# Using WebSphere Application Server on UNIX

Ade Rixon

April 21, 2005

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Notices</td>
<td>2</td>
</tr>
<tr>
<td>2 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Other useful documents</td>
<td>3</td>
</tr>
<tr>
<td>3 Requirements</td>
<td>4</td>
</tr>
<tr>
<td>3.1 Platform</td>
<td>4</td>
</tr>
<tr>
<td>3.2 Installation media</td>
<td>5</td>
</tr>
<tr>
<td>3.3 Licensing and support</td>
<td>5</td>
</tr>
<tr>
<td>3.4 WebSphere fix pack</td>
<td>5</td>
</tr>
<tr>
<td>3.5 WebSphere application code</td>
<td>5</td>
</tr>
<tr>
<td>3.6 Other requirements</td>
<td>6</td>
</tr>
<tr>
<td>4 Preparation</td>
<td>6</td>
</tr>
<tr>
<td>4.1 Platform resources</td>
<td>6</td>
</tr>
<tr>
<td>4.2 Web servers</td>
<td>6</td>
</tr>
<tr>
<td>4.3 Database server</td>
<td>7</td>
</tr>
<tr>
<td>4.4 Operating system</td>
<td>7</td>
</tr>
<tr>
<td>5 Installation</td>
<td>8</td>
</tr>
<tr>
<td>5.1 Messaging service (MQ)</td>
<td>8</td>
</tr>
<tr>
<td>5.2 Overview</td>
<td>8</td>
</tr>
<tr>
<td>5.3 Installing WebSphere Application Server</td>
<td>9</td>
</tr>
<tr>
<td>5.4 Installing WebSphere Application Server fix pack</td>
<td>10</td>
</tr>
<tr>
<td>5.5 Installing the HTTP plugin</td>
<td>10</td>
</tr>
<tr>
<td>5.6 Initial WebSphere state</td>
<td>11</td>
</tr>
<tr>
<td>6 Starting and stopping WAS</td>
<td>12</td>
</tr>
</tbody>
</table>
1 Notices

WebSphere and AIX are trademarks of International Business Machines Corporation.

Solaris is a registered trademark of Sun Microsystems Inc.

All other products mentioned are trademarks of their respective owners.
2 Introduction

Every effort has been made to ensure that the information in this document is accurate and correct. However, no responsibility is accepted for misunderstandings, damage or losses of any kind resulting from the advice given, or for omissions or errors contained herein. For full and approved product support, contact the product vendor.

2 Introduction

This document describes how to install and use IBM WebSphere Application Server (WAS) on UNIX systems (chiefly Sun Solaris but most of the information will be generally applicable). It is aimed at UNIX system administrators who must support WebSphere in an enterprise or production context.

It assumes that you are generally familiar with UNIX system administration, TCP/IP networking and the operation of any additional components you may require, such as web servers and database management systems.

The goal of this document is get you from A to B along the quickest, simplest route, where 'A' is a UNIX system and a WebSphere installation CD and 'B' is an installed WebSphere application server with a working web application. It deliberately avoids explaining J2EE technology, discussing the many implementation scenarios and other related subjects unless such information is explicitly required to complete a task. In particular, a full and immediate understanding of J2EE terms and acronyms such as EJB, JMS, JDBC, etc., often distracts from implementation of the practical issues involved. When you need to know about these topics, refer to the various IBM publications given in the bibliography.

Large scale WebSphere deployments (i.e. the Network Deployment product) are not described in detail, although there is an overview of the relevant issues towards the end.

This document refers specifically to WebSphere Application Server V5.1. Some of the information will also be valid for WebSphere Application Server V5.0 and V6.0. Previous major releases of WebSphere differ markedly from the V5.1 release and are quickly becoming obsolete.

Note that the configurations chosen here are not the only options and may not be correct or optimal for your situation. However, they have been proven to work and have been tested in production environments.

2.1 Other useful documents

This document was written partly in response to the lack of a direct, unambiguous guide to installing WebSphere on UNIX. However, while it is designed to get you
3 Requirements

up and running quickly, it cannot cover every aspect. You should expect to consult other sources where necessary.

The WebSphere Application Server Information Center is the complete, definitive guide to the product. It is supplied by IBM in HTML and PDF formats, either online (see section 13) or as part of the product media set.

IBM’s International Technical Support Organisation (ITSO) also publishes a number of “Redbooks”, which cover specific aspects of implementation. Of these, the most useful for WebSphere administrators is *IBM WebSphere Application Server V5.1 System Management and Configuration*. It includes a platform-specific chapter for installation on AIX; this information is also relevant to other UNIX-based systems. (There is a related Redpaper dealing specifically with Linux, *Installing WebSphere Application Server Version 5.1 on RedHat Enterprise Linux Advanced Server Version 3.0*.) The Redbooks can be downloaded from IBM’s web site.

Of these two main sources, the Redbook is easier to follow and more immediately helpful than the Infocenter. The latter makes heavy use of hyperlinking, which in the author’s view makes following a chronological sequence more difficult. The Infocenter is, however, much more detailed and comprehensive, fully searchable and useful as a reference.

IBM Press now publishes several books about WebSphere that cover administration and deployment topics more approachably than the official documentation. These are listed in the bibliography.

3 Requirements

This section covers the necessary hardware, software and site information required before proceeding with WAS installation and deployment.

3.1 Platform

You will need at least one UNIX server with an installed operating system listed by IBM in the *Prerequisites for WebSphere*.

Capacity planning and WebSphere topology decisions are outside the scope of this document.

Ensure that your platform is up to date with all recent recommended patches and fixes before beginning. The server must also be connected to a suitable TCP/IP-based network. If your WebSphere system will be part of a larger site that includes web servers and database servers, these other systems should be installed ready for use in advance. The configuration of these systems is not covered here.
The examples in this document were taken from a Sun Enterprise 280R server running Solaris 8.

### 3.2 Installation media

You will require the IBM-supplied CD-ROM (or download) labelled “WebSphere Application Server and IBM HTTP Server” for your platform.

### 3.3 Licensing and support

It is your responsibility to ensure that you have made the appropriate licensing and support arrangements for WebSphere with IBM or one of its authorised agents.

WebSphere Application Server does not require a license key or registration.

### 3.4 WebSphere fix pack

Download Fix Pack 1 (FP1) for WebSphere Application Server V5.1 Base from IBM’s WebSphere support site. This will update the installed product to release V5.1.1.

If a later fix pack is available, download it instead. You are unlikely to require any cumulative or individual fixes initially, unless your development team specifically notes any dependencies for the application.

### 3.5 WebSphere application code

You will also require a build of the application(s) you will be running under WebSphere. WebSphere applications are normally deployed via Enterprise Application Repository (EAR) files, which are archives containing all the required files.

EAR files are built and supplied by application developers. The development of WebSphere applications is outside the scope of this document.

Along with the application, the developers should also supply a list of supplementary requirements such as:

- Databases, including schemas and initial contents;
- Messaging services and queues;

---

1EAR files, like most other J2EE archives, are actually in ZIP format and can be unpacked and viewed using the `zip/unzip(1)` utilities.
4 Preparation

- Key URIs, such as home pages, entry points, administration pages, etc.;
- Minimum CPU, memory and disk requirements;
- Any additional required setup (e.g. logging configurations, static content files, supporting utilities, URI aliases, etc.)

If this is not available, you may need to ask for more information during the deployment. Generally, deployment of WebSphere applications is eased if you have access to the developer(s).

3.6 Other requirements

The WebSphere graphical installer requires X Windows. You must be able to view X11-based programs running on the host(s) on which you will be installing WAS. (A text-based, ‘silent’ installation method is also available but is not covered here.) Either a local graphics monitor or a remote X11 server/emulator such as Cygwin/X will suffice.

For administration of WebSphere, a supported web browser is required (generally Internet Explorer or a Mozilla variant).

4 Preparation

This section covers the preparation of other required resources prior to installing WAS.

It is possible to install supporting products on the same system as WAS. This can often reduce the hardware requirements in test and development environments. However, it is usually undesirable on production sites for performance, scalability and security reasons.

4.1 Platform resources

Ensure that each WebSphere server meets the minimum CPU, memory and disk space requirements given by IBM for running WebSphere V5.1.

If possible, dedicate a separate, large disk partition to application log data.

4.2 Web servers

Most J2EE applications are web-based. You are therefore likely to require a web server or web serving infrastructure (such as a load balancer farm). While you
4 Preparation

can access applications on WebSphere directly using HTTP or HTTPS, this is not recommended for production use as WebSphere contains no facilities for request logging or access control, nor will it perform as well as a dedicated web server.

Read IBM’s list of supported web servers for WebSphere before selecting a suitable product. The web server communicates with WAS using a component called the HTTP Plugin that is supplied with WebSphere. The plugin is available in a number of forms to support different web server products. It can support:

- EAPI-compatible products (based on Apache, including IBM HTTP Server)
- NSAPI-compatible products (Sun ONE and others)
- ISAPI-compatible products on Windows platforms (e.g. Microsoft IIS).
- Lotus Domino web server.

Generally, providing your web server supports an API compatible with at least one of these products, it should work.

Before proceeding, install and configure your web server, including IP addresses, names, DNS entries and secure certificates where required. Ensure that it has network access to your WAS nodes.

4.3 Database server

Many J2EE applications use a database at the backend to store application and session data, audit information, etc. Databases are typically accessed over TCP using JDBC-compliant drivers, although WebSphere supports a number of other methods.

Some databases require the installation of client software on the WebSphere node, to access a remote server. For example, for DB2 V7 servers, you must first install the DB2 client on the WebSphere node and catalog the remote DB2 nodes and databases. WebSphere uses JDBC to communicate with the local DB2 client, which in turn forwards the queries to a remote DB2 server via the network. (Such configuration is outside the scope of this document.)

Before proceeding, install and configure any required database management system, and create & initialise the databases required by your applications. Ensure that remote access to the DBMS is possible from the WAS system.

4.4 Operating system

If you are not using a naming service such as DNS or LDAP:

1. Add the hostnames and IP addresses of any supporting servers to the `/etc/hosts` file on the WebSphere node(s).
5 Installation

2. Add the hostnames and IP addresses of the WebSphere node(s) to the /etc/hosts files on the supporting servers.

If you are using a naming service, add entries for all the hostnames and IP addresses.

5 Installation

This section covers the installation procedures for WAS base edition and the fix pack.

5.1 Messaging service (MQ)

The J2EE 1.3 standard specifies that a messaging service must be available to applications. IBM meets this requirement in WebSphere by bundling a version of its MQ Series software with the product, also known as WebSphere Embedded Messaging or the WebSphere JMS provider. The applications you are installing will not necessarily require this feature; confirm this with the developers before installation.

IBM’s installation instructions and examples assume that you will install the MQ components, which require the creation of dedicated user and group accounts and some kernel configuration. However, this is an option and is unnecessary if your applications do not use a messaging service. Although the MQ components are selected for installation by default, it is safe to deselect them if they will not be needed. By doing so, you will avoid additional effort and overhead.

(It is possible to install the MQ components separately after installing WebSphere, should they be required at a later stage. There does not appear to be a supported method for uninstalling the MQ components separately from the rest of WebSphere.)

In this document, we assume that you will not install the MQ components. If this is not the case, read the relevant Infocenter sections carefully before proceeding.

5.2 Overview

We will perform a custom installation of the WAS Base product. During installation, we will ensure that only the required components are selected and any others are not installed. After installation, we will also install the most recent fix pack to update the product with later fixes from IBM.
5.3 Installing WebSphere Application Server

1. Insert the WebSphere CD-ROM into the drive on the server and mount it. (If this is not possible, either mount the CD-ROM via NFS from a remote system or recursively copy the contents over the network to a directory on the server, using rsync(1) or a similar utility.)

2. In the mounted CD-ROM directory, change to the subdirectory for your platform (e.g. sun/).

3. Run the install script as root:

   # ./install

4. The installer first checks the current OS patch levels. It may warn about missing or obsolete patches; however, these may not be essential to WAS operation (e.g. window manager and X11 patches are not required if you do not run a desktop environment on the system). If in doubt, cancel the installation and update the patches.

5. Select Custom setup.

6. As a minimum, select the following components:
   
   - Application Server;
   - Administration & all subcategories;
   - Performance And Analysis Tools & all subcategories.

   Note that WebSphere will only run under Java SDK releases bundled or supplied by IBM.

   If you have a web server installed on this system, also select ‘Web server plug-ins’ and indicate the type of server. If you will be using the IBM HTTP Server version 1.3 on this server, select that component and ensure that the Plug-in for IHS 1.3 is also selected.

   If you wish to test WebSphere with a demo application, select ‘Samples’. Otherwise, do not install them.

   Deselect all other features, including subcategories not listed above.

7. Accept the default path for the installation directory unless you have a different requirement (/opt/WebSphere/AppServer on Solaris).

8. Accept the default node name and host name.

9. Review your choices and proceed with the installation.

10. Registering the product is an optional step.

11. After installation, use the launch pad tool to start the server and verify its operation. Do not launch the administration console from this window.
If you are installing IHS 1.x on a separate system, run the installer again there and select the IHS and plugin components only.

### 5.4 Installing WebSphere Application Server fix pack

1. If the default server is running (e.g. because you started it using the launch pad during the installation), stop it using the command:
   ```bash
   # /opt/WebSphere/AppServer/bin/stopServer.sh server1
   Wait for the command to complete.
   ```
2. Unpack the fix pack in an empty directory using `tar(1)`:
   ```bash
   $ gzip -dc was51_fp1_solaris.tar.gz | tar -xvf -
   ```
3. Become superuser and source the WAS setup script:
   ```bash
   # /opt/WebSphere/AppServer/bin/setupCmdLine.sh
   ```
4. If your WAS installation is already configured, back up its current state as a precaution, using the command:
   ```bash
   # /opt/WebSphere/AppServer/bin/backupConfig.sh
   ```
5. Run the update wizard:
   ```bash
   # ./updateWizard.sh
   An X window will appear asking you to select the language for the installation. (Note that the installer may be sluggish on older hardware, with long delays between screens.)
   ```
6. Continue past the initial information dialogue.
7. Select IBM WebSphere Application Server v5.1 from the product list and hit **Next**.
8. Select **Install fix packs** and hit **Next**.
9. The correct fix pack directory path should already be displayed. Hit **Next**.
10. Select the v5.1.1 fix pack (`was51_fp1`).
11. Proceed with installation of the fix pack. Do not interrupt the update.
12. Once installation is complete, exit the update wizard. The unpacked archive files can now be removed.

The WAS command script `versionInfo.sh` (in the `bin/` directory) can be used to verify the current release and fix level of the product.

### 5.5 Installing the HTTP plugin

The HTTP plugin is a shared object that links into the web server to handle browser requests for WAS applications. If the web server is running on the same
5 Installation

host as the application server then you can install it while installing WAS. If the web server is separate, then you must also run the WAS installer on it as above.

Different versions of the plugin are supplied for different web servers. Use a version that is compatible with your web server software. Some less common web servers support third party APIs such as the Netscape one (NSAPI) or the Microsoft IIS one (ISAPI); in these cases, use the plugin version that is most compatible with your server (e.g. use the iPlanet plugin for the Zeus Web Server, since it supports NSAPI).

This procedure assumes you are installing the plugin on a separate web server. However, the dialogues described are identical to those that appear when installing the plugin as part of WAS base.

1. Select custom installation.
2. Select the appropriate HTTP Plugin and any required components.
3. You will be prompted for the path to the web server configuration file (e.g. httpd.conf for Apache/IHS). If this file does not exist yet, create an empty file with the correct name and give the path to that (e.g. /tmp/obj.conf for iPlanet/NSAPI). You can configure the web server manually later, using the temporary file as an example.

After installation, also apply the WebSphere Application Server fix pack to update the plugin. Keep the plugin fix level synchronised with the application server fix level.

5.6 Initial WebSphere state

By default, WAS is installed with a single application server instance called server1. This server runs the administration console application (‘adminconsole’).

WAS topology and configuration is hierarchical. At the highest level is the cell. (The administration console can only manage a single cell.) A cell contains one or more nodes, representing physical servers. Each node may contain one or more application servers. In the base edition, the cell can only contain a single node and both are named after the system hostname. However, you may create multiple servers under the node. (WebSphere Application Server Network Deployment is used to manage multiple nodes in a distributed cell.)

Certain aspects of WebSphere can be configured at any one of these levels; this is called configuration scoping. Duplicate entries in lower levels override those in higher levels, allowing you to create cell-wide ‘default’ settings that can be overridden for individual nodes and servers.
Warning: On installation, WAS contains several default settings in the node scope. Some of these settings have empty values. Nevertheless, they will override any similar settings created manually in the cell scope unless they are deleted first. To avoid confusion, we will make all our initial configuration changes in the node scope, which is the default.

WAS also initially defines two virtual hosts that map the destination IP/port number pairs (host aliases) of incoming web requests to applications within the cell. These virtual hosts are called default_host and admin_host. The latter is used by the adminconsole application and maps requests for the host alias *:9090 (i.e. any IP address on TCP port 9090) to the console. The former can be used by your own applications. Assuming WAS was installed on a fresh system, it defines host aliases for *:9080 and *:9443, which are the default ports for server1.

6 Starting and stopping WAS

Application servers in the base edition are started and stopped individually using the startServer.sh and stopServer.sh scripts in the WAS bin/ directory.

For example (assuming the WAS bin directory is in your PATH):

```bash
# startServer.sh server1
ADMU0116I: Tool information is being logged in file  
/opt/WebSphere/AppServer/logs/server1/startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.  
ADMU3000I: Server server1 open for e-business; process id is 12995
```

The server1 instance must be running for you to manage WAS base edition, as it contains the adminconsole application.

You may wish to write a single wrapper script for the various WAS command scripts. Such a script is useful for controlling WAS during system boot and shutdown. Here is an example:

```bash
#!/bin/sh
#
# script to start and stop WebSphere Application Server 5.x
#
WAS_HOME="/opt/WebSphere/AppServer" # WAS install dir
SERVERS="server1 MyAppServer" # list of app servers

if [ ! -d "${WAS_HOME}" ]; then
    echo "$0: ${WAS_HOME} does not exist, aborting" >&2
    exit 1
```

```bash
```

12
7 Configuring WebSphere Application Server

To administer WAS, you must connect to the Administration Console on the node using a web browser. The default URL will be http://hostname:9090/admin/ (Use the app server hostname directly, do not go via the web server.)

You will first be prompted to enter your user name. When WebSphere Security is not enabled, this user name has no special meaning and a password is not required. You can use any name, although you should pick something unique to you so that different console users can be identified.

The normal admin console display contains four frames: a top banner; a left hand side menu of options that can each be expanded; a main frame showing the current task; and a bottom bar showing a status summary.

The left hand expanding menu lists the various aspects of WebSphere that can be configured. Configuration dialogues appear in the main frame. After changing any values, remember to select Apply or OK before proceeding further.
After making a set of configuration changes to WAS using the console, you must save these changes to the repository before they will have any effect. A warning dialogue appears in the main frame to remind you of this; it contains a link to the save screen. Unsaved changes are lost when you leave the console. WAS application servers usually require restarting to pick up changes.

The administration console can only be used to control the default server1 application server in a base installation, although it can configure any server. If you create other application server instances, you must start and stop them manually using the command scripts.

### 7.1 Information from the developers

Before proceeding, you must gather some information about your application(s) from the developers:

- Database(s) required: schema; purpose; initial contents; etc.
- Any JNDI resource names used by the application to access databases or other external data sources. E.g. jdbc/MyAppDataSource.
- Any additional external resources required by the application. These may include remote web services, mainframe access or other remote data servers. Ensure that your WAS hosts have access to these resources.
- Any aspect of application configuration that may differ in a production environment compared to a development or test environment. (Note that developers often fail to recognise such issues as they lie outside their experience.) Absolute URLs and other information that may differ between environments should not be hardcoded into the application; place them either in a database, a Java properties file or a WebSphere variable or URL Provider.

### 7.2 Preparation

Create and initialise any databases required for the application. Configure any database client software on your application server so that it can connect to the database server if it is remote.

Although it is possible to install your application into the default server1 application server, we are going to create a separate server to enforce isolation from the administration console application.

It is helpful to develop a consistent naming strategy for the following elements in advance:

---

\(^2\)WAS stores all configuration data in a tree of XML files called the repository, held under the config/ directory.
7 Configuring WebSphere Application Server

- J2C authentication aliases for JDBC resources (e.g. MyDatabaseAuth)
- Data sources (e.g. MyDatabaseDataSource)
- Application servers (e.g. MyAppServer)

Names should be single or combined words with no whitespace, using underscores or upper case letters as word delimiters.

7.3 Administration console configuration

1. Using the left hand bar menu, go to Environment -> Manage WebSphere Variables. Ensure that the current scope is at node level in the dialogue box at the top of the main frame.

   If you are using a database with a recognised JDBC driver, set the path to the driver in the relevant variable. For example, if you are using DB2 V7, set DB2_JDBC_DRIVER_PATH to the path to your DB2 Java directory (e.g. /opt/IBMdb2/V7.1/java12).

   Examine the other defined variables, noting which ones have default values. Set any empty variables that appear relevant to your installation. Note that variable listings can spread across multiple pages, which are indicated at the top of the frame.

   Remember to click Apply after entering a value.

2. Go to Environment -> Virtual Hosts. Select default_host, then Host Aliases. Add the aliases in table 1. (Adding port 443 allows SSL/HTTPS to be used to access the application.)

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>443</td>
</tr>
<tr>
<td>*</td>
<td>9081</td>
</tr>
<tr>
<td>*</td>
<td>9444</td>
</tr>
</tbody>
</table>

   Table 1: Additional host aliases

3. JDBC resources usually require authentication details for access. These are configured under Security -> JAAS Configuration -> J2C Authentication Data. For each set of authentication values, you must supply an alias (name), user ID, password and optionally a description.

   For example, for the DB2 CLI legacy client driver, the user name would be the DB2 client instance user, such as db2inst1.

4. To create the actual JDBC resources, go to Resources -> JDBC Providers. WebSphere knows about many common driver types and provides templates
for them, such as ‘DB2 Legacy CLI-based Type 2 JDBC Driver’ for DB2 V7. Otherwise, you may have to create a driver entry from scratch; consult the driver documentation for the required class path, environment, libraries and any other settings.

5. For each configured driver, create the necessary Data Sources. (Do not create Version 4 compatible data sources unless you will be running a legacy application that predates the J2EE 1.3 standard.)

Each data source requires a name and a JNDI name. The JNDI name should match the name used by the application (e.g. jdbc/UserDataSource). Set the Alias fields to the appropriate access alias defined under the J2C Authentication Data. Mapping-Configuration Alias should usually be left as DefaultPrincipalMapping.

Many data sources require certain Custom Properties. For example, a DB2 V7 CLI-based client data source requires the databaseName defining.

After saving the configuration, return to the data source, select the checkbox next to it and click Test Connection. Note: Testing the connection may fail if the local database client is not configured or any required environment variables were not defined when WAS was started.

6. Go to Servers -> Application Servers. Create a new application server and give it a unique name. Use the default/server1 template.

7. Select the new application server. Go to Web Container, then HTTP Transport. Confirm that there are entries for *:9081 and *:9444 only. If the entries differ, edit and correct them. Ensure that there is no conflict with an existing application server or other program.

8. There are several items under Process Definition that you may wish to alter. Under Logging & Tracing -> JVM Logs, you can increase the number of Backup files (old logs) retained and change the log rotation schedule. Under Java Virtual Machine, you can set the class path and heap sizes for the application server JVM (Java Virtual Machine). The class path may need setting if you need to use other libraries and properties files that are local to the server (i.e. not installed with the application). The JVM Working Directory is set to the WebSphere install directory by default; this is an extremely poor choice and should be altered immediately. Use either a new, dedicated directory or the WebSphere $USER_INSTALL_ROOT/temp/ subdirectory.

After making all your changes, remember to Save the configuration to the repository.
8 Deploying an application

Complete WAS applications are normally supplied as a single Enterprise Application Repository (EAR) file. EAR files are actually in standard ZIP archive format and can be unpacked manually using `unzip(1)` if required. Normally, the administration console unpacks the files in the correct location during deployment.

Before proceeding, copy the EAR file to the application server host.

1. Ideally, stop the relevant application server before deploying a new version of an existing application. (It can be left running, but you may see some odd behaviour when the updated configuration is saved.) WAS also provides a facility to update a running application; consult the Infocenter for details.

2. In the administration console, go to Applications -> Install New Application.

3. Select Server path and enter the path to the EAR file on the application server. Click Next.

4. If the application has been built correctly to match the details previously given by the developers (e.g. JNDI names, etc.), you should be able to select Generate default bindings and have most of the settings configured correctly.

5. The rest of the installation steps are described in section 8.6.6 of the System Configuration Redbook ([5]) and in the Infocenter. The most likely places where intervention will be required are:

   **Provide options to perform the installation:** Ensure that the Deploy EJBs option is selected. Enter a unique Application Name.

   **Map virtual hosts for web modules:** Select default_host for all modules.

   **Map modules to application servers:** Select your dedicated application server for all modules (use the Module checkbox), rather than the default server1.

   **Provide options to perform the EJB Deploy:** Select the correct database type and schema.

   Otherwise, the defaults can be accepted. You will have an opportunity to verify your selections on the final Summary page.

6. Save the updated configuration to the repository. This activates the new application code.

7. Restart the application server.

Do not install multiple copies of the same application on the same application server, as naming conflicts are likely to occur.


8 Deploying an application

8.1 Automated EAR installation

Because installing an application using the console is time-consuming and error-prone, it is preferable to develop a JACL script instead (see section 11.1). This reduces the number of steps involved to a single command and ensures that the procedure is always executed consistently.

The following brief example shows how to install an application contained in an EAR file called myapp.ear on to an application server called MyAppServer. The application uses DB2 V7.2 for the EJB data sources.

```
ear_install.jacl

puts "Installing MyApp on MyAppServer ..."
$AdminApp install myapp.ear MyAppServer [list -appname MyApp
 -usedefaultbindings -defaultbinding.virtual.host default_host
 -distributeApp -deployejb -deployejb.dbtype DB2UDB_V72
 -createMBeansForResources]

puts "Saving configuration ..."
$AdminConfig save
```

If this script is put into a file called ear_install.jacl and the EAR file is in the current directory, it can be run using:

```
# wsadmin.sh -f /path/to/ear_install.jacl
```

8.2 Generating the plugin configuration

After installing a new or modified application, it is necessary to update the HTTP plugin configuration file. In the administration console, go to Environment -> Update Web Server Plugin. Click OK.

If the web server is installed on the same host as the application server, this is usually all that is required. The plugin configuration file path will be set correctly in the web server configuration.

However, if the web server is separate then the plugin configuration file must be copied to it (and to any other identical web servers configured for redundancy or load balancing). The file is in /opt/WebSphere/AppServer/config/cells/- plugin-cfg.xml. For simplicity, it is easiest to use the same path on the web server.

It is common to customise the plugin configuration to suit local needs. For example, any mention of the adminconsole application or other sample applications should be removed from production systems. There are also some parameters, such as connection settings, that can only be edited manually. The file is XML-based and care must be taken when editing to ensure that its structure is pre-
served and the syntax remains correct. Be especially careful when copying and pasting lines from another file that spurious line breaks are not introduced.

If you have edited the plugin configuration, you may find it easier to compare your current file with a newly regenerated one and make any changes to it manually, to preserve your own edits.

The plugin configuration should be regenerated and/or edited when:

- a new application is installed;
- the structure of an existing application changes (i.e. new web application module or URI path);
- the network details of the cell are changed (new addresses or port numbers);
- an application server is added to or removed from a distributed cell.

### 8.3 Starting the application

It should now be sufficient to start the application server in which the application was installed:

```
# startServer.sh MyAppServer
```

Tail the `SystemOut.log` file for the server under the WAS logs/ directory and watch for any errors. If the application writes its own log file, examine that too. Typical problems are:

- Libraries not found (not installed or not in JVM class path), resulting in `ClassNotFoundException` exceptions.
- Data sources fail to bind correctly due to wrong configuration or environment, leading to further application failures.
- Database connections fail due to incorrect database client or server configuration. Such errors may have secondary effects if the application must retrieve data during initialisation.
- Insufficient JVM heap space configured for application requirements (possibly a sign of a poorly-written application!).

If all goes well, you should be able to enter the application web URL into a browser and receive a page back.

### 8.4 Debugging a problem

First, check the fundamentals:

- Can your browser connect to the web server? (View web server logs.)
8 Deploying an application

- Is WebSphere running?
- Are all the backend services up? (Database, etc.)
- Has the updated plugin configuration file been copied to the web server?
- If the configuration was altered, has the application server been restarted?

Check the WebSphere server logs and any application logs for obvious errors or failures. For example, the web server or application server TCP ports may not be included in the virtual host definition (commonly indicated by the error message 'Virtual Host/WebGroup Not Found'). The logs can be quite verbose, particularly if they contain Java stack traces; look for timestamps and keywords that match up with key error events. Exceptions often spawn multiple stack traces so be sure to scroll back until you locate the root cause. If you are debugging a specific application problem, it is easier to submit test requests singularly and observe the logs as you do so.

If there is no evidence that the application is receiving requests, verify network connectivity between systems using tools like `telnet(1)` (with a port number) and `curl(1)`. The latter utility is especially useful for debugging web applications, since it allows you to formulate arbitrary URL requests and view the full server responses, including cookies. You can use it to send HTTP requests to the web server or from the web server to the application server. For example:

```bash
$ curl -vi http://appserver:9081/MyAppURL/file.jsp
```

(An alternative to `curl` is the Live HTTP Headers extension for Mozilla/Firefox.)

If locally made requests on the application server host work, this may indicate a problem on the web server (perhaps with the plugin configuration) or between the web server and application server.

Remember that requests to the web server via HTTPS are forwarded to the application server via HTTPS on the alternate port (e.g. 9444).

`lsof(8)` is a useful utility for discovering applications that have a connection or file descriptor leak. Used against the application server JVM process, it will show all the open files and connections to JDBC resources, etc.

If the application appears to be malfunctioning, developers may find it useful to view a stack trace from the JVM running the application. The following command will cause such a trace to be written to the server's `native_stdout.log` file (without stopping the process):

```bash
# kill -QUIT \\
`cat /opt/WebSphere/AppServer/logs/MyAppServer/MyAppServer.pid`
```

WAS contains a number of utilities to assist with application debugging, including the Log Analyzer and the Trace facility. You can also enable performance monitoring (PMI) in the application server and use the supplied Tivoli Performance Viewer application (which can be installed separately on a remote host such as
9 Monitoring WAS

The immediate status of all WAS servers can be obtained using the command:

```bash
# serverStatus.sh -all
```

WAS processes can be listed using the following command (Solaris):

```bash
$ pgrep -f -l 'WebSphere.*java'
```

Current WebSphere process IDs are also written to files in the application server log directories (e.g. `MyAppServer.pid`).

The most effective way to monitor WebSphere and its applications is to request one or more known stable URLs directly from the application server. For example, either `curl` or the `check_http` plugin in the Nagios monitoring package can do this. You can optionally verify the contents of the response to be sure that the application is working correctly.

It is also useful to monitor the load (uptime) and memory usage on each system, as faulty application server processes often consume all the CPU time and/or virtual memory.

10 General tips for running WAS

- Run `backupConfig.sh` at regular intervals via `cron` on each installed WAS host. Ideally, copy each backup archive file to a remote system. Note that these files can be quite large, as they also contain a copy of every installed application.

- Review the section on WAS Security and check the issues raised against your local security policy (section 11.4).

- Install the Tivoli Performance Viewer and Log Analyzer components from WebSphere on a desktop system, such as your PC (you may need the Windows version of the software for this).

- If you are developing shell scripts to manage WebSphere, it is worth sourcing the WAS setup script at the start of them as it provides a number of useful environment settings. For example, deployed application files can be found under `$(WAS_HOME)/installedApps/$(WAS_CELL)/`.

- WebSphere should be properly tuned for good performance under intensive loads. See the bibliography for suitable references.
11 Advanced topics

This section contains a brief overview of other WebSphere features and issues, particularly those that are relevant to enterprise use.

11.1 WAS scripting

In addition to the administration console, WAS operations can be scripted using Java implementations of TCL and Python, called JACL and Jython. These scripts are run using the `wsadmin` utility (`wsadmin.sh`). Scripts configure and control WAS via various method calls on different JMX (Java Management eXtension) objects, known as Managed Beans (MBeans).

Unfortunately, there is no comprehensive documentation on all the possible MBeans available (their number and types vary depending on the WAS topology and installation). The syntax and usage of the MBeans is rather verbose and unfriendly. The best – in fact, the only – way to learn how to perform tasks via scripting is by example. (While `wsadmin` has a help facility, its contents are rather terse.) There are a number of sample scripts in the Infocenter, the Redbooks and other online articles from IBM (see also [1]).

It is also difficult to debug faulty scripts other than by running `wsadmin` interactively and entering the individual commands, as script error messages do not include line numbers.

Despite these difficulties, scripting is very useful if you wish to configure and control a number of separate, identical WAS cells, or perform the same operations repeatedly on a consistent basis. Scripting also offers a path towards automated WebSphere deployments and recovery procedures.

It may be easier to tackle those tasks that are simplest to script, better documented and more frequently performed first (for example, installing EAR files), and gradually develop more complex scripts as you gain experience. A good TCL reference book is helpful.

Working with the `wsadmin` interface is easier if you have command line editing and history facilities. Although the program does not support this itself, the open source `rlwrap` utility can be used to add these features.

11.2 WebSphere Application Server Network Deployment

The base edition of WebSphere only supports a single node in a cell. WebSphere Application Server Network Deployment (WAS ND) is used to support and manage several nodes in a distributed cell. WAS ND is a separate product that can be installed either alongside the base edition or on a dedicated system. It provides
a component called the *WebSphere Deployment Manager* (DMgr), which is a specialised application server process. The DMgr has a web-based administration console very similar to the base edition, but with extra features for managing a cell containing more than one node. Distributed cells provide workload balancing, scalability and resilience improvements via the use of *clusters* (groups of application servers running the same application). For example, user sessions can be migrated from a failed application server to another one in the cluster using the *session persistence* feature.

Existing node base installations are *federated* or added into a DMgr cell. During this process, their own adminconsole application is removed and they are henceforth configured from the DMgr. To allow this, a separate, dedicated server called the *Node Agent* is created on the remote node. New application servers can be created or cloned from existing ones on any node in the cell. However, there is no facility for making identical changes to several application servers simultaneously.

When configuring WebSphere variables or JDBC providers, it is best to make these changes in the cell scope so that they are identical on every node and only configured in one place. Be sure to delete any duplicate entries at lower levels, such as the node scope.

If you are running critical applications in production environments, you should use at least two servers for resilience and performance reasons. In this case, you will need to identify a server suitable for the Deployment Manager. This can either be one of your existing WebSphere servers, a lightly used backend system (in a dual role) or even a dedicated system.

For maximum availability, the use of multiple identical cells is recommended. Each cell requires its own deployment manager and group of nodes, and must be managed separately. This allows individual cells to be taken down for maintenance or the staged implementation of fixes and application updates. See *WebSphere Deployment and Advanced Configuration* ([1]) for details.

### 11.2.1 Key differences in WAS ND

- EAR deployment is performed on the DMgr, which takes care of copying the files to each node.
- The console also contains configuration dialogues for the *dmgr* server and *node agents*.
- Network communication must be enabled between the node agents and the DMgr (see section [11.5]).
- The JDBC *Test Connection* option in the administration console only runs on the DMgr. This may or may not work, but it will not reliably indicate
whether the data source works on the application nodes. (It is possible to work around this by running the connection test on the nodes via wsadmin.)

11.3 Product xes

IBM regularly releases updates and xes for its products, including WebSphere. These can be downloaded from IBM's web site for WebSphere support; a support agreement or contract number is not required.

WebSphere xes come in three forms:

**Interim fixes:** These are fixes for individual problems or issues (referred to by IBM Support as Authorised Program Analysis Reports or ‘APARs’).

**Cumulative fixes:** These are bundled collections of fixes designed to be installed on a base release or fix pack. Each cumulative fix (CF) supersedes previous cumulative fix releases. It can be installed on any previous version of the related base and fix pack. Installing a cumulative fix updates the fourth number in the WAS version identifier (e.g. CF3 for Fix Pack 1 updates any V5.1.1.x release (where x < 3) to V5.1.1.3). Cumulative fixes for individual components, such as the Java Software Development Kit (SDK) and HTTP Plugin, are also made available separately.

**Fix pack:** A fix pack is a bundled collection of fixes and minor new features. It may include updates to the Java SDK used by the product and the bundled IHS release. Installing a fix pack overrides any previous fix pack or cumulative fixes. It can be installed on any previous version of the major product release. A fix pack updates the third number in the version identifier (e.g. FP1 updates V5.1.0.x to V5.1.1).

Each type of fix is released in several versions corresponding to the various supported WebSphere platforms, products and architectures. Ensure that you download the correct file for your hardware, operating system and product name. (Note that WAS ND xes are different to base edition xes.)

It is recommended that you install only the most recent fix pack for WebSphere V5.1. If you encounter problems that appear to be related to the product, you may then wish to install the latest cumulative fix relevant to the installed fix pack. In rare cases, you may require an interim fix to address a specific issue that is not covered by any of the bundled fixes.

At time of writing, you would download Fix Pack 1 (FP1) for WebSphere Base. If you are using WAS ND, you must also download FP1 for that product.
11 Advanced topics

11.4 WebSphere Security

By default, WebSphere is installed with inbuilt security features disabled. You can configure and control WAS without authentication – as indeed can anyone else with access to the administration console or host system. WebSphere Security is a complex topic; you must read the Infocenter, the Security Redbook[3] and all related documents carefully before enabling it. The following is an overview of the subject.

Once enabled, WAS Security encrypts some connections within cells using SSL and limits administrative access to defined username/password combinations authenticated against a defined registry. It can also be leveraged by applications for user access control and authentication, and internal security.

- WebSphere is installed with a default (‘dummy’) set of SSL certificates and keys. Ideally, these should be replaced with a new set generated with IBM's ikeyman utility before enabling Security. The certificate sets must be installed on each node in a cell and any web servers that use the HTTP Plugin to communicate with the cell.

- WebSphere can use either the local OS password file, an LDAP server or a custom registry to retrieve authentication details. Sample code is provided for a file-based custom registry, but IBM do not recommend its use in a production deployment.

- WAS ND can only use LDAP or a custom registry for authentication. All nodes in the cell must be able to access the registry.

- WebSphere does not support replicated LDAP servers (you can only specify a single LDAP server address). Any LDAP server you use must have a transparent high availability mechanism.

- Once Security is enabled, a user name and password must also be given as arguments when using wsadmin or the command scripts. If you write shell scripts around these to be run unattended, you will have to embed the user name & password details in them. Obviously, this makes such information less secure (although user actions can be restricted to one of four defined roles).

- WebSphere products install with global file read permissions. However, many of the files within them should be protected from general user access, including the configuration repository (config/), the SSL key files (etc/) and possibly the logs (logs/). This requires changing the default file permissions manually.

- If the HTTP plugin configuration file is automatically generated, it will include the URI for the administration console. You do not want this to be accessible on a production web server: copy and edit the file to remove it,
and include an ACL rule on your web servers blocking all external access to 
/admin/*.

It is worth considering whether careful firewalls and IP access controls that 
limit connections to a small defined set of addresses can effectively control ad-
ministrative access without the need to enable WAS Security, with its attendant overheads and complexities. This is particularly true if your application uses its own authentication system rather than relying on WebSphere to provide security.

11.4.1 Recommendation for minimum security

The following configuration enables the minimum amount of WAS Security, mainly 
to protect the administration functions, without requiring significant overhead. It may not be sufficient for production environments.

- Using the file-based custom registry example, create a flat file containing defined users for the Administrator and Operator roles. Use the former for the Administration Console and the latter for scripts that control WAS. Copy the file to all nodes in the cell and ensure that only the user ID under which WAS runs has read access.
  If you have an existing resilient LDAP infrastructure, you may wish to use that instead.
- Enable WAS Global Security and configure the registry details for the chosen method.
- Run WAS processes under a non-root account with limited access rights and privileges. (See Infocenter topic “Running the application server with a non-root user ID”. This can also be done for the WAS ND Deployment Manager.)
- Limit access to the WAS configuration, application and log files using normal UNIX file permissions and ownerships.

11.5 WebSphere and Firewalls

WebSphere Application Server on a single node is reasonably simple to firewall; just ensure that the web server can reach the application server HTTP and HTTPS ports, and the application server can reach any backend services such as databases. Production application servers and web servers should ideally be on separate, dedicated network segments in demilitarised zones (DMZs).

In a distributed cell with the deployment manager firewalled from the application nodes (recommended), a number of ports must be opened up between the two. Regrettably, this includes the default administration console port (9090), since it is shared with the filetransfer application that the application servers use to pull
data. Hence you may still need WebSphere Security to protect the console (e.g. in
the event of the application servers being compromised).

Table 2 shows the TCP ports that must be opened between node agents and the
DMgr for management and initial node federation, together with the default port
numbers following a standard installation. You should confirm the port assign-
ments by examining the serverindex.xml files in the WAS configuration repos-
sitory, or the details for each server in the administration console. Additional ports
may be required when WAS Security is enabled.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMgr</td>
<td>Nodes</td>
<td>BOOTSTRAP(2809)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCOVERY(7272)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORB(9900)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP(8878)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP(8880) for server1</td>
</tr>
<tr>
<td>Nodes</td>
<td>DMgr</td>
<td>BOOTSTRAP(9809)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCOVERY(7277)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORB(9100)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP(8879)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILETRANSFER(9090)</td>
</tr>
</tbody>
</table>

Table 2: Default WAS TCP port connections

12 Conclusion

You should now have a working single application server and some knowledge of
more advanced configurations. To go further, you must rely on the documentation
made available by IBM, which is hopefully now more approachable.

Remember that WebSphere, perhaps more so than other applications, requires
a close partnership between developers and system administrators. If you en-
counter problems, it is often productive to tackle them together, since developers
have a good understanding of Java and the application code while administra-
tors possess an appreciation of the bigger picture and the interdependencies with
external services.

13 Online resources

- IBM WebSphere Information Center:
  [http://www-306.ibm.com/software/webservers/appserv/was/library/](http://www-306.ibm.com/software/webservers/appserv/was/library/)
### 14 About the author

Ade Rixon is a UNIX System Administrator working in the UK on e-commerce and related systems. He has a personal web site, Big Bubbles (no troubles) at [http://www.big-bubbles.fluff.org/](http://www.big-bubbles.fluff.org/) and he can be emailed as info@big-bubbles.fluff.org.

While he welcomes short queries and comments about this document, he regrets that he cannot offer product support.

This document was written using LATEX2e.

*Text © A. J. Rixon 2005*

### References

References


Document history

$Log: websphere-unix.tex,v $
Revision 1.4 2005/04/21 16:20:39 ajr
Added debugging info
Improve URL display

Revision 1.3 2005/04/17 12:16:38 ajr
Corrections, clarifications and additional info
Add tips section
Reference tables explicitly
Init script is inlined
Updated email address
Improve app deployment steps

Revision 1.2 2005/04/13 12:50:18 ajr
First release

Revision 1.1 2004/12/08 14:12:06 ajr
Initial revision