



Carla Sadtler
Jan Bajerski
Davide Barillari
Libor Cada
Susan Hanson
Guo Liang Huang
Rispna Jain
Shishir Narain
Jennifer Ricciuti
Christian Steege

WebSphere Application Server V8.5.5 Technical Overview

IBM® WebSphere® Application Server is the leading software foundation for service-oriented architecture (SOA) applications and services for your enterprise. With IBM WebSphere Application Server, you can build business-critical enterprise applications and solutions and combine them with innovative new functions. The WebSphere Application Server family includes and supports a range of products that helps you develop and serve your business applications. These products make it easier for clients to build, deploy, and manage dynamic websites and other more complex solutions productively and effectively.

IBM WebSphere Application Server is the implementation by IBM of the Java Platform, Enterprise Edition (Java EE) platform. It conforms to the Java EE 6 specifications as one of its supporting programming models. WebSphere Application Server is available in unique packages that are designed to meet a wide range of client requirements. Each package has its own unique characteristics, but at the heart of all of them is the same runtime application server environment.

This IBM Redpaper™ publication provides an overview of the technical features of the runtime server component in WebSphere Application Server V8.5.5. Throughout the paper, the phrase (*New in V8.5.5*) indicates changes introduced in this version.

For more detailed information about topics included in this paper, see the following companion IBM Redbooks® and Redpaper publications:

- ▶ *WebSphere Application Server: New Features in V8.5.5* (published)
<http://www.redbooks.ibm.com/abstracts/redp4870.html?Open>
- ▶ *WebSphere Application Server V8.5 Concepts, Planning, and Design Guide*
<http://www.redbooks.ibm.com/abstracts/sg248022.html>
- ▶ *WebSphere Application Server V8.5 Administration and Configuration Guide for the Full Profile*
<http://www.redbooks.ibm.com/abstracts/sg248056.html?Open>
- ▶ *WebSphere Application Server Liberty Profile Guide for Developers*
<http://www.redbooks.ibm.com/abstracts/sg248076.html>
- ▶ *WebSphere Application Server V8.5 Administration Guide for the Liberty Profile*
<http://www.redbooks.ibm.com/abstracts/sg248170.html>

Overview of WebSphere Application Server

WebSphere Application Server is a key SOA building block, providing the role of the business application services (circled in Figure 1) in the SOA Reference Architecture.

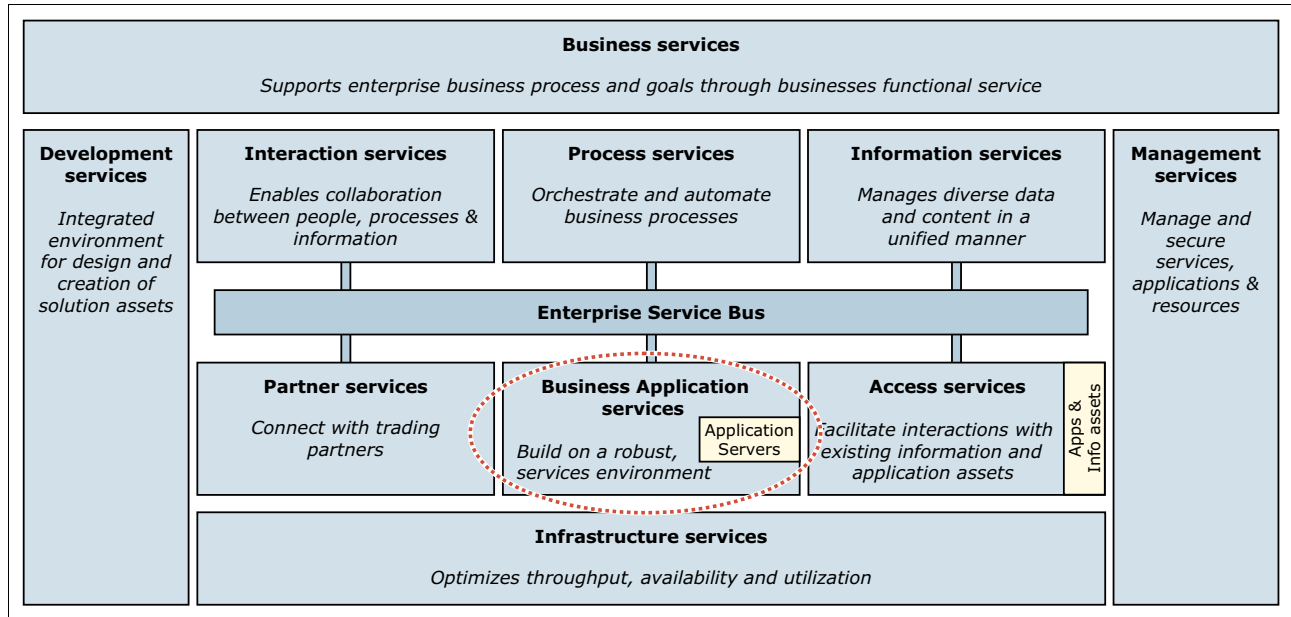


Figure 1 Position of business application services in an SOA reference architecture

From an SOA perspective, with WebSphere Application Server you can perform the following functions:

- ▶ Build and deploy reusable application services quickly and easily
- ▶ Run services in a secure, scalable, and highly available environment
- ▶ Connect software assets and extend their reach
- ▶ Manage applications effortlessly
- ▶ Grow as needs evolve, reusing core skills and assets

WebSphere Application Server packaging

WebSphere Application Server is available on a range of platforms and in multiple packages to meet specific business needs. By providing the application server that is required to run specific applications, it also serves as the base for other WebSphere products and many other IBM software products. In addition to the application server component, each package contains an appropriate combination of complementary products such as IBM HTTP Server, IBM Assembly and Deploy Tools for WebSphere Administration, Edge Components, and other products.

Because different application scenarios require different levels of application server capabilities, WebSphere Application Server is available in multiple packaging options. Although these options share a common foundation, each provides unique benefits to meet the needs of applications and the infrastructure that supports them. At least one WebSphere Application Server product fulfills the requirements of any particular project and its supporting infrastructure. As your business grows, the WebSphere Application Server family provides a migration path to more complex configurations.

The following packages are available:

- ▶ WebSphere Application Server Express
- ▶ WebSphere Application Server Base
- ▶ WebSphere Application Server Network Deployment
- ▶ WebSphere Application Server for IBM z/OS®
- ▶ WebSphere Application Server for Developers
- ▶ WebSphere Application Server Hypervisor Edition
- ▶ WebSphere Application Server Liberty Core
- ▶ WebSphere Application Server Community Edition

Figure 2 provides a high-level look at the packaging options for WebSphere Application Server.

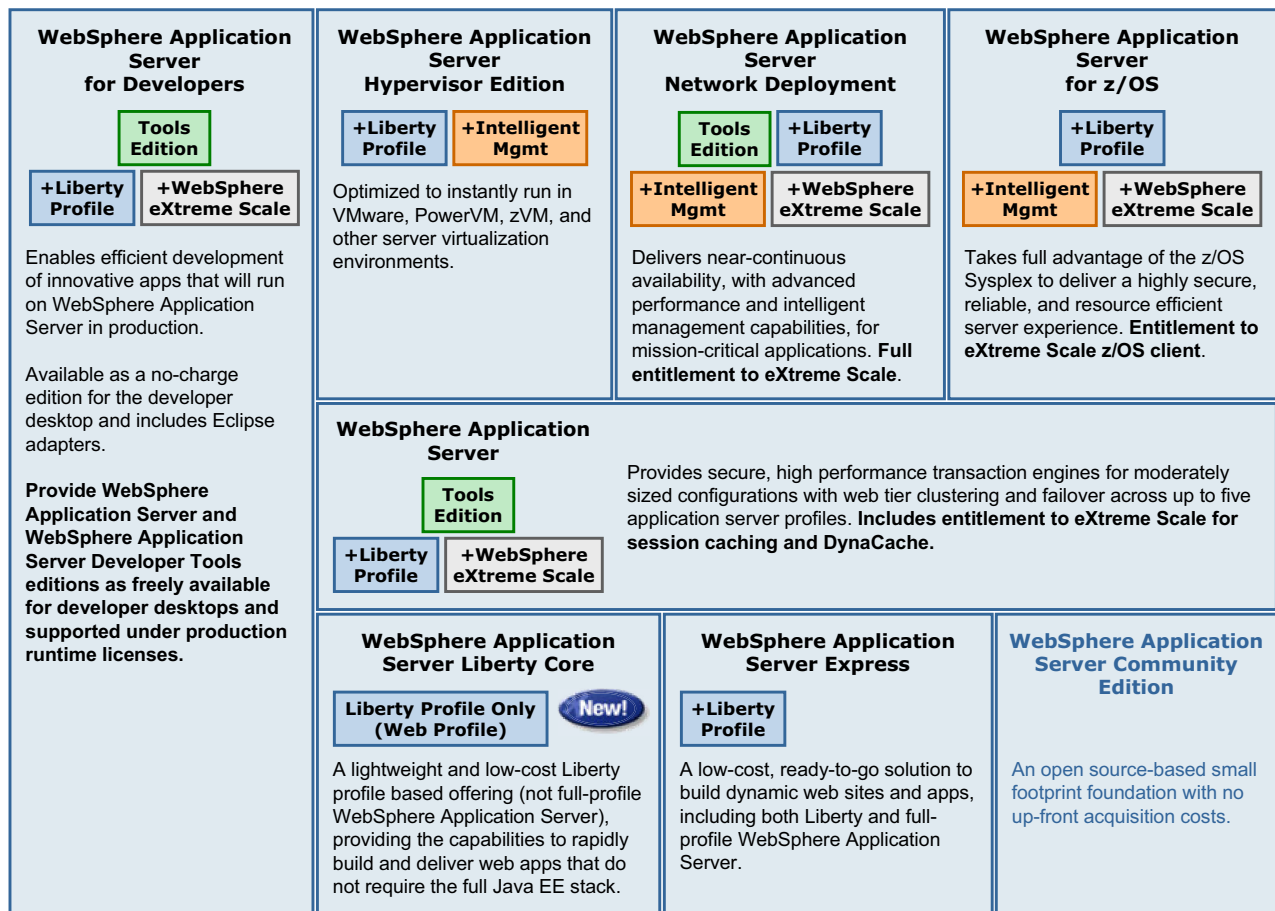


Figure 2 WebSphere Application Server packing overview

Figure 2 shows several differentiators among the packages, including the following items:

- ▶ Profiles: Each package, with the exception of the Liberty Core and Community Edition, includes two runtime profiles. The full profile offers a full range of programming and runtime options. The Liberty profile is a lightweight server and low-cost runtime that provides the capabilities to rapidly build and deliver web applications that do not require the full Java EE stack.

The Liberty Core runtime meets the specifications of the Java EE 6 Web profile. The Liberty profile runtime included with the other packages includes the Web profile capabilities and additional capabilities that vary with the package.

WebSphere Application Server Community Edition is a lightweight Java EE application server that is built on a different code base from the other packages. It offers the IBM distribution of Apache Geronimo, with broader platform support and enhanced documentation.

- ▶ **Intelligent Management:** This feature enables the runtime to dynamically scale to meet service level agreements (SLAs) and provides enhanced resiliency capabilities.
- ▶ **WebSphere eXtreme Scale:** This product provides elastic caching capabilities, providing consistent application and transaction response times, and providing for linear scaling and fault tolerance features for data.
- ▶ **Tools Edition:** The Tools Editions are bundles of WebSphere Application Server runtime and development tools. For more information about the Tools Editions, see:
<http://www-01.ibm.com/software/webservers/appserv/was/tools/>

There are additional differences in the licensing terms, features, and products included with each package.

WebSphere Application Server Express

The WebSphere Application Server Express package provides a strong and affordable application server based on standards. It is a ready-to-go application foundation for single server, small scale deployments of dynamic web applications. When your business needs to grow, this package is easily integrated with more advanced versions of the WebSphere Application Server family. This package is also referred to as the *Express* package.

For more information about WebSphere Application Server Express, see:

<http://www.ibm.com/software/webservers/appserv/express/>

WebSphere Application Server (Base)

The WebSphere Application Server package is the next level of server infrastructure in the WebSphere Application Server family. Although the WebSphere Application Server package is functionally equivalent to WebSphere Application Server Express (single-server environment), this package differs in packaging and licensing. This package is ideal for lightweight application solutions where cost and simplicity are key. This package is also referred to as the *Base* package.

The Tools Edition bundle of this package includes WebSphere Application Server for production use and an unlimited number of licenses of IBM Rational® Application Developer and WebSphere Application Server Developer Tools for Eclipse for application development against the runtime included in the bundle.

For more information about the WebSphere Application Server Base package, see:

<http://www.ibm.com/software/webservers/appserv/was/>

WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment provides enterprises with advanced performance, management, and high availability for mission-critical applications. It extends the base package of WebSphere Application Server to include key features that tend to be more important in larger enterprises, where applications tend to service a larger client base and where more elaborate performance and high availability requirements are in place.

These features include various functions for high availability and high scalability, to ensure responsiveness to your application requests. This package is also referred to as the *Network Deployment* package.

The Tools Edition bundle of this package includes WebSphere Application Server Network Deployment for production use and an unlimited number of licenses of Rational Application Developer and WebSphere Application Server Developer Tools for Eclipse for application development against the runtime included in the bundle.

For more information about WebSphere Application Server Network Deployment, see:

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

WebSphere Application Server for z/OS

IBM WebSphere Application Server for z/OS provides the same functions of WebSphere Application Server Network Deployment, but adds key functions that use IBM z/OS qualities of service. You can use the z/OS qualities of service to optimize performance and provide continuous availability for mission-critical applications. WebSphere Application Server for z/OS includes capabilities to deliver business objectives that can help contain or reduce costs for business-critical applications using the full capabilities of the z/OS platform.

For more information about WebSphere Application Server for z/OS, see:

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

WebSphere Application Server for Developers

The WebSphere Application Server for Developers package is functionally equivalent to the Base and Express packages, but it is licensed for development use only. It provides simplified and no-charge access to enable developers to build and test in the same environment that ultimately supports their applications. WebSphere Application Server for Developers is an easy-to-use development environment to build and test applications for clients who are not using IBM Rational Application Developer as their integrated development environment (IDE) for developing and testing their applications.

The Tools Edition bundle of this package includes WebSphere Application Server for Developers and WebSphere Application Server Developer Tools for Eclipse. This bundle is for single user, development use only.

For more information about WebSphere Application Server for Developers V8.5, see:

<http://www.ibm.com/software/webservers/appserv/developer/index.html>

WebSphere Application Server Hypervisor Edition

The WebSphere Application Server Hypervisor Edition is a self-contained virtual machine image that contains a guest operating system and WebSphere Application Server Network Deployment Version V8.5. You can run the virtual image on VMware ESX or ESXi hypervisors.

WebSphere Application Server Liberty Core

New with V8.5.5, WebSphere Application Server Liberty Core is a lightweight and low-cost runtime that provides the capabilities to rapidly build and deliver web applications that do not

require the full Java EE stack. Liberty core is a production-ready web profile runtime. It does not include the wider range of features that you will find in the Liberty profile shipped with the other WebSphere Application Server packages.

WebSphere Application Server Community Edition

WebSphere Application Server Community Edition is a lightweight, Java EE application server. It offers the IBM distribution of Apache Geronimo, with broader platform support and enhanced documentation. WebSphere Application Server Community Edition is available at no charge. Note that in Figure 2 on page 3, this edition is shown in a lighter print because it is not built on the same code base as the other WebSphere Application Server editions.

WebSphere Application Server profiles

WebSphere Application Server V8.5 provides two runtime profiles. Every WebSphere Application Server package includes both profile types.

Full WebSphere Application Server

The runtime traditionally available with the WebSphere Application Server packages is referred to as the *full profile*. The application serving runtime (*application server*) provided with this profile is composed of a wide spectrum of runtime components that are always available when the server is started. The full profile provides support for Java Platform, Enterprise Edition 6 (Java EE 6), and Enterprise OSGi technologies.

In addition to the application servers, the full profile supports the creation of generic server definitions to configure other servers or processes that are necessary to support the application server environment.

Additional capabilities, such as clustering application servers for load balancing and high availability, vary depending on the WebSphere Application Server package.

Liberty profile

In addition to the full profile, a new *Liberty profile* is included with each package and (*New in V8.5.5*) the Liberty profile is also available as a stand-alone offering known as WebSphere Application Server Liberty Core.

The Liberty profile provides a simplified stand-alone run time for web applications, supporting a subset of the programming model available with the full profile. It is a useful option for developers who are building web applications that do not require the full Java EE environment of traditional enterprise application server profiles. Any application that runs on the Liberty profile will also run on the full profile.

