version 5 or Version 6 managed nodes are now managed by the Version 7 deployment manager in compatibility mode. You can migrate individual Version 5 or Version 6 managed nodes in the cell to Version 7. To do so, you must create a Version 7 profile with the same node name as the Version 5 or Version 6 managed node.

The deployment manager must be running and accessible when you click Next on the Federation panel to federate the custom node. If the custom profile is on a machine that does not have a deployment manager, then the deployment manager must be running and accessible over the network to allow the federation of the node. If the deployment manager is not running or not accessible before you click Next, but you can start it and make it accessible at this time, then do so. Otherwise, select the Federate the node later check box.

If you are unsure whether the deployment manager is running or accessible, then do not federate now. Federate the node when you can verify the availability of the deployment manager.

A possibility exists that the deployment manager is reconfigured to use the non-default remote method invocation (RMI) as the preferred Java Management Extensions (JMX) connector. Click System Administration > Deployment manager > Administrative services in the administrative console of the deployment manager to verify the preferred connector type.

If RMI is the preferred JMX connector, then you must use the addNode command to federate the custom profile later. Use the addNode command so that you can specify the JMX connector type and the RMI port.

If the deployment manager uses the default SOAP JMX connector type, specify the host name and SOAP port and federate the node now to create a functional node that you can customize.

Federating when the deployment manager is not available

If you federate a custom node when the deployment manager is not running or is not accessible, then an error message is displayed. If the deployment manager becomes unavailable during the profile creation process, then the installation indicator in the logs is INSTCONFFAIL, to indicate a complete failure. The resulting custom profile is unusable. You must delete the profile. Read about deleting a profile for more information.

If you chose to federate now, and you previously selected Advanced profile creation, then the Security certificate panel displays next. Go to the step on creating and importing certificates.

Best practice: When you import a personal certificate as the default personal certificate, import the root certificate that signed the personal certificate. Otherwise, the Profile Management Tool adds the signer of the personal certificate to the trust.p12 file.

Verify that the ports within the custom profile are unique, or intentionally conflicting, and click Next.

Port conflict resolution

If you suspect a port conflict, then you can investigate the port conflict after the profile is created. Determine the ports that are used during profile creation by examining the following files.

* $profile_root/properties/portdef.props file

Included in this file are the keys and values that are used in setting the ports. If you discover ports conflicts, then you can reassign ports manually. To reassign ports, run the updatePorts.ant file by using the ws_ant script.

Refer to the description of the manageprofiles command to learn about creating a profile using a command instead of the Profile Management Tool.

The Profile Management Tool creates a log during profile creation. The logs are in the install_dir/logs/manageprofiles directory. The files are named in this pattern: manageprofiles_create_profile_name.log.

Federating the node to a cell using the addNode command does not merge any cell-level configuration, including virtual host information. If the virtual host and aliases for the new cell do not match the product, you cannot access the applications running on the servers. You have to manually add all the virtual host and host aliases to the new cell, using the administrative console running on the deployment manager.

When the -includeapps parameter is specified, an OutOfMemoryError might occur if the Java virtual machine (JVM) heap size is too small. When this error occurs, the following error message is issued:

```
ADMU0111E: Program exiting with error: java.lang.OutOfMemoryError
```

This error can occur when large applications are processed, or when there is a large number of applications in the Base Application Server.

To recover from this error and successfully federate the application server, complete the following actions:

1. Issue the cleanupNode command on your deployment manager server. Read about the cleanupNode command for more information about this command.
2. Increase the JVM heap size for the addNode script. When you issue the addNode

command, the JVM heap size is set to -Xms128m -Xmx512m. To increase these values, use the -javaoption parameter.

Verify that the deployment manager and nodes are updated to the same revision level within the WebSphere Application Server. For example, a deployment manager at level 6.0.1 will be unable to federate with nodes at 6.0.2.

If the deployment manager is running, you can use the cleanupNode command on deployment manager where the incomplete node resides.

Manually delete the directory that was created on the deployment manager configuration during an addNode command operation that was incomplete.

B) Create clusters and cluster members.

A cluster is a set of application servers that you manage together as a way to balance workload.

You might want to create a cluster if you need to:

* Balance your client requests across multiple application servers.

* Provide a highly available environment for your applications.

A cluster enables you to manage a group of application servers as a single unit, and distribute client requests among the application servers that are members of the cluster.

When you create the first cluster member, remember that a copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

Weight

Specifies the amount of work that is directed to the application server.

If the weight value for the server is greater than the weight values that are assigned to other servers in the cluster, the server receives a larger share of the cluster workload. The value can range from 0 to 20. Enter zero to indicate that you do not want requests to route to this application server unless this server is the only server that is available to receive requests.

Core group

Specifies the core group in which the application server resides. This field displays only if you have multiple core groups configured. You can change this value only for the first cluster member.

Generate unique HTTP ports

Specifies that a unique HTTP port is generated for the application server. By generating unique HTTP ports for the application server, you avoid potential port collisions and configurations that are not valid.

Select basis for first cluster member:

* If you select Create the member using an application server template, the settings for the new application server are identical to the settings of the application server template you select from the list of available templates.

* If you select Create the member using an existing application server as a template, the settings for the new application server are identical to the settings of the application server you select from the list of existing application servers.

* If you select Create the member by converting an existing application server, the application server you select from the list of available application servers becomes a member of this cluster.

* If you select None. Create an empty cluster , a new cluster is created but it does not

contain any cluster members.

Important: The basis options are available only for the first cluster member. All other members of a cluster are based on the cluster member template which is created from the first cluster member.

Clusters and workload management

Clusters are sets of servers that are managed together and participate in workload management. Clusters enable enterprise applications to scale beyond the amount of throughput capable of being achieved with a single application server. Clusters also enable enterprise applications to be highly available because requests are automatically routed to the running servers in the event of a failure. The servers that are members of a cluster can be on different host machines. In contrast, servers that are part of the same node must be located on the same host machine. A cell can include no clusters, one cluster, or multiple clusters.

Servers that belong to a cluster are members of that cluster set and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data. One cluster member might be running on a huge multi-processor enterprise server system, while another member of that same cluster might be running on a smaller system. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical. This allows client work to be distributed across all the members of a cluster instead of all workload being handled by a single application server.

When you create a cluster, you make copies of an existing application server template. The template is most likely an application server that you have previously configured. You are offered the option of making that server a member of the cluster. However, it is recommended that you keep the server available only as a template, because the only way to remove a cluster member is to delete the application server. When you delete a cluster, you also delete any application servers that were members of that cluster. There is no way to preserve any member of a cluster. Keeping the original template intact allows you to reuse the template if you need to rebuild the configuration.

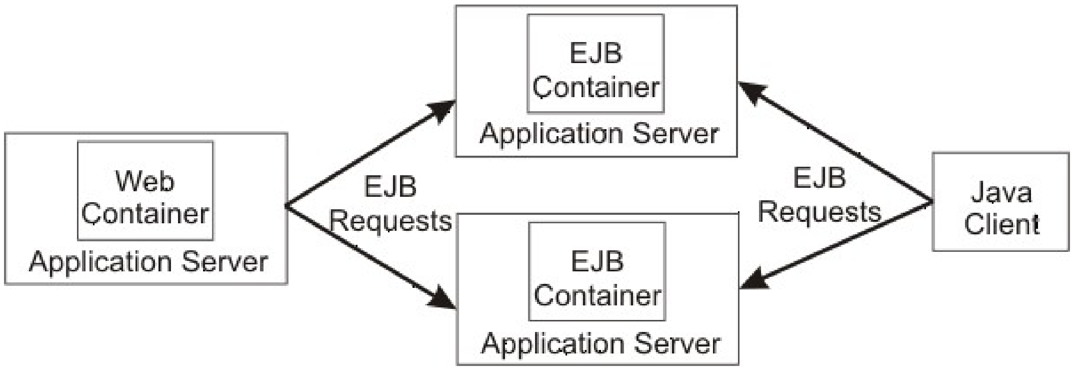
A vertical cluster has cluster members on the same node, or physical machine. A horizontal cluster has cluster members on multiple nodes across many machines in a cell. You can configure either type of cluster, or have a combination of vertical and horizontal clusters.

You can use the administrative console to specify a weight for a cluster member. The weight you assign to a cluster member should be based on its approximate, proportional ability to do work. The weight value specified for a specific member is only meaningful in the context of the weights you specify for the other members within a cluster. The weight values do not indicate absolute capability. If a cluster member is unavailable, the Web server plug-in temporarily routes requests around that cluster member.

For example, if you have a cluster that consists of two members, assigning weights of 1 and 2 causes the first member to get approximately 1/3 of the workload and the second member to get approximately 2/3 of the workload. However, if you add a third member to the cluster, and assign the new member a weight of 1, approximately 1/4 of the workload now goes to the first member, approximately 1/2 of the workload goes to the second member, and approximately 1/4 of the workload goes to the third member. If the first cluster member becomes unavailable, the second member gets approximately 2/3 of the workload and third member gets approximately 1/3 of the workload.

The weight values only approximate your load balance objectives. There are other application dependencies, such as thread concurrency, local setting preferences, affinity, and resource availability that are also factors in determining where a specific request is sent. Therefore, do not use the exact pattern of requests to determine the weight assignment for specific cluster members.

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers can be clustered with the EJB containers, enabling the distribution of enterprise bean requests between EJB containers on different application servers.



In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java™ programs using RMI/IIOP, or other EJBs.

The server weighted round robin routing policy ensures a balanced routing distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster is that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism sends more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members.

You can set up workload management to balance the tasks between different clusters.

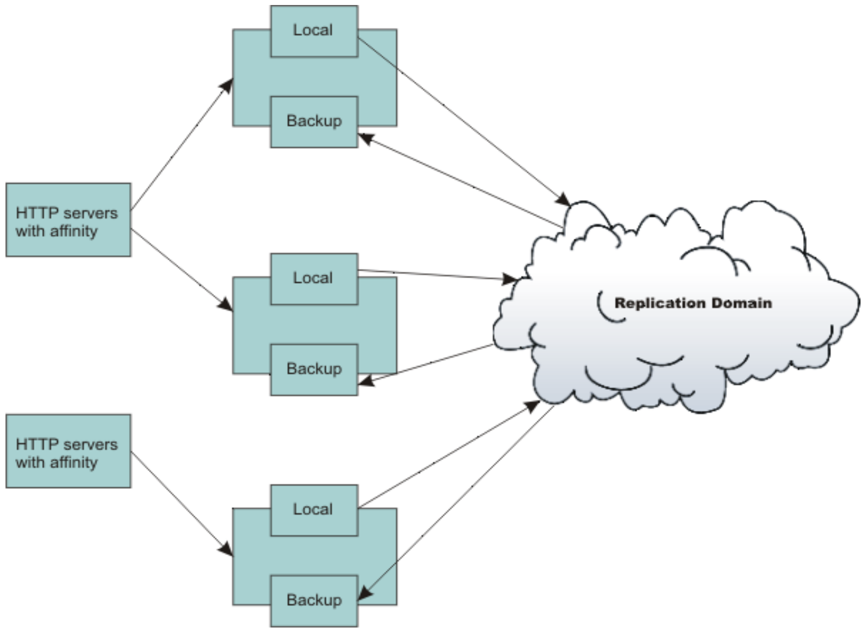
You can choose to have requests sent to the node on which the client resides as the preferred routing. In this case, only cluster members on that node are chosen (using the round robin weight method). Cluster members on remote nodes are chosen only if a local server is not available.

Multiple servers that can service the same client request form the basis for failover support. If a server fails while processing a client request, the failed request can be rerouted to any of the remaining cluster members. Even if several servers fail, as long as at least one cluster member is running, client requests continue to be serviced.

C) Configure session management (memory-to-memory, database persistence).

Memory-to-memory topology: Peer-to-peer function

The basic peer-to-peer (both client and server function, or both mode) topology is the default configuration and has a single replica. However, you can also add additional replicas by configuring the replication domain.



In this basic peer-to-peer topology, each server Java™ Virtual Machine (JVM) can:

* Host the Web application leveraging the HTTP session
* Send out changes to the HTTP session that it owns
* Receive backup copies of the HTTP session from all of the other servers in the cluster

This configuration represents the most consolidated topology, where the various system parts are collocated and requires the fewest server processes. When using this configuration, the most stable implementation is achieved when each node has equal capabilities (CPU, memory, and so on), and each handles the same amount of work.

It is also important to note that when using the peer-to-peer topology, that one session replication backup must run at all times for invalidation to occur. For example, if you have a cluster of 2 application servers, server1 and server2, that are both configured in the peer-to-peer mode and server2 goes down. All of backup information for server1 is lost and those sessions are no longer invalidated properly.

Session hot failover

A new feature called session hot failover has been added to this release. This feature is only applicable to the peer-to-peer mode. In a clustered environment, session affinity in the WebSphere® Application Server plug-in routes the requests for a given session to the same server. If the current owner server instance of the session fails, then the WebSphere Application Server plug-in routes the requests to another appropriate server in the cluster. For a cluster configured to run in the peer-to-peer mode this feature causes the plug-in to failover to a server that already contains the backup copy of the session, therefore avoiding the overhead of session retrieval from another server containing the backup. However, hot failover is specifically for servant region failures. When an entire server, meaning both controller and server fail, sessions may have to be retrieved over the network.

You must upgrade all WebSphere Application Server plug-in instances that front the Application Server cluster to version 6.0 to ensure session affinity when using the peer-to-peer mode.

Memory-to-memory session replication is the session replication to another WebSphere® Application Server. In this mode, sessions can replicate to one or more Application Servers to address HTTP Session single point of failure (SPOF).

The WebSphere Application Server instance in which the session is currently processed is referred to as the owner of the session. In a clustered environment, session affinity in the WebSphere Application Server plug-in routes the requests for a given session to the same server. If the current owner server instance of the session fails, then the WebSphere Application Server plug-in routes the requests to another appropriate server in the cluster. In a peer-to-peer cluster, the hot failover feature causes the plug-in to failover to a server that already contains the backup copy of the session, avoiding the overhead of session retrieval from another server containing the backup. In a client/server cluster, the server retrieves the session from a server that has the backup copy of the session. The server now becomes the owner of the session and affinity is now maintained to this server.

There are three possible modes to run in:

* Server mode: Only store backup copies of other WebSphere Application Server sessions and not to send out copies of any session created in that particular server

* Client mode: Only broadcast or send out copies of the sessions it owns and not to receive backup copies of sessions from other servers

* Both mode: Simultaneously broadcast or send out copies of the sessions it owns and act as a backup table for sessions owned by other WebSphere Application Server instances.

You can select the replication mode of server, client, or both when configuring the session management facility for memory-to-memory replication. The default is both. This storage option is controlled by the mode parameter.

The memory-to-memory replication function is accomplished by the creation of a data replication service instance in an application server that talks to other data replication service instances in remote application servers. You must configure this data replication service instance as a part of a replication domain. Data replication service instances on disparate application servers that replicate to one another must be configured as a part of the same domain. You must configure all session managers connected to a replication domain to have the same topology. If one session manager instance in a domain is configured to use the client/server topology, then the rest of the session manager instances in that domain must be a combination of servers configured as Client only and Server only. If one session manager instance is configured to use the peer-to-peer topology, then all session manager instances must be configured as Both client and server. For example, a server only data replication service instance and a both client and server data replication service instance cannot exist in the same replication domain. Multiple data replication service instances that exist on the same application server due to session manager memory-to-memory configuration at various levels that are configured to be part of the same domain must have the same mode.

With respect to mode, the following are the primary examples of memory-to-memory replication configuration:

* Peer-to-peer replication

* Client/server replication

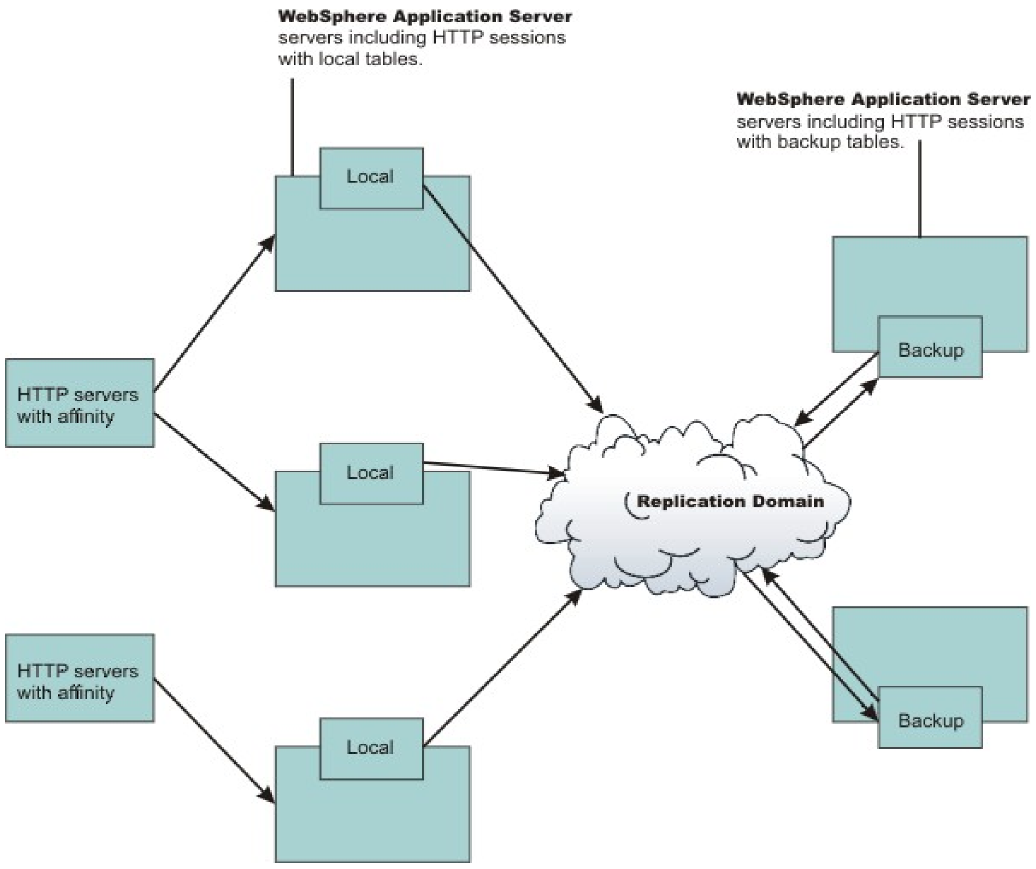
Although the administrative console allows flexibility and additional possibilities for memory-to-memory replication configuration, only the configurations provided above are officially supported.

There is a single replica in a cluster by default. You can modify the number of replicas through the replication domain.

Memory-to-memory topology: Client/server function

The client/server configuration, used to attain session affinity, consists of a cluster of servers that are configured as client only and server only. Using the client/server configuration has benefits such as isolating the handling of backup data from local data, recycling backup servers without affecting the servers running the application, and removing the need for a one-to-one correspondence between servers to attain session affinity.

The following figure depicts the client/server mode. There is a tier of applications servers that host Web applications using HTTP sessions, and these sessions are replicated out as they are created and updated. There is a second tier of servers without a Web application installed, where the session manager receives updates from the replication clients.


WebSphere Application Server servers including HTTP sessions with local tables.
WebSphere Application Server servers including HTTP sessions with backup tables.
Local
Local
Local
Backup
Backup
HTTP servers with affinity
HTTP servers with affinity
Replication Domain

Benefits of the client/server configuration include:

Isolation for failure recovery

In this case we are isolating the handling of backup data from local data; aside from isolating the moving parts in case of a catastrophic failure in one of them, you again free up memory and processing in the servers processing the Web application.

Isolation for stopping and starting

You can recycle a backup server without affecting the servers running the application (when there are two or more backups, failure recovery is possible), and conversely recycle an application JVM without potentially losing that backup data for someone.

Consolidation

There is most likely no need to have a one-to-one correspondence between servers handling backups and those processing the applications; hence, you are again reducing the number of places to which you transfer the data.

Disparate hardware:

While you run your Web applications on cheaper hardware, you may have one or two more powerful computers in the back end of your enterprise that have the capacity to run a couple of session managers in replication server mode; allowing you to free up your cheaper Web application hardware to process the Web application.

Timing consideration: Start the backup application servers first to avoid unexpected timing windows. The clients attempt to replicate information and HTTP sessions to the backup servers as soon as they come up. As a result, HTTP sessions that are created prior to the time at which the servers come up might not replicate successfully.

Memory-to-memory session partitioning

Session partitioning gives the administrator the ability to filter or reduce the number of destinations that the session object gets sent to by the replication service. You can also configure session partitioning by specifying the number of replicas on the replication domain. The Single replica option is chosen by default. Since the number of replicas is global for the entire replication domain, all the session managers connected to the replication domain use the same setting.

Single replica

You can replicate a session to only one other server, creating a single replica. When this option is chosen, a session manager picks another session manager that is connected to the same replication domain to replicate the HTTP session to during session creation. All updates to the session are only replicated to that single server. This option is set at the replication domain level. When this option is set, every session manager connected to this replication domain creates a single backup copy of HTTP session state information on a backup server.

Full group replica

Each object is replicated to every application server that is configured as a consumer of the replication domain. However, in the peer-to-peer mode, this topology is the most redundant because everyone replicates to everyone and as you add servers, more overhead (both CPU and memory) is needed to deal with replication. This mode is most useful for dynamic caching replication. Redundancy does not affect the client/server mode because clients only replicate to servers that are set to server mode.

Specific number of replicas

You can specify a specific number of replicas for any entry that is created in the replication domain. The number of replicas is the number of application servers that the user wants to use to replicate in the domain. This option eliminates redundancy that occurs in a full group replica and also provides additional backup than a single replica. In peer-to-peer mode, the number of replicas cannot exceed the total number of application servers in the cluster. In the client/server mode, the number of replicas cannot exceed the total number of application servers in the cluster that are set to server mode.

Creating a table for session persistence

You can use a database table to collect and store session data. If you are using a database table for session persistence, you must create and define a database table that is associated with the application server.

About this task

Whenever the session manager is set for database persistence, the session manager creates a table for its use. If you want to expand the column size limits to make it more appropriate for your Web site, you can create the table externally. If the external table is specified as the target table in the session manager database persistence configuration, then during the session manager start up, the external table is used. In most cases it is better to let the session manager create the table during startup.

Note:

1. At run time, the session manager accesses the target table using the identity of the J2EE server in which the owning Web application is deployed. Any Web container that is configured to use persistent sessions must have both read and update access to the subject database table.

2. HTTP session processing uses the index defined using the CREATE INDEX statement to avoid database deadlocks. In some situations, such as when a relatively small table size is defined for the database, DB2 may decide not to use this index. When the index is not used, database deadlocks can occur. If database deadlocks occur, see the DB2 Administration Guide for the version of DB2 you are using for recommendations on how to calculate the space required for an index and adjust the size of the tables that you are using accordingly.

3. It might be necessary to tune DB2 to make efficient use of the sessions database table and to avoid deadlocks when accessing it. Your DB2 administrator should refer to the DB2 Administration Guide for specific information about tuning the version of DB2 that you are using.

Configuring for database session persistence

You can configure a database to collect session data for database session persistence.

About this task

To configure the session management facility for database session persistence, complete the following steps.

Procedure

1. Create and configure a JDBC provider.

2. Create a data source pointing to a database.

Use the JDBC provider that you defined: Resources > JDBC > JDBC Providers > JDBC_provider > Data Sources > New. The data source should be non-JTA, for example, non-XA enabled. Note the JNDI name of the data source.

Point to an existing database.

3. Verify that the correct database is listed under Resources > JDBC Providers > JDBC_provider > Data Sources > datasource_name. If necessary, contact your database administrator to verify the correct database name.

4. Go to the appropriate level of Session Management.

5. Under Additional Properties, click Distributed Environment Settings

6. Select and click Database.

7. Specify the Data Source JNDI name from a previous step. The database user ID and password are case-sensitive.

8. Specify the database user ID and password that is used to access the database and for table creation. When you created your data source, you might have specified a Container Managed Authentication Alias or a Component Managed Authentication Alias; however, these two settings are not used by the session manager for session persistence. The session manager uses the userId and password specified in this step for session persistence.

9. Retype the password for confirmation.

10. Configure a table space and page sizes for DB2® session databases.

11. Switch to a multirow schema.

12. Click OK.

13. If you want to change the tuning parameters, click Custom Tuning Parameters under Additional properties.

14. Click Apply.

15. Click Save.

<u>Best practices for using HTTP sessions</u>

Best practice: Browse the following recommendations for implementing HTTP sessions.bprac

* Enable Security integration for securing HTTP sessions

HTTP sessions are identified by session IDs. A session ID is a pseudo-random number generated at the runtime. Session hijacking is a known attack HTTP sessions and can be prevented if all the requests going over the network are enforced to be over a secure connection (meaning, HTTPS). But not every configuration in a customer environment enforces this constraint because of the performance impact of SSL connections. Due to this relaxed mode, HTTP session is vulnerable to hijacking and because of this vulnerability, WebSphere® Application Server has the option to tightly integrate HTTP sessions and WebSphere Application Server security. Enable security in WebSphere Application Server so that the sessions are protected in a manner that only users who created the sessions are allowed to access them.

* Release HttpSession objects using javax.servlet.http.HttpSession.invalidate() when finished.

