WebSphere software

IBM

Redpaper

Arden Agopyan
Hermann Huebler
Tze Puah
Thomas Schulze
David Soler
Martin Keen

WebSphere Application Server V7.0: Technical Overview

WebSphere® Application Server is the implementation by IBM® of the Java™ Platform, Enterprise Edition (Java EE) platform. It conforms to the Java EE 5 specification. WebSphere Application Server is available in unique packages that are designed to meet a wide range of customer requirements. At the heart of each package is a WebSphere Application Server that provides the runtime environment for enterprise applications.

This discussion centers on the runtime server component of WebSphere Application Server.

For more information about topics discussed in this paper, refer to the IBM Redbooks® publication *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708.

WebSphere Application Server packaging

WebSphere Application Server comes in several packaging options. In addition to the application server component, each package contains an appropriate combination of complementary products (for example, IBM HTTP Server, Rational® Application Developer Assembly and Deploy, Edge components, and so on).

Distributed platforms

WebSphere Application Server V7.0 has the following packaging options for distributed platforms, including IBM AIX®, HP-UX, Linux®, Solaris™, and Microsoft® Windows®:

- IBM WebSphere Application Server - Express V7.0, referred to as *Express*

  For more information about Express, see the following Web page:

  `http://www.ibm.com/software/webservers/appserv/express/`

- IBM WebSphere Application Server V7.0, referred to as *Base*

  For more information about Base, see the following Web page:

  `http://www.ibm.com/software/webservers/appserv/was/`

- IBM WebSphere Application Server Network Deployment V7.0, referred to as *Network Deployment*

  For more information about Network Deployment, see the following Web page:

  `http://www.ibm.com/software/webservers/appserv/was/network/`

System z

For WebSphere Application Server on System z®, IBM WebSphere Application Server for z/OS® V7.0, a full-function version of the Network Deployment product is available.

For more information about WebSphere Application Server for z/OS V7.0, see the following Web page:

`http://www.ibm.com/software/webservers/appserv/zos_os390/`

System i

WebSphere Application Server on System i® has the following packaging options:

- IBM WebSphere Application Server V7.0 for IBM i
- IBM WebSphere Application Server for Developers V7.0 for IBM i
- IBM WebSphere Application Server Network Deployment V7.0 for IBM i
- IBM WebSphere Application Server - Express V7.0 for IBM i

For more information about WebSphere Application Server on System i, see the following Web page:

http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/

Application support

WebSphere Application Server V7.0 can run the following types of applications:

- Java EE applications
- Portlet applications
- Session Initiation Protocol (SIP) applications
- Business level applications

Java EE applications

The Java EE specification is the standard for developing, deploying, and running enterprise applications. WebSphere Application Server V7.0 provides full support for the Java EE 5 specification. The Java EE programming model has multiple types of application components:

- Enterprise beans
- Servlets and JavaServer™ Pages files
- Application clients

The primary development tool for WebSphere Application Server Java EE 5 applications is Rational Application Developer for WebSphere V7.5.

The Rational Application Developer Assembly & Deploy V7.5, shipped with WebSphere Application Server, contains the tools needed to create, test, and deploy Java EE 5 applications. It includes full support for the new features of Java SE 6.0. Applications are packaged as enterprise application archives (EAR files).

For information about the Java EE specification, see the following Web page:

http://java.sun.com

Portlet applications

The Portlet container in WebSphere Application Server V7.0 provides the runtime environment for JSR 268 compliant portlets.

Portlet applications are intended to be combined with other portlets collectively to create a single page of output. The primary development tool for portlets on WebSphere Application Server portlet applications is Rational Application Developer for WebSphere V7.5. You can also use Rational Application Developer Assembly & Deploy V7.5, which is shipped with WebSphere Application Server.

Portlets are packaged in WAR files. The portlet runtime does not provide the advanced capabilities of WebSphere Portal, such as portlet aggregation and page layout, personalization and member services, or collaboration features.

For more information about JSR 286, see the following Web page:

http://jcp.org/en/jsr/detail?id=286

Session Initiation Protocol applications

Session Initiation Protocol (SIP) applications are Java programs that use at least one SIP servlet written to the JSR 116 specification. SIP is used to establish, modify, and terminate multimedia IP sessions. SIP negotiates the medium, the transport, and the encoding for the call. After the SIP call has been established, the communication takes place over the specified transport mechanism, independent of SIP. Examples of application types that use SIP include voice over IP, click-to-call, and instant messaging.

Rational Application Developer Assembly & Deploy V7.5 provides special tools for developing SIP applications. SIP applications are packaged as SIP archive (SAR) files, and are deployed to the application server using the standard WebSphere Application Server administrative tools. SAR files can also be bundled in a Java EE application archive (EAR file), similar to other Java EE components.

For more information about SIP applications, see the following Web pages:

- JSR 116 SIP Servlet API 1.0 Specification

  http://www.jcp.org/aboutJava/communityprocess/final/jsr116/
- JSR 116

  http://jcp.org/en/jsr/detail?id=116
- RFT 3261

  http://www.ietf.org/rfc/rfc3261.txt

Business level applications

A *Business level application* is a notion of an application beyond Java EE's definition. This is a new administration concept that expands the options previously offered by Java EE. This grouping notion for enterprise-level applications includes WebSphere and non-WebSphere artifacts like Service Component Architecture (SCA) packages, libraries, and proxy filters under a single application definition (Figure 1). Every artifact in the group is a *composition unit*.

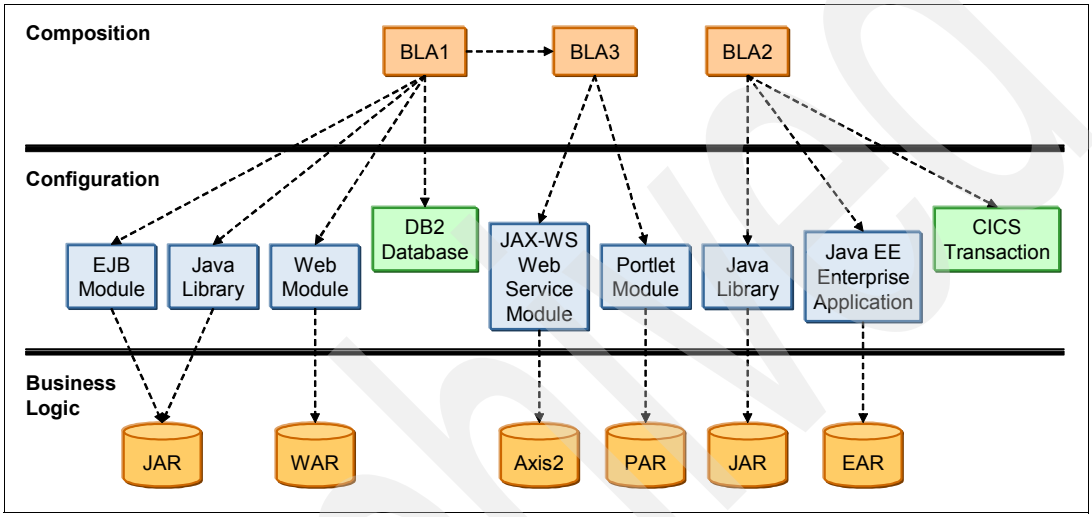

Figure 1 Business level applications

A business level application can be useful when an applicationhas the following characteristics:

- Is composed of multiple packages
- Applies to the post-deployment side of the application life cycle
- Contains additional libraries, or non-Java EE artifacts
- Includes artifacts that run on heterogeneous environments that include WebSphere and non-WebSphere runtimes
- Is defined in a recursive manner (for example, if an application includes other applications)

Application server configurations

At the heart of each member of the WebSphere Application Server family is an application server. Each family has essentially the same architectural structure. Although the application server structure for the Base and Express platforms is identical, there are differences in licensing terms and platform support.

With the Base and Express platforms, you are limited to stand-alone application servers. The Network Deployment platform enables more advanced topologies that provide workload management, scalability, high availability, and central management of multiple application servers. You can also manage multiple Base profiles centrally, but you will not have workload management, scalability, and high availability capabilities.

Runtime environments are built by creating profiles. A profile can define a deployment manager, a stand-alone application server, or an empty node to be federated (added) to a cell. Each profile contains files specific to that runtime (such as logs and configuration files). Profiles can be created during and after installation. After the profiles have been created, further configuration and administration is performed using the WebSphere administrative tools.

Stand-alone application servers

All WebSphere Application Server packages support a single stand-alone server environment. With this configuration, each application server acts as an unique entity. An application server runs one or more applications and provides the services required to run those applications. Each stand-alone server is created by defining an application server profile. See Figure 2.

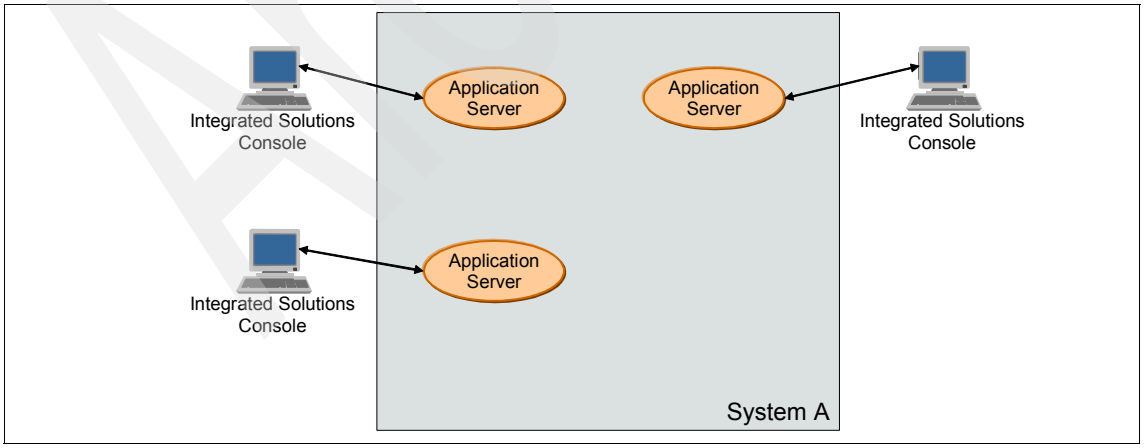

Figure 2 Stand-alone application server configuration

A stand-alone server can be managed from its own administrative console and functions independent from all other application servers. You can also use WebSphere Application Server's scripting facility, **wsadmin**, to perform every function that is available in the administrative console application.

Multiple stand-alone application servers can exist on a machine, either through independent installations of the WebSphere Application Server product binaries, or by creating multiple application server profiles within one installation. Stand-alone application servers do not provide workload management or failover capabilities. They run isolated from each other.

With WebSphere Application Server for z/OS, it is possible to use workload load balancing and response time goals even on a transactional base, as well as a special clustering mechanism, the *multi-servant region*, with a stand-alone application server.

Distributed application servers

With the Network Deployment packaging, you can build a *distributed server* configuration to enable central administration, workload management, and failover. In this environment, you integrate one or more application servers into a cell that is managed by a central administration instance, a deployment manager. The application servers can reside on the same machine as the deployment manager or on multiple separate machines. Administration and management is handled centrally from the administration interfaces by the deployment manager. An example is shown in Figure 3.