The application-serving environment is configured with the correct level of capabilities that are required for the individual applications. You can use the Liberty profile to specify only those features that are required for the applications that are deployed, thus reducing the memory footprint and increasing performance. The Liberty profile has a simplified installation and uses an easy-to-configure XML configuration file format.

The Liberty profile is optimized for use in both development and production environments. Within the development environment, the Liberty profile supports the same platforms as the

base application server and Mac OS. Enterprise qualities of service, such as security and transaction integrity, are enabled as required.

WebSphere Application Server full profile concepts

WebSphere Application Server V8.5 provides the infrastructure and capabilities that are required to host applications and proxy servers that distribute work to the application servers. You can define generic external servers to the administration process and associate web servers for routing purposes.

Servers

When using WebSphere Application Server V8.5, you can have the following types of servers:

- ▶ Application servers
- ▶ Proxy servers
- ▶ Generic servers

Application servers

The *application server* is the core of the WebSphere Application Server product and is the runtime environment on which business applications run. It provides containers and services, such as database connectivity, threading, and workload management, that can be used by applications. Each application server runs in its own Java virtual machine (JVM) and can have one or many applications deployed to it.

Application server *containers* provide runtime support for applications. They are specialized to execute specific types of applications and can interact with other containers by sharing session management, security, and other attributes. WebSphere Application Server V8.5 application servers include the following containers that each provide support for a different programming model:

- ▶ The *web container* supports the Java EE programming model around servlets, JSPs, and other server-side includes.
- ▶ The *Enterprise JavaBeans (EJB) container* supports the Java EE programming model around deploying and managing enterprise beans, including session and entity beans.
- ▶ The *portlet container* supports JSR 286-compliant portlets as an extension to the web container.
- ▶ The *SIP container* supports Session Initiation Protocol servlets that are written to the JSR-116 specification, and handles UDP, TCP, and TLS/TCP protocols.
- ▶ The *batch container* supports the execution of batch applications using either the transactional batch or compute-intensive programming models. The job scheduler sends jobs to the batch container and, within the batch container, the scheduled jobs (written in XML job control language (xJCL)) are run.
- ▶ The *OSGi Blueprint container* supports applications that are based on the OSGi framework. Applications can contain any mix of Java EE and OSGi technologies.

Because each application server runs in a single JVM, you can deploy applications using one of the following options, based on machine resources and risk:

- ▶ A model that uses one application per application server can help reduce the risk of JVM issues that are caused by one application that affects other applications. However, additional machine resource impact for each JVM can occur. So, there is a tradeoff.
- ▶ A model that uses multiple applications per application server optimizes system resources by minimizing the JVM machine resource impact. However, poorly written applications can cause negative impacts to other applications within the same JVM.

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 that enter the environment.

You can use WebSphere Application Server to create the following types of proxy servers:

- ▶ *WebSphere Application Server Proxy* is used to classify, prioritize, and route HTTP and SIP requests to servers in the enterprise. The proxy can also static and dynamic cache content from servers.
- ▶ *DMZ Secure Proxy Server* comes in a separate installation package and provides security enhancements to allow deployments inside a DMZ. It improves security by minimizing the number of external ports that are opened and running as an unprivileged user when binding to well-known ports.
- ▶ *On Demand Router (ODR)* is an intelligent Java-based HTTP proxy server and SIP proxy server built on the WebSphere run time. The ODR manages the flow of requests to application servers based on a set of configurable routing rules. It is asynchronous, high performing, scalable, and can be clustered for high availability.

(New in V8.5.5) In V8.5.5, the ODR can be implemented as separate server or as an ODR plug-in for the web server. Using the plug-in option reduces the number of systems in the topology.

Generic servers

A *generic server* is a server that can be managed as part of the administrative domain of WebSphere Application Server, but it is not supplied by WebSphere Application Server. With the generic servers function of WebSphere Application Server, you can define a generic server as an application server instance within the WebSphere Application Server administration and associate it with a non WebSphere Application Server or process.

Standalone and distributed application server environments

Application servers can be stand-alone or can be part of a distributed network of application servers. A stand-alone server does not provide workload management or high availability features. With a distributed application server topology, multiple application servers can be clustered to provide workload distribution and high availability.

With the Base and Express packages, the only option is stand-alone server. With the Network Deployment package, you can choose between stand-alone servers and a distributed environment.

WebSphere Application Server administration

WebSphere Application Server is organized based on the concept of a hierarchy of cells, nodes, and servers. Cells and nodes play a key role in the administrative structure and can be managed, along with their servers, by using a combination of deployment managers, administrative agents, and job managers.

The following components are used in the basic administrative structure:

- ▶ Nodes

A *node* is an administrative grouping of application servers for configuration and operational management within one operating system instance. You can create multiple nodes inside one operating system instance, but a node cannot span the operating system boundaries. A stand-alone application server configuration has only one node. With Network Deployment, you can configure a distributed server environment that consists of multiple nodes, which are managed from one central process called the *deployment manager*.

- ▶ Node groups

A *node group* is collection of nodes within a cell that have similar capabilities in terms of installed software, available resources, and configuration. A node group is used to define a boundary for server cluster formation so that the servers on the same node group host the same applications. A node group validates that the node can perform certain functions before allowing them. A default node group is created automatically. This default 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.

- ▶ Cells

A *cell* is a group of nodes in a single administrative domain that encompasses the entire management domain. In the Base, Developer, and Express packages, a cell contains one node and that node contains one server, which is largely hidden from view. Each cell contains at least one node, at least one application server, and an administration console. The configuration and application files for all nodes in the cell are centralized into the *master repository*.

Administering stand-alone application servers

Multiple stand-alone application servers can exist on a system, either through independent installations of WebSphere Application Server binary files or by creating multiple application server profiles within one installation, as shown in Figure 3. Each application server has its own administrative console application and is administered separately.

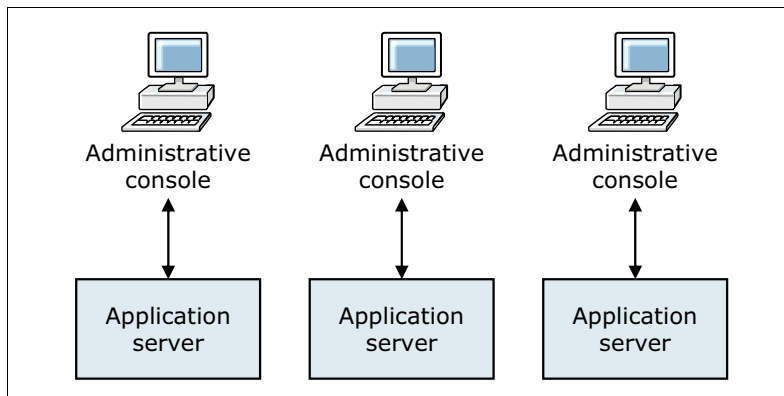


Figure 3 Stand-alone application server environment

The use of an *administrative agent* provides enhanced management capabilities for stand-alone application servers. All configurations that are 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 (Figure 4 on page 10). Using an administrative agent can reduce the redundancy and the associated administration footprint as the number of application server instances increases.

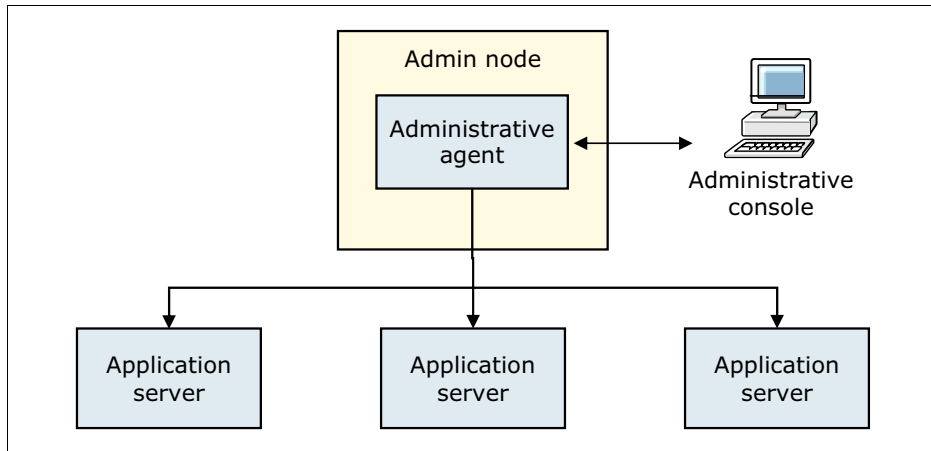


Figure 4 Administrative agent in a stand-alone configuration

Administering distributed application server environments

In Network Deployment and WebSphere Application Server for z/OS environments, a cell can consist of multiple nodes and node groups, which are all administered from a single point, the *deployment manager* (using GUI or scripting), as illustrated in Figure 5.

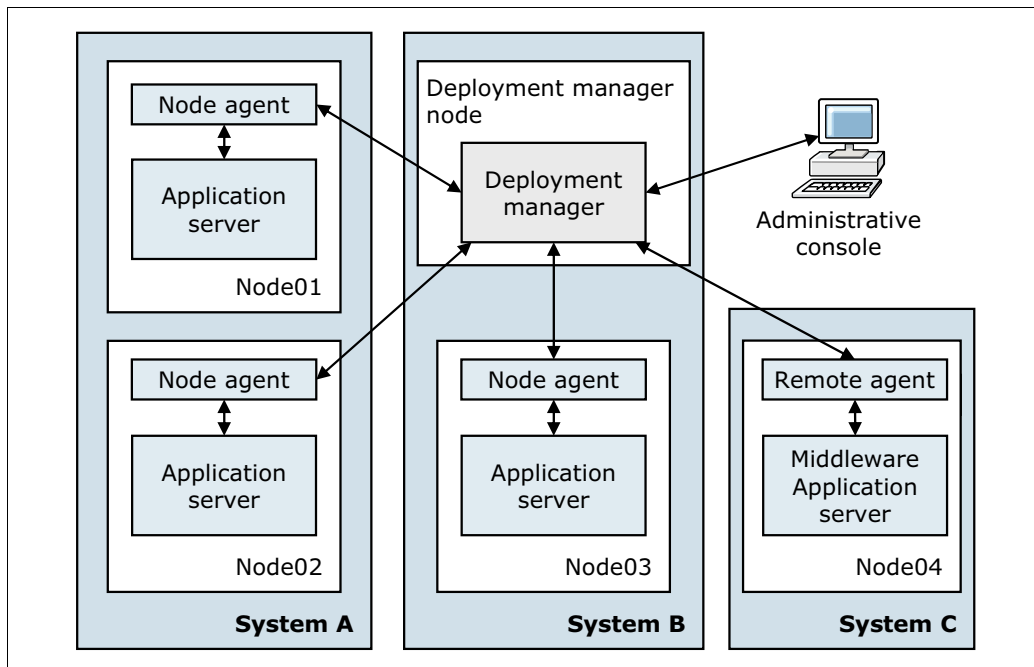


Figure 5 Network Deployment environment

Cells can contain nodes that are running any of the supported operating systems or platforms, including spanning z/OS sysplex environments and other operating systems. For

example, z/OS nodes, Linux nodes, UNIX nodes, and Windows system nodes can exist in the same WebSphere Application Server cell.

The deployment manager communicates with the *node agents* of the cell to manage the applications servers within each node. A node is federated into the cell and can be managed only by the deployment manager that is managing that cell. The node agent is an administrative server that is running in the node. It monitors the application servers on that node and routes administrative requests from the deployment manager to those application servers.

The master repository is managed by the deployment manager and regularly synchronized with local copies that are held on each of the nodes. The application server continues to run even if the deployment manager is not available. If this situation occurs, any configuration changes are synchronized when the connection with the deployment manager is reestablished.

Administration tools

An enterprise must be able to efficiently and effectively manage the WebSphere Application Server environment. The combination of administrative tools that you employ depends on the size and complexity of the runtime environment. Automation and scripting can also be valuable in situations where the same tasks must be performed multiple times in a consistent and repeatable fashion.

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

WebSphere Application Server V8.5 provides the following administrative tools that you can use to configure and manage a runtime environment:

- ▶ WebSphere scripting client (**wsadmin**)

The **wsadmin** client provides a non-graphical scripting and interactive interface that you can use to administer WebSphere Application Server from a command-line prompt. By connecting the client to an application server instance, you can execute commands using either the Jython or Jacl scripting languages.

- ▶ Administrative console

The administrative console is a browser-based client that allows administrators to monitor, update, and configure the WebSphere Application Server environment. Unfederated application servers, administrative agents, deployment managers, and job managers can have their own administrative consoles.

For actions run in the administrative console, you can use command assistance to view the **wsadmin** scripting commands in Jython for the last action that was run. This assistance provides a way to develop automation using **wsadmin** scripting by first stepping through the tasks that you need in the administrative console.

- ▶ Command-line utilities

Administrative utilities can help manage the environment from the command line. These utilities include the ability to perform common administrative tasks, such as starting and stopping application server instances and backing up the configuration. The utilities are run on and act against the local server or node, including the deployment manager.

► **WebSphere Customization Toolbox**

The WebSphere Customization Toolbox includes the following tools for customizing various parts of the WebSphere Application Server environment:

- You can use the Web Server Plug-ins Configuration Tool to configure web server plug-ins for any operating system on which the WebSphere Customization Toolbox can be installed.
- You can use the WebSphere Customization Toolbox command-line utility to start the command-line version of the Plug-ins Configuration Tool (PCT).
- The Profile Management Tool (z/OS only) can be started on an Intel-based Windows or Linux operating system to generate jobs and instructions for creating profiles for WebSphere Application Server on z/OS systems.
- The z/OS Migration Management Tool generates definitions for migrating WebSphere Application Server for z/OS profiles from an Intel-based Windows or Linux operating system.

► **Job manager**

You can use the job manager to manage multiple WebSphere Application Server domains (multiple deployment managers and administrative agents) through a single administration interface. It also provides a centralized interface for asynchronous job submissions.

You can also use the job manager to package the Liberty profile runtime environments, configurations, and applications to distribute and deploy a Liberty profile server and applications, and to start embedded profile packages.

► **Centralized installation manager**

You can use the centralized installation manager to consolidate and simplify the required steps in installing and applying maintenance. With the centralized installation manager, you can install, update, uninstall, and apply maintenance to WebSphere Application Server instances remotely.

You access centralized installation manager functions through the job manager or deployment manager. Because the functions are implemented as jobs, the process also supports job scheduling. Using the centralized installation manager with the job manager, you can manage multiple product offerings in an agentless manner for multiple cells.

► **Administrative applications**

With WebSphere Application Server Version V8.5, you can develop custom Java applications using Java Management Extensions (JMX) to perform any of the administrative features of the WebSphere Application Server administrative tools. For example, you can prepare, install, uninstall, edit, and update applications through programming.

Using a job manager

The *job manager* is available with WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS. It provides centralized management capabilities for multiple stand-alone application servers, administrative agents, Liberty servers, and deployment managers. From a daily task perspective, the job manager can execute daily tasks in one step for multiple installations, including tasks such as starting and stopping servers and distributing and deploying applications. These jobs can be submitted and executed asynchronously to application server environments and topologies and to any number of servers over a geographically dispersed area.

Figure 6 illustrates the job manager architecture.