HttpSession objects live inside the Web container until:

- o The application explicitly and programmatically releases it using the javax.servlet.http.HttpSession.invalidate method; quite often, programmatic invalidation is part of an application logout function.
- o WebSphere Application Server destroys the allocated HttpSession when it expires (default = 1800 seconds or 30 minutes). The WebSphere Application Server can only maintain a certain number of HTTP sessions in memory based on session management settings. In case of distributed sessions, when maximum cache limit is reached in memory, the session management facility removes the least recently used (LRU) one from cache to make room for a session.

* Avoid trying to save and reuse the HttpSession object outside of each servlet or JSP file.

The HttpSession object is a function of the HttpRequest (you can get it only through the req.getSession method), and a copy of it is valid only for the life of the service method of the servlet or JSP file. You cannot cache the HttpSession object and refer to it outside the scope of a servlet or JSP file.

* Implement the java.io.Serializable interface when developing new objects to be stored in the HTTP session.

Serializability of a class is enabled by the class implementing the java.io.Serializable interface. Implementing the java.io.Serializable interface allows the object to properly serialize when using distributed sessions. Classes that do not implement this interface will not have their states serialized or deserialized. Therefore, if a class does not implement the Serializable interface, the JVM cannot persist its state into a database or into another JVM. All subtypes of a serializable class are serializable. An example of this follows:

public class MyObject implements java.io.Serializable {...}

Make sure all instance variable objects that are not marked transient are serializable. You cannot cache a non-serializable object.

In compliance with the Java™ Servlet specification, the distributed servlet container must create an IllegalArgumentException for objects when the container cannot support the mechanism necessary for migration of the session storing them. An exception is created only when you have selected distributable.

* The HTTPSession API does not dictate transactional behavior for sessions.

Distributed HTTPSession support does not guarantee transactional integrity of an attribute in a failover scenario or when session affinity is broken. Use transactional aware resources like enterprise Java beans to guarantee the transaction integrity required by your application.

* Ensure the Java objects you add to a session are in the correct class path.

If you add Java objects to a session, place the class files for those objects in the correct class path (the application class path if utilizing sharing across Web modules in an enterprise application, or the Web module class path if using the Servlet 2.2-complaint session sharing) or in the directory containing other servlets used in WebSphere Application Server. In the case of session clustering, this action applies to every node in the cluster.

Because the HttpSession object is shared among servlets that the user might access, consider adopting a site-wide naming convention to avoid conflicts.

* Avoid storing large object graphs in the HttpSession object.

In most applications each servlet only requires a fraction of the total session data. However, by storing the data in the HttpSession object as one large object, an application forces WebSphere Application Server to process all of it each time.

* Utilize Session Affinity to help achieve higher cache hits in the WebSphere Application Server.

WebSphere Application Server has functionality in the HTTP Server plug-in to help with session affinity. The plug-in reads the cookie data (or encoded URL) from the browser and helps direct the request to the appropriate application or clone based on the assigned session key. This functionality increases use of the in-memory cache and reduces hits to the database or another WebSphere Application Server instance

* Maximize use of session affinity and avoid breaking affinity.

Using session affinity properly can enhance the performance of the WebSphere Application Server. Session affinity in the WebSphere Application Server environment is a way to maximize the in-memory cache of session objects and reduce the amount of reads to the database or another WebSphere Application Server instance. Session affinity works by caching the session objects in the server instance of the application with which a user is interacting. If the application is deployed in multiple servers of a server group, the application can direct the user to any one of the servers. If the users starts on server1 and then comes in on server2 a little later, the server must write all of the session information to the external location so that the server instance in which server2 is running can read the database. You can avoid this database read using session affinity. With session affinity, the user starts on server1 for the first request; then for every successive request, the user is directed back to server1. Server1 has to look only at the cache to get the session information; server1 never has to make a call to the session database to get the information.

You can improve performance by not breaking session affinity. Some suggestions to help avoid breaking session affinity are:

o Combine all Web applications into a single application server instance, if possible, and use modeling or cloning to provide failover support.

o Create the session for the frame page, but do not create sessions for the pages within the frame when using multi-frame JSP files. (See discussion later in this topic.)

* When using multi-framed pages, follow these guidelines:

o Create a session in only one frame or before accessing any frame sets. For example, assuming there is no session already associated with the browser and a user accesses a multi-framed JSP file, the browser issues concurrent requests for the JSP files. Because the requests are not part of any session, the JSP files end up creating multiple sessions and all of the cookies are sent back to the browser. The browser honors only the last cookie that arrives. Therefore, only the client can retrieve the session associated with the last cookie. Creating a session before accessing multi-framed pages that utilize JSP files is recommended.

o By default, JSP files get a HTTPSession using request.getSession(true) method. So by default JSP files create a new session if none exists for the client. Each JSP page in the browser is requesting a new session, but only one session is used per browser instance. A developer can use <% @ page session="false" %> to turn off the automatic session creation from the JSP files that do not access the session. Then if the page needs access to the session information, the developer can use <%HttpSession session = javax.servlet.http.HttpServletRequest.getSession(false); %> to get the already existing session that was created by the original session creating JSP file. This action helps prevent breaking session affinity on the initial loading of the frame pages.

o Update session data using only one frame. When using framesets, requests come into the HTTP server concurrently. Modifying session data within only one frame so that session changes are not overwritten by session changes in concurrent frameset is recommended.

o Avoid using multi-framed JSP files where the frames point to different Web applications. This action results in losing the session created by another Web application because the JSESSIONID cookie from the first Web application gets overwritten by the JSESSIONID created by the second Web application.

* Secure all of the pages (not just some) when applying security to servlets or JSP files that use sessions with security integration enabled, .

When it comes to security and sessions, it is all or nothing. It does not make sense to protect access to session state only part of the time. When security integration is enabled in the session management facility, all resources from which a session is created or accessed must be either secured or unsecured. You cannot mix secured and unsecured resources.

The problem with securing only a couple of pages is that sessions created in secured pages are created under the identity of the authenticated user. Only the same user can access sessions in other secured pages. To protect these sessions from use by unauthorized users, you cannot access these sessions from an unsecured page. When a request from an unsecured page occurs, access is denied and an UnauthorizedSessionRequestException error is created. (UnauthorizedSessionRequestException is a runtime exception; it is logged for you.)

* Use manual update and either the sync() method or time-based write in applications that read session data, and update infrequently.

With END_OF_SERVICE as write frequency, when an application uses sessions and anytime data is read from or written to that session, the LastAccess time field updates. If database sessions are used, a new write to the database is produced. This activity is a performance hit that you can avoid using the Manual Update option and having the record written back to the database only when data values update, not on every read or write of the record.

To use manual update, turn it on in the session management service. (See the tables above for location information.) Additionally, the application code must use the com.ibm.websphere.servlet.session.IBMSession class instead of the generic HttpSession. Within the IBMSession object there is a sync method. This method tells the WebSphere Application Server to write the data in the session object to the database. This activity helps the developer to improve overall performance by having the session information persist only when necessary.

Note: An alternative to using the manual updates is to utilize the timed updates to persist data at different time intervals. This action provides similar results as the manual update scheme.

* Implement the following suggestions to achieve high performance:

o If your applications do not change the session data frequently, use Manual Update and the sync function (or timed interval update) to efficiently persist session information.

o Keep the amount of data stored in the session as small as possible. With the ease of using sessions to hold data, sometimes too much data is stored in the session objects. Determine a proper balance of data storage and performance to effectively use sessions.

o If using database sessions, use a dedicated database for the session database. Avoid using the application database. This helps to avoid contention for JDBC connections and allows for better database performance.

o If using memory-to-memory sessions, employ partitioning (either group or single replica) as your clusters grow in size and scaling decreases.

o Verify that you have the latest fix packs for the WebSphere Application Server.

* Utilize the following tools to help monitor session performance.

o Run the com.ibm.servlet.personalization.sessiontracking.IBMTrackerDebug servlet. - To run this servlet, you must have the servlet invoker running in the Web application you want to run this from. Or, you can explicitly configure this servlet in the application you want to run.

o Use the WebSphere Application Server Resource Analyzer which comes with WebSphere Application Server to monitor active sessions and statistics for the WebSphere Application Server environment.

o Use database tracking tools such as "Monitoring" in DB2®. (See the respective documentation for the database system used.)

D) Create and configure Data Replication Service (DRS) replication domains.

Replication is a service that transfers data, objects, or events among application servers. Data replication service (DRS) is the internal WebSphere Application Server component that replicates data.

Use data replication to make data for session manager, dynamic cache, and stateful session beans available across many application servers in a cluster. The benefits of using replication vary depending on the component that you configure to use replication.

* Session manager uses the data replication service when configured to do memory-to-memory replication. When memory-to-memory replication is configured, session manager maintains data about sessions across multiple application servers, preventing the loss of session data if a single application server fails.
* Dynamic cache uses the data replication service to further improve performance by copying cache information across application servers in the cluster, preventing the need to repeatedly perform the same tasks and queries in different application servers.
* Stateful session beans use the replication service so that applications using stateful session beans are not limited by unexpected server failures.

Important: When you use the replication services, ensure that the Propagate security attributes option is enabled. Security attribute propagation is enabled, by default.

You can define the number of replicas that DRS creates on remote application servers. A replica is a copy of the data that copies from one application server to another. The number of replicas that you configure affects the performance of your configuration. Smaller numbers of replicas result in better performance because the data does not have to copy many times. However, if you create more replicas, you have more redundancy in your system. By configuring more replicas, your system becomes more tolerant to possible failures of application servers in the system because the data is backed up in several locations.

Defining a single replica configuration helps you to avoid a single point of failure in the system. However, if your system must be tolerant to more failure, introduce extra redundancy in the system. Increase the number of replicas that you create for any HTTP session that is replicated with DRS. The Number of replicas property for any replication domain that is used by the dynamic cache service must be set to Entire domain.

Session manager, dynamic cache, and stateful session beans are the three consumers of replication. A consumer is a component that uses the replication service. When you configure replication, the same types of consumers belong to the same replication domain. For example, if you are configuring both session manager and dynamic cache to use DRS to replicate objects, create separate replication domains for each consumer. Create one replication domain for all the session managers on all the application servers and one replication domain for the dynamic cache on all the application servers. The only exception to this rule is to create one replication domain if you are configuring replication for HTTP sessions and stateful session beans. Configuring one replication domain in this case ensures that the backup state information is located on the same backup application servers.

Replicating data across application servers in a cluster

Use this task to configure a data replication domain to transfer data, objects, or events for session manager, dynamic cache, or stateful session beans. Data replication domains use the data replication service (DRS), which is an internal component that performs replication services, including replicating data, objects, and events among application servers.

Before you begin

Determine if you are using a multi-broker replication domain. If you configured a data replication domain with a previous version of the product, you might be using a multi-broker replication domain. Any replication domains that you create with the current version of the product are data replication domains. You should migrate any multi-broker replication domains to data replication domains.

About this task

Use this task to configure replication, a service that transfers data, objects, or events among the application servers in a cluster. Use replication to prevent loss of session data with session manager, to further improve the performance of the dynamic cache service, and to provide failover in stateful session beans.

Avoid trouble: If you select the Configure HTTP memory-to-memory replication option when you create a cluster, the replication domain is automatically created for you.gotcha

Similarly if, instead of WAS01Network , the cell name is simply WAS1, you have to pad the high level qualifier with the first three characters of the string DRSSTREAM. The high level qualifier then becomes WAS1DRS.

Complete the following steps to enable data replication among the application servers in a cluster.

Procedure

1. Create a replication domain. Use one of the following methods to create a replication domain:

   * Create a replication domain manually.

   To create a replication domain manually without creating a new cluster, click Environment > Replication domains > New in the administrative console.

   On this page you can specify the properties for the replication domain, including timeout, encryption, and number of replicas.

   * Create a replication domain when you create a cluster.

To create a replication domain when you create a cluster, click Servers > Clusters > Clusters > New in the administrative console. Then click Configure HTTP memory-to-memory replication . The replication domain that is created has the same name as the cluster and has the default settings for a replication domain. The default settings for a replication domain are to create a single replica of each piece of data and to have encryption disabled. To modify the replication domain properties, click Environment > Replication domains > New replication_domain_name in the administrative console.

2. Configure the consumers, or the components that use the replication domains. Dynamic cache, session manager, and stateful session beans are the three types of replication domain consumers. Each type of consumer must be configured with a different replication domain. For example, session manager uses one replication domain and dynamic cache uses a different replication domain. However, use one replication domain if you are configuring HTTP session memory-to-memory replication and stateful session bean replication. Using one replication domain in this case ensures that the backup state information of HTTP sessions and stateful session beans are on the same application servers.

3. Determine whether your configuration requires additional thread resources.

The replication service uses threads obtained from the Default thread pool for various tasks, including processing messages. Other application server components also use this thread pool. Therefore, during application server startup the default maximum thread pool size of 20 might not be sufficient to allow the replication service to obtain enough threads from the pool to process all of the incoming replication messages. The number of incoming messages is influenced by the number of application servers in the domain and the number of replication domain consumers on each application server. The number of messages to be processed increases as the number of application servers in the domain increases and/or the number of replication consumers increases.

Persistent data not being replicated to the application servers during server startup might be an indication that you need to increase the setting for the maximum thread pool size. In larger configurations, doubling the maximum size of the Default thread pool to 40 is usually sufficient. However, if the number of application servers in a replication domain is greater ten and the number of replication domain consumers in each application server is greater than two, it might have to set the maximum thread pool size to a value greater than 40.

Results

Data is replicating among the application servers in a configured replication domain.

What to do next

If you select DES or 3DES as the encryption type for a replication domain, an encryption key is used for the encryption of messages. At regular intervals, for example once a month, you should go to the Environment > Replication domains > New page in the administrative console, and click Regenerate encryption key to regenerate the key. After the key is regenerated, you must restart all of the application servers that are configured as part of the replication domain. Periodically regenerating the key improves data security.

E) Manage Web servers in a managed and unmanaged node.

To administer or manage a Web server using the administrative console, you must create a Web server definition or object in the WebSphere® Application Server repository.

Web server definition

To administer or manage a Web server using the administrative console, you must create a Web server definition or object in the WebSphere® Application Server repository.

The creation of this object is exclusive of the actual installation of a Web server. The Web server object in the WebSphere Application Server repository represents the Web server for administering and managing the Web server from the administrative console.

The Web server object contains the following Web server properties:

* installation root
* port
* configuration file paths
* log file paths

In addition to Web server properties, the Web server contains a plug-in object. The plug-in object contains properties that define the plugin-cfg.xml file.

The definitions of the Web server object are made using the wsadmin command or the administrative console. You can also define a Web server object in the WebSphere Application Server repository using the profile create script during installation, a .jacl script, and by using the administrative console wizard.

There are three types of WebSphere Application Server nodes upon which you can create a Web server. The type depends on the version of WebSphere Application Server, as follows:

* Managed node. A node that contains a node agent. This node can exist only in a deployment manager environment. The importance of defining a Web server on a managed node is that the administration and configuration of the Web server is handled through the node agent from the administrative console. Support for administration and configuration through the administrative console is limited to IBM® HTTP Server only. Non-IBM HTTP Server Web servers must be on a managed node to handle plug-in administrative functions and the generation and propagation of the plugin-cfg.xml file.

* Stand-alone node. A node that does not contain a node agent. This node usually exists in an base or Express WebSphere Application Server environment. A stand-alone node can become a managed node in a deployment manager environment after the node is federated . A stand-alone node does not contain a node agent, so to administer and manage IBM HTTP Server, there must be an IBM HTTP Server administration server installed and running on the stand-alone machine that the node represents. IBM HTTP Server ships with the IBM HTTP Server administration server and is installed by default. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only.

* Unmanaged node. A node that is not associated with a WebSphere Application Server node agent. This node cannot be federated. Typically, the unmanaged node represents a remote machine that does not have WebSphere Application Server installed. However, you can define an unmanaged node on a machine where WebSphere Application Server is installed. This node can exist in a WebSphere Application Server – Express, base, or deployment manager environment. An unmanaged node does not contain a node agent, so to administer and manage IBM HTTP Server, an IBM HTTP Server administration server must be installed and running on the stand-alone machine that the node represents. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only.

Web servers, which are not IBM HTTP Servers for WebSphere Application Server, are not fully administered from the WebSphere Application Server administrative console. The administration functions for Web servers, which are not IBM HTTP Servers for WebSphere Application Server, are:

* On managed nodes:
    - o Web server status in the Web server collection panel or serverStatus.sh
    - o Generation of the plugin-cfg.xml

  - o Propagation of the plugin-cfg.xml

* On unmanaged nodes:
  - o Web server status in the Web server collection panel or serverStatus.sh
  - o Generation of the plugin-cfg.xml

Web server configuration

Plug-in configuration involves configuring the Web server to use the binary plug-in module that WebSphere® Application Server provides. Plug-in configuration also includes updating the plug-in XML configuration file to reflect the current application server configuration. The binary module uses the XML file to help route Web client requests.

After installing a supported Web server, you must install a binary plug-in module for the Web server. The plug-in module lets the Web server communicate with the application server. The Plug-ins installation wizard installs the Web server plug-in. The wizard configures the Web server. The wizard also creates a Web server definition in the configuration of the application server. The Plug-ins installation wizard uses the following files to configure a plug-in for the Web server that you select:

* The Web server configuration file on the Web server machine, such as the httpd.conf file for IBM® HTTP Server.
* The binary Web server plug-in file that the Plug-ins installation Wizard installs on the Web server machine.
* The plug-in configuration file, plugin-cfg.xml, on the application server machine that you propagate (copy) to a Web server machine.
* The default (temporary) plug-in configuration file, plugin-cfg.xml, on the Web server machine.
* The configureweb_server_name script that you copy from the Web server machine to the application server machine.

<u>Web server configuration file</u>

The Web server configuration file is installed as part of the Web server.

The wizard must reconfigure the configuration file for a supported Web server.

Configuration consists of adding directives that identify file locations of two files:

* The binary plug-in file
* The plugin-cfg.xml configuration file

<u>The binary Web server plug-in file</u>

An example of a binary plug-in module is the mod_ibm_app_server_http.dll file for IBM HTTP Server on the Windows® platform.

The binary plug-in file does not change. However, the configuration file for the binary plug-in is an XML file. The application server changes the configuration file when certain changes to your WebSphere Application Server configuration occur.

The binary module reads the XML file to adjust settings and to route requests to the application server.

The plug-in configuration file, plugin-cfg.xml

The plug-in configuration file is an XML file with settings that you can tune in the administrative console. The file lists all of the applications installed on the Web server definition. The binary module reads the XML file to adjust settings and to route requests to the application server.

The standalone application server regenerates the plugin-cfg.xml file in the profile_root/config/cells/cell_name/nodes/web_server_name_node/servers/web_server_name directory. Regeneration occurs whenever a change occurs in the application server configuration that affects deployed applications.

The deployment manager regenerates the plugin-cfg.xml file in the profile_root/config/cells/cell_name/nodes/node_name/servers/web_server_name directory whenever a change occurs in application server configuration that affects deployed applications on the managed node.

After regeneration, propagate (copy) the file to the Web server machine. The binary plug-in then has access to the most current copy of its configuration file.

The Web server plug-in configuration service automatically regenerates the plugin-cfg.xml file after certain events that change the configuration. The configuration service automatically propagates the plugin-cfg.xml file to an IBM HTTP Server machine when the file is regenerated. You must manually copy the file on other Web servers.

Default plug-in configuration file, plugin-cfg.xml

The Plug-ins installation wizard creates the temporary plugin-cfg.xml file in the plugins_root/config/web_server_name directory. The wizard creates the file for every remote installation scenario. The wizard creates the file at the same time that it installs the binary plug-in module for the Web server.

The default file is a placeholder that you must replace with the plugin-cfg.xml file from the Web server definition on the application server. The default file is a replica of the file that the application server creates for a default standalone application server that has the samples installed.

Run the configureweb_server_name script from the app_server_root/bin directory of the application server machine for a remote installation, or directly from the plugins_root/bin directory for a local installation. The script creates the Web server definition in the configuration files of the default profile. To configure a different profile than the default, edit the configureweb_server_name script. Use the -profileName parameter to identify a different profile than the default.

After the Web server definition is created, the Web server plug-in configuration service within the application server creates the first plugin-cfg.xml file in the Web server definition on the application server machine. If you install an application, create a virtual host, or do anything that changes the configuration, you must propagate the updated plugin-cfg.xml file from the application server machine to the Web server machine to replace the default file.

The configureweb_server_name script for the Web server definition

The Plug-ins installation wizard creates the configureweb_server_name script on the Web server machine in the plugins_root/bin directory. If one machine in a remote scenario is running under an operating system like AIX® or Linux® and the other machine is running under Windows, use the script created in the plugins_root/bin/crossPlatformScripts directory. The script is created for remote installation scenarios only.

Copy the script from the Web server machine to the app_server_root/bin directory on a remote application server machine. You do not have to copy the script on a local installation. Run the script to create a Web server definition in the configuration of the application server.

When using the IBM HTTP Server, configure the IBM HTTP Administration Server also. The IBM HTTP Administration Server works with the administrative console to manage Web server definitions. Also, use the administrative console to update your Web server definition with remote

Web server management options. Click Servers > Server Types > Web servers > web_server_name to see configuration options. For example, click Remote Web server management to change such properties as:

* Host name
* Administrative port
* User ID
* Password

Important: Always open a new command window before running this script. You can avoid a potential problem by doing so.

The problem is a potential conflict between a shell environment variable, the WAS_USER_SCRIPT environment variable, and the actual default profile. The script always works against the default profile. However, if the WAS_USER_SCRIPT environment variable is set, a conflict arises as the script attempts to work on the profile identified by the variable.

The variable is easy to set accidentally. Issue any command from the profile_root/bin directory of any profile and the variable is set to that profile.

If a Web server definition already exists for a standalone application server, running the script does not add a new Web server definition. Each standalone application server can have only one Web server definition.

You cannot use the administrative console of a standalone application server to add or delete a Web server definition. However, you can do both tasks using the administrative scripting interface:

* Add a Web server definition through the wsadmin facility using the configureweb_server_name script. The script uses a Java™ Command Language (Jacl) script named configureWebserverDefintion.jacl to create and configure the Web server definition.
* Delete a Web server definition using wsadmin commands. The Web server is named webserver1 in the following example:

```
set webserverName webserver1
set webserverNodeSuffix _node
set webserverNodeName   $webserverName$webserverNodeSuffix
$AdminConfig remove [$AdminConfig getid /Node:
$webserverNodeName/Server:$webserverName]
$AdminConfig remove [$AdminConfig getid /Node:$webserverNodeName]
$AdminConfig save
```

A managed node, on the other hand, can have multiple Web server definitions. The script creates a new Web server definition unless the Web server name is the same.

Replacing the default plug-in configuration file with the file from the Web server definition (propagation)

The default file uses fixed parameter values that might not match the parameter values in the actual file on the application server. The default file is a placeholder only.

The file cannot reflect changes that occur in the application server configuration. The file also cannot reflect non-default values that might be in effect on the application server.

The application server must have the following values in the actual plugin-cfg.xml file. If so, the default file can successfully configure the binary plug-in module. Then, the plug-in module can successfully communicate with the Web server and the application server.

Personal certificates contain a private key and a public key. You can extract the public key, called the signer certificate, to a file, then import the certificate into another keystore. The client requires the signer portion of a personal certificate for Security Socket Layer (SSL) communication.

Administrative functions for Web server nodes

WebSphere Application Server supports basic administrative functions for all supported Web servers. For example, the generation of a plug-in configuration can be performed for all Web servers. However, propagation of a plug-in configuration to remote Web servers is supported only for IBM® HTTP Servers that are defined on an unmanaged node. If the Web server is defined on a managed node, propagation of the plug-in configuration is done for all the Web servers by using node synchronization. The Web server plug-in configuration file is created according to the Web server definition and is based on the list of applications that are deployed on the Web server. You can also map all supported Web servers as potential targets for the modules during application deployment.

WebSphere Application Server supports some additional administrative console tasks for IBM HTTP Servers on managed and unmanaged nodes. For instance, you can start IBM HTTP Servers, stop them, terminate them, display their log files, and edit their configuration files.

F) Manage WebSphere messaging in a clustered environment.

A typical JMS messaging pattern involves a requesting application that sends a message to a JMS queue for processing by a messaging service, for example, a message-driven bean.

When the requesting application sends the request message, the message identifies another JMS queue to which the service should send a reply message. After sending the request message, the requesting application either waits for the reply message to arrive, or it reconnects later to retrieve the reply message.

The request and reply pattern requires the following conditions:

* The requesting application can identify, in the requesting message, where the service must send the reply message.

* The requesting application can consume the reply message from the reply location.

A JMS queue can refer to a service integration bus destination that is defined on a server bus member or cluster bus member.

* If the bus member is a server (which can have only one messaging engine), or a cluster with a single messaging engine, a JMS queue identifies a single service integration bus queue point.

* If the bus member is a cluster with multiple messaging engines (typically, to provide workload sharing or scalability), a JMS queue identifies multiple queue points; one on each messaging engine in the bus member.

The following behavior occurs by default:

* The queue point that an application sends messages to, or receives messages from, is determined by the messaging system.

* During its lifetime, a consumer, in this case a JMS message consumer, consumes from only one queue point.

This request and reply behavior allows a reply message to be sent to a different queue point from the one on which the requestor is waiting for it. This can result in the reply message not being received.

To overcome this situation, there are various options when you configure the service integration bus system or the JMS applications:

* Use a temporary queue as the reply queue.

* Use a scoped service integration bus alias destination to restrict messages to a single queue point.

* Restrict reply messages to the queue point that is local to the requesting application.

* Configure the requestor to consume messages from all queue points simultaneously.