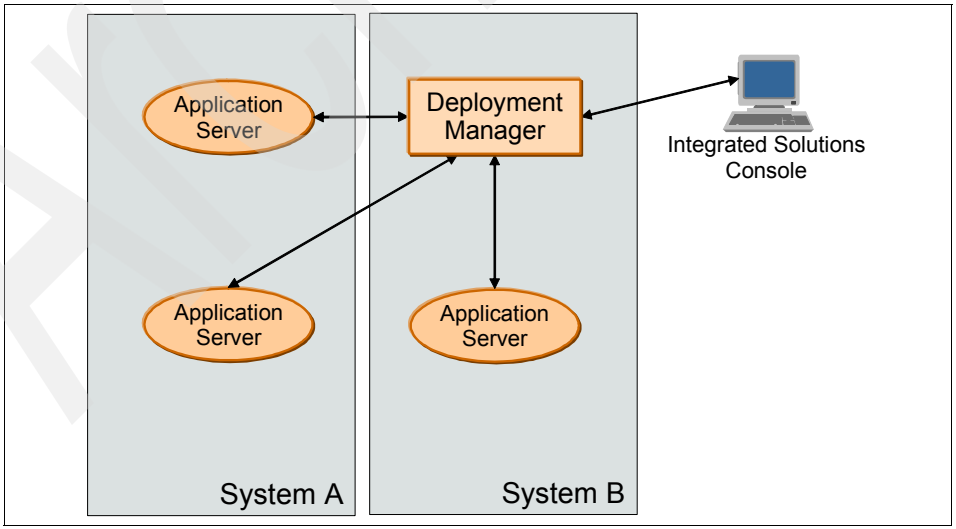

Figure 3 Distributed application servers with WebSphere Application Server V7.0

With a distributed server configuration, you can create multiple application servers to run unique sets of applications and manage those applications from a central location. More importantly, you can cluster application servers to allow for workload management and failover capabilities. Applications that are installed in the cluster are replicated across the application servers. When one server fails, another server in the cluster continues processing. Workload is distributed among Web and EJB™ containers in a cluster using a weighted round-robin scheme, or randomly, depending upon your configuration.

In z/OS the weighted round-robin mechanism is replaced by the integration of WebSphere Application Server for z/OS in the Workload Manager (WLM), that is an integral part of the operating system. This allows requests to be dispatched to a cluster member according to real-time load and whether or not the member reaches its defined response time goals. It is also possible to replicate sessions saved in an application server of the cluster to other cluster members with the session replication feature of WebSphere Application Server.

A distributed server configuration can be created in one of three ways:

- Create a *deployment manager profile* to define the deployment manager. Then, create one or more custom node profiles. The nodes defined by each custom profile can be federated into the cell managed by the deployment manager during profile creation or later, manually. The custom nodes can exist inside the same operating system image as the deployment manager or in another operating system instance. Application servers can then be created using the Integrated Solutions Console or **`wsadmin`** scripts.
- Create a deployment manager profile to define the deployment manager. Then, create one or more application server profiles and federate these profiles into the cell managed by the deployment manager. This process adds both nodes and application servers into the cell. The application server profiles can exist on the deployment manager system or on multiple separate system or z/OS image.
- Create a *cell profile*. This actually creates two profiles: a deployment manager profile and a federated application server profile. Both reside on the same machine.

Application servers concepts

Regardless of the configuration, WebSphere Application Server is organized based on the concept of cells, nodes, and servers. Although all of these elements are present in each configuration, cells and nodes do not play an important role until you take advantage of the features provided with Network Deployment. These cells, nodes, and servers can be managed by a combination of deployment managers, administrative agents, and job managers.

Application servers

The application server is the primary runtime component in all configurations and is where an application actually executes. All WebSphere Application Server configurations can have one or more application servers. In the Express and Base configurations, each application server functions as a separate entity. There is no workload distribution or central administration among application servers. With Network Deployment, you can build a distributed server environment consisting of multiple application servers maintained from a central administration point. In a distributed server environment, you can cluster application servers for workload distribution and failover.

Nodes, node groups, and node agents

This section defines node related concepts.

Nodes

A *node* is an administrative grouping of application servers for configuration and operational management within one operating system instance (virtualization allows multiple operating systems on one machine). It is possible to create multiple nodes inside one operating system instance, but a node cannot leave the operating system boundaries. In a stand-alone application server configuration, there is only one node. With Network Deployment, you can configure a distributed server environment consisting of multiple nodes that are managed from one central administration server. See Figure 4 on page 10.

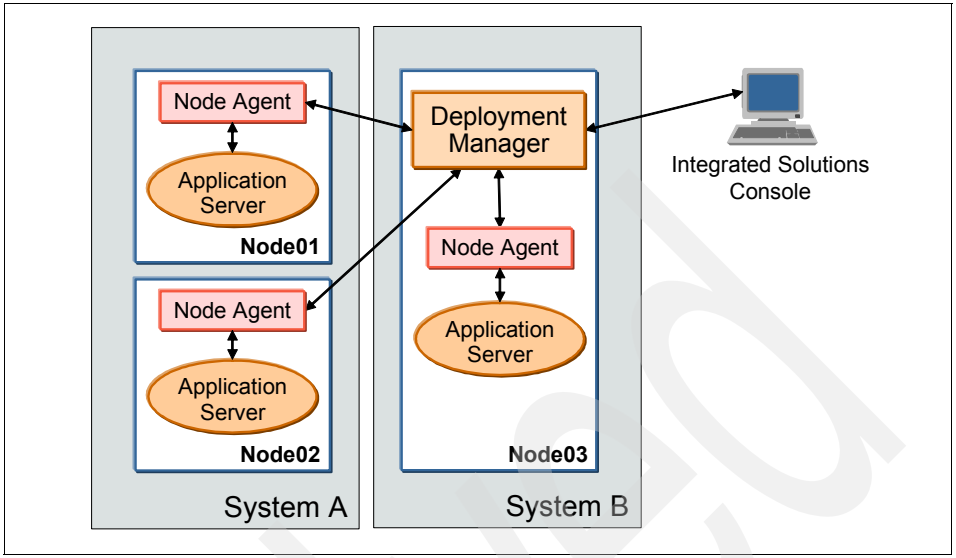

Figure 4 Node concept in a WebSphere Application Server network deployment configuration

Node agents

In distributed server configurations, each node has a *node agent* that works with the deployment manager to manage administration processes. See Figure 4. A node agent is created automatically when you add (federate) a stand-alone node to a cell. It is not included in Base and Express configurations.

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.

On the z/OS platform, a node must be a member of a *system complex* (sysplex) node group. Nodes in the same sysplex must be in the same sysplex node group. A node can be in one sysplex node group only.

Cells

A *cell* is a grouping of nodes into a single administrative domain. In the Base and Express configurations, a cell contains one node. That node contains one server. See Figure 5.

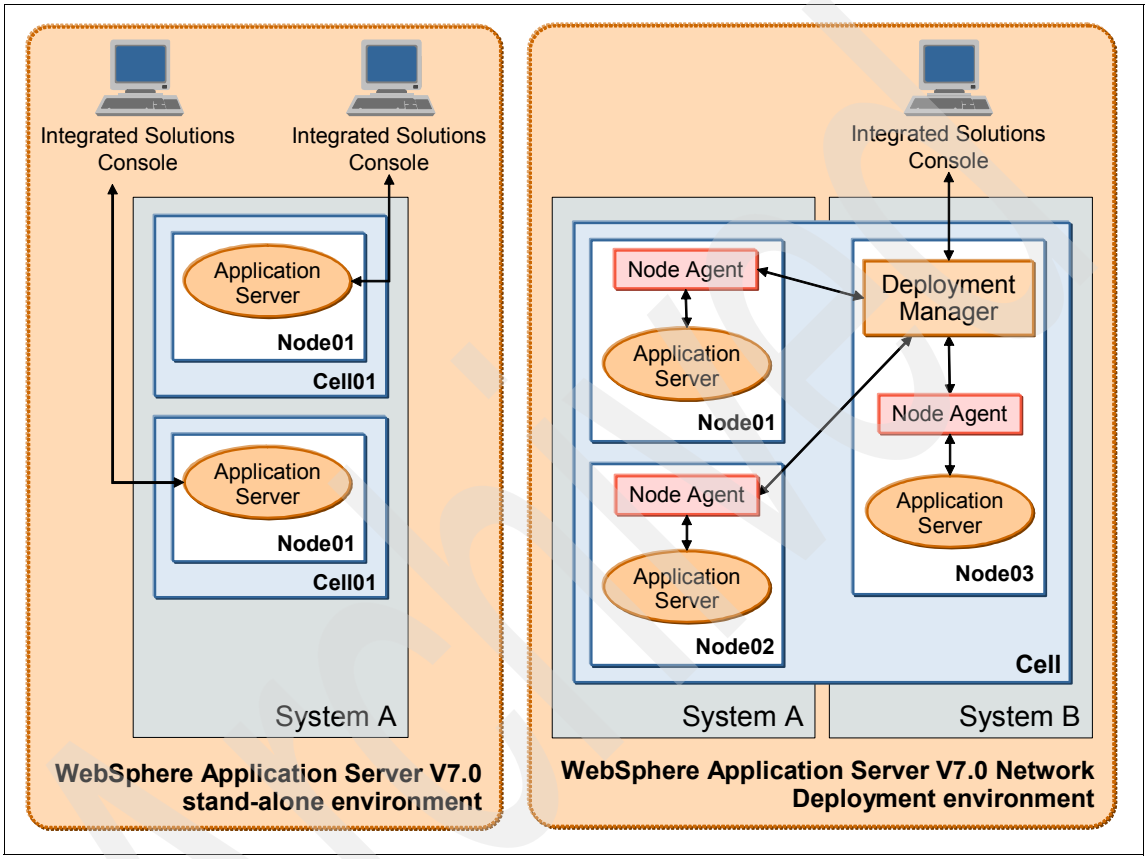

Figure 5 Cell component in a WebSphere Application Server topology

In a Network Deployment environment, a cell can consist of multiple nodes (and node groups), which are all administered from a single point, the deployment manager. See Figure 5. If your cell configuration contains nodes running on the same platform, it is called a *homogeneous cell*. It is also possible to have a cell made up of nodes on mixed platforms. This is referred to as a *heterogeneous cell*.

Deployment managers

The *deployment manager* is the central administration point of a cell, which 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. See Figure 6.