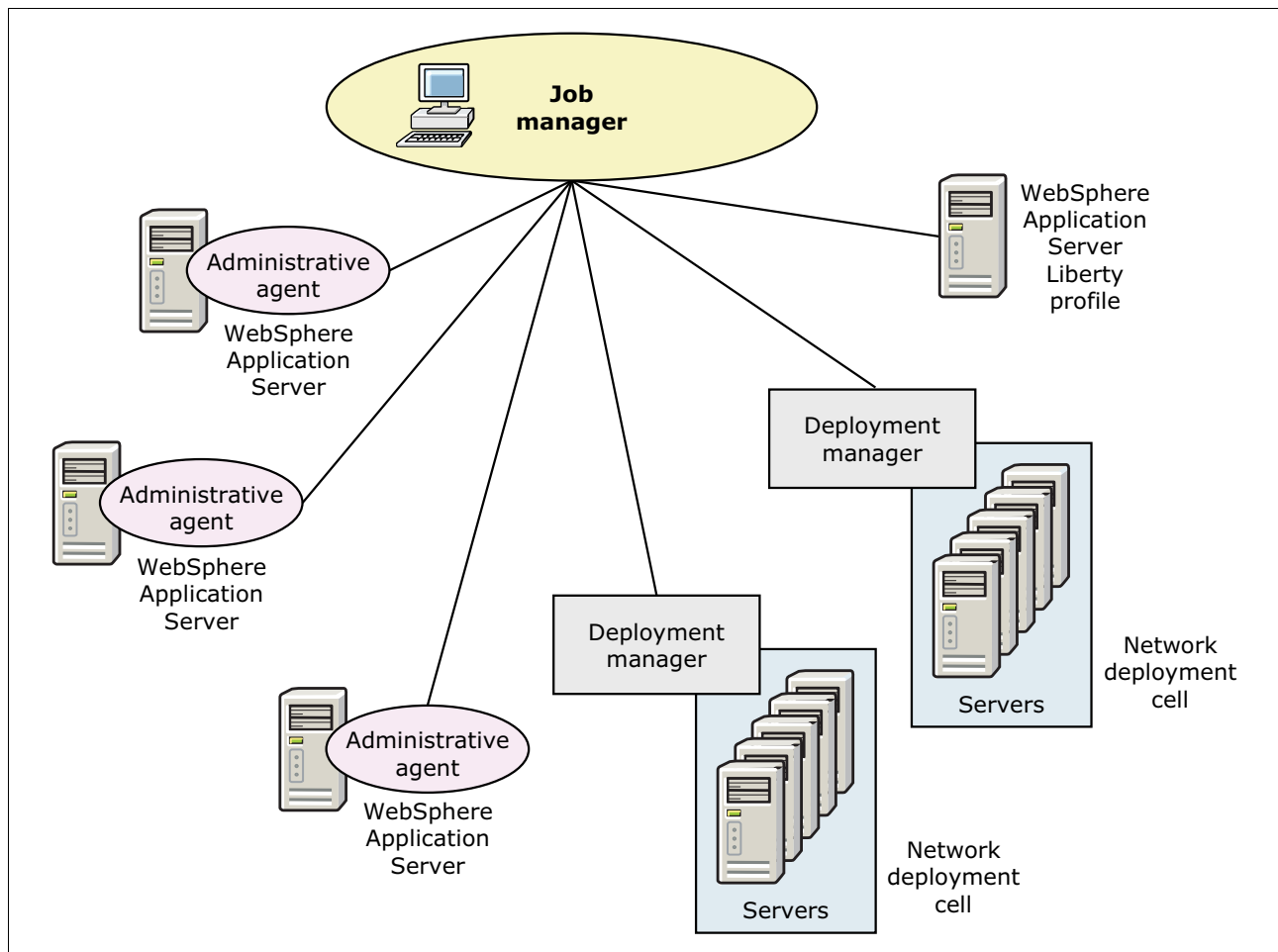


Figure 6 Job manager architecture

From an installation perspective, using a job manager allows for enhanced multiple node installation options for your environment by submitting centralized installation manager jobs to multiple cells and server topologies.

The job scheduler provides all job management functions, such as submit, cancel, and restart. It maintains a history of all jobs, including jobs waiting to run, jobs that are running, and jobs that have already run. The job scheduler is hosted in a WebSphere Application Server or cluster in a WebSphere Network Deployment environment.

Intelligent management

Intelligent Management features extend the quality of service provided by your middleware environment. Configurable operational policies govern the performance and health of your applications. Total cost of ownership is decreased through server consolidation and less administrative effort, and you experience lower response times and increased availability. In short, you experience the benefits of an autonomic middleware environment that is self-configuring, self-protecting, self-healing, and self-optimizing.

A key component of Intelligent Management is the On Demand Router (ODR). The ODR is an intelligent Java-based HTTP proxy server and SIP proxy server built on the WebSphere run time that can manage the flow of requests into both WebSphere and non WebSphere

environments. The ODR is asynchronous, high performing, scalable, and can be clustered for high availability.

(New in V8.5.5) Prior to V8.5.5, the ODR can only be implemented as a separate server placed in the topology between the web server and the application servers. Requests that arrive at the web server are forwarded to the ODR by the WebSphere web server plug-in. The ODR then sends the request to the appropriate server in a dynamic cluster based on current workload conditions and service policies.

With V8.5.5, ODR functionality can also be implemented as web server plug-in, referred to as an *ODR plug-in*, thus eliminating the additional ODR tier and simplifying the topology.

Intelligent Management includes the following primary features:

- ▶ *Intelligent routing* improves business results by ensuring priority is given to business critical applications. Requests to applications are prioritized and routed based on administrator-defined rules.
- ▶ *Health management* allows you to specify conditions to automatically watch for and corrective actions to take when the conditions are observed. You can monitor the status of your application servers, sense problem areas, and then respond to these problem areas before an outage occurs. The health monitoring and management subsystem continuously monitors the operation of servers against user-defined health policies. It detects functional degradation that is related to user application malfunctions.
- ▶ *Application edition management* allows you to roll out new versions of applications without experiencing downtime for a maintenance window. You can manage interruption-free production application deployments by using this feature. You can validate a new edition of an application in your production environment without affecting users, and upgrade your applications without incurring outages to your users. And you can run multiple editions of a single application concurrently, directing different users to different editions.
- ▶ *Performance management* provides a self-optimizing middleware infrastructure. Dynamic clusters automatically scale up and down the number of running cluster members as needed to meet response time goals for users. You can take advantage of overload protection to limit the rate at which the on-demand router forwards traffic to application servers. Doing so helps prevent heap exhaustion, processor exhaustion, or both from occurring.

All of these capabilities together allow you to extend qualities of service through autonomic computing. These capabilities are called *dynamic operations*, which are the core functions that provide application infrastructure virtualization.

ODR plug-in support: Not every ODR server feature is implemented by the plug-in. The plug-in does not support load balancing or failover across cells; there is no CPU or memory load protection; it does not queue or reorder requests based on service policies; and it does not support a highly available deployment manager topology.

The Intelligent Management feature provides full support for those servers that are known as *complete lifecycle servers*. This term includes any server that the WebSphere Application Server environment can instantiate or create. These server types include WebSphere Application Server types such as application servers, generic servers, web servers, and proxy servers.

Intelligent Management functionality also provides support for a range of middleware servers. The term *assisted lifecycle server* refers to servers that you define to WebSphere Application Server by using templates to create representations of the servers in the administrative console. However, these servers still exist within the administrative domain of their respective

middleware platform. You add them as generic servers to the deployment manager capabilities. You can control the servers operationally, monitor and view server health and performance, and configure the administrative console to display log files and configuration files for these servers. This support includes the following middleware servers:

- ▶ Apache Tomcat (deprecated feature)
- ▶ JBoss Application Server (deprecated feature)
- ▶ Custom HTTP servers
- ▶ BEA WebLogic Server (deprecated feature)
- ▶ WebSphere Application Server Community Edition
- ▶ Apache HTTP Server
- ▶ External WebSphere application server (deprecated feature)

Workload management

Workload management is the concept of sharing requests for multiple instances of a resource. Workload management techniques are implemented expressly to provide scalability and availability within a system. These techniques allow the system to serve more concurrent requests.

Workload management encompasses the following main concepts:

- ▶ *Load balancing* is the ability to send the requests to alternative instances of a resource. Workload management allows for better use of resources by distributing loads evenly. Components that are overloaded, and therefore a potential bottleneck, can be routed around with workload management algorithms. Workload management techniques also provide higher resiliency by routing requests around failed components to duplicate copies of that resource.
- ▶ *Affinity* is the ability to route concurrent requests to the same component that served the first request. A web client can establish session affinity for requests to be routed to a particular application server.

On WebSphere Application Server installed on distributed platforms, workload management across application servers is achieved through the use of static or dynamic clustering along with a workload balancing mechanism at the web server tier. Requests for applications are shared across one or more application servers, where each server is running a copy of the application. In more complex topologies, workload management is embedded in load balancing technologies that can be used in front of web servers to balance the flow of incoming requests across the web servers.

Workload Manager (WLM) for WebSphere Application Server for z/OS works differently from the Workload Manager for distributed platforms. With z/OS, the workload management structure for incoming requests is handled by the workload management subsystem features of z/OS. You can define business-oriented rules that classify incoming requests and that assign SLA types of performance goals. This definition is done at transaction-level granularity, compared to server-level granularity with distributed workload management. The system then assigns resources automatically in terms of processor, memory, and I/O to try to achieve these goals.

Balancing work across application servers

Workload management is a WebSphere Application Server facility that provides load balancing and affinity between nodes in a clustered environment. Workload management can be an important facet of performance. WebSphere Application Server uses workload management to send requests for an application to alternative members of the cluster where the application is running.

There are two types of clustering mechanisms to choose from: static clusters and dynamic clusters.

Static clustering

A *cluster* is a collection of servers that are managed together to provide both workload balancing and high availability. Clusters are created with a specific number of application servers. An identical set of applications run on each application server in the cluster. A plug-in in the web server balances incoming requests across the application servers, using affinity to a particular server for multiple requests in a session. 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 the same system or on different systems.

WebSphere Application Server Network Deployment or WebSphere Application Server for z/OS is required for clustering.

Figure 7 shows an example of a cluster that has four application server members. The cluster uses multiple members inside one operating system image (on one system) that are spread over multiple physical systems. This configuration provides a mix of failover and performance.

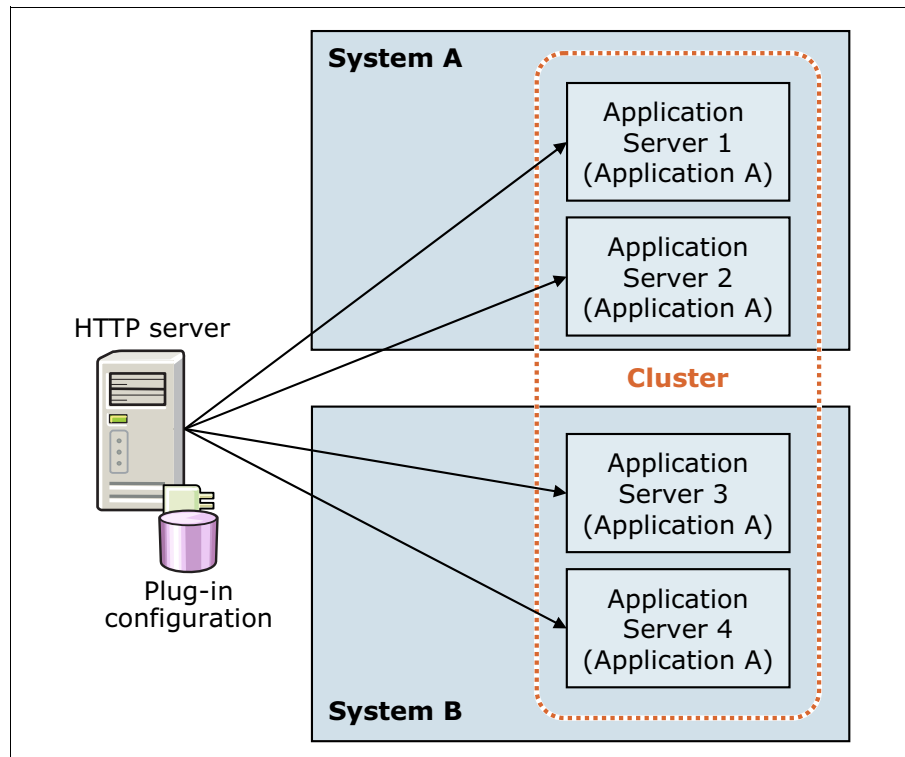


Figure 7 Workload management with a static cluster

Dynamic operations

WebSphere Application Server V8.5 provides integrated support for dynamic clusters as part of the Intelligent Management dynamic operations features. A *dynamic cluster* is an application deployment target that can expand and contract depending on the workload in the environment. Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload manager, to maximize the use of computing resources. A dynamic cluster uses weights and workload management. This management is used to balance the workloads of its cluster members dynamically, and is based on performance information that is collected from the cluster members.

Dynamic workload management is a feature of the ODR that applies the same principles as workload management, such as routing based on a weight system, which establishes a prioritized routing system. With dynamic workload management, the system can dynamically modify the weights to stay current with the business goals, as compared to manually setting static weights in workload management. It also balances requests across the available nodes to regulate response times.

Figure 8 shows a dynamic cluster in WebSphere Application Server V8.5. Note that the HTTP server uses a web server plug-in to route requests to the ODR. The ODR distributes requests across the servers in the dynamic cluster based on workload and other criteria.

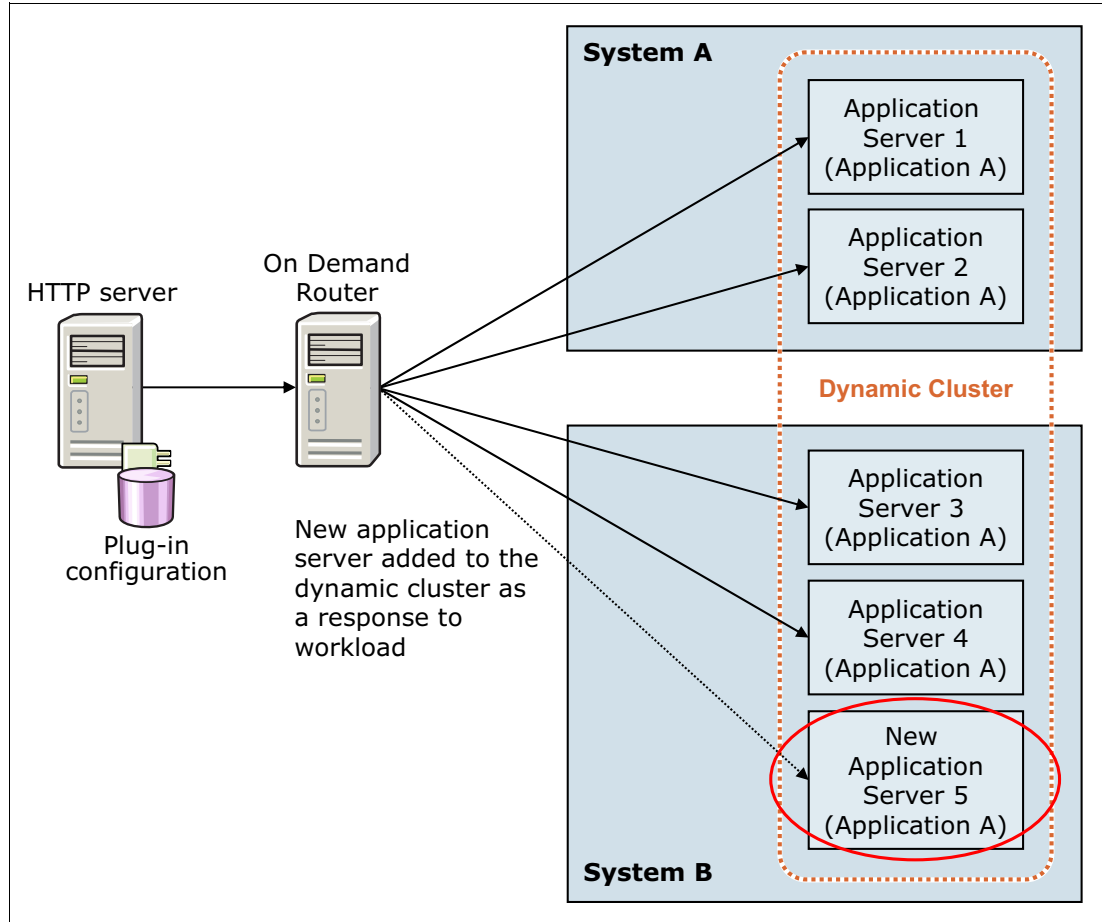


Figure 8 Workload management with a dynamic cluster in V8.5

Figure 9 shows the same dynamic cluster in WebSphere Application Server V8.5.5. The ODR function in this topology has been moved into the HTTP server using the ODR plug-in.