# Using a temporary queue as a reply queue

JMS can create a temporary queue dynamically for use as a reply queue. You can use this to ensure that a reply message is sent to the appropriate queue point for a cluster bus member.

# Using a scoped service integration bus alias destination to restrict messages to a single queue point

You can use a service integration bus alias destination to target a service integration bus queue that has multiple queue points. You can do this to ensure that a reply message is sent to the appropriate queue point for a cluster bus member.

# Restricting reply messages to the queue point that is local to the requesting application

When a JMS queue identifies a service integration bus queue as the reply queue, and that service integration bus queue has multiple queue points, you can configure a JMS queue to restrict the messages to the queue point that is local to the application that referenced the JMS queue. You do this to ensure that a reply message is sent to the appropriate queue point for a cluster bus member.

# Configuring the requestor to consume messages from all queue points simultaneously

By default a JMS message consumer consumes from only one queue point for the lifetime of the message consumer. Therefore, if the reply queue has more than one queue point, unless the reply message is restricted to one particular queue

Consider the advantages and disadvantages of each option and the requirements of your application, before you choose an approach.

Use the simplest solution that satisfies the requirements of the request/reply scenario. For example:

* If reply messages are required only while the requesting application is initially connected, use non-persistent messages and temporary queues. Also consider setting a timeToLive of the initial

request message, if the requesting application will wait for a reply for only a finite time.

* If a single queue point (and its messaging engine) can handle all the reply message traffic for the requesting applications, but a cluster bus member with multiple messaging engines is required for other messaging traffic (for example, the request messages), use a service integration bus alias destination to scope the messages to that single queue point.

If necessary, you can combine these options to achieve the solution that best satisfies the requirements of your application and has the best performance and scalability.

For example, if requesting applications typically receive their reply messages during the initial connection but under certain rare conditions (for example, a failure) they have to reconnect to receive the reply, the following approach might be suitable:

* Enable the scopeToLocalQP option of the JMS queue, and allow the requesting application to connect to any of the messaging engines in the cluster bus member (that is, target the JMS connection factory at the bus member). This allows the connections to be workload balanced but restricts reply messages to the local queue point. The result is that reply messages can be found while using the same connection to receive the reply that was used to send the request.

* When re-connecting after a failure, enable the Message gathering option on the JMS queue so that the reply message can be received from wherever it is held.

This approach enables dynamic workload balancing of the requesting applications and minimizes the performance implications of message gathering by reducing its use to failure scenarios.

Configuration for workload sharing or scalability

This configuration consists of multiple messaging engines running in a cluster, with each messaging engine restricted to running on one particular server. A workload sharing configuration achieves greater throughput of messages by spreading the messaging load across multiple servers.

There are two ways to achieve this configuration:

* You can add a cluster to the service integration bus using messaging engine policy assistance, and use the scalability messaging engine policy. This procedure creates a messaging engine for each server in the cluster. Each messaging engine has just one preferred server and cannot fail over or fail back, that is, it is configured to run only on that server. New core group policies are automatically created, configured, and associated with each messaging engine.

* You can add a cluster to the service integration bus without using messaging engine policy assistance. One messaging engine is created automatically, then you add the further messaging engines that you require to the cluster, for example, one messaging engine for each server in the cluster.

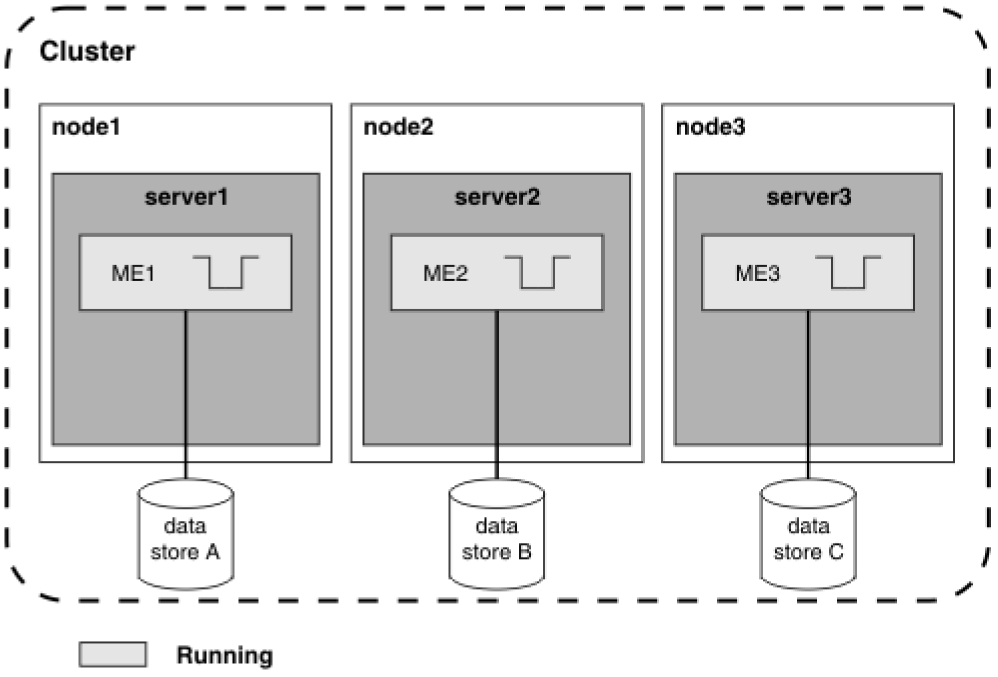
You create a core group policy for each messaging engine. Because no failover is required, you configure those policies so that each messaging engine is restricted to a particular server. To restrict a messaging engine to a particular server, you can configure a Static policy for each messaging engine.

After you create the new policies, use the match criteria to associate each policy with the required messaging engine.

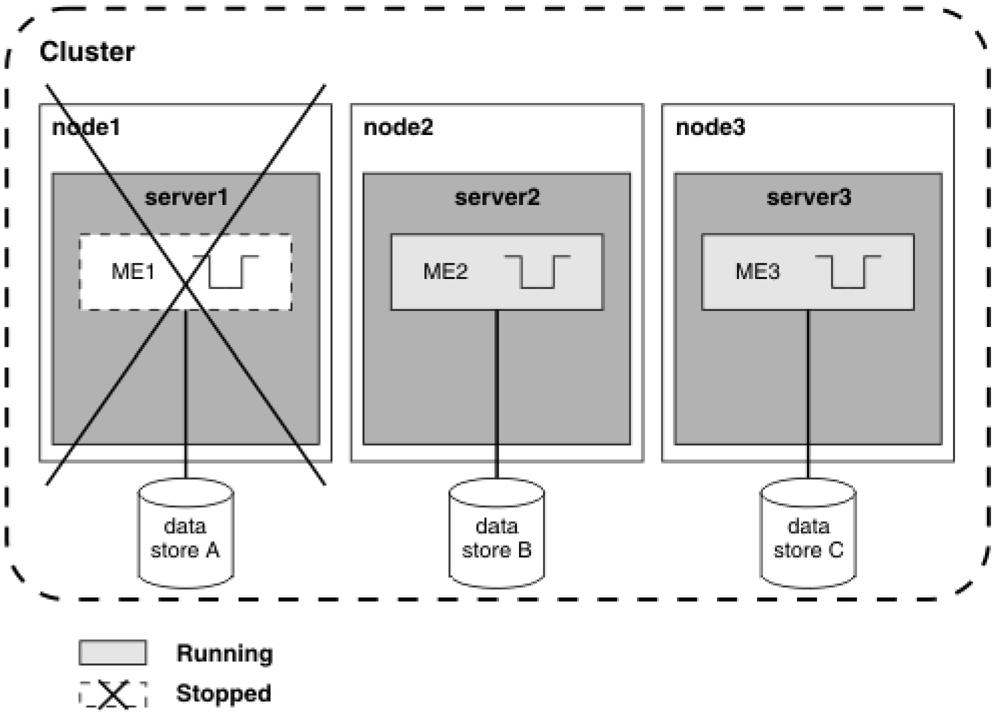
This type of deployment provides workload sharing through the partitioning of destinations across multiple messaging engines. This configuration does not enable failover, because each messaging engine can run on only one server. The impact of a failure is lower than in a simple deployment, because if one of the servers or messaging engines in the cluster fails, the remaining messaging engines still have operational destination partitions. However, messages being handled by a messaging engine in a failed server are unavailable until the server can be restarted.

The workload sharing configuration also provides scalability, because it is possible to add new servers to the cluster without affecting existing messaging engines in the cluster.

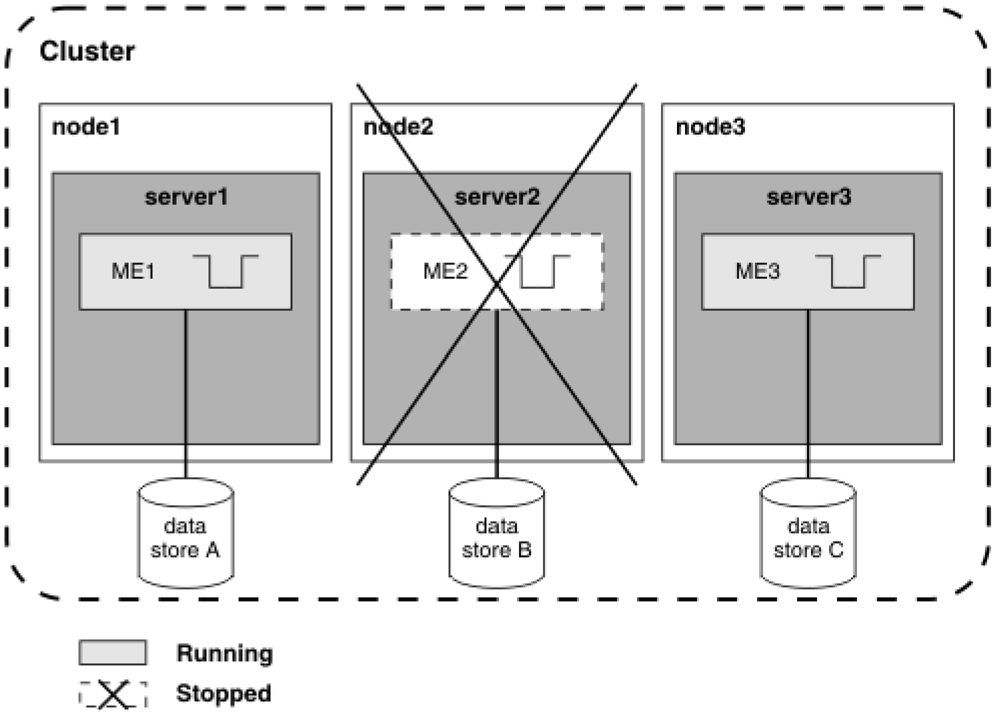
The following diagram shows a workload sharing or scalability configuration in which there are three messaging engines, ME1, ME2, and ME3, with data stores A, B, and C, respectively. The messaging engines run in a cluster of three servers and share the traffic passing through the destination. Each server is on a separate node, so that if one node fails, the servers on the remaining nodes are still available.



The following diagram shows what happens if server1 fails. ME1 cannot run, and data store A is not accessible. ME1 cannot process messages until server1 recovers. ME2 and ME3 are unaffected and continue to process messages. They will now handle all new traffic through the destination.



The following diagram shows what happens if server1 recovers and server2 fails. ME2 cannot run, and data store B is not accessible. ME2 cannot process messages until server2 recovers. ME1 and ME3 can process messages and will now handle all new traffic through the destination.



You can configure high availability and workload sharing of service integration without using messaging engine policy assistance.

Before you begin

Ensure that you want to use the following procedure. As an alternative, you can configure high availability and workload sharing of service integration by using messaging engine policy assistance when you add a server cluster to a bus. You create messaging engines and their associated policies as part of the procedure, by using the appropriate predefined messaging engine policy type. Alternatively, you can use a custom policy type and configure the messaging engine policy as you require, and the relevant core group policies and match criteria are created automatically.

About this task

When you set up a service integration environment, you create bus members, either servers or clusters, that run messaging engines. For high availability, where the messaging engine can fail over, or workload sharing, where multiple messaging engines share the load on a destination, you need to create a cluster bus member and configure high availability and workload sharing characteristics of the messaging engines.

If you do not require high availability or workload sharing, you can use a simple configuration and create a server bus member. You do not need the steps described in this topic.

The high availability and workload sharing characteristics of the messaging engines in the cluster are set by core group policies.

To see the policies that are configured in your system, you can use the administrative console to open the Policies page. In the navigation pane, click Servers > Core groups > Core group settings > core_group_name > [Additional Properties] Policies.

One of the available policies is the default service integration policy, "Default SIBus Policy", which is the policy that a messaging engine uses unless you configure the system so that the engine uses another policy. The default policy is sufficient for many purposes and you might not need to alter the policy configuration. It is not advisable to alter the default service integration policy, because those changes will affect all messaging engines that the policy manages. Therefore, it is better to create and configure one or more new specific policies.

Procedure

1. Optional: Create a cluster, if it is not created already. See Creating clusters.
2. Add the cluster to the service integration bus. See Adding a cluster to a bus without using messaging engine policy assistance.

A single messaging engine that uses the default service integration policy is created automatically.

For high availability without workload sharing, you can use this configuration and do not need to change it further. If you want to configure the messaging engine behavior further, for example to specify preferred servers for the messaging engine, or enable the messaging engine to fail back, complete step 3.

3. Optional: For high availability when you want to configure the messaging engine behavior, create and configure a policy for the messaging engine. Create a policy with the type "One of N" . See Creating a policy for messaging engines and Configuring a core group policy for messaging engines.

4. Optional: For workload sharing without high availability, use the following steps:

1. Add as many messaging engines as you require to the cluster. Typically, a workload sharing configuration has one messaging engine for each server in the cluster. See Adding a messaging engine to a cluster.

2. Create and configure a policy for each messaging engine in the cluster. Create policies with the type Static. See Creating a policy for messaging engines and Configuring a core group policy for messaging engines.

5. Optional: For workload sharing with high availability, use the following steps:

1. Add as many messaging engines as you require to the cluster. Typically, a workload sharing configuration has one messaging engine for each server in the cluster. See Adding a messaging engine to a cluster.

2. Create and configure a policy for each messaging engine in the cluster. Create policies with the type "One of N". See Creating a policy for messaging engines and Configuring a core group policy for messaging engines.

6. Optional: To use an external high availability (HA) framework to manage high availability or workload sharing behavior, use the following steps:

1. If you require workload sharing, add as many messaging engines as you require to the cluster. Typically, a workload sharing configuration has one messaging engine for each server in the cluster. See Adding a messaging engine to a cluster.

2. Create and configure one policy for the messaging engines in the cluster. Create a policy with the type "No operation". See Creating a policy for messaging engines and Configuring a core group policy for messaging engines.

What to do next

If you created a high availability configuration for service integration, you might also want to configure high availability for the transaction service.

If you created a workload sharing configuration, you might want to deploy a queue destination to the cluster, so that the queue is partitioned across the set of messaging engines.

Subtopics

* Creating a policy for messaging engines

You create one or more core group policies for service integration to control the behavior of the messaging engine, or engines, in a server cluster. The policies support behavior such as high availability, workload sharing or scalability in a server cluster.

* Configuring a core group policy for messaging engines

You can configure a core group policy for service integration to associate the policy with specific messaging engines and to specify messaging engine behavior, such as which server a messaging engine runs on, whether a messaging engine can fail over or fail back, and the frequency of messaging engine monitoring.

* Configuring messaging engine failover for mixed version clusters

A messaging engine that is hosted on a WebSphere® Application Server Version 7.0 server cannot fail over to a messaging engine that is hosted on a WebSphere Application Server

Version 6 server. If you have a cluster bus member that consists of a mixture of Version 6 and Version 7.0 servers, you must make sure the high availability policy is configured to prevent this type of failover.

Scalability with high availability messaging engine policy

The scalability with high availability messaging engine policy is a predefined messaging engine policy type that is provided when you use messaging engine policy assistance. It helps you to configure a cluster that is a member of a bus when you require both high availability and scalability in the cluster.

The scalability with high availability configuration ensures that there is a messaging engine for each server in a cluster, and that each messaging engine has a failover location.

The scalability with high availability messaging engine policy creates a single messaging engine for each server in the cluster. Each messaging engine can fail over to one other specified server in the cluster. Each server can host up to two messaging engines, such that there is an ordered circular relationship between the servers. Each messaging engine can fail back, that is, if a messaging engine fails over to another server, and then the original server becomes available again, the messaging engine automatically moves back to that server.

Each messaging engine is assigned to a specific server by configuring it to run only on servers in its list of preferred servers, then specifying only two servers in that preferred servers list. Each server is the first preferred server for one messaging engine and the second preferred server for another one, which creates the circular relationship between the servers. Failback is enabled so that each messaging engine is always hosted by its preferred server if that server is running.

Both servers that can host a specific messaging engine must be able to access the message store (either a database or a file system) that is configured for that messaging engine.

Use the scalability with high availability policy for a system where you want to add more servers to a cluster without affecting the existing messaging engines, but you also want to ensure that messaging is always available.

When you select the scalability with high availability messaging engine policy type on the administrative console, a diagram shows the selected cluster and the eventual outcome of the policy.

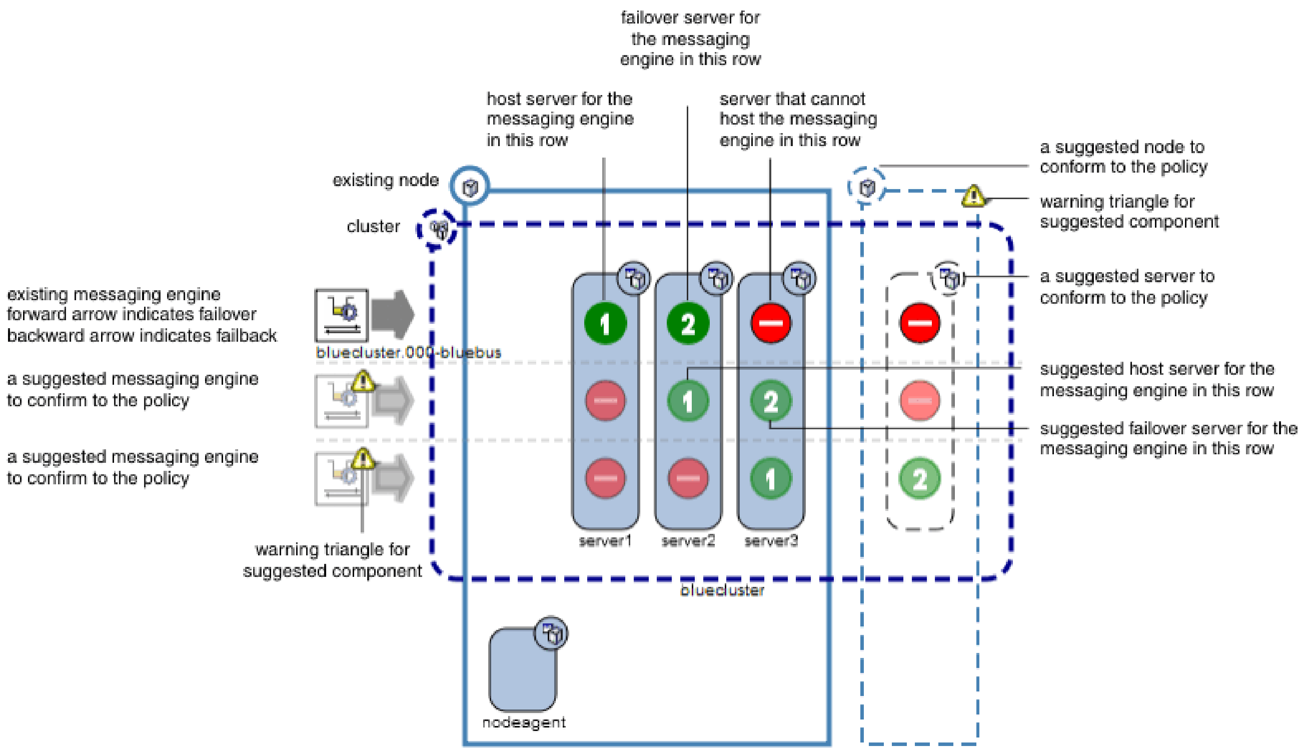
If there are no warning triangles in the diagram, and the "Is further configuration required" column shows No in the Scalability with high availability row, the topology of the cluster and the configuration of the messaging engine is suitable, and you can continue.

If there are warning triangles in the diagram, examine the messages in the Scalability with high availability row for guidance on how to achieve a suitable messaging engine configuration.

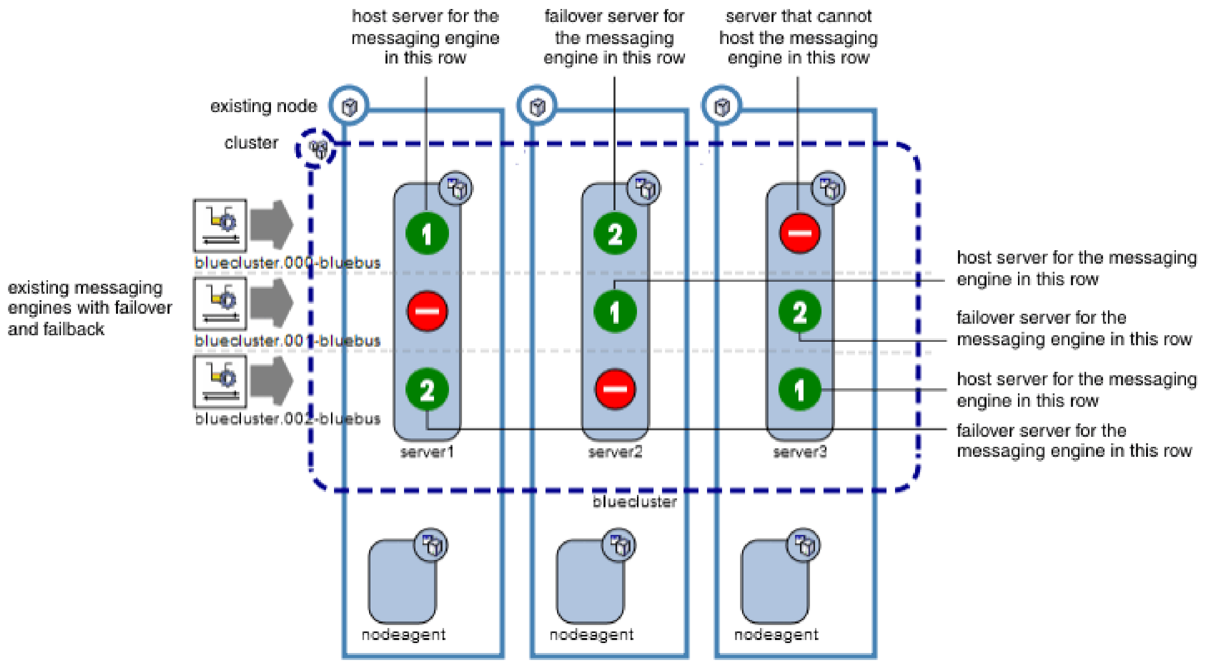
If you require high availability in a cluster, that cluster should contain at least two nodes, each with a server on it, that is, there should be at least two separate physical machines in the cluster.

If the messages advise you to add another server on another node, you must go back and redefine the topology of the cluster before you add the cluster as a member of a bus.

For example, the following figure shows three servers configured on one node. If that node fails, there will be no servers available for any of the messaging engines to fail over to. To provide some high availability, there must be at least one other server on a separate node to ensure that there is a server on which at least one messaging engine can run. Also, there is only one messaging engine configured. To provide some scalability, there must be one messaging engine for each server.


failover server for the messaging engine in this row
host server for the messaging engine in this row
server that cannot host the messaging engine in this row
existing node
cluster
a suggested node to conform to the policy
warning triangle for suggested component
a suggested server to conform to the policy
existing messaging engine forward arrow indicates failover backward arrow indicates failback
bluecluster.000-bluebus
a suggested messaging engine to confirm to the policy
suggested host server for the messaging engine in this row
a suggested messaging engine to confirm to the policy
suggested failover server for the messaging engine in this row
warning triangle for suggested component
server1
server2
server3
bluecluster
nodeagent

The figure two is an example of when the messaging engine configuration is suitable for the scalability with high availability policy. There are three servers, each on a separate node, and three messaging engines. Each messaging engine has a preferred server and one other server it can use for failover. Each server is the preferred host for one messaging engine, and the failover host for one other messaging engine. There are no warning triangles and no faded out components because the policy can be used successfully.


host server for the messaging engine in this row
failover server for the messaging engine in this row
server that cannot host the messaging engine in this row
existing node
cluster
existing messaging engines with failover and failback
bluecluster.000-bluebus
bluecluster.001-bluebus
bluecluster.002-bluebus
server1
server2
server3
bluecluster
nodeagent
nodeagent
nodeagent
host server for the messaging engine in this row
failover server for the messaging engine in this row
host server for the messaging engine in this row
failover server for the messaging engine in this row

The following table shows the messaging engine policy settings for a cluster of three servers that use the scalability with high availability messaging engine policy.

Messaging engine name	Failover	Failback	Preferred servers list	Only run on preferred servers
clustername.000-busname	true	true	* server1 * server2	true
clustername.001-busname	true	true	* server2 * server3	true
clustername.002-busname	true	true	* server3 * server1	true

The predefined scalability with high availability messaging engine policy creates a configuration with aspects of scalability and high availability. It is not the only way to configure a cluster to provide scalability and high availability, but it is a frequently-used configuration. If you have other requirements, for example, message transmission is a priority and you want to increase the number of possible locations for each messaging engine, you can use the custom messaging engine policy.