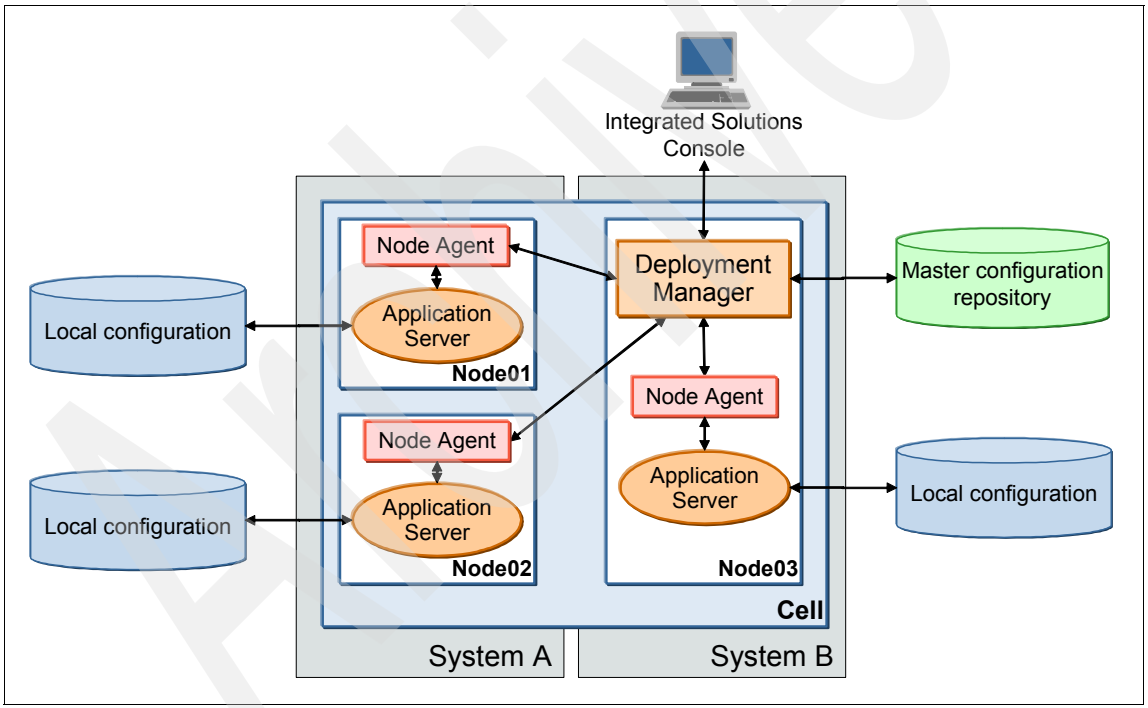

Figure 6 Configuration repositories in a network deployment installation

Administrative agents

An *administrative agent* is a component that provides enhanced management capabilities for stand-alone (Express and Base) application servers. This is a new concept introduced with WebSphere Application Server V7.0.

In previous versions of WebSphere Application Server, each stand-alone server was a single point of management containing its own administrative console. With V7.0, most of the administrative components are separated from the application server runtime (see Figure 7).

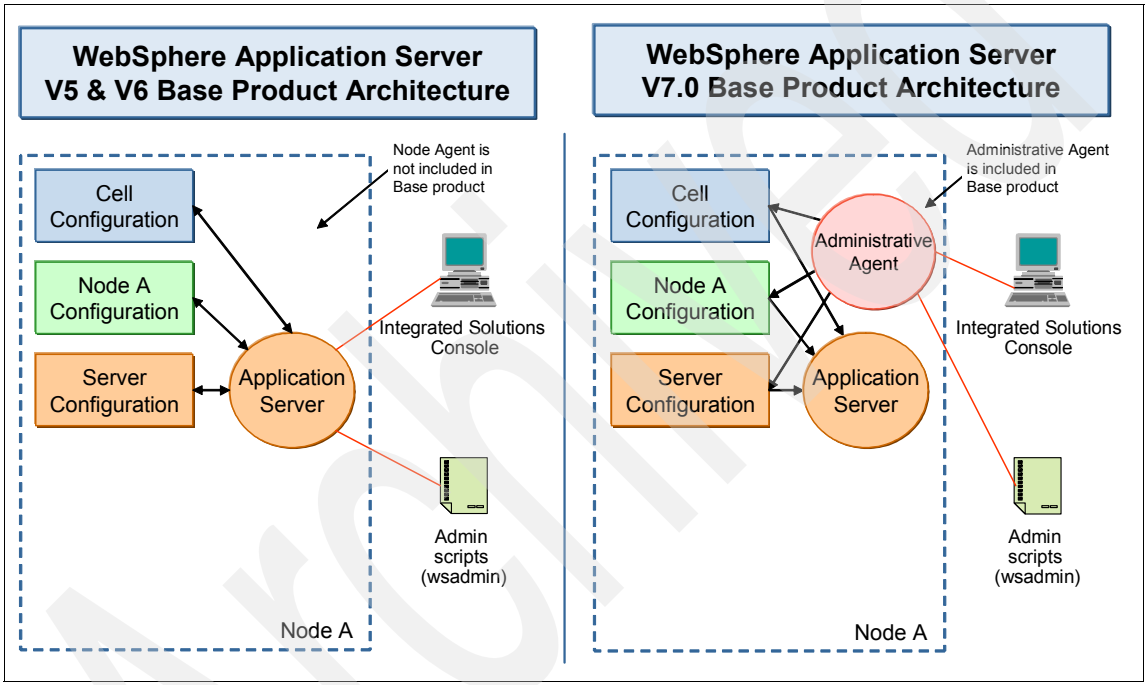

Figure 7 Comparison of V5, V6 and V7.0 base product architectures

All configurations related to the application server are directly connected to the administrative agent that provides services to administrative tools.

An administrative agent can manage multiple stand-alone server instances on a single system or z/OS image. Therefore, when using an administrative agent, as the number of application server instances increases, the redundancy of the administration footprint (for each application server) is eliminated.

Job managers

A *job manager* is a component that provides management capabilities for multiple stand-alone application servers, administrative agents, and deployment managers (see Figure 8). It brings enhanced multiple node installation options for your environment.

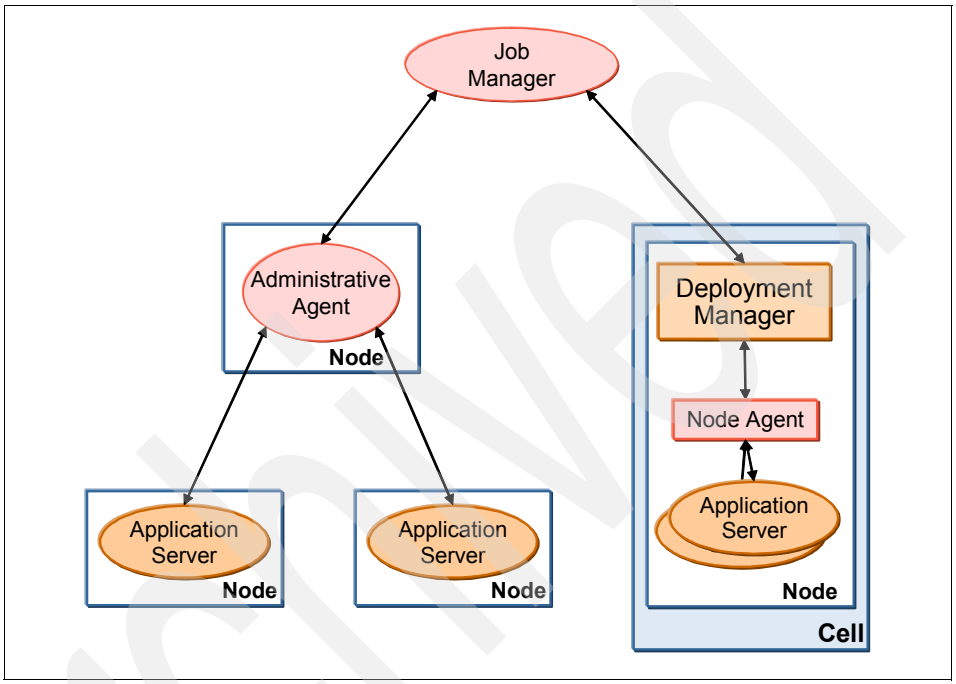

Figure 8 High-level overview of a job manager architecture

The purpose of the job manager is to execute daily tasks in one step for multiple installations. This includes the starting and stopping of servers, distribution and deployment of applications, and various other actions.

The job manager is a new concept introduced with WebSphere Application Server V7.0 and is only available with WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS.

Servers

WebSphere Application Server supplies application servers, which provide the functions that are required to host applications and proxy servers that distribute work to the application servers. It also provides the ability to define external servers to the administration process, and to associate Web servers for routing purposes.

Application servers

Application servers provide the runtime environment for application code. They provide containers and services that specialize in enabling the execution of specific Java application components. Each application server runs in its own Java Virtual Machine (JVM™).

WebSphere Application Server V7.0 has a new JVM designed to improve stability and performance. It provides a Java language compiler and execution environment to support the Java Platform, Standard Edition (Java SE) 6 specification. This new JVM is supported on all platforms that ship with an IBM JDK™.

The most important enhancements to Java SE 6 related to performance are as follows:

- Improved startup performance
- Smaller memory footprint
- Improved 64-bit performance
- Higher garbage collection throughput

Application server clusters

An *application server cluster* is a logical collection of application server processes that provides workload balancing and high availability. It is a grouping of application servers that run an identical set of applications managed so that they behave as a single application server (parallel processing). WebSphere Application Server Network Deployment or WebSphere Application Server for z/OS is required for clustering.

Application servers that are a part of a cluster are called *cluster members*. When you install, update, or delete an application, the updates are automatically distributed to all members in the cluster. A rollout update option enables you to update and restart the application servers on each node, one node at a time, providing continuous availability of the application to the user.

Proxy servers

A *proxy server* is a specific type of application server that routes requests to content servers that perform the work. The proxy server is the initial point of entry, after the protocol firewall, for requests entering the environment.

WebSphere Application Server allows you to create two types of proxy servers:

- WebSphere Application Server Proxy

  This proxy server is used to classify, prioritize, and route HTTP and SIP requests to servers in the enterprise, as well as cache content from servers.

- DMZ Secure Proxy Server

  This proxy server comes in a separate install package and provides security enhancements to allow deployments inside of a demilitarized zone

Web servers

Although independent products, Web servers can be defined to the WebSphere Application Server administration process. The primary purpose for this is to enable the administrator to associate applications with one or more defined Web servers in order to generate the proper routing information for Web server plug-ins if multiple servers are used.

Web servers are associated with nodes. These nodes can be *managed* or *unmanaged*.

- Managed nodes have a node agent on the Web server machine that allows the deployment manager to administer the Web server. You can start or stop the Web server from the deployment manager, generate the Web server plug-in for the node, and automatically push it to the Web server. In most installations, you have managed Web server nodes behind the firewall with the WebSphere Application Server installations.
- Unmanaged nodes are not managed by WebSphere. You usually find these outside the firewall or in the demilitarized zone. You have to manually transfer the Web server plug-in configuration file to the Web server on an unmanaged node. In a z/OS environment, you have to use unmanaged nodes if the Web server is a not running on the z/OS platform.

Note: As a special case, if the unmanaged Web server is an IBM HTTP Server, you can administer it from the Integrated Solutions Console. This allows you to automatically push the plug-in configuration to the Web server with the deployment manager. To perform this, HTTP commands are sent to the IBM HTTP Server administration process. This configuration does not require a node agent.

The IBM HTTP Server ships with all WebSphere Application Server packages.

Web server plug-ins

A Web server can be used to serve static contents and requests, like HTML pages. When a request requires dynamic content, such as JSP™ or servlet processing, it must be forwarded to WebSphere Application Server for handling.

To forward a request, use a Web server plug-in that is included with the WebSphere Application Server packages for installation on a Web server. You transfer (manually or automatically with the deployment manager) an Extensible Markup Language (XML) configuration file (configured on the WebSphere Application Server) to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When WebSphere Application Server receives a request for an application server, it forwards the request to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPS to transmit the request. See Figure 9 on page 18. The plug-in is also used for routing requests to one of multiple application servers.

