Sun Java Enterprise System 5 Installation Planning Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–5079–10
February 2007

070226@16599

Contents

Preface

Installing a Sun Java™ Enterprise System (Java ES) solution is an extended process. In a typical solution you install Java ES components on several networked computers, and then configure component instances that interoperate. This book, *Sun Java Enterprise System Installation Planning Guide*, describes how to analyze a Java ES architecture and develop a plan for installing it.

Who Should Use This Book

Installation planning is one stage of the Java ES solution life cycle. The *Installation Planning Guide* assumes that the earlier stages of the life cycle covered in *Sun Java Enterprise System Deployment Planning Guide* have been completed and the high-level technical description of the solution known as the deployment architecture has been developed.

The *Installation Planning Guide* is for the person who analyzes the deployment architecture and determines how the solution is installed and configured.

Before You Read This Book

The *Installation Planning Guide* does not assume that one person will carry out all stages of the solution life cycle. The person who develops an installation plan should have knowledge of the following:

- General knowledge of the components that make up the Java Enterprise System and the services provided by each component. For more information, see "Java ES Components" in *Sun Java Enterprise System 5 Technical Overview*.
- Thorough understanding of IP networking, including network addressing, the use of load balancing hardware or software, the use of firewalls for securing networks, and setting up DNS servers.
- Thorough knowledge of the operating system platform on which you are installing the solution, including installing the operating system, assigning network addresses, and configuring storage devices.
- General knowledge of the Java ES installer. For more information, see "How the Java ES Installer Works" in *Sun Java Enterprise System 5 Installation Guide for UNIX*.

- General knowledge of LDAP directories.
- Sufficient knowledge of hardware to estimate the disc space requirements for the solution.

You may find that more than person is needed to develop the installation plan. For example, the person with primary responsibility for the plan might need to consult with an LDAP expert to develop some of the information required to install and configure a solution.

How This Book Is Organized

Chapter 1 provides an overview of the installation planning process.

Chapter 2 describes how to develop additional information, not included in the deployment architecture, that is needed to install a Java ES solution.

Chapter 3 describes installation planning in general, and then describes how to develop an installation plan for your specific Java ES solution.

Java ES Documentation Set

The Java ES documentation set describes deployment planning and system installation. The URL for system documentation is `http://docs.sun.com/coll/1286.2`. For an introduction to Java ES, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Java Enterprise System Documentation

Document Title	Contents
Sun Java Enterprise System 5 Release Notes for UNIX	Contains the latest information about Java ES, including known problems. In addition, components have their own release notes.
Sun Java Enterprise System 5 Technical Overview	Introduces the technical and conceptual foundations of Java ES. Describes components, the architecture, processes, and features.
Sun Java Enterprise System Deployment Planning Guide	Provides an introduction to planning and designing enterprise deployment solutions based on Java ES. Presents basic concepts and principles of deployment planning and design, discusses the solution life cycle, and provides high-level examples and strategies to use when planning solutions based on Java ES.
Sun Java Enterprise System 5 Installation Planning Guide	Helps you develop the implementation specifications for the hardware, operating system, and network aspects of your Java ES deployment. Describes issues such as component dependencies to address in your installation and configuration plan.

TABLE P–1 Java Enterprise System Documentation *(Continued)*

Document Title	Contents
Sun Java Enterprise System 5 Installation Guide for UNIX	Guides you through the process of installing Java ES on the Solaris Operating System or the Linux operating system. Also shows how to configure components after installation, and verify that they function properly.
Sun Java Enterprise System 5 Installation Reference for UNIX	Gives additional information about configuration parameters, provides worksheets to use in your configuration planning, and lists reference material such as default directories and port numbers.
Sun Java Enterprise System 5 Upgrade Guide for UNIX	Provides instructions for upgrading Java ES on the Solaris Operating System or the Linux operating environment.
Sun Java Enterprise System 5 Monitoring Guide	Gives instructions for setting up the Monitoring Framework for each product component and using the Monitoring Console to view real-time data and set threshold alarms.
Sun Java Enterprise System Glossary	Defines terms that are used in Java ES documentation.

Related Books

The manuals most likely to help you develop an installation plan are the following:

- *Sun Java Enterprise System 5 Technical Overview* describes the Java ES components and the services they provide.
- *Sun Java Enterprise System Deployment Planning Guide* describes how business needs are analyzed to develop a deployment architecture.
- *Sun Java Enterprise System 5 Installation Guide for UNIX* describes how to operate the Java ES installer.
- *Sun Java Enterprise System 5 Installation Reference for UNIX* includes a complete list of Java ES installer input values.
- Deployment planning guides for individual components, such as *Sun Java System Communications Services 6 2005Q4 Deployment Planning Guide*contain detailed information about configuring the components.
- For a complete list of terms that are used in this documentation set, refer to the *Sun Java Enterprise System Glossary*.

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–2 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file.

Shell Prompts in Command Examples

The following table shows default system prompts and superuser prompts.

TABLE P–3 Shell Prompts

Shell	Prompt
C shell on UNIX and Linux systems	`machine_name%`
C shell superuser on UNIX and Linux systems	`machine_name#`
Bourne shell and Korn shell on UNIX and Linux systems	`$`
Bourne shell and Korn shell superuser on UNIX and Linux systems	`#`
Microsoft Windows command line	`C:\`

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P–4 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional arguments and command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
`${ }`	Indicates a variable reference.	`${com.sun.javaRoot}`	References the value of the `com.sun.javaRoot` variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Searching Sun Product Documentation

Besides searching Sun product documentation from the docs.sun.com web site, you can use a search engine by typing the following syntax in the search field:

search-term `site:docs.sun.com`

For example, to search for “broker,” type the following:

```
broker site:docs.sun.com
```

To include other Sun web sites in your search (for example, java.sun.com, www.sun.com, developers.sun.com), use "sun.com" in place of "docs.sun.com" in the search field.

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-5079.

◆ ◆ ◆ CHAPTER 1

Introduction to Installation Planning

Sun Java Enterprise System 5 Installation Planning Guide describes how to prepare to install the Sun Java™ Enterprise System (Java ES) software. This chapter describes defines the scope and contents of the Installation Planning Guide. This chapter contains the following sections:

Java ES Components Used in This Release

The Java ES 5 release includes the following components. These components can be individually selected for installation.

Any alternate or abbreviated names used in this guide are in parentheses following the component name and version.

- Access Manager 7.1
- Application Server 8.2 Enterprise Edition + patches (Application Server)
- Directory Preparation Tool 6.4
- Directory Proxy Server 6.0
- Directory Server Enterprise Edition 6.0 (Directory Server)
- High Availability Session Store 4.4 (HADB)
- Java DB 10.1
- Message Queue 3.7 UR1
- Monitoring Console 1.0
- Portal Server 7.1
- Portal Server Secure Remote Access 7.1
- Service Registry 3.1
- Sun Cluster 3.1 8/05 (Sun Cluster software)
- Sun Cluster Agents 3.1
- Sun Cluster Geographic Edition 3.1 2006Q4 (Sun Cluster Geographic)

- Web Proxy Server 4.0.4
- Web Server 7.0

Note – HP-UX does not support Sun Cluster components, Directory Preparation Tool, HADB, or third-party web containers. Linux does not support Sun Cluster components, and only supports the IBM WebSphere third-party container.

Note – This guide also refers to components from the Sun Java System Communications Suite, which are often deployed with Java ES components.

Note – The Directory Preparation Tool is only used with Communications products, and is included with Directory Server in the Java ES release as a convenience. Information on the Directory Preparation Tool can be found in the Chapter 8, "Directory Preparation Tool (comm_dssetup.pl)," in *Sun Java Communications Suite 5 Installation Guide*.

Java ES Installation Defined

The installation process described in this manual includes the installation and basic configuration of a Java ES solution. Installation, as used in this manual, means using the Java ES installer to copy the files for Java ES components to computer systems. Configuration, as used in this manual, means using either the installer or a configuration wizard to configure an instance of a Java ES component. After you perform this basic configuration, you can start the instance, verify its basic operation, and verify that it interoperates correctly with other components in your solution.

The basic configuration described in this manual does not cover many areas of component functionality. For example, basic installation and configuration of Portal Server creates an instance that interoperates with other components, such as Access Manager and Directory Server. Basic configuration does not address other aspects of Portal Server functionality, such as adding your content to the basic portal desktop. To configure these aspects of component functionality, you need to refer to component documentation, such as *Sun Java System Portal Server 7.1 Configuration Guide*.

Installation Planning Tasks

The following table lists the installation planning tasks that are common to all Java ES solutions. The left column lists high-level tasks and subtasks, and the right column lists the location of instructions for performing the tasks.

TABLE 1–1 Installation Planning Tasks

Task	Location of Information
1. Develop Your Implementation Specifications	Chapter 2
Analyze your deployment architecture	"Analyzing the Deployment Architecture" on page 15
Develop a network connectivity specification	"Developing a Network Connectivity Specification" on page 18
Develop a computer hardware and operating system specification	"Developing Your Computer Hardware and Operating System Specification" on page 17
Develop a user management specification	"Developing Your User Management Specifications" on page 20
2. Learn About Installation and Configuration Issues	"Installation Planning Issues" on page 25
Learn how distributed installations affect an installation plan	"Distributed Installations" on page 26
Learn how configuring for component interoperation is part of an installation plan	"Configuring for Interoperation" on page 31
Learn how component dependencies affect an installation plan	"Component Dependencies" on page 26
Learn how the redundancy strategies used in a solution affect an installation plan	"Redundancy Strategies" on page 32
Lean how LDAP directory issues affect an installation plan	"LDAP Schema and LDAP Directory Tree Structure" on page 33
Lean how the installer operating modes affect an installation plan	"Java ES Installer Behavior" on page 34
3. Develop Your Installation Plan	"Developing Your Installation Plan" on page 38
Determine the order in which component instances should be installed and configured.	"Component Dependencies" on page 26
Determine the specific input values for each component instance.	"Configuring for Interoperation" on page 31

It is important to approach the installation planning tasks in an orderly way, following the methodology described in this guide.

◆ ◆ ◆ CHAPTER 2

Developing Your Implementation Specifications

The deployment architecture is a high-level technical description of your Java ES solution, and it does not have all of the information needed to install and configure the solution. This chapter describes the process of analyzing a deployment architecture and developing a set of implementation specifications. The reason you develop implementation specifications is to help you prepare the additional information you will need when you install and configure your solution.

This chapter describes the implementation specifications in the following sections:

Analyzing the Deployment Architecture

A typical deployment architecture is illustrated in Figure 2–1. This deployment architecture defines a Java ES solution that provides portal and communications services. This particular architecture uses Access Manager to provide single sign-on to the communications services, and it uses both Portal Server and Communications Express to deliver the messaging and calendar services to end users. This architecture includes components from the Communications Suite.