Tuning messaging performance with service integration technologies

To help optimize performance, you can set tuning properties that control the performance of message-driven beans and other messaging applications deployed to use service integration technologies.

About this task

To optimize the performance of messaging with service integration technologies, you can use the administrative console to set various parameters as described in the steps below. You can also set these parameters by using the wsadmin tool.

Procedure

* View the Available Message Count on a destination.

Viewing the Available Message Count on a destination enables you to determine whether your message consumers are able to cope with your current workload. If the available message count on a given destination is too high, or is increasing over time, consider some of the tuning recommendations in this topic.

1. Enable AvailableMessageCount statistics for a queue. If you restart the administrative server, enable AvailableMessageCount statistics again because such runtime settings are not preserved when the server is restarted.

    1. In the navigation pane, click Monitoring and Tuning > Performance Monitoring Infrastructure (PMI).
    2. In the content pane, click server_name.
    3. Click the Runtime tab.
    4. In the Currently monitored statistic set, click Custom.
    5. On the Custom monitoring level panel, click SIB Service > SIB Messaging Engines > engine_name > Destinations > Queues > queue_name.
    6. Select the AvailableMessageCount option.
    7. Click Enable at the top of the panel.

2. View the available message count for a queue.

    1. In the navigation pane, click Monitoring and Tuning > Performance Viewer > Current activity.
    2. In the content pane, click server_name.
    3. Click Performance Modules > SIB Service > SIB Messaging Engines > engine_name > Destinations > Queues > queue_name.
    4. Click View Module(s) at the top of the Resource Selection panel, located on the left side. This displays the AvailableMessageCount data in the Data Monitoring panel, located on the right side.

You can use the Data Monitoring panel to manage the collection of monitoring data; for example, you can use the buttons to start or stop logging, or to change the data displayed as either a table or graph.

* Monitor MDB Thread Pool Size for the Default Message Provider.

You might experience a performance bottleneck if there are insufficient threads available for the message-driven beans. There is a trade-off between providing sufficient threads to maximize the throughput of messages and configuring too many threads, which can lead to CPU starvation of the threads in the application server. If you notice that the throughput for express nonpersistent, reliable nonpersistent, or reliable persistent messaging has fallen as a result of increasing the size of the default thread pool, then decrease the size of the thread pool and reassess the message throughput.

1. View or change the number of threads in the default thread pool for an application server. By default, message-driven beans use the default thread pool.

    1. Click Servers > Server Types > WebSphere application servers > server_name > [Additional Properties] Thread Pools > Default. By default the Minimum size value is set to 5 and the Maximum size value is set to 20. The best performance is obtained by setting the Maximum size value to the expected maximum concurrency for all message-driven beans. For high throughput using a single message bean, 41 was found to be the optimal Maximum size value.
    2. Change the Maximum size value, then click OK.

2. Optional: Create your own thread pool. The default thread pool is also used by other WebSphere® Application Server components, so you might want to define a separate thread pool for the message-driven beans. This reduces thread contention for the default thread pool.

    1. Click Servers > Server Types > WebSphere application servers > server_name > [Additional Properties] Thread Pools.

2. Create a new thread pool.

3. Create sufficient threads to support the maximum amount of concurrent work for the message-driven beans.

4. Change the SIB JMS Resource Adapter to use the new thread pool:

1. Click Resources > Resource Adapters > Resource adapters.

2. If you cannot see any SIB JMS Resource Adapter instances in the list, expand Preferences and enable Show built-in resources.

3. Select the SIB JMS Resource Adapter with the appropriate scope depending upon the scope of the connection factories.

4. Add the name of the new thread pool in the Thread pool alias box.

5. Click Apply .

3. Save your changes to the master configuration.

* Tune MDB performance with the default messaging provider.

1. Click Resources > JMS > Activation specifications > activation_specification_name.

2. Set the maximum batch size for this activation specification.

Delivering batches of messages to each MDB endpoint can improve performance, particularly when used with Acknowledge mode set to Duplicates-ok auto-acknowledge. However, if message ordering must be retained across failed deliveries, set this parameter to 1.

3. Set the maximum number of concurrent endpoints for this activation specification.

The maximum concurrent endpoints parameter controls the amount of concurrent work that can be processed by a message bean. The parameter is used with message-driven beans. Increasing the number of concurrent endpoints can improve performance but can increase the number of threads in use at one time. To benefit from a change in this parameter, there should be sufficient threads available in the MDB thread pool to support the concurrent work. However, if message ordering must be retained across failed deliveries, set this parameter to 1.

4. Save your changes to the master configuration.

For additional information about tuning the throttling of message-driven beans, including controlling the maximum number of instances of each message bean and the message batch size for serial delivery, see Configuring MDB throttling for the default messaging provider.

* For distributed platformsFor IBM i platforms Ensure that application servers hosting one or more messaging engines are provided with an appropriate amount of memory for the message throughput you require.

You can tune the initial and maximum Java™ Virtual Machine (JVM) heap sizes when adding a server to a messaging bus, that is when you create a messaging engine. You have the option to do this in any of the following cases:

When adding a single server to a bus

When adding a cluster to a bus

When adding a new server to an existing cluster that is itself a bus member

For an application server that is a bus member of at least one bus, or a member of a cluster that is a bus member of at least one bus, the recommended initial and maximum heap sizes are 768MB.

When you are adding a cluster to a bus, you are recommended to increase the initial and maximum JVM heap sizes for every server in the cluster to 768MB.

Increasing the initial heap size improves the performance for small average message sizes

Increasing the maximum heap size improves the performance for higher average message sizes

* Reduce the occurrence of OutOfMemoryError exceptions

If the cumulative size of the set of messages being processed within a transaction by the service integration bus is large enough to exhaust the JVM heap, OutOfMemoryError exceptions occur. Consider one of the following options for reducing the occurrence of OutOfMemoryError exceptions when processing a large set of messages within a transaction.

Increase the JVM heap sizes for the application server.

Reduce the cumulative size of the set of messages being processed within the transaction.

* Change the maximum connections in a connection factory for the default messaging provider.

The maximum connections parameter limits the number of local connections. The default is 10. This parameter should be set to a number equal to or greater than the number of threads (enterprise beans) concurrently sending messages.

1. Click Resources > JMS > Topic connection factories > factory_name > [Additional Properties] Connection pool properties.
2. Enter the required value in the Maximum connections field.
3. Click Apply.
4. Save your changes to the master configuration.

* Tune reliability levels for messages.

The reliability level chosen for the messages has a significant impact on performance. In order of decreasing performance (fastest first), the reliability levels are:

Best effort nonpersistent

Express nonpersistent

Reliable nonpersistent

Reliable persistent

Assured persistent

For MDB point-to-point messaging, best effort nonpersistent throughput is more than six times greater than assured persistent. For more information about reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

G) Configure workload management via the web server plug-in

Important tips for Web server plug-in tuning include how to balance workload and improve performance in a high stress environment. Balancing workloads among application servers in a network fronted by a Web server plug-in helps improve request response time.

During normal operation, the backlog of connections pending to an application server is bound to grow. Therefore, balancing workloads among application servers in a network fronted by a Web server plug-in helps improve request response time.

You can limit the number of connections that can be handled by an applications server. To do this:

1. Go to the Servers > Server Types > WebSphere application servers > server_name.

2. In the Additional Properties section, click Web Server Plug-in properties .

3. Select Use maximum number of connections for the Maximum number of connections that can be handled by the Application server field.

4. Specify in the Connections field the maximum number of connections that you want to allow.

5. Then click Apply and Save.

When this maximum number of connections is reached, the plug-in, when establishing connections, automatically skips that application server, and tries the next available application server. If no application servers are available, an HTTP 503 response code will be returned to the client. This code indicates that the server is currently unable to handle the request because it is experiencing a temporary overloading or because maintenance is being performed.

The capacity of the application servers in the network determines the value you specify for the maximum number of connections. The ideal scenario is for all of the application servers in the network to be optimally utilized. For example, if you have the following environment:

* There are 10 application servers in a cluster.

* All of these application servers host the same applications (that is, Application_1 and Application_2).

* This cluster of application servers is fronted by five IBM HTTP Servers.

* The IBM HTTP Servers get requests through a load balancer.

* Application_1 takes approximately 60 seconds to respond to a request

* Application_2 takes approximately 1 second to respond to a request.

Depending on the request arrival pattern, all requests to Application_1 might be forwarded to two of the application servers, say Appsvr_1 and Appsvr_2. If the arrival rate is faster than the processing rate, the number of pending requests to Appsvr_1 and Appsvr_2 can grow.

Eventually, Appsvr_1 and Appsvr_2 are busy and are not able to respond to future requests. It usually takes a long time to recover from this overloaded situation.

If you want to maintain 2500 connections, and optimally utilize the Application Servers in this example, set the number of maximum connections allowed to 50. (This value is arrived at by dividing the number of connections by the result of multiplying the number of Application Servers by the number of Web servers; in this example, 2500/(10x5)=50.)

Limiting the number of connections that can be established with an application server works best for Web servers that follow use a single, multithreaded process for serving requests.

[Windows] IBM HTTP Server uses a single, multithreaded process for serving requests. No configuration changes are required.

[AIX HP-UX Solaris] [z/OS] IBM HTTP Server typically uses multiple multithreaded processes for serving requests. Specify the following values for the properties in the Web server configuration file (httpd.conf) to prevent the IBM HTTP Server from using more than one process for serving requests.

The WebSphere® Application Server Web server plug-ins are used to establish and maintain persistent HTTP and HTTPS connections to Application Servers .

When the plug-in is ready to send a request to the application server, it first checks its connection pool for existing connections. If an existing connection is available the plug-in checks its connection status. If the status is still good, the plug-in uses that connection to send the request. If a connection does not exist, the plug-in creates one. If a connection exists but has been closed by the application server, the plug-in closes that connection and opens a new one.

After a connection is established between a plug-in and an application server, it will not be closed unless the application server closes it for one of the following reasons:

* If the Use Keep-Alive property is selected and the time limit specified on the Read timeout or Write timeout property for the HTTP inbound channel has expired.

* The maximum number of persistent requests which can be processed on an HTTP inbound channel has been exceeded. This number is set using the Maximum persistent requests property that is specified for the HTTP inbound channel.

* The Application Server is shutting down.

Even if the application server closes a connection, the plug-in will not know that it has been closed until it tries to use it again. The connection will be closed if one of the following events occur:

* The plug-in receives a new HTTP request and tries to reuse the existing connection.

* The number of httpd processes drop because the Web server is not receiving any new HTTP requests. For the IBM® HTTP Server, the number of httpd processes that are kept alive depends on the value specified on the Web server's MinSpareServers directive.

* The Web server is stopped and all httpd processes are terminated, and their corresponding sockets are closed.

Avoid trouble: Sometimes, if a heavy request load is stopped or decreased abruptly on a particular application server, a lot of the plug-in's connections to that application server will be in CLOSE_WAIT state. Because these connections will be closed the first time the plug-in tries to reuse them, having a large number of connections in CLOSE-WAIT state should not affect performance.

<u>LoadBalanceWeight (zero or one attribute for each Server)</u>

Specifies the weight associated with this server when the plug-in does weighted Round Robin load balancing. The starting value for a server can be any integer between 0 and 20. However, zero should be specified only for a server that is shut down.

The LoadBalanceWeight for each server is decremented for each request processed by that server. After the weight for a particular server in a server cluster reaches zero, only requests with session affinity are routed to that server. When all servers in the cluster reach a weight of zero, the weights for all servers in the cluster are reset, and the algorithm starts over.

When a server is shut down, it is recommended that you set the weight for that server to zero. The plug-in can then reset the weights of the servers that are still running, and maintain proper load balancing.

Section 6 - Maintenance, Performance Monitoring and Tuning (14%)

A) Manage application configurations (e.g., application bindings, HTTP session tuning, etc.).

<u>Application bindings</u>

Before an application that is installed on an application server can start, all enterprise bean (EJB) references and resource references defined in the application must be bound to the actual artifacts (enterprise beans or resources) defined in the application server.

When defining bindings, you specify Java™ Naming and Directory Interface (JNDI) names for the referenceable and referenced artifacts in an application. The jndiName values specified for artifacts must be qualified lookup names. An example referenceable artifact is an EJB defined in an application. An example referenced artifact is an EJB or a resource reference used by the application.

Binding definitions are stored in the ibm-xxx-bnd.xml or ibm-xxx-bnd.xmi files of an application. Version 7.0 binding definitions support files with the suffix of XML for EJB 3.0 and Web 2.5 modules. Modules earlier than Java EE 5 continue to use binding definition files with the suffix of XMI as in previous versions of WebSphere® Application Server. The xxx can be ejb-jar, web, application or application-client.

For transitioning users: For EJB 3.0 modules, you do not need to specify JNDI binding names for each of the home or business interfaces on your enterprise beans. If you do not explicitly assign bindings, the EJB container assigns default bindings. Further, binding definitions are stored in ibm-ejb-jar-bnd.xml.trns

This topic provides the following information about bindings:

* Times when bindings can be defined
* Required bindings
* Other bindings that might be needed

Times when bindings can be defined

You can define bindings at the following times:

* During application development

An application developer can create binding definitions in ibm-xxx-bnd.xml files for EJB 3.0 and Web 2.5 modules and in ibm-xxx-bnd.xmi files for pre-Java EE 5 modules. The application developer can create the files using a tool such as an IBM® Rational® developer tool or, for EJB 3.0 or Web 2.5 modules, using an XML editor or text editor. The developer then gives an enterprise application (.ear file) complete with bindings to an application assembler or deployer. When assembling the application, the assembler does not modify the bindings. Similarly, when installing the application onto a server supported by WebSphere Application Server, the deployer does not modify or override the bindings or generate default bindings unless changes to the bindings are necessary for successful deployment of the application.

* During application assembly

An application assembler can define bindings in annotations or deployment descriptors of an application. Java EE 5 modules contain annotations in the source code. To declare an annotation, an application assembler precedes a keyword with an @ character ("at" sign). Bindings for pre-Java EE 5 modules are specified in the WebSphere Bindings section of a deployment descriptor editor. Modifying the deployment descriptors might change the binding definitions in the ibm-xxx-bnd.xmi files created when developing an application. After defining the bindings, the assembler gives the application to a deployer. When installing the application onto a server supported by WebSphere Application Server, the deployer does not modify or override the bindings or generate default bindings unless changes to the bindings are necessary to deploy the application.

* During application installation

An application deployer or server administrator can modify the bindings when installing the application onto a server supported by WebSphere Application Server using the administrative console. New binding definitions can be specified on the installation wizard pages.

Also, a deployer or administrator can select to generate default bindings during application installation. Selecting Generate default bindings during application installation instructs the product to fill in incomplete bindings in the application with default values. Existing bindings are not changed.

Restriction: You cannot define or override bindings during application installation for application clients. You must define bindings for application client modules during assembly and store the bindings in the ibm-application-client-bnd.xmi file.

* During configuration of an installed application

After an application is installed onto a server supported by WebSphere Application Server, an application deployer or server administrator can modify the bindings by changing values in administrative console pages such as those accessed from the settings page for the enterprise application.

Required bindings

Before an application can be successfully deployed, bindings must be defined for references to the following artifacts:

EJB JNDI names

For each EJB 2.1 or earlier enterprise bean (EJB), you must specify a JNDI name. The name is used to bind an entry in the global JNDI name space for the EJB home object. An example JNDI name for a Product EJB in a Store application might be store/ejb/Product. The binding definition is stored in the META-INF/ibm-ejb-jar-bnd.xmi file.

If a deployer chooses to generate default bindings when installing the application, the installation wizard assigns EJB JNDI names having the form prefix/EJB_name to incomplete bindings. The default prefix is ejb, but can be overridden. The EJB_name is as specified in the deployment descriptor <ejb-name> tag.

During and after application installation, EJB JNDI names can be specified on the Provide JNDI names for beans page. After installation, click Applications > Application Types > WebSphere enterprise applications > application_name > EJB JNDI names in the administrative console.

You do not need to specify JNDI binding names for each of the EJB 3.0 home or business interfaces on your enterprise beans because the EJB container assigns default bindings.

Data sources for entity beans

Entity beans such as container-managed persistence (CMP) beans store persistent data in data stores. With CMP beans, an EJB container manages the persistent state of the beans. You specify which data store a bean uses by binding an EJB module or an individual enterprise bean to a data source. Binding an EJB module to a data source causes all entity beans in that module to use the same data source for persistence.

An example JNDI name for a Store data source in a Store application might be store/jdbc/store. For modules earlier than Java EE 5, the binding definition is stored in IBM binding files such as ibm-ejb-jar-bnd.xmi. A deployer can also specify whether authentication is handled at the container or application level.

WebSphere Application Server Version 7.0 supports CMP beans in EJB 2.x or 1.x modules. Version 7.0 does not support CMP beans in EJB 3.0 modules.

If a deployer chooses to generate default bindings when installing the application, the install wizard generates the following for incomplete bindings:

* For EJB 2.x .jar files, connection factory bindings based on the JNDI name and authorization information specified

* For EJB 1.1 .jar files, data source bindings based on the JNDI name, data source user name and password specified

The generated bindings provide default connection factory settings for each EJB 2.x .jar file and default data source settings for each EJB 1.1 .jar file in the application being installed. No bean-level connection factory bindings or data source bindings are generated unless they are specified in the custom strategy rule supplied during default binding generation.

During and after application installation, you can map data sources to 2.x entity beans on the 2.x CMP bean data sources page and on the 2.x entity bean data sources page. After installation, click Applications > Application Types > WebSphere enterprise applications > application_name in the administrative console, then select 2.x CMP bean data sources or 2.x entity bean data sources. You can map data sources to 1.x entity beans on the Map data sources for all 1.x CMP beans page and on the Provide default data source mapping for modules containing 1.x entity beans page. After installation, access console pages similar to those for 2.x CMP beans, except click links for 1.x CMP beans.

<u>Backend ID for EJB modules</u>

If an EJB .jar file that defines CMP beans contains mappings for multiple backend databases, specify the appropriate backend ID that determines which persister classes are loaded at run time.

Specify the backend ID during application installation. You cannot select a backend ID after the application is installed onto a server.

To enable backend IDs for individual EJB modules:

1. During application installation, select Deploy enterprise beans on the Select installation options page. Selecting Deploy enterprise beans enables you to access the Provide options to perform the EJB Deploy page.

2. On the Provide options to perform the EJB Deploy page, set the database type to "" (null).

During application installation, if you select Deploy enterprise beans on the Select installation options page and specify a database type for the EJB deployment tool on the Provide options to perform the EJB Deploy page, previously defined backend IDs for all of the EJB modules are overwritten by the chosen database type.

The default database type is DB2UDB_V81.

The EJB deployment tool does not run during installation of EJB 3.0 modules.

For information on backend databases, refer to EJB deployment tool. For information on EJB Deploy options, refer to The ejbdeploy command.

EJB references

An enterprise bean (EJB) reference is a logical name used to locate the home interface of an enterprise bean. EJB references are specified during deployment. At run time, EJB references are bound to the physical location (global JNDI name) of the enterprise beans in the target operational environment. EJB references are made available in the java:comp/env/ejb Java naming subcontext.

The product assigns default JNDI values for or automatically resolves incomplete EJB 3.0 reference targets.

For each EJB 2.1 or earlier EJB reference, you must specify a JNDI name. An example JNDI name for a Supplier EJB reference in a Store application might be store/ejb/Supplier. The binding definition is stored in IBM binding files such as ibm-ejb-jar-bnd.xmi. When the referenced EJB is also deployed in the same application server, you can specify a server-scoped JNDI name. But if the referenced EJB is deployed on a different application server or if ejb-ref is defined in an application client module, then you should specify the global cell-scoped JNDI name.

If a deployer chooses to generate default bindings when installing the application, the install wizard binds EJB references as follows: If an <ejb-link> is found, it is honored. If the ejb-name of an EJB defined in the application matches the ejb-ref name, then that EJB is chosen. Otherwise, if a unique EJB is found with a matching home (or local home) interface as the referenced bean, the reference is resolved automatically.

During and after application installation, you can specify EJB reference JNDI names on the Map EJB references to beans page. After installation, click Applications > Application Types > WebSphere enterprise applications > application_name > EJB references in the administrative console.

Best practice: To enable EJB reference targets to resolve automatically if the references are from EJB 2.1 or earlier modules or from Web 2.3 or earlier modules, select Generate default bindings on the Preparing for application installation page or select Allow EJB reference targets to resolve automatically on the Select installation options, Map EJB references to beans, or EJB references console pages.bprac

Resource references

A resource reference is a logical name used to locate an external resource for an application. Resource references are specified during deployment. At run time, the references are bound to the physical location (global JNDI name) of the resource in the target operational environment. Resource references are made available as follows

Resource reference type	Subcontext declared in
Java DataBase Connectivity (JDBC) data source	java:comp/env/jdbc
JMS connection factory	java:comp/env/jms
JavaMail connection factory	java:comp/env/mail

Uniform Resource Locator (URL) connection factory	java:comp/env/url

For each resource reference, you must specify a JNDI name. If a deployer chooses to generate default bindings when installing the application, the install wizard generates resource reference bindings derived from the <res-ref-name> tag, assuming that the java:comp/env name is the same as the resource global JNDI name.

During application installation, you can specify resource reference JNDI names on the Map resource references to references page. Specify JNDI names for the resources that represent the logical names defined in resource references. You can optionally specify login configuration name and authentication properties for the resource. After specifying authentication properties, click OK to save the values and return to the mapping step. Each resource reference defined in an application must be bound to a resource defined in your WebSphere Application Server configuration. After installation, click Applications > Application Types > WebSphere enterprise applications > application_name > Resource references in the administrative console to access the Resource references page.

Virtual host bindings for Web modules

You must bind each Web module to a specific virtual host. The binding informs a Web server plug-in that all requests that match the virtual host must be handled by the Web application. An example virtual host to be bound to a Store Web application might be store_host. The binding definition is stored in IBM binding files such as WEB-INF/ibm-web-bnd.xmi.

If a deployer chooses to generate default bindings when installing the application, the install wizard sets the virtual host to default_host for each .war file.

During and after application installation, you can map a virtual host to a Web module defined in your application. On the Map virtual hosts for Web modules page, specify a virtual host. The port number specified in the virtual host definition is used in the URL that is used to access artifacts such as servlets and JavaServer Pages (JSP) files in the Web module. For example, an external URL for a Web artifact such as a JSP file is http://host_name:virtual_host_port/context_root/jsp_path. After installation, click Applications > Application Types > WebSphere enterprise applications > application_name > Virtual hosts in the administrative console.

Message-driven beans

For each message-driven bean, you must specify a queue or topic to which the bean will listen. A message-driven bean is invoked by a Java Messaging Service (JMS) listener when a message arrives on the input queue that the listener is monitoring. A deployer specifies a listener port or JNDI name of an activation specification as defined in a connector module (.rar file) under WebSphere Bindings on the Beans page of an assembly tool EJB deployment descriptor editor. An example JNDI name for a listener port to be used by a Store application might be StoreMdbListener. The binding definition is stored in IBM bindings files such as ibm-ejb-jar-bnd.xmi.

If a deployer chooses to generate default bindings when installing the application, the install wizard assigns JNDI names to incomplete bindings.

* For EJB 2.0 or later message-driven beans deployed as JCA 1.5-compliant resources, the install wizard assigns JNDI names corresponding to activationSpec instances in the form eis/MDB_ejb-name.

* For EJB 2.0 or later message-driven beans deployed against listener ports, the listener ports

are derived from the message-driven bean <ejb-name> tag with the string Port appended.

During application installation using the administrative console, you can specify a listener port name or an activation specification JNDI name for every message-driven bean on the Bind listeners for message-driven beans page. A listener port name must be provided when using the JMS providers: Version 5 default messaging, WebSphere MQ, or generic. An activation specification must be provided when the application's resources are configured using the default messaging provider or any generic J2C resource adapter that supports inbound messaging. If neither is specified, then a validation error is displayed after you click Finish on the Summary page. Also, if the module containing the message-driven bean is deployed on a 5.x deployment target and a listener port is not specified, then a validation error is displayed after you click Next.

After application installation, you can specify JNDI names and configure message-driven beans on console pages under Resources > JMS > JMS providers or under Resources > Resource adapters.

Restriction: You can only bind message driven-beans that are defined in an EJB 3.0 module to an activation specification.

Message destination references

A message destination reference is a logical name used to locate an enterprise bean in an EJB module that acts as a message destination. Message destination references exist only in J2EE 1.4 and later artifacts such as--

* J2EE 1.4 application clients

* EJB 2.1 projects

* 2.4 Web applications

If multiple message destination references are associated with a single message destination link, then a single JNDI name for an enterprise bean that maps to the message destination link, and in turn to all of the linked message destination references, is collected during deployment. At run time, the message destination references are bound to the administered message destinations in the target operational environment.

If a message destination reference and a message-driven bean are linked by the same message destination, both the reference and the bean should have the same destination JNDI name. When both have the same name, only the destination JNDI name for the message-driven bean is collected and applied to the corresponding message destination reference.

If a deployer chooses to generate default bindings when installing the application, the install wizard assigns JNDI names to incomplete message destination references as follows: If a message destination reference has a <message-destination-link>, then the JNDI name is set to ejs/message-destination-linkName. Otherwise, the JNDI name is set to eis/message-destination-refName.

Other bindings that might be needed

Depending on the references in and artifacts used by your application, you might need to define bindings for references and artifacts not listed in this topic.