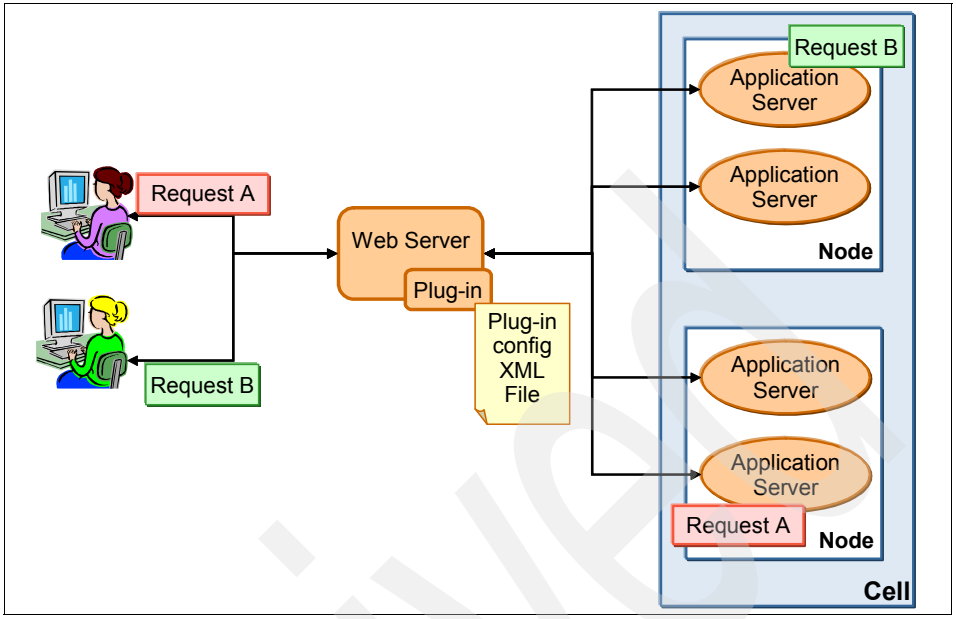

Figure 9 Web server plug-in concept with WebSphere Application Server

Generic servers

A *generic server* is a server that is managed in the WebSphere Application Server administrative domain even though the server is not supplied by WebSphere Application Server. The WebSphere Application Server generic servers function enables you to define a generic server as an application server instance within the WebSphere Application Server administration, and associate it with a WebSphere Application Server or process.

There are two basic types of generic application servers:

- Non-Java applications or processes
- Java applications or processes

A generic server can be any server or process that is necessary to support the application server environment:

- Java server
- C or C++ server or process
- CORBA server
- Remote Method Invocation (RMI) server

Containers

Containers provide runtime support for applications. WebSphere Application Server V7.0 has the following container support.

Application server containers

Each application server provides the following container support:

- Web container

  The Web container processes servlets, JSPs (processed as servlets), and other types of server-side includes. Each application server runtime has one logical Web container, which can be modified but not created or removed. Runtime provisioning can be used to avoid starting the container if it is not needed.

  Requests are received by the Web container through the Web container inbound transport chain. The chain consists of a TCP inbound channel that provides the connection to the network, an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests, and a Web container channel over which requests for servlets and JSPs are sent to the Web container for processing. Requests for HTML and other static content directed to the Web container are served by the Web container inbound chain.

  It is suggested that you use an external Web server to receive client requests and a Web server plug-in to forward requests for servlets to the Web container.

- Enterprise JavaBeans™ (EJB) container

  The EJB container provides all of the runtime services that are needed to deploy and manage enterprise beans. It is a server process that handles requests for both session and entity beans.

  The container provides many low-level services, including transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained enterprise beans. A single container can host more than one EJB Java archive (JAR) file.

- Portlet container

  The portlet container processes JSR 286 compliant portlets. The portlet container is an extension to the Web container.

- Session Initiation Protocol container

  The SIP container processes applications that use at least one SIP servlet written to the JSR 116 specification.

Intelligent runtime provisioning

Intelligent runtime provisioning is a concept introduced with WebSphere Application Server V7.0. This mechanism selects only the runtime functions needed for an application. See Figure 10. Each application is examined by WebSphere Application Server during the deployment to generate an *activation plan*. At run time, the server uses the activation plan to start only those components that are required inside the application server.

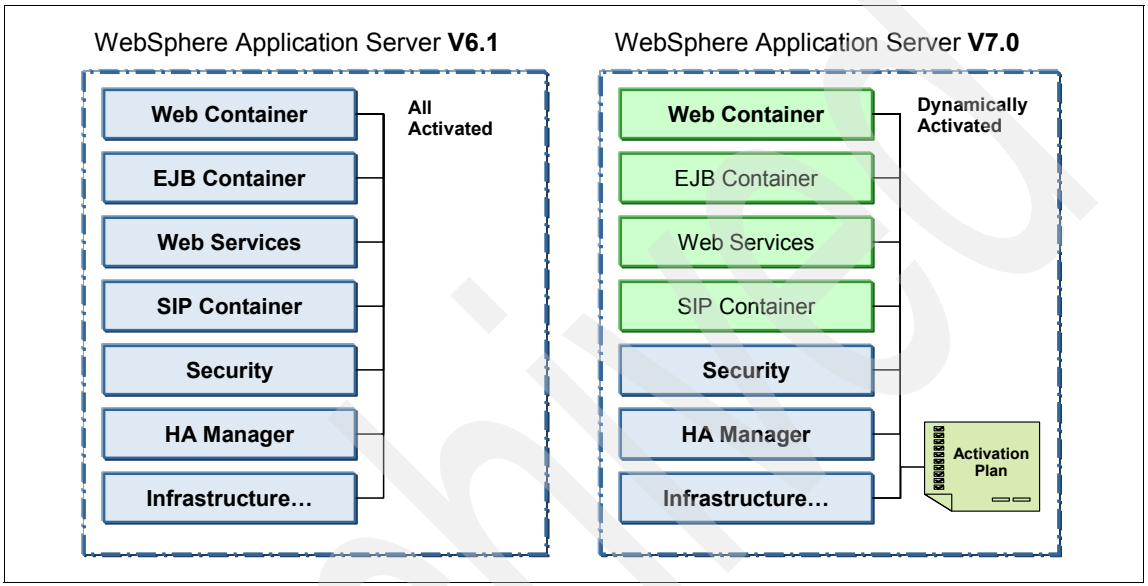

Figure 10 Intelligent runtime provisioning

Intelligent runtime provisioning is a feature that reduces memory footprint, application server startup time, and used CPU resources needed to start the application server.

Application client container

The application client container provides the run time for Java EE client applications to run with or without using services provided by the Java EE Client Container. The application client container is a Java application program that accessess enterprise beans, Java DataBase Connectivity (JDBC™) APIs, and Java Message Service message queues.

Workload management

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS, as shown in Figure 11.

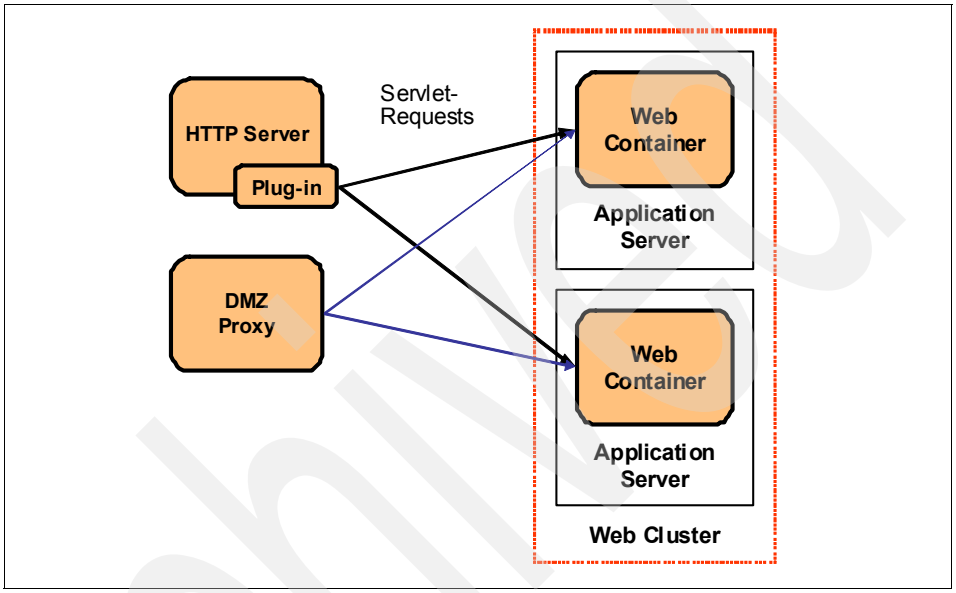

Figure 11 Plug-in (Web container) workload management

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 no strong affinity configurations. If the weights are scaled in a range from 0–20, the plug-in routes requests to those cluster members with the higher weight value more often. No requests are sent to cluster members with a weight of 0 unless no other servers are available.

A guideline formula for determining routing preference is as follows:

```
% routed to Server1 = weight1 / (weight1+weight2+...+weightn)
```

Where there are *n* cluster members in the cluster.

Workload management for EJB containers take place when the EJB client application and the EJB itself run in different JVMs. Multiple application servers with the EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers, shown in Figure 12 on page 22.

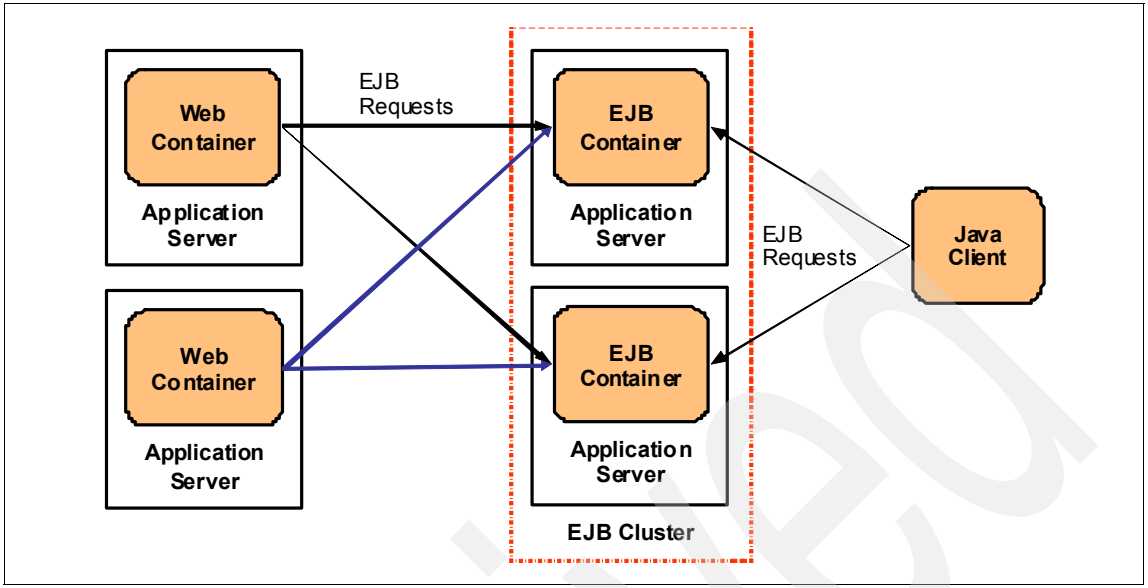

Figure 12 EJB workload management

In this configuration, EJB client requests are routed to available EJB servers 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 distribution based on the set of server weights that have been assigned to the members of a cluster. Transaction and process affinity are considered in this policy. 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 also choose that EJB requests are preferably routed to the same host as the host of the requesting EJB client. In this case, only cluster members on that host are chosen (using the round-robin weight method). Cluster members on remote host are chosen only if a local server is not available.

