Sun Java™ System

Sun Java Enterprise System 2005Q1 Technical Overview

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

Part No: 819-0061-10

Contents

List of Tables **7**

List of Figures **9**

Preface **11**
Audience 12
Using the Documentation 12
Conventions 14
Resources on the Web 14
How to Report Problems 15
Sun Welcomes Your Comments 15

Chapter 1 Introduction to Java Enterprise System **17**
Why Do You Need Java Enterprise System? 18
Java Enterprise System Components 20
  System Service Components 21
  Service Quality Components 22
    Availability Components 22
    Access Components 23
    Administrative Components 24
  Shared Components 24
Working With Java Enterprise System 25
  Java Enterprise System Solution Life Cycle 25
  Java Enterprise System Adoption Scenarios 27
Key Terms in This Chapter 29

Chapter 2 Java Enterprise System Solution Architectures 31
Java Enterprise System Architectural Framework 32
Dimension 1: Infrastructure Service Dependencies 33
Infrastructure Service Levels 33
Java Enterprise System Infrastructure Service Components 36
Java Enterprise System Infrastructure Service Dependencies 37
Dimension 2: Logical Tiers 38
Description of Logical Tiers 39
Logical and Physical Independence 40
Tiered Architecture Applied to System Components 40
Dimension 3: Quality of Service 41
Service Qualities 42
Java Enterprise System Service Quality Components 43
Sun Cluster Software 44
Synthesis of the Three Architectural Dimensions 45
Example Java Enterprise System Solution Architecture 46
Enterprise Communications Scenario 46
Logical Architecture for Example Scenario 47
Deployment Architecture for Example Scenario 48
Key Terms in This Chapter 49

Chapter 3 Java Enterprise System Integration Features 51
The Java Enterprise System Integrated Installer 52
Preexisting Software Checking 52
Dependency Checking 53
Initial Configuration 53
Uninstallation 54
Integrated Identity and Security Services 54
Single Identity 54
Authentication and Single Sign-On 55
Authentication 56
Single Sign-On 57
Authorization 57
Key Terms in This Chapter 58

Chapter 4 Java Enterprise System Solution Life-Cycle Tasks 59
Predeployment 61
Deployment 62
Deployment Design 62
Deployment Architecture 62
Implementation Specifications 64
Implementation Plans 64

Deployment Implementation 65
Hardware Setup 65
Software Installation, Upgrade, and Migration 65
System Configuration and Customization 65
Development and Integration 66
Testing of Prototypes and Pilots 66
Production Rollout 67
Postdeployment 67
Key Terms in This Chapter 68

Appendix A Reference Listing: Java Enterprise System Components 71
System Service Component Descriptions 72
Sun Java System Access Manager 6 2005Q1 72
Sun Java System Application Server Enterprise Edition 8 2005Q1 73
Sun Java System Calendar Server 6 2005Q1 74
Sun Java System Directory Server 5 2005Q1 74
Sun Java System Instant Messaging 7 2005Q1 74
Sun Java System Message Queue 3 2005Q1 75
Sun Java System Messaging Server 6 2005Q1 75
Sun Java System Portal Server 6 2005Q1 76
Sun Java System Web Server 6.1 2005Q1 76
Service Quality Component Descriptions 77
Availability Component Descriptions 77
Sun Cluster 3.1 9/04 and Sun Cluster Agents 77
High Availability Session Store 2005Q1 78
Access Component Descriptions 78
Sun Java System Communications Express 2005Q1 78
Sun Java System Connector for Microsoft Outlook 6 2005Q1 79
Sun Java System Directory Proxy Server 5 2005Q1 79
Sun Java System Portal Server Secure Remote Access 6 2005Q1 79
Administrative Component Descriptions 80
Sun Java System Administration Server (and Console) 5 2005Q1 80
Sun Java System Directory Preparation Tool 2005Q1 80
Sun Java System Delegated Administrator 6 2005Q1 81
Sun Remote Services Net Connect 2005Q1 81
Shared Components 81

Index 83

List of Tables

Table 1 Java Enterprise System Documentation 12
Table 2 Typeface Conventions 14
Table 1-1 Java ES System Service Components 21
Table 1-2 Java ES Availability Components 23
Table 1-3 Java ES Access Components 23
Table 1-4 Java ES Administrative Components 24
Table 1-5 Java ES User Categories for Life-Cycle Tasks 27
Table 1-6 Concerns Related to Different Java ES Adoption Scenarios 29
Table 2-1 Relationships Between Java ES System Service Components 37
Table 2-2 Service Qualities Impacting Solution Architecture 42
Table 2-3 Service Quality Components and the System Qualities Impacted 43
Table 2-4 Business Requirements Summary: Communications Scenario 46

List of Figures

Figure 1-1 Support Needed for Distributed Enterprise Applications 18
Figure 1-2 Categories of Java Enterprise System Components 20
Figure 1-3 Solution Life-Cycle Phases and User Categories 26
Figure 2-1 Dimensions of Java ES Solution Architecture 32
Figure 2-2 Dimension 1: Infrastructure Service Levels 34
Figure 2-3 Java ES System Service Components 36
Figure 2-4 Dimension 2: Logical Tiers for Distributed Enterprise Applications 38
Figure 2-5 Messaging Server: Example of Tiered Architecture 41
Figure 2-6 Availability Design Using Sun Cluster Nodes 44
Figure 2-7 Logical Architecture for Enterprise Communications Scenario 47
Figure 3-1 Single User Entry Supports Many Services 55
Figure 3-2 Authentication Sequence 56
Figure 3-3 Authorization Sequence 57
Figure 4-1 Solution Life-Cycle Tasks 60
Figure 4-2 Specifying a Deployment Scenario 61
Figure 4-3 Translating a Deployment Scenario into a Deployment Architecture 63

Preface

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

This overview attempts to clarify technical concepts and terminology used in the Java Enterprise System documentation set. Key technical terms are explained in the "Key Terms" section of each chapter, which clarifies how these terms are used in the Java Enterprise System context.

This preface contains the following sections:

- "Audience" on page 12
- "Using the Documentation" on page 12
- "Conventions" on page 14
- "Resources on the Web" on page 14
- "How to Report Problems" on page 15
- "Sun Welcomes Your Comments" on page 15

Audience

The *Java Enterprise System Technical Overview* is meant for individuals who will be designing, deploying, or maintaining software solutions based on Java Enterprise System. This constitutes a broad audience, which includes business analysts, system architects, field engineers, and system administrators.

Individuals reading the *Java Enterprise System Technical Overview* should have some familiarity with the following technologies:

- General networking concepts
- Security fundamentals relating to authentication and authorization
- The Java language, Java 2 Standard Edition components, and Java 2 Enterprise Edition components

Using the Documentation

Java Enterprise System manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML) formats. Both formats are readable by assistive technologies for users with disabilities. The Sun™ documentation web site can be accessed at the following location:

`http://docs.sun.com`.

Java Enterprise System documentation can be accessed here:

`http://docs.sun.com/prod/entsys.05q1`

The following table lists the system-level manuals in the Java Enterprise System documentation collection. The left column provides the name and part number location of each document and the right column describes the general contents of the document.

Table 1 Java Enterprise System Documentation

Document	**Contents**
Java Enterprise System Release Notes `http://docs.sun.com/doc/819-0057`	Contains the latest information about Java Enterprise System, including known problems. In addition, components have their own release notes.
Java Enterprise System Documentation Roadmap `http://docs.sun.com/doc/819-0055`	Provides descriptions of the documentation related to Java Enterprise System. Includes links to the documentation associated with components.

Table 1 Java Enterprise System Documentation *(Continued)*

Document	Contents
Java Enterprise System Technical Overview http://docs.sun.com/doc/819-0061	Introduces the technical and conceptual foundations of Java Enterprise System. Describes components, architecture, processes, and features.
Java Enterprise System Deployment Planning Guide http://docs.sun.com/doc/819-0058	Provides an introduction to planning and designing enterprise deployment solutions based on Java Enterprise System. 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 Enterprise System.
Java Enterprise System User Management Guide http://docs.sun.com/doc/817-5761	Helps you plan, deploy, and manage information about the users of your Java Enterprise System solution. Complements the *Java Enterprise System Deployment Planning Guide* by describing user management issues in each phase of the solution life cycle.
Java Enterprise System Deployment Example Series: Evaluation Scenario http://docs.sun.com/doc/819-0059	Describes how to install Java Enterprise System on one system, establish a set of core, shared, and networked services, and set up user accounts that can access the services that you establish.
Java Enterprise System Installation Guide http://docs.sun.com/doc/819-0056	Guides you through the process of installing Java Enterprise System for the Solaris™ Operating System or the Linux operating system. Shows how to select components to install, how to configure those components after installation, and how to verify that the configured components function properly.
Java Enterprise System Upgrade and Migration Guide http://docs.sun.com/doc/819-0062	Provides the information and instructions to upgrade Java Enterprise System for the Solaris™ Operating System or the Linux operating environment.
Java Enterprise System Glossary http://docs.sun.com/doc/816-6873	Defines terms that are used in Java Enterprise System documentation.

In addition to the system-level documents listed in this table, the Java Enterprise System documentation set includes component-specific documentation for many Java Enterprise System components. See the *Java Enterprise System Documentation Roadmap* for details.

Conventions

The following table describes the typeface conventions used in this book.

Table 2 Typeface Conventions

Typeface	Meaning	Examples
`AaBbCc123` (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, on-screen computer output, sample code.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
AaBbCc123 (Italic)	Book titles.	Read Chapter 6 in the *User's Guide*.
	New words or terms.	These are called *class* options.
	Words to be emphasized.	You *must* be superuser to do this.
	Command-line variables to be replaced by real names or values.	The file is located in the *install-dir*`/bin` directory.