B) Perform WebSphere Application Server Network Deployment V7.0 backup, restore and configuration tasks.

With WebSphere Application Server V7.0, the job manager and administrative agent profile types have been introduced to enhance the administration capabilities.

Service integration environment backup

You should back up your service integration setup on a regular basis so that it can be restored in the event of an unrecoverable failure.

For service integration, these are the main components that you can back up:

Configuration files

The configuration of a service integration setup is stored as XML files. To back up or restore these configuration files, use the relevant command as detailed in Backing up and restoring administrative configurations. Any backup or restore of a service integration environment must include a backup or restore of the configuration files.

Data stores that are accessed by the messaging engines

Backing up and restoring your data stores is optional. As messages are transient in nature, you might not want to back up or restore the data stores.

* If you do not back up the data stores, and you modified your current configuration since it was last backed up, when you restore the configuration backup be aware that you might lose messages. For example, if you back up the configuration and then create a bus destination, when you restore the configuration backup the bus destination will no longer exist. Any messages for this destination will be deleted when the server that hosted that messaging engine is restarted.

* If you do back up your data stores, you must also back up the configuration files. You must back up or restore the configuration files at the same time as the data store is backed up or restored. Backing up and restoring at the same time maintains the consistency of the system and reduces the possibility of loss of messages, or duplication of messages from the time of the backup.

To back up a data store, see Backing up a data store.

File stores that are accessed by the messaging engines

Backing up and restoring your file stores is optional. As messages are transient in nature, you might not want to back up or restore the files. To back up a file store, see Backing up a file store.

If your environment includes multiple servers, you should back up all the servers at the same time, otherwise messages from the time of the backup might be lost or duplicated. You should also minimize the amount of message traffic reduce the possibility of lost or duplicate messages.

When you restart a messaging engine after restoring a backup, you should take steps to minimize loss of messages. See Restoring a data store and recovering its messaging engine.

Backing up a data store

Backing up a data store enables you to restore the data store from the backup if a failure occurs that cannot be dealt with by the system.

About this task

To back up the tables that comprise a data store, refer to the documentation for your chosen database.

Note: If your relational database management system (RDBMS) is DB2®, and it is being used as the persistent data store, the backup process can use the suspended I/O feature of DB2. With other databases that do not possess this capability, the backup might present a longer interruption to service, or require that the messaging engine is stopped while the backup is made. If you attempt to back up the data store for a messaging engine that is still running, you might lose or corrupt important data.

Procedure

1. Unless your backup process uses the suspended I/O feature of DB2 stop the messaging engine.

2. Back up the data store in accordance with the documentation for your chosen database. Include the tables described in Data store tables.

Data store tables

A data store uses a relational database management system (RDBMS), working through JDBC, to store data as rows in a set of tables. This data is important when you are backing up or restoring a data store.

The following table summarizes the purpose of each data store table.

Table name	Purpose
SIBOWNER	Ensures exclusive access to the data store by an active messaging engine.
SIBOWNERO	Used for locking the data store. This table stores no data in its one EMPTY_COLUMN column.
SIBCLASSMAP	Catalogs the different object types in the data store.
SIBLISTING	Catalogs the SIBnnn tables.
SIBXACTS	Maintains the status of active two-phase commit transactions.
SIBKEYS	Assigns unique identifiers to objects in the messaging engine.
SIBnnn, where nnn is a number	Contains persistent objects such as messages and subscription information. These tables hold both persistent and nonpersistent objects, using separate tables for the different types of data.

Note: The SIBOWNERO table was introduced for WebSphere® Application Server Version 7.0 and must be created when you are migrating to WebSphere Application Server Version 7.0 from an earlier version of WebSphere Application Server. See Migrating a messaging engine based on a data store for things to consider when you are migrating a messaging engine based on a data store.

Restoring a data store and recovering its messaging engine

When a failure occurs that cannot be dealt with by the system, you can restore the data store or data stores from a backup. Use this task to restore a backup of a data store and to recover its associated messaging engine afterward.

About this task

You should also restore the configuration files for the system, to ensure that it functions as it did at the time the backup was taken, for more information about why you should do this see Service integration environment backup. After you have restored the data store, you must restart the associated messaging engine.

When you restart a messaging engine after restoring a backup you must start it in Restart after restore mode, to minimize the effects of the messaging engine not being synchronized with any other messaging engines it was in communication with before the failure. If you restart the messaging engine in Normal mode, some of the new messages produced at this messaging engine might be discarded by the receiving messaging engine, for an indeterminate amount of time after restart. In Restart after restore mode, previously transmitted messages might be resent, potentially creating duplicates of messages that were produced before the backup was taken. However new messages are not lost or duplicated (if this is specified by the quality of service for the message).

You can restart a messaging engine in Restart after restore mode only by using the wsadmin client; you cannot do it from the administrative console. You must only start a messaging engine in this mode when starting the messaging engine for the first time after restoring the backup. After the initial restart, you can undertake further restarts as usual.

Restart after restore mode is ignored if you start the server in Recovery mode. If you require both a Recovery mode start and a Restart after restore mode start:

1. Start the server in recovery mode
2. Wait for the startup to complete and for the server to stop
3. Start the messaging engine in Restart after restore mode

If you see the following message in the JVM System output fileSystemOut.log, it might indicate that you have restored from a backup and restarted the messaging engine without using the Restart after restore mode.

CWSIP0784E: Messaging engine: receivingME received a message from messaging engine: producingME that was not expected.

To resolve this issue, stop the messaging engine and restart it in Restart after restore mode.

Note: This message might also appear in other situations, so you should restart the messaging engine in Restart after restore mode only if you know you have restored a backup.

For information about the JVM System output fileSystemOut.log and how to view it, see Viewing the JVM logs.

You can recover any number of messaging engines at the same time, by following the actions below for each messaging engine in turn.

Procedure

1. Change the initial state of the messaging engine to Stop, so that the messaging engine will not be automatically restarted by a server process:

    1. Use the administrative console to select the messaging engine by clicking Service integration > Buses > bus_name > [Topology] Messaging engines > engine_name.

    2. In the Initial state list, click Stopped.

    3. Click OK.

2. Save your changes to the master configuration, ensuring that you select the Synchronize changes with Nodes check box.

3. Stop the messaging engine if it is running (see Stopping a messaging engine for instructions on how to do this). If the messaging engine does not respond, stop the server process that is hosting the messaging engine.

4. Restore the backup of the data store that is accessed by the messaging engine, by referring to Restoring a data store.

5. Restore the backup of the configuration files by using the backupConfig command (see Backing up and restoring administrative configurations). This backup should have been taken at the same time as the data store backup.

6. Restart any servers that were stopped by the failure.

7. Restart the messaging engine in Restart after restore mode by performing the following steps:

    1. Start the wsadmin client.

    2. Invoke the start command, with the FLUSH parameter, on the MBean for the messaging engine. For example:

```
wsadmin>myME=AdminControl.queryNames("type=SIBMessagingEngine,*").splitlines()[0]
wsadmin>AdminControl.invoke(myME , "state")
'stopped'
wsadmin>AdminControl.invoke(myME , "start" , ["Flush"])
wsadmin>AdminControl.invoke(myME , "state")
'started'
```

A number of messages might be output to the JVM SystemOut.log file to indicate the progress of the restart process.

8. Check the JVM SystemOut.log file for the following message that indicates that the restart was successful, in other words, no failures occurred while attempting to restart the messaging engine.

```
CWSIP0783E: Messaging engine: messagingEngine started, flush of all delivery streams completed.
```

If this message does not appear, a failure has occurred that has prevented the messaging engine from restarting. Resolve the cause of the failure and repeat the Restart after restore procedure until the restart is successful.

Backing up a file store

You can back up the files in a file store by using the facilities of your operating system, or by using a backup tool.

About this task

A file store is a type of message store that directly uses files in a file system through the operating system. Use the facilities of your operating system or a backup tool to back up the log file, the permanent store file, and the temporary store file, which comprise a file store. For more information about these files, see File store configuration attributes.

Important: If you start backing up files while the messaging engine is still running, data might be corrupted.

Note: You must treat the log file, the temporary store file, and the permanent store file as one unit; that is, any operations must be performed on all three files.

Restoring a file store

You can restore the files in a file store by using the facilities of your operating system, or you can restore the backup copies of your files by using a backup tool.

Before you begin

Before you start this task, make sure that the messaging engine is not running.

Important: If you restore a file store while the messaging engine is still running, data might be corrupted.

About this task

When you complete this task, you must treat the log file, the temporary store file, and the permanent store file (which together comprise a file store) as one unit; that is, any operations must be performed on all three files. For more information about these files, see File store configuration attributes.

Procedure

Use the facilities of your operating system, or a backup tool, to restore the backup copy of the log file, the permanent store file, and the temporary store file.

backupConfig command

The backupConfig command is a simple utility to back up the configuration of your node to a file.

By default, all servers on the node stop before the backup is made so that partially synchronized information is not saved. For more information about where to run this command, see the information on using command line tools. If you do not have root authority, you must specify a path for the backup file in a location where you have write permission. The backup file will be in zip format and a .zip extension is recommended.

In a UNIX® or Linux® environment, the backupConfig command does not save file permissions or ownership information. The restoreConfig command uses the current umask and effective user ID (EUID) to set the permissions and ownership when restoring a file. If it is required that the restored files have the original permissions and ownership, use the tar command (available on all UNIX or Linux systems) to back up and restore the configuration.

[iSeries] The backupConfig command does not save authorities that were granted to the configuration directory structure of the profile. The restoreConfig command sets the owner of the directory structure and its contents to QEJBSVR and restores private authorities to the QTMHHTTP and QNOTES user profiles (if they exist). It does not restore any other private authorities that were granted.

Supported configurations: In previous versions of the product, user ID and password information was obtained by this command from the profile_root/properties/soap.client.props file. In this version, the same information is obtained from the profile_root/properties/ipc.client.props file. To avoid user ID and password prompts when you use this command, add the user ID and password information to the ipc.client.props file.

```
backupConfig.sh /path/to/myBackup.zip -nostop
```

restoreConfig command

Use the restoreConfig command to restore the configuration of your node after backing up the configuration using the backupConfig command.

The restoreConfig command is a simple utility to restore the configuration of your node after backing up the configuration using the backupConfig command. By default, all servers on the node stop before the configuration restores so that a node synchronization does not occur during the restoration. If the configuration directory already exists, it is renamed before the restoration occurs. For more information about where to run this command, see the Using command line tools article.

If you directly make changes to the application files in the app_server_root/installedApps directory, a process known as "hot deployment", but do not make the same changes to the application files in the app_server_root/config directory, the changes might be overwritten if you use the restoreConfig command.

[Linux] [iSeries] [z/OS] The backupConfig command does not save file permissions or ownership information. The restoreConfig command uses the current umask and effective user ID (EUID) to set the permissions and ownership when restoring a file. If it is required that the restored files have the original permissions and ownership, use the tar command (available on all UNIX® or Linux® systems) to back up and restore the configuration.

[AIX] [z/OS] If you are using a logical directory for app_server_root/config, the restoreConfig command will not work.

restoreConfig.sh /path/to/myBackup.zip -location /tmp -nostop

Be aware that if you restore the configuration to a directory that is different from the directory that was backed up when you performed the backupConfig command, you might need to manually update some of the paths in the configuration directory.

Configuration Files

- admin-authz.xml config/cells/cell_name/ Define a role for administrative operation authorization.
- app.policy config/cells/cell_name/nodes/node_name/ Define security permissions for application code. (Manual Editing Required).
- cell.xml config/cells/cell_name/ Identify a cell.
- cluster.xml config/cells/cell_name/clusters/cluster_name/ Identify a cluster and its members and weights. This file is only available with the Network Deployment product.
- deployment.xml config/cells/cell_name/applications/application_name/ Configure application deployment settings such as target servers and application-specific server configuration.
- filter.policy config/cells/cell_name/ Specify security permissions to be filtered out of other policy files. (Manual Editing Required).
- integral-jms-authorizations.xml config/cells/cell_name/ Provide security configuration data for the integrated messaging system. (Manual Editing Required).
- library.policy config/cells/cell_name/nodes/node_name/ Define security permissions for shared library code. (Manual Editing Required).
- multibroker.xml config/cells/cell_name/ Configure a data replication message broker.
- namestore.xml config/cells/cell_name/ Provide persistent name binding data. (Manual Editing Required).
- naming-authz.xml config/cells/cell_name/ Define roles for a naming operation authorization. (Manual Editing Required).
- node.xml config/cells/cell_name/nodes/node_name/ Identify a node.
- pmirm.xml config/cells/cell_name/ Configure PMI request metrics. (Manual Editing Required).

- resources.xml cell, node, server scopes Define operating environment resources, including JDBC, JMS, JavaMail, URL, JCA resource providers and factories.
- security.xml config/cells/cell_name/ Configure security, including all user ID and password data.
- server.xml config/cells/cell_name/nodes/node_name/servers/server_name/ Identify a server and its components.
- serverindex.xml config/cells/cell_name/nodes/node_name/ Specify communication ports used on a specific node.
- spi.policy config/cells/cell_name/nodes/node_name/ Define security permissions for service provider libraries such as resource providers. (Manual Editing Required).
- variables.xml cell, node, server scopes Configure variables used to parameterize any part of the configuration settings.
- virtualhosts.xml config/cells/cell_name/ Configure a virtual host and its MIME types.

C) Tune performance of WebSphere Application Server Network Deployment V7.0 (e.g., JDBC data source, configure caching, queuing and thread pooling parameters, tune JVM heap size, etc.).

Connection pooling

Using connection pools helps to both alleviate connection management overhead and decrease development tasks for data access.

Each time an application attempts to access a backend store (such as a database), it requires resources to create, maintain, and release a connection to that datastore. To mitigate the strain this process can place on overall application resources, the Application Server enables administrators to establish a pool of backend connections that applications can share on an application server. Connection pooling spreads the connection overhead across several user requests, thereby conserving application resources for future requests.

The application server supports JDBC 4.0 APIs for connection pooling and connection reuse. The connection pool is used to direct JDBC calls within the application, as well as for enterprise beans using the database.

Benefits of connection pooling

Connection pooling can improve the response time of any application that requires connections, especially Web-based applications. When a user makes a request over the Web to a resource, the resource accesses a data source. Because users connect and disconnect frequently with applications on the Internet, the application requests for data access can surge to considerable volume. Consequently, the total datastore overhead quickly becomes high for Web-based applications, and performance deteriorates. When connection pooling capabilities are used, however, Web applications can realize performance improvements of up to 20 times the normal results.

With connection pooling, most user requests do not incur the overhead of creating a new connection because the data source can locate and use an existing connection from the pool of connections. When the request is satisfied and the response is returned to the user, the resource returns the connection to the connection pool for reuse. The overhead of a disconnection is avoided. Each user request incurs a fraction of the cost for connecting or disconnecting. After the initial resources are used to produce the connections in the pool, additional overhead is insignificant because the existing connections are reused.

When to use connection pooling

Use connection pooling in an application that meets any of the following criteria:

* It cannot tolerate the overhead of obtaining and releasing connections whenever a connection is used.

* It requires Java™ Transaction API (JTA) transactions within the Application Server.

* It needs to share connections among multiple users within the same transaction.

* It needs to take advantage of product features for managing local transactions within the application server.

* It does not manage the pooling of its own connections.

* It does not manage the specifics of creating a connection, such as the database name, user name, or password

Avoid trouble: Connection pooling is not supported in an application client. The application client calls the database directly and does not go through a data source. If you want to use the getConnection() request from the application client, configure the JDBC provider in the application client deployment descriptors, using Rational® Application Developer or an assembly tool. The connection is established between application client and the database. Application clients do not have a connection pool, but you can configure JDBC provider settings in the client deployment descriptors.gotcha

How connections are pooled together

When you configure a unique data source or connection factory, you must give it a unique Java Naming and Directory Interface (JNDI) name. This JNDI name, along with its configuration information, is used to create the connection pool. A separate connection pool exists for each configured data source or connection factory.

Furthermore, the application server creates a separate instance of the connection pool in each application server that uses the data source or connection factory. For example:

* If you run a three server cluster in which all of the servers use myDataSource, and myDataSource has a Maximum Connections setting of 10, then you can generate up to 30 connections (three servers times 10 connections).

Consider how this behavior potentially impacts the number of connections that your backend resource can support. See the topic, Connection pool settings, for more information.

Other considerations for determining the maximum connections setting:

* Each entity bean transaction requires an additional database connection, dedicated to handling the transaction.

* If clones are used, one data pool exists for each clone.

* [AIX] [HP-UX] [Solaris] On supported UNIX® systems, a separate DB2® process is created for each connection; these processes quickly affect performance on systems with low memory and cause errors.

It is also important to note that when using connection sharing, it is only possible to share connections obtained from the same connection pool.

Thread pool settings

Use this page to configure a thread pool that an application server uses. A thread pool enables components of the server to reuse threads, which eliminates the need to create new threads at run time. Creating new threads expends time and resources.

[z/OS] Note: This page does not display for z/OS® because thread pools are not used in a z/OS environment.

To view this administrative console page, you can choose more than one navigational route. For example, click Servers > Server Types > WebSphere application servers > server_name > Thread pool, and then select the thread pool you need to configure.

To configure the thread pool for the ORB Service, click Servers > Server Types > WebSphere application servers > server_name > Container services > ORB service. Then, under Thread Pool Settings, either:

* Select Use the ORB.thread.pool settings associated with the Thread Pool Manager (recommended), and then click ORB thread pool settings, or

* Select Use the thread pool settings directly associated with the ORB service, and then click Thread pool settings.

Note: Because these console panels display information dynamically, you might not see all of the fields listed on any particular panel.

Name

The name of the thread pool to create. The name must be unique within the server.

This field does not appear if you click thread pool settings.

Data type	String

Description

A text description of the thread pool.

This field does not appear if you click thread pool settings.

Data type	String

Minimum size

Specifies the minimum number of threads to allow in the pool. When an application server starts, no threads are initially assigned to the thread pool. Threads are added to the thread pool as the workload assigned to the application server requires them, until the number of threads in the pool equals the number specified in the Minimum size field. After this point in time, additional threads are added and removed as the workload changes. However the number of threads in the pool never decreases below the number specified in the Minimum size field, even if some of the threads are idle.

This field does not appear if you click thread pool settings.

Data type	Integer
Default	50

Maximum size

Specifies the maximum number of threads to maintain in the default thread pool.

If your Tivoli® Performance Viewer shows the Percent Maxed metric to remain consistently in the double digits, consider increasing the Maximum size. The Percent Maxed metric indicates the amount of time that the configured threads are used.

Data type	Integer
Default	50
Recommended	50 (25 on Linux® systems)

Thread inactivity timeout

Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.

Note: The administrative console does not allow you to set the inactivity timeout to a negative number. To do this you must modify the value directly in the server.xml file.

Data type	Integer
Units	Milliseconds

Default	3500

Allow thread allocation beyond maximum thread size

Specifies whether the number of threads can increase beyond the maximum size configured for the thread pool.

The maximum number of threads that can be created is constrained only within the limits of the Java™ virtual machine and the operating system. When a thread pool, that is allowed to grow, expands beyond the maximum size, the additional threads are not reused and are discarded from the pool after processing of the work items for which they were created is completed.

Data type	Boolean
Default	Not enabled (false)

Configure the heap size.

The Java heap parameters influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes garbage collection to take longer.

The JVM uses defined thresholds to manage the storage that it is allocated. When the thresholds are reached, the garbage collector is invoked to free up unused storage. Therefore, garbage collection can cause significant degradation of Java performance. Before changing the initial and maximum heap sizes, you should consider the following information:

* In the majority of cases you should set the maximum JVM heap size to a value that is higher than the initial JVM heap size. This setting allows for the JVM to operate efficiently during normal, steady state periods within the confines of the initial heap. This setting also allows the JVM to operate effectively during periods of high transaction volume because the JVM can expand the heap up to the value specified for the maximum JVM heap size. In some rare cases, where absolute optimal performance is required, you might want to specify the same value for both the initial and maximum heap size. This setting eliminates some overhead that occurs when the JVM expands or contracts the size of the JVM heap. Before changing any of the JVM heap sizes, verify that the JVM storage allocation is large enough to accommodate the new heap size.

* Do not make the size of the initial heap so large that while it initially improves performance by delaying garbage collection, when garbage collection does occur, the collection process affects response time because the process has to run longer.

To use the administrative console to configure the heap size:

1. In the administrative console, click Servers > Server Types > WebSphere application servers > server_name.

2. In the Server Infrastructure section, click Java and process management > Process definition > Java virtual machine.

3. Specify a new value in either the Initial heap size or the Maximum heap size field.

You can also specify values for both fields if you need to adjust both settings.

For performance analysis, the initial and maximum heap sizes should be equal.

The Initial heap size setting specifies, in megabytes, the amount of storage that is allocated for the JVM heap when the JVM starts. The Maximum heap size setting specifies, in megabytes, the maximum amount of storage that can be allocated to the JVM heap. Both of these settings have a significant effect on performance.

If you are tuning a production system where you do not know the working set size of the enterprise applications that are running on that system, an appropriate starting value for the initial heap size is 25 percent of the maximum heap size. The JVM then tries to adapt the size of the heap to the working set size of the application.

The following illustration represents three CPU profiles, each running a fixed workload with varying Java heap settings. In the middle profile, the initial and maximum heap sizes are set to 128 MB. Four garbage collections occur. The total time in garbage collection is about 15 percent of the total run. When the heap parameters are doubled to 256 MB, as in the top profile, the length of the work time increases between garbage collections. Only three garbage collections occur, but the length of each garbage collection is also increased. In the third profile, the heap size is reduced to 64 MB and exhibits the opposite effect. With a smaller heap size, both the time between garbage collections and the time for each garbage collection are shorter. For all three configurations, the total time in garbage collection is approximately 15 percent. This example illustrates an important concept about the Java heap and its relationship to object utilization. A cost for garbage collection always exists when running enterprise applications.

Run a series of tests that vary the Java heap settings. For example, run experiments with 128 MB, 192 MB, 256 MB, and 320 MB. During each experiment, monitor the total memory usage. If you expand the heap too aggressively, paging can occur.

Use the vmstat command or the Windows® Performance Monitor to check for paging. If paging occurs, reduce the size of the heap or add more memory to the system.

When all the runs are finished, compare the following statistics:

* Number of garbage collection calls

* Average duration of a single garbage collection call

* Ratio between the length of a single garbage collection call and the average time between calls

If the application is not over utilizing objects and has no memory leaks, the state of steady memory utilization is reached. Garbage collection also occurs less frequently and for short duration.

If the heap free space settles at 85 percent or more, consider decreasing the maximum heap size values because the application server and the application are under-utilizing the memory allocated for heap.

Test for memory leaks

Memory leaks in the Java language are a dangerous contributor to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently until the heap is exhausted and the Java code fails with a fatal out-of-memory exception. Memory leaks occur when an unused object has references that are never freed. Memory leaks most commonly occur in collection classes, such as Hashtable because the table always has a reference to the object, even after real references are deleted.

High workload often causes applications to crash immediately after deployment in the production environment. These application crashes if the applications are having memory leaks because the high workload accelerates the magnification of the leakage, and a memory allocation failures occur.

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list provides insight on how to interpret the results of your memory leak testing:

* Long-running test

Memory leak problems can manifest only after a period of time, therefore, memory leaks are found easily during long-running tests. Short running tests might provide invalid indications of where the memory leaks are occurring. It is sometimes difficult to know when a memory leak is occurring in the Java language, especially when memory usage has seemingly increased either abruptly or monotonically in a given period of time. The reason it is hard to detect a memory leak is that these kinds of increases can be valid or might be the intention of the developer. You can learn how to differentiate the delayed use of objects from completely unused objects by running applications for a longer period of time. Long-running application testing gives you higher confidence for whether the delayed use of objects is actually occurring.

* Repetitive test

In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the number of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is easier. First, the memory usage before running the module is recorded. Then, a fixed set of test cases are run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection must be suggested when recording the actual memory usage by inserting System.gc() in the module where you want garbage collection to occur, or using a profiling tool, to force the event to occur.

* Concurrency test

Some memory leak problems can occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to memory leaks because of the added complication in the program logic. Careless programming can lead to kept or not-released references. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

Consider the following points when choosing which test cases to use for memory leak testing:

o A good test case exercises areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario can suggest creation of data spaces, such as adding a new record, creating an HTTP session, performing a transaction and searching a record.

o Look at areas where collections of objects are used. Typically, memory leaks are composed of objects within the same class. Also, collection classes such as Vector and Hashtable are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the get method of a Hashtable object does not remove its reference to the retrieved object.

You can use the Tivoli Performance Viewer to help find memory leaks.

For optimal results, repeat experiments with increasing duration, such as 1,000, 2,000, and 4,000 page requests. The Tivoli Performance Viewer graph of used memory should have a jagged shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following conditions is appears in the graph:

* The amount of memory used immediately after each garbage collection increases significantly. When this condition occurs, the jagged pattern looks more like a staircase.

* The jagged pattern has an irregular shape.

* The gap between the number of objects allocated and the number of objects freed increases over time.

Heap consumption that indicates a possible leak during periods when the application server is consistently near 100 percent CPU utilization, but disappears when the workload becomes lighter or near-idle, is an indication of heap fragmentation. Heap fragmentation can occur when the JVM can free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small free memory areas in the heap to larger contiguous spaces.

Another form of heap fragmentation occurs when objects that are less than 512 bytes are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation until a heap compaction occurs.

Heap fragmentation can be reduced by forcing compactions to occur. However, there is a performance penalty for forcing compactions. Use the Java -X command to see the list of memory options.