High availability

WebSphere Application Server uses a high availability manager to eliminate single points of failure. The high availability manager is responsible for running key services on available application servers rather than on a dedicated one (such as the deployment manager). It continually polls all of the core group members to verify that they are active and healthy.

For certain functions (like transaction peer recovery) the high availability manager takes advantage of fault tolerant storage technologies such as Network Attached Storage (NAS), which significantly lowers the cost and complexity of high availability configurations. The high availability manager also provides peer-to-peer failover for critical services by maintaining a backup for these services. WebSphere Application Server also supports other high availability solutions such as HACMP™, Parallel Sysplex®, and so on.

A high availability manager continually monitors the application server environment. If an application server component fails, the high availability manager takes over the in-flight and in-doubt work for the failed server. This introduces some overhead, but significantly improves application server availability.

A high availability manager focuses on recovery support and scalability in the following areas:

- Embedded messaging
- Transaction managers
- Workload management controllers
- Application servers
- WebSphere partitioning facility instances
- On-demand routing
- Memory-to-memory replication thorugh Data Replication Service (DRS)
- Resource adapter management

Administration

WebSphere Application Server V7.0 provides a variety of administrative tools for configuring and managing your runtime environment:

- Integrated Solutions Console

  The Integrated Solutions Console is a browser-based client that uses a Web application running in the Web container to administer WebSphere Application Server.

- WebSphere scripting client (**wsadmin**)

  The **wsadmin** client is a non-graphical scripting interface that can be used to administer WebSphere Application Server from a command line prompt. It can connect to WebSphere Application Server using one of the two communication mechanisms:

  - Using Simple Object Access Protocol (SOAP is the default) by communicating with the embedded HTTP server in the Web container.
  - Using Remote Method Invocation (RMI) to communicate with the administrative services.

- Task automation with Ant

  With Another Neat Tool (Ant), you can create build scripts that compile, package, install, and test your application on WebSphere Application Server.

- Administrative applications

  You can develop custom Java applications that use the Java Management Extensions (JMX™) based on the WebSphere Application Programming Interface (API).

- Command line utilities

  WebSphere Application Server provides administrative utilities to help manage your environment. They offer the following features:

  - Called from a command line
  - Can be used to perform common administrative tasks such as starting and stopping WebSphere Application Server, backing up the configuration, and so on
  - Work on local servers and nodes only, including the deployment manager

The choice of which combination of administrative tools you will employ depends on the size and complexity of your runtime environment.

There are multiple levels of administration in WebSphere Application Server:

- In the WebSphere Application Server Express and Base packages, you can administer stand-alone server instances individually.
- In the WebSphere Application Server Express and Base packages, you can administer multiple stand-alone server instances on a single system using an administrative agent.
- In the WebSphere Application Server Network Deployment package, you can administer an entire cell of application servers using a deployment manager.
- You can administer multiple stand-alone application servers, administrative agents, and deployment managers using a job manager.

Web services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. WebSphere Application Server supports SOAP-based Web service hosting and invocation.

WebSphere Application Server can act as both a Web service provider and as a requester. As a requester, WebSphere Application Server hosts applications that invoke Web services from other locations. As a provider, WebSphere Application Server hosts Web services that are published for use by clients.

WebSphere Application Server V7.0 supports the Web Services for Java Enterprise Edition (Java EE) V1.2 specification. This specification defines the programming model and run-time architecture to deploy and look up Web services in the J2EE™ environment. More specifically, to deploy and look up Web services in the J2EE environment in the Web, EJB, and Client Application containers.

Web services support in WebSphere Application Server V7.0 also includes the following standards and specifications:

- Java API for XML Web Services (JAX-WS)

  The core programming model and bindings for developing and deploying Web services on the Java platform.

- WS Transaction support

  Defines how Web services applications can work within global transactions in enterprise environments using the following three specifications:

  - WS-Atomic Transaction (WS-AT)

    A specific coordination type that defines protocols for atomic transactions.

  - WS-Business Activity (WS-BA)

    A specific coordination type that defines protocols for business activities. A business activity is a group of general tasks that you want to link together so that the tasks have an agreed outcome.

  - WS-Coordination (WS-Coor)

    Specifies a context and a registration service with which participant Web services can enlist to take part in the protocols that are offered by specific coordination types.

- WS-I Basic Profile

  A set of non-proprietary Web services specifications that promote interoperability.

- WS- Notification

  Publish and subscribe messaging for Web services.

- WS-Addressing

  Enables systems to support message transmission and identification through networks that include firewalls or gateways in a transport-neutral manner.

- WS-Security

  This specification covers a standard set of SOAP extensions that can be used when building secure Web services to provide integrity and confidentiality. It is designed to be open to other security models including PKI, Kerberos, and SSL. WS-Security provides support for multiple security tokens, multiple signature formats, multiple trust domains, and multiple encryption technologies. It includes security token propagation, message integrity, and message confidentiality.

For a complete list of supported standards and specifications see the Web services section at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rovr_specs.html

Service integration

Service integration technology provides the communication infrastructure for messaging and service-oriented applications, unifying this support into a common component. Service integration includes the following features:

- A JMS 1.1 compliant JMS provider

  This provider is referred to as the default messaging provider.

- The service integration bus (referred to as the *bus*)

  The service integration bus provides the communication infrastructure for the default messaging provider. The bus supports the attachment of Web services requestors and providers.

- Support for the Web services gateway

  This provides you with a single point of control, access, and validation of Web service requests, enables you to control which Web services are available to different groups of Web service users.

Service integration bus

Service integration bus capabilities are fully integrated into WebSphere Application Server, enabling it to take advantage of WebSphere security, administration, performance monitoring, trace capabilities, and problem determination tools.

Figure 13 illustrates the service integration bus and how it fits into the larger picture of an enterprise service bus.

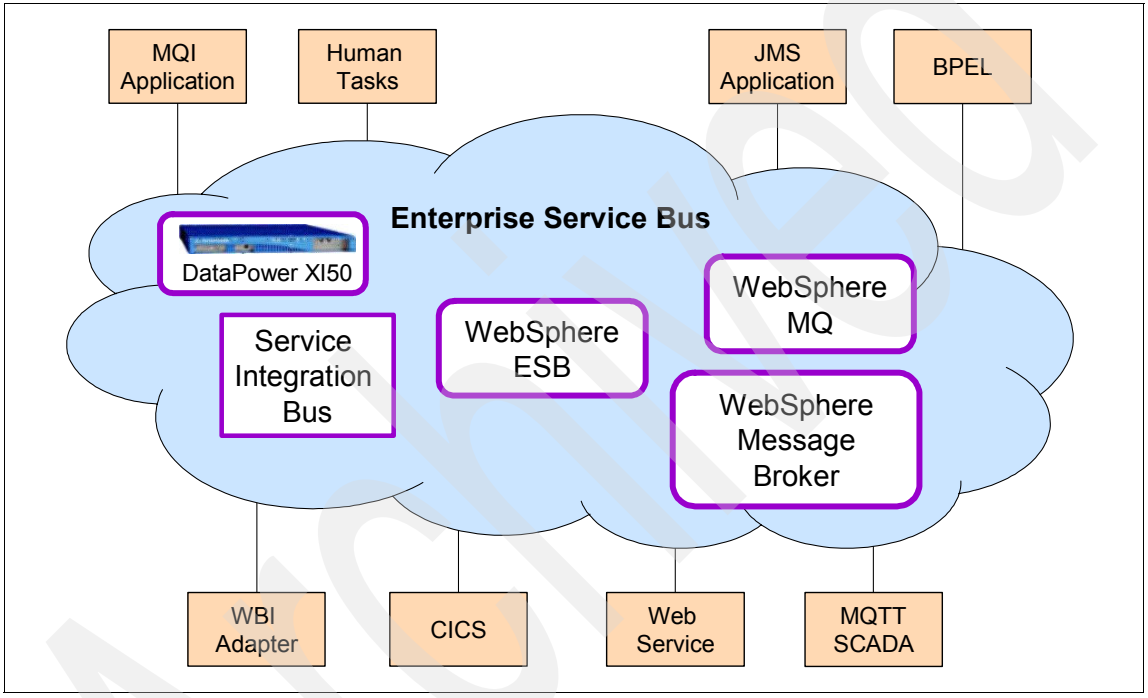

Figure 13 The enterprise service bus

A service integration bus consists of:

- Bus members

  Application servers or clusters that have been added to the bus.

- Messaging engine

  The application server or cluster component that manages bus resources. When a bus member is defined, a messaging engine is created automatically on the application server or cluster. The messaging engine provides a connection point for clients to produce or from where to consume messages.

  An application server has one messaging engine per bus of which it is a member. A cluster has at least one messaging engine per bus and can have more.

- Destinations

  The place within the bus to which applications attach to exchange messages. Destinations can represent Web service endpoints, messaging point-to-point queues, or messaging publish/subscribe topics. Destinations are created on a bus and hosted on a messaging engine.

- Message store

  A messaging engine uses a message store for message persistence and to save information that is needed for recovery in the event of a failure (including messages, subscription information, and transaction states). Each messaging engine has only one message store. This can be either a file-based message store or a database-based message store

  With a file store (the default), information is stored in a file system through the operating system.

  With a database-based message store, information is stored in tables of a relational database. Multiple messaging engines can share a database for the data store, each with its own set of tables and schema.

The service integration bus supports the following application attachments:

- Messaging applications

  JMS applications running in either WebSphere Application Server can connect to the bus using the JMS programming model.

- Web services
  - Requestors using the JAX-RPC API
  - Providers running in WebSphere Application Server as stateless session beans and servlets (JSR-109)
  - Requestors or providers attaching through SOAP/HTTP or SOAP/JMS

Service integration bus and messaging

With the Express or Base packages, you typically have one stand-alone server with one messaging engine on one service integration bus. With the Network Deployment package, you have more flexibility to use multiple buses for high availability and scalability.

Figure 14 illustrates two application servers, each with a messaging engine on a service integration bus.

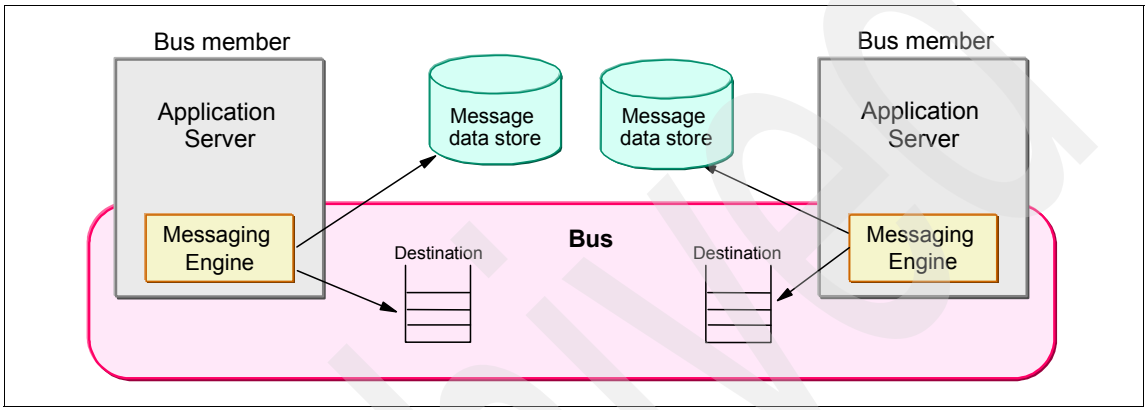

Figure 14 Service integration bus

The following topologies are valid:

- One bus and one messaging engine (application server or cluster)
- One bus with multiple messaging engines
- Multiple buses within a cell that may or may not be connected to each other
- Buses connected between cells
- One application server that is a member of multiple buses and that has one messaging engine per bus
- A connection between a bus and a WebSphere MQ queue manager

  When using this type of topology, you should consider the following points:

  - A messaging engine cannot participate in a WebSphere MQ cluster.
  - You can configure the messaging engine to look like another queue manager to WebSphere MQ.
  - WebSphere applications can send messages directly to WebSphere MQ, or though the service integration bus.
  - In WebSphere Application Server V7.0, you can use a JMS thin client as opposed to a full WebSphere Application Server client installation.