Resources on the Web

The following location contains information about Java Enterprise System and its component products:

`http://www.sun.com/software/javaenterprisesystem/index.html`

How to Report Problems

If you have problems with Java Enterprise System, contact Sun customer support using one of the following mechanisms:

- Sun Software Support services online at

  `http://www.sun.com/service/sunone/software`

  This site has links to the Knowledge Base, Online Support Center, and ProductTracker, as well as to maintenance programs and support contact numbers.

- The telephone dispatch number associated with your maintenance contract

So that we can best assist you in resolving problems, please have the following information available when you contact support:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

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 document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java Enterprise System Technical Overview*, and the part number is 819-0061-10.

Chapter 1

Introduction to Java Enterprise System

Sun Java™ Enterprise System (Java ES) is a set of software components that provide services needed to support enterprise-strength applications distributed across a network or Internet environment. These applications are referred to in this book as distributed enterprise applications.

Java Enterprise System also represents a Sun software release and delivery methodology and a business and pricing strategy. The focus of this book, however, is on the software components of Java Enterprise System and the services they provide.

This chapter introduces Java Enterprise System and the tasks involved in using the system. It covers the following topics:

- "Why Do You Need Java Enterprise System?" on page 18
- "Java Enterprise System Components" on page 20
- "Working With Java Enterprise System" on page 25
- "Key Terms in This Chapter" on page 29

Why Do You Need Java Enterprise System?

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.

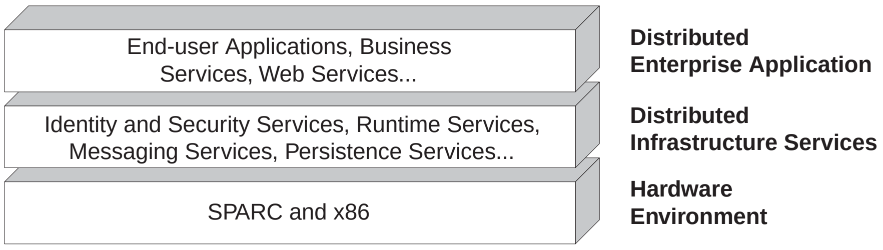
Java Enterprise System provides infrastructure services needed to support such *distributed enterprise applications*, that is, applications that generally have the following characteristics:

- **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 an underlying set of infrastructure *services* that allows their distributed components to communicate with each other, coordinate their work, implement secure access, and so forth. These infrastructure services are, in turn, 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 Enterprise System provides the distributed infrastructure services layer shown in Figure 1-1. However, Java Enterprise System services also include a number of application-level services accessible to end users.

Figure 1-1 Support Needed for Distributed Enterprise Applications


End-user Applications, Business Services, Web Services...
Distributed Enterprise Application
Identity and Security Services, Runtime Services, Messaging Services, Persistence Services...
Distributed Infrastructure Services
SPARC and x86
Hardware Environment

Among the featured services provided by Java Enterprise System are the following:

- **Portal services.** These services enable employees, telecommuters, knowledge workers, business partners, suppliers, and customers to access corporate resources from anywhere inside or outside the corporate network. These services provide access capabilities to user communities anytime, anywhere, delivering personalized integration, aggregation, security, mobile access, and search.
- **Communications 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.
- **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 Java 2 Platform, Enterprise Edition (J2EE™) technology.

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

- **Availability services.** These services provide near-continuous availability and for application components and for the infrastructure components that support them.
- **Access services.** These services provide Internet or browser-based access to Java Enterprise System services.
- **Administrative services.** These services help maintain and tune the performance of applications supported by Java Enterprise System.

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

Java Enterprise System Components

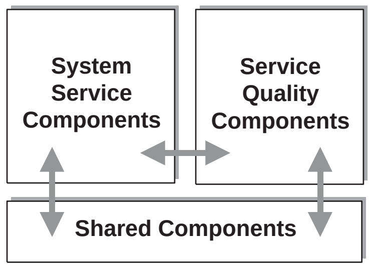
Java Enterprise System is an integration of previously discrete software products into a single software system. The components of this system have been tested together to ensure interoperability. Their integration is facilitated by a number of system-level features:

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

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

- **System service components.** These components provide the main Java Enterprise System infrastructure services that support distributed enterprise applications.
- **Service quality components.** These components enhance the availability, security, scalabililty, serviceability and other qualities 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.

Figure 1-2 Categories of Java Enterprise System Components



System Service Components

A number of Java Enterprise System components provide the main services that support distributed software solutions. These *system services* include portal services, communication and collaboration services, identity and security services, web container services, and J2EE application services.

The *system service components* that provide these distributed services, and the services they provide are briefly described in the following table. Each system service component is a multi-threaded server process capable of supporting a large number of clients. For more details on any component, see "System Service Component Descriptions" on page 72.

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.
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. The Application Server also provides web container services.
Sun Java System Calendar Server	Provides calendar and scheduling services to end users and groups of end users. Calendar Server includes a browser-based client that interacts with the server.
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.
Sun Java System Instant Messaging	Provides secure, real-time communication between end users, such as instant messaging (chat), conferencing, alerts, news, polls, and file transfer. The service includes a presence manager that tells users who is currently on line and includes a browser-based client that interacts with the server.

Table 1-1 Java ES System Service Components *(Continued)*

Component	System Services Provided
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 Messaging Server	Provides secure, reliable, high-capacity store-and-forward messaging that supports email, fax, pager, voice, and video. Messaging Server can concurrently access multiple message stores and provides content filtering to help reject unsolicited email and prevent virus attacks.
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 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.

Service Quality Components

In addition to the system service components shown in Table 1-1, Java Enterprise System includes a number of components used to enhance the quality of services provided by system service components or custom-developed application components. These *service quality components* fall into the following categories:

- Availability components
- Access components
- Administrative components

Availability Components

Availability components provide near-continuous uptime for system service components and application components. The availability components included in Java Enterprise System and the services that they provide are shown in the following table. For more details on any component, see "Availability Component Descriptions" on page 77.

Table 1-2 Java ES Availability Components

Component	Availability Services Provided
Sun Cluster	Provides high availability and scalability services for Java Enterprise System, the applications that run on top of the Java Enterprise System infrastructure, and the hardware environment in which both are deployed.
High Availability Session Store	Provides a data store that makes application data, especially session state data, available even in the case of failure.

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 function as well. The access components included in Java Enterprise System and the services that they provide are shown in the following table. For more details on any component, see "Access Component Descriptions" on page 78.

Table 1-3 Java ES Access Components

Component	Access Services Provided
Sun Java System Directory Proxy Server	Provides security services for Directory Server from outside a corporate firewall. Directory Proxy Server provides enhanced directory access control, schema compatibility, attribute filtering, and routing for multiple Directory Server instances.
Sun Java System Portal Server Secure Remote Access	Provides secure, Internet access from outside a corporate firewall to Portal Server content and services, including internal portals and Internet applications.
Sun Java System Communications Express	Provides web-based access to Messaging Server, Calendar Server, and Directory Server, depending on configuration.
Sun Java System Connector for Microsoft Outlook	Provides desktop clients using Microsoft Outlook with an interface to both Messaging Server and Calendar Server.

Administrative Components

Administrative components provide management functions, such as configuration and monitoring, for system services. The administrative components included in Java Enterprise System and the services that they provide are shown in the following table. For more details on any component, see "Administrative Component Descriptions" on page 80.

Table 1-4 Java ES Administrative Components

Component	Administrative Services Provided
Sun Java System Administration Server (and Console)	Provides a graphical administration tool that lets you configure and manage Directory Server and Messaging Server.
Sun Java System Directory Preparation Tool	Provides a script for configuring Directory Server with the schema needed to provision users for Messaging Server and Calendar Server.
Sun Java System Delegated Administrator	Provides both command-line and GUI tools for populating user entries in Directory Server with user attributes needed by Messaging Server and Calendar Server.
Sun Remote Services Net Connect	Provides a remote monitoring capability.

Shared Components

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

Shared components are often used to provide portability across different operating systems. Examples of Java Enterprise System shared components include: Java 2 Platform, Standard Edition (J2SE™ platform), Netscape Portable Runtime (NSPR), Network Security Services (NSS), Network Security Services for Java (JSS), and so forth. For a complete list, see "Shared Components" on page 81.

Working With Java Enterprise System

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

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

Java Enterprise System Solution Life Cycle

The tasks involved in creating business solutions based on Java Enterprise System software can be divided into several phases, as shown in Figure 1-3 on page 26. The illustration also shows the category of Java Enterprise System user that generally performs the various tasks.

The life-cycle phases shown in Figure 1-3 can be divided into the following general groupings:

- **Predeployment**. In these phase, a business need is translated into a deployment scenario: a logical architecture and a set of quality of service requirements. The deployment scenario serves as a specification used to design 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 solution life cycle and the tasks in each of the phases shown in Figure 1-3 are discussed more fully in Chapter 4, "Java Enterprise System Solution Life-Cycle Tasks."