Tune garbage collection

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection typically consumes from 5 to 20 percent of total run time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application.

Monitoring garbage collection while a fixed workload is running, provides you with insight as to whether the application is over using objects. Garbage collection can even detect the presence of memory leaks.

You can use JVM settings to configure the type and behavior of garbage collection. When the JVM cannot allocate an object from the current heap because of lack of contiguous space, the garbage collector is invoked to reclaim memory from Java objects that are no longer being used. Each JVM vendor provides unique garbage collector policies and tuning parameters.

You can use the Verbose garbage collection setting in the administrative console to enable garbage collection monitoring. The output from this setting includes class garbage collection statistics. The format of the generated report is not standardized between different JVMs or release levels.

To adjust your JVM garbage collection settings:

1. In the administrative console, click Servers > Server Types > WebSphere application servers > server_name.

2. In the Server Infrastructure section, click Java and process management > Process definition > Java virtual machine

3. Enter the –X option you want to change in the Generic JVM arguments field.

4. Click Apply.

5. Click Save to save your changes to the master configuration.

6. Stop and restart the application server.

Tuning application servers

The product contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application.

About this task

This group of interrelated components is known as the queuing network. The queuing network helps the system achieve maximum throughput while maintaining the overall stability of the system.

The following steps describe various tuning tasks that may improve your application server performance. You can choose to implement any of these application server settings. These steps can be performed in any order.

Procedure

1. Tune the object request broker. An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It supports client requests and responses received from servers in a network-distributed environment. You can use the following parameters to tune the ORB:

* Set Pass by reference (com.ibm.CORBA.iiop.noLocalCopies) as described in the information on Object Request Broker service settings.

* Set the Connection cache minimum (com.ibm.CORBA.MaxOpenConnections) as described in the information about Object Request Broker service settings.

* Set Maximum size as described in the topic about thread pool settings.

* Set com.ibm.CORBA.ServerSocketQueueDepth as described in the information about Object Request Broker custom properties.

* Set the com.ibm.CORBA.FragmentSize as described in the information about Object Request Broker custom properties.

See the information about Object Request Broker tuning guidelines for tips on using these parameters to tune the ORB.

2. Tune the XML parser definitions.

* Description: Facilitates server startup by adding XML parser definitions to the jaxp.properties and xerxes.properties files in the ${app_server_root}/jre/lib directory. The XMLParserConfiguration value might change as new versions of Xerces are provided.

* How to view or set: Insert the following lines in both files:

```
 javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl

 javax.xml.parsers.DocumentBuildFactory=org.apache.xerces.jaxp.

        DocumentBuilderFactoryImpl

 org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.

        StandardParserConfiguration
```

* Default value: None

* Recommended value: None

3. Tune the dynamic cache service.

Using the dynamic cache service can improve performance. See the information on using the dynamic cache server to improve performance for information about using the dynamic cache service and how it can affect your application server performance.

4. Tune the Web container. The product Web container manages all HTTP requests to servlets, JavaServer Pages and Web services. Requests flow through a transport chain to the Web container. The transport chain defines the important tuning parameters for performance for the Web container. There is a transport chain for each TCP port that the product is listening on for HTTP requests. For example, the default HTTP port 9080 is defined in Web container inbound channel chain. Use the following parameters to tune the Web container:

* HTTP requests are processed by a pool of server threads. The minimum and maximum thread pool size for the Web container can be configured for optimal performance. Generally, 5 to 10 threads per server CPU provides the best throughput. The number of threads configured does not represent the number of requests that the product can process concurrently. Requests are queued in the transport chain when all threads are busy. To specify the thread pool settings:

1. Click Servers > Server Types > WebSphere application servers >server_name Web container settings > Web container > Web container transport chains.

2. Select the normal inbound chain for serving requests. This chain is typically called WCInboundDefault, and listens on port 9080.

3. Click TCP Inbound Channel (TCP_2).

4. Set Thread Pools under Related Items.

5. Select WebContainer.

6. Enter values for Minimum Size and Maximum Size.

* The HTTP 1.1 protocol provides a keep-alive feature to enable the TCP connection between HTTP clients and the server to remain open between requests. By default the product closes a given client connection after a number of requests or a timeout period. After a connection is closed, it is recreated if the client issues another request. Early closure of connections can reduce performance. Enter a value for the maximum number of persistent requests to (keep-alive) to specify the number of requests that are allowed on a single HTTP connection. Enter a value for persistent timeouts to specify the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests. To specify values for Maximum persistent requests and Persistent timeout:

1. Click Servers > Server Types > WebSphere application servers >server_name. Then in the Container Settings section, click Web container > Web container transport chains.

2. Select the normal inbound chain for serving requests. This chain is typically called WCInboundDefault, and listens on port 9080.

3. Click HTTP Inbound Channel (HTTP_2).

4. Enter values for Maximum persistent requests and Persistent timeout.

5. Tune the EJB container. An Enterprise JavaBeans™ (EJB) container is automatically created when you create an application server. After the EJB container is deployed, you can use the following parameters to make adjustments that improve performance.

* Set the Cleanup interval and the Cache size. See the topic on EJB cache settings for more information.

* Break CMP enterprise beans into several enterprise bean modules. See the topic on assembling EJB modules for more information.

See the topic on EJB method invocation queuing for further information.

6. Tune the session management.

The installed default settings for session management are optimal for performance.

7. Tune the data sources and associated connection pools. A data source is used to access data from the database; it is associated with a pool of connections to that database.

* Review the topic on connection pooling to understand how the number of physical connections within a connection pool can change performance.

* Use the topic on data access tuning parameters as a reference for the data source and connection pool properties that most affect performance.

8. Tune the URL invocation cache.

Each JavaServer Page is a unique URL. If you have more than 50 unique URLs that are actively being used, increase the value specified for the invocationCacheSize JVM custom property. This property controls the size of the URL invocation cache.

D) Use Tivoli Performance Viewer (TPV) / Request Metrics to gather information about resources and analyze results.

Performance Monitoring Infrastructure (PMI)

Use this page to learn about Performance Monitoring Infrastructure and other tools to help monitor the overall health of the application server.

A typical Web system consists of a Web server, application server, and a database. Monitoring and tuning the application server is critical to the overall performance of the Web system. Performance Monitoring Infrastructure (PMI) is the core monitoring infrastructure for WebSphere® Application Server and WebSphere family products like Portal, Commerce, and so on. The performance data provided by WebSphere PMI helps to monitor and tune the application server performance.

When tuning the WebSphere Application Server for optimal performance, or fixing a poorly performing Java™ Platform, Enterprise Edition (Java EE) application, it is important to understand how the various run time and application resources are behaving from a performance perspective. PMI provides a comprehensive set of data that explains the runtime and application resource behavior. For example, PMI provides database connection pool size, servlet response time, Enterprise JavaBeans™ (EJB) method response time, Java virtual machine (JVM) garbage collection time, CPU usage, and so on. This data can be used to understand the runtime resource utilization patterns of the thread pool, connection pool, and so on, and the performance characteristics of the application components like servlets, JavaServer Pages (JSP), and enterprise beans.

Using PMI data, the performance bottlenecks in the application server can be identified and fixed. For instance, one of the PMI statistics in the Java DataBase Connectivity (JDBC) connection pool is the number of statements discarded from prepared statement cache. This statistic can be used to adjust the prepared statement cache size in order to minimize the discards and to improve the database query performance. PMI data can be monitored and analyzed by Tivoli® Performance Viewer (TPV), other Tivoli tools, your own applications, or third party tools. TPV is a graphical viewer for PMI data that ships with WebSphere Application Server. Performance advisors use PMI data to analyze the run-time state of the application server, and provide tuning advice to optimize the application server resource utilization.

PMI data can also be used to monitor the health of the application server. Some of the health indicators are CPU usage, Servlet response time, and JDBC query time. Performance management tools like Tivoli Monitoring for Web Infrastructure and other third party tools can monitor the PMI data and generate alerts based on some predefined thresholds.

Why use Tivoli Performance Viewer?

Administrators and programmers can monitor the overall health of WebSphere® Application Server from within the administrative console by using Tivoli® Performance Viewer (TPV).

From TPV, you can view current activity or log Performance Monitoring Infrastructure (PMI) performance data for the following:

* System resources such as CPU utilization
* WebSphere pools and queues such as a database connection pool
* Customer application data such as average servlet response time

You can also view data for other products or customer applications that implement custom PMI by using TPV. For more information on custom PMI, refer to Enabling PMI data collection.

By viewing PMI data, administrators can determine which part of the application and configuration settings to alter in order to improve performance. For example, in order to determine what part of the application to focus on, you can view the servlet summary report, enterprise beans and Enterprise JavaBeans™ (EJB) methods, and determine which of these resources has the highest response time. You can then focus on improving the configuration for those application resources with the longest response times.

Tivoli Performance Viewer is used to help manage configuration settings by viewing the various graphs or using the Tivoli Performance Advisor. For example, by looking at the summary chart for thread pools, you can determine whether the thread pool size needs to be increased or decreased by monitoring the percent (%) usage. After configuration settings are changed based on the data provided, you can determine the effectiveness of the changes. To help with configuration settings, use the Tivoli Performance Advisor. The Advisor assesses various data while your application is running, and provides configuration setting advice to improve performance.

For transitioning users: In Version 4.0, Tivoli Performance Viewer was originally named the Resource Analyzer. In Version 5.0, TPV was a standalone Java™ application. In Versions 6.0.x and later, TPV is embedded in the administrative console.trns

Monitoring performance with Tivoli Performance Viewer (TPV)

Tivoli® Performance Viewer (TPV) enables administrators and programmers to monitor the overall health of WebSphere® Application Server from within the administrative console.

Before you begin

In Version 4.0, Tivoli Performance Viewer was named the Resource Analyzer. In Version 5.0, TPV is a standalone Java™ application. In Versions 6.0.x and later, TPV is embedded in the administrative console. From TPV, you can view current activity and summary reports, or log Performance Monitoring Infrastructure (PMI) performance data. TPV provides a simple viewer for the performance data collected by the Performance Monitoring Infrastructure.

About this task

By viewing TPV data, administrators can determine which part of the application and configuration settings to change in order to improve performance. For example, you can view the servlet summary reports, enterprise beans, and Enterprise JavaBeans™ (EJB) methods in order to determine what part of the application to focus on. Then, you can sort these tables to determine which of these resources has the highest response time. Focus on improving the configuration for those application resources taking the longest response time.

If a problem happens when using Tivoli Performance Viewer to view Performance Monitoring Infrastructure (PMI) data, refer to the troubleshooting Technote.

Procedure

1. Optional: Adjust the Performance Monitoring Infrastructure (PMI) settings for the servers that you want to monitor. The PMI service is enabled by default with a basic set of counters enabled.

2. Monitor current server activity. You can view real-time data on the current performance activity of a server using TPV in the administrative console.

* Use the performance advisors to examine various data while your application is running. The performance advisor in TPV provides advice to help tune systems for optimal performance by using collected PMI data.

* Configure user and logging settings for TPV. These settings may affect the performance of your application server.

* View summary reports on servlets, Enterprise JavaBeans (EJB) methods, connections pools and thread pools in WebSphere Application Server.

* View performance modules that provide graphs and of various performance data on system resources such as CPU utilization, on WebSphere pools and queues such as database connection pools, and on customer application data such as servlet response time. In addition to providing a viewer for performance data, TPV enables you to view data for other products or customer applications that have implemented custom PMI.

* WebSphere Application Server Version 7 for IBM® i requires Portable Application Solutions Environment (PASE), 5722SS1 (IBM i V5R4) or 5761SS1 (IBM i V6Rx) option 33, for graphics

3. View server performance logs. You can record and view data that has been logged by TPV in the administrative console. Be sure to configure user and logging settings for TPV.

4. Log performance data. You can record real-time data for later retrieval and analysis.

Enabling PMI data collection

Enable PMI data collection to diagnose problems and tune application performance.

Before you begin

Performance Monitoring Infrastructure (PMI) must be enabled (by default PMI is enabled) before collecting any performance data. PMI must be enabled before the server starts. If PMI is enabled after the server is started, the server needs to be restarted to start the PMI.

About this task

When PMI service is enabled, the monitoring of individual components can be enabled or disabled dynamically. PMI provides four predefined statistic sets that can be used to enable a set of statistics. The following table provides details about the statistic sets. If the predefined statistic sets does not meet your monitoring requirement, the Custom option can be used to selectively enable or disable individual statistics.

Enable PMI using the administrative console, wsadmin tool, or Java™ Virtual Machine Tool Interface (JVMTI).

Statistic set	Description
None	All statistics are disabled.
Basic	Statistics specified in J2EE 1.4, as well as top statistics like CPU usage and live HTTP sessions are enabled. This set is enabled out-of-the-box and provides basic performance data about runtime and application components.
Extended	Basic set plus key statistics from various WebSphere® Application Server components like WLM, and dynamic caching are enabled. This set provides detailed performance data about various runtime and application components.
All	All statistics are enabled.
Custom	Enable or disable statistics selectively.

Custom setting

WebSphere Application Server Version 6.0 introduces fine-grained control to enable/disable statistics individually. The fine-grained control is available under the custom statistic set.

Though the Version 5.x monitoring levels {N, L, M, H, X} are deprecated in Version 6.0 and above, the 5.x PMI APIs are supported for backward compatibility. It is possible to use a Version 6.0 or later API to set the monitoring level and a Version 5.0 API to get the monitoring level. A new level ‘F' – “fine-grained” is introduced to indicate to the Version 5.x API that the fine-grained or Version 6.x monitoring specification is in effect. This new level ‘F' might bereturned if a V5.x API is used to get the monitoring level from a server using Version 6.x monitoring specification.

In WebSphere Application Server Version 4.0 and Version 5.0, the statistics were enabled based on a monitoring/instrumentation level. The levels are None, Low, Medium, High, and Max {N, L, M, H, X}. Enabling at a given level might include all the statistics at the given level plus counters from levels below the given level. So if you enable the Web Application module at level Medium {M} it enables all the counters at level M, plus all the Low {L} level counters. See PMI data collection for more information.

Sequential Update

In order to minimize the monitoring overhead, the updates to CountStatistic, AverageStatistic, and TimeStatistic are not synchronized. Since these statistic types track total and average, the extra accuracy is not worth the performance cost. The RangeStatistic and BoundedRangeStatistic are sensitive; therefore, they are always are synchronized. If needed, updates to all the statistic types can be synchronized by checking the Use sequential update check box.

Procedure

Enable PMI using one of the following methods:

* Enabling PMI using the administrative console

* Enabling PMI using wsadmin tool

* Enabling the Java virtual machine profiler data

PMI data collection

PMI data collection can occur in three different interfaces, Java™ Management Extension (JMX) interface (Java EE MBeans and WebSphere® Application Server Perf MBean), Performance Servlet, or PMI client API (deprecated).

JMX Interface

JMX interface is part of the J2EE specification and the recommended way to gather WebSphere Application Server performance data. PMI data can be gathered from the J2EE managed object MBeans or the WebSphere Application Server PMI Perf MBean. While the J2EE MBeans provide performance data about the specific component, the Perf MBean acts as a gateway to the WebSphere Application Server PMI service, and provides access to the performance data for all the components.

Performance Servlet

Performance Servlet provides a way to use an HTTP request to query the PMI data for the entire WebSphere Application Server administrative domain. Since the servlet provides the performance data through HTTP, issues such as firewalls are trivial to resolve. The performance servlet outputs the PMI data as an XML document.

PMI client API (deprecated)

PMI client API provides a wrapper class to deliver PMI data to a client. The PMI client API uses the JMX infrastructure and the Perf MBean to retrieve the PMI data. PMI client provides the data using a WebSphere Application Server-specific data structure.

PMI client API provides a wrapper class to deliver PMI data to a client. The PMI client API uses the JMX infrastructure and the Perf MBean to retrieve the PMI data. PMI client provides the data using a WebSphere Application Server-specific data structure.

PMI architecture

The Performance Monitoring Infrastructure (PMI) uses a client-server architecture.

The server collects performance data from various WebSphere® Application Server components. A client retrieves performance data from one or more servers and processes the data. WebSphere Application Server, Version 6 supports the Java™ Platform, Enterprise Edition (Java EE) Management Reference Implementation (JSR-77).

In WebSphere Application Server, Version 6 and later, PMI counters are enabled, based on a monitoring or instrumentation level. The levels are None, Basic, Extended, All, and Custom. These levels are specified in the PMI module XML file. Enabling the module at a given level includes all the counters at the given level plus counters from levels below the given level. So, enabling the module at the extended level enables all the counters at that level plus all the basic level counters as well.

JSR-077 defines a set of statistics for Java EE components as part of the StatisticProvider interface. The PMI monitoring level of Basic includes all of the JSR-077 specified statistics. PMI is set to monitor at a Basic level by default.

E) Develop and implement a maintenance application strategy using Update Installer and/or Centralized Installation Manager (CIM) (e.g., FixPacks, cumulative fixes and interim fixes.)

The Version 7.0 Update Installer supports multiple releases. The Version 7.0 Update Installer is also compatible with earlier releases; it works with Version 6.0.2 Fix Pack 21 and newer maintenance and any maintenance for Version 6.1.0.x and Version 7.0 releases. This allows a single instance of the Update Installer to apply maintenance to more than one version of the application server. For Version 6.0.2 Fix Pack 19 and previous releases, apply maintenance with the Version 6.0.2.x Update Installer.

Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

The Update Installer wizard is an InstallShield for Multiplatforms wizard that runs with either a graphical user interface or in silent mode with a response file.

Use the proper authorizations to successfully uninstall product updates. Use the update installer program as the root user on a Linux® or UNIX® platform, or as the administrator on a Windows® platform.

Do not launch multiple copies of the Update Installer wizard at one time: Concurrent launches of the update installer program are not supported. Performing more than one update at the same time can produce unpredictable results, which might include a failed or faulty installation.

The Update Installer should not check OS prerequisites when uninstalling Custom Installation Package created by the Install Factory. When you launch the Update Installer to uninstall a Custom Installation Package, additional command line options need to be passed into Update Installer to disable OS prerequisite checking. These command line options are:

```
-W maintenancewarningprereqcheckactionInstallWizardBean.active=false
-W prereqswarningpanelInstallWizardBean.active=false
-W maintenanceprereqcheckactionInstallWizardBean.active=false
-W prereqsfailedpanelInstallWizardBean.active=false
```

Installing fix packs silently

If you want to install maintenance without the graphical user interface, you can use the Update Installer for WebSphere® Software to install a fix pack in silent mode. Ensure that the most recent version of the Update Installer is installed on a target system locally.

Use the Update Installer program from the same user ID that installed the product that you are updating. Otherwise, the file ownership mismatches might require correction by the root user.

Procedure

1. Download the required fix pack from the official IBM® support Web site into the updi_root/maintenance directory.

2. Ensure that all running processes have been stopped.

3. Edit a response file. The one located at the bottom of this page can be used as an example. There are also sample response files found in the updi_root/responsefiles directory.

    1. Specify the location of the product to the response file.

    2. Specify the choice of install maintenance in the response file. For example: -W update.type="install"

    3. Add the maintenance location where packages can be found to the response file.

    4. Run the Update Installer.

For example:

```
./update.sh -silent -options "responsefiles/file_name"
```

5. Review the log file to verify that maintenance is installed successfully. You can find the log at app_server_root/logs/update/maintenance_package.install. If the maintenance package is not applicable to the installation, a log file found in updi_root/logs/tempX lists the reason for the failure. The most recent log file, tmpX, where X refers to the first available empty directory, is created to reflect the status for this attempted install. You might not receive an error message for a failed installation in some cases. If you silently install a maintenance package and you do not receive a response after a short period of time, view the logs. If logs are not generated, then an invalid or missing argument might be causing the installation to fail. Verify the Update Installer syntax for the command line with the response files install.txt and uninstall.txt, located under <Installed_UPDI_root>/responsefiles

Results

One of the following results appears in the log:

INSTCONFSUCCESS

The operation was a success.

INSTCONFPARTIALSUCCESS

The operation was partially successful, refer to the log for more details.

INSTCONFFAILED

The operation failed, refer to the log for more details. In some cases, you might not receive an error message for a failed installation. If you silently install a maintenance package and you do not receive a response after a short period of time, view the logs. If logs are not generated, then an invalid or missing argument might be causing the installation to fail. Verify the Update Installer syntax for the command line with the response files install.txt and uninstall.txt, located under <Installed_UPDI_root>/responsefiles

Installing multiple maintenance packs silently

The Update Installer is capable of automatically selecting the recommended maintenance for a product stack and determining the appropriate installation sort order of the maintenance packages. For example, if multiple feature packs are both installed and there is a recommended fix pack and interim fix for each product, the Update Installer can determine if this maintenance is applicable and install them in the correct order. The following steps lead you through the process of installing multiple maintenance packages using the silent mode.

Add the maintenance location where packages can be found to the response file.

There are two options for installing the fix pack:

1. In the response file, point to the directory containing the fix packages and allow the Update Installer to determine which maintenance packages to install.

2. In the response file, provide a list of all the maintenance packages you want installed, with their complete file path.

```
updi_root = /usr/IBM/WebSphere/UpdateInstaller
```

Deleting profiles created by a service level that is now rolled back: Profiles should be at a service level that is less than or equal to the service level of the WebSphere Application Server product. For example, if you install a fix pack, create a profile, and then uninstall the fix pack, then you must also delete the profile.

Uninstalling fix packs, interim fixes, interim features, test fixes, or fix packs delivered as a CIP slip install

When uninstalling a fix pack that is delivered via a CIP slip install, any interim fixes for the same product must be uninstalled first. They will need to be uninstalled individually, using the Update Installer. (This is the same rule that applies to fix packs installed normally using Update Installer.)

Uninstalling a fix pack delivered by a CIP slip install is similar to uninstalling a regular fix pack. However, the package installed using the CIP uses a different fix ID than a regular fix pack. Once you know the fix ID, you can uninstall the CIP slip install package.

About this task

This task is divided into two major parts:

1. Identifying the CIP slip install fix ID

Iif you are uninstalling a fixpack delivered as a CIP slip install, you need to formulate the CIP slip install fix ID prior to using part two of this task to uninstall the maintenance. This CIP slip install fix ID is critical to a successful uninstall of the CIP slip install fix.

2. Using the graphical user interface to uninstall maintenance

Use this part if you are uninstalling any maintenance, such as fix packs, interim features, test fixes. If you uninstalling a fixpack delivered as a CIP slip install, use this part after you have formulated the CIP slip install fix ID from Part (1),.

Part (1) Identifying the CIP slip install fix ID

1. Navigate to the WAS_HOME/cip directory.

2. Look at the subdirectories listed in the WAS_HOME/cip directory. For each CIP that is installed, a subdirectory is created using the CIP's ID.

3. Identify the CIP that you want to uninstall.

Note that you cannot uninstall the CIP if it was used to initially install the product, but you can uninstall the CIP if it was used to upgrade the product.

4. Take the ID of the CIP you want to uninstall and add the character string "was.primary.pak" to the end of that ID. That combined name is the full fixpack identifier to be uninstalled.

For example, if you wanted to uninstall the CIP that upgraded WAS that was named "com.ibm.was_6.1.0.23", then the full fixpack identifier of that fix pack is:

```
com.ibm.was_6.1.0.23was.primary.pak
```

5. Uninstall the CIP using the Update Installer, described in Part (2) of this task. Follow the same process that you normally would use to uninstall a fix pack. When you supply the name of the maintenance package to be uninstalled, use the full fix pack identifer you formulated in Part (1).

Results

One of the following results appears in the log:

INSTCONFSUCCESS

The operation was a success.

INSTCONFPARTIALSUCCESS

The operation was partially successful, refer to the log for more details.

INSTCONFFAILED

The operation failed, refer to the log for more details.

<u>Logic the Update Installer uses for uninstalling</u>

The Update Installer uses logical procedures when it uninstalls maintenance. The uninstall command works similar to the stack logic in computer science.

Uninstall fix packs in the reverse order that they were installed. The last fix pack installed should be the first one uninstalled.

For example, if maintenance packages A,B,C,D have been installed in that sequence, then the packages should be uninstalled in reverse order: D, C, B, A. This is true if these applications are made up of fix packs, interim fixes or feature packs.

Section 7 - Problem Determination

A) Configure, review and analyze logs (e.g., Web server, IBM WebSphere Application Server Network Deployment V7.0, first failure data capture (FFDC)).

Types of problem symptoms

When a user of an application that is running on WebSphere Application Server first notices a problem, a problem symptom is observed. Sometimes the problem symptom provides clues about the cause of the problem. Other times, a significant amount of problem determination is needed to determine the problem’s root cause.

Here are the common types of symptoms that you might see. Almost every symptom falls into one of these categories:

You cannot install or migrate WebSphere Application Server or install an application into WebSphere Application Server.

You experience difficulties in WebSphere Application Server system management or configuration.

An application or WebSphere Application Server process (for example, an application server, node agent, or deployment manager) is unable to start.

An application does not respond to incoming requests.

An application produces unexpected results (possibly errors or exceptions).

An application cannot connect to an external system or resource.

An application performs slowly or its performance degrades over time.

Level	Content / Significance

Off	No events are logged.
Fatal	Task cannot continue and component cannot function.
Severe	Task cannot continue, but component can still function
Warning	Potential error or impending error
Audit	Significant event affecting server state or resources
Info	General information outlining overall task progress
Config	Configuration change or status
Detail	General information detailing subtask progress
Fine	Trace information - General trace + method entry / exit / return values
Finer	Trace information - Detailed trace
Finest	Trace information - A more detailed trace - Includes all the detail that is needed to debug problems
All	All events are logged. If you create custom levels, All includes your custom levels, and

convertlog command

The convertlog command is used to convert the message IDs in log entries from the old standard to the new standard, or the new standard back to the old.