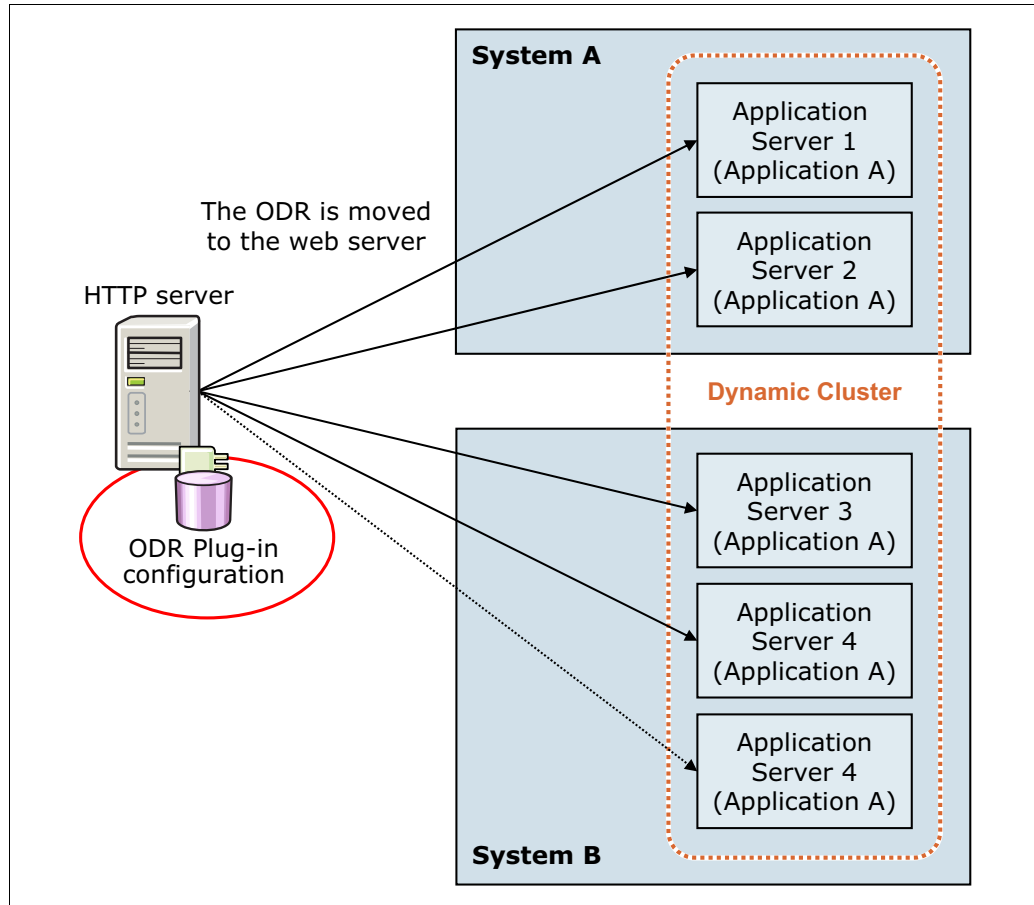


Figure 9 Workload management with the ODR plug-in

Balancing work across HTTP servers

An IP sprayer, like the Load Balancer for IPV4 and IPV6 included with the Edge Components or a network appliance, can perform load balancing and workload management functionality for incoming web traffic. There are several routing techniques, including static and dynamic weighting. In static weighting, as the name implies, each HTTP server being handled by the IP sprayer has a weight and the IP sprayer routes requests based on this weight. It does not take into account the workload of the HTTP server itself. Dynamic weighting, alternatively, calculates the load of each HTTP server and dynamically routes requests to a server that is not as busy.

(New in V8.5.5) The Load Balancer for IPV4 and IPV6 has been enhanced in V8.5.5 to improve flexibility in configuration and to improve workload balancing. The new features include the following items:

- ▶ The load balancer can now be run on the same machine as the servers it is balancing. This feature is supported on Linux and IBM AIX® only.
- ▶ The Content Based Routing (CBR) component has been added to enable load balancing based on the content of client requests, for example, the URI.
- ▶ The Site Selector component has been added to enable balancing load using domain name service (DNS) round robin, or using a user-provided algorithm.

- ▶ Network Address Translation (NAT) has been added, removing the limitation that back-end servers are on the same locally attached network.

Balancing work across DMZ proxy servers

As with HTTP servers, you can use an IP sprayer component, such as the Load Balancer for IPV4 and IPV6 included with the Edge Components, to perform load balancing and workload management functionality for incoming web traffic. In addition, the DMZ proxy server provides workload management capabilities for applications that are running in an application server.

High availability

High availability is also known as *resiliency*. High availability is the ability of a system to tolerate a number of failures and remain operational. It is achieved by adding redundancy in the infrastructure to support failures. It is critical that your infrastructure continues to respond to client requests regardless of the circumstances, and that you remove single points of failure. Planning for a highly available system takes planning across all components of your infrastructure, because the overall infrastructure is only available when all of the components are available. As part of the planning, you must define the level of high availability that is needed in the infrastructure.

WebSphere Application Server Network Deployment uses a high availability manager (HA manager) to address single points of failure within the WebSphere Application Server environment. An HA 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 HA 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 that allow the HA manager instance to detect when other members of the core group start, stop, or fail.

Within a core group, HA 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.

The HA 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 component uses this mechanism to build and maintain routing table information. Routing tables built and maintained using this mechanism are highly available.

Application server clusters provide various levels of high availability. Clustering multiple application servers on the same node provides availability in the event of a failure of an application server. Clustering application servers across different systems provides high availability in the event of an application server or system failure. Each member of a cluster must belong to the same cell, and cannot belong to more than one cluster. Cluster members are required to have identical application components but can be sized differently. The cluster is a logical view of the application servers and does not correspond to a process.

The workload management is responsible for sharing the workload between the cluster members. The Data Replication Service (DRS) that is provided with WebSphere Application Server is used to replicate stateful EJB sessions, HTTP session data, and dynamic cache information among cluster members. When DRS is configured for memory-to-memory

replication, the transport channels defined for the HA managers are used to pass this data among the cluster members.

High availability of application servers is not only about failover; it also means ensuring that you have adequate resources to meet changing demands. With the dynamic clustering capabilities, applications are mapped to dynamic clusters that are spread throughout hardware pools. Each node in the dynamic cluster can run one or more instances of an application server that runs that cluster's applications. The dynamic cluster can expand and contract to adjust to workload requirements.

With dynamic operations, you can also use the ODR to set up a highly available deployment manager using a hot-standby model for availability. For example, you can have two or more deployment managers, where one deployment manager is active (known as the *primary* deployment manager) and one or more deployment managers are backup managers in standby mode. The primary deployment manager hosts the cell's administrative function, and if the active manager is stopped or fails, a standby manager takes over and is designated as the new active deployment manager. This feature is available in the ODR server only, not in the ODR plug-in.

Although the deployment manager process is not considered a single point of failure (SPOF) for applications and application server functionality, using a highly available deployment manager eliminates the deployment manager as a SPOF for cell administration. This type of high availability is especially important in environments that have significant reliance on automated operations, including application deployment and server monitoring.

The WebSphere Application Server *service integration bus* (bus) provides high availability to the messaging system process. The messaging engines of the bus can be clustered, providing failover capabilities if a messaging engine fails. To achieve a seamless failover, the queue information and message data must be stored in a shared location that is accessible by all members of the cluster. This shared location can be an external database or shared disk environment.

Health management

Another important key to business success is the ability to autonomically manage the health of your servers and quickly respond to potential issues, preferably before they impact your environment. The Intelligent Management capabilities in WebSphere Application Server provides several health management features to monitor the status of application servers to sense and respond to problem areas before an outage occurs, similar to preventive medicine.

You can manage the health of an application-serving environment with a policy-driven approach that enables specific actions to occur when monitored criteria is met. For example, when memory usage exceeds a percentage of the heap size for a specified time, health actions can run to correct the situation.

Health monitoring can help with unexpected issues and unanticipated problems in your environment. It can help you bypass problems that will otherwise disrupt operations and affect the performance and availability of your applications. You can automatically detect and handle application health problems without requiring administrator intervention, or in a supervisor mode that requires administrator approval before an autonomic action is performed. Health monitoring intelligently handles health issues in a way that maintains continuous availability, and allows you to configure how the system is to handle individual applications and requests.

You can have the system monitor for various conditions, such as high response timeout rates, excessive response times, or high memory usage, and proactively execute certain actions in an attempt to correct these issues.

Messaging

Generically, the term *messaging* describes communication or the exchange of information between two or more interested parties. Messaging can take many shapes and forms. For example, sending a fax message from one point to another is point-to-point messaging. Sending a single message to many destinations, such as sending an email to a mailing list, is an example of the publish/subscribe messaging concept. Messaging is a facility for exchanging data between applications and clients of different types. It is also an excellent tool for communication between heterogeneous platforms.

WebSphere Application Server implements a powerful and flexible messaging platform within the WebSphere Application Server environment called the service integration bus. WebSphere Application Server applications invoke asynchronous messaging services, using the Java Message Service (JMS) application programming interface (API) to interface with a messaging provider. The messaging provider can be the WebSphere MQ messaging provider, the default messaging provider (bus), or a third-party messaging provider.

Applications can use one of the following styles of asynchronous messaging:

- ▶ *Point-to-point* applications typically use queues to pass messages to each other. An application sends a message to another application by identifying a destination queue. The underlying messaging and queuing system receives the message from the sending application and routes the message to its destination queue, where the receiving application retrieves it.
- ▶ In *publish/subscribe* messaging, the subscriber is the consumer of the information and specifies the topics of interest by submitting a subscription request. The publisher supplies information in the form of messages, which contain both a topic and the actual information. A message broker sits between the subscriber and the publisher. As it receives published messages, the broker forwards those messages to the subscribers who requested messages about the topic given in each individual message.

Service integration

Service integration is a set of technologies that provides asynchronous messaging services. In asynchronous messaging, producing applications do not send messages directly to consuming applications. Instead, producing applications send messages to *destinations*. Consuming applications then receive messages from these destinations. A producing application can send a message and continue processing without waiting until a consuming application receives the message.

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. A service integration bus is a group of one or more *bus members* in a WebSphere Application Server cell that cooperate to provide asynchronous messaging services.

A cell *requires* only one bus, but a cell can contain any number of buses. A messaging engine is a component that is responsible for processing messages, sending and receiving requests, and hosting destinations. A destination is defined within a bus and represents a logical address to which applications can attach as message producers, consumers, or both.

Service integration has the following types of bus destinations, each with a different purpose:

- ▶ A *queue* destination is used for point-to-point messaging.
- ▶ A *topic space* destination is used for publish/subscribe messaging.
- ▶ A *foreign destination* is a destination that is defined in another bus (called a *foreign bus*) and that is also used in point-to-point messaging.
- ▶ An *alias* destination is an alternative name for either a queue destination or a topic space destination.

Service mapping

(*New in V8.5.5*) The use of services depends on the ability of the service client to locate and communicate with the service provider. Changes to either the location of the service or to the interface can impact the applications that use the service. The new service mapping feature is designed to shield applications from minor changes in the services they use.

This feature gives administrators the ability to define a mapping service that can intercept service client invocations bound for a particular service. The mapping service can determine which service location the message is to be routed to, which operation on the service provider is to be invoked, and how the fields in the client and server messages are to be mapped to each other. Administrators can control to which service interactions the service mapping applies.

To implement this function, a service map is created using a graphical interface in Rational Application Developer, and then deployed to WebSphere Application Server. The administrator then creates a mapping service using the deployed service map.

The administrator can also configure a mapping service to publish JMS publish/subscribe events for each service request and response. A JMS topic subscriber can be created to consume these events and, additionally, the IBM Integration Bus product has built-in support for consuming and processing these events.

Security

WebSphere Application Server provides a security infrastructure and mechanisms to protect sensitive resources and to address enterprise end-to-end security requirements. You can secure the environment at various levels, from the physical security of the location where the server hardware is, to the security within the application itself.

WebSphere Application Server security includes the following levels:

- ▶ Java virtual machine (JVM) security

The JVM provides a set of standards-based security services for Java applications and an installation layer between Java applications and the various services within the operating system. It provides an isolated environment to the Java application that is running in it, which in this case is WebSphere Application Server. In addition, the JVM protects memory from unrestricted access, creates exceptions when errors occur within a thread, and defines array types.

- ▶ Java 2 security

The Java 2 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 2 uses security policy files, which can control the access to the resources

by applications. A WebSphere Application Server application has its own policy files, so that it can use files and directories on the host operating system.

- ▶ Java EE security API

The Java EE standard API describes methods with which the application can obtain the logged-in user's name and role membership. WebSphere Application Server never returns the password of any user using the API methods. The Java EE security policy describes how application resources are accessed. The developer, when creating the application, has no information about real users of the application. Instead, the developer defines *user roles*, and during development the user roles are mapped to access rights. The rule set is stored in the descriptor files of the application. Then, when the application is deployed, the deployer is responsible for mapping users and groups to the security roles.

- ▶ CSIv2 security

CSIv2 is an IOP-based, three-tiered security protocol that is developed by the Object Management Group (OMG). This protocol provides message protection, interoperable authentication, and delegation. Any calls made among secure Object Request Brokers (ORBs) are invoked over the CSIv2 security protocol, which sets up the security context and the necessary quality of protection. After the session is established, the call is passed up to the enterprise bean layer.

- ▶ WebSphere security

WebSphere security enforces security policies and services regarding access to its resources. It covers a wide range of features. It begins with the administrator user management in the administrative console, controlling which administrative user is allowed to do what on the administrative console. Administrative security is enabled by default.

WebSphere security provides security services for applications that are running in WebSphere Application Server. WebSphere Application Server supports the J2EE security standards and provides a means for applications to focus on business logic. WebSphere security handles the authentication, authorization, secure communications, and security auditing needs of the applications.

The WebSphere Application Server security infrastructure provides services. The services are integrated with the underlying operating system, the runtime environment, and other servers and server components. The WebSphere Application Server security infrastructure includes the following key security areas:

- ▶ Authentication
- ▶ Authorization
- ▶ Key and certificate management
- ▶ Auditing
- ▶ Security domains and user registries

Authentication

Authentication is the process of confirming a user or system identity. The authentication mechanism in WebSphere Application Server uses a user registry to perform validation. 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.

WebSphere Application Server supports the following types of web login authentication mechanisms:

- ▶ Basic authentication
- ▶ Certificate-based authentication
- ▶ Form-based authentication

WebSphere Application Server supports the following authentication mechanisms:

- ▶ Lightweight Third Party Authentication (LTPA)
- ▶ Kerberos
- ▶ Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO)
- ▶ Rivest Shamir Adleman (RSA) token authentication
- ▶ Web services security Security Assertion Markup Language (SAML) Token Profile
- ▶ SAML web SSO post-binding profile

Authorization

Authorization is the process of checking whether a user has the privileges necessary to access a requested resource. WebSphere Application Server differentiates between two kinds of authorization, according to the following types of users:

- ▶ Administrative user authorization

The administrative security in WebSphere Application Server controls access to the configuration and management interfaces. Fine-grained administrative security can grant access to each user role per resource instance, instead of granting access to all of the resources in the cell. This allows for better separation of administrative duties.

- ▶ Application user authorization

The Java EE specification defines the building blocks and elements of a Java EE application. The specification provides the details about security that are related to several elements. The application maps user roles with resource access rights during development. During application deployment, the administrator or deployer maps the user roles to real users and groups.

Key and certificate management

Secure Sockets Layer (SSL) is the industry standard for data interchange encryption between clients and servers. In WebSphere Application Server, Java Secure Socket Extension (JSSE) is used to handle the handshake negotiation and protection capabilities that are provided by SSL to ensure that secure connectivity exists across most protocols. SSL security can be used for establishing communications inbound to and outbound from an endpoint.

To establish secure communications, you must specify a certificate and an SSL configuration for the endpoint. WebSphere Application Server differentiates the following kinds of communication:

- ▶ Internal management communication

The WebSphere Application Server uses SSL to communicate within the cell among the nodes. It maintains certificates for each node in the cell. When a new profile is created, including the deployment manager profile, a new unique chained certificate is also generated for the profile. This certificate consists of a signer certificate, which has a 15-year expiration time, and server personal certificates, which have a one-year expiration time by default. WebSphere Application Server has its own, built-in mini-certificate authority (CA) with which it signs the certificates in the cell. Alternatively, you can use your own certificate settings and import your certificates to override settings.

- ▶ External service communication

The following components are some of the external connections that require SSL encryption:

- JDBC database connection
- LDAP directory protocol connection
- Messages channels
- Web services communication

You can create or manage certificates using the administrative console. WebSphere Application Server provides mechanisms for creating and managing CA clients and keystores and for creating self-signed certificates and certificate authority requests. Keystores in WebSphere Application Server profiles hold personal certificates, and truststores store signer certificates from other servers with which it is communicating.

All certificates have an expiration time. The server personal certificate has a default expiration time of one year, and the signer certification has a default expiration time of 15 years. You can replace certificates manually, but the more effective method is to let the application server replace the certificates automatically with the help of the built-in expiration manager.

Auditing

With the *audit service*, WebSphere Application Server can log significant system and application events and then you can review these long-term logs later.

Security auditing has the following primary goals:

- ▶ Confirm the effectiveness and integrity of the existing security configuration (accountability and compliance with policies and laws) with the most common aspect of reviewing who performed what operation
- ▶ Identify areas where improvement to the security configuration might be needed (vulnerability analysis)

During run time, all code (except the Java EE application code) is considered to be trusted. Each time a Java EE application accesses a secured resource, any internal application server process with an audit point included can be recorded as an auditable event.

