Sun Java Enterprise System 5 Technical Overview

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

Part No: 820–0167–10
March 2007

080731@20490

Contents

Tables

Figures

Preface

The *Sun Java Enterprise System 5 Technical Overview* introduces the technical and conceptual foundations of Sun Java™ Enterprise System (Java ES). It also describes Java ES components, architecture, processes, and features.

This overview describes the technical concepts and terminology used in the Java ES documentation set. Key technical terms are explained in the last section of each chapter.

Who Should Use This Book

This book is meant for individuals who are designing, deploying, or maintaining software solutions based on Java ES, including business analysts, system architects, field engineers, and system administrators.

Individuals reading this book should have some familiarity with the following technologies:

- General networking concepts
- Security fundamentals relating to authentication and authorization
- The Java programming language
- Java 2 Platform, Standard Edition (J2SE™ platform) components and Java 2 Platform, Enterprise Edition (J2EE™ platform) components

How This Book Is Organized

This book is organized in the following chapters:

- Chapter 1, "Introduction to Java Enterprise System," introduces Java ES and the tasks involved in using the system.
- Chapter 2, "Java ES Solution Architectures," describes an architectural framework for designing Java ES solution architectures and provides an example architecture based on that framework.
- Chapter 3, "Java ES Integration Features," provides information about features that play key roles in integrating Java ES components into a single software system.
- Chapter 4, "Java ES Solution Life Cycle," describes concepts and terminology relevant to each phase of the Java ES solution life cycle.

- Appendix A, "Java ES Components," provides a listing of Java ES components.

Java Enterprise System Documentation Set

The Java ES system 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. Java ES information and resources are also available at `http://www.sun.com/bigadmin/hubs/javaes/`.

TABLE P–1 Java Enterprise System Documentation

Document Title	Contents
Sun Java Enterprise System 5 Release Notes for UNIX *Sun Java Enterprise System 5 Release Notes for Microsoft Windows*	Contains the latest information about Java ES, including known problems. In addition, components have their own release notes listed in the Release Notes Collection (`http://docs.sun.com/coll/1315.2`).
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.
Sun Java Enterprise System 5 Installation Guide for UNIX *Sun Java Enterprise System 5 Installation Guide for Microsoft Windows*	Guides you through the process of installing Java ES. 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 on the Solaris^{TM} Operating System (Solaris OS) and Linux operating environment.

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

Document Title	Contents
Sun Java Enterprise System 5 Upgrade Guide for UNIX	Provides instructions for upgrading to Java ES 5 from previously installed versions.
Sun Java Enterprise System 5 Upgrade Guide for Microsoft Windows	
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 create monitoring rules.
Sun Java Enterprise System Glossary	Defines terms that are used in Java ES documentation.

Typographic Conventions

The following table describes the typographic conventions 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.

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^{SM} 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, and 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 820–0167–10.

CHAPTER 1

Introduction to Java Enterprise System

Sun Java^{TM} Enterprise System (Java ES) is a set of software components that provides services needed to support enterprise-strength applications distributed across a network or Internet environment. These applications are referred to as distributed enterprise applications. This book focuses on the software components of Java ES and the services they provide.

This chapter introduces Java ES and the tasks involved in using the system. The chapter contains the following sections:

- "Why You Need Java ES" on page 15
- "Java ES Components" on page 17
- "Components in Sun Java Suites" on page 21
- "Working With Java ES" on page 24
- "Key Terms in This Chapter" on page 28

Why You Need Java ES

Today's business requirements demand software solutions that are distributed across a network or Internet environment and have high levels of performance, availability, security, scalability, and serviceability.

Java ES provides infrastructure services needed to support such distributed enterprise applications, which generally have the following characteristics:

- **Distributed.** The application consists of interacting software components deployed across a networked environment that might include geographically remote sites. These distributed components, running on the various computers in the environment, work together to deliver specific business functions to end users and to other business applications.
- **Enterprise-strength.** The application's scope and scale meet the needs of a production environment or Internet service provider. The application typically spans an entire enterprise, integrating many departments, operations, and processes into a single software

system. The application must meet high quality of service requirements regarding performance, availability, security, scalability, and serviceability.

Distributed enterprise applications require underlying infrastructure services that allow their distributed components to communicate with each other, coordinate their work, implement secure access, and so forth. These infrastructure services are supported by a hardware environment of computers and network links. This hardware environment includes SPARC® and x86 (Intel and AMD) hardware architectures.

The overall layering scheme is shown in the following figure. For the most part, Java ES provides the distributed infrastructure services layer shown in the figure.

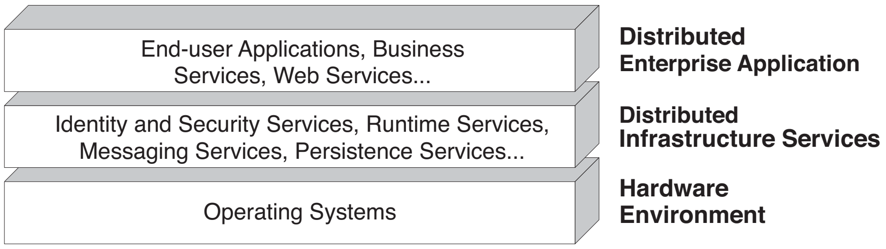
End-user Applications, Business Services, Web Services...
Identity and Security Services, Runtime Services, Messaging Services, Persistence Services...
Operating Systems
Distributed Enterprise Application
Distributed Infrastructure Services
Hardware Environment

FIGURE 1–1 Support Needed for Distributed Enterprise Applications

Among the featured services provided by Java ES are the following:

- **Portal services.** These services enable employees, telecommuters, knowledge workers, business partners, suppliers, and customers to access corporate resources from inside or outside the corporate network. These services provide access capabilities to user communities anytime and anywhere, delivering personalized integration, aggregation, security, mobile access, and search.
- **Communication and collaboration services.** These services enable the secure interchange of information among diverse user communities. Specific capabilities include messaging, real-time collaboration such as instant messaging and conferencing, and calendar scheduling in the context of the user's business environment.

Note – This book refers to components from the Sun Java Communications Suite, which depend upon Java ES components and are used within Java ES deployment architectures. Communication and collaboration components are not included in Java ES.

- **Network identity and security services.** These services improve security and protection of key corporate information assets by ensuring that appropriate access control policies are enforced across all communities, applications, and services on a global basis. These services work with a repository for storing and managing identity profiles, access privileges, and application and network resource information.

- **Web container and application services.** These services enable distributed components to communicate with one another at runtime and support the development, deployment, and management of applications for a broad range of servers, clients, and devices. These services are based on J2EE technology.

Java ES also provides services that enhance availability, scalability, serviceability, and other application or system qualities. Among the quality of service features provided by Java ES are the following:

- **Availability services.** These services provide near-continuous availability for application components and for the infrastructure components that support them.
- **Access services.** These services provide Internet or browser-based access to Java ES services.
- **Monitoring services.** These services provide real-time information about Java ES components.

You can deploy one or more Java ES services, each of which might include a number of Java ES components.

Java ES Components

Java ES is an integration of discrete software products and components into a single software system. This integration is facilitated by a number of system-level features, as listed below:

- All components are synchronized on a common set of shared libraries.
- All Java ES components are installed using a single installer.
- All Java ES components can share an integrated user identity and security management system.
- All Java ES components have a common monitoring framework.

These features are described in subsequent chapters of this book. The focus of this section is to introduce the components that are integrated into Java ES. These system components can be grouped into three main categories, as shown in the following figure.

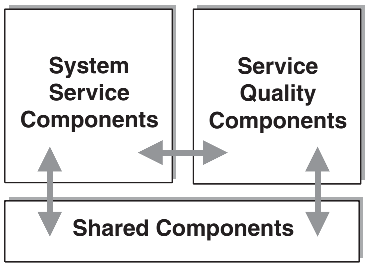

FIGURE 1–2 Categories of Java ES Components

The components provide services as follows:

- **System service components.** These components provide the main Java ES infrastructure services that support distributed enterprise applications.
- **Service quality components.** These components enhance the availability, security, scalability, serviceability and other components of system service components and distributed application components.
- **Shared components.** These components provide the environment in which many system service components and service quality components run.

For a list of Java ES components, see Appendix A, "Java ES Components."

System Service Components

A number of Java ES components provide the main services that support distributed software solutions. These system services include portal services, identity and security services, web container services, J2EE application services, and persistence services.

The system service components that provide these distributed services and the services they provide are listed alphabetically and briefly described in the following table. Each system service component is a multithreaded server process capable of supporting a large number of clients. For more details on each component, see "System Service Components" on page 67.

TABLE 1–1 Java ES System Service Components

Component	System Services Provided
Sun Java System Access Manager	Provides access management and digital identity administration services. Access management services include authentication (including single sign-on) and role-based authorization for access to applications and/or services. Administration services include centralized administration of individual user accounts, roles, groups, and policies.

TABLE 1–1 Java ES System Service Components *(Continued)*

Component	System Services Provided
Sun Java System Application Server	Provides J2EE container services for Enterprise JavaBeans™ (EJB) components, such as session beans, entity beans, and message-driven beans. The container provides the infrastructure services needed for tightly coupled distributed components to interact, making the Application Server a platform for the development and execution of e-commerce applications and web services. Application Server also provides web container services.
Sun Java System Directory Server	Provides a central repository for storing and managing intranet and Internet information such as identity profiles (employees, customers, suppliers, and so forth), user credentials (public key certificates, passwords, and pin numbers), access privileges, application resource information, and network resource information.
Java DB[1]	Provides a lightweight database for Java application development. Java DB is Sun's supported distribution of the open source Apache Derby 100% Java technology database.
Sun Java System Message Queue	Provides reliable, asynchronous messaging between loosely coupled distributed components and applications. Message Queue implements the Java Message Service (JMS) API specification and adds enterprise features such as security, scalability, and remote administration.
Sun Java System Portal Server	Provides key portal services, such as content aggregation and personalization, to browser-based clients accessing business applications or services. Portal Server also provides a configurable search engine.
Sun Java System Service Registry	Provides a registry and repository to support web service-oriented architecture (SOA) applications. Service Registry implements industry standards for registering and discovering web services as well as for managing the associated information and facts, artifacts, such as XML schema, business process rules, access control, version control, and so forth.
Sun Java System Web Server	Provides J2EE web container services for Java web components, such as Java servlet and JavaServer Pages™ (JSP™) components. Web Server also supports other web application technologies for delivering static and dynamic web content, such as CGI scripts and Sun Java System Active Server Pages.

[1] Java ES 5 is the first release to include Java DB as a product component. Java DB was first released as a shared component named Derby Database and was included in Java ES 2005Q4.

Service Quality Components

In addition to the system service components shown in Table 1–1, Java ES includes a number of components used to enhance the quality of services provided by system service components. Service quality components can also enhance custom-developed application services. The service quality components fall into the following categories:

- Availability components

- Access components
- Monitoring components

Availability Components

Availability components provide near-continuous uptime for system service components and custom application services. The availability components included in Java ES and the services they provide are shown in the following table. For more details on each component, see "Availability Components" on page 71.

TABLE 1–2 Java ES Availability Components

Component	Availability Services Provided
High Availability Session Store	Provides a data store that makes application data, especially session state data, available even in the case of failure.
Sun Cluster	Provides high availability and scalability services for Java ES, the applications that run on top of the Java ES infrastructure, and the hardware environment in which both are deployed.
Sun Cluster Geographic Edition[1]	Protects applications from unexpected disruptions using multiple clusters that are geographically separated and a redundant infrastructure that replicates data between these clusters. Sun Cluster Geographic Edition software is a layered extension of Sun Cluster software.

[1] Java ES 5 is the first release to include Sun Cluster Geographic Edition as a Java ES product component.

Access Components

Access components provide front-end access to system services, often secure access from Internet locations outside an enterprise firewall. In addition to providing such access, many provide a routing and caching function as well. The access components included in Java ES and the services they provide are shown in the following table. For more details on each component, see "Access Components" on page 73.

TABLE 1–3 Java ES Access Components

Component	Access Services Provided
Sun Java System Portal Server (includes Secure Remote Access)	Provides secure, Internet access from outside a corporate firewall to Portal Server content and services, including internal portals.
Sun Java System Web Proxy Server	Provides for caching, filtering, and distribution of web content for both outgoing and incoming Internet requests.

Monitoring Components

Java ES includes a new monitoring feature that provides real-time system status and customizable monitoring jobs. Monitoring is implemented by the Sun Java System Monitoring Console product component, which is supported by the Sun Java System Monitoring Framework shared component. For more information, see "Monitoring Components" on page 74.

Shared Components

Java ES includes a number of locally installed shared libraries upon which many system service components and service quality components depend. Java ES shared components provide local services to Java ES product components running on the same host computer.

Shared components are often used to provide portability across different operating systems. Examples of Java ES shared components include: Java 2 Platform, Standard Edition (J2SE), Netscape Portable Runtime (NSPR), Network Security Services (NSS), Java Security Services for Java (JSS), and so forth. For a complete list, see "Shared Components" on page 75.

Shared components are installed automatically by the Java ES installer depending on the system service and service quality components that are being installed.

Components in Sun Java Suites

Java ES is available both as a single, end-to-end infrastructure software distribution and as individual suite distributions that target critical business needs. Java ES includes all Java ES components, while Sun Java System Suites include subsets of those components selected to meet particular business needs. The Java ES installer and uninstaller are included in all suite distributions, but are pared down to handle just the components in the suite. All shared components are also included in all suite distributions.

The contents of the individual suites and the business requirements each suite is intended to meet are listed in the following table.

TABLE 1–4 Components in Sun Java Suites

Suite	Business Requirement	Contents
Sun Java Application Platform Suite	Develop, deploy, and manage next-generation service-oriented architectures (SOA)	Access Manager Application Server Directory Server HADB Java DB Message Queue Monitoring Console Portal Server (includes Secure Remote Access and Mobile Access) Service Registry Web Proxy Server Web Server
Sun Java Availability Suite	Disaster recovery and high availability for mission-critical applications	Sun Cluster software Sun Cluster Agents Sun Cluster Geographic Edition

TABLE 1–4 Components in Sun Java Suites *(Continued)*

Suite	Business Requirement	Contents
Sun Java Communications Suite[1]	Secure and reliable messaging and collaboration services	Access Manager Application Server Calendar Server* Communications Express* Delegated Administrator* Directory Server HADB Instant Messaging* Java DB Message Queue Messaging Server* Monitoring Console Web Proxy Server Web Server
Sun Java Identity Management Suite	User identity management across computing infrastructures and application environments	Access Manager Application Server Directory Server HADB Java DB Message Queue Monitoring Console Web Server

[1] Components with asterisks (*) are communications components that are no longer included with Java ES or installed through the Java ES installer. These components are available as part of the Sun Java Communications Suite.

TABLE 1–4 Components in Sun Java Suites *(Continued)*

Suite	Business Requirement	Contents
Sun Java Web Infrastructure Suite	Web applications and services for small to medium-sized enterprises	Access Manager Application Server Directory Server HADB Java DB Message Queue Monitoring Console Service Registry Web Proxy Server Web Server

Working With Java ES

Creating business solutions based on Java ES software involves a number of standard tasks. These tasks vary in scope and difficulty depending on your starting point for the adoption of Java ES and on the nature of the solution that you are trying to create and deploy.

This section discusses two aspects of working with Java ES: the Java ES solution life cycle and the adoption scenarios that are typically involved.

Java ES Solution Life Cycle

The tasks involved in creating business solutions based on Java ES software can be divided into several phases, as shown in the following figure. The illustration also shows the category of Java ES user that generally performs the tasks.

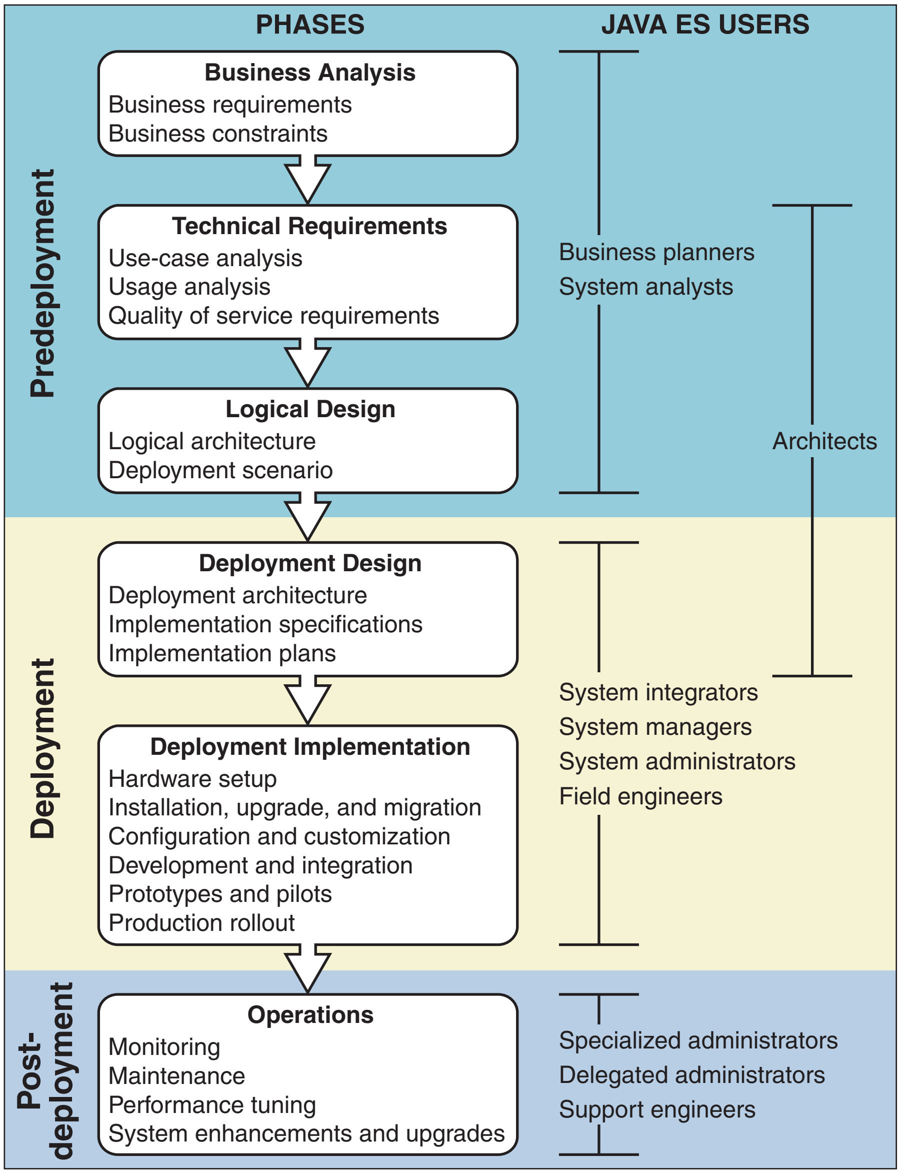

FIGURE 1–3 Solution Life Cycle Phases and User Categories

The life cycle phases shown in the preceding figure can be divided into the following general groupings:

- **Predeployment.** In these phases, a business need is translated into a deployment scenario, which is a logical architecture and a set of quality of service requirements. The deployment scenario serves as a specification for designing a deployment architecture.
- **Deployment.** In these phases, a deployment scenario is translated into a deployment architecture. This architecture can be used as a basis for project approval and budgeting. The deployment architecture is also the basis for an implementation specification that provides the details needed to deploy (build, test, and roll out) a software solution in a production environment.
- **Postdeployment.** In the operations phase, a deployed solution is run under production conditions and monitored and optimized for performance. The deployed solution is also upgraded to include new functionality as necessary.

The tasks in each life cycle phase shown in Figure 1–3 are discussed more fully in Chapter 4, "Java ES Solution Life Cycle."

Figure 1–3 shows the Java ES users who typically perform the tasks shown for the life cycle phases. The following table describes the skills and background for each category of user.

TABLE 1–5 Java ES User Categories for Life Cycle Tasks

User	Skills and Background	Phases
Business planner System analyst	Has general rather than in-depth technical knowledge. Understands strategic direction of the enterprise. Knows business processes, objectives, and requirements.	Business analysis Technical requirements Logical design
Architect	Is highly technical. Has broad knowledge of deployment architectures. Is familiar with latest technologies. Understands business requirements and constraints.	Technical requirements Logical design Deployment design
System integrator Field engineer System administrator System manager	Is highly technical. Is intimately familiar with information technology environments. Is experienced in implementing distributed software solutions. Knows network architecture, protocols, devices, and security. Knows scripting and programming languages.	Deployment design Deployment implementation
Specialized system administrator Delegated administrator Support engineer	Has specialized technical or product knowledge. Is familiar with hardware, platforms, directories, and databases. Is skilled at monitoring, troubleshooting, and upgrading software. Knows system administration for operating system platforms.	Operations

Java ES Adoption Scenarios

The business needs that lead to the adoption of Java ES vary widely. However, the high-level goal for nearly every Java ES deployment fits into one of the following adoption scenarios:

- **New system.** Starting with no existing software system, you deploy Java ES software to support a new business solution.
- **Enhancement.** Starting with an existing information technology (IT) infrastructure, you replace one, many, or all parts of that system with Java ES software. You typically replace systems or subsystems because they are too complicated, too limited, or too expensive to maintain. For example, you might require better security, higher availability, increased scalability, more flexibility, less complexity, additional capabilities (such as single sign-on), or better use of IT resources.
- **Extension.** Starting with an existing IT infrastructure, you deploy Java ES software that is not currently part of your system. You typically extend your software system because you need to meet new business needs. You might need new functional capabilities such as personalized aggregation of existing services through a Java ES portal or Java authentication and authorization for existing services.
- **Upgrade.** Starting with an IT infrastructure consisting of an earlier version of Java ES or of Sun products that predate Java ES, you upgrade to the most current version of Java ES components.

Each adoption scenario has its own considerations and challenges. Depending on your adoption scenario, the issues you need to address and the resources you need to invest in the life cycle phases shown in Figure 1–3 might vary.

The following considerations apply in varying degrees to the adoption scenarios:

- **Migration.** Enhancing or upgrading the existing infrastructure with new software often requires the migration of data from the existing system to the new system. The data could be configuration information, user information, or application information. You might also need to migrate business or presentation logic due to new programming interfaces.
- **Integration.** Adding new software to an existing system or replacing software subsystems often requires that you integrate new software components with the remaining subsystems. Integration might involve developing new interface layers, using J2EE connectors or resource adaptors, reconfiguring existing components, and implementing data transformation schemes.
- **Training.** Almost any change in infrastructure implies changes in IT procedures and skill sets. Your IT department must have adequate time to acquire new skills or transfer old skills to support Java ES technologies.
- **Hardware.** When you replace or enhance an existing system or subsystem, business constraints might require you to reuse existing hardware. Depending on your adoption scenario, hardware resources can become an important factor.

The following table summarizes the nature of the concerns that apply to each of the Java ES adoption scenarios.

TABLE 1–6 Java ES Adoption Scenario Concerns

Adoption Scenario	Migration	Integration	Training	Hardware
New system	Not a concern	Relatively easy to integrate new components	Can be a significant concern	Trade-offs between equipment costs and labor costs[1]
Enhancement	Can be a major concern	Need to integrate new components with existing system	Can be a significant concern	Can involve significant constraints due to existing equipment
Extension	Not normally a concern	Might need to integrate new components with existing system	Might be a significant concern	Usually requires new hardware with same trade-offs as with a new system
Upgrade	Can be a significant concern	Relatively easy to integrate upgraded components	Relatively minor concern	Relatively minor concern

[1] Using a few powerful computers generally increases equipment costs while requiring fewer IT resources. Using many smaller computers generally decreases equipment costs while requiring more IT resources.

Key Terms in This Chapter

This section explains key technical terms used in this chapter, with an emphasis on clarifying how these terms are used in the Java ES context.

adoption scenario	An overall reason for deploying Java ES software that describes the software system you start with and the goal you are trying to achieve. Four basic Java ES adoption scenarios exist: new system, replacement, extension, and upgrade.
component	A unit of software logic from which distributed applications are built. A component can be one of the system components included in Java ES or an application component that is custom developed. An application component usually conforms to a distributed component model (such as CORBA or the J2EE™ platform) and performs a specific computing function. These components, singly or combined, provide business services and can be encapsulated as web services.
distributed enterprise application	An application with logic that spans a network or Internet environment (the distributed aspect) and with a scope and scale that meet the needs of a production environment or service provider (the enterprise aspect).
end user	A person who uses a distributed application often through a graphical user interface such as an Internet browser or mobile device GUI. The number of concurrent end users supported by an application is an important determinant of the deployment architecture of the application.

service	A software function performed for one or more clients. This function might be at a very low level such as a memory management service or at a high level such as a credit check business service. A high-level service can consist of a family of individual services. Services can be local (available to local clients) or distributed (available to remote clients).
product component	Java ES system service components, which provide the main Java ES infrastructure services, and Java ES service quality components, which enhance those system services. Product components are selectable within the Java ES installer.
service quality component	A type of system component included in Java ES. These components enhance the availability, security, scalability, serviceability, and other qualities of system service components and distributed application components.
shared component	A type of system component included in Java ES. Shared components, usually libraries, provide local services to other system components.
system component	Any software package or set of packages included in Java ES and installed by the Java ES installer. Several kinds of system components exist: product components that provide Java ES infrastructure services, and shared components that provide local services to other system components.
system service	One or more distributed services that define the unique functionality provided by Java ES. System services normally require the support of a number of service quality components, a number of shared components, or some of both.
system service component	A type of system component included in Java ES. System services components provide the main Java ES infrastructure services: portal services, identity and security services, web and application services, and availability services.

CHAPTER 2

Java ES Solution Architectures

This chapter provides an overview of the architectural concepts upon which Java ES solutions are based. The chapter demonstrates how system service components and service quality components are used to support distributed enterprise solutions.

Java ES solution architectures have two aspects: a logical architecture and a deployment architecture. The logical architecture depicts the interactions between the logical building blocks (the software components) of a solution. The deployment architecture depicts the mapping of the logical architecture to a physical computing environment. Java ES components play important roles in both logical architectures and deployment architectures.

This chapter describes an architectural framework for designing Java ES solution architectures and an example architecture based on that framework. The chapter contains the following sections:

Java ES Architectural Framework

Java ES components support the deployment of distributed software solutions. To achieve the required functionality at the levels of performance, availability, security, scalability, and serviceability mandated by business requirements, these software solutions must be properly designed.

A number of architectural dimensions are involved in designing enterprise-strength solutions. These dimensions represent different perspectives from which to view the interactions of the many software components used to build such systems. In particular, the design of distributed systems involves the following three architectural dimensions:

- **Infrastructure service dependencies.** This dimension emphasizes the role of system service components in supporting distributed solutions (see "System Service Components" on page 18).
- **Logical tiers.** This dimension emphasizes the logical and physical independence of solution components for the purpose of deploying them across a network or Internet environment.
- **Quality of service.** This dimension emphasizes how quality of service requirements such as availability, security, scalability, and serviceability are achieved, including the role of service quality components (see "Service Quality Components" on page 19).

These three dimensions of solution architecture are shown in the following figure.

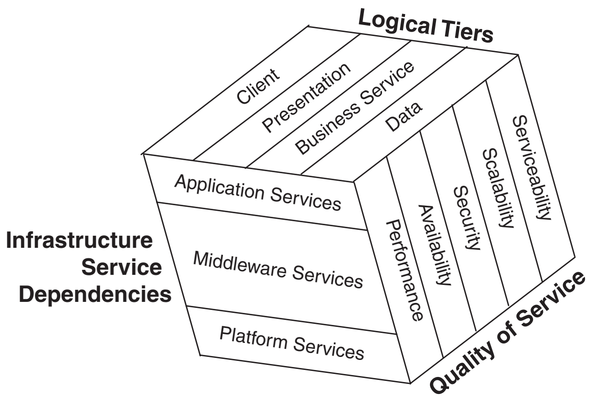

FIGURE 2–1 Dimensions of a Java ES Solution Architecture

Together these three dimensions represent a single framework that incorporates the relationships between the software components, both application components and infrastructure components, that are needed to achieve the service functions and service quality required of a software solution.

The following sections describe the three dimensions individually, followed by a synthesis of the three dimensions into a unified framework.

Dimension 1: Infrastructure Service Dependencies

The interacting software components of distributed enterprise applications require underlying infrastructure services that allow the distributed components to communicate with each other, coordinate their work, implement secure access, and so forth. This section explains the key role played by a number of Java ES components in providing these infrastructure services.

Infrastructure Service Levels

In designing a distributed software system, whether it consists mostly of custom-developed components or out-of-the-box Java ES components, you need to incorporate a number of infrastructure services. These services operate at many levels.

The infrastructure service dependencies of solution architecture are illustrated in Figure 2–2. The levels shown in this figure are an expanded view of the infrastructure service layer of Figure 1–1. The hierarchy of services in Figure 2–2 and the dependencies between them constitute an important dimension of a solution's logical architecture. These infrastructure services provide the main rationale for Java ES system service components (see "System Service Components" on page 18).

In general, the services shown in the following figure divide into three broad groupings: low-level platform services, high-level application services, and a group of middleware services named for their location between the other two groupings.

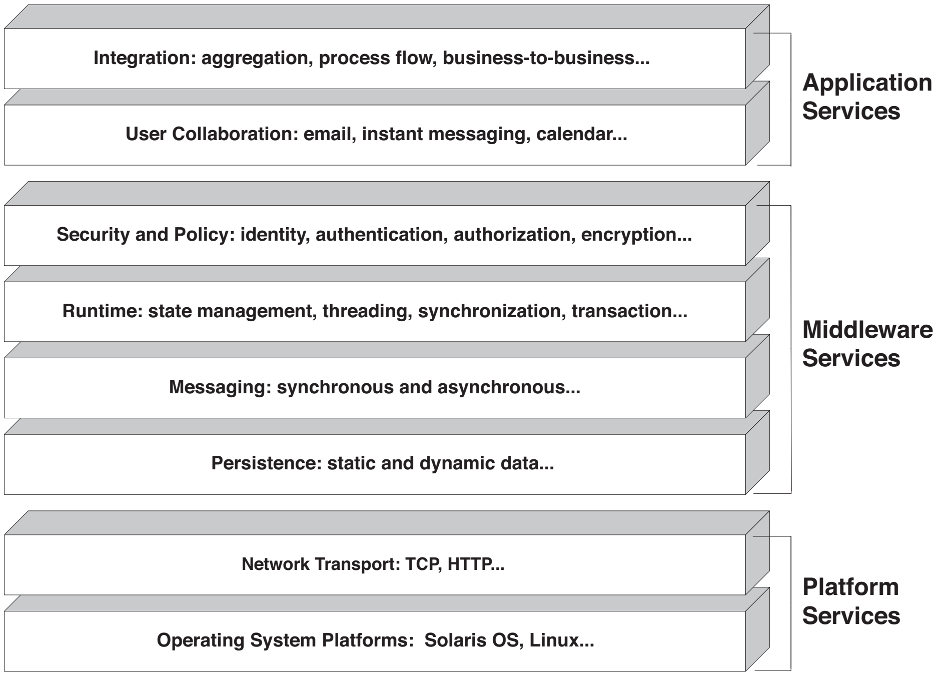
Integration: aggregation, process flow, business-to-business...
User Collaboration: email, instant messaging, calendar...
Application Services
Security and Policy: identity, authentication, authorization, encryption...
Runtime: state management, threading, synchronization, transaction...
Messaging: synchronous and asynchronous...
Persistence: static and dynamic data...
Middleware Services
Network Transport: TCP, HTTP...
Operating System Platforms: Solaris OS, Linux...
Platform Services

FIGURE 2–2 Dimension 1: Infrastructure Service Levels

The following descriptions of the different infrastructure service levels refer to Java programming language artifacts, where relevant, and are listed from lowest to highest, as shown in Figure 2–2:

- **Operating system platforms.** Provides the basic support for any process running on a computer. The operating system manages physical devices as well as memory, threads, and other resources necessary to support the Java Virtual Machine (JVM™ machine).
- **Network transport.** Provides basic networking support for communication between distributed application components running on different computers. These services include support for protocols such as TCP and HTTP. Other higher-level communication protocols (see the messaging level) depend on these basic transport services.
- **Persistence.** Provides support for accessing and storing both static data (such as user, directory, or configuration information) and dynamic application data (information that is frequently updated).
- **Messaging.** Provides support for both synchronous and asynchronous communication between application components. Synchronous messaging is the real-time sending and receipt of messages and includes remote method invocation (RMI) between J2EE components and SOAP interactions with web services. Asynchronous messaging is communication in which the sending of a message does not depend on the readiness of the consumer to immediately receive it. Asynchronous messaging specifications, for example, Java Message Service (JMS) and ebXML, support guaranteed reliability and other messaging semantics.
- **Runtime.** Provides support required by any distributed component model, such as the J2EE or CORBA models. In addition to the remote method invocation needed for tightly coupled distributed components, runtime services include component state (life cycle) management, thread pool management, synchronization (mutex locking), persistence services, distributed transaction monitoring, and distributed exception handling. In a J2EE environment, these runtime services are provided by EJB, web, and message-driven bean containers in an application server or web server.
- **Security and policy.** Provides support for secure access to application resources. These services include support for policies that govern group or role-based access to distributed resources as well as single sign-on capabilities. Single sign-on allows a user's authentication to one service in a distributed system to be automatically applied to other services (J2EE components, business services, and web services) in the system.
- **User collaboration.** Provides services that play a key role in supporting direct communication between users and collaboration among users in enterprise and Internet environments. These services are application-level business services, normally provided by stand-alone servers such as an email server or a calendar server.
- **Integration.** Provides the services that aggregate existing business services. Provides a common interface for accessing the services, as in a portal, or by integrating the services through a process engine that coordinates them within a production work flow. Integration can also take place as business-to-business interactions between different enterprises.

The service levels shown in Figure 2–2 reflect a dependence of the infrastructure services on one another, from the lowest-level operating system services to the highest-level application and integration services. Each service generally depends on services below it and supports services above it. Figure 2–2, however, does not represent a strict layering of infrastructure services. Higher-level services can directly interact with lower-level services without depending on intermediate levels. For example, some runtime services might depend directly on platform services without requiring any of the service levels in between. In addition, other service levels, such as a monitoring or management service, might also be included in this conceptual illustration.

Java ES Infrastructure Service Components

Java ES components implement the distributed infrastructure service levels shown in Figure 2–2. The positioning of system service components within the different levels is shown in the following figure.

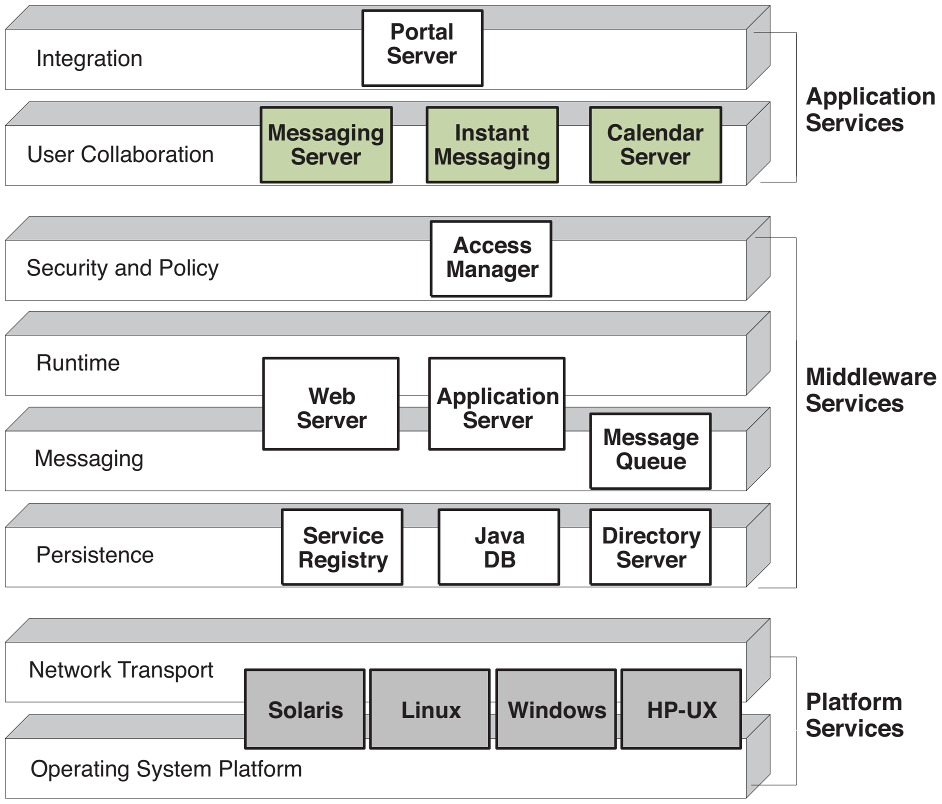

FIGURE 2–3 Java ES System Service Components

Note – The shaded boxes in the figure indicate components that are not included in Java ES. User collaboration components are not part of Java ES but are often deployed with Java ES components and used within Java ES architectures. These components are part of the Sun Java Communications Suite and are referenced in this document for illustrative purposes only. Also, operating system platforms are not formally part of Java ES, but are included to show the operating system platforms on which Java ES components are supported.

Java ES Infrastructure Service Dependencies

In general, each Java ES system service component shown in Figure 2–3 depends on components below it in the infrastructure and supports components above it. These dependency and support relationships are a key factor in designing logical architectures.

The following table shows the specific relationships between the Java ES system service components, listed from top to bottom, as shown in Figure 2–3.

TABLE 2–1 Relationships Between Java ES System Service Components

Component	Depends On	Provides Support To
Portal Server	Application Server or Web Server Access Manager Directory Server If configured to use corresponding channels: Calendar Server, Messaging Server, and Instant Messaging[1]	None
Access Manager	Application Server or Web Server Directory Server	Portal Server If configured for single sign-on: Calendar Server, Messaging Server, and Instant Messaging
Application Server	Message Queue Directory Server (for administered objects)	Portal Server Access Manager
Message Queue	Directory Server (for administered objects)	Application Server
Web Server	Access Manager (for access control)	Portal Server Access Manager

[1] Calendar Server, Messaging Server, and Instant Messaging components are available as part of the Sun Java Communications Suite.

TABLE 2–1 Relationships Between Java ES System Service Components *(Continued)*

Component	Depends On	Provides Support To
Directory Server	None	Portal Server Access Manager Calendar Server Messaging Server Instant Messaging
Service Registry	Java DB	Application Server-based components
Java DB	None	Service Registry

Dimension 2: Logical Tiers

The interacting software components of distributed enterprise applications can be viewed as residing in a number of logical tiers. These tiers represent the logical and physical independence of software components based on the nature of the services they provide.

The logical tier dimension of solution architecture is illustrated in the following figure.

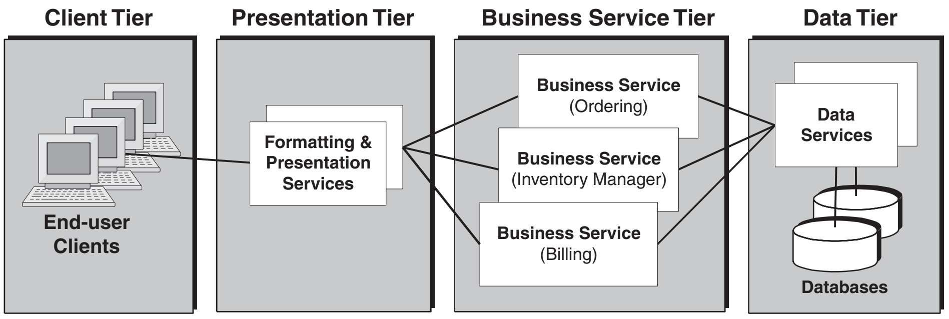
Client Tier
Presentation Tier
Business Service Tier
Data Tier
End-user Clients
Formatting & Presentation Services
Business Service (Ordering)
Business Service (Inventory Manager)
Business Service (Billing)
Data Services
Databases

FIGURE 2–4 Dimension 2: Logical Tiers for Distributed Enterprise Applications

For the most part, logical tier architectures represent the distributed enterprise application layer of Figure 1–1 . The Java ES system service components discussed in "Infrastructure Service Levels" on page 33 provide support to application components in all of the logical tiers shown in Figure 2–4. While logical tier concepts apply mostly to custom enterprise applications, they also apply to collaboration services provided by Sun Java Communications Suite components and some portal services.

Description of Logical Tiers

This section provides brief descriptions of the four logical tiers shown in Figure 2–4. The descriptions refer to application components implemented using the J2EE platform component model. However, other distributed component models, such as CORBA, also support this architecture.

- **Client tier.** The client tier consists of application logic accessed directly by an end user through a user interface. The logic in the client tier could include browser-based clients, Java components running on a desktop computer, or Java™ 2 Platform, Micro Edition (J2ME™ platform) mobile clients running on a handheld device.
- **Presentation tier.** The presentation tier consists of application logic that prepares data for delivery to the client tier and processes requests from the client tier for delivery to back-end business logic. The logic in the presentation tier typically consists of J2EE components such as Java servlet or JSP components that prepare data for delivery in HTML or XML format or receive requests for processing. This tier might also include a portal service that can provide personalized, secure, and customized access to business services in the business service tier.
- **Business service tier.** The business service tier consists of logic that performs the main functions of the application: processing data, implementing business rules, coordinating multiple users, and managing external resources such as databases or legacy systems. Typically, this tier consists of tightly coupled components that conform to the J2EE distributed component model, such as Java objects, EJB components, or message-driven beans. Individual J2EE components can be assembled to deliver complex business services, such as an inventory service or a tax calculation service. Individual components and service assemblies can be encapsulated as loosely coupled web services within a service oriented architecture model that conform to Simple Object Access Protocol (SOAP) interface standards. Business services can also be built as stand-alone servers, such as an enterprise calendar server or messaging server.
- **Data tier.** The data tier consists of services that provide persistent data used by business logic. The data can be application data stored in a database management system or it can be resource and directory information stored in a Lightweight Directory Access Protocol (LDAP) data store. The data services can also include data feeds from external sources or data accessible from legacy computing systems.

Logical and Physical Independence

The architectural dimension illustrated in Figure 2–4 emphasizes the logical and physical independence of components, represented by four separate tiers. These tiers signify the partitioning of application logic across various computers in a networked environment:

- **Logical independence.** The four tiers in the architectural model represent logical independence: You can modify application logic in one tier (for example, in the business service tier) independently of the logic in other tiers. You can change your implementation of business logic without having to change or upgrade logic in the presentation or client tier.

This independence means, for example, that you can introduce new types of client components without having to modify business service components.

- **Physical independence.** The four tiers also represent physical independence: You can deploy the logic in different tiers on different hardware platforms (that is, different processor configurations, chip sets, and operating systems). This independence allows you to run distributed application components on the computers best suited to their individual computing requirements and best suited to maximizing network bandwidth.

  How you map application components or infrastructure components to a hardware environment (that is, your deployment architecture) depends on many factors, depending on the scale and complexity of your software solution. For very small deployments, a deployment architecture might involve only a few computers. For large-scale deployments, the mapping of components to a hardware environment might take into account factors such as the speed and power of different computers, the speed and bandwidth of network links, security and firewall considerations, and component replication strategies for high availability and scalability.

Tiered Architecture Applied to System Components

As shown in Figure 2–3, Java ES infrastructure service components provide the underlying infrastructure support for distributed software solutions. Some of these solutions include application-level services provided by Sun Java Communications Suite components and some Java ES components. These solutions use logical tier design approaches.

For example, the email communication services provided by Messaging Server are implemented using a number of logically distinct configurations of Messaging Server. These distinct configurations each provide a distinct set of services. When designing messaging solutions, these distinct configurations are represented as separate components that are situated in different logical tiers, as shown in the following figure where lines connecting components represent interactions.

Note – The following figure is not meant to be a complete logical architecture. A number of Java ES components are omitted to simplify the illustration.

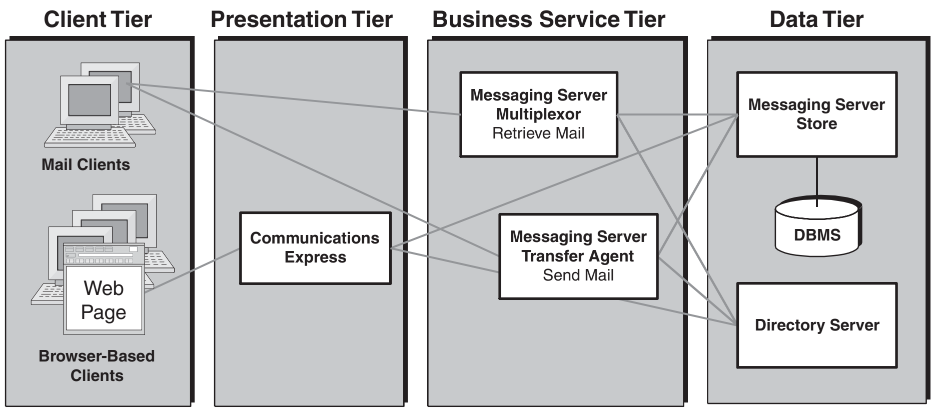

FIGURE 2–5 Messaging Server: Example of Tiered Architecture

Note – Communications components are not part of Java ES but are often deployed with Java ES components and used within Java ES architectures. These communications components are part of the Sun Java Communications Suite and are referenced in this document for illustrative purposes only.

The logical separation of Messaging Server functions into different tiers allows the logically distinct configurations of Messaging Server to be deployed on different computers in a physical environment. The physical separation allows for flexibility in meeting quality of service requirements (see "Dimension 3: Quality of Service" on page 40). For example, it provides for different availability solutions for the different instances, and different security implementations for different Messaging Server functions.

Dimension 3: Quality of Service

The previous two architectural dimensions (infrastructure service dependencies and logical tiers) mostly concern logical aspects of architecture, namely which components are needed to interact in what way to deliver services to end users. Another equally important dimension of any deployed solution is the ability of the solution to meet quality of service requirements.

The quality of service dimension of solution architecture highlights the roles played by Java ES service quality components.

Service Qualities

As Internet and e-commerce services have become more critical to business operations, the performance, availability, security, scalability, and serviceability of these services have become key quality of service requirements of large-scale, high-performance deployment architectures.

To design a successful software solution, you must establish relevant quality of service requirements and design an architecture that meets those requirements. A number of important service qualities are used to specify quality of service requirements. These service qualities are summarized in the following table.

TABLE 2–2 Service Qualities Impacting Solution Architecture

System Service Qualities	Description
Performance	The measurement of response time and latency with respect to user load conditions.
Availability	A measure of how often a system's resources and services are accessible to end users (the *uptime* of a system).
Security	A complex combination of factors that describes the integrity of a system and its users. Security includes physical security of systems, network security, application and data security (authentication and authorization of users), as well as the secure transport of information.
Scalability	The ability to add capacity to a deployed system over time. Scalability typically involves adding resources to the system but should not require changes to the deployment architecture.
Latent capacity	The ability of a system to handle unusual peak load usage without additional resources.
Serviceability	The ease by which a deployed system can be maintained, including monitoring the system, repairing problems that arise, and upgrading hardware and software components.

The quality of service dimension strongly impacts a solution's deployment architecture: how application components and infrastructure components are deployed in a physical environment.

The service qualities that affect deployment architecture are closely interrelated. Requirements for one system quality often affect the design for other service qualities. For example, higher levels of security might affect performance, which in turn might affect availability. Adding additional computers to address availability issues through redundancy often affects maintenance costs (serviceability).

Understanding how service qualities are interrelated and which trade-offs to make is key to designing deployment architectures that satisfy both business requirements and business constraints.

Java ES Service Quality Components

Several Java ES components are used principally to enhance the quality of services provided by system service components or distributed application components. These software components are often used in conjunction with hardware components such as load balancers and firewalls.

The Java ES service quality components, introduced in "Service Quality Components" on page 19, are summarized as follows:

- **Availability components.** Provide near-continuous uptime of a deployed solution.
- **Access components.** Provide secure Internet access to system services, and often provide a routing function as well.
- **Monitoring components.** Provide real-time information about Java ES components.

The following table lists the most important Java ES service quality components from an architectural perspective with the system qualities they impact most.

TABLE 2–3 Service Quality Components and the System Qualities Impacted

Component	System Qualities Impacted
High Availability Session Store	Availability
Monitoring Console	Serviceability
Portal Server Secure Remote Access	Security Scalability
Sun Cluster	Availability Scalability
Sun Cluster Geographic Edition	Availability Scalability
Web Proxy Server	Security Performance Scalability

Sun Cluster Software

Sun Cluster software provides high availability and scalability services for Java ES components and for applications supported by Java ES infrastructure. A cluster is a set of loosely coupled computers that collectively provides a single client view of services, system resources, and data. Internally, the cluster uses redundant computers, interconnects, data storage, and network interfaces to provide high availability to cluster-based services and data.

Sun Cluster software continuously monitors the health of member nodes and other cluster resources. In case of failure, Sun Cluster software intervenes to initiate failover of the resources it monitors and uses the internal redundancy to provide near-continuous access to these resources.

Sun Cluster data service packages (sometimes referred to as Sun Cluster agents) are available for all Java ES system service components. You can also write agents for custom-developed application components.

Because of the control afforded by Sun Cluster software, it can also provide for scalable services. By leveraging a cluster's global file system and the ability of multiple nodes in a cluster to run infrastructure or application services, increased demand on these services can be balanced among multiple concurrent instances of the services. When properly configured, Sun Cluster software can therefore provide for both high availability and scalability in a distributed enterprise application.

Because of the redundancy necessary to support Sun Cluster environments, inclusion of Sun Cluster in a solution substantially increases the number of computers and network links required in your physical environment.

Unlike the services provided by other Java ES components, Sun Cluster availability services are distributed peer-to-peer services. Sun Cluster software therefore needs to be installed on every computer in a cluster.

An extension to Sun Cluster software is provided by Sun Cluster Geographic Edition, which protects applications from unexpected disruptions using multiple clusters that are geographically separated and an infrastructure that replicates data between clusters.

Note – Sun Cluster and Sun Cluster Geographic Edition are supported on Solaris™ Operating System (Solaris OS) only.

Synthesis of the Three Architectural Dimensions

When viewed together, the three architectural dimensions shown in Figure 2–1 and discussed in the previous sections provide a framework for designing distributed software solutions. The three dimensions (infrastructure service dependencies, logical tiers, and quality of service) highlight the role played by Java ES components in solution architectures.

Each dimension represents a distinct architectural perspective. Every solution architecture must take them all into account. For example, distributed components in each logical tier of a solution architecture (dimension 2) need to be supported by appropriate infrastructure components (dimension 1) and appropriate service quality components (dimension 3).

Similarly, any component within a solution architecture plays different roles with respect to the different architectural dimensions. For example, Directory Server can be seen both as a back-end component in the data tier (dimension 2) and as a provider of persistence services (dimension 1). Because of the centrality of Directory Server with respect to these two dimensions, quality of service issues (dimension 3) are paramount for this Java ES component. A Directory Server failure would have a tremendous impact on a business system, so high availability design for this component is very important. Because Directory Server is used to store sensitive user or configuration information, security design for this component is also very important.

The interplay of the three dimensions with respect to Java ES components impacts the design of solution logical architectures and solution deployment architectures.

Detailed design methodologies based on the architectural framework represented by "Java ES Architectural Framework" on page 31 are not described in this book. However, the three-dimensional architecture framework highlights aspects of design that are important to understand when deploying software solutions based on Java Enterprise System.

Example Java ES Solution Architecture

Java ES supports a broad range of software solutions. Many solutions can be designed and deployed out-of-the-box, with no development effort, by using components included in Java ES. Other solutions might require extensive development efforts, requiring you to develop custom J2EE components that provide new business or presentation services. You might encapsulate these custom components as web services that conform to SOAP interface standards. Many solutions involve a combination of these two approaches.

This section provides an example that demonstrates how Java ES supports an out-of-the-box solution, drawing from the architectural concepts of the previous section.

Enterprise Communications Scenario

Businesses commonly need to support communication among their employees, specifically email and calendar services. Such businesses find it advantageous for their employees to have personalized access to internal web sites and other resources based on enterprise-wide authentication and authorization services. In addition, these businesses want employee identity to be tracked across all enterprise services, so that a single web sign-on provides access to all such services.

These specific business requirements, which represent only an example set of business requirements, are summarized in the following table.

TABLE 2–4 Business Requirements Summary: Communications Scenario

Business Requirement	Description	Services Needed
Single sign-on	Access to secure enterprise resources and services based on a single identity with single sign-on for web access	Identity services
Messaging Calendar	Email messaging between employees and with the outside world Electronic employee calendaring and meeting arrangements	Communication and collaboration services
Portal access	Single, web-based, personalized access points to communication services such as email and calendar as well as internal web pages	Portal services

In addition, an enterprise has requirements concerning the performance, availability, network security, and scalability of the software system that provides these services.

Logical Architecture for Example Scenario

A logical architecture for delivering the portal, communication, and identity services identified in Table 2–4 using Java ES components and Sun Java Communications Suite components (Messaging Server, Calendar Server, Instant Messaging, and so on) is shown in the following figure. The architecture treats logically distinct configurations of Messaging Server as separate components because of the distinct services they each provide.

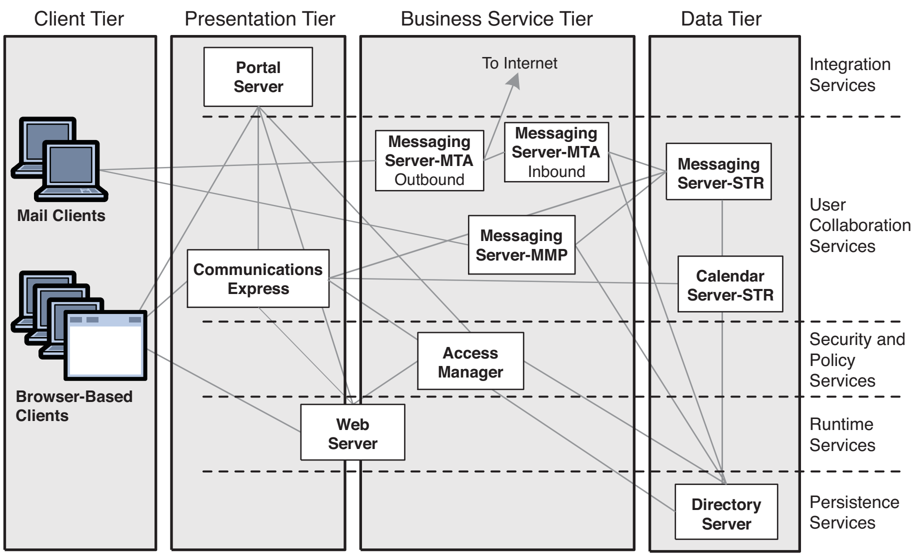

FIGURE 2–6 Logical Architecture for Enterprise Communications Scenario

The components are placed within a horizontal dimension that represents standard logical tiers and within a vertical dimension that represents infrastructure service levels. The interactions between the components depend on their functions as distributed infrastructure services (interactions between infrastructure service levels) or their roles within a tiered application architecture (interactions within and between logical tiers).

In this architecture, Access Manager, accessing user information stored in Directory Server, is the arbiter of single sign-on authentication and authorization for Portal Server and other web-based components in the presentation tier. Messaging Server components include a message store (Messaging Server-STR) in the data tier, sending and retrieving components in the business services tier, and an HTTP access component and Communications Express in the presentation tier.

The logical architecture also shows the infrastructure service dependencies between the various components. Portal Server, for example, depends on Communications Express for its messaging and calendar channels and on Access Manager for authentication and authorization services. These components, in turn, depend upon Directory Server for user information and configuration data. A number of components require web container services provided by Web Server.

For more information about Java ES solution logical design, see the *Sun Java Enterprise System Deployment Planning Guide*.

Deployment Architecture for Example Scenario

In moving from the logical architecture to a deployment architecture, quality of service requirements become paramount. For example, protected subnets and firewalls might be used to create a security barrier to back-end data. Availability and scalability requirements for many components might be met by deploying them on multiple computers and using load balancers to distribute requests among the replicated components.

However, where more demanding availability requirements apply and where large amounts of disk storage is involved, other availability solutions are more appropriate. For example, Sun Cluster can be used for the Messaging Server store and multimaster replication can be used for Directory Server.

For more information about Java ES solution deployment design, see the *Sun Java Enterprise System Deployment Planning Guide*.

Key Terms in This Chapter

This section explains key technical terms used in this chapter, with an emphasis on clarifying how these terms are used in the Java ES context.

application component	A custom-developed software component that performs a specific computing function, providing business services to end users or to other application components. An application component usually conforms to a distributed component model (such as CORBA and the J2EE platform). These components, singly or combined, can be encapsulated as web services.
architecture	A design that shows the logical and physical building blocks of a distributed application (or some other software system) and their relationships to one another. In the case of a distributed enterprise application, the architectural design generally includes both the application's logical architecture and deployment architecture.
business service	An application component or component assembly that performs business logic on behalf of multiple clients (and is therefore a multithreaded process). A business service can also be an assembly of distributed components encapsulated as a web service or a stand-alone server.
client	Software that requests software services. A client can be a service that requests another service or a GUI component accessed by an end user.
deployment architecture	A high-level design that depicts the mapping of a logical architecture to a physical computing environment. The physical environment includes the computers in an intranet or Internet environment, the network links between them, and any other physical devices needed to support the software.
logical architecture	A design that depicts the logical building blocks of a distributed application and the relationships (or interfaces) between these building blocks. The logical architecture includes both the distributed application components and the infrastructure services components needed to support them.

server	A multithreaded software process (as distinguished from a hardware server) that provides a distributed service or cohesive set of services for clients that access the service by way of an external interface.
web service	A service that conforms to standardized Internet protocols for accessibility, service encapsulation, and discovery. The standards include the SOAP messaging protocol, the WSDL (Web Services Description Language) interface definition, and the UDDI (Universal Description, Discovery and Integration) registry standard.

◆ ◆ ◆ CHAPTER 3

Java ES Integration Features

This chapter provides conceptual and technical background for understanding features that play key roles in integrating Java ES components into a single software system. These features help you understand some of the benefits of using Java ES, as compared to manually integrating disparate infrastructure products.

The chapter contains the following sections:

The Java ES Integrated Installer

All Java ES components are installed using a single installer. The Java ES installer transfers Java ES software to a host system. The installer lets you select and install any number of Java ES components on a host in your computing environment. The installer also provides for some installation-time configuration, depending on the particular Java ES components being installed.

The Java ES installer does not, in itself, perform distributed installations. To deploy a distributed Java ES software solution, you use the Java ES installer to install the appropriate components on each computer in your environment, one computer at a time. You must use a reasonable sequence of installation sessions and configuration procedures, based on your deployment architecture and component dependencies.

The installer runs interactively in both a graphical mode and a text-based mode, and also provides a parameter-driven silent installation mode. In addition to English, the installer supports the following languages: French, German, Japanese, Korean, Spanish, Simplified Chinese, and Traditional Chinese.

This section discusses aspects of the integrated Java ES installer. For more detailed information, see the *Sun Java Enterprise System 5 Installation Guide for UNIX*.

Preexisting Software Checking

The installer examines the host on which you are installing and identifies the Java ES components that are already installed. The installer then checks at several levels to ensure that all existing components are at the appropriate release level to interoperate successfully. The installer informs you about those software components that are incompatible and must be upgraded or removed.

Similarly, the installer checks for Java ES shared components such as J2SE or NSS that are already installed and lists any incompatibilities (see "Shared Components" on page 21). If you proceed with installation, the installer automatically upgrades the shared components to newer versions.

Dependency Checking

The installer does extensive checking of components to verify that the installation components you select function properly together. Many components have dependencies on other components. For this reason, when you select a component to install, the installer automatically includes the components and subcomponents upon which the selected component has dependencies. You cannot deselect a component if another selected component depends upon that component locally. However, if the dependency is not local, you receive a warning but can proceed under the assumption that the dependency is satisfied by a component on a different host computer.

Initial Configuration

Many Java ES components require initial configuration before they can be started. For some components, the Java ES installer is used to perform this initial configuration.

You can choose to have the installer perform this initial configuration (Configure Now option) or to install the software without performing initial configuration (Configure Later option), in which case you must explicitly configure each installed component after installation is complete.

If you choose to have the installer do the initial configuration, you supply the required configuration information during installation. In particular, you can specify a set of parameter values that are common across all component products, such as an administrator ID and password.

Uninstallation

Java ES also provides an uninstallation program to remove components that were installed on the local computer by the Java ES installer. The uninstaller checks for local dependencies and issues warnings when a dependency is found. The uninstaller does not remove Java ES shared components. As with the installer, the uninstaller can be run in graphical mode, text-based mode, or silent mode.

System Monitoring Services

Java ES includes a new monitoring feature that provides real-time monitoring of system services. Monitoring is implemented by the Sun Java System Monitoring Framework (shared component), and the Sun Java System Monitoring Console (product component). The Monitoring Framework is automatically configured and enabled to gather data for each installed Java ES component, and the Monitoring Console is the graphical interface used to view the monitored data. The Monitoring Console is a component that can be selected during the installation of Java ES, and the Monitoring Framework is automatically installed.

Monitoring is the process of gathering runtime data, exposing it, and computing quality of service criteria so that system administrators can assess performance and receive alarms. During runtime operation, administrators interact with the Monitoring Console to view performance statistics, set thresholds to monitor dynamically, define custom monitoring jobs, and acknowledge alarms.

Integrated Identity and Security Services

An important feature of Java ES is its integrated management of user identities and its integrated authentication and authorization framework. This section provides the technical background for understanding the integrated identity and security services provided by Java ES.

Single Identity

Within a Java ES environment, an end user has a single integrated identity. Based on this single identity, a user can be allowed access to various resources, such as a portal, web pages, and services such as messaging, calendar, and instant messaging.

This integrated identity and security capability is based on close collaboration between Directory Server, Access Manager, and other Java ES components.

User access to a Java ES service or resource is achieved by storing user-specific information in a single user entry in a user repository or directory. This information typically includes data such as a unique name and password, an email address, a role in an organization, web page preferences, and so forth. The information in the user entry can be used to authenticate the user, authorize access to specific resources, or provide various services to that user.

In the case of Java ES, user entries are stored in a directory provided by Directory Server. When a user wants to request a service provided by a Java ES component, that service uses Access Manager to authenticate the user and authorize access to specific resources. The requested service checks user-specific configuration information in the user's directory entry. The service uses that information to perform the work requested by the user.

The following figure illustrates access to user entries for performing user authentication and authorization and for providing services to a user.

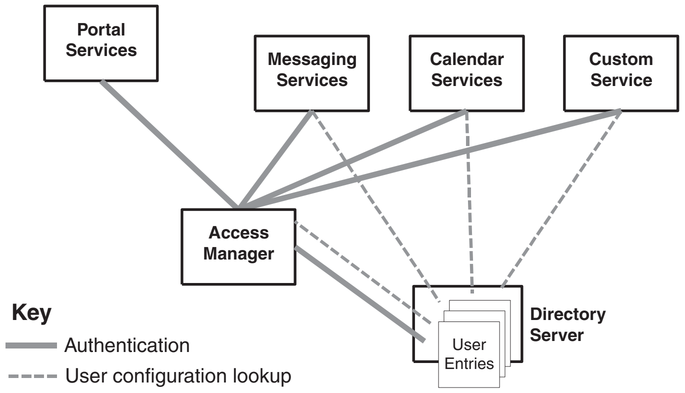

FIGURE 3–1 Single User Entry Supports Many Services

One of the features derived from this system is the ability of a web-based user to sign on to any Java ES service, and in so doing be automatically authenticated to other system services. This capability, known as single sign-on, is a powerful feature provided by Java ES.

Authentication and Single Sign-On

Access Manager provides Java ES authentication and authorization services. Access Manager uses information in Directory Server to broker the interaction of users with Java ES web services or other web-based services in an enterprise.

Access Manager uses an external component known as a policy agent. The policy agent plugs into the web server hosting a service or resource being secured by Access Manager. The policy agent intercedes on behalf of Access Manager in requests made by users to the secured resources. For some Java ES components such as Portal Server, the functionality of the policy agent is provided by the Access Manager SDK subcomponent.

Authentication

Access Manager includes an authentication service for verifying the identities of users who request access by way of HTTP or HTTPS to web services within an enterprise. For example, a company employee who needs to look up a colleague's phone number uses a browser to go to the company's online phone book. To log in to the phone book service, the user must provide a user ID and password.

The authentication sequence is shown in Figure 3–2. A policy agent intercedes in the request to log on to the phone book (1), and sends the request to the authentication service (2). The authentication service checks the user ID and password against information stored in Directory Server (3). If the login request is valid, the user is authenticated (4), (5), and (6), and the company phone book is displayed to the employee (7). If the login request is not valid, an error is generated and authentication fails.

The authentication service also supports certificate-based authentication over HTTPS.

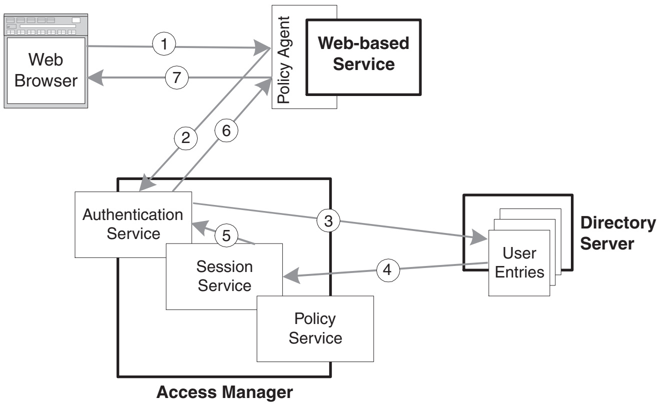

FIGURE 3–2 Authentication Sequence

Single Sign-On

The authentication scenario discussed in the previous paragraphs glosses over an important step. When a user's authentication request is verified, the Access Manager's session service is engaged (4), as shown in Figure 3–2. The session service generates a session token, which holds the user's identity information and a token ID (5). The session token is sent back to the policy agent (6), which forwards the token (as a cookie) to the browser (7) from which the authentication request was made.

When the authenticated user attempts to access another secured service, the browser passes the session token to the corresponding policy agent. The policy agent verifies with the session service that the user's previous authentication remains valid, and the user is granted access to the second service without being asked to reenter a user ID and password.

Accordingly, a user needs to sign on only once to be authenticated to multiple web-based services provided by Java ES. The single sign-on authentication remains in effect until the user explicitly signs off or the session expires.

Authorization

Access Manager also includes a policy service that provides access control to web-based resources in a Java ES environment. A policy is a rule that describes who is authorized to access a specific resource under specific conditions. The authorization sequence is shown in the following figure.

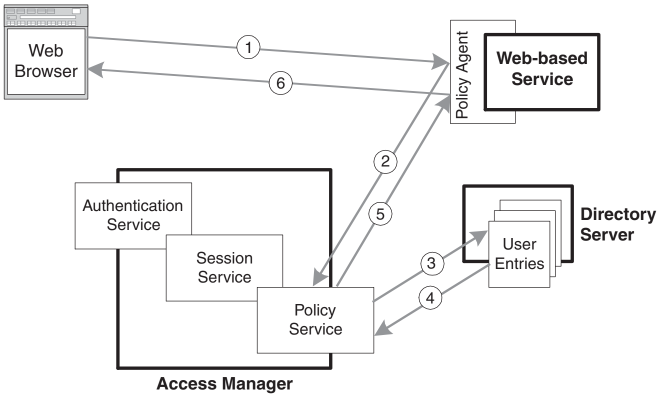

FIGURE 3–3 Authorization Sequence

When an authenticated user makes a request for any resource secured with Access Manager (1), the policy agent notifies the policy service (2), which uses information in Directory Server (3) to evaluate the access policy governing the resource to see if the user has permission to access the resource (4). If the user has access privileges (5), then the resource request is fulfilled (6).

Access Manager provides the means for defining, modifying, granting, revoking, and deleting policies within an enterprise. The policies are stored in Directory Server and configured through policy-related attributes in organization entries. Roles can also be defined for users and incorporated in policy definitions.

Access Manager policy agents are the policy enforcers. When the policy service rejects an access request, the policy agent denies the requesting user access to the secured resources.

Key Terms in This Chapter

This section explains key technical terms used in this chapter, with an emphasis on clarifying how these terms are used in the Java ES context.

directory	A special kind of database optimized for reading data rather than writing data. Most directories are based on LDAP (Lightweight Directory Access Protocol), an industry-standard protocol.
policy	A rule that describes who is authorized to access a specific resource under specific conditions. The rule can be based on groups of users or roles in an organization.
single identity	An identity that a user has by virtue of a single user entry in a Java ES directory. Based on this single user entry a user can be allowed access to various Java ES resources such as a portal and web pages and services such as messaging, calendar, and instant messaging.
single sign-on	A feature that allows a user's authentication to one service in a distributed system to be automatically applied to other services in the system.

CHAPTER 4

Java ES Solution Life Cycle

The chapter discusses concepts and terminology relevant to each phase of the Java ES solution life cycle. The focus of this chapter is on deployment tasks, particularly deployment design and deployment implementation tasks.

This chapter describes the tasks involved in each phase of the life cycle. The chapter contains the following sections:

- "Predeployment" on page 59
- "Deployment" on page 60
- "Postdeployment" on page 64
- "Key Terms in This Chapter" on page 65

Solution Life Cycle Tasks

The solution life cycle was introduced in Chapter 1, "Introduction to Java Enterprise System," as a standard approach to implementing business solutions using Java ES software. The life cycle diagram is repeated for easy reference.

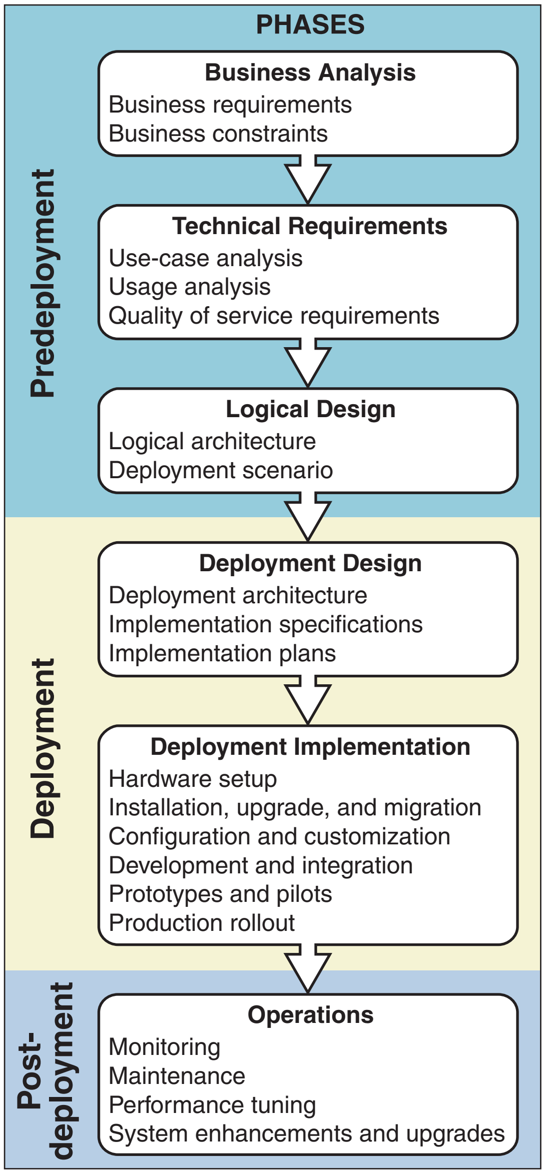

FIGURE 4–1 Solution Life Cycle Tasks

Predeployment

In the predeployment phases of the life cycle, you translate an analysis of business needs into a deployment scenario. The deployment scenario serves as a specification for a deployment design.

Predeployment tasks are grouped into three phases, as shown in Figure 4–1:

- **Business analysis.** Define the business goals of a proposed deployment effort and state the business requirements and constraints that must be met to achieve that goal.
- **Technical requirements.** Use the results of the business analysis to create use cases that model user interaction with an anticipated software system. You also determine the usage patterns expected for those use cases. Using both the business analysis and the usage analysis, you formulate quality of service requirements (see Table 2–2) that the proposed deployment must meet.
- **Logical design.** Analyze the use cases developed in the technical requirements phase to determine which Java ES infrastructure components and which custom-developed application components are needed to provide end-user services. Using the concepts discussed in Chapter 2, "Java ES Solution Architectures," you design a logical architecture. The logical architecture shows all components and all interactions between components that are needed to effect the use cases of a particular software solution.

The logical architecture combined with performance, availability, security, and other quality of service requirements is encapsulated in a deployment scenario, as shown in the following figure. For more information about predeployment phases of the life cycle, see the *Sun Java Enterprise System Deployment Planning Guide*.

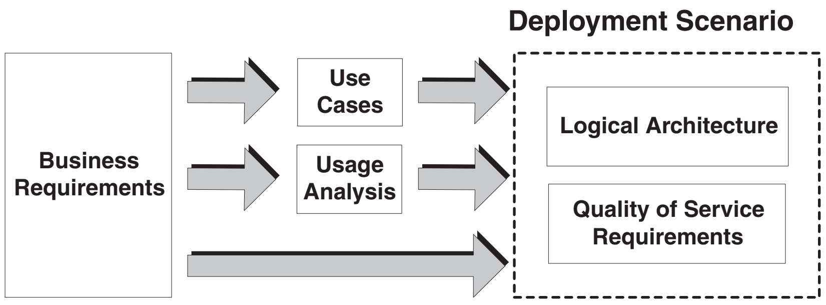

FIGURE 4–2 Specifying a Deployment Scenario

Deployment

In the deployment phases of the life cycle, you translate a deployment scenario into a deployment design that you then implement, test, and roll out in a production environment.

The deployment process generally encompasses software components in all tiers and all infrastructure service levels required to support a software solution. In general, you must deploy both custom-developed application components (J2EE components, web services, or other servers) and the Java ES components needed to support the solution.

Deployment tasks are grouped into two phases, as shown in Figure 4–1:

- "Deployment Design" on page 60. Deployment design depends on a solution's logical architecture and on the performance, availability, security, scalability, serviceability, and other quality of service requirements a solution must meet. The quality of service dimension of deployment architecture plays a big role in the deployment design phase.
- "Deployment Implementation" on page 62. Implementation of a deployment design is an iterative process that involves hardware setup, software installation and configuration, development and integration, testing, and other aspects of production rollout.

  The following sections examine these two phases of the deployment process.

Deployment Design

In the deployment design phase, you create a high-level deployment architecture followed by low-level implementation specifications.

Deployment Architecture

A deployment architecture is created by mapping the logical building blocks of an application (the logical architecture) to a physical computing environment in a way that meets the quality of service requirements specified in the deployment scenario. The deployment scenario is translated into a deployment architecture, as shown in the following figure.

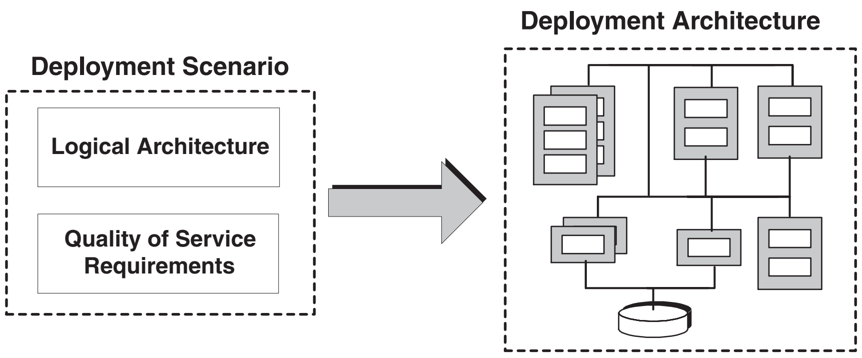

FIGURE 4–3 Translating a Deployment Scenario into a Deployment Architecture

One aspect of this architectural design is sizing the physical environment (determining the number of computers and estimating their processor power and RAM requirements) to meet performance, availability, security, and other quality of service requirements. Once the sizing is complete, you map Java ES components and application components to the various computers in the physical environment. The resulting deployment architecture must take into account the capabilities of different computers, characteristics of system infrastructure services, and restrictions on the total cost of ownership or total cost of availability.

The larger the number of Java ES components in the deployment scenario and the more demanding your quality of service requirements, the more demanding your design is on high-power computers and high network bandwidth. Where hardware is limited or prohibitively expensive, you might need to assess trade-offs between fixed costs (hardware) and variable costs (human resource requirements) or between different quality of service requirements, or you might have to increase the sophistication of your design.

The design of a deployment architecture often evolves in an iterative fashion. Reference deployment architectures serve as a starting point Java ES deployment design.

A reference architecture is based on a specific deployment scenario: a logical architecture with specific quality of service requirements. In the reference architecture, a software solution is deployed across a specific physical environment in a way that meets specified quality of service requirements. Performance testing at specified loads is based on the same set of use cases from which the deployment scenario was developed. Reference architecture documentation is available to Java ES customers under non-disclosure.

Based on a reference deployment architecture or a combination of reference architectures, you can design a first approximation of a deployment architecture that meets your own deployment scenario requirements. You can adjust the reference architectures or use them as reference points, taking into account the difference between your own deployment scenario and those upon which the reference architectures are based. In this way, you can assess the impact of your own sizing, performance, security, availability, capacity, and serviceability needs.

Implementation Specifications

Implementation specifications provide details needed to implement a deployment architecture. The specifications generally include the following information:

- Actual hardware, including computers, storage devices, load balancers, and network cabling
- Operating systems
- Network design, including subnets and security zones
- Availability design details
- Security design details
- Directory design information needed for provisioning end users

Implementation Plans

Implementation plans describe how you plan to perform the various tasks in the deployment implementation phase. The plans generally cover the following tasks:

- Hardware setup
- Software installation, upgrade, and migration
- System configuration and customization
- Development and integration
- Testing
- Production rollout

Deployment Implementation

Implementation of a deployment design consists of the tasks listed in the previous section and shown in Figure 4–1. The ordering of these tasks is not rigid because the deployment process is iterative in nature. The following subsections discuss each of the major deployment implementation tasks in the order in which they are typically performed.

Hardware Setup

The implementation specification includes all details of your physical environment: the computers, network design, network hardware (including cabling, switches, routers, and load balancers), storage devices, and so forth. All of this hardware needs to be set up as the platform that supports your Java ES solution.

Software Installation, Upgrade, and Migration

The deployment architecture, along with additional details provided in implementation specifications, tells you which application components and which Java ES components are to reside on each computer in your physical environment. You use the Java ES integrated installer to install the appropriate Java ES components on each computer in your deployment architecture (see "The Java ES Integrated Installer" on page 49).

Your installation plan describes the sequence and scope of installer sessions. The approach you take to performing installation, however, might depend on whether you are performing a new installation of Java ES, whether you are upgrading previously installed Java ES components, or whether you are replacing third-party components with Java ES. The last two of these Java ES adoption scenarios often require you to migrate data or application code to achieve compatibility.

System Configuration and Customization

You must complete a number of system configuration tasks for the various system components to work together as an integrated system. First among these tasks is the initial configuration needed for each individual system component to start. Secondly, each Java ES component must be configured to communicate with those components with which it interacts.

High availability must also be configured, depending on your availability solution for each component. Users must be provisioned so they can access various services, and authentication and authorization policies and controls must be set up (see "Integrated Identity and Security Services" on page 51).

In most cases configuration tasks include some degree of customization of Java ES components to achieve the exact feature set you require. For example, you typically customize Portal Server to provide portal channels, Access Manager to perform authorization tasks, and so forth.

Development and Integration

The logical architecture specified in the deployment scenario generally determines the scope of custom development work needed to implement a solution.

For some deployments, development might be quite extensive, requiring you to develop new business and presentation services from the beginning using J2EE components that run in an Application Server or Web Server environment. In such cases, you create a prototype of your solution and perform proof-of-concept testing before embarking on a full development effort.

For solutions that require extensive development, Sun Java™ Studio software provides tools for programming distributed components or business services. Sun Java Studio developer tools simplify the programming and testing of applications supported by the Java ES infrastructure.

In some situations, Java ES components might be integrated with legacy applications or third-party services. These integrations might involve existing directories or data services in the data tier or existing components in the business services tier. Integrating Java ES components with these systems might require migrating data or application code.

The J2EE platform provides a connector framework that enables you to plug existing applications into the Application Server environment by developing J2EE resource adapters, and Message Queue provides a robust asynchronous messaging capability for integrating diverse applications.

Testing of Prototypes and Pilots

Depending on the amount of customization or development work required, at some point you must verify your deployment architecture: you must test the solution against the use cases to verify that you can meet quality of service requirements.

If you have relatively few custom-developed services (a mostly out-of-the-box deployment), your solution might simply require customization of Java ES components and a pilot test of the system.

However, if you have developed significant new application logic and created custom services, this testing might be more extensive, involving prototype testing, integration testing, and so forth.

If this testing reveals shortcomings in your deployment architecture, you modify the architecture and test again. This iterative process should eventually result in a deployment architecture and implementation that is ready for deployment in a production environment.

Production Rollout

Production rollout involves building out your deployment implementation in a production environment. This phase involves installing, configuring, and starting distributed applications and infrastructure services in a production environment, provisioning production system end users, setting up single sign-on and access policies, and so forth. You typically start with a limited deployment and move to organization-wide implementation. In this process, you perform trial runs in which you apply increasing loads to confirm that quality of service requirements are being met.

Postdeployment

In the postdeployment stage of the life cycle, you run a deployed solution in a production environment. The operations phase of the life cycle involves the following tasks:

- **Monitoring.** Includes regular monitoring of system performance and system functions.
- **Maintenance.** Includes everyday administrative functions such as adding new end users to a system, changing passwords, adding new administrative users, changing access privileges, performing regular backups, and so forth.
- **Performance tuning.** Includes the use of regular monitoring information to find bottlenecks in system operations and attempting to eliminate those bottlenecks by changing configuration properties, adding capacity, and so forth.
- **System enhancements and upgrades.** Includes adding new Java ES components to a system to add new functionality or to replace non-Java ES components. These changes might require a redesign of the system, beginning with the initial phases of the solution life cycle. Upgrade tasks are more limited, usually amounting to upgrades of Java ES components.

Key Terms in This Chapter

This section explains key technical terms used in this chapter, with an emphasis on clarifying how these terms are used in the Java ES context.

deployment	A stage of the Java ES solution life cycle in which a deployment scenario is translated into a deployment design, implemented, prototyped, and rolled out in a production environment. The end product of this process is also referred to as a deployment (or deployed solution).
deployment scenario	A logical architecture for a Java ES solution and the quality of service requirements the solution must satisfy to meet business needs. The quality of service requirements include requirements regarding performance, availability, security, serviceability, and scalability or latent capacity. A deployment scenario is the starting point for deployment design.
development	A task in the Java ES solution deployment process by which the custom components of a deployment architecture are programmed and tested.
predeployment	A stage of the Java ES solution life cycle process in which business needs are translated into a deployment scenario: a logical architecture and a set of quality of service requirements that a solution must meet.
postdeployment	A stage of the Java ES solution life cycle process in which distributed applications are started up, monitored, tuned to optimize performance, and dynamically upgraded to include new functionality.
reference deployment architecture	A deployment architecture that has been designed, implemented, and tested for performance. Reference deployment architectures are used as starting points for designing deployment architectures for custom solutions.
use case	A specific end-user task or set of tasks performed by a distributed enterprise application and used as a basis for designing, testing, and measuring the performance of the application.

APPENDIX A

Java ES Components

Java ES consists of a collection of product components and shared components that work together to support distributed applications across a network. During installation the Java ES installer presents selectable components, many of which have selectable subcomponents. These components and subcomponents are listed in this appendix.

This appendix provides brief descriptions of Java ES components and is intended as an overview. For detailed information about specific components, see the component documentation sets available at `http://docs.sun.com/app/docs/prod/entsys.5`. A wide range of Java ES information and resources is also available at `http://www.sun.com/bigadmin/hubs/javaes/`.

The Java ES components listed in this appendix are grouped by category and described in the following sections:

- "System Service Components" on page 67
- "Service Quality Components" on page 71
- "Shared Components" on page 75

System Service Components

Java ES system service components provide the infrastructure services needed to support distributed enterprise applications. These services, as described in "Why You Need Java ES" on page 15, include portal services, identity and security services, web and application services, and availability services. Java ES system service components are described in the following sections:

- "Access Manager 7.1" on page 68
- "Application Server Enterprise Edition 8.2" on page 68
- "Directory Server Enterprise Edition 6.0 " on page 69
- "Java DB 10.1" on page 70
- "Message Queue 3.7 UR 1" on page 70
- "Portal Server 7.1" on page 70

Access Manager 7.1

Sun Java System Access Manager (Access Manager) integrates authentication and authorization services, policy agents, and identity federation to provide a comprehensive solution for protecting network resources. Access Manager prevents unauthorized access to web service applications and web content, providing an infrastructure for organizations to manage the digital identities of customers, employees, and partners who use their web-based services and non-web applications. Because these resources might be distributed across a range of internal and external computing networks, the attributes, policies, and entitlements are defined and applied to each identity to manage access to these technologies.

Access Manager includes the following subcomponents:

- **Access Manager Core Services.** Provides the means for creating and managing user identities and for defining and evaluating policies that provide access to Java ES resources based on user identities.
- **Access Manager Administration Console.** Consolidates identity services and policy management and provides a single graphical interface for users to create and manage user accounts, service attributes, and access rules in the Directory Server.
- **Common Domain Services for Federation Management.** Enables users to use a single identity to access applications offered by multiple affiliated service providers.
- **Access Manager SDK.** Provides a remote interface to Access Manager. This subcomponent must be installed on any computer hosting a Java ES component that accesses Access Manager remotely.
- **Access Manager Distributed Authentication User Interface.** Provides a user interface that enables a policy agent or an application that is deployed in an unsecured area to communicate with the Access Manager Authentication Service that is installed in a secured area of the deployment.
- **Access Manager Client SDK.** Enables users to implement stand-alone applications that can access an Access Manager server to use services such as authentication, single sign-on, authorization, auditing, logging, and Security Assertion Markup Language (SAML).
- **Access Manager Session Failover Client.** Required to configure Access Manager session failover.

Application Server Enterprise Edition 8.2

Sun Java System Application Server (Application Server) provides a J2EE compatible platform for developing and delivering server-side Java applications and web services. Key features include scalable transaction management, container-managed persistence runtime, web services performance, clustering, high availability session state, security, and integration capabilities.

Application Server includes the following subcomponents:

- **Domain Administration Server.** Provides server-side administration functions, such as managing and configuring Application Server and deploying J2EE components and applications.
- **Application Server Node Agent.** A lightweight process that runs on every machine that hosts server instances and performs a number of administrative tasks, including stopping, starting, and restarting server instances.
- **Command Line Administration Tool.** Provides command-line administration clients that enable you to manage and configure Application Server installations and hosted applications. The tool also assists with deploying applications.
- **Load Balancing Plug-in.** Used to evenly distribute the workload among multiple Application Server instances (either stand-alone or clustered), thereby increasing the overall throughput of the system. Also used to enable requests to fail over from one server instance to another.
- **Sample Applications.** Installed with the full installation of Application Server.

Directory Server Enterprise Edition 6.0

Sun Java System Directory Server (Directory Server) is an LDAP-based directory server that provides a centralized directory service for your intranet, network, and extranet information. Directory Server integrates with existing systems and acts as a centralized repository for the consolidation of employee, customer, supplier, and partner information. You can extend Directory Server to manage user profiles and preferences and extranet user authentication.

Directory Server includes the following subcomponents:

- **Directory Server 6 Core Server.** Provides the scalable, secure, and flexible means to store and manage identity data.
- **Directory Service Control Center.** Provides a browser-based administration interface to configure directory and directory proxy services.
- **Directory Server Command-Line Utility.** Enables you to perform administration tasks from the command line.
- **Directory Proxy Server 6 Core Server.** Enhances security by offering virtual directory capabilities and increasing directory service availability and scalability.

Java DB 10.1

Java DB provides a lightweight database for Java application development. Java DB is Sun's supported distribution of the open source Apache Derby 100% Java technology database. Java ES 5 is the first release to include Java DB as a product component. Java DB was first released as a shared component named Derby Database and was included Java ES 2005Q4.

Java DB includes the following subcomponents:

- Java DB Client
- Java DB Server

Message Queue 3.7 UR 1

Sun Java System Message Queue (Message Queue) is a standards-based solution to the problem of inter-application communication and reliable message delivery. Message Queue is an enterprise messaging system that implements the Java Message Service (JMS) open standard.

In addition to being a JMS provider, Message Queue has features that exceed the minimum requirements of the JMS specification. With Message Queue software, processes running on different platforms and operating systems can connect to a common Message Queue service to send and receive information. Application developers can focus on the business logic of their applications rather than on the low-level details of how their applications communicate across a network.

The Java ES installer provides Message Queue as a single installable component.

Portal Server 7.1

Sun Java System Portal Server (Portal Server) is an identity-enabled portal server solution. Portal Server combines portal services such as personalization, aggregation, security, integration, and search.

The Java ES installer provides Portal Server as a single installable component.

Service Registry 3.1

Sun Java System Service Registry (Service Registry) is a repository that serves as both a web services (UDDI) registry and an enterprise business XML (ebXML) registry for supporting web service-oriented architecture (SOA) applications. The UDDI registry is used to register and discover web services, and the ebXML registry is used to store and manage the information artifacts needed to support business process integration. These artifacts include metadata such as XML schema, business process rules, web service access controls, version controls, classifications schemes, and so forth.

Service Registry includes the following subcomponents:

- Service Registry Client Support
- Service Registry Deployment Support

Web Server 7.0

Sun Java System Web Server (Web Server) is a multiprocess, multithreaded, secure web server built on industry standards. Web Server provides high performance, reliability, scalability, and manageability for medium to large enterprises.

Web Server includes the following subcomponents:

- Web Server CLI
- Web Server Core
- Web Server Samples

Service Quality Components

Java ES service quality components enhance the quality of services provided by system service components or distributed application components. Some are availability components that are used to provide for near-continuous system uptime, some are access components that are used to support secure end-user access to system services, and others are system management components that are used to enhance the serviceability of Java ES solutions.

The components that support Java ES services components are grouped into the following categories and described in this section:

- "Availability Components" on page 71
- "Access Components" on page 73
- "Monitoring Components" on page 74

Availability Components

Availability components provide for near-continuous uptime for system service components and application components. The following Java ES availability components are described in this section:

- "High Availability Session Store 4.4.3" on page 72
- "Sun Cluster 3.1 8/05 and Sun Cluster Agents 3.1" on page 72
- "Sun Cluster Geographic Edition 3.1 2006Q4" on page 73

High Availability Session Store 4.4.3

Sun Java System High Availability Session Store (HADB) provides a data store that can be used to make application data available even in the case of failure. This capability is especially important for restoring state information associated with a client session. Without this capability, failure during a session requires that all operations be repeated when the session is reestablished.

The following Java ES components provide services that store session state information: Application Server, Access Manager, and Message Queue. However, Application Server is the only component that can use HADB services to maintain session state during failure.

The Java ES installer provides HADB as a single installable component. However, a server and a client subcomponent are both required to provide HADB services.

Sun Cluster 3.1 8/05 and Sun Cluster Agents 3.1

Note – Sun Cluster components are supported on the Solaris platform only.

Sun Cluster software provides high availability and scalability services for Java ES and for applications based on the Java ES infrastructure.

A cluster is a set of loosely coupled computers (cluster nodes) that collectively provide a single client view of services, system resources, and data. Internally, the cluster uses redundant computers, interconnects, data storage, and network interfaces to provide high availability to cluster-based services and data. Sun Cluster software continuously monitors the health of member nodes and other cluster resources and uses the internal redundancy to provide near-continuous access to these resources even when failure occurs.

The Java ES installer provides the Sun Cluster Core subcomponent and the Sun Cluster Agents as separately installable components. The following Sun Cluster agents are included in Java Enterprise System.

Note – *HA* in the following list stands for *high availability*.

- HA Application Server
- HA Message Queue
- HA Directory Server
- HA Messaging Server
- HA Application Server EE (HADB)
- HA/Scalable Web Server
- HA Instant Messaging
- HA Calendar Server
- HA Apache Tomcat

- HA Apache
- HA DHCP
- HA DNS
- HA MySQL
- HA Sun N1 Service Provisioning
- HA NFS
- HA Oracle
- HA Samba
- HA Sun N1 Grid Engine
- HA Solaris Containers

Note – The list of agents is not the same on SPARC and x86. For detailed information about Sun Cluster Agents, see Sun Cluster documentation at `http://docs.sun.com/app/docs/prod/entsys.5`.

Sun Cluster Geographic Edition 3.1 2006Q4

Sun Cluster Geographic Edition is a layered extension of Sun Cluster software. This extension protects applications from unexpected disruptions by using multiple clusters that are geographically separated and a redundant infrastructure that replicates data between these clusters. Java ES 5 is the first release to include Sun Cluster Geographic Edition as a Java ES product component.

Sun Cluster Geographic Edition includes the following subcomponents:

- Sun Cluster Geographic Edition Core
- Sun StorEdge Availability Suite
- Hitachi Truecopy Data Replication Support (SPARC only)
- EMC SRDF Data Replication

Note – Sun Cluster Geographic Edition is not supported on Solaris x86.

Access Components

Access components provide front-end access to system services, often from Internet locations outside an enterprise firewall. The following Java ES access components are described in this section:

- "Portal Server Secure Remote Access 7.1" on page 74
- "Web Proxy Server 4.0.4" on page 74

Portal Server Secure Remote Access 7.1

Sun Java System Portal Server Secure Remote Access (Portal Server Secure Remote Access) extends Portal Server by offering browser-based secure remote access to Portal Server content and services from any remote browser, eliminating the need for client software. Integration with Portal Server ensures that users receive secure access to the content and services that they have permission to access.

Portal Server Secure Remote Access includes the following subcomponents:

- **Portal Server Secure Remote Access Core.** Provides core functionality.
- **Gateway.** Provides the interface and security barrier between remote user sessions originating from the Internet and a corporate intranet. Gateway presents content securely from internal web servers and application servers through a single interface to a remote user and controls communication between the Portal Server and the various Gateway instances.
- **Netlet Proxy.** Enables users to securely run common TCP/IP services over the Internet and other non-secure networks. Netlet allows you to run applications such as telnet, SMTP, HTTP, and fixed-port applications. Netlet enables remote access and operation of file systems and directories and ensures secure communication between the Netlet applet on the client browser, the Gateway, and the application servers.
- **Rewriter Proxy.** Enables secure HTTP traffic between the Gateway and intranet computers. Rewriter provides secure access to corporate intranet web pages from outside of the intranet by transforming web links and creating rule sets for handling intranet web pages.

Web Proxy Server 4.0.4

Sun Java System Web Proxy Server (Web Proxy Server) provides caching, filtering, and distribution of web content. Web Proxy Server is often used inside enterprise firewalls to reduce the number of requests to remote content servers, and outside firewalls to provide a secure gateway for incoming Internet requests.

The Java ES installer provides Web Proxy Server as a single installable component.

Monitoring Components

The Sun Java System Monitoring Console 1.0 (Monitoring Console) includes a master agent that connects to all node agents in a Java ES deployment. The Monitoring Console is supported by the Sun Java System Monitoring Framework 2.0 (Monitoring Framework), a shared component that provides the instrumentation and node agent needed by every monitored component to expose its attributes for observation. Each product component exposes the objects that represent its observable attributes, and a node agent aggregates a view of multiple components on one host. For detailed information about monitoring, see the *Sun Java Enterprise System 5 Monitoring Guide*.

Shared Components

Shared components provide local services and technology support upon which Java ES system service components and service quality components depend. These components are local libraries that can be shared by any Java ES component running on a particular host computer. The Java ES installer automatically installs any shared components required to support other Java ES components installed on a host computer.

Java ES includes the following shared components:

- ACL (Apache Common Logging) 1.0.4
- ANT (Jakarta ANT Java/XML-based build tool) 1.6.5
- BDB (Berkeley Database) 4.2.52
- Common Agent Container 1.1 (Sun Cluster only)
- Common Agent Container 2.0
- FastInfoSet 1.0.2
- ICU 3 (International Components for Unicode) 3.2
- J2SE (Java 2 Platform, Standard Edition) 5.0 Update 6 (version 5.0 Update 3 is supported for HP-UX)
- JAF (JavaBeans™ Activation Framework) 1.0.3
- JATO (Java Studio Web Application Framework) 2.1.5
- JavaHelp™ 2.0
- JavaMail™ API 1.3.2
- JAXB (Java Architecture for XML Binding) 2.0.3
- JAXP (Java API for XML Processing) 1.3.1
- JAXR (Java API for XML Registries) 1.0.8
- JAXRPC (Java API for XML-based Remote Procedure Call) 1.1.3_01
- JAXWS (Java API for Web Services) 2.0
- JDMK (Java Dynamic Management Kit) 5.1.2
- JSS (Java Security Services) 4.2.4
- JSS3 (Network Security Services for Java) 3.1.11
- JSTL (JavaServer Pages™ Standard Tag Library) 1.0.6
- KTSE (KT Search Engine) 1.3.4
- LDAP C SDK 6.0
- LDAP Java SDK 4.19
- MA Core (Mobile Access Core) 6.3.1
- NSPR (Netscape Portable Runtime) 4.6.4

- NSS (Network Security Services) 3.11.4
- NSSU (Network Security Service Utilities) 3.11
- SAAJ (SOAP with Attachments API for Java) 1.3
- SASL (Simple Authentication and Security Layer) 2.19
- Sun Explorer Data Collector (Solaris OS only) 4.3.1
- Sun Java System Monitoring Framework 2.0 (supports the Monitoring Console 1.0)
- Sun Java Web Console 3.0.2
- WSCL (Web Services Common Library) 2.0
- XWSS (XML Web Services Security) 2.0

Index