Basic format Message events displayed in basic format use the following format. The notation <name> indicates mandatory fields that will always appear in the basic format message. The notation [name] indicates optional or conditional fields that will be included if they can be determined. <timestamp><threadId><shortName><eventType>[className][methodName]<message>

CORBA minor codes

Applications that use CORBA services generate minor codes to indicate the underlying cause of a failure. These codes are written to the exception stack. Look for ″minor code″ in the exception stack to locate these exceptions.

Overview

Common Object Request Broker Architecture (CORBA) is an industry-wide standard for object-oriented communication between processes, which is supported in several programming languages. Several subcomponents of the product use CORBA to communicate across processes.

When a CORBA process fails, that is a request from one process to another cannot be sent, completed, or returned, a high-level exception is created, such as TransactionRolledBackException: CORBA TRANSACTION_ROLLEDBACK. Minor codes that are used by product components.

Range	Related subcomponent	Where to find details
0x49424300-0x494243FF	Security	Security components troubleshooting tips
0x49421050-0x4942105F, 0x49421070-0x4942107F	ORB services	Object request broker troubleshooting tips
0x4f4d and above	Standard CORBA exceptions	http://www.omg.org
0x49421080-0x4942108F	Naming services	
0x49421080-0x4942108F	Workload Management	

Collector Tool

Gathering information with the collector tool The collector tool gathers information about your WebSphere Application Server installation and packages it in a Java archive (JAR) file that you can send to IBM Customer Support to assist in determining and analyzing your problem. Information in the JAR file includes logs, property files, configuration files, operating system and Java data, and the presence and level of each software prerequisite.

The sort of information that you gather is not something that most people use. In fact, the collector tool packages its output into a JAR file. IBM includes the collector tool in the product code, along with other tools that help capture the information that you must provide when reporting a problem.

The collector tool is part of a strategy of making problem reporting as easy and complete as possible. There are two phases of using the collector tool. The first phase runs the collector tool on your WebSphere Application Server product and produces a Java archive (JAR) file. The IBM Support team performs the second phase, which is analyzing the Java archive (JAR) file that the collector program produces. The collector program runs to completion as it creates the JAR file, despite any errors that it might find like missing files or invalid commands. The collector tool collects as much data in the JAR file as possible. The collector tool is a Java application that requires a Java SE Runtime Environment 6 (JRE6) to run.

The tool is within the installation root directory for WebSphere Application Server Network Deployment. But you run the tool from a working directory that you create outside of the installation root directory. This procedure describes both of those steps and all of the other steps for using the tool and reporting the results from running the tool.

B) Use trace facility (e.g., enabling, selecting components, and log configuration).

Working with trace

Use trace to obtain detailed information about running the WebSphere® Application Server components, including application servers, clients, and other processes in the environment.

Trace files show the time and sequence of methods called by WebSphere Application Server base classes, and you can use these files to pinpoint the failure. Collecting a trace is often requested by IBM® technical support personnel. If you are not familiar with the internal structure of WebSphere Application Server, the trace output might not be meaningful to you.

[z/OS] You can configure trace settings with the administrative console, or you can configure tracing from the MVS™ console using the modify command.

Procedure

1. Configure an output destination to which trace data is sent.

2. Enable trace for the appropriate WebSphere Application Server or application components.

3. Run the application or operation to generate the trace data.

4. [AIX Solaris HP-UX Linux Windows] [iSeries] Analyze the trace data or forward it to the appropriate organization for analysis.

5. [z/OS] Analyze the trace data or forward it to the appropriate organization for analysis.

Log and trace settings

To view this administrative console page, click:

[AIX Solaris HP-UX Linux Windows] [iSeries] Troubleshooting > Logs and Trace > server_name

[z/OS] Servers > Application Servers > server_name > Troubleshooting > Diagnostic Trace Service

[z/OS] Note: You can configure tracing from the MVS™ console using the modify command.

Diagnostic Trace

The diagnostic trace configuration settings for a server process determine the initial trace state for a server process. The configuration settings are read at server startup and used to configure the trace service. You can also change many of the trace service properties or settings while the server process is running.

Java virtual machine (JVM) Logs [AIX Solaris HP-UX Linux Windows] [iSeries]

The JVM logs are created by redirecting the System.out and System.err streams of the JVM to independent log files. WebSphere® Application Server writes formatted messages to the System.out stream. In addition, applications and other code can write to these streams using the print() and println() methods defined by the streams.

Process Logs [AIX Solaris HP-UX Linux Windows] [iSeries]

WebSphere Application Server processes contain two output streams that are accessible to native code running in the process. These streams are the stdout and stderr streams. Native code, including Java™ virtual machines (JVM), might write data to these process streams. In addition, JVM provided System.out and System.err streams can be configured to write their data to these streams also.

IBM Service Logs [AIX Solaris HP-UX Linux Windows] [iSeries]

The IBM® service log contains both the WebSphere Application Server messages that are written to the System.out stream and some special messages that contain extended service information that is normally not of interest, but can be important when analyzing problems. There is one service log for all WebSphere Application Server JVMs on a node, including all application servers. The IBM Service log is maintained in a binary format and requires a special tool to view. This viewer, the Log and Trace Analyzer, provides additional diagnostic capabilities. In addition, the binary format provides capabilities that are utilized by IBM support organizations.

Change Log Level Details

Enter a log detail level that specifies the components, packages, or groups to trace. The log detail level string must conform to the specific grammar described in this topic. You can enter the log detail level string directly, or generate it using the graphical trace interface.

Enabling trace on a running server

Use the administrative console to enable tracing on a running server. You can use trace to assist you in monitoring system performance and diagnosing problems.

About this task

You can modify the trace service state that determines which components are being actively traced for a running server by using the following procedure.

[z/OS] You can also configure tracing from the MVS™ console using the modify command.

Procedure

1. Start the administrative console.
2. [AIX Solaris HP-UX Linux Windows] [iSeries] Click Troubleshooting > Logs and Trace in the console navigation tree, then click server > Diagnostic Trace.
3. [z/OS] Click Servers > Application Servers > server_name > Troubleshooting > Diagnostic Trace Service.
4. Select the Runtime tab.
5. Select the Save runtime changes to configuration as well check box if you want to write your changes back to the server configuration.
6. Change the existing trace state by changing the trace specification to the desired state.
7. Configure the trace output if a change from the existing one is desired.
8. Click Apply.

Turning traces on and off in servers processes using scripting

You can use scripting to turn traces on or off in server processes.

Before starting this task, the wsadmin tool must be running.

Perform the following steps to turn traces on and off in server processes:

Procedure

1. Identify the object name for the TraceService MBean running in the process:

* Using Jacl:

```
$AdminControl completeObjectName type=TraceService,node=mynode,process=server1,*
```

* Using Jython:

```
AdminControl.completeObjectName('type=TraceService,node=mynode,process=server1,*')
```

2. Obtain the name of the object and set it to a variable:

* Using Jacl:

```
set ts [$AdminControl completeObjectName type=TraceService,process=server1,*]
```

* Using Jython:

```
ts = AdminControl.completeObjectName('type=TraceService,process=server1,*')
```

3. Turn tracing on or off for the server. For example:

* To turn tracing on, perform the following step:

Using Jacl:

```
$AdminControl setAttribute $ts traceSpecification com.ibm.*=all=enabled
```

Using Jython:

```
AdminControl.setAttribute(ts, 'traceSpecification', 'com.ibm.*=all=enabled')
```

* To turn tracing off, perform the following step:

Using Jacl:

```
$AdminControl setAttribute $ts traceSpecification com.ibm.*=all=disabled
```

Using Jython:

```
AdminControl.setAttribute(ts, 'traceSpecification', 'com.ibm.*=all=disabled')
```

C) Analyze the content of the JNDI namespace using dumpNameSpace.

You can use the dumpNameSpace tool to dump the contents of a namespace accessed through a name server. The dumpNameSpace tool is based on Java™ Naming and Directory Interface (JNDI).

When you run the dumpNameSpace tool, the naming service must be active. The dumpNameSpace tool cannot dump namespaces local to the server process, such as those with java: and local: URL schemes. The local: namespace contains references to enterprise beans with local interfaces. Use the namespace dump utility for java:, local: and server namespaces to dump java: and local: namespaces.

The tool dumps the server root context for the server at the specified host and port unless you specify a non-default starting context which precludes it. The tool does not dump the server root contexts for other servers.

Avoid trouble: If your system has more than one IP address, you need to modify the order in which the system configured name lookup mechanism processes during the running of the dumpNameSpace tool. You need to use the-Dsun.net.spi.nameservice.provider.1=dns,sun setting, a Java option, to ensure that the dumpNameSpace tool works successfully on systems which have more than one IP address.gotcha

Running dumpNameSpace

You can run the tool from a command line or using its program interface. This topic describes command-line invocations. To access the dumpNameSpace tool through its program interface, refer to the class com.ibm.websphere.naming.DumpNameSpace in the WebSphere® Application Server API documentation.

To run the tool from a command line, enter the dumpNameSpace command from the app_server_root/bin directory.

[AIX] [HP-UX] [Linux] [Solaris]

```
dumpNameSpace.sh [[-keyword value]...]
```

[Windows]

```
dumpNameSpace [[-keyword value]...]
```

If you run the dumpNameSpace tool with security enabled and the com.ibm.CORBA.loginSource property is set in the profile_root/properties/sas.client.props file, a login prompt is displayed.

If you cancel the login prompt, the dumpNameSpace tool continues outbound with an "UNAUTHENTICATED" credential. Thus, by default, an "UNAUTHENTICATED" credential is used that is equivalent to the "Everyone" access authorization policy. You can modify this default setting by changing the value for the com.ibm.CSI.performClientAuthenticationRequired property to true in the app_server_root/properties/sas.client.props file.

If you do not set the com.ibm.CORBA.loginSource property in the sas.client.props file, the dumpNameSpace tool continues outbound with the user name and password that is set in the credential.

If Kerberos (KRB5) is enabled for administrative authentication, the authenticationTarget supports both BasicAuth and KRB5. To use Kerberos authentication, you must update the sas.client.props, soap.client.props, and ipc.client.props files according to the connector type. When using Kerberos authentication, the user password does not flow across the wire. A one-way hash of the password identifies the client.

Parameters

The keywords and associated values for the dumpNameSpace tool follow:

-host myhost.company.com

Indicates the bootstrap host or the WebSphere Application Server host whose namespace you want to dump. The value defaults to localhost. Specify a value for -host if the tool is not run from the local machine. The -host parameter instructs the tool to connect to a server on a remote machine. For example, run dumpNameSpace -host myhost.mycompany.com to display the namespace of the server running on myhost.mycompany.com.

-port nnn

Indicates the bootstrap port which, if not specified, defaults to 2809.

-root { cell | server | node | host | legacy | tree | default }

Indicates the root context to use as the initial context for the dump. The applicable root options and default root context depend on the type of name server from which the dump is being obtained. Descriptions of -root options follow.

For WebSphere Application Server servers:

Table 1. -root option descriptions for product servers. The root context provides the initial context for the dump.

-root option	Description
cell	DumpNameSpace default for product Version 5.0 or later servers. Dumps the tree starting at the cell root context.
server	Dumps the tree starting at the server root context.
node	Dumps the tree starting at the node root context.
tree	Dumps the tree starting at the tree root context.

For all WebSphere Application Server and other name servers:

Table 2. -root option description for product and non-product servers. The root context provides the initial context for the dump.

-root option	Description
default	Dumps the tree starting at the initial context which JNDI returns by default for that server type. This is the only -root option that is compatible with non-product name servers.

-url some_provider_URL

Indicates the value for the java.naming.provider.url property used to get the initial JNDI context. This option can be used in place of the -host, -port, and -root options. If the -url option is specified, the -host, -port, and -root options are ignored.

-factory com.ibm.websphere.naming.WsnInitialContextFactory

Indicates the initial context factory to be used to get the JNDI initial context. The value defaults to com.ibm.websphere.naming.WsnInitialContextFactory. The default value generally does not need to be changed.

-startAt some/subcontext/in/the/tree

Indicates the path from the bootstrap host's root context to the top level context where the dump should begin. The tool recursively dumps subcontexts below this point. It defaults to an empty string, that is, the bootstrap host root context.

-format { jndi | ins }

Table 3. -format option descriptions. Options include jndi and ins.

-format option	Description
jndi	The default. Displays name components as atomic strings.
ins	Shows name components parsed using Interoperable Naming Service (INS) rules (id.kind).

-report { short | long }

Table 4. -report option descriptions. Options include short and long.

-report option	Description
short	The default. Dumps the binding name and bound object type. This output is also provided by JNDI Context.list().
long	Dumps the binding name, bound object type, local object type, and string representation of the local object (that is, the IORs, string values, and other values that are printed).

For objects of user-defined classes to display correctly with the long report option, you might need to add their containing directories to the list of directories searched. Set the environment variable WAS_USER_DIRS at a command line. The value can include one or more directories.

[AIX] [HP-UX] [Linux] [Solaris]

```
WAS_USER_DIRS=/usr/classdir1:/usr/classdir2 export WAS_USER_DIRS
```

[Windows]

```
  set WAS_USER_DIRS=c:\classdir1;d:\classdir2
```

All .zip, .jar, and .class files in the specified directories can then be resolved by the class loader when running the dumpNameSpace tool.

-traceString "some.package.name.to.trace.*=all=enabled"

Represents the trace string with the same format as that generated by the servers. The output is sent to the file DumpNameSpaceTrace.out.

Return codes

The dumpNameSpace tool has the following return codes:

Table 5. dumpNameSpace tool return codes. A return code of 0 indicates no error. Return codes of 1 through 3 indicate an error condition.

Return code	Description
0	Normal system exit. No error resulted from running dumpNameSpace.
1	Error in getting the starting context
2	Other error occurred with exception. Running dumpNameSpace resulted in an error other than an error in getting the starting context.
3	Unsupported option specified

When you run the dumpNameSpace tool, the naming service must be active. The dumpNameSpace tool cannot dump namespaces local to the server process, such as those with java: and local: URL schemes. The local: namespace contains references to enterprise beans with local interfaces. Use the namespace dump utility for java:, local: and server namespaces to dump java: and local: namespaces.

Invoke the namespace dump tool from a command line by entering either of the following commands:

```
dumpNameSpace -?

dumpNameSpace -host myhost.mycompany.com -port 901

dumpNameSpace -url corbaloc:iiop:myhost.mycompany.com:901
```

You can access the dumpNameSpace tool through its program interface. Refer to the class com.ibm.websphere.naming.DumpNameSpace in the WebSphere® Application Server API documentation for details on the DumpNameSpace program interface.

If you cannot use the dumpNameSpace command line utility to dump the java: namespace for a Java Platform, Enterprise Edition (Java EE) specification application because the application's java: namespace is accessible only by that application, run the namespace dump utility for java:, local: and server namespaces.

If you see an exception that appears to be CORBA related ("CORBA" appears as part of the exception name) look for a naming-services-specific CORBA minor code, further down in the exception stack, for information on the real cause of the problem. For a list of naming service exceptions and explanations, see the class com.ibm.websphere.naming.WsnCorbaMinorCodes in the API documentation that is included in the Reference section of the information center.

Develop your application using either JNDI or CORBA CosNaming interfaces. Use these interfaces to look up server application objects that are bound into the namespace and obtain references to them. Most Java developers use the JNDI interface. However, the CORBA CosNaming interface is also available for performing Naming operations on WebSphere Application Server name servers or other CosNaming name servers.

Assemble your application using an assembly tool. Application assembly is a packaging and configuration step that is a prerequisite to application deployment. If the application you are assembling is a client to an application running in another process, you should qualify the jndiName values in the deployment descriptors for the objects related to the other application. Otherwise, you might need to override the names with qualified names during application deployment. If the objects have fixed qualified names configured for them, you should use them so that the jndiName values do not depend on the other application's location within the topology of the cell.

The following example shows a lookup of an EJB 3.0 remote business interface. The actual home lookup name is determined by the interface's ibm-ejb-jar-bnd.xml binding file, if present, or by the default name assigned by the EJB container if no binding file is present. For more information, see topics on default bindings for business interfaces and homes and on user-defined bindings for EJB business interfaces and homes.

```
// Get the initial context as shown in a previous example.
// Look up the business interface using the JNDI name.
try {
        java.lang.Object ejbBusIntf =
        initialContext.lookup(
        "java:comp/env/com/mycompany/accounting/Account");
        accountIntf =(Account)javax.rmi.PortableRemoteObject.narrow(ejbBusIntf,
Account.class);
}
catch (NamingException e) { // Error getting the business interface
}
```

The following example shows a lookup of an EJB 1.x or 2.x EJB home. The actual home lookup name is determined by the application's deployment descriptors. The enterprise bean (EJB) resides in an EJB container, which provides an interface between the bean and the application server on which it resides.

```
// Get the initial context as shown in a previous example
// Look up the home interface using the JNDI name
try {
        java.lang.Object ejbHome =
        initialContext.lookup("java:comp/env/com/mycompany/accounting/AccountEJB");
        accountHome =
(AccountHome)javax.rmi.PortableRemoteObject.narrow( (org.omg.CORBA.Object) ejbHome,
AccountHome.class);
}
catch (NamingException e) { // Error getting the home interface
}
```

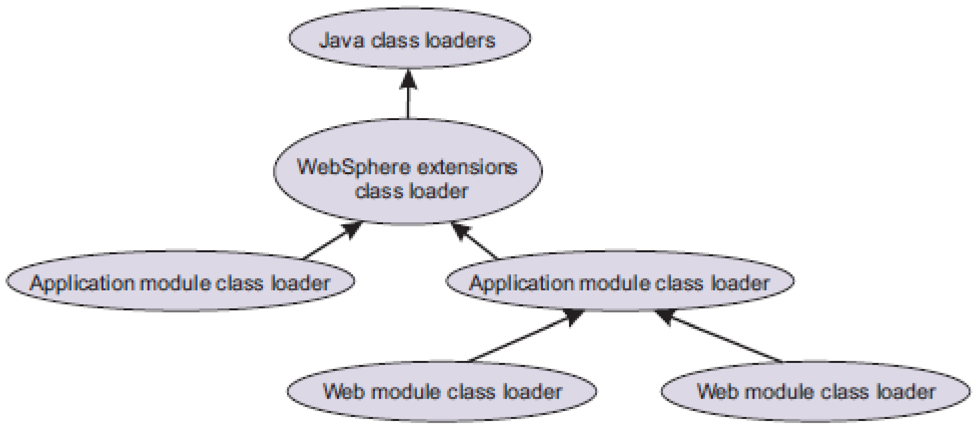
D) Perform JVM troubleshooting tasks (e.g., thread dump, JVM core dump, and heap dump, verbose Garbage Collection (GC)).

Troubleshooting class loaders

Class loaders find and load class files. For a deployed application to run properly, the class loaders that affect the application and its modules must be configured so that the application can find the files and resources that it needs. Diagnosing problems with class loaders can be complicated and time-consuming. To diagnose and fix the problems more quickly, use the administrative console class loader viewer to examine class loaders and the classes loaded by each class loader.

Before you begin

This topic assumes that you have installed an application on a server supported by the product and you want to examine class loaders used by the application or its modules. The modules can be Web modules (.war files) or enterprise bean (EJB) modules (.jar files). The class loader viewer enables you to examine class loaders in a runtime environment. This topic also assumes that you have enabled the class loader viewer service. Click Servers → Server Types → WebSphere application servers → server_name → Class loader viewer service, enable the service and restart the server. About this task The runtime environment of WebSphere Application Server uses the following class loaders to find and load new classes for an application in the following order: 1. The bootstrap, extensions, and CLASSPATH class loaders created by the Java virtual machine 2. A WebSphere extensions class loader 3. One or more application module class loaders that load elements of enterprise applications running in the server 4. Zero or more Web module class loaders.


Java class loaders
WebSphere extensions class loader
Application module class loader
Application module class loader
Web module class loader
Web module class loader

Each class loader is a child of the previous class loader. That is, the application module class loaders are children of the WebSphere extensions class loader, which is a child of the CLASSPATH Java class loader. Whenever a class needs to be loaded, the class loader usually delegates the request to its parent class loader. If none of the parent class loaders can find the class, the original class loader attempts to load the class. Requests can only go to a parent class loader; they cannot go to a child class loader. After a class is loaded by a class loader, any new classes that it tries to load reuse the same class loader or go up the precedence list until the class is found. If the class loaders that load the artifacts of an application are not configured properly, the Java virtual machine (JVM) might throw a class loading exception when starting or running that application.

The types of exceptions include:

ClassCastException

ClassNotFoundException

NoClassDefFoundException

UnsatisfiedLinkError

Use the class loader viewer to examine class loaders and correct problems with application or class loader configurations.

ClassCastException

A class cast exception results when the following conditions exist and can be corrected by the following actions:

The type of the source object is not an instance of the target class (type).

The class loader that loaded the source object (class) is different from the class loader that loaded the target class.

The application fails to perform or improperly performs a narrow operation.

ClassNotFoundException

A class not found exception results when the following conditions exist and can be corrected by the following actions:

The class is not visible on the logical classpath of the context class loader.

The application incorrectly uses a class loader API.

A dependent class is not visible.

NoClassDefFoundException

A no class definition found exception results when the following conditions exist and can be corrected by the following actions:

The class is not in the logical class path.

The class cannot load. There are various reasons for a class not loading. The reasons include: failure to load the dependent class, the dependent class has a bad format, or the version number of a class.

UnsatisfiedLinkError

A linkage error results when the following conditions exist and can be corrected by the following actions:

A user action caused the error.

System.mapLibraryName returns the wrong library file.

The native library is already loaded.

A dependent native library was used.

JVM log interpretation

Message formats Formatted messages are written to the JVM logs in one of two formats:

Basic Format The format used in earlier versions of WebSphere Application Server.

Advanced Format Extends the basic format by adding information about an event, when possible.

Basic and advanced format fields

Basic and Advanced Formats use many of the same fields and formatting techniques. The various fields that may be found in these formats follow:

TimeStamp The timestamp is formatted using the locale of the process where it is formatted. It includes a fully qualified date (for example YYMMDD), 24 hour time with millisecond precision and a time zone.

ThreadId An 8 character hexadecimal value generated from the hash code of the thread that issued the message.

ThreadName The name of the Java thread that issued the message or trace event.

ShortName The abbreviated name of the logging component that issued the message or trace event. This is typically the class name for WebSphere Application Server internal components, but can be some other identifier for user applications.

LongName The full name of the logging component that issued the message or trace event. This is typically the fully qualified class name for WebSphere Application Server internal components, but can be some other identifier for user applications.

EventType A one character field that indicates the type of the message or trace event. Message types are in upper case. Possible values include:

F A Fatal message.

E An Error message.

W A Warning message.

A An Audit message.

I An Informational message.

C An Configuration message.

D A Detail message.

O A message that was written directly to System.out by the user application or internal components.

R A message that was written directly to System.err by the user application or internal components.

Z A placeholder to indicate the type was not recognized.

ClassName The class that issued the message or trace event.

MethodName The method that issued the message or trace event.

Organization The organization that owns the application that issued the message or trace event.

Product The product that issued the message or trace event.

Component The component within the product that issued the message or trace event.

Basic format

Message events displayed in basic format use the following format. The notation <name> indicates mandatory fields that will always appear in the basic format message. The notation [name] indicates optional or conditional fields that will be included if they can be determined. <timestamp><threadId><shortName><eventType>[className][methodName]<message>

Advanced format

Message events displayed in advanced format use the following format. The notation <name> is used to indicate mandatory fields that will always appear in the advanced format for message entries. The notation [name] is used to indicate optional or conditional fields that will be included if they can be determined.

<timestamp><threadId><eventType><UOW><source=longName>[className] [methodName]<Organization><Product><Component> [thread=threadName]<message>

Configuring the hang detection policy

The hang detection option for WebSphere Application Server is turned on by default. You can configure a hang detection policy to accommodate your applications and environment so that potential hangs can be reported, providing earlier detection of failing servers. When a hung thread is detected, WebSphere Application Server notifies you so that you can troubleshoot the problem.

Before you begin

A common error in Java Platform, Enterprise Edition (Java EE) applications is a hung thread. A hung thread can result from a simple software defect (such as an infinite loop) or a more complex cause (for example, a resource deadlock). System resources, such as CPU time, might be consumed by this hung transaction when threads run unbounded code paths, such as when the code is running in an infinite loop. Alternately, a system can become unresponsive even though all resources are idle, as in a deadlock scenario. Unless an end user or a monitoring tool reports the problem, the system may remain in this degraded state indefinitely. Using the hang detection policy, you can specify a time that is too long for a unit of work to complete. The thread monitor checks all managed threads in the system (for example, Web container threads and object request broker (ORB) threads) . Unmanaged threads, which are threads created by applications, are not monitored.

The thread hang detection option is enabled by default. To adjust the hang detection policy values, or to disable hang detection completely:

1. From the administrative console, click Servers > Application Servers > server_name

2. Under Server Infrastructure, click Administration > Custom Properties

3. Click New.

4. Add the following properties:

Name: com.ibm.websphere.threadmonitor.interval

Value: The frequency (in seconds) at which managed threads in the selected application server will be interrogated. Default: 180 seconds (three minutes).