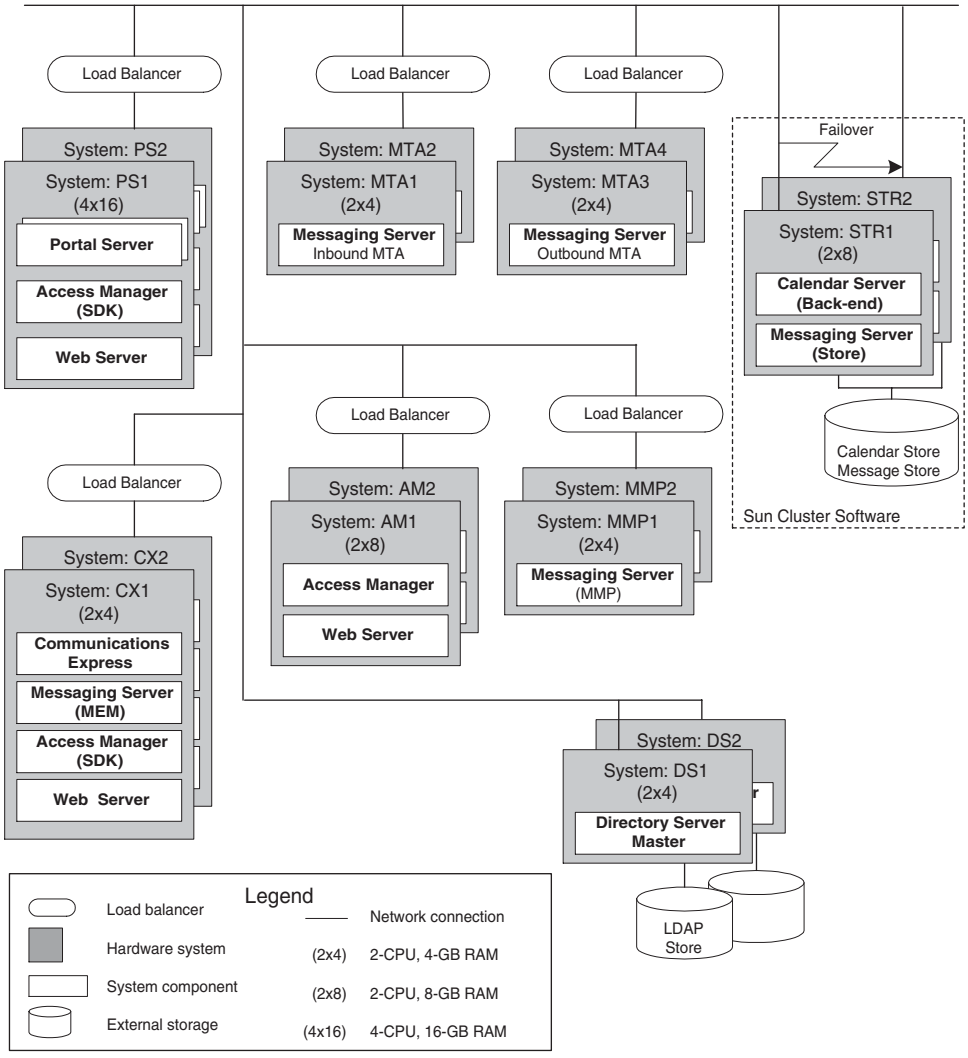

FIGURE 2–1 Example Deployment Architecture

Figure 2–1 contains much information about the solution, including the following:

- The number of computers used in the solution
- The number of CPUs and the amount of RAM required for each computer
- The component instances installed on each computer
- The number of instances of each component used in the solution

- The redundancy strategies used in the solution (load balancing, Directory Server multimaster replication, and Sun Cluster technology) to meet quality-of-service requirements
- The distributed installation of the Messaging Server subcomponents, another technique used to meet quality-of-service requirements

These characteristics of the example deployment architecture affect how the solution is installed and configured. You begin planning for your installation by analyzing your deployment architecture in the same way, observing how many computer systems are used, how many component instances are installed on each computer system, which redundancy strategies are used, and so on.

Developing Your Computer Hardware and Operating System Specification

In addition to the information that appears in the deployment architecture, you must specify the operating system that will be used on each computer used in your solution. You must also develop more information about the hardware on which you will install. Your decisions will be based on your quality of service requirements, and represent your best guess at the hardware and operating systems required to satisfy your quality of service requirements.

To meet the quality of service requirements for the deployment architecture shown in Figure 2–1the operating system and computer hardware specifications in Table 2–1 were developed.

TABLE 2–1 Computer Hardware/OS Specification for the Sample Deployment Architecture

Computer System	Hardware Model	Number of CPUs	RAM (in Gigabytes)	Number of Disks	Operating System
mscs01 mscs02	Sun Fire V440 Server	4	16	4	Solaris 9
commx01	Sun Fire V240 Server	2	4	2	Solaris 10
commx02				4	
ds01 ds02	Sun Fire V240 Server	2	8	4	Solaris 10
am01 am02	Sun Fire V240 Server	2	8	4	Solaris 10
ms-mmp01 ms-mmp02	Sun Fire V240 Server	2	4	2	Solaris 10

TABLE 2–1 Computer Hardware/OS Specification for the Sample Deployment Architecture *(Continued)*

Computer System	Hardware Model	Number of CPUs	RAM (in Gigabytes)	Number of Disks	Operating System
ms-mtai01 ms-mtai02	Sun Fire V240 Server	2	4	2	Solaris 10
ms-mtao01 ms-mtao02	Sun Fire V240 Server	2	4	2	Solaris 10
ps01 ps02	Sun Fire V440 Server	4	16	4	Solaris 10
protect	Sun Fire V240	2	4	2	Solaris 10

You must develop similar information about the computer systems used in your solution.

Tip – Once the Computer Hardware/OS specification is complete, the computer systems can be set up. Memory and disk drives can be installed, operating system can be installed, and the systems made ready for installation of Java ES components.

Developing a Network Connectivity Specification

The deployment architecture contains much of the information needed to connect all of the hardware used in a solution. To help you develop the additional information you need to connect your network, you need to prepare a network connectivity specification like the example in Figure 2–2.

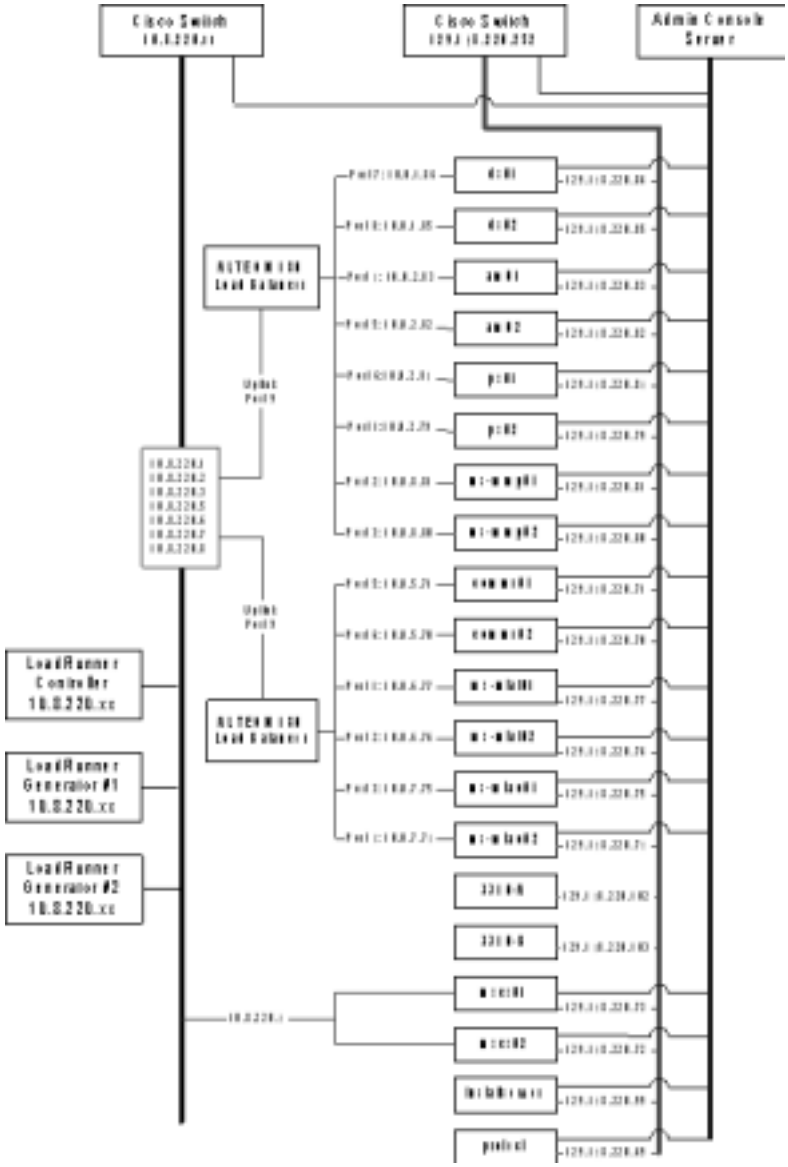

FIGURE 2–2 Example Network Connectivity Specification

The network connectivity specification for the example deployment architecture adds the following information that is not found in the deployment architecture diagram:

- IP addresses for every computer and hardware load balancer used in the solution
- Load balancer port numbers that are used to connect the computers to the load balancers

- The IP addresses for the load balancers show the logical addresses that are used to access the services provided by load-balanced computers

You must develop similar information about the connectivity required for your solution.

Tip – When the network connectivity specification is complete, the network can be connected and made ready for the installation and configuration of your Java ES components.

Developing Your User Management Specifications

The process of installing and configuring your Java ES solution configures your LDAP directory. Installing and configuring Java ES creates both an LDAP schema and an LDAP directory tree. The details of the schema and the directory tree are determined by values that you input during the installation and configuration process. Therefore, installation planning includes developing specifications for a schema and a directory tree structure that will support your Java ES solution.

Your directory tree structure and your schema must support the services your solution provides. This section provides basic descriptions of the options that are available, and the services that each option supports. The main purpose of this section, however, is describing how you select input values for the Java ES installer and the Java ES configuration tools in order to create a schema and a directory tree structure that will support your Java ES solution.

For more information on choosing a schema and designing a directory tree, see additional documentation, such as *Sun Java System Directory Server Enterprise Edition 6.0 Deployment Planning Guide*.

Specifying the LDAP Schema for a Solution

Java ES solutions that use Directory Server can use either of two versions of a standard LDAP schema, which are known as Schema 1 and Schema 2. Your user management specification must specify whether your solution uses Schema 1 or Schema 2.

Schema 2 supports the use of Access Manager, and Access Manager's single sign-on feature. If a solution uses Access Manager, it must use Schema 2.

The installation process configures the directory for the specified schema as follows:

- To establish a Schema 1 directory, simply install Directory Server. Schema 1 is the default schema version.
- To establish a Schema 2 directory, install Directory Server and Access Manager. Installing Access Manager modifies the directory and converts it to a Schema 2 directory.