Clustering

In a distributed server environment, you can use clustering for high availability and scalability. You can add a cluster as a bus member and achieve the following results:

- High availability

  One messaging engine is active in the cluster. In the event that the messaging engine or server fails, the messaging engine on a standby server is activated.

- Scalability

  A single messaging destination can be partitioned across multiple active messaging engines in the cluster. Messaging order is not preserved.

Quality of service

You can define quality of service on a destination basis to determine how messages are (or are not) persisted. You can also specify quality of service within the application.

Message driven beans

Message driven beans (MDB) in the application server that listen to queues and topics are linked to the appropriate destinations on the service integration bus using JCA connectors (ActivationSpec objects).

Security

WebSphere Application Server provides you a set of features to help you to secure your systems and manage all resources. Figure 15 illustrates the components that make up the operating environment for security in WebSphere Application Server.

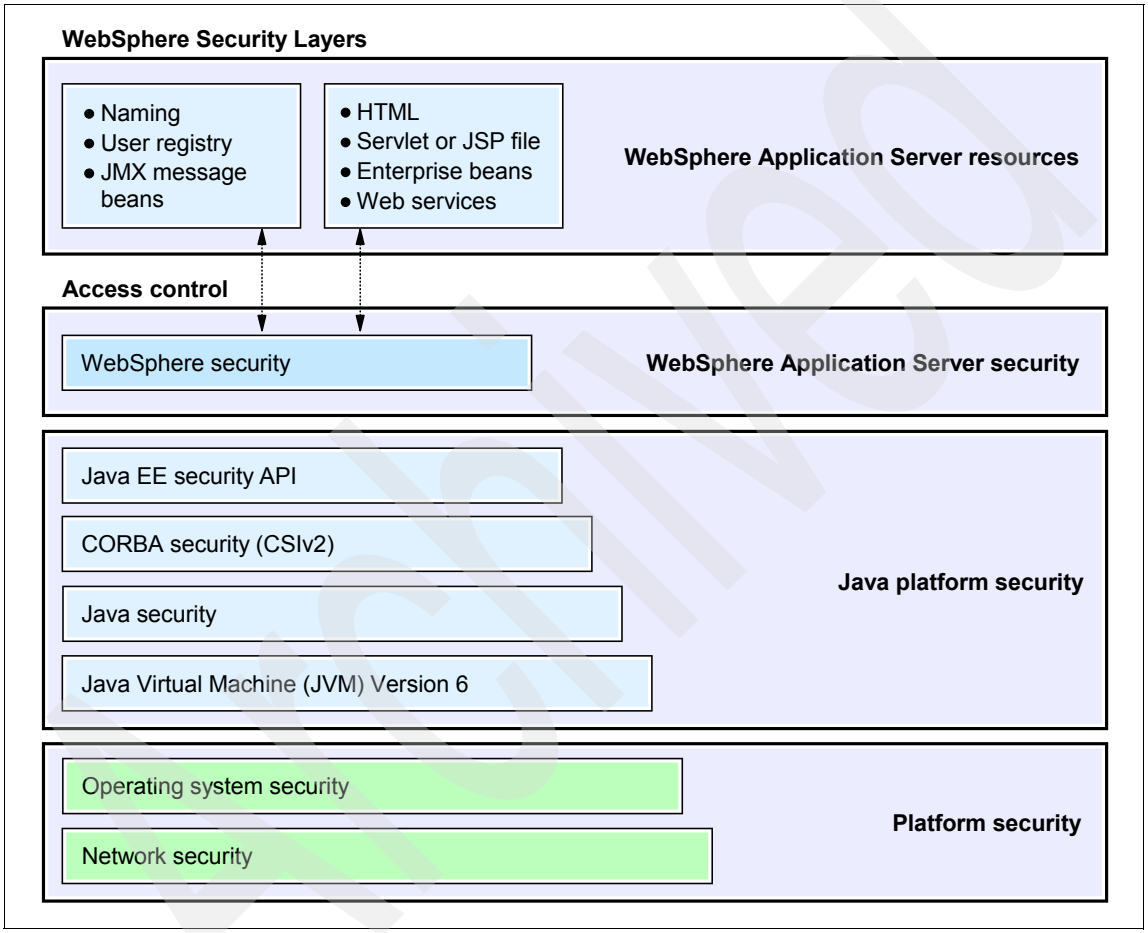

Figure 15 WebSphere Application Server security layers

These components consist of the following security components:

- WebSphere Application Server security

  WebSphere Application Server security enforces security policies and services in a unified manner on access to Web resources, enterprise beans, Web services, and JMX administrative resources. It consists of WebSphere Application Server security technologies and features to support the needs of a secure enterprise environment.

- Java platform security
  - Java EE security API

    The security collaborator enforces Java EE-based security policies and supports Java EE security APIs.

  - CSIv2 CORBA security

    Any calls made among secure Object Request Brokers (ORBs) are invoked over the Common Secure Interoperability Version 2 (CSIv2) security protocol, which sets up the security context and the necessary quality of protection.

  - Java security

    The Java security model offers access control to system resources including file system, system property, socket connection, threading, class loading, and so on. Application code must explicitly grant the required permission to access a protected resource.

  - Java virtual machine (JVM) 6.0

    The JVM security model provides a layer of security above the operating system layer. For example, JVM security protects the memory from unrestricted access, creates exceptions when errors occur within a thread, and defines array types.

- Platform security
  - Operating system security

    The security infrastructure of the underlying operating system provides certain security services for WebSphere Application Server. These services include the file system security support that secures sensitive files in the product installation for WebSphere Application Server.

  - Network security

    The network security layers provide transport level authentication and message integrity and confidentiality. You can configure the communication between separate application servers to use SSL. Additionally, you can use IP security and Virtual Private Network (VPN) for added message protection.

User registry

The information about users and groups reside in a *user registry*. In WebSphere Application Server, a user registry authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization. Before configuring the user registry or repository, decide which user registry or repository to use.

Although WebSphere Application Server supports different types of user registries, only one can be active in a certain scope. WebSphere Application Server supports the following types of user registries:

- Local operating system
- Standalone Lightweight Directory Access Protocol (LDAP)
- Federated repository (a combination of a file-based registry and one or more LDAP servers in a single realm).
- Custom registry

In the event that none of the first three options are feasible for you, you can implement a custom registry (for example, a database). WebSphere provides a service provider interface (SPI) that you can implement to interact with your custom user registry.

Authentication

Authentication is the process of identifying who is requesting access to a resource. For the authentication process, the server implements a challenge mechanism to gather unique information to identify the client. Secure authentication can be knowledge-based (user and password), key-based (physical keys, encryption keys), or biometric (fingerprints, retina scan, DNA, and so forth).

The authentication mechanism in WebSphere Application Server typically collaborates closely with a user registry. When performing authentication, the user registry is consulted. A successful authentication results in the creation of a credential, which is the internal representation of a successfully authenticated client user. The abilities of the credential are determined by the configured authorization mechanism.

Depending on the type of client, the authentication information is sent by using different protocols, as follows:

- Enterprise Beans clients use CSIv2
- Web clients use HTTP or HTTPS

Although WebSphere Application Server provides support for multiple authentication mechanisms, you can configure only a single active authentication mechanism at a time. WebSphere Application Server supports the following authentication mechanisms:

- Lightweight Third Party Authentication (LTPA)

  LTPA is intended for distributed, multiple application server and machine environments. It supports forwardable credentials and single sign-on (SSO). LTPA can support security in a distributed environment through cryptography. This support permits LTPA to encrypt, digitally sign, and securely transmit authentication-related data, and later decrypt and verify the signature.

- Kerberos

  Kerberos is a mature, standard authentication mechanism that enables interoperability with other applications that support Kerberos authentication. It provides single sign on (SSO) end-to-end interoperable solutions and preserves the original requester identity.

- Rivest Shamir Adleman (RSA) token authentication

  The RSA token authentication mechanism aids the flexible management objective to preserve the base profiles configurations and isolate them from a security perspective. This mechanism permits the base profiles managed by an administrative agent to have different Lightweight Third-Party Authentication (LTPA) keys, different user registries, and different administrative users. The RSA token authentication mechanism can only be used for administrative requests.

Authorization

Authorization is the process of checking whether a given user has the privileges necessary to get access to a requested resource. WebSphere Application Server supports many authorization technologies:

- Authorization involving the Web container and Java EE technology
- Authorization involving an enterprise bean application and Java EE technology
- Authorization involving Web services and Java EE technology
- Java Message Service (JMS)
- Java Authorization Contract for Containers (JACC)

WebSphere Application Server V7.0 supports both a default authorization provider, and, alternatively, an authorization provider that is based on the JACC specification. The JACC-based authorization provider enables third-party security providers to handle the Java EE authorization.

When a JACC provider is used for authorization, the Java EE application-based authorization decisions are delegated to the provider per the JACC specification. Figure 16 shows the communications flow.

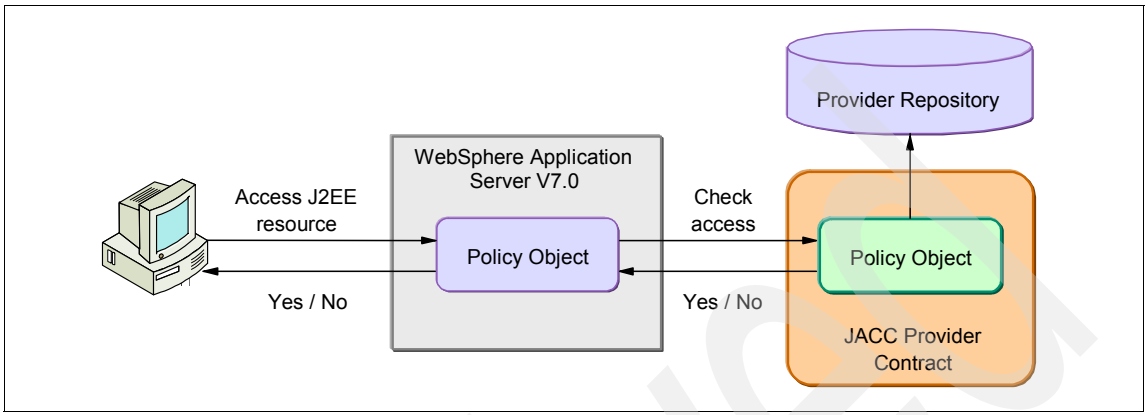

Figure 16 JACC provider architecture

Application development and deployment

The WebSphere Application Server V7.0 environment comes with a rich set of development tools. All editions of WebSphere Application Server V7.0 include a full licensed version of the Rational Application Developer Assembly and Deploy V7.5, as well as a 60 day trial version of the Rational Application Developer for WebSphere Software V7.5.