WebSphere Application Server auditing works through event logging. All security-related events are filtered with an audit filter and with an event outcome filter. The captured events, which go through both filters, are logged in to the audit log.

The repository checkpoints service improves administration configuration changes. The repository checkpoints service helps an administrator use checkpoints to track changes made to the application server configuration. Repository checkpoints represent saved images of the repository before configuration changes are made. To track those changes, a new type of event is added to the security auditing component. This event is emitted when a checkpoint is saved in the extended repository service.

WebSphere Application Server has a built-in auditor administrative role. Only the administrators in the auditor role can change settings related to the audit subsystem and review the audit logs. By default, the primary administrative user is a member of the auditor administrative role, but you can remove this role from this user. For example, you can create a separate auditor user role and user principal, and then assign these roles to a security team member for WebSphere Application Server. With this approach, only appropriate users have access to the audit data, and the audit subsystem and console administrator users cannot tamper with the audit content.

Security domains and user registries

WebSphere Application Server provides the capability to use security domains. Each security domain can have its own, separately configured Virtual Member Manager (VMM) instance. WebSphere Application Server provides the following options and combinations to select the best user registry setting for an application environment:

- ▶ Security domains

WebSphere *security domains* provide the flexibility to use several security configurations within a single WebSphere Application Server cell. With security domains, you can configure several security attributes, such as the user registry, for various applications in the same cell. The global security configuration applies to all administrative functions, naming resources, and Mbeans, and is the default security configuration for user applications. You must define a global security configuration before you can create the security domains. If you do not configure security domains, the applications use information from the global 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 and the naming operations in that scope use the global security configuration. Each security domain must be associated with a scope (cell, or specific clusters, servers, and service integration buses) where it is applied.

- ▶ User registries

Information about users and groups is in a *user registry*. In WebSphere Application Server, a user registry authenticates a user. It contains information about users and groups so that security-related functions, including authentication and authorization, can be performed. Although WebSphere Application Server supports several types of user registries, only one user registry can be active in a certain scope.

WebSphere Application Server supports the following types of user registries:

- Local operating system
- Stand-alone 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

Technologies available to applications

The WebSphere Application Server full profile supports multiple technologies and programming models that you can use within enterprise applications. In addition to the programming models and technologies, applications can use either the Java 6 or Java 7 Java runtime environment (JRE).

Java 6 is installed with the product and used by default. WebSphere Application Server supports IBM WebSphere SDK Java Technology Edition Version 7.0 as a pluggable JDK that can be optionally installed and enabled by using the **managesdk** tool. For more information, see the IBM SDK Java Technology Edition Version 7 Information Center, found at:

<http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp>

The following programming models and technologies are supported in WebSphere Application Server:

- ▶ Java EE applications
- ▶ Portlet applications
- ▶ SIP applications

- ▶ Business-level applications
- ▶ WebSphere Batch applications
- ▶ OSGi applications
- ▶ Communications enabled applications
- ▶ Service Component Architecture applications
- ▶ WebSphere Application Server Web 2.0 and Mobile Toolkit

Java EE applications

The Java EE specification is the standard for developing, deploying, and running enterprise applications. WebSphere Application Server provides full support for the Java EE 6 specification. The Java EE programming model has multiple types of application components:

- ▶ Enterprise beans
- ▶ Servlets and JavaServer Pages (JSP)
- ▶ Application clients

WebSphere Application Server includes the IBM Assembly and Deploy Tools for WebSphere Administration. This tool replaces the previously available IBM Rational Application Developer Assembly and Deploy Tool. A developer can use this tool to accomplish key assembly and deployment needs, including editing deployment artifacts, developing and testing scripts, and deploying and debugging applications. This tool is not intended for general application development.

For information about the Java EE specification, see:

<http://www.oracle.com/java>

Portlet applications

The portlet container in WebSphere Application Server provides the runtime environment for Java Specification Request (JSR) 286-compliant portlets. Portlet applications are intended to be combined with other portlets collectively to create a single page of output. Portlets are packaged in web archive (WAR) files. The portlet run time 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:

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

SIP applications

SIP applications are Java programs that use at least one SIP servlet written to the JSR 116 specification. WebSphere Application Server also supports SIP Servlet Specification 1.1, also referred to as *JSR 289*. 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 is established, communication takes place over the specified transport mechanism, independent of SIP. Examples of application types that use SIP include voice over IP (VOIP), click-to-call, and instant messaging.

In the application server, the web container and SIP container are converged and can share session management, security, and other attributes. In this model, elements of an application, which include SIP servlets, HTTP servlets, and portlets, can interact regardless of the protocol. High availability of these converged applications is possible because of the integration of HTTP and SIP in the base application server.

(New in V8.5.5) WebSphere Application Server V8.5.5 enhances serviceability and troubleshooting capabilities to enable more resilient processing of SIP sessions.

For more information about SIP applications, see the following resources:

- ▶ JSR 289 SIP Servlet API 1.1 Specification
<http://www.jcp.org/aboutJava/communityprocess/final/jsr289/index.html>
- ▶ JSR 116
<http://jcp.org/en/jsr/detail?id=116>
- ▶ RFC3261
<http://www.ietf.org/rfc/rfc3261.txt>

Communications enabled applications

Communications enabled applications (CEA) allow developers to add dynamic web communications to any application or business process. CEA provides Representational State Transfer (REST) and web service interfaces to enable existing applications to take advantage of communication features that involve phone calls and web collaboration.

CEA applications do not require developers to have extensive knowledge of telephony or SIP to implement these applications. CEA capabilities deliver call control, notifications, and interactivity, providing the platform for more complex communications.

Business-level applications

Business-level applications (BLAs), such as those shown in Figure 10, expand the definition of an application beyond the Java EE definition. Typically, a business considers the pieces that make up an application, including both WebSphere and non WebSphere artifacts such as SCA packages, libraries, and proxy filters, as being under a single application definition that is stored in the product configuration repository. Business-level applications do not introduce new programming, runtime, or packaging models. Thus, application business logic or runtime settings do not need to change.

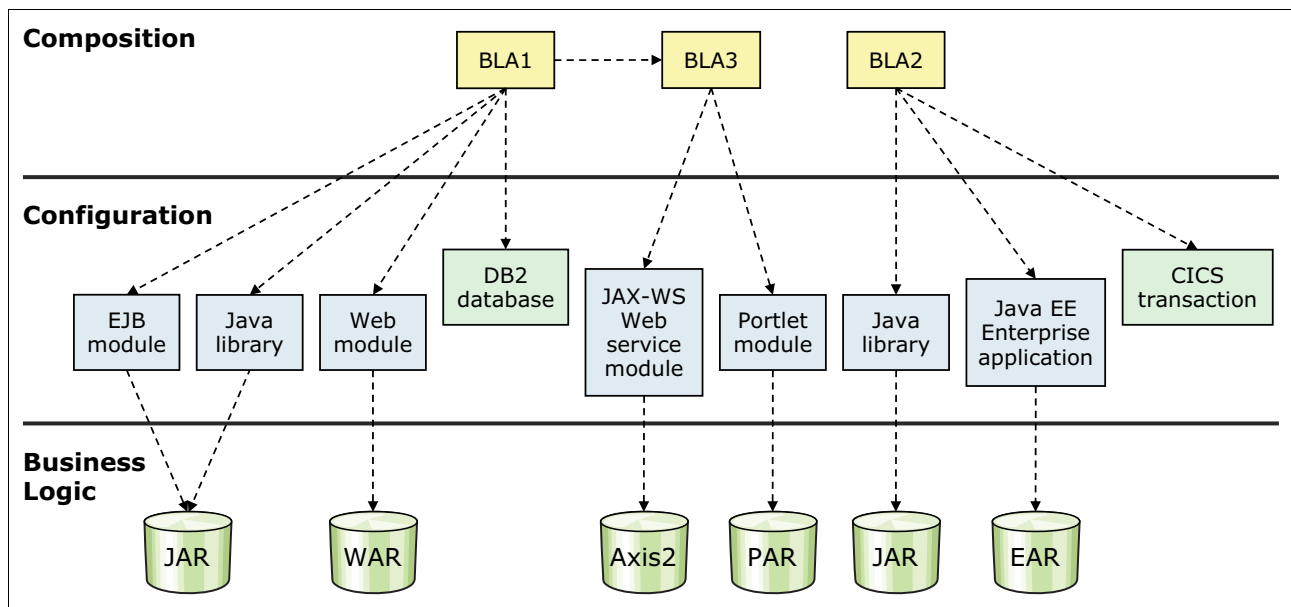


Figure 10 Business-level applications

Business-level applications have the following characteristics:

- ▶ Contain one or more composition units (CUs) that represent the application binary files
- ▶ Define an application that does not represent or contain application binary files

- ▶ Can span more than WebSphere Application Server deployment target run times, such as a proxy server, a web server, and WebSphere Application Server Community Edition
- ▶ Provide installation, distribution, activation, monitor, update, and removal management features for applications
- ▶ Support application service provider (ASP) scenarios by allowing single application binary files to be shared between multiple deployments
- ▶ Align WebSphere applications closer with business rather than with the IT configuration

WebSphere Batch applications

Batch applications can perform long-running bulk transaction processing and compute-intensive work. They run as background jobs, described by a job control language, and are supported by infrastructure components that are aimed to support batch workloads. WebSphere Batch provides both a transactional batch programming model and a compute-intensive programming model. Both models are implemented as Java objects, packaged in an enterprise archive (EAR) file, and deployed into the application server environment. The individual programming models provide details about how the lifecycle of the application and jobs that are submitted to it are managed by WebSphere Application Server.

The XML job control language provides a method to describe the steps that are involved in the batch jobs. This application runs in batch containers that also run at the same time in designated WebSphere Application Server environments. A job scheduler determines the appropriate container to run the job and maintains job histories to support job visibility and control.

OSGi applications

OSGi applications are modular applications that use both Java EE and OSGi technologies. With this design, you can improve control and flexibility by designing and building applications and groups of applications from coherent, multiversion, and reusable OSGi bundles.

WebSphere Application Server support for OSGi applications includes the capability of deploying web applications that use the Java Servlet 3.0 Specification. Additionally, you can update a running application and impact only those bundles that are affected by the update. You can use this method to extend and scale running applications as your business demands it.

OSGi applications allow the composition of isolated enterprise applications using multiple, multiversion bundles that have dynamic lifecycles. Application maintenance and upgrades can be simplified using standard OSGi mechanisms to simultaneously load multiple versions of classes in the same application. Existing web archives (WAR files) can be reused and deployed as OSGi web application bundles.

For more information about OSGi applications, see:

<http://www.osgi.org/About/WhatIsOSGi>

Service Component Architecture

The Service Component Architecture (SCA) programming model supports the principles of service-oriented architecture (SOA) through application flexibility and agility. Service implementation and access method details are moved out of business application logic and into metadata that can be operated on by middleware. As service details or locations change, the application can remain unaffected. SCA helps application developers maximize productivity through focus on solving business problems with application code rather than protocols and locations.

Service compositions can be created by using the following protocols:

- ▶ Plain Object Java Objects (POJOs)
- ▶ EJB 2.1, 3.0, and 3.1
- ▶ OSGi applications
- ▶ Spring components
- ▶ Java Servlets
- ▶ JavaScript for Asynchronous JavaScript and XML (Ajax)

WebSphere Application Server Web 2.0 and Mobile Toolkit

The WebSphere Application Server Web 2.0 and Mobile Toolkit simplifies the addition of Asynchronous JavaScript and XML (Ajax) rich desktop and mobile user interfaces and REST web services to Java web applications. Web 2.0 capabilities, such as Ajax and REST, help application developers create more connected, interactive applications, which result in higher client satisfaction, user productivity, and enhanced decision-making. Mobile Ajax components enable developers to create mobile web applications that run on mobile devices, such as smartphones and tablets.

Serviceability and troubleshooting

After your system is operational, you must be able to determine problems when something goes wrong. WebSphere Application Server includes the following modes of logging for problem determination:

- ▶ Basic logging and tracing

A full profile application server writes system messages into several general-purpose logs. The `SystemOut.log` and `SystemErr.log` files contain system output and error messages. For processes that run native code, `native_stdout.log` and `native_stderr.log` are used.

These logs are plain-text files that can be read using a text editor on the system where the logs are stored. You can also view the logs from the administrative console, including logs for remote systems.

Cross-component tracing can be enabled to augment the log and trace files with correlation information that clarifies which threads and application server processes participated in handling each request.

The IBM service log (`activity.log`) is a special log file written in a binary format. You cannot view the log file directly with a text editor, but you can view the log file using one of the following methods:

- The Log Analyzer tool provides interactive viewing and analysis capabilities that help identify problems.
 - The Showlog tool can be used to convert the contents of the service log to a text format.
- ▶ Binary logging, implemented using High Performance Extensible Logging (HPEL), stores log and trace data in a proprietary binary format. Log and trace performance is enhanced with HPEL because of the following factors:
 - Log and trace events are stored only in one place and not redundantly in several locations. Log events, `System.out`, and `System.err` information is stored in the log data repository. Trace events are stored in the trace data repository.
 - Repositories are not shared across processes. Each server process has its own repositories and text log file. The server environment, therefore, does not need to synchronize with other processes when writing data.

- Data is not formatted until it is viewed. Log and trace data is stored in a proprietary binary format in the repositories rather than being formatted at run time. The log and trace data is not formatted until it is viewed through the LogViewer tool.
- Log and trace data is buffered before being written to disk.

The HPEL LogViewer is a simple command-line tool to enable HPEL users to work with the log and trace data repositories. The LogViewer provides filtering and formatting options that make finding important content in the log and trace data repositories easy. For example, a user can filter any errors or warnings and then filter all log and trace entries that occurred within 10 seconds of a key error message. This filtering can be done on the same thread.

The full profile also provides extensive troubleshooting aids, including a hung thread detection policy, a memory leak policy, and the ability to collect Java and core dump files.

Liberty profile concepts

The Liberty profile is a dynamic and composable profile of WebSphere Application Server. It enables WebSphere Application Server to provision only the features that are required by the application (or set of applications) that are deployed to the server. For example, if an application requires simply a servlet engine, a Liberty profile server can be configured to start the WebSphere Application Server kernel, the HTTP transport, and the web container. The Liberty profile server therefore starts quickly and has a small footprint.

The Liberty profile provides a simplified and lightweight development and application-serving environment that is optimized for developer and operational productivity. This profile is intended for use as a development or production environment for running web applications that do not require a full Java EE stack. The Liberty profile provides enterprise qualities of service, including security and transaction integrity.

The Liberty profile includes the following key features:

- ▶ A dynamic and flexible run time to load only what the application needs.
- ▶ A quick startup time (under 5 seconds with simple web applications).
- ▶ A simplified configuration that uses a single configuration file or modular configuration.
- ▶ Support for deploying applications developed in the Liberty profile to run in the full profile.
- ▶ Support of web applications, OSGi applications, and Java Persistence API.
- ▶ *(New in V8.5.5)* Support for web services.
- ▶ *(New in V8.5.5)* Support for Java Message Service (JMS).
- ▶ A secure server environment. User registry options include single or *(New in V8.5.5)* federated LDAP registries, role-based authorization, SSL, single sign-on, custom login modules, *(New in V8.5.5)* OAuth support, and more.
- ▶ *(New in V8.5.5)* Support for the use of MongoDB, a NoSQL database system. The server and client drivers are not shipped with WebSphere Application Server.
- ▶ The ability to deploy an application and configured server as a package.
- ▶ *(New in V8.5.5)* The ability to extend the Liberty profile with custom features.
- ▶ *(New in V8.5.5)* The ability to cluster servers for high availability and scalability.
- ▶ *(New in V8.5.5)* Centralized operational management of groups of Liberty servers and of Liberty clusters.

- ▶ Managed, centralized deployment for many nodes of a packaged application and server using the job manager.
- ▶ Availability of WebSphere Application Server Developer Tools as Eclipse plug-ins for broad tool support.
- ▶ Support for z/OS platform native features like System Authorization Facility (SAF), Resource Recovery Services (RRS), and z/OS Workload Manager (WLM).

Because a Liberty profile server is lightweight, it can be packaged easily with applications in a compressed file. This package can be stored, distributed to colleagues, and used to deploy the application to a different location or to another system. It can even be embedded in the product distribution.

Liberty servers

The Liberty profile is built on OSGi technologies. The server process runs as OSGi bundles and comprises a single Java virtual machine (JVM), the Liberty profile kernel, and any number of optional features. The Liberty profile configuration operates from a set of built-in configuration defaults. You can specify only the required changes for your environment by using a simple XML format.

Liberty profile feature support

A functional server is produced by starting the runtime environment with a configuration that includes a list of features that are to be used. *Features* are the units of capability by which the runtime environment is defined and controlled. They are the primary mechanism that makes the server composable. For example, if the servlet feature is specified, the runtime environment operates as a servlet engine. By default, a server runs with no features. You can use the feature manager to add the features that are needed.

In V8.5.5, the support for features has been expanded to complete the Java EE 6 web profile support and to add support for messaging, web services support, MongoDB, and dynamic cache features. Features have also been added to support server clustering and a new administrative structure called *collectives*.

Liberty profile configuration

The Liberty profile runtime environment is designed for easy administration, and it operates from a set of built-in configuration defaults. A Liberty profile server configuration consists of a `server.xml` file, a `bootstrap.properties` file, and any optional files that are included by these two main configuration files.

The `server.xml` file is the primary configuration file for the server. It contains information about the features to be included in the runtime environment, applications deployed into that runtime environment and their data sources, and optional properties such as an override to a configuration default or a trace specification.

The include functionality for the `server.xml` file can be useful in managing multiple Liberty profile environments. The `server.xml` file can point to (include) one or more remote XML files. When you change values in the remote XML file, that change takes effect in all of the Liberty profile runtime environments that include that remote XML file. For example, if an application and its data sources are defined in thousands of Liberty profile runtime environments, an administrator, using the remote include functionality, can change one value for a data source,

and have that change reflected in all of the runtime environments without changing each `server.xml` file individually.

There is no administrative console for the Liberty profile. However, administrators and developers can use the Liberty profile developer tools or a text editor to edit the configuration files.

Liberty profile administration

Typical administration actions include creating, starting, and stopping a server, querying the status of a server, packaging a server, and performing troubleshooting actions such as creating a dump for diagnosis.

Liberty servers are administered using line commands or from the developer tools. On a z/OS platform, you can use IBM MVS™ operator commands to start, stop, or modify the Liberty profile. Liberty servers can also be accessed from JMX clients, such as using the jConsole tool provided in the Java SDK to monitor data.

Liberty servers can be administered individually; or from a job manager (see “Using a job manager” on page 12); or (*New in V8.5.5*) by a collective controller as part of a collective.

Liberty collectives provide a common management domain for Liberty servers. This new structure has been added to the administration options for Liberty in V8.5.5 for operational efficiency and convenience, and to introduce high availability features. The collective controller provides operational access to all servers in the collective. This includes operations to start and stop servers, invoke administrative operations, and perform file transfer in support of configuration changes and application installation.

Liberty servers join the collective by registering with an operational registry and adding the `managedServer` feature to their configuration. All Liberty profiles can be members of a collective, but only Network Deployment or WebSphere Application Server for z/OS provide the support needed to create a collective controller.

Both the job manager and the collective controller provide “agent-less” management of Liberty servers. Either tool allows you to create, update, and remove servers, and to update server configurations and applications. The collective controller, like the job manager, allows you to monitor server status but without having to submit a job. The new collective controller, however, also allows you to cluster Liberty servers for high availability and scalability.

Figure 11 illustrates Liberty servers being managed as a collective.

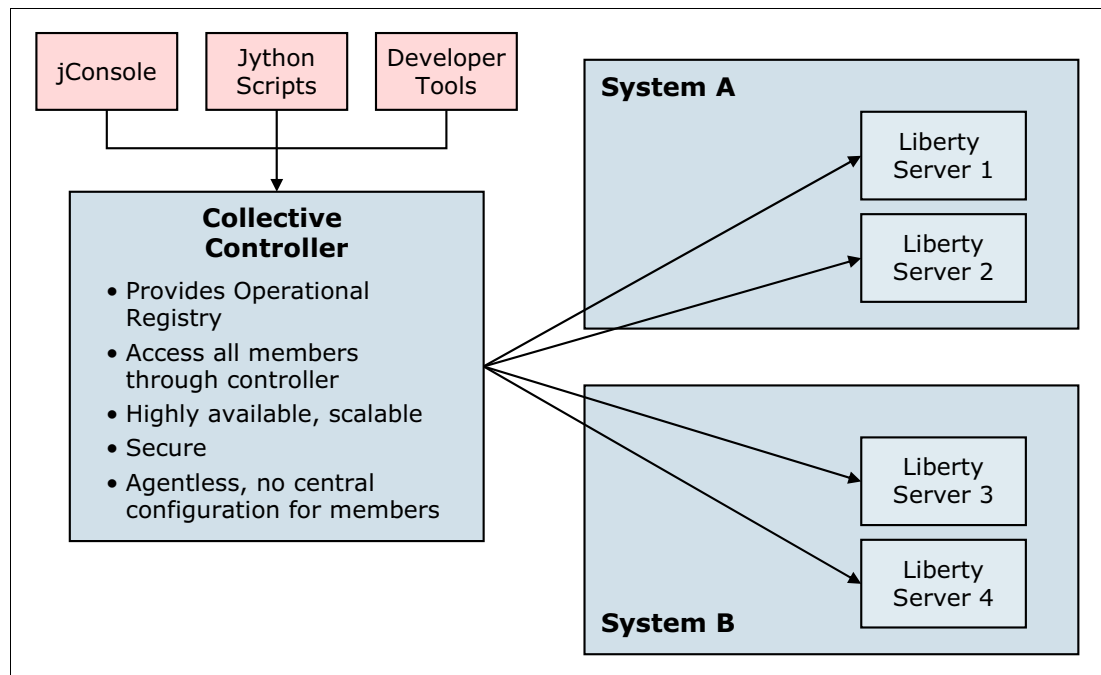


Figure 11 Liberty servers managed as part of a collective

A collective can have more than one controller, forming a replica set. There can be only one replica set per collective, and all controllers must be part of it. In a replica set, each controller replicates its data to the other controllers, thus providing high availability and data protection.

Liberty clusters

(New in V8.5.5) Liberty servers that are members of a collective can be configured into a server cluster for high availability and scalability. The cluster can be treated as a single object in the collective, thereby simplifying the operational management of the servers in the cluster. The members of the cluster can be configured individually, or can share a configuration. A web server plug-in is used to distribute work across the servers in the cluster.

The collective controller provides support for managing the servers in the cluster as one object, including starting and stopping the servers, updating the configuration of the servers, and installing and uninstalling applications. The collective controller also provides you with the capability of adding capacity to an existing cluster and generating the web server plug-in.

Figure 12 shows Liberty servers in a cluster.

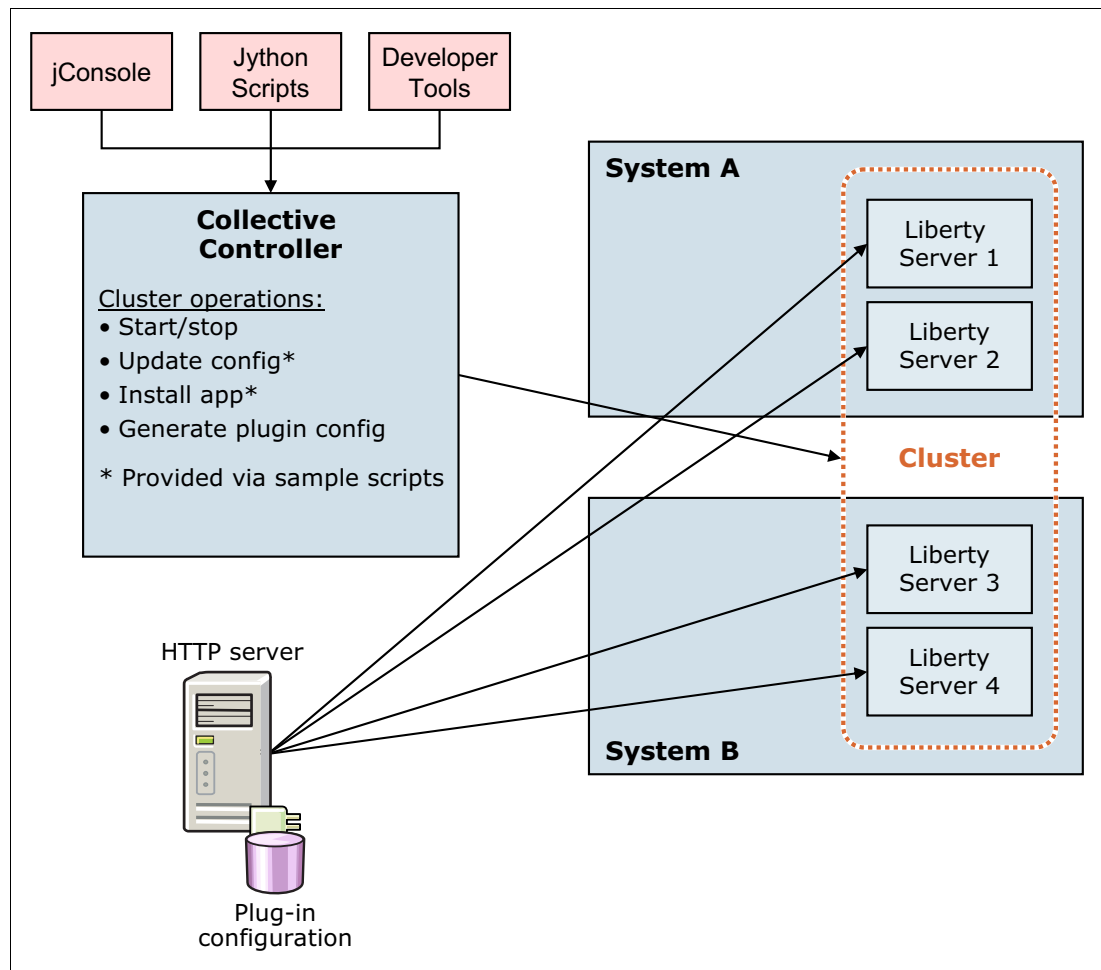


Figure 12 Liberty servers in a cluster

Programming model support and features

The Liberty profile supports a subset of the programming model features supported by the full profile. The programming models supported differ between Liberty Core and the Liberty profile shipped with WebSphere Application Server base, Express, Network Deployment, and z/OS.

- ▶ In Version 8.5.5, the Liberty profile programming model has been extended to further support the technology of the Java EE 6 web profile. The Liberty profile of Liberty Core and the Liberty profile shipped with WebSphere Application Server Base, Express, Network Deployment, and z/OS support the following technology of the Java EE web profile:
 - Servlet 3.0
 - JavaServer Pages (JSP) 2.2
 - Expression Language (EL) 2.2
 - Debugging Support for Other Languages (JSR-45) 1.0
 - Standard Tag Library for JavaServer Pages (JSTL) 1.2
 - JavaServer Faces (JSF) 2.0
 - Common Annotations for Java Platform (JSR-250) 1.1
 - Java Transaction API (JTA) 1.1

- Java Persistence API 2.0
- Bean Validation 1.0
- *(New in V8.5.5)* Enterprise JavaBeans (EJB) 3.1 Lite and Interceptors 1.1 (a subset of the full EJB 3.1 specification, focused on session beans and local interfaces)
- *(New in V8.5.5)* Managed beans 1.0
- *(New in V8.5.5)* Contexts and Dependency Injection (JSR-299 1.0 and JSR-330 1.0)

The Liberty profile shipped with the base, Express, Network Deployment, and z/OS editions also support the following web profile feature:

- Interceptors 1.1
- ▶ *(New in V8.5.5)* The Liberty profile shipped with the Base, Express, Network Deployment, and z/OS editions support Java Message Service (JMS) API 1.1.

This support includes support for the following messaging providers:

- A new lightweight single server message provider is included with the Liberty profile runtime for developing and testing messaging applications.
- Liberty profile now supports configuring WebSphere MQ as the messaging provider, allowing Liberty applications to interact with the WebSphere MQ network.
- The JMS support in Liberty is fully interoperable with the service integration bus in the full profile. The JMS client in Liberty can access the service integration bus, and the full profile JMS client can access messaging providers running in Liberty.

JMS support is not included in Liberty Core.

- ▶ *(New in V8.5.5)* The Liberty profile shipped with the Base, Express, Network Deployment, and z/OS editions support both client and server functionality for SOAP-based Web Services. The supported APIs include the following items:
 - Implementing Enterprise Web Services 1.3
 - Java API for XML-Based Web Services (JAX-WS) 2.2
 - Java Architecture for XML Binding (JAXB) 2.2
 - Web Services Metadata for the Java Platform
 - SOAP with Attachments (SAAJ) 1.3

The following technology for web services is supported in both Liberty Core and in the Liberty profile shipped with the Base, Express, Network Deployment, and z/OS editions:

- Java API for RESTful Web Services (JAX-RS) 1.1
- Java APIs for WSDL 1.2

The SOAP/HTTP and WS-Security protocols are supported.

Web services can be secured without WS-Security through the use of basic authentication along with web application transport constraints. With WS-Security, a default keystore and truststore contains the keys for authentication and encryption. The following WS-Security policies are available:

- Username Token Profile 1.1
- X.509 Token Profile 1.1
- WS-I Basic Security Profile 1.1

Additional support for developing web services has been added to the WebSphere Application Server Developer Tools for Eclipse, including a wizard for top-down WSDL-to-Java generation and a wizard to simplify adding policies to WSDL.

- ▶ Both Liberty Core and the Liberty profile shipped with the Base, Express, Network Deployment, and z/OS editions support the following Java EE-related specifications in Java SE:
 - Java API for XML Processing (JAXP) 1.3
 - Java Database Connectivity (JDBC) 4.0

- Java Management Extensions (JMX) 2.0
 - JavaBeans Activation Framework (JAF) 1.1
 - Streaming API for XML (StAX) 1.0
- ▶ *(New in V8.5.5)* The Liberty profile shipped with the Base, Express, Network Deployment, and z/OS editions support MongoDB to provide access to a scalable, document-oriented NoSQL database. Applications can get a reference to the database using injection, Java Naming and Directory Interface (JNDI) lookup, or J2SE style. After the reference is obtained, access to the database is through the MongoDB Java API.

MongoDB support is not available in Liberty Core.

- ▶ To maintain HTTP session failover and high availability, the Liberty profile can persist session data to a database, or can interoperate with IBM WebSphere eXtreme Scale and the IBM WebSphere DataPower® Appliance XC10 V2 caching appliance.
- ▶ *(New in V8.5.5)* Liberty supports caching for dynamic web content with a local caching service. For scalability or to cache other data, Liberty can interoperate with WebSphere eXtreme Scale or the WebSphere DataPower XC10 V2 Appliance.

For a complete list of programming model features supported, by profile type, see “Summary of programming support” on page 47.

Liberty security

The *appSecurity-1.0* feature of the Liberty profile provides support for securing the server runtime environment and web applications. The *appSecurity-1.0* feature provides support for user registries, authentication, and authorization. The supported user registry types are basic user registry and LDAP user registry.

For secure communication between the client and the server, you can enable SSL for the Liberty profile. A minimal or detailed configuration can be performed by adding the *ssl-1.0* server feature to the server configuration file.

For authenticating users, the Liberty profile supports the following configurations:

- ▶ A basic user registry that defines user and group information for authentication to the Liberty profile.
- ▶ A Lightweight Directory Access Protocol (LDAP) server for authentication.
- ▶ SAF registry (z/OS).
- ▶ *(New in V8.5.5)* Federated LDAP registries, where two or more LDAP registries are defined so that the operations, such as a search for a user, are executed on all the registries.
- ▶ *(New in V8.5.5)* Custom user registries installed as an extension to Liberty.
- ▶ Integration with a third-party security service using trust association interceptors (TAIs). A TAI is used to validate HTTP requests between a third-party security server and a Liberty profile server. The TAI can be called before or after single sign-on (SSO).
- ▶ Single sign-on (SSO) so that web users can authenticate once when accessing the Liberty profile resources such as HTML, JSP files, and servlets. Users can also authenticate once when accessing resources in multiple Liberty profile servers that share Lightweight Third Party Authentication (LTPA) keys.
- ▶ A custom Java Authentication and Authorization Service (JAAS) login module to make additional authentication decisions or to make finer-grained authorization decisions inside an application.

(New in V8.5.0.2) The sync-to-OS-thread feature for z/OS allows the synchronization of a Java thread identity (or JAAS subject) with the OS thread identity for the duration of the current Java EE application request. If you do not choose this option, the OS thread identity value is the same as the servant identity value.

To configure authorizations for an application, you can add authorization tables to the application. The server then reads the deployment descriptor of the application to determine whether the user or group has the privilege to access the resource.