Tip – If Directory Server and Access Manager are installed on one computer in one installer session, the directory is configured for Schema 2.

If your solution is distributed, you install Directory Server first, on one computer. You then install Access Manager on a second computer. When you install Access Manager you specify the existing directory on the remote computer, and the directory's schema is configured for Schema 2.

Depending on your solution, the following procedures for extending the schema might be necessary:

- If your solution uses components from the Communications Suite (Messaging Server and or Calendar Server), your installation process must apply some additional schema extensions with the Directory Preparation Tool. These extensions are applied before Messaging Server or Calendar Server are installed. They can be applied to either Schema 1 or Schema 2 directories. For an example of an installation plan that includes instructions for the Directory Preparation Tool, see *Sun Java Enterprise System 2005Q4 Deployment Example: Telecommunications Provider Scenario*
- If your solution uses Schema 2, the installation process must apply some additional schema extensions with Delegated Administrator to support Access Manager authentication and authorization for the messaging and calendar services. For an example of an installation plan that applies these schema extensions, see *Sun Java Enterprise System 2005Q4 Deployment Example: Telecommunications Provider Scenario*.

Your LDAP schema specification identifies the schema used in your solution and any schema extensions required by your solution.

Specifying the Directory Tree Structure for a Solution

The LDAP directory for a Java ES solution can be simple or complex, depending on the solution's needs for organizing user data. LDAP directories are, by their nature, flexible in structure. Java ES does not impose structure on the directory, but the installation and configuration process does implement a directory tree structure. You must design your directory tree before you begin the installation and configuration process.

The installation and configuration process establishes the directory tree structure as follows:

1. Running the installer to install Directory Server requires an input value for the directory's base suffix (also referred to as root suffix or root DN). The Java ES installer uses the input value to establish the directory's base suffix. You must specify the base suffix name for your directory tree.

Tip – Solutions with simple directory trees, that do not use Messaging Server or Calendar Server, can store user and group data directly under the base suffix.

2. Running the Messaging Server (a Communications Suite component) configuration wizard to create a Messaging Server instance requires an input value for an LDAP organization DN. The configuration wizard branches the directory tree and creates an LDAP organization using the DN input in the wizard. This organization represents the email domain managed by the Messaging Server instance. The wizard also configures the Messaging Server instance to use the email domain organization for user and group data. The installation plan includes the DN for the email domain organization. For an example of a directory tree structure created by this process, see Figure 2–3. In the example, the base suffix created by the installer is `o=examplecorp`. The email domain organization created by the Messaging Server configuration wizard is `o=examplecorp.com,o=examplecorp`.
3. The configuration wizards for Calendar Server, Communications Express, Instant Messaging, and Delegated Administrator (Communications Suite components) require an input value for an LDAP DN. (The names that appear in the wizards vary.) If a solution uses single sign-on, the same value is input in all of the configuration wizards. The input value is the email domain organization created by the Messaging Server wizard. The result of this configuration is that all of the components store and look up user data in the same LDAP organization. All of the information about a user can be stored in a single directory entry, and the Access Manager single sign-on feature can be used.

An example of a directory tree structure created by this process is illustrated in Figure 2–3. In this example, the Java ES installer established the base suffix `o=examplecorp` and the Messaging Server configuration wizard added the organization `o=examplecorp.com,o=examplecorp`. This organization represents the email domain named `examplecorp.com`. The user data for the email domain is stored in `ou=people,o=examplecorp.com,o=examplecorp`. The other Java ES components in the solution are also configured to look up user data in ou=people,o=examplecorp.com,o=examplecorp.

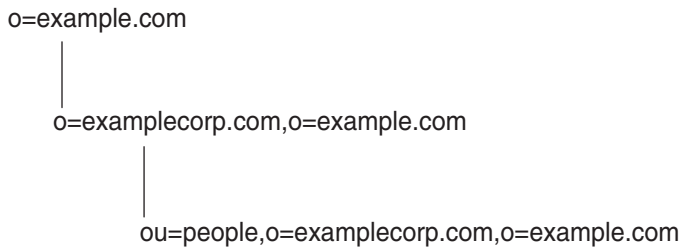

FIGURE 2–3 Example LDAP Directory Tree

If the solution requires the directory tree shown in Figure 2–3, the names for the base suffix and the organization representing the email domain are added to the user management specification.

The example directory tree includes only one mail domain. Many solutions require more complex trees to organize user data. The same basic installation and configuration procedure can establish more complex directory structures. For example, a directory can be configured to support multiple email domains if the solution requires it.

To establish multiple email domains, configure multiple instances of Messaging Server. Each instance manages one email domain. For an example, see *Sun Java Enterprise System 2005Q4 Deployment Example: Telecommunications Provider Scenario*.

It is possible to use other LDAP directories in a Java ES solution, if the solution uses Access Manager to interact with the directory. The directory server must be an LDAP version 3 (LDAP v3) compliant directory server.

◆ ◆ ◆ CHAPTER 3

Preparing Your Installation Plan

After you have developed your implementation specifications, as described in Chapter 2, you have the information you need to prepare your installation plan. An installation plan lists all of the steps needed to install and configure a Java ES solution. Your installation plan will list all of the steps needed to implement your specific Java ES solution.

This chapter explains how to prepare your installation plan. You begin with the information in the deployment architecture and the implementation specifications, which describe the deployed state of your Java ES solution. You analyze the information in these documents, and you determine how to use the Java ES installer and the configuration wizards to implement the solution described in the specification documents.

This chapter describes how to develop an installation plan in the following sections:

Installation Planning Issues

The goal of the installation and configuration process is the distributed system described in the deployment architecture. The distributed system is composed of component instances that run on multiple computers and interoperate with each other. To achieve a functioning distributed system, you must install the component instances on multiple computers and perform the basic configuration that establishes interoperation among the component instances.

The procedures for installation and configuration are determined by the behavior of the Java ES installer and the requirements of the individual components. To ensure that you achieve a functioning distributed system, you must develop an installation plan that uses the installer appropriately and considers the requirements of the components used in the solution. Your plan must describe the correct order for installing each component instance and performing basic configuration. The plan must also specify the configuration values that configure the component instances to interoperate.

This section describes the major issues you must consider when developing an installation plan.

Distributed Installations

The quality-of-service requirements for production Java ES solutions lead to architectures that place component instances on more than one computer. For example, to achieve a reliable portal service the architecture might require two instances of Portal Server on two different computers and use load balancing to establish a failover relationship between the two instances.

The Java ES installer, however, operates on only one computer at a time. Therefore, when you install a distributed solution, you must run the installer on every computer used in the solution.

In many cases, you must install a component or components on a computer and then run configuration wizards to perform the basic configuration. You typically complete installation and configuration on one computer before you proceed to install and configure another set of components on another computer. To install and configure distributed component instances, you might perform a sequence of tasks similar to the one illustrated in Figure 3–1.

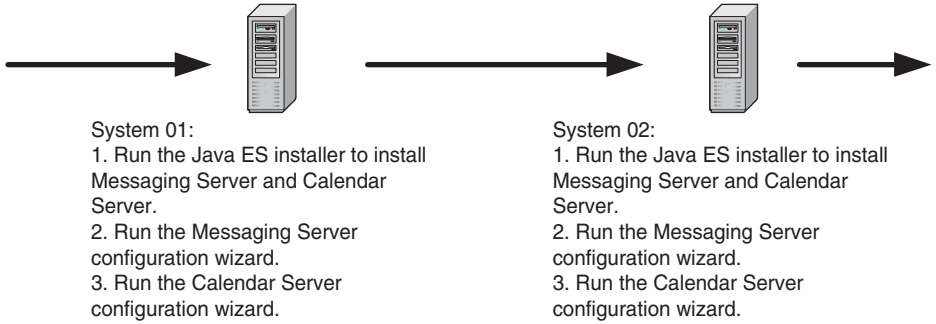

FIGURE 3–1 Distributed Installation Procedure Example

Component Dependencies

Some Java ES components cannot be installed and configured unless other components are installed and configured first. Dependencies occur for several reasons:

- Some components cannot function unless certain other components are installed and configured. For example, for Access Manager to operate properly, it must have access to information about users and services that is provided by an LDAP directory. The installation and configuration procedure for Access Manager requires you to input URLs that enable Access Manager to interoperate with an already functioning directory service. Because of this dependency, you must install and configure Directory Server before you install and configure Access Manager.

- Some components modify the configuration of an existing component. For example, installing and configuring Access Manager modifies the LDAP directory schema. If your solution uses Access Manager, your installation plan must specify that an LDAP directory is installed and configured before Access Manager is installed.
- A number of Java ES components are web applications. These components must be deployed into web containers to function. You must plan to install a web container and start it up before you install and configure your web application components. You can use Web Server, Application Server, or some third-party web containers, but you must plan to have a web container on the computer when you install the web application component.

  Tip – If the solution uses Web Server or Application Server, the Java ES installer can install the web container and the web application component at the same time and automatically deploy the web application component to the web container.

- Your architecture may call for components to be installed in a high-availability cluster provided by Sun Cluster software. The Sun Cluster software must be installed and running before the other components are installed and configured. Additionally, the Sun Cluster Agents for the other components must be installed and configured.

Notice that some of these dependencies are solution-wide and some are local. You consider solution-wide dependencies and local dependencies differently when you develop your installation plan. The difference is described in the following example:

The dependency of Access Manager on Directory Server is a solution-wide dependency. When you install Access Manager, you supply a URL for a directory service provided by one or more instances of Directory Server. Once Directory Server is installed and configured, it provides a directory service that is available to all of the components in the solution. This type of dependency determines the solution-wide sequence for installing and configuring component instances-you must install and configure Directory Server before Access Manager. In your installation plan, solution-wide dependencies determine the overall sequence of installation and configuration steps. You can plan to install Directory Server first and then add components such as Access Manager that depend on the directory service.

The dependency of Access Manager on a web container is a local dependency. To satisfy this dependency, a web container must be installed on the computer that runs Access Manager. This web container, however, does not provide web container services for the entire solution. If your distributed architecture specifies that you install Portal Server on a different computer than Access Manager, you must plan to install a web container on both computers. Each web container supports a different component locally. Therefore, in a distributed solution there is no single location for a web container to provide services for the entire solution, and you must plan to install web containers several times during your overall installation sequence.

To develop an installation plan for your solution, you analyze the deployment architecture that describes the solution and identify dependencies among the components. Your plan must

install and configure components in a sequence that satisfies all of the dependencies. In general, you develop the overall installation sequence from the solution-wide dependencies. Then you consider the local dependencies that might exist on each computer.

The component dependencies are listed in Table 3–1. For more information about working with these dependencies, see the descriptions of the individual components in "Developing Your Installation Plan" on page 38.