- Rational Application Developer Assembly and Deploy V7.5

  Supports only the version of the WebSphere Application Server with which it ships. This means that Rational Application Developer Assembly and Deploy V7.5 supports most new features of WebSphere Application Server V7.0 and supports it as an integrated test environment. It does not, however, support any of the previous versions of WebSphere Application Server as integrated test environments.

- Rational Application Developer for WebSphere Software V7.5

  Supports all new features of WebSphere Application Server V7.0. It is a fully featured integrated development environment for developing SIP, Portlet, Web services, and Java EE applications. It supports previous versions of WebSphere Application Server (V6.0 and V6.1) as an integrated test environment.

Source code management

Support for team development is provided by source code management (SCM) systems. Rational Application Developer Assembly and Deploy V7.5 and Rational Application Developer for WebSphere Software V7.5 support the following SCM systems:

- Rational ClearCase®

  Rational ClearCase organizes its code repositories as Versioned Object Bases (VOBs). VOBs contain versioned file and directory elements. Users of Rational ClearCase are organized according to their roles. Each user has their own view of the data that is in the VOB on which they are working. Rational ClearCase tracks VOBs and views. It also co-ordinates the checking in and checking out of VOB data to and from views.

- Concurrent Versions System (CVS)

  CVS uses a branch model to support multiple courses of work that are somewhat isolated from each other but still highly interdependent. Branches are where a development team shares and integrates ongoing work. A branch can be thought of as a shared workspace that is updated by team members as they make changes to the project. This model enables individuals to work on a CVS team project, share their work with others as changes are made, and access the work of others as the project evolves.

- Subversion

  Subversion is a free open source version control system that tracks the entire file systems and files. It versions directories and individual files and stores them into a repository.

Application deployment

Applications are installed on application servers using the administrative console or the **wsadmin** scripting interface. You can deploy an application to a single server or a cluster. In the case of a cluster, it is installed on each application server in the cluster. Installing an application involves the following tasks:

- Binding resource references (created during packaging) to actual resources (For example, a data source would have to be bound to a real database)
- Defining JNDI names for EJB home objects.
- Specifying data source entries for entity beans.
- Binding EJB references to the actual EJB JNDI names.
- Mapping Web modules to virtual hosts.
- Specifying listener ports for message-driven beans.
- Mapping application modules to application servers.
- Mapping security roles to users or groups.

After a new application is deployed, the Web server plug-in configuration file has to be regenerated and copied to the Web server.

Application update

WebSphere Application Server allows partial updates to applications and makes it possible to restart only parts of an application. Updates to an application can consist of individual application files, application modules, zipped files that contain application artifacts, or the complete application. All module types can be started (but only Web modules can be stopped).

WebSphere Application Server has a rollout start option for installing applications on a cluster that will stop, update, and start each cluster member in turn, ensuring availability.

WebSphere Rapid Deployment

WebSphere Rapid Deployment is designed to simplify the development and deployment of WebSphere applications. It is a collection of Eclipse plug-ins that can be integrated within development tools or run in a headless mode from a user file system. WebSphere Rapid Deployment is currently integrated into the WebSphere Application Server V7.0 packaging.

During development, annotation-based programming is used. The developer adds metadata tags into the application source code, which are used to generate artifacts needed by the code, thus reducing the number of artifacts the developer has to create.

These applications are packaged into an enhanced EAR file that contains the Java EE EAR file along with deployment information, application resources, and properties (environment variables, JAAS authentication entries, shared libraries, classloader settings, and JDBC resources). During installation, this information is used to create the necessary resources. Moving an application from one server to another also moves the resources.

WebSphere Rapid Deployment automates installation of applications and modules onto a running application server by monitoring the workspace for changes and then driving the deployment process.

WebSphere Application Server Feature Packs

A WebSphere Application Server Feature Pack is an optionally installable product extension for WebSphere Application Server that provides a set of new related standards and innovative features. With feature packs, users can take advantage of these new standards and features without having to wait for a new release of WebSphere Application Server.

The currently available feature packs for WebSphere Application Server V7.0 are as follows:

- WebSphere Application Server Feature Pack for Web 2.0

  This feature pack extends SOA by connecting external Web services, internal SOA services, and Java EE objects into highly-interactive Web application interfaces. It provides a supported, best-in-class Ajax development toolkit for WebSphere Application Server, and also a rich set of extensions to Ajax.

  Figure 17 shows the main components of this feature pack.

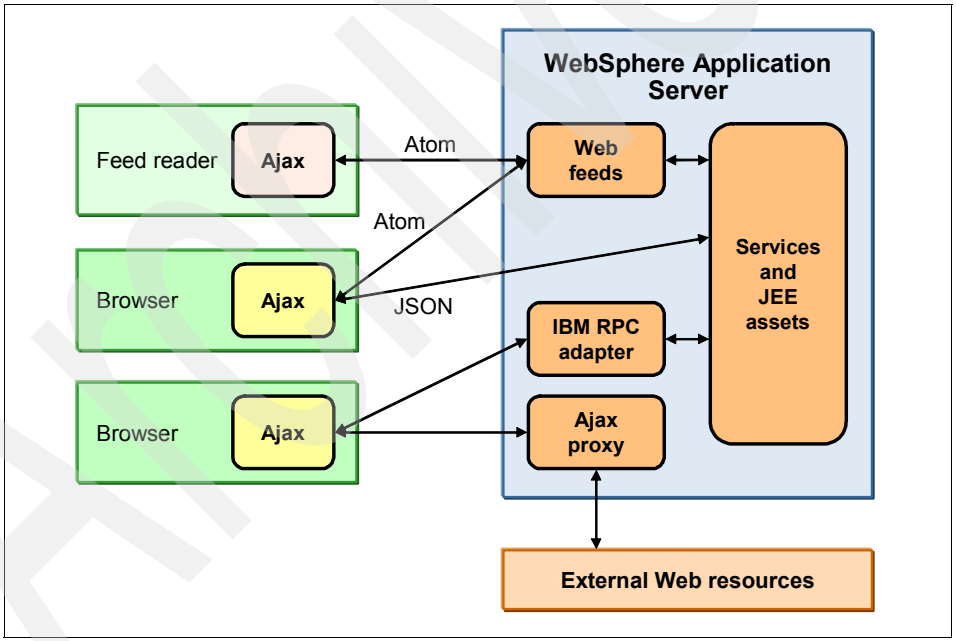

Figure 17 Components of WebSphere Application Server Feature Pack for Web 2.0

- WebSphere Application Server Feature Pack for Service Component Architecture (SCA)

  SCA is a set of specifications that constitutes a programming model for building applications using a service oriented architecture (SOA). SCA extends other SOA technologies, like Web services, while providing a platform and language-neutral component model based on open standards specified by the Open SOA Collaboration (OSOA).

  The WebSphere Application Server V7.0 Feature Pack for SCA adds support for deploying SCA applications to the application server. Both JAR and WAR files are supported. This feature pack is based on a Tuscany open source Java implementation and covers SCA V1.0.

Integration with other products

WebSphere Application Server works closely with other IBM products to provide a fully integrated solution. This section introduces some of these products, including those that provide enhanced security and messaging options, and broad integration features.

Tivoli Access Manager

IBM Tivoli® Access Manager provides centralized authentication and authorization services.

The WebSphere Application Server security infrastructure is adequate for many situations and circumstances. However, integrating WebSphere Application Server with Tivoli Access Manager allows for an end-to-end integration of application security across the entire enterprise.

This solution uses the following components:

- User repository

  Tivoli Access Manager requires a user repository such as IBM Tivoli Directory Server or Microsoft Active Directory®. Tivoli Access Manager can be configured to use the same user repository as WebSphere Application Server, enabling you to share user identities with both Tivoli Access Manager and WebSphere Application Server.

- Tivoli Access Manager policy server

  This component maintains the master authorization policy database, which contains the security policy information for all resources and all credentials information of all participants in the secure domain (both users and servers).

- Tivoli Access Manager client

  This client is embedded in WebSphere Application Server. The Tivoli Access Manager client can be configured using the scripting and GUI management facilities of WebSphere Application Server.

Figure 18 shows the integration interfaces between WebSphere Application Server and Tivoli Access Manager.

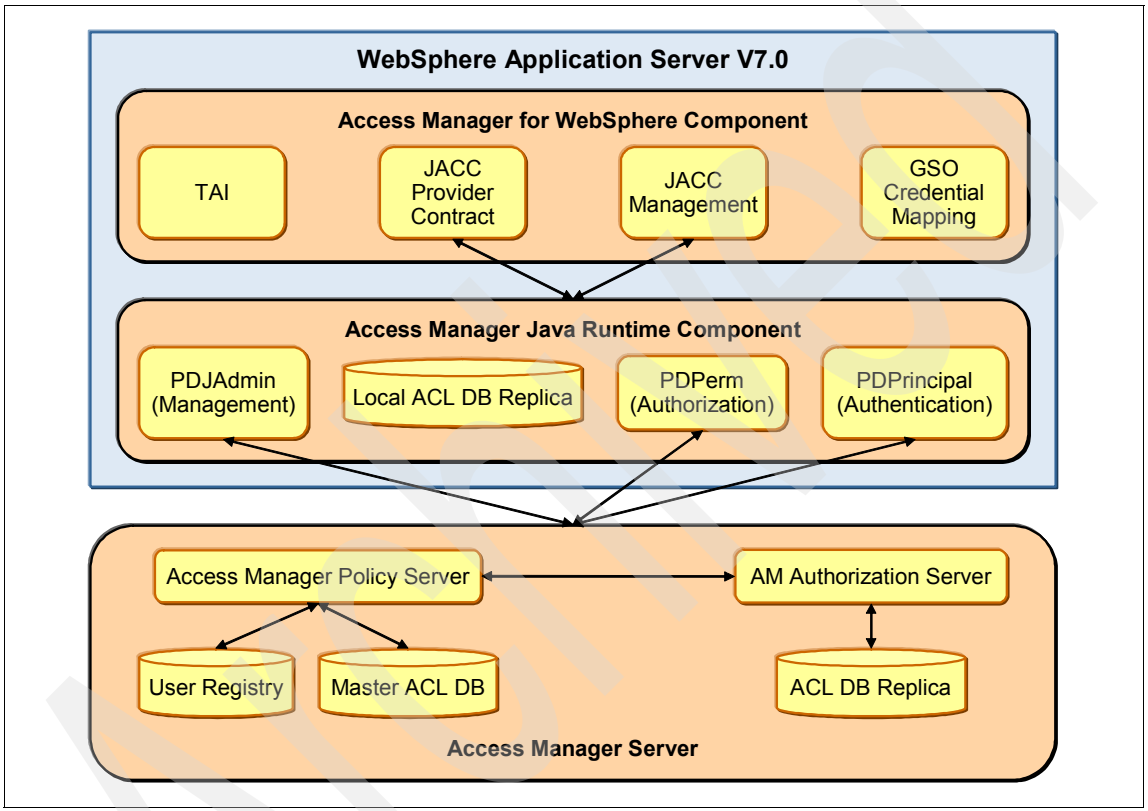

Figure 18 Integration of WebSphere Application Server with Tivoli Access Manager

Tivoli Directory Server

IBM Tivoli Directory Server provides a high-performance LDAP identity infrastructure. Tivoli Directory Server can be used as a standalone LDAP registry for the user account repository of WebSphere Application Server. You can configure your user account repository through the Integrated Solutions Console or through the **wsadmin** command line tool.