(New in V8.5.0.2) Authorization to resources by using the OAuth 2.0 protocol is supported. OAuth is an open standard for delegated authorization. With the OAuth authorization framework, a user can grant a third-party application access to their information stored with another HTTP service without sharing their access permissions or the full extent of their data.

In V8.5.5, the following security features have been added to support the new programming models included in the Liberty profile:

- ▶ *(New in V8.5.5)* Security support is available for web applications using Servlet 3.0 and for EJBs when the `ejbLite-3.1` feature is present.
- ▶ *(New in V8.5.5)* Web Services security is supported at the transport layer and at the message level. Transport-level security is based on SSL or Transport Layer Security (TLS), and is used to protect HTTP message contents point to point. Message level security is based on WS-Security.

(New in V8.5.5) The `server.xml` file that defines a Liberty server is composed of configuration elements. These elements can require a user ID and password for access to the feature defined. The Liberty profile provides a utility to support Advanced Encryption Standard (AES) encryption for passwords that are stored in the `server.xml` file.

JMX clients connecting to a Liberty profile have two options:

- ▶ Clients can use the local connector, which is protected by the policy implemented by the SDK in use. Currently that policy requires that the client runs on the same host as the Liberty profile, and under the same user ID.
- ▶ Clients that want to connect to a remote Liberty profile use the REST connector. Remote access through the REST connector is protected by a single administrator role and the use of SSL.

(New in V8.5.5) There are several security configuration examples on the `wasdev.net` website for reference when configuring security for your applications on the Liberty profile:

<https://www.ibm.com/developerworks/mydeveloperworks/blogs/wasdev/entry/snippets?lang=en>

The Liberty profile server also provides various plug points that extend the security infrastructure.

Liberty Extensions System Programming Interface

The Liberty Extensions System Programming Interface (SPI) provides the ability to extend the Liberty profile with custom features by using OSGi bundles, including web application bundles. For example, you can extend the OSGi applications programming model by adding new annotations or custom configurations. Or, you can provide a custom user registry. This support can be used to integrate third-party function into the Liberty runtime and development tools.

The product extensions consist of a set of directories and files that are made known to Liberty by a properties file. Independent products can require a Liberty installation and then register with Liberty using the properties file. Users that create features can install them in a built-in extension for convenience. This built-in extension can also be used by products that embed a Liberty install and service.

The SPI supports the full lifecycle of packaging, installation, and uninstallation. The SPI includes plug points that you can implement as features, services for runtime, and instrumentation for monitoring and problem determination.

The SPI contains the following plug points:

- ▶ Custom user registry
- ▶ Trust Association Interceptor
- ▶ Cache provider
- ▶ Webcontainer extension

The SPI contains the following services:

- ▶ File Monitor
- ▶ Application container services
- ▶ Classloaders
- ▶ Publish information to Atlas repository
- ▶ Metatype helpers for processing advanced configuration
- ▶ Location service to access local resources
- ▶ Trace and first-failure data capture (FFDC)

The SPI supports the following instrumentation

- ▶ Entry and exit trace
- ▶ Monitoring
- ▶ Timed operations

Serviceability and troubleshooting

The Liberty profile provides basic implementations of logging, trace, and FFDC services to help you identify and diagnose problems. Messages are sent to a single log file that contains INFO and other (AUDIT, WARNING, ERROR, FAILURE) messages in addition to System.out and System.err. The log is a plain-text file that can be read using a text editor. If trace is enabled, the trace entries are sent to a separate trace data file.

Tracing and logging settings can be set in the `server.xml` file. Or, to diagnose errors with server startup (before `server.xml` is read), these properties can be set in the bootstrap properties.

(New in V8.5.5) The binary logging implementation in the full profile has been ported to Liberty, providing you with the same performance benefits you will find with the full profile. Log and trace entries are stored in a binary format in a log data or trace data repository. The binary data can be copied into a plain-text format for viewing with the `binaryLog` command.

(New in V8.5.5) Timed operations generate a logged warning when JDBC calls in the application server are operating more slowly or quickly than expected. Periodically, the timed operation feature will create a report in the application server log detailing which operations took longest to execute. If you run the `server dump` command, the timed operation feature will generate a report containing information about all operations it has tracked.

The Liberty profile also provides the `server dump` command for problem diagnosis for a Liberty profile server. The result file that is obtained from this command contains server

configuration, log information, and details of the deployed applications in the work area directory.

Usually, a running server includes the following information:

- ▶ State of each OSGi bundle in the server
- ▶ Wiring information for each OSGi bundle in the server
- ▶ A component list that is managed by the Service Component Runtime (SCR)
- ▶ Detailed information about each component from SCR

The Liberty profile also provides a **server javadump** command to help you diagnose problems on a running server at the JVM level, such as hung threads, deadlocks, excessive processor, excessive memory consumption, memory leaks, and defects in the virtual machine.

To help you avoid problems, the Liberty profile provides monitoring support for the following runtime components:

- ▶ JVM
- ▶ Web applications
- ▶ Thread pools
- ▶ *(New in V8.5.5)* Database connection pools
- ▶ *(New in V8.5.5)* Messaging
- ▶ *(New in V8.5.5)* Web services

Development and testing of applications

IBM Rational Application Developer for WebSphere Software V9 provides a development environment for building applications that run on WebSphere Application Server. This tool supports all Java EE artifacts that are supported by WebSphere Application Server, such as servlets, JavaServer Pages (JSP), JavaServer Faces (JSF), Enterprise JavaBeans (EJB), Extensible Markup Language (XML), SIP, Portlet, and web services. It also includes integration with the Open Services Gateway initiative (OSGi) programming model. The workbench contains wizards and editors that help build standards-compliant, business-critical Java EE, Web 2.0, and service-oriented architecture applications. Code quality tools help teams find and correct problems before they escalate into expensive problems. Rational Application Developer for WebSphere Software can be used to develop applications for both the full profile and the Liberty profile.

IBM WebSphere Application Server Developer Tools for Eclipse V8.5.5 provides a development environment for developing, assembling, and deploying Java EE, OSGi, Web 2.0 and Mobile applications, and it supports multiple versions of WebSphere Application Server. When combined with Eclipse SDK and Eclipse Web Tools Platform, WebSphere Application Server Developer Tools for Eclipse provides a lightweight environment for developing Java EE applications.

WebSphere Application Server Developer Tools for Eclipse is a no-charge edition for developer desktops, and it includes Eclipse adapters. With V8.5.5, WebSphere Application Server and WebSphere Application Server Developer Tools for Eclipse editions are provided at no cost for developer desktops, and supported under production runtime licenses.

Although not as rich in features as Rational Application Developer for WebSphere Software, this tool is an attractive option for developers using both the Liberty profile and the full profile.

For more information about WebSphere Application Server Developer Tools for Eclipse and access to the tool, see:

https://www.ibm.com/developerworks/community/blogs/wasdev/entry/downloads_final_releases?lang=en

For developers not using one of the development environments based in WebSphere Application Server, IBM Assembly and Deploy Tools for WebSphere Administration provides a runtime environment for development and testing. This runtime environment is identical to the production runtime environment on which the applications will eventually run. It provides for efficient development and aids in reducing testing effort.

Enhancements have also been made to Rational Application Developer 9.0 and WebSphere Application Server Developer Tools for Eclipse in support of the new 8.5.5 capabilities:

- ▶ Support for developing JAX-WS web services for the Liberty server, new EJB support, and additional templates to support WS-Security X.509 token profile
- ▶ Support for targeting and installing user-defined Liberty features
- ▶ Enhancements to Maven integration, most notably OSGi with EJB and JPA project conversion
- ▶ Enhancements to the web and mobile web development tools including jQuery and Dojo support
- ▶ Improvements to EJB and CDI development tools, including the new beans.xml deployment descriptor editor
- ▶ New editor for configuring the server's dynamic cache mechanism for caching servlets, JSPs, web services, and commands
- ▶ Support of multiple versions of WebSphere Application Server
- ▶ SCA tools enhancements including improved integration of web services bindings
- ▶ Code quality and productivity enhancements with sample-based profiling capabilities from integration with the IBM Monitoring and Diagnostic Tools for Java - Health Center
- ▶ Updates to Java EE Connector Architecture tools including new adapters for IBM CICS®, IBM IMS™ and WebSphere

Deploying, maintaining, and upgrading applications in full profile

During development and test cycles, developers can use the developer tools that are supported by WebSphere Application Server to add an application to the test server and run it. In addition, WebSphere Application Server includes the monitored (or “drop-ins”) directory feature, which increases developer productivity by installing, updating, and uninstalling applications automatically as they are moved in and out of the monitored directory. If you add an enterprise archive (EAR file), Java archive (JAR file), web archive (WAR file), or store archive (SAR file) to any monitored directory, the application is installed and started automatically. The monitored directory works for applications that do not require additional configuration, such as security role mapping.

Applications can be packaged for deployment using the developer tools supported for WebSphere Application Server, or by using the IBM Assembly and Deploy Tools for WebSphere Administration shipped with WebSphere Application Server. With IBM Assembly and Deploy Tools for WebSphere Administration, developers can accomplish key assembly and deployment needs, including editing of deployment artifacts, script development and testing, and application deployment and debugging. This tool is not intended for general application development.

Outside of the developer test environment, applications are typically installed, updated, and uninstalled using the administrative tools, such as wsadmin or the administrative console. As changes are made to an application, multiple versions are created and deployed. The application edition functionality makes it possible to store these application versions in the system management repository and then deploy them as needed.

Using application edition management, you can validate a new edition of an application in the production environment without affecting users, and you can upgrade applications without incurring outages to users. You can also run multiple editions of a single application concurrently, directing different users to different editions. The application edition management feature provides an application versioning model that supports multiple deployments of the same application in a cell. You can choose which edition to activate on a cluster, so that you can either roll out an application update or revert to a previous level.

Deploying, maintaining, and upgrading applications in Liberty profile

Liberty profile provides the following options to deploy an application:

- ▶ Dropping the application into a previously defined “drop-ins” directory
You can use the “drop-ins” directory for applications that do not require additional configuration, such as security role mapping.
- ▶ Adding an application entry to the server configuration file
With the Liberty profile you can deploy a new application by adding an application entry to the server configuration file. To uninstall an application, remove the application entry from the file. When the configuration file is saved, WebSphere Application Server dynamically determines the applications that must be deployed or uninstalled.
- ▶ Using the developer tools
You can use also the developer tools that are supported by WebSphere Application Server V8.5 to develop and deploy applications to a Liberty profile.

Integration with other products

WebSphere Application Server works closely with other IBM products to provide a fully integrated solution. Within your solution environment, these additional products can provide enhanced security and messaging options and broad integration features.

The following sections provide an overview of simply a subset of the products that integrate with WebSphere Application Server V8.5.

IBM WebSphere eXtreme Scale

IBM WebSphere eXtreme Scale provides an extensible framework to simplify the caching of data that is used by an application. You can use WebSphere eXtreme Scale to build a highly scalable, fault-tolerant data grid with nearly unlimited horizontal scaling capabilities. WebSphere eXtreme Scale creates an infrastructure with the ability to deal with extreme levels of data processing and performance. When the data and resulting transactions experience incremental or exponential growth, the business performance does not suffer, because the grid is easily extended by adding additional capacity in the form of JVMs and hardware.

You can use a WebSphere eXtreme Scale grid as a data cache for a database or other data sources that are generally slower to respond because of the need to access data on a hard disk. These data sources are generally more complicated and expensive to scale beyond a single instance and thus they have a limit for the total throughput that can be achieved. A WebSphere eXtreme Scale grid has no such limitation and, when properly used, can scale with a linear response time without any reasonable upper bound.

WebSphere eXtreme Scale can replace the existing HTTP session state management facilities of the application server by placing the session data in a data grid. It fetches user session information from the grid and writes changes back to the grid as necessary. Because the HTTP session data is transient in nature, it does not need to be backed up to disk and can be contained completely in a highly available replicated grid. The grid is not constrained to any one application server product or to any particular management unit, such as WebSphere Application Server cells. User sessions can be shared between any set of application servers and even across data centers in the case of a failover scenario; this allows for a more reliable and fault-tolerant user session state.

Many web-based applications use dynamic page generation techniques, such as JSP. JSP files are used for pages that contain data that rarely changes, such as product details or information about corporate policies. WebSphere Application Server provides an in-memory dynamic cache. This cache is used to store the generated output the first time the page is rendered, and saves both the processing work and back-end system load for subsequent requests. You can configure dynamic cache to use WebSphere eXtreme Scale as your cache provider instead of the default dynamic cache engine.

(New with V8.5.5) The WebSphere eXtreme Scale V8.6 is now packaged with WebSphere Application Server V8.5.5 and comes with entitlements that vary by edition. WebSphere eXtreme Scale V8.6 has the following enhancements:

- ▶ Java and .NET applications can now interact natively with the same data in the same data grid, leading the way toward a true enterprise-wide data grid.
- ▶ A new REST Gateway provides simple access from other languages.
- ▶ WebSphere eXtreme Scale V8.6 delivers a faster, more compact serialization format called eXtreme Data Format (XDF), which is neutral to programming languages.
- ▶ A new transport mechanism, eXtreme IO (XIO), removes the dependency on a Java-specific transport to allow for greater cross-platform communication.
- ▶ Built-in publish/subscribe capabilities enable WebSphere eXtreme Scale V8.6 to update client “near caches” whenever data is updated, deleted, or invalidated on the server side.
- ▶ API enhancements enable continuous query of data that is inserted and updated in the grid.

For more information about WebSphere eXtreme Scale, see:

<http://www-01.ibm.com/software/webservers/appserv/extremescale/>

IBM Tivoli Access Manager

IBM Tivoli® Access Manager provides a holistic security solution at the enterprise level. Integrating WebSphere Application Server with Tivoli Access Manager allows for end-to-end integration of application security for the entire enterprise.

Tivoli Access Manager uses a user repository, such as IBM Tivoli Directory Server, and you can configure it to use the same user repository as WebSphere Application Server, so that you can share user identities with both Tivoli Access Manager and WebSphere Application

Server. The Tivoli Access Manager policy server 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). You can configure the Tivoli Access Manager client using the scripting and GUI management facilities of WebSphere Application Server.

IBM DB2

Almost every enterprise application stores some amount of data. IBM DB2® is an open standards, multiplatform, relational database system that is powerful enough to meet the demands of large corporations and flexible enough to serve medium-sized and small businesses.

IBM DB2 pureScale® is a high performance technology database solution. It implements in-memory transactions in a distributed server environment and efficiently scales your database across several nodes in a shared-disk cluster architecture.

IBM DB2 pureXML® provides intelligent XML data management services without forcing you to transform or “shred” XML data into tabular structures behind the scenes. This capability minimizes administrative impact, simplifies database design, and reduces the complexity of XML applications.

Because access to data is critical to business, DB2 provides high availability using DB2 pureScale and DB2 high availability disaster recovery (HADR), which allows failover to a standby system in the event of a software or hardware failure on the primary database system. The DB2 Storage Optimization feature gives you the ability to transparently compress data on disk to decrease disk space and storage infrastructure requirements. Because disk storage systems can often be the most expensive components of a database solution, even a small reduction in the storage subsystem can result in substantial cost savings for the entire database solution.

In addition to application data, you can integrate DB2 with WebSphere Application Server in various scenarios. You can configure a service integration bus member to use DB2 as a data store for messaging or to use DB2 as the data store for the UDDI registry data. You can configure the session management facility of WebSphere Application Server for database session persistence using DB2 as the data store, and session data is then collected and stored in a DB2 database. The scheduler database for storing and running tasks of the scheduler service of WebSphere Application Server can be a DB2 database.

IBM WebSphere DataPower Appliances

IBM WebSphere DataPower Appliances simplify, govern, and optimize the delivery of services and applications and enhance the security of XML and IT services. They extend the capabilities of an infrastructure by providing a multitude of functions. The capabilities of the WebSphere DataPower Appliances spans SOA connectivity, business-to-business (B2B) connectivity, advanced application caching, rapid integration with cloud-based systems, and more. WebSphere DataPower Appliances are rack-mountable hardware devices or blade servers that mount in an IBM BladeCenter® chassis.

The WebSphere DataPower Appliance family contains the models described in the following sections. Each appliance has its own characteristics and fits various business needs.

IBM WebSphere DataPower Appliance XC10 V2

IBM WebSphere DataPower Appliance XC10 V2 is a purpose-built, easy-to-use appliance that is designed for simplified deployment and hardened security, which is performed in the caching tier of the enterprise application infrastructure. WebSphere DataPower Appliance XC10 V2 incorporates a large 240 GB cache into the DataPower line of appliances from IBM. It adds elastic caching functions that enable business-critical applications to scale cost effectively with consistent performance. WebSphere DataPower Appliance XC10 V2 is designed for drop-in use in various application environments. These environments include WebSphere Application Server V6.1, V7.0, V8.0, and V8.5, and other WebSphere family products. With these environments, it can deliver a cost-effective, distributed caching solution in support of data-oriented distributed caching scenarios.

For more information about WebSphere DataPower Appliance XC10 V2, see:

<http://www.ibm.com/software/webservers/appserv/xc10/>

IBM WebSphere DataPower B2B Appliance XB62

IBM WebSphere DataPower B2B Appliance XB62 delivers secure trading partner data integration tracking, routing, and security functions in a network device. These functions cut operational costs and improve performance. The WebSphere DataPower B2B Appliance XB62 is a nondisruptive technology that you can use to extend existing B2B implementations and internal integration infrastructure, which can deliver rapid return on investment and reduce total cost of ownership.

For more information about WebSphere DataPower B2B Appliance XB62, see:

http://www.ibm.com/software/integration/datapower/b2b_xb60/

IBM WebSphere DataPower Service Gateway XG45

IBM WebSphere DataPower Service Gateway XG45 is a network appliance that is built for web services deployments, governance, light integrations, and hardened security in a single drop-in box. The WebSphere DataPower Service Gateway XG45 provides protection against XML vulnerabilities by acting as an XML proxy and by performing XML well-formed checks, buffer overrun checks, XML schema validation, XML filtering, and XDoS protection.

WebSphere DataPower Service Gateway XG45 also includes many essential security functions beyond those functions of an XML firewall, including web services access control, XML Encryption and Digital Signature, WS-Security, and content-based routing.

For more information about WebSphere DataPower Service Gateway XG45, see:

<http://www.ibm.com/software/integration/datapower/XG45/>

IBM WebSphere DataPower Integration Appliance XI52

IBM WebSphere DataPower Integration Appliance XI52 is a hardware enterprise service bus (ESB) that delivers common message transformation, integration, and routing functions in a network device. These functions can cut operational costs and improve performance. By making on-demand data integration part of the shared SOA infrastructure, the WebSphere DataPower Integration Appliance XI52 is one of the few nondisruptive technologies for application integration.

For more information about WebSphere DataPower Integration Appliance XI52, see:

<http://www.ibm.com/software/integration/datapower/xi52/>

IBM WebSphere DataPower Integration Appliance XI50B and XI50z

IBM WebSphere DataPower Integration Blade XI50B and the WebSphere DataPower Integration XI50z for IBM zEnterprise® are also hardware ESBs. WebSphere DataPower Integration Blade XI50B provides the same functions as the WebSphere DataPower Integration Appliance XI52, but is available in a blade form-factor.

The WebSphere DataPower Integration XI50z for zEnterprise is also similar in function to the WebSphere DataPower Integration Appliance XI52 and is designed specifically for IBM System z® environments.

For more information about WebSphere DataPower Integration Blade XI50B and WebSphere DataPower Integration XI50z for IBM zEnterprise, see:

<http://www.ibm.com/software/integration/datapower/xi50/>

IBM WebSphere MQ

IBM WebSphere MQ is an asynchronous messaging technology for application-to-application communication rather than application-to-user and user interface communication. It offers a fast, robust, and scalable messaging solution. WebSphere MQ ensures one-time only delivery of messages to queue destinations that are hosted by queue managers. Java Message Service (JMS) allows applications to interface with various messaging systems, including WebSphere MQ.

WebSphere Application Server supports asynchronous messaging based on the JMS programming interface using a JMS provider that conforms to the JMS specification V1.1. In WebSphere Application Server V8.5, you can use WebSphere MQ and the default messaging provider.

IBM Tivoli Directory Server

In today's highly connected world, directory servers are the foundation of authentication systems for internal and external user populations in the corporate infrastructure. IBM Tivoli Directory Server provides a high-performance LDAP identity infrastructure that can handle millions of entries. It is built to serve as the identity data foundation for web applications and identity management initiatives.

Using LDAP is a fast and simple way to query and maintain user entities in a hierarchical data structure. It provides benefits over using databases as a user repository in terms of speed, simplicity, and standardized models or schemas for defining data. The LDAP implementations are optimized based on the assumption that data changes relatively infrequently. The directory is primarily for looking up data rather than updating data.

When using the Tivoli Directory Server with WebSphere Application Server, you select either a stand-alone LDAP registry or a federated registry. With a stand-alone registry, WebSphere Application Server can connect to one directory server at a time. Thus, you can have only one LDAP server in your environment. Alternatively, you can set up a failover cluster of the LDAP servers and have the failover managed by WebSphere Application Server.

IBM Tivoli Workload Scheduler

Enterprises are continually looking for ways to reduce costs through automated workload management and monitoring capabilities. IBM Tivoli Workload Scheduler provides the backbone of an automation solution. With Tivoli Workload Scheduler, you can create simple

or complex workload schedules and plans to drive cross-enterprise workloads according to business policies and service levels from a single, web-based operations console. You can increase your business and IT flexibility through dynamic, on-demand execution of workloads according to wanted service levels. Tivoli Workload Scheduler is highly scalable and fault-tolerant, ensuring that workloads are executed even during unplanned incidences with built-in redundancies and recovery processes.

Tivoli Workload Scheduler integrates with many other Tivoli products for monitoring workload automation events in a business context or for additional autonomic capabilities.

The WebSphere Application Server V8.5 batch infrastructure works with enterprise schedulers, such as Tivoli Workload Scheduler, to submit batch work. WebSphere Application Server provides powerful batch application features, including support for parallel batch processing to reduce batch job elapsed time; provide memory-overload protection for batch workloads; and improve xJCL and programming models to increase application development flexibility.

For more information about Tivoli Workload Scheduler, see:

<http://www-01.ibm.com/software/tivoli/products/scheduler/>

IBM WebSphere Adapters

Many enterprise applications rely on, or interact with, other systems within an enterprise. For example, a single transaction within an application can trigger activities across systems, such as supply chain and payment systems or human resource and payroll systems. It is critical to be able to create easy, quick, and reliable connections between these systems. IBM WebSphere Adapters provide a set of generic technology and business application adapters with wizards that allow applications to communicate with other systems in the enterprise.

WebSphere Adapters plug in to WebSphere Application Server V8.5 and provide bidirectional connectivity between enterprise applications (or Java EE components), WebSphere Application Server, and your enterprise information systems (EIS). In addition, the WebSphere Adapter Toolkit integrates with IBM Rational Application Developer, providing an IDE and an integrated test environment.

Using WebSphere Application Server V8.5 with IBM WebSphere Adapters, you can deliver core business applications that are interconnected with diverse IT environments, which can help you optimize current investments and benefit from existing skills.

Summary of programming support

Table 1 shows the Java EE 6 technology support for the full profile, the Liberty profile shipped with the base, Express, and Network Deployment editions, and Liberty Core.

Table 1 Java EE 6 technology support in WebSphere Application Server V8.5.5

Technology	Specification reference	Full profile	Liberty profile	Liberty Core
Java Platform, Enterprise Edition 6 (Java EE 6)	JSR 316	X		
Java Platform, Enterprise Edition 6 Web Profile	JSR 316	X	X (8.5.5)	X
Web services technologies				

Java API for RESTful Web Services (JAX-RS) 1.1	JSR 311	X	X	X
Java APIs for WSDL 1.2	JSR 110	X	X	X
Implementing Enterprise Web Services 1.3	JSR 109	X	X (8.5.5)	
Java API for XML-based Web Services (JAX-WS) 2.2	JSR 224	X	X (8.5.5)	
Java Architecture for XML Binding (JAXB) 2.2	JSR 222	X	X (8.5.5)	
Web Services Metadata for the Java Platform	JSR 181	X	X (8.5.5)	
Java API for XML-based RPC (JAX-RPC) 1.1	JSR 101	X		
Java APIs for XML Messaging 1.3	JSR 67	X		
Java API for XML Registries (JAXR) 1.0	JSR 93	X		
SOAP with Attachments API for Java (SAAJ) 1.3	JSR 67	X	X (8.5.5)	
Web application technologies				
Java Servlet 3.0	JSR 315	X	X	X
JavaServer Faces 2.0	JSR 314	X	X	X
JavaServer Pages 2.2/Expression Language 2.2	JSR 245	X	X	X
Express Language (EL) 2.2	JSR 245	X	X	X
Managed Beans 1.0	JSR 316	X	X	X
Standard Tag Library for JavaServer Pages (JSTL) 1.2	JSR 52	X	X	X
Debugging Support for Other Languages 1.0	JSR 45	X	X	X
Enterprise application technologies				
Contexts and Dependency Injection for Java (Web Beans 1.0)	JSR 299	X	X (8.5.5)	X (8.5.5)
Dependency Injection for Java 1.0	JSR 330	X	X (8.5.5)	X (8.5.5)
Bean Validation 1.0	JSR 303	X	X	X
Enterprise JavaBeans 3.1	JSR 318	X	X (8.5.5) EJB Lite	X (8.5.5) EJB Lite
Interceptors 1.1	JSR 318	X	X (8.5.5)	
Java EE Connector Architecture 1.6	JSR 322	X		
Java Persistence 2.0	JSR 317	X	X	X
Common Annotations for the Java Platform 1.1	JSR 250	X	X	X
Java Message Service API 1.1	JSR 914	X	X (8.5.5)	
Java Transaction API (JTA) 1.1	JSR 907	X	X	X
JavaMail 1.4	JSR 919	X		
Management and security technologies				
Java Authentication Service Provider Interface for Containers	JSR 196	X		
Java Authorization Contract for Containers 1.3	JSR 115	X		
Java EE Application Deployment 1.2	JSR 88	X		

J2EE Management 1.1	JSR 77	X		
Java EE-related specifications in Java SE				
Java API for XML Processing (JAXP) 1.3	JSR 206	X	X	X
Java Database Connectivity 4.0	JSR 221	X	X	X
Java Management Extensions (JMX) 2.0	JSR 255	X	X	X
JavaBeans Activation Framework (JAF) 1.1	JSR 925	X	X	X
Streaming API for XML (StAX) 1.0	JSR 173	X	X	X

Summary of capabilities

Table 2 on page 50 provides information about the capabilities found in each profile for each edition.

Table 2 Comparison of capabilities across WebSphere Application Server offerings

Capability	Liberty Core	Express		Base		Network Deployment/zOS	
		Liberty profile	Full profile	Liberty profile	Full profile	Liberty	Full profile
Administration tools							
Administrative console			X		X		X
Optional provisioning and management through Network Deployment job manager	X	X	X	X	X	X	X
Collective Membership (1) and Management (2)	X (1)	X (1)		X (1)		X (1,2)	
Load balancing and high availability							
IBM HTTP Server routing (# of servers allowed)	2	2	2	25	25	Unlimited	Unlimited
Load balancing and failover with clusters (# of servers allowed)	2	2	2	5	5	Unlimited	Unlimited
Intelligent Management (dynamic cluster, Edition Management, Health Policy)							X
Installation options							
IBM Installation Manager install	X	X	X	X	X	X	X
Archive install option	X	X		X		X	
External caching options							
WebSphere eXtreme Scale entitlement - container JVMs for distributed cache scenarios				Unltd (a)	Unltd (a)	Unlimited (b)	Unlimited (b)
Edge of Network Services						X	X
Liberty z/OS extensions						(z/OS)	
Legend: a) HTTP Session Management and Dynacache only and b) WXS Client on z/OS only							

Authors

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

Carla Sadtler is a Consulting IT Specialist at the ITSO, Raleigh Center. She writes extensively about WebSphere products and solutions. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative, supporting IBM MVS clients. She holds a degree in Mathematics from the University of North Carolina at Greensboro.

Jan Bajerski is a WebSphere Connectivity IT Specialist in the Software Group Community of Practice in Central and Eastern Europe (CEE). He has been involved in the IT industry for 11 years. Previously, he worked in IBM Software Services for WebSphere in Poland, supporting clients in implementing solutions using WebSphere Application Server, WebSphere MQ, and

WebSphere Message Broker. Jan holds a BSc degree from Warsaw University of Technology (Poland).

Davide Barillari is a Certified IT Specialist working for IBM Global Technology Services® in Italy. He joined IBM in 1996 and worked for three years as a z/OS instructor in IBM Education. He has 12 years of experience with IBM Technical Support and has deep knowledge of IBM System z architecture and distributed environments. His main areas of expertise are infrastructure design, implementation, maintenance, and debugging in the WebSphere environment. Davide is an IBM Certified Solution Developer and an IBM Certified System Administrator for WebSphere Application Server, WebSphere Process Server, and SOA Solutions. Since 2011, he has been a Certified Solution Architect for Cloud Computing. He is accredited at the Senior level in the Product Services Profession, and is certified as a Level 1 experienced IT Specialist by IBM Profession Office AITS. He currently provides consulting services at client sites in the banking sector on WebSphere Application Server for z/OS.

Libor Cada is an IT Specialist who works at the Integrated Delivery Center SSO in Brno, Czech Republic. He has 8 years of experience in the IT and banking industries with mainframe System z and Linux for System z environments. He previously held the position of z/OS database and data communication (DB/DC) system programmer for IBM CICS, DB2, WebSphere MQ, and IBM IMS products. Libor currently supports clients from multiple geographies in his role as a WebSphere Application Server and z/OS Certified system programmer.

Susan Hanson is a member of the WebSphere Application Server foundation development team. She has 22 years of experience in developing and delivering IBM software products across the WebSphere and Tivoli brands. Her current focus products are WebSphere Application Server, WebSphere Virtual Enterprise, and WebSphere eXtreme Scale with focus areas in release management, project management, and development process transformation. She also works part-time in the IBM Redbooks organization as a Project Leader focused on Growth Market Unit (GMU) areas, and is part of the ITSO strategy team focused on industries and GMU enablement. Susan holds a Bachelor's degree in Computer Science from East Carolina University and a Master's degree in Computer Information Systems from The University of Phoenix. She is based in Research Triangle Park, North Carolina and is temporarily working and residing in Shanghai, China.

Guo Liang Huang is an experienced technical support engineer working for the IBM WebSphere AIM group in the China lab. He has five years of experience in supporting IBM WebSphere Process Server, and more than 11 years of experience in developing, testing, and supporting software products. He also has expertise in SOA. Guo Liang holds a Bachelor's degree in Computer Science from Central South University of China.

Rispna Jain is a Technical Software Deployment Manager for the WebSphere suite of products in IBM Global Technology Services, working with clients in North America. She has seven years of experience in WebSphere Application Server product development at IBM Software Group in roles such as development, Level 3 support, and testing. Rispna has also been a technical speaker on topics related to WebSphere Application Server at various WebSphere conferences. She is an IBM Certified SOA associate and holds a Master of Technology degree in Computer Science.

Shishir Narain is an Open Group Certified Master IT Specialist with deep skills in IBM middleware products. He works in IBM Software Services for WebSphere at the India Software Lab, Gurgaon. He has 13 years of experience with multiple clients in developing solutions. Shishir has led several end-to-end IT implementations based on SOA. He holds a Master of Technology degree from Indian Institute of Technology, Kanpur.

Jennifer Ricciuti is a Course Developer and Instructor in WebSphere Education. She has 15 years of experience in developing and delivering education courses on various WebSphere products, including WebSphere Application Server, WebSphere Process Server, WebSphere eXtreme Scale, and IBM Business Process Manager Advanced. Her areas of expertise include course design, development, and delivery. Jennifer holds a Bachelor's degree in Computer Science from Point Park University. She works and resides in Pittsburgh, Pennsylvania.

Christian Steege is an IT Architect with IBM Software Group Services for WebSphere in Zurich, Switzerland. He has more than 10 years of experience in designing infrastructures and applications for WebSphere Application Server, WebSphere Business Process Management, WebSphere Portal, and WebSphere MQ. He has implemented many of these infrastructures and applications at a variety of Swiss IBM clients. Christian holds a Master's degree in Information Management from the University of St. Gallen, Switzerland.

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

Margaret Ticknor, Linda Robinson, Debbie Willmschen
International Technical Support Organization, Raleigh Center

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks publications

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

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

© Copyright International Business Machines Corporation 2013. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document REDP-4855-01 was created or updated on June 25, 2013.

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:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400 U.S.A.



Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX®	IMS™	Redbooks (logo)  ®
BladeCenter®	MVS™	System z®
CICS®	pureScale®	Tivoli®
DataPower®	pureXML®	WebSphere®
DB2®	Rational®	z/OS®
Global Technology Services®	Redbooks®	zEnterprise®
IBM®	Redpaper™	

The following terms are trademarks of other companies:

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

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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