Name: com.ibm.websphere.threadmonitor.threshold

Value: The length of time (in seconds) in which a thread can be active before it is considered hung. Any thread that is detected as active for longer than this length of time is reported as hung. Default: The default value is 600 seconds (ten minutes).

Name:com.ibm.websphere.threadmonitor.false.alarm.threshold

Value: The number of times (T) that false alarms can occur before automatically increasing the threshold. It is possible that a thread that is reported as hung eventually completes its work, resulting in a false alarm. A large number of these events indicates that the threshhold value is too small. The hang detection facility can automatically r espond to this situation: For every T false alarms, the threshold T is increased by a factor of 1.5. Set the value to zero (or less) to disable the automatic adjustment. Default: 100

Name: com.ibm.websphere.threadmonitor.dump.java

Value: Set to true to cause a javacore to be created when a hung thread

is detected and a WSVR0605W message is printed. The threads section of the javacore can be analyzed to determine what the reported thread and other related threads are doing.

Default: False

To disable the hang detection option, set the com.ibm.websphere.threadmonitor.interval property to less than or equal to zero.

5. Click Apply.

6. Click OK.

7. Save the changes. Make sure a file synchronization is performed before restarting the servers.

8. Restart the Application Server for the changes to take effect.

Collector Tool

Gathering information with the collector tool The collector tool gathers information about your WebSphere Application Server installation and packages it in a Java archive (JAR) file that you can send to IBM Customer Support to assist in determining and analyzing your problem. Information in the JAR file includes logs, property files, configuration files, operating system and Java data, and the presence and level of each software prerequisite.

Before you begin

The sort of information that you gather is not something that most people use. In fact, the collector tool packages its output into a JAR file. IBM includes the collector tool in the product code, along with other tools that help capture the information that you must provide when reporting a problem. The collector tool is part of a strategy of making problem reporting as easy and complete as possible.

There are two phases of using the collector tool. The first phase runs the collector tool on your WebSphere Application Server product and produces a Java archive (JAR) file. The IBM Support team performs the second phase, which is analyzing the Java archive (JAR) file that the collector program produces. The collector program runs to completion as it creates the JAR file, despite any errors that it might find like missing files or invalid commands. The collector tool collects as much data in the JAR file as possible.

The collector tool is a Java application that requires a Java SE Runtime Environment 6 (JRE6) to run.

About this task

The tool is within the installation root directory for WebSphere Application Server Network Deployment. But you run the tool from a working directory that you create outside of the installation root directory. This procedure describes both of those steps and all of the other steps for using the tool and reporting the results from running the tool.

There are two ways to run the collector tool. Run the collector tool to collect summary data or to traverse the system to gather relevant files and command results. The collector tool produces a Java archive (JAR) file of information needed to determine and solve a problem. The collector summary option produces a lightweight collection of version and other information that is useful when first reporting the problem to IBM Support. Run the collector tool from the root user or from the administrator user to access system files that contain information about kernel settings, installed packages, and other vital data.

Results

The collector program creates the Collector.log log file and an output JAR file in the current directory.

The name of the JAR file is composed of the host name, cell name, node name, and profile name: host_name-cell_name-node_name-profile_name.JAR

The Collector.log log file is one of the files collected in the host_name-cell_name-node_name-profile_name.JAR file.

Configuring first failure data capture log file purges

The first failure data capture (FFDC) log file saves information that is generated from a processing failure. These files are deleted after a maximum number of days has passed. The captured data is saved in a log file for analyzing the problem.

The first failure data capture (FFDC) feature preserves the information that is generated from a processing failure and returns control to the affected engines. The captured data is saved in a log file for analyzing the problem. FFDC is intended primarily for use by IBM Support. FFDC instantly collects events and errors that occur during the product runtime. The information is captured as it occurs and is written to a log file that can be analyzed by IBM Support personnel. The data is uniquely identified for the servant region that produced the exception.

The FFDC configuration properties files are located in the properties directory under the Application Server product installation. You must set the ExceptionFileMaximumAge property to the same value in all three files: ffdcRun.properties, ffdcStart.properties, and ffdcStop.properties. You can set the ExceptionFileMaximumAge property to configure the amount of days between purging the FFDC log files. The value of the ExceptionFileMaximumAge property must be a positive number. The FFDC feature does not affect the performance of the Application Server product.

IBM Support Assistant

IBM Support Assistant is a free troubleshooting application that helps you research, analyze, and resolve problems using various support features and tools. IBM Support Assistant enables you to find solutions yourself using the same troubleshooting techniques used by the IBM Support team, and it allows you to organize and transfer your troubleshooting efforts between members of your team or to IBM for further support.

IBM Support Assistant version 4.0 enhancements include:

Remote System Troubleshooting: Explore file systems, run automated data collectors and troubleshooting tools, and view the system inventory on remote systems.

Activity-based Workflow: Choose from support-related activities, or use the Guided Troubleshooter for step-by-step help with analysis and resolution.

Case Management: Organize your troubleshooting data in ″cases″; then export and share these cases with other problem analysts or with IBM Support.

Improved Flexibility: Add your own search locations, control updates by hosting your own update site, get the latest product news and updates.

Diagnosing problems using IBM Support Assistant tooling

The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. About this task

Tools for IBM Support Assistant perform numerous functions from memory-heap dump analysis and Java core-dump analysis to enabling remote assistance from IBM Support. All of these tools come with help and usage documentation that allow you to learn about the tools and start using them to analyze and resolve your problems. The following are samples of the tools available in IBM Support Assistant:

Memory Dump Diagnostic for Java (MDD4J)

The Memory Dump Diagnostic for Java tool analyzes common formats of memory dumps (heap dumps) from the Java virtual machine (JVM) that is running the WebSphere Application Server or any other standalone Java applications. The analysis of memory dumps is targeted towards identifying data structures within the Java heap that might be root causes of memory leaks. The analysis also identifies major contributors to the Java heap footprint of the application and their ownership relationship. The tool is capable of analyzing very large memory dumps obtained from production-environment application servers encountering OutOfMemoryError issues.

IBM Thread and Monitor Dump Analyzer (TMDA)

IBM Thread and Monitor Dump Analyzer (TMDA) provides analysis for Java thread dumps or javacores such as those from WebSphere Application Server. You can analyze thread usage at several different levels, starting with a high-level graphical view and drilling down to a detailed tally of individual threads. If any deadlocks exist in the thread dump, TMDA detects and reports them.

Log Analyzer

Log Analyzer is a graphical user interface that provides a single point of contact for browsing, analyzing, and correlating logs produced by multiple products. In addition to importing log files from multiple products, Log Analyzer enables you to import and select symptom catalogs against which log files can be analyzed and correlated.

IBM Visual Configuration Explorer

The IBM Visual Configuration Explorer provides a way for you to visualize, explore, and analyze configuration information from diverse sources.

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT)

The IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) parses IBM verbose garbage-collection (GC) trace, analyzes Java heap usage, and recommends key configurations based on pattern modeling of Java heap usage. Only verbose GC traces that are generated from IBM Java Development Kits (JDKs) are supported.

IBM Assist On-site

IBM Assist On-site provides remote desktop capabilities. You run this tool when you are instructed to do so by IBM Support personnel. With this live remote-assistance tool, a member of the IBM

Support team can view your desktop and share control of your mouse and keyboard to help you find a solution. The tool can speed up problem determination, data collection, and ultimately your problem solution.

<u>Verbose garbage collection</u>

Specifies whether to use verbose debug output for garbage collection. The default is not to enable verbose garbage collection.

Data type	Boolean
Default	false

When this field is enabled, a report is written to the output stream each time the garbage collector runs. This report should give you an indication of how the Java garbage collection process is functioning.

You can check the verboseGC report to determine:

* How much time the JVM is spending performing garbage collection.

Ideally, you want the JVM to spend less than 5 percent of its processing time doing garbage collection. To determine the percentage of time the JVM spends in garbage collection, divide the time it took to complete the collection by the length of time since the last AF and multiply the result by 100. For example,

83.29/3724.32 * 100 = 2.236 percent

If you are spending more than 5 percent of your time in garbage collection and if garbage collection is occurring frequently, you might need to increase your Java heap size.

* If the allocated heap is growing with each garbage collection occurrence.

To determine if the allocated heap is growing, look at the percentage of the heap that is remains unallocated after each garbage collection cycle, and verify that the percentage is not continuing to decline. If the percentage of free space continues to decline you are experiencing a gradual growth in the heap size from garbage collection to garbage collection. This situation might indicate that your application has a memory leak.

To help you analyze memory leak problems when memory leak detection occurs, use the Heap Analysis Tools for Java™. Use the Heap Analysis Tools component (also known as Heap Analyzer) to perform Java application heap analysis and object create profiling (size and identification) over time. Heap Analyzer includes information about:

* Java virtual machine (JVM) heap growth or size

* The objects being created that include type of object, count and object size, object heap size

* The application "Heap Footprint®" for memory sizing and performance considerations

* Includes a call stack for every snapshot when running in profile mode so objects created can be correlated to functions in the application.

Do not analyze heap dumps on the WebSphere® Application Server machine because analysis is very expensive. For analysis, transfer heap dumps to a dedicated problem determination machine.

About this task

When a memory leak is detected and heap dumps are generated, you must analyze heap dumps on a problem determination machine and not on the application server because the analysis is very central processing unit (CPU) and disk I/O intensive.

IBM® heap dump files are usually named in the following way:

```
heapdump.<date>..<timestamp><pid>.phd
```

Configure the heap size.

The heap size settings control garbage collection in the Java SE Development Kit 6 (Classic) that is provided with IBM i. The initial heap size is a threshold that triggers new garbage collection cycles. For example, if the initial heap size is 10 MB, a new collection cycle is triggered as soon as the JVM detects that 10 MB have been allocated since the last collection cycle.

Smaller heap sizes result in more frequent garbage collections than larger heap sizes. If the maximum heap size is reached, the garbage collector stops operating asynchronously, and user threads are forced to wait for collection cycles to complete. This situation has a significantly negative impact on performance. A maximum heap size of 0 (*NOMAX) assures that garbage collection operates asynchonously.

The maximum heap size can affect application performance. The maximum heap size specifies the maximum amount of object space that the garbage collected heap can consume. If the maximum heap size is too small, performance might decrease significantly, or the application might receive out of memory errors when the maximum heap size is reached.

Because of the complexity of determining a correct value for the maximum heap size, a value of 0, which indicates that there is no size limit, is recommended unless an absolute limit on the object space for the garbage collected heap size is required.

If , because of memory limitations, you need to set a maximum heap size other than *NOMAX, you should run multiple tests to determine the proper value for the maximum heap size. Running multiple tests, helps to determine the appropriate value for your configurations and workload combinations. To prevent a run-away JVM, set the maximum heap size to a value that is larger than the size to which you expect the heap to grow, but not so large that it affects the performance of the rest of the machine.

For one of the tests you should complete the following actions:

1. Run your application server under a heavy workload with a maximum heap value of 0.

2. Use the DMPJVM command or iDoctor to determine the maximum size of the garbage collected heap for the JVM.

3. Multiply the size of the garbage collection heap by 1.25. The result is a reasonable estimate for maximum heap size because the smallest acceptable value for the maximum heap size is 125 percent of the garbage collected heap size.

Because you can specify a larger value for the maximum heap size without affecting performance, it is recommended that you set the largest possible value based on the resource restrictions of the JVM or the limitations of your system configuration.

After you determine an appropriate value for the maximum heap size, you might need to set up or adjust the pool in which the JVM runs. By default, application server jobs run in the base system pool, which is storage pool 2 as defined by system value WRKSYSSTS. However, you can specify a different pool. Do not set the maximum heap size to a value that is larger than 125 percent of the size of the pool in which the JVM is running. It is recommended that you run the JVM in its own memory pool with the memory permanently assigned to that pool, if possible.

If the performance adjuster is set to adjust the memory pools, that is, the system value QPFRADJ is set to a value other than 0, then it is recommended that you use the system value WRKSHRPOOL to specify a minimum size for the pool. The minimum size should be approximately equal to your garbage collected heap working set size. Setting a correct maximum heap size, and properly configuring the memory pool can prevent a JVM with a memory leak from consuming system resources, while yielding high performance.

When a JVM must run in a shared pool, it is more difficult to determine an appropriate value for the maximum heap size. Other jobs running in the pool can cause the garbage collected heap pages to be aged out of the pool. If the garbage collected heap pages are removed from the pool because of their age, the garbage collector must fault the pages back into the pool on the next garbage collection cycle because the garbage collector requires access to all of the pages in the garbage collected heap. The Classic JVM does not stop all of the JVM threads to clean the heap, you might expect that excessive page faulting causes the garbage collector to slow down and the garbage collected heap to grow. However, the operating system automatically increases the size of the heap, and the threads continue to run.

This heap growth is an artificial inflation of the garbage collected heap working set size, and must be considered if you want to specify a maximum heap value. When a small amount of artificial inflation occurs, the garbage collector reduces the size of the heap over time if the space remains unused and the activity in the pool returns to a steady state. However, in a shared pool, you might experience the following problems if the maximum heap size is not set correctly:

* If the maximum heap size is too small, artificial inflation can result in severe performance degradation or system failure if the JVM experiences an out-of-memory error.

* If the maximum heap size is set too large, the garbage collector might reach a point where it is unable to recover the artificial inflation of the garbage collected heap. In this case, performance is also negatively affected. A value that is too large might also keep the garbage collector from preventing a JVM failure. Even if the value is too large, the garbage collector can still prevent the JVM from consuming excessive amounts of system resources.

If you must set the maximum heap size to guarantee that the heap size does not exceed a given level, specify an initial heap size that is 80 - 90 percent smaller than the maximum heap size.

However, specify a value that is large enough to not negatively affect performance.

The JVM uses defined thresholds to manage the storage that it is allocated. When the thresholds are reached, the garbage collector is invoked to free up unused storage. Therefore, garbage collection can cause significant degradation of Java performance. Before changing the initial and maximum heap sizes, you should consider the following information:

* In the majority of cases you should set the maximum JVM heap size to a value that is higher than the initial JVM heap size. This setting allows for the JVM to operate efficiently during normal, steady state periods within the confines of the initial heap. This setting also allows the JVM to operate effectively during periods of high transaction volume because the JVM can expand the heap up to the value specified for the maximum JVM heap size. In some rare cases, where absolute optimal performance is required, you might want to specify the same value for both the initial and maximum heap size. This setting eliminates some overhead that occurs when the JVM expands or contracts the size of the JVM heap. Before changing any of the JVM heap sizes, verify that the JVM storage allocation is large enough to accommodate the new heap size.

* Do not make the size of the initial heap so large that while it initially improves performance by delaying garbage collection, when garbage collection does occur, the collection process affects response time because the process has to run longer.

Although heap dumps are generated only in response to a detected memory leak, you must understand that generating heap dumps can have a severe performance impact on WebSphere Application Server for several minutes. When generating multiple heap dumps manually for memory leak analysis, make sure that significant objects are leaked in between the two heap dumps. This approach enables problem determination tools to identify the source of the memory leak.

Dump Analyzer overview

The IBM® Monitoring and Diagnostic Tools for Java™ - Dump Analyzer (referred to hereafter as the Dump Analyzer) is intended to perform automated analysis of dump files produced by the IBM Java VM. Starting with the name of the dump to be analyzed the analysis will attempt to localise the problem and if successful will produce a diagnosis of the error together with sufficient information to fix it or suggestions on how to continue the analysis using other tools (e.g. MDD4J to diagnose out of memory situations). If localization fails then the tool will default to producing summary information from the dump intended to aid further diagnosis.

Background

The Java language has come to be predominant in software development, and thus the reliability of the Java Virtual Machine (VM) has become a very important issue. The VM is typically a reliable piece of software, but of course failures do occur during execution for a variety of reasons. A small number of these problems are due to errors in the VM itself; however, in the majority of cases, they are due to errors or misconfigurations in the software stack above the VM (in WebSphere™ Application Server, for instance) or in the application itself.

The software stack for a typical projects has increased in complexity as information technology has matured, which has led to increasing difficulties for developers trying to determination the causes of problems. In such a complex environment, you may be faced with an overwhelming excess of information with which to diagnose the fault. In a production environment there may well be many gigabytes of heap, hundreds of threads, thousands of classloaders, tens of thousands of classes, and a huge number of objects.

The Dump Analyzer is an extensible framework that seeks to solve this problem for the IBM Java SDK's. It uses analyzers to interrogate a formatted system dump (each analyzer asking the dump a specific question) and links them together with a script to produce a concise report of the analysis. Typical problems analyzed might be the following.

* Out of memory

* Deadlock detected

* VM terminated due to signal (internal or middleware/java application error)

* Further investigation is required

The aim is to diagnose the correct category of problem and either provide a complete diagnosis of the problem or give some information designed to aid diagnosis together with a recommended next course of action.

E) Have insight to IBM Support Assistant.

http://www-01.ibm.com/support/docview.wss?rs=3455&uid=swg27012682

IBM Support Assistant agents #Agents are based on the Tivoli Common Agent runtime

#Provide services such as data collection, software inventory, file transfer #Install the agent on any system you want to manage

#Agents can be customized with product-specific content. The service agents that are now available with IBM Support Assistant are based on the Tivoli Common Agent runtime. They work in conjunction with the agent manager to provide remote debugging capability to your IBM Support Assistant Workbench. The agent needs to be installed on any system that you want manage. The basic agent provides services such as data collection, file transfer, and the ability to generate a software inventory for the system, but the agents can also be customized with product-specific content. For example, you can install agent add-ons for WebSphere Application Server V7 that allow you to remotely connect to a system, trigger a dump for the Java Virtual Machine in your WebSphere environment, and automatically transfer the dump back to your workstation.

Sample Installation Flow:

- Install the workbench on your desktop
- Install the agent manager – start it
- Install the common agent – start it
- Register the agent with the agent manager, can be done during installation process
- Register the workbench with the agent manager
- Workbench can communicate with the agent manager and agents

Recall that the agent manager is a lightweight server component that is used to coordinate communication between the workbench and service agents, so you will need to configure the ports that the agent manager will use to communicate with the other components. The default registration port is 9511, and the default public port is 9513. You will need to use these port values when registering the agent and the workbench with the agent manager, so be sure to make a note of the port values that you choose if you change them from the defaults.

https://<host name>:9511/AgentMgr/Info

Using IBM Support Assistant

IBM® Support Assistant is a free troubleshooting application that helps you research, analyze, and resolve problems using various support features and tools. IBM Support Assistant enables you to find solutions yourself using the same troubleshooting techniques used by the IBM Support team, and it allows you to organize and transfer your troubleshooting efforts between members of your team or to IBM for further support.

About this task

New feature: IBM Support Assistant V4.0 is released with a host of new features and enhancements, making this version the most comprehensive and flexible yet. Our one-stop-shop solution to research, analyze and resolve software issues is now better than ever before, and you can still download it at no charge.

IBM Support Assistant version 4.0 enhancements include:

* Remote System Troubleshooting: Explore file systems, run automated data collectors and troubleshooting tools, and view the system inventory on remote systems.

* Activity-based Workflow: Choose from support-related activities, or use the Guided Troubleshooter for step-by-step help with analysis and resolution.

* Case Management: Organize your troubleshooting data in "cases"; then export and share these cases with other problem analysts or with IBM Support.

* Improved Flexibility: Add your own search locations, control updates by hosting your own update site, get the latest product news and updates.

newfeat

The IBM Support Assistant V4.0 consists of the following three distinct entities:

IBM Support Assistant Workbench

The IBM Support Assistant Workbench, or simply the Workbench, is the client-facing application that you can download and install on your workstation. It enables you to use all the troubleshooting features of the Support Assistant such as Search, Product Information, Data Collection, Managing Service Requests, and Guided Troubleshooting. However, the Workbench can only perform these functions locally, for example, on the system where it is installed (with the exception of the Portable Collector).

If you need to use the IBM Support Assistant features on remote systems, additionally install the Agent Manager and Agent. However, if your problem determination needs are purely on the local system, the Agent and Agent Manager are not required.

The Workbench has a separate download and this is all that is required to get started with the Support Assistant.

IBM Support Assistant Agent

The IBM Support Assistant Agent, or simply the Agent, is the piece of software that needs to be installed on EVERY system that you need to troubleshoot remotely. Once an Agent is installed on a system, it registers with the Agent Manager and you can use the Workbench to communicate with the Agent and use features such as remote system file transfer, data collections and inventory report generation on the remote machine.

IBM Support Assistant Agent Manager

The IBM Support Assistant Agent Manager, or simply the Agent Manager, needs to be installed only ONCE in your network. The Agent Manager provides a central location where information on all available Agents is stored and acts as the certificate authority. For the remote troubleshooting to work, all Agent and Workbench instances register with this Agent Manager.

Any time a Support Assistant Workbench needs to perform remote functions, it authenticates with the Agent Manager and gets a list of the available Agents. After this, the Workbench can communicate directly with the Agents.

The Agent and Agent Manager can be downloaded in a combined installer, separate from the Workbench.

IBM Support Assistant Version 4 has the following functions:

Search interface and access to the latest product information

IBM Support Assistant allows you to search multiple knowledge repositories with one click and gives you quick access to the latest product information so that you spend less time looking for the solution and more time building skills and solving problems.

Troubleshooting tools

Whether you are new to an IBM product or an advanced user, IBM Support Assistant can help. You can choose to be guided through your problem symptoms or view a complete listing of advanced tooling for analyzing everything from logs to memory dumps.

Access to local and remote systems

Using the IBM Support Assistant Workbench installed on a local workstation running the Windows® or Linux® Intel® operating system, you can connect to the IBM Support Assistant Agent installed on a remote system running on the AIX®, Linux, Windows, or Solaris operating system through the IBM Support Assistant Agent Manager on the Workbench. This function enables you to explore, transfer data, and run diagnostic tooling not only on your system but on any other system where the IBM Support Assistant Agent is installed.

Automated data gathering and efficient support

Instead of manually gathering information, you can use IBM Support Assistant to run automated, symptom-specific data collectors. This data can then be attached to an IBM Service Request so that you can get support from the experts at IBM Support.

Procedure

* Follow the installation instructions on IBM Support Assistant (ISA) Web site at: IBM Support Assistant (ISA).

* Read the “First Steps" section of the documentation for IBM Support Assistant to run the customization wizard, or migrate from a previous version of IBM Support Assistant. Read the "Tutorials" section to learn more about the capabilities of ISA.

Diagnosing problems using IBM Support Assistant tooling

The IBM® Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products.

About this task

Tools for IBM Support Assistant perform numerous functions from memory-heap dump analysis and Java™ core-dump analysis to enabling remote assistance from IBM Support. All of these tools come with help and usage documentation that allow you to learn about the tools and start using them to analyze and resolve your problems.

The following are samples of the tools available in IBM Support Assistant:

Memory Dump Diagnostic for Java (MDD4J)

The Memory Dump Diagnostic for Java tool analyzes common formats of memory dumps (heap dumps) from the Java virtual machine (JVM) that is running the WebSphere® Application Server or any other standalone Java applications. The analysis of memory dumps is targeted towards identifying data structures within the Java heap that might be root causes of memory leaks. The analysis also identifies major contributors to the Java heap footprint of the application and their ownership relationship. The tool is capable of analyzing very large memory dumps obtained from production-environment application servers encountering OutOfMemoryError issues.

IBM Thread and Monitor Dump Analyzer (TMDA)

IBM Thread and Monitor Dump Analyzer (TMDA) provides analysis for Java thread dumps or javacores such as those from WebSphere Application Server. You can analyze thread usage at several different levels, starting with a high-level graphical view and drilling down to a detailed tally of individual threads. If any deadlocks exist in the thread dump, TMDA detects and reports them.

Log Analyzer

Log Analyzer is a graphical user interface that provides a single point of contact for browsing, analyzing, and correlating logs produced by multiple products. In addition to importing log files from multiple products, Log Analyzer enables you to import and select symptom catalogs against which log files can be analyzed and correlated.

IBM Visual Configuration Explorer

The IBM Visual Configuration Explorer provides a way for you to visualize, explore, and analyze configuration information from diverse sources.

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT)

The IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) parses IBM verbose garbage-collection (GC) trace, analyzes Java heap usage, and recommends key configurations based on pattern modeling of Java heap usage. Only verbose GC traces that are generated from IBM Java Development Kits (JDKs) are supported.

IBM Assist On-site

IBM Assist On-site provides remote desktop capabilities. You run this tool when you are instructed to do so by IBM Support personnel. With this live remote-assistance tool, a member of the IBM Support team can view your desktop and share control of your mouse and keyboard to help you find a solution. The tool can speed up problem determination, data collection, and ultimately your problem solution.

Procedure

To install tools for the IBM Support Assistant Workbench on a Windows® or Linux® Intel® operating system, go to the Update menu and select Tool Add-ons. A list of all available tools appears, and you can select the tools that you would like to install.

You can install, update, or remove tools from the IBM Support Assistant Workbench at any time.

Resources

The information contained within this document is not the original work of the author of this document.

The information contained within is a collection from the following IBM sources:

http://www-03.ibm.com/certify/tests/sam377.shtml

http://www-03.ibm.com/certify/tests/edu377.shtml

http://www-03.ibm.com/certify/tests/obj377.shtml

http://www.redbooks.ibm.com/redbooks/pdfs/sg246688.pdf

IBM Infocenter - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp

IBM Education Assistant -
http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp

IBM Redbooks

WebSphere Application Server V7 Messaging Administration Guide

ISBN - 0738433055

IBM Form Number - SG24-7770-00

IBM WebSphere Application Server V7 Administration and Configuration Guide

ISBN - 0738433047

IBM Form Number - SG24-7615-00