TABLE 3–1 Java ES Component Dependencies

Product Component	Dependencies	Nature of Dependency	Must be Local?
Access Manager	Directory Server	To store configuration data; to store and enable lookup of user data	No
	J2EE web container, one of: -Application Server -Web Server -BEA WebLogic Server -IBM WebSphere Application Server	Access Manager must be deployed to one of these web containers	Yes
Access Manager SDK	Access Manager	To provide the underlyingAccess Manager services	No
	J2EE web container, one of: -Application Server -Web Server -BEA WebLogic Server -IBM WebSphere Application Server	Access Manager SDK must be deployed to one of these web containers	Yes
Access Manager Distributed Authentication	Access Manager	To provide the underlyingAccess Manager services	No
	J2EE web container, one of: -Application Server -Web Server -BEA WebLogic Server -IBM WebSphere Application Server	Access Manager SDK must be deployed to one of these web containers	Yes

TABLE 3–1 Java ES Component Dependencies *(Continued)*

Product Component	Dependencies	Nature of Dependency	Must be Local?
Access Manager Session Failover	Access Manager	To provide the underlyingAccess Manager services	No
	Message Queue	To provide reliable asynchronous messaging	No
Application Server	Message Queue	To provide reliable asynchronous messaging	Yes
	Web Server (optional)	To provide load balancing between Application Server instances	Yes
	High Availability Session Store (optional)	To store session state, which supports failover between Application Server instances	Yes
Directory Proxy Server	Directory Server	To provide underlying LDAP directory services	No
Directory Server	None		
High Availability Session Store	None		
Java DB	None		
Message Queue	Directory Server (Optional)	To store administered objects and persistent messages	No
	J2EE web container, one of (Optional): -Application Server -Web Server	To support HTTP transport between clients and Message Broker	No
	Sun Cluster (Optional)	To support use of Message Queue in high availability solutions	No
Portal Server	J2EE web container, one of: -Application Server -Web Server -BEA WebLogic Server -IBM WebSphere Application Server	Portal Server must be deployed to one of these web containers	Yes

TABLE 3–1 Java ES Component Dependencies *(Continued)*

Product Component	Dependencies	Nature of Dependency	Must be Local?
	Directory Server	To store user data used for authentication and authorization	No
	Access Manager or Access Manager SDK	To provide Access Manager services; a local Access Manager SDK provides access to a remote Access Manager	Yes
	Service Registry Client	To provide libraries needed for compilation	No
Portal Server Secure Remote Access	Portal Server	To provide the underlying portal service.	No
	Either Access Manager or Access Manager SDK	To provide Access Manager services; a local Access Manager SDK provides access to a remote Access Manager	Yes
Rewriter Proxy	Portal Server	To provide the underlying portal service.	No
Netlet Proxy	Portal Server	To provide the underlying portal service.	No
Service Registry	Application Server	To provide the necessary container service.	Yes
	Service Registry Client	To provide the necessary client interface	Yes
Service Registry Client	None		
Sun Cluster Software	None		
Sun Cluster Agents	Sun Cluster	To provide underlying clustered services.	Yes
Sun Cluster Geographic Edition	Sun Cluster	To provide underlying clustered services.	Yes
Web Proxy Server	Web Server	To provide remote access to web applications running on Web Server	Yes
	Directory Server (Optional)	To store user data used for authentication and authorization	No

TABLE 3–1 Java ES Component Dependencies *(Continued)*

Product Component	Dependencies	Nature of Dependency	Must be Local?
Web Server	Directory Server (Optional)	To store user data used for authentication and authorization	No

Configuring for Interoperation

The goal of the installation and configuration process is a system of interoperating component instances. Since you install components and perform basic configuration on one computer at a time, you must determine in advance the configuration values that will result in successful interoperation with components on other computers.

The configuration values that result in interoperation include such values as the URLs or port numbers that one component instance uses to communicate with another component instance. For example, if your solution uses Access Manager, you must first install and configure an LDAP repository, such as a Directory Server instance. Then, when you install and configure an Access Manager instance, you must provide values that configure the Access Manager to interoperate with the LDAP directory you have already installed and configured.

The Java ES installer does not know what components are installed on the other computers used in the solution. For example, when you install Access Manager, the installer does not know where the appropriate LDAP directory is located. To ensure the success of your installation and configuration process, you must determine in advance which installation and configuration values will lead to successful interoperation between your Access Manager instance and your Directory Server instance. You include these values in your installation plan. Then, when you are installing and configuring components, you type the values in your plan, and you successfully configure your components to interoperate with each other.

You might perform a sequence of installation and configuration tasks similar to the one illustrated in Figure 3–2.

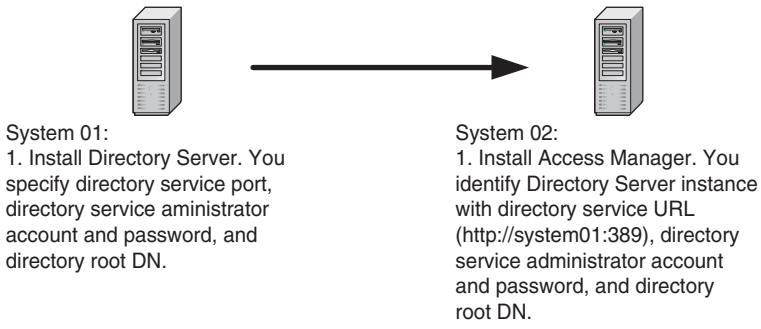

FIGURE 3–2 Configuring Components to Interoperate

Whatever the architecture of your solution, you must develop an installation plan that includes all the configuration values needed to configure the components and achieve an interoperating, distributed solution.

Redundancy Strategies

Most solutions intended for production use include some type of redundancy. Redundancy strategies use multiple instances of a component to provide a single service. Redundancy is used to satisfy quality of service requirements. For example, redundancy is used to increase throughput in order to satisfy performance requirements, or to avoid a single point of failure to in order satisfy reliability requirements.

Three strategies are available for using redundant instances of Java ES components: load balancing, clustering with Sun Cluster software, and Directory Server replication. The recommended installation and configuration procedure for each of these strategies is outlined briefly in the following paragraphs:

- Load balancing can be implemented either in hardware or software. Load balancing is best set up by installing and configuring one instance of the load-balanced component, and then testing that the service provided by the first instance is available through the load balancer. After verifying that the service is available, you install and configure the additional instances of the component required by your deployment architecture. This phased approach to installing and configuring facilitates troubleshooting configuration problems.
- Clustering is implemented in several steps. The first step is to install the Sun Cluster software and establish and configure the cluster. The next step is to install the components that run in the cluster. For example, the first step towards implementing the cluster shown in Figure 2–1 is installing Sun Cluster software on computers STR1 and STR2, and establishing and configuring the cluster. The second step is installing and configuring Messaging Server and Calendar Server. The third, and final, step is installing and

configuring the Sun Cluster data services for Messaging Server and Calendar Server. When the Sun Cluster data services are configured, the cluster nodes recognize the Messaging Server and Calendar Server instances.

- Directory Server replication is also implemented in several steps. For example, when you implement multimaster replication the first step is installing, configuring, and verifying all of the Directory Server instances. The second step is shutting down all but one of the Directory Server instances. The third step is installing and configuring the other components in the solution. Any changes to the schema or directory structure are made to the single running Directory Server instance. The final step, after all component instances in the solution are installed, configured, and verified, is restarting the other instances of Directory Server and using the replication feature to configure synchronization and failover. This copies the modified and updated directory data to all of the Directory Server instances.

When your deployment architecture uses any of these redundancy strategies, your installation plan must include procedures for installing multiple instances of a component and configuring the instances to operate as a single service.

LDAP Schema and LDAP Directory Tree Structure

Most Java ES solutions include Directory Server. When you install and configure a solution with Directory ServerDirectory Server you input values that establish both the directory schema and the directory tree structure. Your installation plan must list the input values that result in the correct LDAP schema and directory tree structure.

You specify you LDAP schema and your directory tree structure before you begin your installation plan. Your installation plan includes the values you type in when running the installer to create the specified schema and directory tree structure. For examples of schema and directory tree specifications, see "Developing Your User Management Specifications" on page 20.

The LDAP schema is established by the following installation and configuration processes:

1. Installing Directory Server automatically establishes a directory with Schema 1. No input is required to select the schema.
2. Installing Access Manager automatically modifies the directory, and converts it to Schema 2. No input is required to select the schema.
3. In solutions that include Communications Suite components, running the Directory Preparation Tool extends the schema for use with Messaging Server, Calendar Server, and Communications Express. The Directory Preparation Tool extends both Schema 1 and Schema 2 directories. Input values for the Directory Preparation Tool are listed in your installation plan.

4. In solutions that include Communications Suite components, running Delegated Administrator extends the schema with object classes and attributes used to authorize and authenticate users for specific services. The input values depend on the service provided by your solution. You list the input values in your installation plan.

The installation and configuration process also establishes the basic directory tree structure:

1. Installing Directory Server creates the base suffix, or directory tree root. The base suffix is a required input value when the Java ES installer installs Directory Server. In your installation plan, you list the base suffix as one of the input values for the installation process.
2. Installing and configuring Messaging Server branches the directory tree and creates an LDAP organization. This organization represents the email domain managed by the Messaging Server instance. The name of the organization is a required input for the Messaging Server configuration wizard. In your installation plan, you list the organization DN as one of the input values for the Messaging Server configuration process.
3. Installing and configuring Calendar Server, Communications Express, Delegated Administrator, and Instant Messaging specifies where in the directory these components look up user data. An LDAP DN is required input for each component's configuration wizard, and your installation plan lists the DN as an input value for each configuration wizard. If the solution uses Access Manager single sign-on, all of these components must be configured to use the same location for user data, which is the organization that the Messaging Server configuration wizard created. The same LDAP DN is input in all of these configuration wizards. In your installation plan, you list the organization DN as one of the input values for all of the configuration wizards.

You take the names for the LDAP base suffix and email domain organization from your user management specification and add them to your installation plan. For more information about the user management specification, see "Developing Your User Management Specifications" on page 20.

Java ES Installer Behavior

This section describes some behaviors of the Java ES installer that affect installation planning.

The Installer is Local

The Java ES installer installs component software on one computer at a time. Most solutions are distributed, and you must run the installer more than once. Your installation plan must include procedures for each time you run the installer. This section describe how to analyze a deployment architecture and determine how many times you must run the installer to implement the architecture.

A few solutions are installed on one computer only, and the installation plans for these solutions provide procedures for running the installer only once. The solutions that require running the installer only once are the following:

- A number of components are installed on one computer to evaluate Java ES features.
- One component instance is added to an established solution. This includes adding component instances that have dependencies on existing components.

Most solutions are distributed across several computers. Installation plans for these solutions must describe running the installer multiple times to install and configure the complete solution. To analyze these solutions, use the following guidelines:

- In most cases, when you combine several components on one computer you run the installer only once. This is particularly true if the installer runs in Configure Now mode, because in Configure Now mode, the installer can install both a web container and the component that runs in the web container. In these cases, your installation plan describes running the installer once on the computer and selecting all of the components specified for the computer.

  Tip – Some components cannot be configured by the installer, even in Configure Now mode. When these components are installed on a computer, the configuration process is completed by running a configuration wizard for each component. When these components are installed in combination with components that are configured by the installer, you run the installer first. After you run installer, you complete the process by running the configuration wizards for the components not configured by the installer. In these cases, your installation plan must describe running the installer and the correct sequence for running the configuration wizards.

- Some combinations of components can only be installed by running the installer more than once on a computer. These combinations include the following:
  - Some component combinations that include a web container. If Web Server or Application Server is installed in Configure Later mode, you must configure an instance of Web Server or Application Server before you can install any other component that will run in the web container. If your solution uses a third-party web container, you must install, start, and verify the web container, before you install the web-based Java ES components. Your installation plan must include procedures for running the installer multiple times on each computer.
  - Component combinations that use Sun Cluster software. If the components installed into the cluster are installed on a cluster file system, the Sun Cluster software must be installed and the cluster file system created before other components can be installed in the cluster nodes. Your installation plan must include procedures for running the installer multiple times on each computer.

The purpose of this section is to introduce the idea that installation plans must sometimes describe running the installer and the configuration wizards on one computer, or running the

installer multiple times on one computer. For more information on the actual installation procedures for different component combinations, see "Developing Your Installation Plan" on page 38.

Installer Operating Modes

The installer runs in two different modes, known as Configure Now and Configure Later. The modes differ in the following ways:

- In Configure Now mode, the installer configures runnable instances of some, but not all, components. The components configured in Configure Now mode can be started and verified as soon as the installer completes. Runnable instances of the remaining components are created after the installer runs, by running component configuration wizards. For components configured by the installer, your installation plan must include the configuration values you will input when you run the installer. For components that are configured after the installer runs, your installation plan must include procedures for running the configuration wizards and the configuration values you will input when you run the configuration wizards.

  Tip – A significant feature of Configure Now mode is its ability to install a web container and components that run in the web container at the same time. The installer automatically deploys the components to the web container.

- In Configure Later mode, the installer copies component software files to the computer but does not create runnable instances. You create the instances after you run the installer, by running the component configuration wizards. Your installation plan must include procedures for running the configuration wizards and the configuration values you will input when you run the configuration wizards.

The configuration option you select applies to an entire installation session. If you want to install some components on the computer in Configure Now mode and some in Configure Later mode, you must run the installer more than once.

Installer Compatibility Checking

The Java ES installer performs some dependency and compatibility checking. However, the installer can only check the local computer. For example, if you are installing Access Manager in a distributed solution, the installer cannot check whether the remote Directory Server is compatible with the Access Manager you are installing.

Compatibility is unlikely to be an issue if you are installing and configuring an all-new solution, with all components from the same Java ES release. It might become an issue if you are adding a new component to an established solution, or building a Java ES solution around existing components. For example, if you are already using Directory Server, and you are building a

solution using Access Manager and Portal Server around the existing Directory Server, compatibility among the components becomes an issue. You need to confirm that the components are compatible before you begin to install and configure new components.

- **Component Dependency Checking.** The Java ES installer will prevent you from omitting components that are required by other components you have selected for installation, but only on the local host. In a distributed solution, the installer does not check the remote host to verify that the remote component is there. In this situation, you are responsible for verifying that the remote component is compatible and in the proper running state.
- **Upgrading.** The Java ES installer will check installed Application Server, Message Queue, HADB, and Java DB for compatibility with the components you are installing and ask if you want to upgrade the components during installation.

  The Java ES installer does perform upgrade of shared components. For more information of this topic, see "Surveying Existing Hosts" in *Sun Java Enterprise System 5 Installation Guide for UNIX*.

Other Installation Issues

This section lists a number of specific issues that occur in some solutions with references to detailed information.

TABLE 3–2 Installation Issues to Consider

Solution Requires	Guidelines or Instructions
Using Solaris 10 zones	If you will be installing into Solaris 10 zones, refer to Appendix A.
Using Directory Server encryption	Configure LDAPS (SSL over LDAP) on the Directory Server instance.
Using a third-party web container with Access Manager	Third-party web containers (BEA WebLogic Server or IBM WebSphere Application Server) can be used with Portal Server and Access Manager. These containers must be installed and running before installing any Java ES components that depend on them. To use a third-party web container for Access Manager SDK, you must configure Access Manager SDK manually after installation. See "Access Manager SDK With Container Configuration Example" in *Sun Java Enterprise System 5 Installation Guide for UNIX* Note: Portal Server can only use third-party web containers on Solaris OS. Note: Access Manager and Portal Server should use the same type of web container.
Using Apache Web Server for load balancing plug-in	The Apache Web Server can be used with the Application Server load balancing plug-in. In this case, the Apache Web Server must be installed and running before installing any Java ES components that depend on it.

TABLE 3–2 Installation Issues to Consider *(Continued)*

Solution Requires	Guidelines or Instructions
Using Schema 1 LDAP	For a Schema 1 deployment, you cannot use Access Manager.
Configuring single user entry and single sign-on	Access Manager is required for single sign-on.
Configuring High availability using HADB	A summary of the procedures for setting up HADB for high availability is contained in "Web and Application Services Example" in *Sun Java Enterprise System 5 Installation Guide for UNIX*.
Application Server load balancing	An summary of the procedures for using the Application Server load balancing plug-in is contained in"Web and Application Services Example" in *Sun Java Enterprise System 5 Installation Guide for UNIX*.
Non-root ownership	If non-root ownership will be required for Application Server or Web Server, refer to "Non-Root Examples" in *Sun Java Enterprise System 5 Installation Guide for UNIX*

Developing Your Installation Plan

Your deployment architecture and implementation specifications describe the final state of your solution. The deployment architecture shows you how many component instances are installed, which computer systems the component instances are installed on, and how the component instances interoperate. To reach the state described in the deployment architecture, you must install and configure the component instances in your solution, one computer system at a time, until you have installed and configured the complete solution. Your installation plan must provide installation and configuration procedures for every component instance in your solution, in the correct order.

To develop an installation and configuration plan, you must apply your knowledge of component dependencies and other installation issues to your Java ES deployment architecture and implementation specifications. You must determine the correct sequence for installing and configuring the component instances in your solution and the installation and the correct configuration input values, which will achieve interoperation of the component instances.

This section is a guide to analyzing a deployment architecture and set of specifications and developing an installation plan. In general, you begin as follows:

1. Open a text file, a blank sheet of paper, or some other medium for recording your plan.
2. In your deployment architecture, examine the components on each computer system and determine what component dependencies exist.

3. Identify the component instances that have no dependencies on other components. These are typically instances of Directory Server. You begin your installation plan with instructions for installing these component instances on the specified computer systems. Begin your installation plan by recording these computer systems, and the component instances installed on them.
4. Determine the correct installation/configuration values in your solution for these component instances on these specific computer systems. Add these configuration values to your installation plan.
5. Among the remaining components, determine which components have dependencies only on Directory Server. These are typically the computer systems with Access Manager. List these computer systems next in your installation plan.
6. Continue analyzing your specifications in order of component dependencies. Determine the necessary configuration values, and record these component instances in your plan.

For example, if you use this process to analyze the deployment architecture illustrated in Figure 2–1, you develop an installation plan that looks like Table 3–3.

Table 3–3 shows the first eight steps of the installation plan. In order to make the structure of this plan clear, the individual configuration values are not listed. In this plan, note the following:

- The plan lists the computers in the solution according to the order in which the component instances will be installed and configured.
- The sequence of installation is determined by applying both solution-level dependencies and the local dependencies. Applying the solution-level dependencies gives a basic sequence of Directory Server, Access Manager, Messaging Server, and then Calendar Server. Applying the local Communications Express dependencies to this sequence adds Web Server instances on computers AM1 and AM2, and also Sun Cluster software and the Sun Cluster agents on computers mscs01 and mscs02.
- The plan includes outlines of the installation and configuration procedures for all of the redundancy strategies employed in Java ES solutions. The list of tasks for DS1 and DS2 is an example of a plan for Directory Server multimaster replication. The list of tasks for AM1 and AM2 is an example of a plan for load balanced components. The list of tasks for STR1 and STR2 is an example of a plan for components that run in a Sun Cluster configuration.
- The tasks for STR1 and STR2 provide an example of installing and configuring multiple components on one computer. The first time you run the installer, you install the Sun Cluster core component. After you configure the Sun Cluster core component, you run the installer again to install Messaging Server and Calendar Server. These components are configured in order, according to their dependencies. The third time you run the installer on the computer, it installs the Sun Cluster agents for Messaging Server and Calendar Server, which depend on the presence of Messaging Server and Calendar Server.

TABLE 3–3 Summary Installation Plan for the Sample Deployment Architecture

Computer	Installation and Configuration Tasks
DS1	1. Run the Java ES installer on this computer. Install and configure a Directory Server instance, using the configuration values specified in the user management specification. 2. Start and verify the Directory Server instance.
DS2	1. Run the Java ES installer on this computer. Install and configure a Directory Server instance with the configuration values specified in the user management specification. 2. Start and verify the Directory Server instance. 3. Verify that the load balancer is working properly for both Directory Server instances. 4. Shut down the Directory Server instance in DS2. Leave the Directory Server instance on DS1 running.
AM1	1. Run the Java ES installer on this computer. Install and configure an Access Manager instance. Configure the Access Manager instance to interoperate with the logical directory service created by the load balanced Directory Server instances. 2. Start and verify the Access Manager instance. 3. Configure the Access Manager instance for load balancing.
AM2	1. Run the Java ES installer on this computer. Install and configure an Access Manager instance. Configure the Access Manager instance to interoperate with the logical directory service created by the load balanced Directory Server instances. 2. Start and verify the Access Manager instance. 3. Configure the Access Manager instance for load balancing. 4. Use the Access Manager console to modify directory entries for Access Manager. 5. Verify that the two Access Manager instances are working correctly with load-balanced operation.
STR1	1. Run the Java ES installer. Install the Sun Cluster core component. 2. Prepare the computer for Sun Cluster configuration. This step includes creating and mounting file systems used by Sun Cluster software. 3. Run the Sun Cluster configuration wizard. Establish and configure the cluster.

TABLE 3–3 Summary Installation Plan for the Sample Deployment Architecture *(Continued)*

Computer	Installation and Configuration Tasks
STR2	1. Run the Java ES installer. Install the Sun Cluster core component. 2. Prepare the computer for Sun Cluster configuration. This step includes creating and mounting file systems used by Sun Cluster software. 3. Run the Sun Cluster configuration wizard. Establish and configure the cluster. 4. Complete the configuration of the Network Timing Protocol (NTP) on STR1 and STR2. 5. Add the quorum device to the cluster (connected to both computers). 6. Create cluster file systems, and resource groups, set up virtual host name and IP address. 7. Verify the cluster's failover capabilities.
STR1	1. Run the Java ES installer. Install Messaging Server and Calendar Server. 2. On computer DS1, run the Directory Server Preparation Tool. 3. Run the Messaging Server configuration wizard to create a Messaging Server instance. Supply configuration values that create a branch in the LDAP directory tree according to the user management specification. Supply configuration values that configure the Messaging Server instance to interoperate with the load-balanced Access Manager instances and load-balanced Directory Server instances. 4. Configure Messaging Server for single sign-on. 5. Start and verify the Messaging Server instance. 6. Run the Calendar Server configuration wizard to create a Calendar Server instance. Supply configuration values that configure the instance to use the LDAP branch created by Messaging Serverconfiguration for user and group data. Supply configuration values that configure the Calendar Server instance to interoperate with the load-balanced Access Manager instances and load-balanced Directory Server instances. 7. On computer STR2 create a Calendar Server user, user group, and directory. 8. Edit the Calendar Server configuration file. Set configuration parameters to use the virtual IP address instead of the computer's IP address. 9. Configure Calendar Server for single sign-on. 10. Start and verify the Calendar Server instance.
STR1	1. Run the Java ES installer. Install Sun Cluster Agent for Messaging Server and Sun Cluster Agent for Calendar Server. 2. Using the Messaging Server agent, create and enable a Messaging Server resource. 3. Verify failover of the Messaging Server resource from STR1 to STR2. 4. Using the Calendar Server agent, create and enable a Calendar Server resource. 5. Verify failover of the Calendar Server resource from STR1 to STR2.

TABLE 3–3 Summary Installation Plan for the Sample Deployment Architecture *(Continued)*

Computer	Installation and Configuration Tasks
STR2	The instances you configured on mscs01 are automatically recognized as shared resources.

◆ ◆ ◆ APPENDIX A

Java ES and Solaris 10 Zones

This appendix describes the issues that arise when you install and configure Java ES components in Solaris 10 zones, and recommends some practices regarding for addressing those issues. This appendix contains the following sections:

What Are Zones?

Zones are an application and resource management feature of the Solaris 10 operating system. This feature allows the operating system to be represented to applications as virtual operating system environments (zones) that are isolated and secure. These zones provide the advantages of operating system independence with some level of centralized resource management. Hence applications can be isolated from one another by being installed and run in different zones, while at the same time certain operating system resources can be centrally allocated and administered.

From the point of view of an operating system supporting multiple zones, operating system resources include resources such as process management, memory, network configuration, file systems, package registries, user accounts, shared libraries, and, in some cases, installed applications.

Structure of a Multi-zone Environment

A multi-zone environment consists of a global zone (the default operating system) and one or more non-global zones. The global zone contains resources that can be allocated among non-global zones by a global (zone) administrator. Non-global zones provide the following features:

- Security. By running distributed services in non-global zones, you limit the damage possible in the event of a security violation. An intruder who successfully exploits a security flaw in software within one zone is confined to that zone. The privileges available within a non-global zone are a subset of those available in the global zone.
- Runtime isolation. Non-global zones allow for the deployment of multiple applications on the same computer even if those applications require different levels of security, require exclusive access to global resources, or require individualized configuration. For example, multiple applications running in different zones can bind to the same network port by using the distinct IP addresses associated with each non-global zone. The applications are prevented from monitoring or intercepting each others network traffic, file system data, or process activity.
- Administrative isolation. The virtualized operating system environment allows for separate administration of each non-global zone. Actions taken by a zone administrator (as opposed to the global administrator) in a non-global zone, such as creating user accounts, installing and configuring software, and managing processes, do not affect other zones.

There are two types of non-global zones: whole root zones and sparse root zones:

- Whole root zones. Contain a read/write copy of the file system existing on the global zone. When a whole root zone is created, all packages that are installed on the global zone are made available to the whole root zone: a package database is created and all files are copied onto the whole root zone for the dedicated and independent use of the zone.
- Sparse root zones. Contain a read/write copy of only a portion of the file system existing on the global zone (hence the name sparse root) while other file systems are mounted read-only from the global zone as loop-back virtual file systems. When a sparse root zone is created, the global administrator selects which file systems to share with the sparse root zone (by default, the `/usr`, `/lib`, `/sbin`, and `/platform` directories are shared as read-only file systems). All packages that are installed on the global zone are made available to the sparse root zone: a package database is created and all files in the mounted file system are shared with the zone.

Whole Root Zones vs. Sparse Root Zones

The choice between using whole root non-global zones versus sparse root non-global zones depends upon a trade-off between resource efficiencies and administrative control. Whole root zones allow you to maximize administrative control (independence and isolation) at the cost of memory and other resources, while sparse root zones optimize the efficient sharing of

executables and shared libraries (while using a much smaller disk footprint) at the cost of administrative independence. There is currently no measure of the performance advantage of sparse root zones over whole root zones; it is very likely to be software-specific.

Package Propagation

Packages installed in a global zone are (by default) available to all non-global zones: a process called package propagation. (For propagation to take place, newly-created non-global zones must be fully booted, that is, be in a running state.) Propagation provides local (non-global) visibility and availability to packages that are installed in the global zone. Propagation allows for application package life-cycle management (installation, upgrade, un-installation) to be performed centrally by a global administrator, while application configuration and runtime management are performed by (non-global) zones administrators.

For whole root zones, propagation is achieved through the automatic copying of installed files from the global zone to the whole root zones and through automatic synchronization of registry information. For sparse root zones, propagation is achieved through read-only file systems that are shared between the global and sparse root zones and through automatic synchronization of registry information.

Propagation of packages to non-global zones is controlled at the package level using internal package attributes. For some values of these attributes (the default values, at least), propagation can be disabled at install time by using the `pkgadd –G` option, which overrides the attribute values. Once installed, the propagation behavior of a package cannot be modified, except by uninstalling and reinstalling the package. Patches, for example, cannot change the propagation behavior of a package; in fact, patches must be applied in accordance with the propagation behavior of the package they are upgrading.

Why Use Zones for Java ES?

The isolation provided to applications running in different zones is similar to the isolation provided by running applications in the operating systems of different computers. Hence, instead of installing, configuring, and running Java ES components on different computers in order to isolate and secure them, those components can be installed, configured, or run in different zones within the same computer.

This consolidation of Java ES components can also enable more efficient resource utilization. Java ES components running in dedicated, under-utilized computers, can be run instead in different non-global zones of a single computer. Global administrators can dynamically allocate resources among the different zones depending on the resource requirements of the components running in those zones. (Note this possibility requires more knowledge and understanding of the resource requirements of different components than is generally available at the present time.)

A multi-zone environment can help achieve other objectives as well:

- Version Separation. Parallel sets of Java ES components of different versions can be run in different zones. This allows for migration from one Java ES version to another over a period of time. For example, Java ES Release 4 components in one non-global zone can be run in parallel with Java ES Release 5 components in another non-global zone. To achieve this type of version separation, life-cycle management (as well as configuration and runtime management) is delegated to zone administrators.
- Centralized Life-cycle Management. While not fully supported due to Java ES limitations, zones make it possible to centralize life-cycle management of Java ES components: components can be installed, upgraded, and uninstalled in the global zone but configured and run in a number of non-global zones to provide for runtime isolation, security, scalability, and other needs. Centralization of life-cycle management is advantageous when there are a number of instances of a component running in different zones, or when you want to ensure that such instances are synchronized to the same release version.

  For example, you would be able to install Application Server once in the global zone and run multiple instances in different non-global zones. The various Application Server instances could support Access Manager, Portal Server, or other Java ES components (these can be the same or different components in different non-global zones). Or different Application Server instances could be used by different development teams in different zones.

  To achieve this objective, life-cycle management is performed by a global administrator, while configuration and runtime management is delegated to the respective zone administrators. This approach requires broad coordination when life-cycle management tasks (such as upgrade) are performed.
- Organizational Independence. Different organizations can maintain separate deployments of Java ES components, or separate runtime instances of Java ES components, all coexisting and running on the same computer. For example, different groups of developers can use their own distinct instances of Java ES components, or different organizations can use different deployments of Java ES for testing, pre-production staging, or production. Organizational independence can be achieved in various ways, depending on specific objectives: either by centralizing Java ES life-cycle management while delegating configuration and runtime management to zone administrators, or by delegating all management functions (life-cycle, configuration, and runtime) to zone administrators.

The different objectives you can achieve using Java ES in a multi-zone environment, and the usage scenarios they imply, require different strategies for deploying and administering Java ES components across a multi-zone environment. Some objectives make use of the isolation of different zones to independently manage different Java ES components and their runtime instances, while other objectives make use of the propagation capabilities of the global zone to simplify life-cycle management of Java ES components.

Installation and administration strategies for using Java ES in a multi-zone environment will be revisited after discussing some of the multi-zone environment limitations imposed by the nature of Java ES software.

Zones Limitations of Java ES Components

Java ES components are grouped into different types, as described in *Sun Java Enterprise System 5 Technical Overview*. Accordingly, system service components provide the main Java ES infrastructure services, while service quality components enhance those system services. These two types of Java ES components are together referred to here as product components, components that are selectable within the Java ES installer.

Each product component depends on one or more locally shared libraries known as Java ES shared components. Shared components are installed automatically by the Java ES installer during product component installation, depending on the product components that are being installed. They are not individually selected, installed, or configured during deployment of Java ES product components.

Java ES Shared Components and Zones

The discussion in "Why Use Zones for Java ES?" on page 45 focused on the use of zones by Java ES product components: those that can be explicitly selected in the Java ES installer and installed and configured in various zones to achieve a desired deployment architecture and functional capability. However, the shared components upon which product components depend set a number of limitations on how Java ES is deployed in a multi-zone environment. There are two issues regarding Java ES shared components and zones:

Synchronization of Shared Components

The difficulty of testing and supporting the large number (around 30) and complex interactions between Java ES shared components and Java ES product components mandates that all shared components within a single operating system instance be synchronized to the same Java ES version. In other words, all Java ES shared components installed in a non-zone environment, or in any single zone within a Solaris 10 environment, must be of the same version. This requirement sets certain restrictions on how Java ES can be used in a multi-zone environment.

This synchronization requirement has the following implications:

- Different versions of Java ES shared components can only reside in different zones. For example, you can install Java ES Release 4 shared components in one zone and Java ES Release 5 shared components in another zone, but you cannot combine them in the same zone.
- If any shared component in a zone is upgraded or any new shared component of higher version is introduced, then all shared components in that zone must also be upgraded at the same time. (Shared components are required to be backwardly compatible, so there is no problem for Release 4 product components to work with Release 5 shared components.) For example, suppose a Release 5 product component is installed in a zone in which one or more Release 4 product components reside. Because the Release 5 product component requires

some number of Release 5 shared components, the synchronization requirement means that all Release 4 shared components residing in that zone must be upgraded to Release 5 at the same time the Release 5 product component is installed. This is the case even if the Release 5 product component being installed requires different shared components from those that are already installed in the zone.

- When shared components are installed in and propagate from the global zone (see "Java ES Propagation Policies" on page 49), then special care must be taken to maintain synchronization of shared components in all zones. Otherwise it would be possible for shared components of an earlier version in a non-global zone to be mixed with Release 5 shared components that have been propagated from the global zone. (Special care normally means that shared component life-cycle management takes place only in the global zone. For more information see Table A–2, and "Shared Component Special Cases" on page 57.)

The shared component synchronization requirement imposes restrictions on what the Java ES installer is constrained to do in a multi-zone environment (for more information, see "Zone Support in the Java ES Installer" on page 49) and also impacts procedures for installing and upgrading Java ES product components in a multi-zone environment.

Shared Components and Sparse Root Zones

Another issue impacting the use of Java ES in a multi-zone environment is that a large number of shared components cannot be installed in sparse root zones because of the read-only file systems in sparse root zones. Hence, those shared components whose base directory is `/usr` (a directory that by default is shared by the global zone) must be installed in the global zone to be available in a sparse root zone.

The inability to install a number of Java ES shared components in sparse root zones means that to successfully install product components which have dependencies on such shared components into sparse root zones, the shared components must first be installed in the global zone and propagated to non-global zones.

Java ES Product Components and Zones

Some of the objectives discussed in "Why Use Zones for Java ES?" on page 45 for using Java ES in a multi-zone environment, and the usage scenarios they imply, make use of the propagation capabilities of the global zone to simplify life-cycle management of Java ES product components. Such usage scenarios, for example, call for life-cycle management of Java ES product components to be performed in the global zone by the global administrator, while the configuration and runtime management of those components is performed in non-global zones by zone administrators.

In other words, product components would be installed and upgraded in the global zone, but instances would be configured and run in non-global zones. This usage scenario would combine the benefits of centralized life-cycle management with the isolation and security afforded by non-global zones.

This scenario, however, depends upon the ability of each product component to be installed in the global zone, but be configured and run in a non-global zone. This separation depends upon how configuration of each product component is achieved, where configuration and dynamic application data is stored, how configuration data is located by executing binaries, and how upgrades are performed. For example separation might depend upon what pre or post install or upgrade scripts do: whether they start or stop component instances, set up links to configuration data, or perform other tasks that blur the distinction between life-cycle and configuration management.

This separation can also depend upon whether configuration is performed in a whole root or sparse root zone. For example, if a product component's configuration script writes to a read-only file system in a sparse root zone (for example `/usr`) or if non-default file systems (such as `/opt`) are shared with a sparse root zone, then the configuration of a component can fail.

Note – Nearly all Java ES product components are installed under `/opt`, which by default, is writable in sparse root zones. For more information, see *Sun Java Enterprise System 5 Installation Reference for UNIX*

At the current time, the ability of each of the roughly 20 Java ES product components to support the separation of life-cycle management and configuration/runtime management between global and non-global zones has not been established. The various product components have adopted different approaches to configuration and upgrade. Given this situation, propagation of Java ES product components (except for Message Queue) is not currently supported. For more information, see "Java ES Propagation Policies" on page 49.

Zone Support in the Java ES Installer

Based on the usage scenarios discussed in "Why Use Zones for Java ES?" on page 45, and the Java ES component requirements and limitations discussed in "Zones Limitations of Java ES Components" on page 47, the Java ES installer provides qualified zones support for installation (and upgrade) of Java ES product components and for synchronization of shared components. Policies have been implemented in the installer to help prevent problematic installation and upgrade scenarios.

Java ES Propagation Policies

Based on the limitations discussed in Section 3, the Java ES installer implements two Java ES propagation policies:

- When product components are installed in the global zone, they are set by default to not propagate into non-global zones (Message Queue is an exception). Hence the non-global zones do not see them in their registries nor have access to the installed components.

- When shared components are installed in the global zone (for example, as part of the installation of product components), they are set to propagate into non-global zones. Hence non-global zones see them in their registries and have access to the installed shared components. This policy helps enforce the requirement that shared component versions be synchronized within any zone as described in "Java ES Shared Components and Zones" on page 47.

Installation of Product Components

The Java ES installer can install product components as well as the shared components needed to support each product component. Before installing a selected product component, the installer checks for the existence of current and previous versions of shared components. If the installer detects that a shared component required by the selected component is of a previous version or is missing, the installer will upgrade all shared components currently installed and install any missing shared components required by the selected component. This behavior, which meets the requirements of "Synchronization of Shared Components" on page 47, applies to non-zone operating systems, global zones, and all non-global zones.

However, there are two exceptions to this behavior:

- In sparse root zones, some shared components cannot be installed or upgraded (see "Shared Components and Sparse Root Zones" on page 48), and installation is halted until such time as such shared components have been installed or upgraded in the global zone. The installer provides the following message: "The following shared components, required by the components you have selected, cannot be installed or upgraded in a sparse root zone. Please install or upgrade these shared components in the global zone before proceeding. Use the All Shared Components option." For more information see "Synchronize All Shared Components" on page 51.
- In a global zone, if non-global zones are present, instead of upgrading all shared components currently installed and installing any missing shared components required by a selected component, the installer synchronizes all Java ES shared components, whether or not they are needed by any specific product component. This allows all shared components to be propagated to non-global zones, thus assuring that there is no intermixing of shared component versions in non-global zones.

Upgrade of Product Components

A new capability has been implemented in the Java ES Release 5 to upgrade product components in a few special cases: Application Server, Message Queue, HADB, and Java DB. When the Java ES installer detects the previously installed release versions of these product components, it marks them as upgradable in the Component Selection page. If any of these four product components are selected, the installer will upgrade them using logic similar to that used for a fresh installation.

In particular, before upgrading a selected product component, the installer checks for the existence of current and previous versions of shared components. If the installer detects that a shared component required by the selected component is of a previous version or is missing, the installer will upgrade all shared components currently installed and install any missing shared components required by the selected component. This behavior, which meets the requirements described in "Synchronize All Shared Components" on page 51, applies to non-zone operating systems, global zones, and all non-global zones.

However, there are three exceptions to this behavior:

- In sparse root zones, some shared components cannot be installed or upgraded and the upgrade operation is halted until such time as such shared components have been installed or upgraded in the global zone. (For more information, see "Shared Components and Sparse Root Zones" on page 48) The installer provides the following message: "The following shared components, required by the components you have selected, cannot be installed or upgraded in a sparse root zone. Please install or upgrade these shared components in the global zone before proceeding. Use the All Shared Components option." (For more information, see "Synchronize All Shared Components" on page 51.)
- Both Application Server and Message Queue are bundled with the Solaris operating system. Neither of these versions can be directly upgraded in a sparse-root zone. For the details regarding these two bundled components, see "Product Component Special Cases" on page 57.
- In a global zone, if non-global zones are present, instead of upgrading all shared components currently installed and installing any missing shared components required by a component selected for installation, the installer synchronizes all Java ES shared components, whether or not they are needed at that time for any of the components selected for installation. This allows all shared components to be propagated to non-global zones, thus assuring that there is no intermixing of shared component versions in the non-global zones.

Note – There are a number of special cases or exceptions that might interfere with the installation or upgrade of product components in non-global zones. These cases are described in "Special Cases or Exceptions" on page 57.

Synchronize All Shared Components

A shared component synchronization option is provided to meet situations in which all shared components must be synchronized. When the All Shared Components option is selected, the installer will upgrade all shared components currently installed and install any missing shared components, whether or not they are needed by any specific product component. This option applies to global zones and whole root zones, but not to sparse root zones.

The All Shared Components option, is needed in the following two zone-based scenarios:

- Manually upgrading product components. The All Shared Components option is needed to perform the shared component installation and upgrade needed when upgrading product components that cannot be upgraded using the Java ES installer.
- Installations or Upgrades in a Sparse Root Zone. Some shared components cannot be installed in default sparse root zones. (For details, see "Installation of Product Components" on page 50 and "Upgrade of Product Components" on page 50.) Hence when running the installer in sparse root zones, you might first be required to synchronize shared components in the global zone, depending on the shared components involved. You use the All Shared Components option in the global zone to perform the shared component installation and upgrade required in this case.

Summary of Java ES Installer Behavior Regarding Shared Components

The behaviors described above are summarized in the following table, which shows how the Java ES installer's treatment of shared components depends on the zone context as well as what has been selected in the component selection page.

TABLE A–1 Installer Behavior Regarding Shared Components

Zones Context	Product Component Selected	All Shared Components Selected
Non-zone operating System	Upgrade all shared components currently installed	Upgrade all shared components currently installed
	Install any missing shared components required by the selected product component	Install any missing shared components, whether or not they are needed by any specific product component
Global zone: no non-global zones	Upgrade all shared components currently installed	Upgrade all shared components currently installed
	Install any missing shared components required by the selected product component	Install any missing shared components, whether or not they are needed by any specific product component
Global zone: non-global zones exist	Upgrade all shared components currently installed	Upgrade all shared components currently installed Install any missing shared components, whether or not they are needed by any specific product component
	Install any missing shared components, whether or not they are needed by any specific product component	

TABLE A–1 Installer Behavior Regarding Shared Components *(Continued)*

Zones Context	Product Component Selected	All Shared Components Selected
Whole root zone	Upgrade all shared components currently installed	Upgrade all shared components currently installed
	Install any missing shared components required by the selected product component	Install any missing shared components, whether or not they are needed by any specific product component
Sparse root zone	Cannot upgrade or install some shared components in read-only directories. If installer encounters such shared components, it blocks and instructs user to manage shared components in the global zone.	Cannot upgrade or install some shared components in read-only directories. Installer therefore blocks and instructs user to manage shared components in the global zone.

Recommended Use of Zones with Java ES

While the deployment of Java ES in a multi-zone environment has the general objective of providing for product component runtime isolation and efficient resource utilization, there area number of more specific objectives for which a multi-zone environment can be used. These are discussed in as discussed in "Why Use Zones for Java ES?" on page 45. Installation and administration strategies for Java ES in a multi-zone environment depend largely on which of these objectives you are trying to achieve.

Table A–2 compares five scenarios, the installation and administration strategies to which they correspond, and the objectives that they are meant to achieve. While mixing of these scenarios might be possible in some cases, the results can be problematic and are likely to cause administrative mayhem. Hence, Java ES Release 5 generally does not support deployments that mix these scenarios.

Also, Scenario 1 and Scenario 5 are problematic, so Java ES Release 5 does not currently support them (though accommodation might be made for specific product components in the case of Scenario 5).

TABLE A–2 Zones Installation and Administration Strategies for Java ES

Scenario (Installation Strategy)	Administration Strategy	Objective (See "Why Use Zones for Java ES?" on page 45)	Comments
1: Install product components and shared components in the global zone with propagation enabled. No components installed in non-global zones.*	Component life-cycle management: Global administrator Configuration and runtime management: Zones administrators	Centralized product component life-cycle management Organizational independence for product component configuration and runtime management	Problematic: Not yet supported for Java ES product components, except for Message Queue. Requires that product components support installation in global zone but configuration and runtime management in non-global zones.
2: Install shared components in the global zone and product components in whole root zones	Shared component life-cycle management: Global administrator Product component life-cycle management: Zones administrators Configuration and runtime management: Zones administrators	Centralized shared component life-cycle management Organizational independence for product component life-cycle, configuration, and runtime management	Mostly applicable when all components are of the same Java ES version, or when upgrading all product components in all whole root zones.
3: Install shared components in the global zone and product components in sparse root zones**	Same as Scenario #2	Centralized management of shared component life cycle. Organizational independence for product component life-cycle, configuration, and runtime management Increased resource efficiencies over Scenario #2 (See "Whole Root Zones vs. Sparse Root Zones" on page 44)	This scenario recommended when installing product components in sparse root zones. (Some shared components cant be installed in sparse root zones and must therefore be installed in global zone.)
4: Install product components and shared components in whole root zones	Component life-cycle management: Zones administrators Configuration and runtime management: Zones administrators	Version separation	No shared components or product components should be installed in the global zone. Recommended scenario for whole root zones.

TABLE A–2 Zones Installation and Administration Strategies for Java ES *(Continued)*

Scenario (Installation Strategy)	Administration Strategy	Objective (See "Why Use Zones for Java ES?" on page 45)	Comments
5: Install product components and shared components in sparse root zones.	Same as Scenario #4	Organizational independence for product component life-cycle, configuration, and runtime management Increased resource efficiencies over Scenario #4 (See "Whole Root Zones vs. Sparse Root Zones" on page 44)	Problematic. Cannot generally be implemented because a number of shared components cannot be installed in sparse root zones.

* Scenario 1 does not distinguish between whole root and sparse root zone environments; it assumes that no product components are installed in non-global zones. Installation of product components in non-global zones is covered in Scenarios 2-5.

** Scenario 3 assumes that `/opt` has not been made a read-only directory in the sparse root zone. If `/opt` were read-only, most Java ES product components could not be installed in sparse root zones and would have to be installed in the global zone, as in Scenario 1, as an alternative approach.

Recommended Practices

With Table A–2 in mind, here are a number of recommended practices:

- Plan your Java ES zones deployment strategy in advance depending on the objective in "Why Use Zones for Java ES?" on page 45that you are trying to achieve. Different objectives require different installation and administration strategies, as shown by the different scenarios of Table A–2.
- Avoid mixing scenarios. In particular:
  - Keep your Java ES zones deployment and administration strategy as simple as possible. Do not mix whole root and sparse root deployments of Java ES components on the same computer. (Procedures and practices needed to support sparse root zone deployments as in Scenario 3 can interfere with whole root zone deployments as in Scenario 4.)
  - Do not install the same Java ES product component in both the global zone and non-global zones, even if they are of different versions. (Procedures needed to upgrade a global zone installation as in Scenario 1 can break the non-global zone installations as in Scenario 4.)
  - When Release 4 (or earlier) Java ES components have been installed in a whole root zone, do not install Java ES Release 5 components (neither product components nor shared components) in the global zone, and do not upgrade Java ES components to

Release 5 in the global zone. In other words, Scenario 2 is not supported when there are pre-existing Java ES installations in a whole root zone. (Installation or upgrade in the global zone could result in a mixing of Release 4 and Release 5 files in the whole root zone.)

- Recommended installation practices:
  - If you want to run different Java ES product components in different zones, install the product components in non-global zones (Scenario 2, 3, 4, 5).
  - If you want to run different Java ES product components in different zones but centrally manage shared component life cycles, synchronize shared components in the global zone, then install the product components in non-global zones (Scenario 2, 3). (This is a recommended practice whenever installing product components in sparse root zones.)
  - If you want to achieve version separation of Java ES product components or, for other reasons to isolate deployments of Java ES product components (Scenario 4), then install and configure all Java ES components in whole root zones. Do not install any Java ES components in the global zone.
- Recommended upgrade practices:
  - If you want to upgrade all installed Release 4 product components to Release 5, synchronize all Java ES shared components in the global zone, then perform the upgrade of the desired product components in the zones where they have been installed. (Release 5 shared components are backwardly compatible.)
  - If you have Release 4 or Release 5 product components installed in a non-zones environment, and you wish to add non-global zones to the environment and install product components in the new non-global zones, be sure to do so in accordance with the recommended practices above. This might mean uninstalling components in the global zone and reinstalling them in non-global zones.

Deployment Architectures

The scenario descriptions in Table A–2 and the recommended practices noted above do not include recommended Java ES deployment architectures for a multi-zone environment. Such architectures would be an adaptation of deployment architectures created for multi-computer network environments. In other words, the availability of multi-zone environments does not change the basic deployment design approaches for achieving high performance, high availability, scalability, security, and serviceability for Java ES deployment systems. What a multi-zone environment allows you to do is consolidate such deployment architectures into fewer computers.

Details of how to adapt a Java ES deployment architecture to a multi-zone environment, however, are highly dependent on the administrative strategies you desire, as discussed in previous sections. Deployment architectures also depend on your strategy for achieving high availability.

Note that Table A–2 and the recommended practices above do not include recommended procedures for implementing the scenarios that are described. In some cases the order in which Java ES components are installed and the order in which non-local zones are created can be important.

Special Cases or Exceptions

There are a number of special cases that arise mostly from the fact that some Java ES shared components and some Java ES product components are bundled with Solaris 10. By virtue of this bundling, these Java ES components exist in the global zone, and therefore in any non-global zone that is created from the global zone.

Product Component Special Cases

- Message Queue is bundled with Solaris 10, and, as a result, is automatically propagated when non-global zones are created (unless you have first removed Message Queue from the global zone). Message Queue cannot be installed in a sparse root zone. When installed or upgraded in a global zone by the Java ES installer, Message Queue, by default, is propagated to non-global zones, unlike other product components.
- Application Server is bundled with Solaris 10, and, as a result, is automatically propagated when non-global zones are created (unless you have first removed Application Server from the global zone). When propagated in this way, the bundled Application Server, which is installed in `/usr`, cannot be upgraded by the Java ES installer in a sparse root zone (by default `/usr` is read-only). To address this problem, the bundled Application Server must be manually removed from the global zone before installing the Release 5 Application Server in a sparse root zone.
- Sun Cluster can only be installed in a global zone. Sun Cluster is not supported in non-global zones.

Shared Component Special Cases

- SJWC packages that are bundled with Solaris 10 (Update 1 and Update 2) cannot be removed by the Java ES installer. These older SJWC packages have had the `SUNW_PKG_ALLZONES` set to True, which means the package must be identical in all zones and can only be managed by the global administrator. As a result these packages must be manually removed in the global zone to be replaced by the correct packages.

  If the Java ES installer is attempting to install a selected component in a non-global zone and detects that SJWC needs to be upgraded, the installer will block. This happens when installing on Solaris 10, Update 1 and 2.

As a workaround, a special script has been developed that will remove the old packages of SJWC from the global zone and replace them with SJWC 2.2.6, which has the correct zones propagation attribute setting. As a result SJWC 2 2.6 is propagated to all non-global zones.

- Common Agent Container. Version 1.1 is installed only when Sun Cluster, Sun Cluster GE, or Sun Cluster Agents are installed. It is not installed when the Synchronize All Shared Components option is selected. Only version 2.0 is installed in that case.
- Sun Explorer Data Collector. This shared component is installed only when Sun Cluster, Sun Cluster GE, or Sun Cluster Agents are installed. It is not installed when the All Shared Components option is selected.

An Illustrative Example: Install Application Server in a Sparse Root Zone

The following example is provided to draw out some of the complexities involved in Java ES zones support. In this example, the objective is to install Application Server in a Solaris 10 sparse root zone. This installation is complicated by the fact that Application Server (as well as Message Queue, upon which Application Server depends), is bundled with Solaris 10, and therefore the bundled version is installed in all non-global zones. For more information, see "Product Component Special Cases" on page 57.

To install Application Server in a sparse root zone, you must first remove the bundled version. (You cannot simply upgrade the bundled version in the sparse root zone because it is installed in a read-only directory). To remove the bundled version from the sparse root zone, you must remove it in the global zone.

In addition, Message Queue is installed in the global zone, representing a departure from Scenario 3 of Table A–2, in which only shared components (not product components) are installed in the global zone. However, Message Queue cannot be installed in a sparse root zone, because it installs in a read-only directory, so it must be installed and upgraded in the global zone.

The procedure is as follows:

1. Verify Solaris 10 is running on your system.

   This example assumes a clean version of Solaris 10 with no Java ES components having been explicitly installed in the global zone.

2. Create a sparse root zone (configure, install, and boot it).

   This zone will include any Java ES components that are already installed in the global zone, namely the versions of Message Queue and Application Server bundled with Solaris 10.

3. Remove the bundled version of Application Server from the global zone.

   This must be performed by manually removing Application Server packages:

```
pkgrm SUNWascmnse SUNWaslb SUNWasut ...
```

Where the full set of packages can be obtained using the following command:

```
pkginfo -I|grep -I application server
```

The results would include packages such as:

```
SUNWascmnse, SUNWaslb, SUNWasut, SUNWasac, SUNWasdem, SUNWasman, SUNWaswbcr,
SUNWasacee, SUNWashdm, SUNWasmanee, SUNWascml, SUNWasJdbcDrivers, SUNWasu,
SUNWascmn, SUNWasjdoc, SUNWasuee
```

And might include localization packages as well:

```
SUNWLocaleasacee, SUNWLocaleascmnse, SUNWLocaleasu, SUNWLocaleasuee
```

The removal of Application Server from the global zone is propagated to the sparse root zone that was created in Step 2. (This step and Step 2 can be performed in reverse order.)

4. Install Java ES 5 Shared Components in the global zone.
   a. Run the Java ES installer in the global zone.
   b. Select All Shared Components from the component selection panel. Do not select any other component.
   c. Complete the synchronization of shared components. All of the shared components are now synchronized in the global zone and propagated to all non-global zones.
5. Upgrade Message Queue in the global zone.

   The version of Message Queue bundled with Solaris 10 is already installed in the sparse root zone by virtue of Step 2. To upgrade Message Queue in the sparse root zone, simply upgrade it in the global zone; the upgrade will propagate to the sparse root zone. (Message Queue is the only product component which cannot be installed in a sparse root zone, but when installed in the global zone, will propagate to non-global zones.)
   a. Run the Java ES installer in the global zone.
   b. Select Message Queue in the component selection panel. Do not select any other component.
   c. Complete the upgrade of Message Queue.
6. Install Application Server in the sparse root zone.
   a. Run the Java ES installer in the sparse root zone.
   b. Select Application Server in the component selection panel. Do not select any other component for upgrade. De-select Message Queue if it is selected.
   c. Complete the installation of Application Server.

Index

L

M

N

P

Q

R

S

T

W