WebSphere MQ

WebSphere MQ is an asynchronous messaging technology designed for application-to-application communication. WebSphere MQ is available on a large number of platforms and operating systems. It offers a fast, robust, and scalable messaging solution that assures one time only delivery of messages to queue destinations that are hosted by queue managers.

WebSphere Application Server provides its own JMS V1.1 compliant default messaging engine through the service integration bus. In some instances it is advantageous to use a topology that includes both WebSphere MQ and the service integration bus. There are two mechanisms to allow interaction between them:

- Extend the WebSphere MQ and service integration bus networks by defining a WebSphere MQ link on a messaging engine in a WebSphere Application Server that connects the service integration bus to a WebSphere MQ queue manager.
- Integrate specific WebSphere MQ resources into a service integration bus for direct, synchronous access from default messaging applications running in WebSphere Application Servers. This is achieved by representing a queue manager or queue sharing group as a WebSphere MQ server in the WebSphere Application Server cell, and adding it to a service integration bus as a bus member.

Figure 19 shows an example integration for WebSphere Application Server and WebSphere MQ.

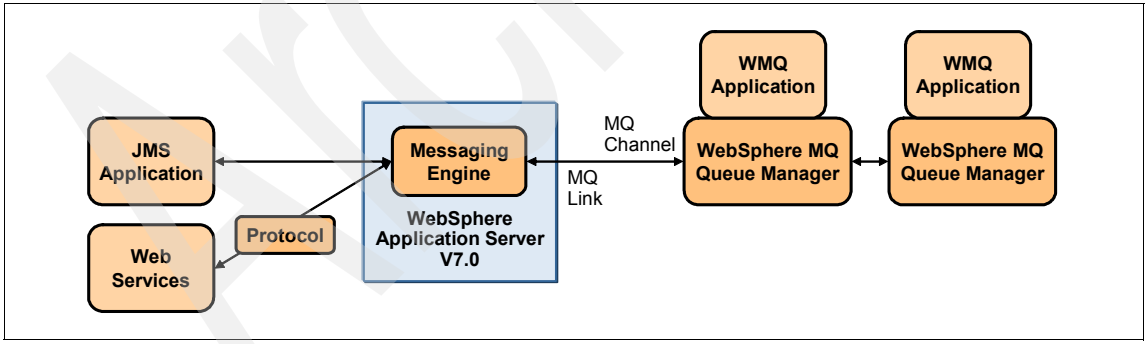

Figure 19 WebSphere Application Server integration with WebSphere MQ

WebSphere adapters

IBM WebSphere Adapters provide a set of generic technology and business application adapters with wizards that quickly and easily service enable Enterprise Information Systems (EISs).

WebSphere Adapters includes three types of adapters:

- Application Adapters

  Application adapters integrate enterprise business application suites. Examples include JD Edwards® EnterpriseOne and SAP® Software.

- Technology Adapters

  Technology adapters deliver file and database connectivity solutions. Example include Flat Files and Java Database Connectivity (JDBC).

- WBI Adapters

  WBI adapters use an asynchronous standalone runtime architecture originally designed for WebSphere InterChange Server.

WebSphere Adapters are designed to plug into WebSphere Application Server and to provide bidirectional connectivity between enterprise applications (or Java EE components), WebSphere Application Server, and EIS. Figure 20 shows the relation between WebSphere Application Server and a WebSphere Adapter.

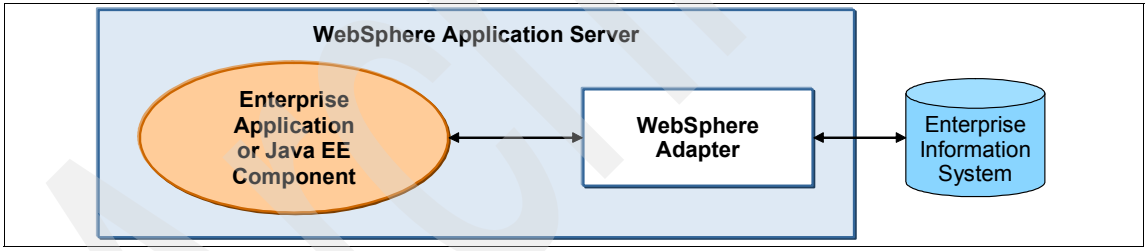

Figure 20 WebSphere Adapter integration with WebSphere Application Server

WebSphere DataPower

IBM WebSphere DataPower® SOA Appliances are rack-mountable network devices that provide a variety of XML, Web service, and security processing capabilities.

In WebSphere Application Server V7.0, the new consolidated administration feature for WebSphere DataPower allows you to manage and integrate appliances into your environment. The Integrated Solutions Console contains an administration interface, the DataPower appliance manager, to manage multiple

WebSphere DataPower appliances. The Integrated Solutions Console is the single point of administration to manage WebSphere Application Server, WebSphere DataPower, and solutions that combines the two.

Tivoli Composite Application Manager for WebSphere

IBM Tivoli Composite Application Manager for WebSphere (ITCAM for WebSphere) is an application management tool that helps maintain the availability and performance of on demand applications.

ITCAM for WebSphere enables you to analyze the health of WebSphere Application Server and the transactions that are invoked in it. It is able to trace the transaction execution to the detailed method-level information, and connects transactions that spawn from one application server and invokes services from other application servers, including mainframe applications in IMS or CICS®. ITCAM for WebSphere provides flexible monitoring, from an non-intrusive production ready monitor, to a detailed deep-dive tracing for problems of locking or even memory leaks. ITCAM for WebSphere provides a separate interactive Web console and allows monitoring data to be displayed on the Tivoli Enterprise Portal. Figure 21 shows the overall architecture of ITCAM for WebSphere.

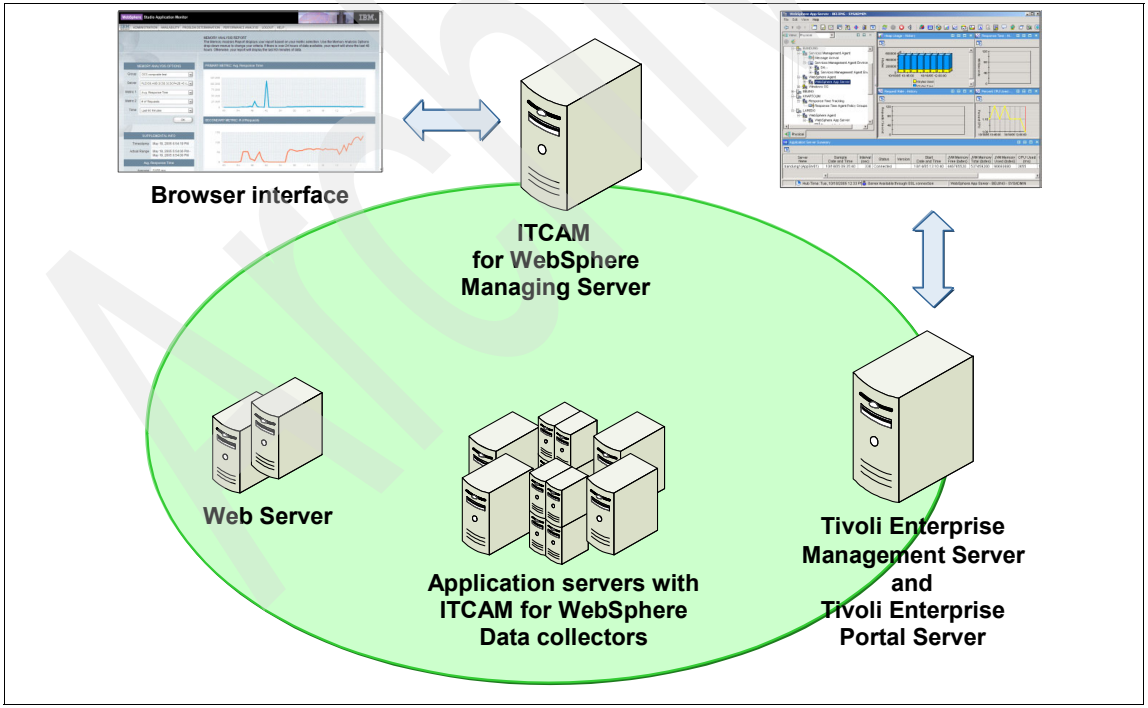

Figure 21 ITCAM for WebSphere architecture

For more information

Consult the following resources for more information.

- Announcement letter

  http://www.ibm.com/common/ssi/rep_ca/6/897/ENUS208-266/
- System requirements for WebSphere Application Server V7.0

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg27012284
- WebSphere Application Server V7.0 Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp

The team that wrote this paper

This paper was produced by a team of specialists from around the world working with the International Technical Support Organization (ITSO).

Arden Agopyan is an IT Specialist for WebSphere Technical Sales working at Software Group, IBM Turkey since 2006. He has worked with WebSphere Application Server since V5.1 and is an expert on planning, design, implementation, and problem determination of WebSphere Application Server solutions. Before joining IBM he worked as a senior Java and .NET solutions developer. He holds a Computer Engineer degree from Galatasaray University in Istanbul (Turkey).

Hermann Huebler is an IT Specialist working for IBM Global Services Strategic Outsourcing Service Delivery in Austria. He has 21 years of experience in IT and is working for IBM since 1994. After working as an System i specialist for several years, he started focusing on WebSphere products on distributed platforms in 2001. His main areas of expertise are implementation, problem determination, high availability, and performance tuning of the WebSphere Application Server product family. These products include WebSphere Application Server, WebSphere Portal, WebSphere MQ, Edge Components.

Tze Puah is an Application Infrastructure Services Principal working for Coles Group Ltd in Melbourne, Australia since 2005. His areas of expertise include infrastructure architecture design, implementation, problem determination and performance tuning on WebSphere and Tivoli products. These products include WebSphere Application Server, WebSphere Portal Server, WebSphere Process Server, WebSphere Commerce Server, WebSphere MQ, WebSphere Message Broker and Tivoli Access Manager. He has 14 years of experience in IT and holds a master degree in Computer Science.

Thomas Schulze is a Senior IT Specialist working since 2003 in System z pre-sales support in Germany. His areas of expertise include WebSphere Application Server running on the System z platform with a speciality on performing Healthchecks and Performance Tuning. He has written extensively on the z/OS parts of this book. He has 10 years of experience in IBM and holds a degree in Computer Science from the University of Cooperative Education in Mannheim (Germany).

David Soler is an IT Architect working for IBM Global Services in Barcelona, Spain. He has more than 19 years of experience playing different technical roles in the IT field and he is working for IBM since 2000. His areas of expertise include WebSphere Application Server, ITCAM for WebSphere, ITCAM for Response Time, and Unix systems. He holds a degree in Computer Science from the Universitat Politècnica de Catalunya (Spain).

Martin Keen is a Senior IT Specialist at the ITSO, Raleigh Center. He writes extensively about WebSphere products, and SOA. He also teaches IBM classes worldwide about WebSphere, SOA, and ESB. Before joining the ITSO, Martin worked in the EMEA WebSphere Lab Services team in Hursley, UK. Martin holds a bachelor's degree in Computer Studies from Southampton Institute of Higher Education.

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

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

This document REDP-4482-00 was created or updated on January 24, 2011.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, 2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks