Chapter 1. WebSphere Application Server V6: Packaging and Architecture

With complete J2EE compatibility and integrated Web services support, Version 6.0 of WebSphere Application Server delivers high performance and a highly scalable transaction engine for dynamic and mission-critical e-business applications. The product, the cornerstone of the IBM WebSphere product family, forms the foundation for IBM’s other middleware offerings. Leading-edge products such as WebSphere Commerce Server, WebSphere Process Server (formerly WebSphere Business Integration Server Foundation), WebSphere Enterprise Service Bus, and WebSphere Portal Server (to name just a few) are built on top of this powerful application server.

In Version 6, WebSphere Application Server supports the full Java 2 Platform, Enterprise Edition (J2EE) 1.4 programming model, including servlets, Java Server Pages (JSPs), Enterprise JavaBeans (EJB), and Web services. WebSphere V6 also continues to support applications developed using the J2EE 1.2 and 1.3 specifications, making it easier to migrate applications deployed on WebSphere Application Server 4.x and 5.x. [Figure 1-1](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig01) summarizes the standards support in WebSphere Application Server V6. (For complete documentation of the J2EE 1.4 specification, see the list of Web references in the Appendix.)

**Table 1-1. Standards support in WebSphere Application Server V6**

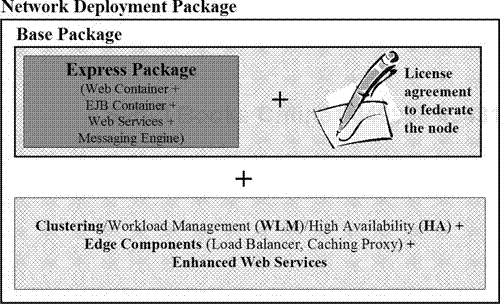
|  |
| --- |
| * Java 2 Platform, Enterprise Edition (J2EE) 1.4   + Servlet 2.4, Java Server Pages (JSP) 2.0   + Enterprise JavaBeans (EJB) 2.1, Java Message Service (JMS) 1.1, J2EE Connector Architecture (JCA) 1.5   + Java Specification Request (JSR) 109, Java API for XML-based Remote Procedure Call (JAX-RPC/JSR 101), Simple Object Access Protocol with Attachments API for Java (SAAJ) 1.2   + Java Database Connectivity (JDBC) 3.0, JavaMail 1.3 * Java Development Kit (JDK) 1.4.2 * WebSphere programming model extensions   + ActivitySessions   + Application Profiling   + Asynchronous Beans   + Distributed Map   + Dynamic Query   + Internationalization   + Last Participant Support   + Object Pools   + Scheduler   + Startup Beans   + Work Areas |

WebSphere V6 is supported on many platforms, including Windows (Windows 2000, 2003, and XP), Unix (AIX, HP-UX, Linux, and Solaris), and IBM’s i5/OS (OS/400) operating system. For a complete list of supported platforms, see the Web references list in the Appendix. In this chapter, we look at how IBM packages the product in Version 6 and examine the architectures associated with each variation. Before concluding the chapter, we’ll review some of the significant new features in V6, including administrative console improvements, high-availability features, and deployment and management enhancements. [Chapter 2](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch02.html) gives you an overview of the installation and configuration process and provides an introduction to the rest of the book.

Packaging

IBM offers WebSphere Application Server V6 in the following packages:

* ***WebSphere Application Server V6 – Express—*.**The Express package provides a fully functional, affordable, and easy-to-use Java application server and development environment. Unlike Version 5, the V6 Express package provides full J2EE support, including an Enterprise JavaBeans container.
* ***WebSphere Application Server V6—*.**In addition to the functionality supported by the Express runtime, the base application server product — what we call “the Base package” in this book — delivers high performance and a highly scalable transaction engine for dynamic e-business applications. The Base package provides the same functionality as the Express package, but an expanded license agreement supports the ability to federate (i.e., add) an application server node to a WebSphere Deployment Manager cell. (You’ll learn more about WebSphere nodes and cells later in this chapter.)
* ***WebSphere Application Server V6 – Network Deployment—*.**In addition to the functionality provided in the Base/Express package, the Network Deployment (ND) package delivers advanced deployment services, including clustering, edge-of-network services, Web services enhancements, and high availability for distributed configurations. [Figure 1-2](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig02) illustrates the relationship among the Base, Express, and Network Deployment packages.



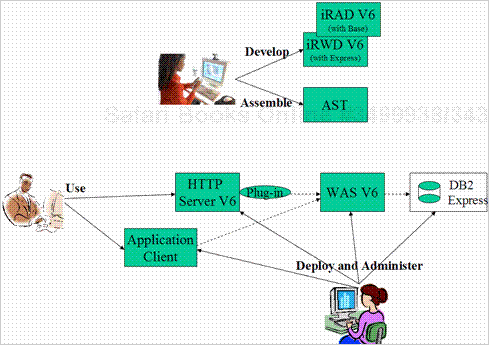
**Figure 1-2. WebSphere Application Server V6 product packaging**

* ***WebSphere Application Server V6 – Extended Deployment—*.**Installed on top of the Network Deployment package, the Extended Deployment (XD) package helps customers with multiple mission-critical and complex applications improve availability and performance by balancing and sharing the workload of multiple application server nodes and clusters to provide on-demand computing.

Base/Express Package Software

[Figure 1-3](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig03) depicts the software products that come with the Express and Base versions of WebSphere Application Server. These two packages include essentially the same set of software (the development tool provided is the only difference). The Base/Express package includes the following software:

* Standalone application server node with support for multiple instances
* Application client
* IBM HTTP Server V6
* Web server plug-ins
* IBM Rational Web Developer (iRWD) V6 (with the Express package) or a trial version of IBM Rational Application Developer (iRAD) V6 (with the Base package)
* Application Server Toolkit (AST)
* DB2 Universal Database (DB2 UDB) Express Edition 8.2



**Figure 1-3. WebSphere V6 Base/Express software**

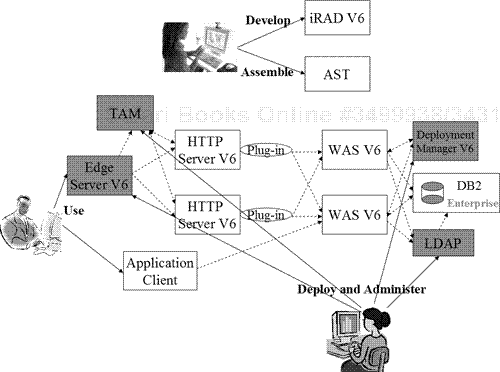
It’s important to understand that no functionality or feature differences exist between the Express and Base versions of WebSphere Application Server in the WebSphere V6 runtime. These packages are built of the same code base and therefore are identical in terms of features and functions. The difference between the two packages is simply a licensing issue. To be able to federate an application server node to a WebSphere cell, you must have a paper agreement; only the Base package permits use of this feature. A license agreement also grants Base installations unlimited CPU use, while Express package users are limited to two CPUs.

Network Deployment Package Software

To the Base/Express package’s support for a standalone application server environment, the WebSphere Application Server V6 – Network Deployment package adds the ability to create and manage a distributed server configuration. This product includes the following software in addition to the software provided in the Base/Express package:

* Deployment Manager
* Edge Components (Load Balancer, Caching Proxy)
* IBM Tivoli Directory Server (a Lightweight Directory Access Protocol, or LDAP, server)
* Tivoli Access Manager (TAM) server
* DB2 UDB Server Edition 8.2
* IBM Cloudscape database

[Figure 1-4](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig04) depicts the software products that come with the Network Deployment package.

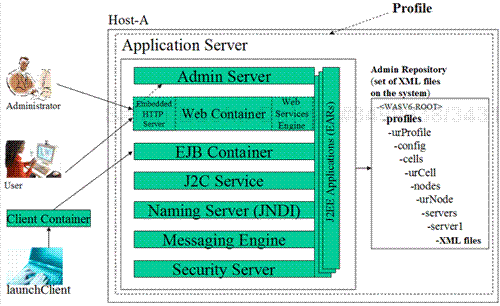


**Figure 1-4. WebSphere V6 Network Deployment software**

The key additional features provided in the Network Deployment package are support for workload management, high availability, and clustering and the ability to run the Web server as a managed node. (If you use IBM HTTP Server V6 as your Web server, you can manage it through IBM HTTP Admin Server V6 instead of using a node agent process from the WebSphere administrative console. This feature is not available with other supported Web servers.)

Standalone Application Server

The heart of a WebSphere Application Server installation is the application server itself. The application server provides a runtime environment in which to deploy, manage, and run J2EE applications. Both the Base and Express packages support a standalone application server structure.[Figure 1-5](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig05) illustrates the components of this basic architecture, which forms the foundation of every WebSphere Application Server installation.



**Figure 1-5. WebSphere Application Server V6 architecture**

Each application server runs in its own Java Virtual Machine (JVM). As you can see in the diagram, the application server process contains the following services and containers:

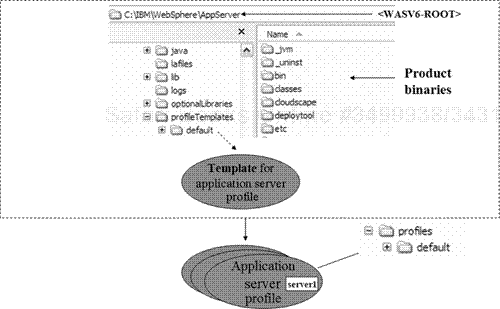
* ***Admin server—*.**The administrative server maintains and manages the administration configuration repository (which consists of Extensible Markup Language, or XML, flat files in the file system). The admin server accepts requests from the WebSphere administrative console (a browser-based graphical interface for configuring and managing WebSphere resources) and wsadmin commands and changes the configuration information accordingly. If you’ve enabled global security, the admin server also secures the administrative repository by authenticating and authorizing the user role.
* ***Web container—*.**The Web container provides a runtime environment for servlets, JSPs, JavaBeans, and static content (if you’ve enabled the file-serving servlet). These artifacts are typically packaged as a Web module and run inside the Web container. A Web module is packaged as a file with a .war (Web archive) extension. If necessary, the Web container works with the application server’s EJB container or the Web services engine to process requests. Files with .war and .jar (Java archive) extensions run inside the Web container.
* ***Embedded HTTP Server (EHS)—*.**The application server’s EHS component receives requests from an external HTTP server (e.g., IBM HTTP Server) using Hypertext Transfer Protocol (default TCP/IP port 9080) and passes the requests to the Web container for processing. The embedded server can serve all static content, just like an external HTTP server. EHS isn’t meant to replace an external HTTP server in a production environment, but you can use it to test your WebSphere applications. By using EHS as an HTTP server to serve static content without an external HTTP server, you can save system resources in development and functional testing environments.
* ***Web services engine —*.**The Web services engine is the part of the application server runtime that supports Web services. Web services are self-contained, modular applications that you can describe, publish, locate, and invoke over a network.
* ***EJB container —*.**The application server’s EJB container provides a runtime environment for deploying, managing, and running Enterprise JavaBeans. These artifacts are typically packaged as an EJB module and run inside the EJB container. An EJB module is packaged as a file with a .jar extension. Files with a .jar extension and an EJB deployment descriptor run inside this container. Java clients communicate with Enterprise JavaBeans in the EJB container. The client container depicted in the figure provides services for running standalone Java clients that use the application server. You invoke the client application container by executing the launchClient command.
* ***J2C service —*.**The Java 2 Connector (J2C) service provides connections between J2EE applications running inside the application server and applications running in legacy enterprise information systems. Connections and their runtime environment are pooled and managed as defined in the J2EE 1.4 specification’s J2C architecture.
* ***JNDI naming server —*.**You can register resources in the application server’s Java Naming and Directory Interface (JNDI) namespace. Client applications can then obtain the references to these resource objects in their programs.
* ***Messaging engine —*.**The messaging engine provides the core messaging functionality of a Service Integration Bus (SIBus). To enable the messaging engine in the application server process (default messaging provider with WebSphere V6), you create an SIBus and attach the application server to it as a member. This messaging engine is all-Java based and runs in-process with the application server. Applications can use an external JMS provider such as WebSphere MQ, a third-party JMS provider such as TIBCO, or the WebSphere V5 embedded messaging provider instead of the default messaging engine. In this book, we cover use of the default messaging engine only.
* ***Security server —*.**The security server provides authentication and authorization services.

Application Server Profile

Notice that [Figure 1-5](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig05) depicts the application server as enclosed within a *profile*. The concept of profiles is new in WebSphere Application Server V6. A profile contains a set of files that enable the runtime environment for a WebSphere server process. You create a profile for the application server during the installation process.

Using the Base/Express package, you can create only one type of profile: an application server profile. The Network Deployment package supports two additional profile types, deployment manager and custom, both of which we describe later.

As the diagram in [Figure 1-6](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig06) illustrates, when you install the WebSphere V6 package, the installation program copies to a directory on the server machine the product binaries, which include a default application server *template* and other files required to create a profile. During the installation process, you use the profile template to define the configuration settings of the new server you’re going to create.



**Figure 1-6. Relationship between an application server template and a profile**

When you create an application server profile, an application server (named server1) is created by default, and necessary applications are deployed on it to help manage the configuration (adminconsole, filetransfer) and verify the installation (default application, installation verification test). You can use the same set of product binaries (and template) to create multiple profiles.

If you desire, you can create multiple application servers (e.g., server2, server3) within the same profile. But for easy maintenance, you may want to create a new profile (an application server process gets created with a profile) if you ever expect to have multiple application servers.

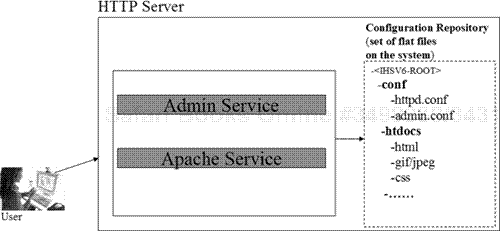
IBM HTTP Server

You can use WebSphere Application Server’s Embedded HTTP Server (available inside the Web container) to serve static content, but EHS isn’t meant to be used this way in a production environment, due to scalability and availability limitations. For this reason, IBM includes the “full-strength” IBM HTTP Server Web server as part of the WebSphere V6 product.

IBM HTTP Server V6, which comes with both the Base/Express and the Network Deployment packages, is based on the Apache Web Server 2.0.47. As a Web server, HTTP Server’s purpose is to process and serve static content, such as Hypertext Markup Language (HTML), images, Cascading Style Sheets (CSS), and JavaScript.

[Figure 1-7](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig07) shows the basic architecture of HTTP Server in V6. The server contains the following services:

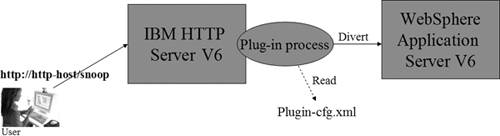
* ***Apache service —*.**This is the core Apache service that processes HTTP requests.
* ***Admin service —*.**If you configured the Web server node as an unmanaged node, the WebSphere admin service can use the HTTP admin service to manage HTTP Server, change the configuration of the HTTP plug-in and HTTP Server, and propagate the plug-in configuration file remotely from the WebSphere administrative console.



**Figure 1-7. IBM HTTP Server V6 architecture**

WebSphere V6 Plug-in for HTTP Server

When a Web application user sends a request, the external HTTP server processes the request first (here we are assuming that the plug-in module is not installed and configured to run with the HTTP server). If the request is for static content only, the HTTP server can serve it. (Requests for static content can be served using the file-serving enabler servlet from the application’s Web module running in the application server’s Web container or, if configured appropriately, from the document root of IBM HTTP Server.) However, if the request is to display dynamic content (served through a servlet and/or a JSP), an application server must fulfill it. In this case, the HTTP server needs to know where and how to divert the request. For this reason, WebSphere provides a *plug-in module* that runs with HTTP Server. The plug-in module uses an XML-based file containing a request-routing table to divert requests to the appropriate application server. [Figure 1-8](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig08) illustrates the plug-in’s relationship with WebSphere V6 and HTTP Server V6. Once you’ve installed and configured the WebSphere plug-in module on the HTTP server, the plug-in process receives the request from the client first. If the requestis for dynamic content, the plug-in diverts the request to the WebSphere application server. If the request is for static content, the plug-in forwards it to the HTTP server.



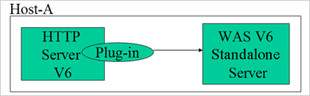
**Figure 1-8. Plug-in relationship to WebSphere V6 and HTTP Server V6**

Architectures Possible with the Base/Express Package

When it comes to architecting a WebSphere V6 Base/Express application server, you have various options with respect to IBM HTTP Server and the plug-in module. In this section, we review a few possibilities to give you a general idea of how you might build a WebSphere V6 system. You can create any of the architectural variations described here (and others) by following the steps given in [Chapters 3](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch03.html) through [6](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch06.html).

Local Plug-in

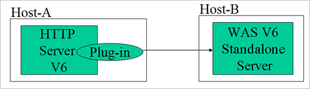
In a local plug-in configuration, both WebSphere Application Server and the HTTP server (along with the plug-in module) exist on the same machine. [Figure 1-9](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig09) illustrates this architecture. This simple configuration works well in training and functional/unit testing environments.



**Figure 1-9. Standalone server: Local plug-in architecture**

Remote Plug-in

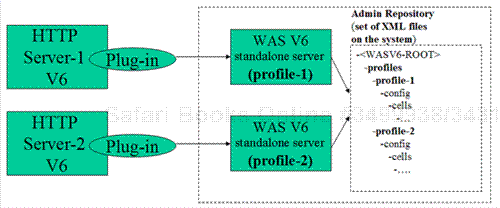
In a remote plug-in configuration ([Figure 1-10](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig10)), WebSphere Application Server and the HTTP server (and plug-in module) exist on different machines. You can use this type of configuration for noncritical, small production or test environments.



**Figure 1-10. Standalone server: Remote plug-in architecture**

Multiprofile Local or Remote Plug-in

In a multiple-profile configuration ([Figure 1-11](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig11)), multiple application server instances share one WebSphere V6 Base/Express installation (i.e., a single set of product binary files).



**Figure 1-11. Single WebSphere V6 install: Multiple profiles with local or remote plug-in architecture**

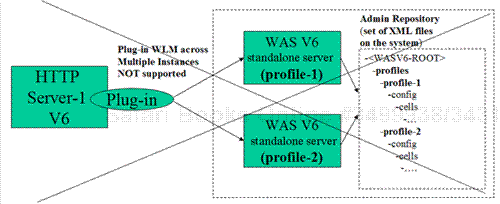
In this setup, the plug-in module can be local or remote. The important point to note is that each plug-in module can talk to only one profile built using the Base/Express package software. You can use this configuration to host different applications as follows:

* one application on each profile
* different versions of the same application, each deployed to their own profile
* one profile for each department in an organization

Of course, you can also have multiple installations of WebSphere V6 (multiple binary files) on a single machine, each with multiple profiles.

Architectures Not Supported

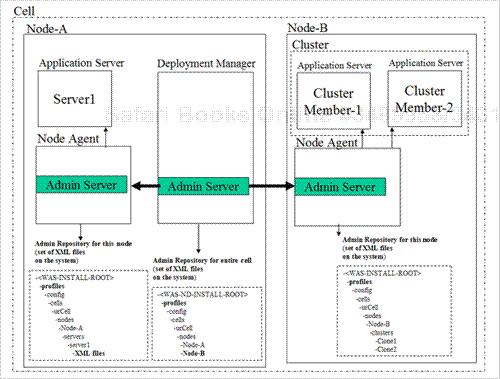
The plug-in file generated from Base/Express servers cannot spray, or route, requests to multiple instances. [Figure 1-12](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig12) illustrates this limitation. If you need this kind of capability (i.e., workload management in the plug-in), you must use the Network Deployment package. In addition, even though you technically can federate an application server installed using the Express package to a Deployment Manager cell, it’s illegal to do so without a Base package license.



**Figure 1-12. No Base/Express ability to spray requests across multiple application servers**

Network Deployment Package: A Cell of Application Servers

WebSphere Application Server’s Network Deployment package builds on the features of the Base/Express package by providing support for a distributed server configuration, complete with centralized administration, advanced deployment services, workload management, and high availability. [Figure 1-13](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig13) depicts the basic architecture of the Network Deployment version of WebSphere Application Server.



**Figure 1-13. WebSphere Application Server V6 – Network Deployment architecture**

The figure shows a WebSphere *cell* that consists of two nodes: node A and node B. Each node is a collection of WebSphere managed server processes (application server, Web server, generic server, and so on). Nodes in a cell can be managed or unmanaged. (You’ll learn more about managed and unmanaged nodes later in this chapter and in the remainder of this book.) Many people wrongly think that a one-to-one relationship exists between a WebSphere node and a physical machine (which is sometimes also called a node). But in a WebSphere cell, a single physical machine can contain multiple WebSphere nodes.

Within each cell, the Deployment Manager process, as its name implies, is responsible for managing the administrative configuration for the entire cell. In the Network Deployment environment, the WebSphere administrative console connects to the Deployment Manager to work with the cell configuration. You thus have a centralized, single point of administration for all the nodes in a cell. (Each node will contain one or more application servers.) The Deployment Manager runs in its own JVM.

Every WebSphere node that has been federated to the Deployment Manager contains one *node agent*. The node agent is responsible for propagating administration configuration changes from the Deployment Manager to its node. Like the Deployment Manager, each node agent process runs in its own JVM. A managed node contains a node agent on that node.

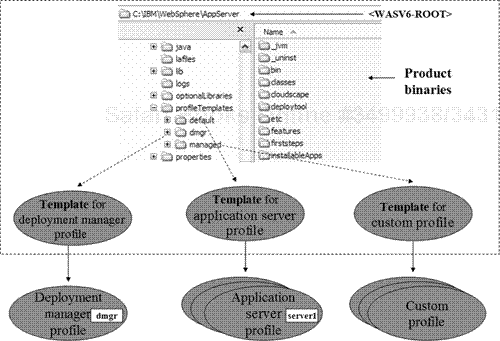
A *cluster* (shown in node B of the figure) is a collection of application server processes hosting an identical collection of J2EE applications, thus providing workload management and high availability. The architecture depicted in the figure has a limitation, though. As you might guess, this configuration provides *process* failover — if cluster member 1 fails, the application remains available through cluster member 2 — but it can’t provide *system* failover. If the entire node B goes down, the applications running in the cluster are unavailable entirely. To maintain high availability during system failover, it would be wise to move a cluster to node A or create an extra cluster member there.

A new feature in WebSphere V6 is the ability to group nodes within a cell for better manageability. The node group feature (not depicted in the figure) is useful if you have nodes on different operating systems and/or platforms. In such an environment, support for node groups lets you organize nodes by operating system or platform.

Profiles Supported in the Network Deployment Package

As you learned earlier, a WebSphere profile contains a set of files that enable a runtime environment for a WebSphere server process. The WebSphere V6 Base and Express packages support one kind of profile, an application server profile. In addition to the application server profile, the WebSphere V6 Network Deployment package supports two more profile types: deployment manager and custom.

As [Figure 1-14](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig14) illustrates, when you install the Network Deployment package, the installation program copies product binaries containing three kinds of templates — deployment manager, application server (default), and custom (managed) — along with the files required to create the associated profiles. You can create multiple profiles from the same set of product binaries for each profile template.



**Figure 1-14. Templates and profiles in the Network Deployment package**

When you create an application server profile, an application server process (server1) is created by default, along with the set of files required to run and manage that application server (server1).

When you create a deployment manager profile, a Deployment Manager process (dmgr) is created by default, along with the necessary applications to manage the configuration (adminconsole, filetransfer) and verify the installation (ivt). The Deployment Manager helps the administrator manage multiple application servers and clusters that exist on one or more nodes. To be managed from the Deployment Manager’s admin console, nodes within the application server and/or custom profile must be federated (added) to the Deployment Manager.

When you create a custom profile, files are created that are necessary to

* federate a node within the custom profile to the Deployment Manager
* run and manage any application servers that will be created on the node from the Deployment Manager’s admin console after federation

Unlike the application server profile, a custom profile contains no application server or applications that manage and verify the configuration immediately after its creation.

A custom profile is equivalent to the application server profile minus the application server process (server1). You use the custom profile when you know you’re going to immediately federate the node and have no intention of using the profile in a standalone configuration.

If you want to manage an HTTP server remotely from the WebSphere admin console, you create a custom profile (or application server profile) on the HTTP server machine and federate it to the Deployment Manager cell. When you federate the node within the custom profile to the Deployment Manager, you’ll see the creation of the node agent process on the machine where you created the custom profile.

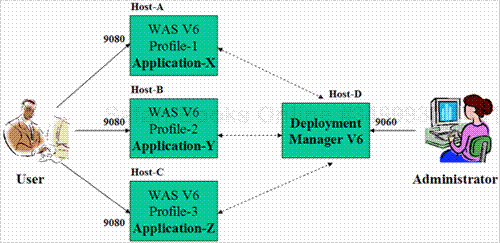
Architectures Possible with the Network Deployment Package

You can configure WebSphere Application Server V6 – Network Deployment in various ways with respect to HTTP Server, the plug-in module, Edge Server – Load Balancer, and the DB2 database. (We should note here that WebSphere supports a variety of options in terms of caching proxy and load balancers that operate on the edge of the network. In addition, WebSphere supports a variety of JDBC-compliant databases — DB2, Informix, Oracle, SQL Server, and Sybase, to name a few — for persistent storage. The examples in this book are restricted to IBM-specific products to serve as representative samples.)

To give you a general idea how to build a WebSphere V6 ND system, let’s look at a few examples. You can create any of the architectural variations described here using the Network Deployment package. This package also supports all the architectural possibilities we discussed earlier for the Base and Express packages, including those you learned weren’t supported in those packages. We’ll delve further into the Network Deployment architecture in [Chapters 7](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch07.html) and beyond.

Manage Multiple Application Servers from the Deployment Manager’s Admin Console

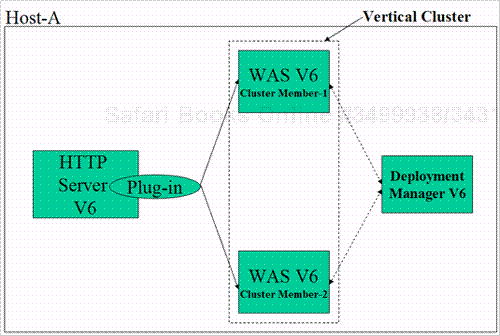
In this configuration (shown in [Figure 1-15](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig15)), you manage multiple application servers through the Deployment Manager’s administrative console. Each profile must be federated to the Deployment Manager to permit managing the application servers on the node corresponding to the profile. Each application server can host one or more applications. Using the Deployment Manager, you can configure a Web server (or multiple Web servers) to send requests from the Web server to the application servers. This configuration works well in education, training, and noncritical small production environments.



**Figure 1-15. Managing multiple application servers from the Deployment Manager’s admin console**

Vertical Clustering with Distributed Local Plug-in

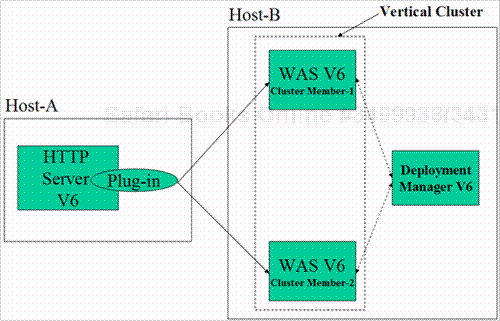
In this configuration ([Figure 1-16](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig16)), all the components (HTTP Server, plug-in module, WebSphere V6 cluster members, and Deployment Manager) exist on the same node. The plug-in module sprays requests to all the application servers in the cluster. This configuration is suitable for education and training environments.



**Figure 1-16. Vertical clustering with distributed local plug-in**

Vertical Clustering with Distributed Remote Plug-in

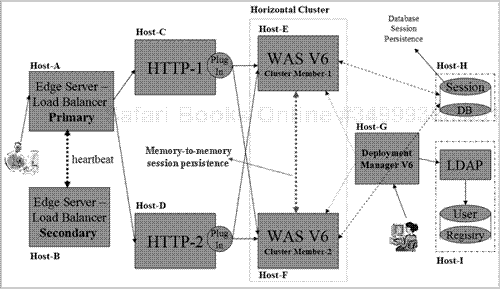
In this configuration ([Figure 1-17](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig17)), the Deployment Manager and HTTP Server (along with the plug-in module) exist on separate nodes. Application servers (cluster members) can exist on the Deployment Manager machine or on a separate machine. The plug-in module can spray requests to both application servers. This configuration works well for education, training, stress testing, staging, and noncritical midsized production environments.



**Figure 1-17. Vertical clustering with distributed remote plug-in**

Highly Available/Workload-Managed Horizontal Clustering

In this configuration ([Figure 1-18](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig18)), WebSphere Application Server, HTTP Server (and the plug-in module), the Deployment Manager, Edge Server – Load Balancer, and the DB2 database exist on different nodes. This configuration provides high availability for each component (WebSphere Application Server, HTTP Server, and Edge Server). The only single points of failure here are the database and the LDAP server. (LDAP serves as the user registry to validate users.) You can use this configuration in production environments once you’ve configured the database and LDAP for high availability.



**Figure 1-18. Highly available/workload-managed horizontal clustering**

In the highly available/workload-managed scenario, when a user issues a request from a browser, the primary Edge Server – Load Balancer processes the request first. If the primary Load Balancer fails, the secondary Load Balancer starts receiving the requests from users. After receiving a request from the browser, the Load Balancer (also known as an *IP sprayer*) sprays requests to multiple HTTP servers. If one HTTP server fails, the remaining HTTP server (or servers) receives all the requests.

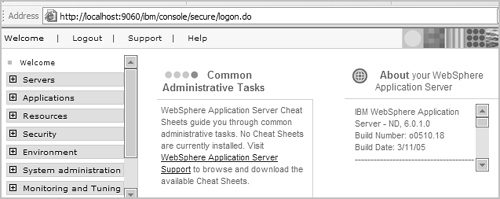
After receiving the request from the Load Balancer, the plug-in module on the HTTP server receives the request and sprays it to multiple application servers:

* If one application server fails, the remaining application server (or servers) serves the requests.
* If you’ve enabled persistent sessions, the user sessions stored in the session database of the failed application server are retrieved by the application server that is serving the request on behalf of the failed application server.
* If you’ve configured session persistence for memory-to-memory replication, the session data will be stored on the other cluster members.

After receiving the request from the plug-in module (in HTTP Server), the Embedded HTTP Server (part of the application server) receives the request and delegates it to the Web container (also part of the application server). In handling the request, the servlet/JSP (running inside the Web container) usually makes method calls with the help of supporting Java bean classes that may use JDBC to access a database server or method calls to Enterprise JavaBeans running in the EJB container. In handling an incoming HTTP request, a servlet/JSP may optionally invoke an EJB that thereafter accesses data in some enterprise resource, such as a JDBC-compliant database. The servlet or EJB may optionally also invoke services from utility or dependent classes that are available to the application.

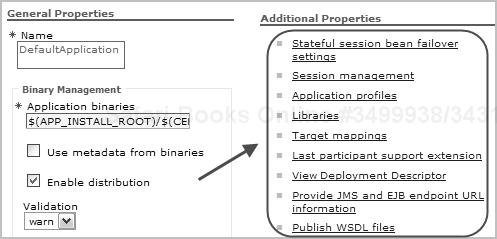
WebSphere Application Server V6: New Features

We’d be remiss to conclude this introductory chapter without a brief look at some of the important new features you’ll find in WebSphere Application Server V6. Among the most easily noticeable changes is a new look for the administrative console. In V6, the WebSphere admin console presents the standard look and feel that IBM has adopted across its entire WebSphere product family. You can see the console’s new look in [Figure 1-19](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig19).



**Figure 1-19. Admin console’s new appearance**

The new console also features simplified navigation, making it easier to find what you want than in previous versions. For example, when you click on an application, you no longer need to go through a series of screens to find all the application’s properties. A single screen now provides links to all these attributes ([Figure 1-20](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig20)). In addition, the console’s main viewing pane now displays its information in two columns, alleviating the need to scroll to locate commonly used items on the screen.



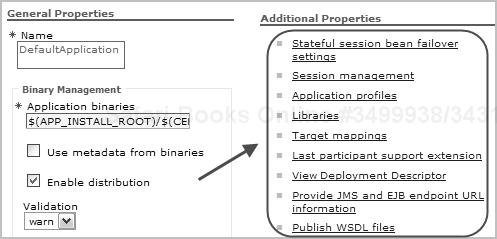
**Figure 1-20. Application properties**

Another console-related change in V6 involves the port number and context root you use to invoke the admin console. Instead of the URL http://washost:**9090/admin**, you now use http://was-host:**9060/ibm/console** to invoke the admin console. The WAS V5 context root, /admin, still works in V6; it is redirected to /ibm/console. For example, you can still use the URL http://was-hostname:9060/admin to invoke the admin console.

Other Changes

You’ll find a host of other improvements and changes in V6, including the following highlights:

* System applications (e.g., admin console, file transfer EARs) are no longer displayed in the admin console or stored in the installedApps directory along with other enterprise applications. In V6, these applications reside in a separate repository.
* Multiple instances can now share the product binary files. As you’ve learned, each instance, or application server process, is called a profile in WebSphere V6.
* The default messaging provider is installed automatically when you install the application server. The messaging engine isn’t enabled by default, but you can enable it by performing a few simple configuration steps.
* You can now configure and manage the Tivoli Performance Viewer (TPV) from the admin console (as shown in [Figure 1-21](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig21)). [Chapter 22](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch22.html) provides a step-by-step description of how to collect data using TPV during performance monitoring.



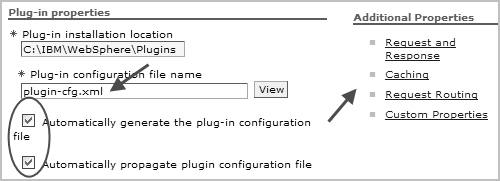
**Figure 1-21. Tivoli Performance Viewer in WebSphere V6 admin console**

* In V6, the Web server plug-in installation is separate from the WebSphere Application Server and IBM HTTP Server installations ([Figure 1-22](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig22)).

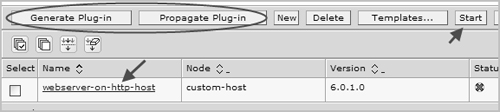


**Figure 1-22. WebSphere V6 installation options**

* You can manage IBM HTTP Server V6 through the WebSphere V6 admin console. You can also use the console to make plug-in file configuration changes ([Figure 1-23](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig23)) and propagate the plug-in to an HTTP Server V6 node ([Figure 1-24](https://www.safaribooksonline.com/library/view/websphere-application-server/9781583470619/ch01.html#ch01fig24)).



**Figure 1-23. Plug-in configuration changes via WebSphere V6 admin console**



**Figure 1-24. Plug-in propagation to HTTP Server via WebSphere V6 admin console**

* The WebSphere Rapid Deployment (WRD) feature enables automatic installation of applications and modules onto a running application server. (This feature is intended for use only in development environments.)
* Fine-grained application-update capability lets you add, change, and/or delete parts of an installed application or selected modules of an installed application.
* Using the V6 configuration archive feature, you can export or import a full or partial configuration between different instances and applications.
* Enhanced enterprise archive (EAR) file support enables easy packaging and deployment. The new support includes the ability to include non-J2EE components (e.g., property files) inside an .ear file. In addition, J2EE resource definitions (e.g., data source definitions) can be embedded in the EAR (in the deployment.xml file) and then created when the EAR is deployed.
* You can configure resources and WebSphere variables in two new scopes (along with the existing cell, node, and server scopes):
  + ***cluster scope —*.**The resource or variable is visible to all members of the cluster.
  + ***application scope —*.**The resource or variable is visible to that application only.
* A new global object, AdminTask, has been introduced in the wsadmin program (along with existing objects AdminConfig, AdminApp, AdminControl, and Help). The new task enables you to run admin commands inside wsadmin.
* The default Java Message Service messaging engine available in the WebSphere V6 Base and Network Deployment packages is the Service Integration Bus technology (also known as Platform Messaging). You can configure the SIBus through the admin console or using the wsadmin command. (You can also configure applications to use any of the three other JMS provider options that WebSphere V6 supports: external WebSphere MQ, a third-party JMS provider, or the WebSphere V5-based embedded JMS provider.)
* High Availability Manager (HA Manager) is used to provide high availability for the following services:
  + the Workload Management (WLM) service —provided on all managed processes in a defined core group (in earlier WebSphere versions, this service was available only in the Deployment Manager process)
  + rapid recovery of transaction logs if a cluster member fails
  + new SIBus messaging engine components that provide JMS services on each application server
* A more optimized design of the communications layer enables improved performance, high availability, and failure recovery of Data Replication Service, or DRS (used for memory-to-memory replication between JVMs in a cluster).
* V6 supports replication and failover of stateful session EJBs.
* Service Data Objects (SDO) provide unified data access and representation across heterogeneous back-end data stores and simplify the application programming tasks required to access data.
* WebSphere V6 implements the Java Authorization Contract for Containers (JACC) specification and includes an integrated Tivoli Access Manager client for enhanced security.