Figure 1-3 Solution Life-Cycle Phases and User Categories

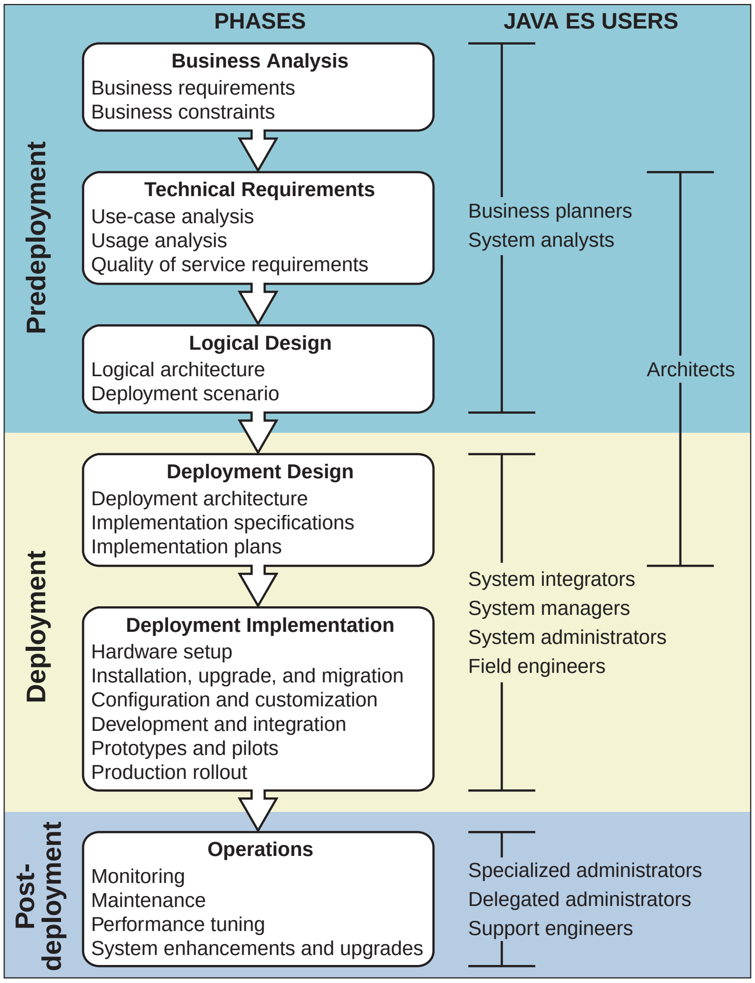


Figure 1-3 shows the Java Enterprise System users who typically perform the tasks shown for the life-cycle phases. If you are working with Java Enterprise System, your job should fit one or more of the user categories shown in Figure 1-3. The following table describes the skills and background for each 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.	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 Enterprise System Adoption Scenarios

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

- **New system.** Starting with no existing software system, you deploy Java Enterprise System 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 Enterprise System software. You normally 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. In other words you want a better return on investment than provided by your existing system.

- **Extension.** Starting with an existing IT infrastructure, you deploy Java Enterprise System software that is not currently part of your system. You normally extend your software system in this way because you need to meet new business needs. You might need new functional capabilities such as personalized aggregation of existing services through a Java Enterprise System portal, or Java authentication and authorization for existing services.
- **Upgrade.** Starting with an IT infrastructure consisting of an earlier version of Java Enterprise System or of Sun products that predate Java Enterprise System, you upgrade to the most current version of Java Enterprise System components.

Each adoption scenario has its own concerns and challenges. Regardless of which adoption scenario characterizes your situation, the solution life-cycle process shown in Figure 1-3 applies. Depending on your adoption scenario, however, the issues you need to address and the resources you need to invest in the life-cycle phases might vary.

The following concerns generally 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 Enterprise System 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 Enterprise System adoption scenarios.

Table 1-6 Concerns Related to Different Java ES Adoption Scenarios

Adoption Scenario	Migration	Integration	Training	Hardware
New system	Not a concern	Relatively easy to integrate new components	Normally a significant concern	Trade-offs between equipment costs and labor costs.*
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	Generally 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

* 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 the relationships between these terms how they are used in the Java Enterprise System context.

adoption scenario An overall reason for deploying Java Enterprise System software, characterizing the software system you start with and the goal you are trying to achieve. There are four basic Java Enterprise System adoption scenarios: 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 component*s included in Java Enterprise System or an *application component* that is custom developed. An application component usually conforms to a distributed component model (such as CORBA and the J2EE™ platform) and performs some specific computing function. These components, singly or combined, provide *business service*s and can be encapsulated as *web service*s.

distributed enterprise application An application whose logic spans a network or Internet environment (the distributed aspect) and whose scope and scale 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, 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).

service quality component One of a number of kinds of *system component*s included in Java Enterprise System. Support components, which include access components and administrative components, provide support for *system service component*s.

shared component One of a number of kinds of *system component*s included in Java Enterprise System. Shared components, usually libraries, provide local services to other system components. By contrast, a *system service component* provides distributed infrastructure services to other system components (or to *application components*).

system component Any software package or set of packages included in the Java Enterprise System and installed by the Java Enterprise System installer. There are several kinds of system components: *system service components* that provide distributed infrastructure *services*, *service quality components* which support the system services components by providing access and administrative 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 Enterprise System. 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 One of a number of kinds of *system components* included in Java Enterprise System. System services components provide the main Java Enterprise System infrastructure services: portal services, communication and collaboration services, identity and security services, web and application services, and availability services.

Chapter 2

Java Enterprise System Solution Architectures

This chapter provides an overview of the architectural concepts upon which Java Enterprise System (Java ES) solutions are based. The chapter demonstrates how Java ES components, both 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, followed by an example solution architecture based on that architectural framework.

The chapter covers the following topics:

- "Java Enterprise System Architectural Framework" on page 32
- "Example Java Enterprise System Solution Architecture" on page 46
- "Key Terms in This Chapter" on page 49

Java Enterprise System Architectural Framework

Java Enterprise System components support the deployment of distributed enterprise-strength software solutions.

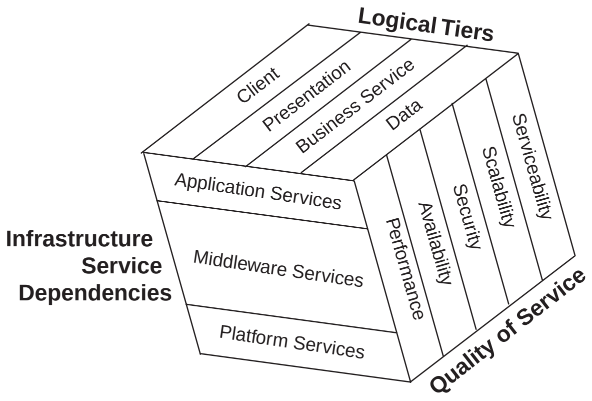
To achieve the required functionality at the levels of performance, availability, security, scalability, and serviceability mandated by business requirements, these software solutions must be properly designed.

A number of architectural dimensions are involved in designing distributed enterprise-strength software 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 (see "System Service Components" on page 21) in supporting distributed solutions.
- **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 22).

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

Figure 2-1 Dimensions of 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 an underlying set of infrastructure services that allows 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

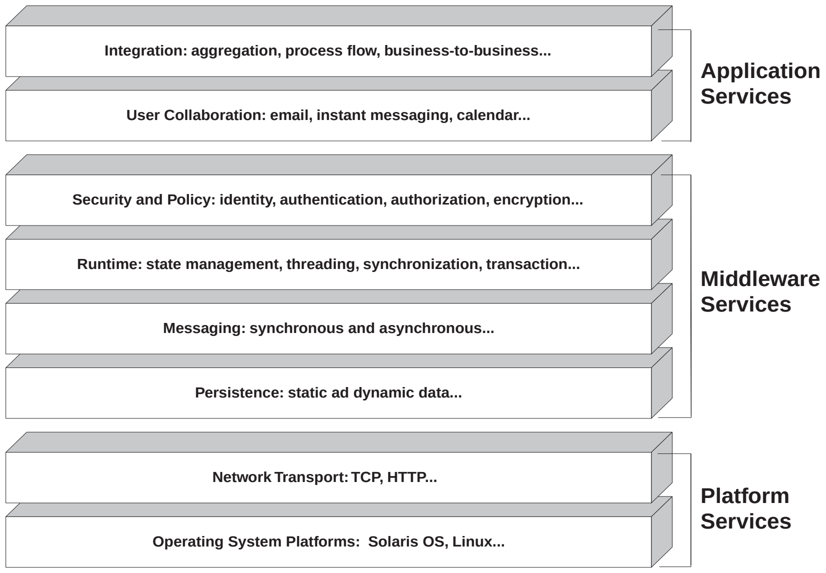
In designing a distributed software system, whether it consists mostly of custom-developed components or of out-of-the-box Java ES components, you need to incorporate a number of infrastructure services. These services operate at many levels.

The infrastructure service dependency dimension of solution architecture is illustrated in Figure 2-2 on page 34. The levels shown in this figure are an expanded view of the infrastructure service layer of Figure 1-1 on page 18.

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 conceptual basis for understanding the role of Java ES system service components (see "System Service Components" on page 21).

In general, the services shown in Figure 2-2 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.

Figure 2-2 Dimension 1: Infrastructure Service Levels


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 ad dynamic data...
Middleware Services
Network Transport: TCP, HTTP...
Operating System Platforms: Solaris OS, Linux...
Platform Services

The following paragraphs describe the different infrastructure service levels and refer to Java programming language artifacts, where relevant. The service levels are described 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 (such as Solaris™ Operating System, Linux, or Microsoft Windows) 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 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. As such, 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 workflow. Integration can also take place as business-to-business interactions between different enterprises.

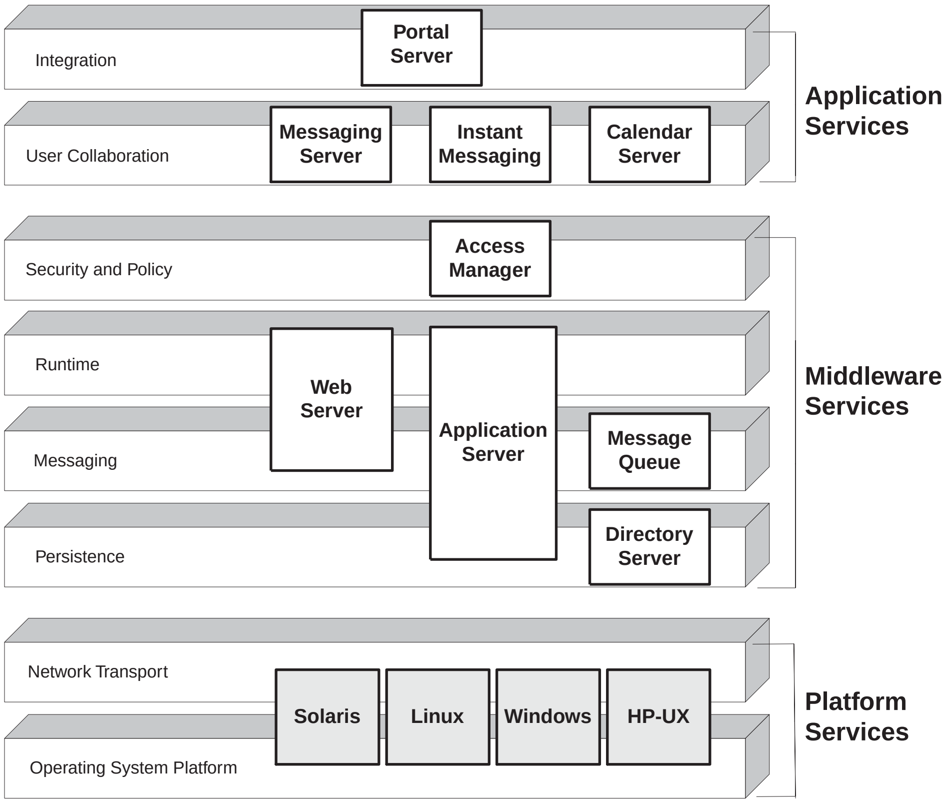
The service levels shown in Figure 2-2 reflect a general dependence of the various infrastructure services on one another, from the lowest-level operating system services to the highest-level application and integration services. Each service generally depends on services below it and supports services 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 Enterprise System Infrastructure Service Components

Java ES components implement the distributed infrastructure service levels shown in Figure 2-2. The positioning of Java ES system service components within the different levels is shown in Figure 2-3.

Figure 2-3 Java ES System Service Components



NOTE The operating system platforms shown in Figure 2-3 are not formally part of Java Enterprise System; however, they are included to show the operating system platforms on which Java ES components are supported.

Java Enterprise System 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.

Table 2-1 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 Instant Messaging	
Messaging Server	Directory Server Access Manager (for single sign-on)	Calendar Server (for email notifications) Portal Server (for messaging channel)
Instant Messaging	Directory Server Access Manager (for single sign-on)	Portal Server (for instant messaging channel)
Calendar Server	Directory Server Messaging Server (for e-mail notification service) Access Manager (for single sign-on)	Portal Server (for calendar channel)
Access Manager	Application Server or Web Server Directory Server	Portal Server If configured for single sign-on: Calendar Server Messaging Server Instant Messaging
Application Server	Message Queue Directory Server (for administered objects)	Portal Server Access Manager
Message Queue	Directory Server (for administered objects)	Application Server

Table 2-1 Relationships Between Java ES System Service Components *(Continued)*

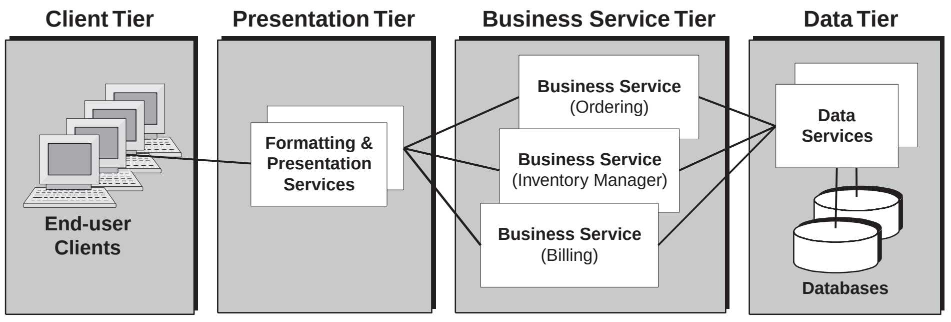
Component	Depends On	Provides Support To
Web Server	Access Manager (for access control)	Portal Server Access Manager
Directory Server	None	Portal Server Calendar Server Messaging Server Instant Messaging Access Manager

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.

Figure 2-4 Dimension 2: Logical Tiers for Distributed Enterprise Applications


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

For the most part, logical tier architectures represent the distributed enterprise application layer of Figure 1-1 on page 18. 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. However, logical tier concepts also apply to system service components that provide application-level services, such as Messaging Server and Calendar Server.

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 Java 2 Platform, Enterprise Edition (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 components or JSP components that prepare data for delivery in HTML or XML format or that 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 tax calculation service. Individual components and service assemblies can be encapsulated as loosely coupled *web services* within a service oriented architecture model and that conform to Simple Object Access Protocol (SOAP) interface standards. Business services can also be built as standalone *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 on page 38 emphasizes the logical and physical independence of components, represented by four separate tiers. These tiers signify the partitioning of application logic across the 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 tier 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.

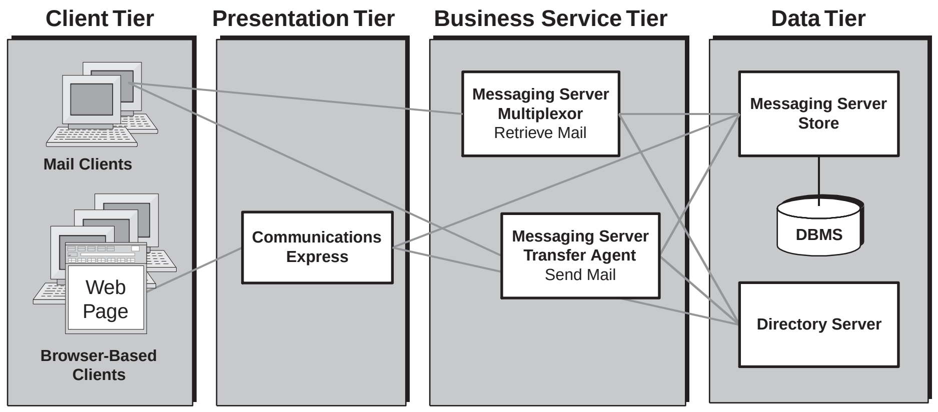
How you map application components or infrastructure components to a hardware environment (that is, your deployment architecture) depends on many factors, depending on the scale and complexity of your software solution. For very small deployments, a deployment architecture might involve only a few computers. 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 on page 36, Java ES infrastructure service components provide the underlying infrastructure support for distributed software solutions. Some of these solutions, however, include application-level services provided directly by 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.

Figure 2-5 Messaging Server: Example of Tiered Architecture



NOTE Figure 2-3 is not meant to be a complete logical architecture; a number of Java ES components are omitted to simplify the illustration. Lines connecting components represent interactions.

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"). 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. However, an 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 have to 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 describe 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 Enterprise System 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 22, are summarized as follows:

- **Availability components.** These components provide near-continuous uptime of a deployed solution.
- **Access components**. These components provide secure Internet access to system services, and often provide a routing function as well.
- **Administrative components.** These components provide enhanced serviceability for system components.

The following table shows 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
Communications Express	Security Scalability
Directory Proxy Server	Security Scalability
High Availability Session Store	Availability
Portal Server Secure Remote Access	Security Scalability
Sun Cluster	Availability Scalability
Sun Remote Services Net Connect	Serviceability

Sun Cluster Software

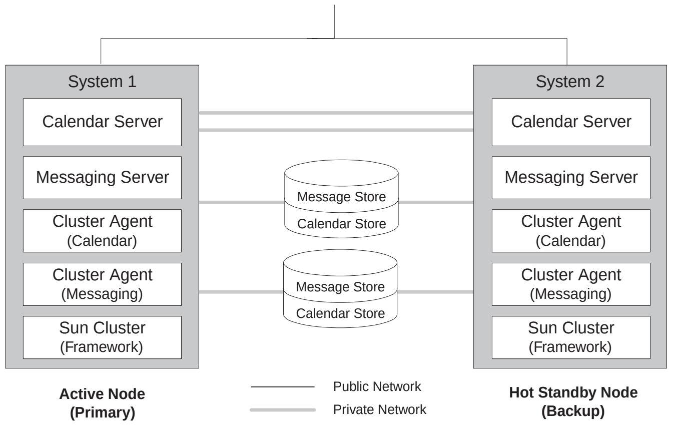
Sun Cluster software provides high availability and scalability services for Java ES components and for applications supported by Java ES infrastructure.

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, thereby using the internal redundancy to provide near-continuous access to these resources.

A two-node cluster to support data store services for Messaging Server and Calendar Server is shown in the following figure.

Figure 2-6 Availability Design Using Sun Cluster Nodes



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.

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. Any solution architecture needs to 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 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; and 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.

It is beyond the scope of this book to outline detailed design methodologies based on the architectural framework represented by Figure 2-1. However, the three-dimensional architecture framework highlights aspects of design that are important to understand in deploying software solutions based on Java Enterprise System.

Example Java Enterprise System Solution Architecture

Java Enterprise System 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 Enterprise System. 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 Simple Object Access Protocol (SOAP) interface standards. Many solutions involve a combination of these two approaches.

This section provides an example that demonstrate how Java Enterprise System 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	Java ES 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	Email messaging between employees and with outside world.	Communication and Collaboration services
Calendar	Electronic employee calendaring and meeting arrangements.	

Table 2-4 Business Requirements Summary: Communications Scenario *(Continued)*

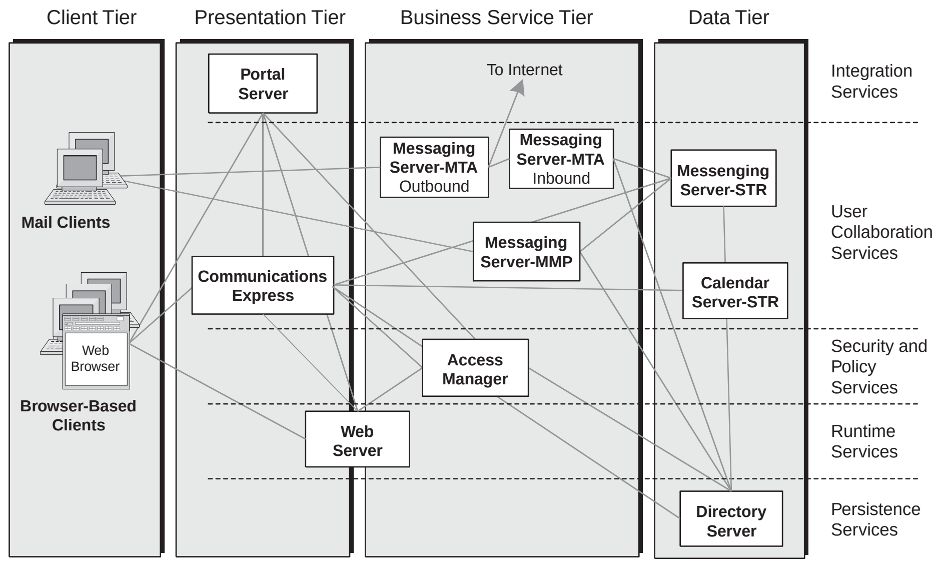
Business Requirement	Description	Java ES Services Needed
Portal access	Single, web-based, personalized access point to communication services such as email and calendar as well as internal web pages.	Portal services

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 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 2-7 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 Java ES 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 *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 might be met for many components 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 *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 the relationships between these terms how they are used in the Java Enterprise System context.

application component A custom-developed software *component* that performs some 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 multi-threaded process). A business service can also be an assembly of distributed components encapsulated as a *web service*, or it can be a standalone *server*.

client Software that requests software *services*. (Note: this is not a person—see *end user*.) 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 multi-threaded 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 (Simple Object Access Protocol) messaging protocol, the WSDL (Web Service definition Language) interface definition, and the UDDI (Universal Discovery, Description, and Integration) registry standard.

Chapter 3

Java Enterprise System 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 Enterprise System, as compared to manually integrating disparate infrastructure products.

The chapter covers the following features:

- "The Java Enterprise System Integrated Installer" on page 52
- "Integrated Identity and Security Services" on page 54
- "Key Terms in This Chapter" on page 58

The Java Enterprise System Integrated Installer

All Java ES components are installed using a single installer. This installer provides consistent installation and uninstallation procedures and behavior across all components.

The Java ES installer is an integrated framework that transfers Java ES software to a host system. The installer lets you select and install any number of Java ES components on any computer 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 seven languages: French, German, Spanish, Korean, Simplified Chinese, Traditional Chinese, and Japanese.

This section discusses the following aspects of the integrated Java ES installer (for more detailed information see the *Java Enterprise System Installation Guide*):

- Preexisting Software Checking
- Dependency Checking
- Initial Configuration
- Uninstallation

Preexisting Software Checking

The installer examines the computer on which you are installing and identifies the Java ES components that are already installed. The installer then checks at several levels to make sure 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 (see "Shared Components" on page 24), such as J2SE or NSS, that are already installed. If the installer finds versions of shared components that are incompatible, it lists them. 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. The installer provides logic to ensure that those dependencies are met. 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 are able to 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 can 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 have to 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 Enterprise System also provides an uninstallation program. You can use this 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 it discovers such a dependency. The uninstaller does not remove Java ES shared components.

The uninstaller, like the installer, can be run in graphical mode, text-based mode, or silent mode.

Integrated Identity and Security Services

An important feature of Java Enterprise System is its integrated management of user identities and its integrated authentication and authorization framework.

The following sections provide technical background for understanding the integrated identity and security services provided by Java Enterprise System:

- Single Identity
- Authentication and Single Sign-On

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.

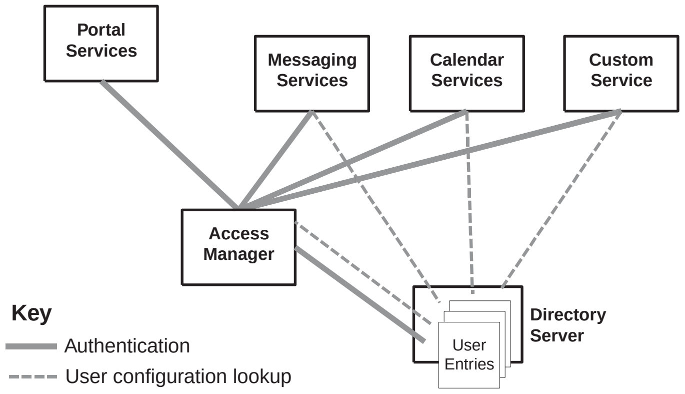
This integrated identity and security capability is based on close collaboration between Directory Server, Access Manager, and other Java ES components.

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*. That information normally includes information 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 Enterprise System, 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 Enterprise System.

Authentication and Single Sign-On

Java ES authentication and authorization services are provided by Access Manager. 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 makes use of 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 and Communications Express, the functionality of the policy agent is provided by an Access Manager subcomponent (see "Sun Java System Access Manager 6 2005Q1" on page 72).

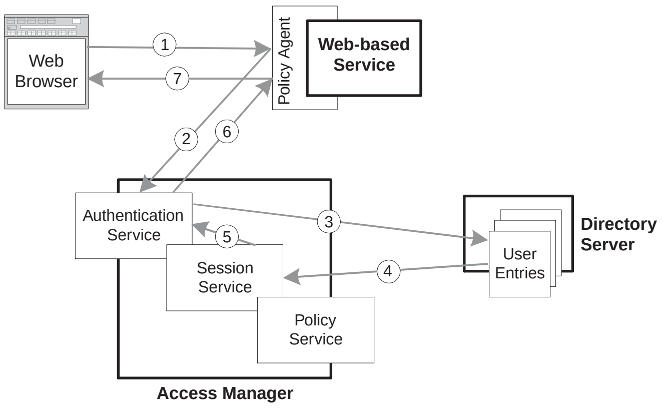
Authentication

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 has to 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 log-in request is valid, the user is authenticated (4), (5), and (6), and the company phone book is displayed to the employee (7). If the log-in 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.

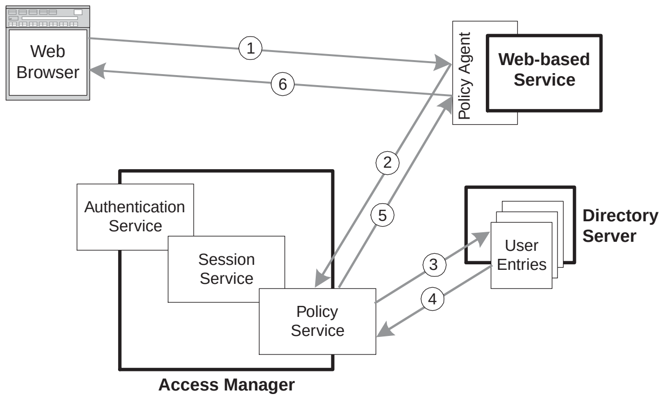
When the authenticated user attempts to access another secured service, the browser passes the session token to the corresponding policy agent. The policy agent verifies with the session service that the user's previous authentication remains valid, and the user is granted access to the second service without being asked to re-enter 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 Enterprise System. 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 prevents 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 the relationships between these terms how they are used in the Java Enterprise System 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 Enterprise System directory. Based on this single user entry a user can be allowed access to various Java Enterprise System resources, such as a portal, 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 Enterprise System Solution Life-Cycle Tasks

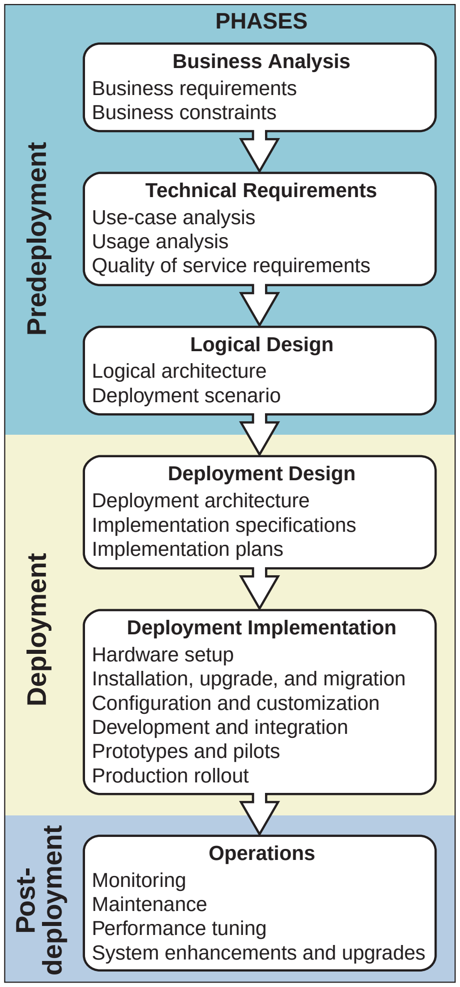
The Java ES solution life cycle was introduced in Chapter 1, "Introduction to Java Enterprise System,"as a standard approach to implementing business solutions using Java ES software. This chapter describes the tasks involved in each phase of the life cycle. The life-cycle diagram is repeated in Figure 4-1 on page 60 for easy reference.

The chapter discusses concepts and terminology relevant to each phase. The focus of this chapter is on deployment tasks, particularly deployment design and deployment implementation tasks.

The chapter is organized around the following three groupings of life-cycle phases:

- "Predeployment" on page 61
- "Deployment" on page 62
- "Postdeployment" on page 67
- "Key Terms in This Chapter" on page 68

Figure 4-1 Solution Life-Cycle Tasks



Predeployment

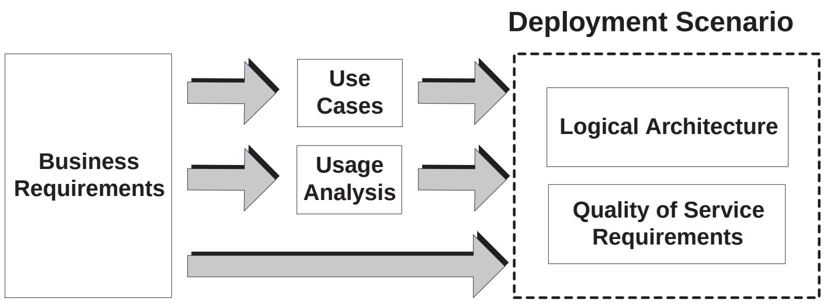
In the *predeployment* phases of the life cycle, you translate an analysis of business needs into a *deployment scenario*. The deployment scenario serves as a specification for a deployment design.

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

- **Business analysis.** In this phase, you 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.** In this phase, you 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 on page 42) that the proposed deployment must meet.
- **Logical design.** In this phase, you 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 Enterprise System Solution Architectures," you design a logical architecture. The logical architecture shows all the components, and all the interactions between the 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 on the predeployment phases of the life cycle, see the *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, which you then implement, test, and roll out in a production environment.

The deployment process generally encompasses software components in all the tiers and in all the infrastructure service levels required to support a software solution. In general, you have to 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.** Deployment design depends both 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.** Implementation of a deployment design is an often iterative process that involves hardware setup, software installation and configuration, development and integration, testing, and other aspects of production rollout.

The following sections look more closely at these two phases of the deployment process.

Deployment Design

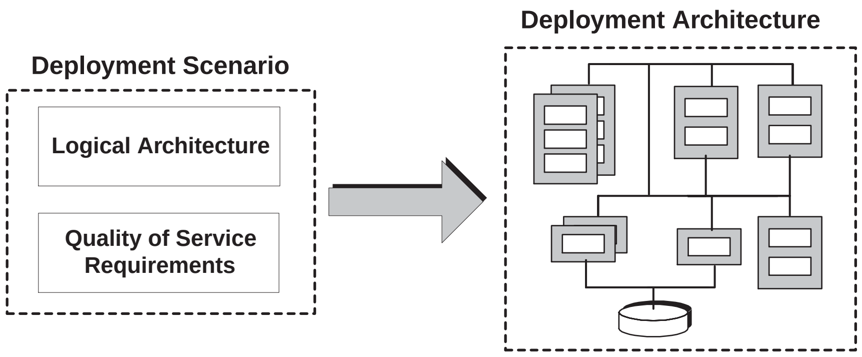
In the deployment design phase, you create a high-level deployment architecture followed by low-level implementation specifications.

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 has been completed, 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 have 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. As a starting point for deployment design, however, Java Enterprise System is developing a set of *reference deployment architectures*.

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 they are normally performed. For detailed documentation of these tasks see the *Java Enterprise System Documentation Roadmap* for details.

Hardware Setup

The implementation specification includes all of the details of your physical environment: the computers, network design, network hardware (including cabling, switches, routers, and load balancers), storage devices, and so forth. All 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 Enterprise System Integrated Installer" on page 52).

Your installation plan describes the sequence and scope of installer sessions. The approach you take to performing installation, however, might depend upon whether you are performing a new installation of Java Enterprise System, whether you are upgrading previously installed Java ES components, or whether you are replacing third-party components with Java Enterprise System. 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 upon which it interacts.

High availability must also be configured, depending on your availability solution for each component. Users need to be provisioned so they can access various services, and authentication and authorization policies and controls need to be set up (see "Integrated Identity and Security Services" on page 54).

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 Messaging Server to use virus checking and anti-spam filtering.

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 provides tools for programming distributed components or business services. Sun Java Studio simplifies 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 lets you 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 need to verify your deployment architecture; you need to 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 the like. You normally 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 following tasks are involved in the operations phase of the life cycle:

- Monitoring. These tasks include regular monitoring of system performance and system functions.
- Maintenance. These tasks include 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. These tasks include 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. These tasks include adding new Java ES components to a system to add new functionality or to replace non-Java ES components. In either case, 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

Each Java ES component has its own administration tools for configuring, tuning, or managing its operations. The goal is to provide a common monitoring and management infrastructure and administration tools for managing the system as a whole.

Key Terms in This Chapter

This section explains key technical terms used in this chapter, with an emphasis on clarifying the relationships between these terms how they are used in the Java Enterprise System context.

deployment A stage of the Java Enterprise System solution life-cycle process 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 Enterprise System solution and the quality-of-service requirements that the solution must satisfy to meet business needs. The quality-of-service requirements include requirement regarding: performance, availability, security, serviceability, and scalability/latent capacity. A deployment scenario is the starting point for deployment design.

development A task in the Java Enterprise System solution deployment process, by which the custom components of a *deployment architecture* are programmed and tested.

predeployment A stage of the Java Enterprise System 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 Enterprise System 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

Reference Listing: Java Enterprise System Components

This appendix provides a reference listing of all Java ES components, grouped into the following categories:

- **System Service Component Descriptions.** These components provide key Java ES infrastructure services needed to support distributed, enterprise applications. These services, as described in "Why Do You Need Java Enterprise System?" on page 18, include portal services, communication and collaboration services, identity and security services, web and application services, and availability services.
- **Service Quality Component Descriptions.** These components are used to 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.
- **Shared Components.** These components are local libraries that can be shared by any Java ES components running on a particular host computer.

In this appendix, Java ES components are listed alphabetically within their respective categories and subcategories.

For a roadmap to documentation of the different components, refer to the *Java Enterprise System Documentation Roadmap* (`http://docs.sun.com/doc/819-0055`).

System Service Component Descriptions

Java ES system service components provide the infrastructure services needed to support distributed enterprise applications. The Java ES system service components are described in the following sections:

- Sun Java System Access Manager 6 2005Q1
- Sun Java System Application Server Enterprise Edition 8 2005Q1
- Sun Java System Calendar Server 6 2005Q1
- Sun Java System Directory Server 5 2005Q1
- Sun Java System Instant Messaging 7 2005Q1
- Sun Java System Message Queue 3 2005Q1
- Sun Java System Messaging Server 6 2005Q1
- Sun Java System Portal Server 6 2005Q1
- Sun Java System Web Server 6.1 2005Q1

Sun Java System Access Manager 6 2005Q1

Sun Java System Access Manager (Access Manager) provides an infrastructure for an organization to administer the processes used to manage the digital identities of customers, employees, and partners who use their web-based services and non web-based applications. Because these resources might be distributed across a wide 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.

The Java ES installer provides Access Manager as a single installable component. If needed, the following Access Manager subcomponents can be installed separately:

- **Identity Management and Policy Services Core.** Provides the means for creating and managing user identities, and for defining and evaluating policies that provide access to Java ES resources based on users' identities. This subcomponent also includes the Access Manager SDK and the Delegated Administrator (see "Sun Java System Delegated Administrator 6 2005Q1" on page 81) subcomponents.
- **Access Manager SDK.** Provides a remote interface to Access Manager. This subcomponent needs to be installed on any computer hosting a Java ES component that accesses Access Manager remotely.

- **Access Manager Administration Console.** This graphical interface consolidates identity services and policy management and provides a single 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.

Sun Java System Application Server Enterprise Edition 8 2005Q1

Sun Java System Application Server (Application Server) provides a J2EE-compatible platform for developing and deploying application services and web services. Application Server provides the infrastructure services for interaction between tightly coupled distributed components, including remote method invocation and other runtime services.

The Java ES installer provides Application Server as a single installable component. If needed, the following Application Server subcomponents can be installed separately:

- **Domain Administration Server.** Provides server-side administration functions, such as managing and configuring Application Server and deploying J2EE components and applications.
- **Application Server Administration Client.** Provides graphical administration clients that allow you to manage and configure Application Server installations and hosted applications. The Administration Client also assists with deploying applications.
- **Command Line Administration Tool.** Provides command-line administration clients that allow you to manage and configure Application Server installations and hosted applications. The tool also assists with deploying applications.
- **Load Balancing Plug-in.**
- **PointBase.** Provides an embedded database that can be used for persistence operations
- **Sample Applications.**

Sun Java System Calendar Server 6 2005Q1

Sun Java System Calendar Server (Calendar Server) is a scalable, web-based solution that is used for centralized calendaring and scheduling for enterprises and service providers. Calendar Server supports personal and group calendars as well as calendars for resources such as conference rooms and equipment.

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

Sun Java System Directory Server 5 2005Q1

Sun Java System Directory Server (Directory Server) 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, as well as extranet user authentication.

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

Sun Java System Instant Messaging 7 2005Q1

Sun Java System Instant Messaging (Instant Messaging) enables end users to participate in instant messaging and chat sessions, to send alert messages to each other, and to share group news instantly. Instant Messaging is suitable for both intranets and the Internet, and supports interaction with other instant messaging providers.

The Java ES installer provides Instant Messaging as a single installable component. The following Instant Messaging subcomponents can be installed separately:

- **Instant Messaging Server Core.** Includes server and multiplexor software.
- **Instant Messaging Resources.**
- **Access Manager Instant Messaging Service.**

Sun Java System Message Queue 3 2005Q1

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 the 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 are free to focus on the business logic of their applications, rather than on the low-level details of how their applications communicate across a network.

Message Queue is available in two editions:

- **Enterprise Edition** (default). Provides support for multi-broker message services, HTTP/HTTPS connections, secure and scalable connections, client connection failover, and client support for the C language. This edition is best suited to deploying and running messaging applications in a large-scale production environment.
- **Platform Edition.** Provides basic JMS support, and is best suited to small-scale deployments and development environments

The Java ES installer provides Message Queue Enterprise Edition and Message Queue Platform Edition as separately installable components.

Sun Java System Messaging Server 6 2005Q1

Sun Java System Messaging Server (Messaging Server) is a powerful, standards-based Internet messaging server for both enterprises and service providers. Designed for high-capacity, reliable message handling, Messaging Server consists of several modular, independently-configurable components that provide support for several e-mail protocols.

The Java ES installer provides Messaging Server as a single installable component. However, after installation, each Messaging Server instance can be configured to provide any of a number of different messaging services, in effect representing the following set of Messaging Server subcomponents:

- **Message Store.** Provides message storing and retrieving.

- **Message Transfer Agent (MTA).** Supports the sending of emails by handling SMTP connections, routing emails, and delivering messages to the proper message stores. Can be configured to deliver email to internal storage (inbound) or to external mail stores (outbound).
- **Message Multiplexor (MMP).** Supports the retrieving of emails by accessing the message store (or a set of stores) for email clients, using either IMAP or POP protocols.
- **Message Express Multiplexor (MEM).** Supports the retrieving and sending of emails by web-based (HTTML) email clients.

Sun Java System Portal Server 6 2005Q1

Sun Java System Portal Server (Portal Server) is an identity-enabled portal server solution. Portal Server combines key portal services, such as personalization, aggregation, security, integration, and search. Mobile access, a subcomponent of Portal Server, provides wireless access to the Portal Server from mobile devices, such as mobile phones and personal digital assistants.

The Java ES installer provides Portal Server, including mobile access, as a single installable component.

Sun Java System Web Server 6.1 2005Q1

Sun Java System Web Server (Web Server) is a multi-process, multi-threaded, secure web server built on open standards. Web Server provides high performance, reliability, scalability, and manageability for any size enterprise. Web Server supports a wide range of web software standards, including JDK 1.4.1, Java Servlet 2.3, JavaServer Pages™ (JSP™) 1.2, HTTP/1.1, PKCS #11, FIPS-140, 168-bit step-up certificates, and various other security-based standards.

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

Service Quality Component Descriptions

The components used to support Java ES services components are grouped into the following categories:

- Availability components
- Access components
- Administrative components

The components in these categories are described in the sections that follow.

Availability Component Descriptions

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:

- Sun Cluster 3.1 9/04 and Sun Cluster Agents
- High Availability Session Store 2005Q1

Sun Cluster 3.1 9/04 and Sun Cluster Agents

Sun Cluster software provides high availability and scalability services for Java Enterprise System as well as for applications based on Java ES infrastructure.

A cluster is a set of loosely coupled computers (cluster nodes) 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, 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 and the Sun Cluster Agents as separately installable components. Additional Sun Cluster agents are available on separate CDs.

High Availability Session Store 2005Q1

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 a the session is re-established.

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 one of these components 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.

Access Component Descriptions

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:

- Sun Java System Communications Express 2005Q1
- Sun Java System Connector for Microsoft Outlook 6 2005Q1
- Sun Java System Directory Proxy Server 5 2005Q1
- Sun Java System Portal Server Secure Remote Access 6 2005Q1

Sun Java System Communications Express 2005Q1

Sun Java System Communications Express (Communications Express) provides an integrated web-based communications and collaboration client that provides end users with a web interface to calendar, email, and address book services. Communications Express consists of three client modules: Calendar, Address Book, and Mail. Configurable to provide access to either Messaging Server or Calendar Server or both, Communications Express works with either Sun Java System LDAP Schema, Version 1 (Schema 1) or Schema 2.

The Java ES installer provides Communications Express as a single installable component.

Sun Java System Connector for Microsoft Outlook 6 2005Q1

Sun Java System Connector for Microsoft Outlook enables Outlook to be used as a desktop client with Sun Java Enterprise System. The connector is an Outlook plug-in that must be installed on the user desktop.

Connector for Microsoft Outlook queries Messaging Server for folder hierarchies and e-mail messages, then converts the information into Messaging API (MAPI) properties that Outlook can display. Similarly, Connector uses WCAP to query Calendar Server for events and tasks which are then converted into MAPI properties. With this model, Sun Java System Connector for Microsoft Outlook builds an end-user Outlook view from two separate information sources: mail from Messaging Server and calendar information from Calendar Server.

Sun Java System Connector for Microsoft Outlook is provided on the accessories CD, with its own installer.

Sun Java System Directory Proxy Server 5 2005Q1

Sun Java System Directory Proxy Server (Directory Proxy Server) is an essential component of any mission-critical directory service for e-commerce solutions. Directory Proxy Server is an LDAP application layer protocol gateway that offers enhanced directory access control, schema compatibility, and high availability using application layer load balancing and failover.

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

Sun Java System Portal Server Secure Remote Access 6 2005Q1

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.

The Java ES installer provides Portal Server Secure Remote Access as a single installable component. If needed, the following Portal Server Secure Remote Access subcomponents can be installed separately:

- **Portal Server Secure Remote Access Core.**
- **Gateway.** Provides an interface and security barrier to a corporate intranet that allows remote access from outside the intranet. Gateway presents content securely from internal web servers and application servers through a single interface to a remote user.

- **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.
- **Rewriter Proxy.** 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.

Administrative Component Descriptions

Administrative components provide management functions, such as configuration and monitoring, for system services. The following Java ES administrative components are described in this section:

- Sun Java System Administration Server (and Console) 5 2005Q1
- Sun Java System Directory Preparation Tool 2005Q1
- Sun Java System Delegated Administrator 6 2005Q1
- Sun Remote Services Net Connect 2005Q1

Sun Java System Administration Server (and Console) 5 2005Q1

Sun Java System Administration Server and Server Console together provide a graphical tool that lets you manage Directory Server, Messaging Server, and Directory Proxy Server. The Administration Server processes requests for servers installed in a server group under the same root directory, and then starts the programs required to fulfill the requests.

Server Console is a stand-alone Java application that works in conjunction with an instance of Directory Server and an instance of Administration Server on your network. Server Console acts as the front-end management application for Java ES software in your enterprise.

The Java ES installer provides Server Console and Administration Server together as a single installable component.

Sun Java System Directory Preparation Tool 2005Q1

The Sun Java System Directory Preparation Script is used for configuring Directory Server with the schema needed to provision users for Messaging Server and Calendar Server.

Sun Java System Delegated Administrator 6 2005Q1

The Sun Java System Delegated Administrator is a command-line utility (`commadmin`) for provisioning users, groups, domains, and resources for Calendar Server, Messaging Server and other Java ES service providers.

Delegated Administrator is automatically installed when you choose to install Access Manager.

Sun Remote Services Net Connect 2005Q1

Sun Remote Services Net Connect is a collection of system management services designed to help you better control your IT environment. These web-delivered services give you the ability to self-monitor systems, create performance and trend reports, and receive automatic notification of system events so you can act more quickly to manage potential issues before they become problems.

The Java ES installer provides Sun Remote Services Net Connect as a single installable component.

Shared Components

Shared components provide local services and technology support upon which Java ES system service components and service quality components depend. The Java ES installer automatically installs any shared components required to support otherJava ES components installed on a host computer.

Java Enterprise System includes the shared components listed below:

- Ant (Jakarta ANT Java/XML-based build tool)
- Apache SOAP (Simple Object Access Protocol) Runtime
- Berkeley DB
- Common agent container
- ICU (International Components for Unicode)
- J2SE™ (Java 2 Platform, Standard Edition) platform 5.0
- JAF (JavaBeans™ Activation Framework)
- JATO (Java Studio Enterprise Web Application Framework)
- JavaHelp™ Runtime
- JavaMail ™ Runtime

- JAXB (Java Architecture for XML Binding) Runtime
- JAXM (Java API for XML Messaging) Client Runtime
- JAXP (Java API for XML Processing)
- JAXR (Java API for XML Registries) Runtime
- JAX-RPC (Java API for XML-based Remote Procedure Call) Runtime
- JCAPI (Java Calendar API)
- JDMK (Java Dynamic Management™ Kit) Runtime
- JSS (Java Security Services)
- KTSE (KT Search Engine)
- LDAP C SDK
- LDAP Java SDK
- NSPR (Netscape Portable Runtime)
- NSS (Network Security Services)
- Perl LDAP, including NSPERL
- SAAJ (SOAP with Attachments API for Java)
- SAML (Security Assertions Markup Language)
- SASL (Simple Authentication and Security Layer)
- SNMP (Simple Network Management Protocol) Peer
- Sun Explorer Data Collector (Solaris only)
- Sun Java Monitoring Framework
- Sun Java Web Console
- Tomcat Servlet JSP Container
- XML C Library (`libxml`)
- WSCL (Web services Common Library)

Index

A

access components
  descriptions of 78
  introduced 23
Access Manager
  as infrastructure service 36
  as system service component 21
  description of 72
access services 19
Administration Server
  as service quality component 24
  description of 80
administrative components
  descriptions of 80
  introduced 24
administrative services 19
adoption scenarios, Java ES
  about 27
  defined 29
  enhancement 27
  extension 28
  new system 27
  upgrade 28
Ant (Jakarta ANT Java/XML-based build tool) 81
Apache SOAP Runtime 81
application components
  defined 49
  in logical tier architectures 38
Application Server
  as infrastructure service 36
  as system service component 21
  description of 73
application services 19, 33
applications
  distributed, *See* distributed enterprise applications
  enterprise, *See* distributed enterprise applications
architectural dimensions
  infrastructure service dependencies 33
  logical tiers 38
  quality of service 41
  synthesis o 45
architecture
  defined 49
  deployment 62
  dimensions of, *See* architectural dimensions
  introduced 31
  logical 61
  reference 63
  solution 32
authentication 56
authorization 57
availability
  requirements 42, 43
  services 44, 77
availability components
  descriptions of 77
  introduced 22
availability services 19

B

Berkeley DB 81
business services
  defined 49
  presentation tier, and 39

C

Calendar Server
  as infrastructure service 36
  as system service component 21
  description 74
CAPI (Java Calendar API) 82
client logical tier 39
clients
  defined 49
  system service components, and 21
clusters, *See* Sun Cluster
collaboration services 19
common agent container 81
communication services 19
Communications Express
  as service quality component 23
  description of 78
components 18
  access 23
  administrative 24
  and infrastructure services 36
  availability 22
  defined 29
  dependencies 37
  detecting installed versions of 53
  distributed 18
  EJB 39
  J2EE 39
  JSP 39
  service quality 22, 77
  Servlet 39
  shared 24, 81
  system service 21, 72
  system, *See* system components
Connector for Microsoft Outlook
  as service quality component 23
  description of 79

D

Delegated Administrator
  as service quality component 24
  description of 81
dependencies 37, 53
dependency checking, installer 53
deployment
  architecture 62
  defined 68
  design 62
  development and customization 66
  implementation 65
  life-cycle stage 62
  phases of life cycle 62
  production rollout 67
  prototype testing 66
  reference architectures 63
  scenarios, *See* deployment scenarios
deployment architectures
  defined 49
  design of 62
  introduced 31
  relation to tiered architectures 40
deployment scenarios
  defined 68
  introduced 61
detecting installed software 53
development
  as deployment task 66
  defined 68
directories
  as user data stores 55
  defined 58
Directory Preparation Tool
  as service quality component 24
  description of 80

Directory Proxy Server
- as service quality component 23
- as system component 43
- description of 79

Directory Server
- as infrastructure service 36
- as system service component 21
- description of 74

distributed
- applications, *See* distributed enterprise applications
- services, *See* distributed services

distributed enterprise applications
- about 18
- defined 30
- infrastructure for 18

distributed services
- access 19
- administrative 19
- application level 33
- availability 19
- communication and collaboration 19
- identity 19
- infrastructure 18
- integration 35
- messaging 34
- middleware 33
- network transport 34
- overview 18
- persistence 34
- platform 33, 34
- portal 19
- runtime 19, 35
- security 19, 35
- user collaboration 35
- web 19

documentation set, Java ES 12

E

EJB components 39

end users
- defined 30
- distributed applications, and 18

enhancements, *See* adoption scenarios

extensions, *See* adoption scenarios

H

hardware, Java ES adoption scenarios, and 28

High Availability Session Store
- as service quality component 23
- description of 78

I

ICU (International Components for Unicode) 81

identity
- management 54
- services 19, 54
- single user 54

implementation specifications 64

infrastructure
- for distributed enterprise applications 18
- service dependencies *See* distributed services

Instant Messaging
- as infrastructure service 36
- as system service component 21
- description of 74

integration
- features, *See* integration features
- Java ES adoption scenarios, and 28
- services 35

integration features
- identity and security 20, 54
- integrated installer 20, 52
- introduced 20
- shared components 20

J

J2EE
- components 39
- distributed component model 39
- platform 22

J2ME platform 39
J2SE (Java 2 Platform, Standard Edition) 24, 81
JAF (JavaBeans™ Activation Framework) 81
JATO (Java Studio Enterprise Web Application Framework) 81
Java Servlet components 39
JavaHelp 81
JavaMail 81
JAXB (Java Architecture for XML Binding) 82
JAXM (Java API for XML Messaging) 82
JAXP (Java API for XML Processing) 82
JAXR (Java API for XML Registries) 82
JAX-RPC 82
JDMK (Java Dynamic Management™ Kit) 82
JMS (Java Message Service) 22
JSP components 39
JSS (Java Security Services) 24, 82

K

KT Search Engine (KTSE) 82

L

language support 52
latent capacity requirements 42
LDAP 39, 58
LDAP C SDK 82
LDAP Java SDK 82
life-cycle phases
- deployment 25, 62
- postdeployment 25, 67
- predeployment 25, 61

Linux 36
logical architectures
- about 61
- defined 49
- example 47
- infrastructure service levels, and 33
- introduced 31

M

Message Queue
- as infrastructure service 36
- as system service component 22
- description of 75

Messaging Server
- as infrastructure service 36
- as system service component 22
- description of 75

messaging services 34
middleware services 33
migration, Java ES adoption scenarios, and 28

N

network transport services 34
NSPR (Netscape Portable Runtime) 24, 82
NSS (Network Security Services) 24, 82

O

operating system services 34

P

performance requirements 42
Perl LDAP 82
persistence services 34
platform services 33
policies
  authorization 57
  defined 58
Portal Server
  as infrastructure service 36
  as system service component 22
  description of 76
Portal Server Secure Remote Access
  as service quality component 23
  as system component 43
  description of 79
portal services 19
postdeployment
  defined 68
  phases of life cycle 67
predeployment
  defined 68
  phases of life cycle 61
production rollout 67
prototyping 66
provisioning users 64

Q

quality of service requirements
  availability 42, 43
  latent capacity 42
  performance 42
  scalability 42, 43
  security 42, 43
  serviceability 42, 43

R

reference deployment architectures
  defined 69
  introduced 63
Remote Services Net Connect
  as service quality component 24
  description of 81
runtime services 35

S

SAAJ (SOAP with Attachments API for Java) 82
SAML (Security Assertions Markup Language) 82
SASL (Simple Authentication and Security Layer) 82
scalability
  requirements 42, 43
  services 44, 77
security
  policy services 35
  requirements 42, 43
  services 19
servers
  defined 49
  standalone 39
service quality components
  defined 30
  descriptions of 77
  introduced 22
serviceability requirements 42, 43
services
  business 39
  defined 30
  high availability 44, 77
  infrastructure 18
  infrastructure, *See* distributed infrastructure services
  scalability 44, 77
  web 39

shared components 81
- defined 30
- introduced 24

single identity
- defined 58
- introduced 54

single sign-on
- defined 58
- implementation of 57
- infrastructure service levels, and 35
- Java ES feature 21, 55

SNMP (Simple Network Management Protocol) 82

SOAP (Apache) 81

Solaris 36

solutions, Java ES
- architecture 31
- custom and out-of-the-box 46
- example 46
- life cycle 25

Sun Cluster
- agents 44
- as availability service 44
- as service quality component 23
- description of 77

Sun Explorer Data Collector 82

Sun Java Monitoring Framework 82

Sun Java System products
- Access Manager, *See* Access Manager
- Administration Server, *See* Administration Server
- Application Server, *See* Application Server
- Calendar Server, *See* Calendar Server
- Communications Express, *See* Communications Express
- Connector for Microsoft Outlook, *See* Connector for Microsoft Outlook
- Delegated Administrator, *See* Delegated Administrator
- Directory Preparation Tool, *See* Directory Preparation Tool
- Directory Proxy Server, *See* Directory Proxy Server
- Directory Server, *See* Directory Server
- High Availability Session Store, *See* High Availability Session Store
- Instant Messaging *See* Instant Messaging
- Message Queue, *See* Message Queue
- Messaging Server, *See* Messaging Server
- Portal Server, Secure Remote Access, *See* Portal Server, Secure Remote Access
- Portal Server, *See* Portal Server
- Web Server, *See* Web Server

Sun Java Web Console 82

system
- components, *See* system components
- configuration 53
- services 18

system components
- about 20
- defined 30
- service quality components 22
- shared components 24, 81
- system service 72
- system service components, *See* system service components

system service components
- defined 30
- dependencies 37
- introduced 21

system services
- about 21
- defined 30

T

tasks, Java Enterprise System 25, 59

tiers, logical
- application architecture, and 38
- business service 39
- client 39
- data 39
- presentation 39

Tomcat Servlet JSP Container 82

training, Java ES adoption scenarios, and 28

U

uninstaller 54
upgrades, *See* adoption scenarios
use cases
  defined 69
  introduced 61
user categories
  architect 27
  business planner 27
  delegated administrator 27
  field engineer 27
  IT manager 27
  specialized system administrator 27
  system administrator 27
  system analyst 27
  system integrator 27
user collaboration services 35
user entry 54
user profiles 27
user provisioning 64
users, *See* end users

W

Web Server
  as infrastructure service 36
  as system service component 22
  description of 76
web services 19
  defined 50
  J2EE components, and 39
WSCL (Web services Common Library) 82

X

XML C Library 82