Sun Java™ System

Sun Java Enterprise System 2004Q2 Technical Overview

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

Part No: 817-5764-10

Contents

List of Figures

List of Tables

Preface

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

This overview attempts to clarify technical concepts and terminology used in the Java Enterprise System documentation set. Italicized terms are found in "Java Enterprise System Key Terms" on page 79, which defines the terms and clarifies how they are used in the Java Enterprise System context.

This preface contains the following sections:

Audience

The *Java Enterprise System Technical Overview* is meant for individuals who will be designing, deploying, or maintaining software solutions based on the 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:

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

Using the Documentation

The 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`.

The Java Enterprise System documentation can be accessed here:

`http://docs.sun.com/prod/entsys.03q4`

The following table lists the system-level manuals in the Java Enterprise System documentation set. 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/817-5503`	Contains the latest information about the Java Enterprise System, including known problems. In addition, component products have their own release notes.

Table 1 Java Enterprise System Documentation *(Continued)*

Document	Contents
Java Enterprise System Documentation Roadmap http://docs.sun.com/doc/817-5763	Provides descriptions of the documentation related to Java Enterprise System. Includes links to the documentation associated with the component products.
Java Enterprise System Technical Overview http://docs.sun.com/doc/817-5764	Introduces technical concepts and terminology used in Java Enterprise System documentation. Describes the Java Enterprise System, its components, and role in supporting distributed enterprise applications. Also covers life-cycle concepts, including an introduction to system deployment.
Java Enterprise System Deployment Planning White Paper http://docs.sun.com/doc/817-5759	Provides an introduction to planning large-scale deployments based on Java Enterprise System. Presents some basic concepts and principles of deployment planning and introduces a number of processes that you can use as a starting point when designing enterprise-wide deployments.
Java Enterprise System Installation Guide http://docs.sun.com/doc/817-5760	Guides you through the process of installing your Java Enterprise System. Shows you how to select the component products that you want to install, how to configure the component products that you install, and how to verify that the software you install functions properly.
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 product-specific documentation for each of the Java Enterprise System component products. 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://wwws.sun.com/software/learnabout/enterprisesystem/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. Use the web-based form to provide feedback to Sun:

http://www.sun.com/hwdocs/feedback/

Please provide the full document title and part number in the appropriate fields. The part number can be found on the title page of the book or at the top of the document, and is usually a seven or nine digit number. For example, the part number of this *Sun Java Enterprise System Technical Overview* is 817-5764-10.

Chapter 1

Introduction

Sun Java™ Enterprise System is a software infrastructure that provides the *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 is also a Sun software release and delivery methodology and a business and pricing strategy. The focus of this book, however, is on Java Enterprise System as a software system.

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

Java Enterprise System Services

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 to support such software solutions.

These software solution are applications with the following characteristics:

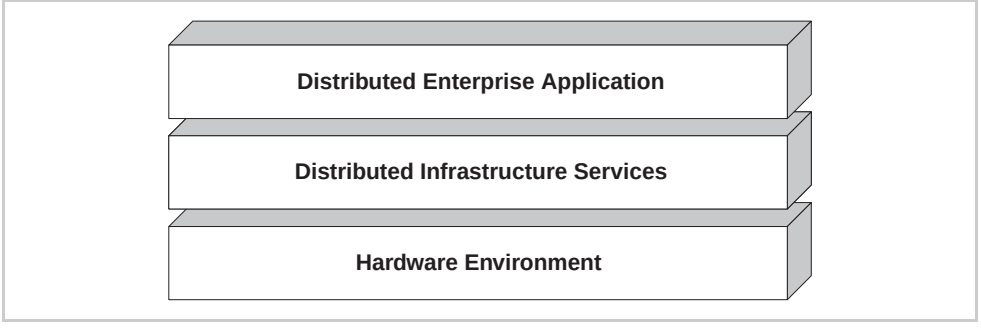
- **Distributed.** The application consists of interacting software components deployed across a networked environment that might include geographically remote sites. These *distributed components*, running on the various *computing nodes* in the environment, work together to deliver specific business functions to *end users* and to other business applications.
- **Enterprise-strength features.** 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 infrastructure that allows their distributed components to communicate with each other, coordinate their work, implement secure access, and so forth. This infrastructure consists of a number of distributed services.

These distributed infrastructure services are, in turn, supported by a hardware environment of computing nodes and network links. This environment includes SPARC and X86 (Intel and AMI) hardware architectures.

The overall scheme is shown in the following figure.

Figure 1-1 Support Needed for Distributed Enterprise Applications



Java Enterprise System provides the distributed infrastructure services layer shown in Figure 1-1. Java Enterprise System infrastructure services support a broad range of business services and applications. Among the infrastructure services provided by Java Enterprise System are the following:

- **Portal services.** Portal services enable mobile employees, telecommuters, knowledge workers, business partners, suppliers, and customers to securely access their personalized corporate portal from anywhere outside the corporate network through the Internet. These services provide anytime, anywhere access capabilities to user communities, delivering integration, aggregation, personalization, 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, 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 and application services.** These services enable IT organizations to develop, deploy, and manage applications for a broad range of servers, clients, and devices, based on Java 2 Platform, Enterprise Edition (J2EE™) technology.
- **Availability services.** These services deliver a unique approach to application service-level management. Availability services provide the patented “Always-On” technology for application and web services, delivering near-continuous availability and scalability.

You can selectively deploy one or more of these infrastructure 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 independent Sun software products into a single software system.

The components of this system (the *component products*) have been tested together to ensure interoperability. Their integration is facilitated by a number of system-level features:

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

The main components of Java Enterprise System and the infrastructure services they provide are listed in the following table. For more details on any component, consult "Java Enterprise System Server Components" on page 66.

Table 1-1 Java Enterprise System Components

System Component	Services Provided
Sun Cluster	Provides high availability and scalability services for the Java Enterprise System, the applications that run on top of the Java Enterprise System infrastructure, and the hardware environment in which both are deployed.
Sun ONE 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 it 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 Proxy Server	Provides security services for Directory Server from outside a corporate firewall. Directory Proxy Server provides enhanced directory access control, schema compatibility, routing, and load balancing for multiple Directory Server instances.
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.

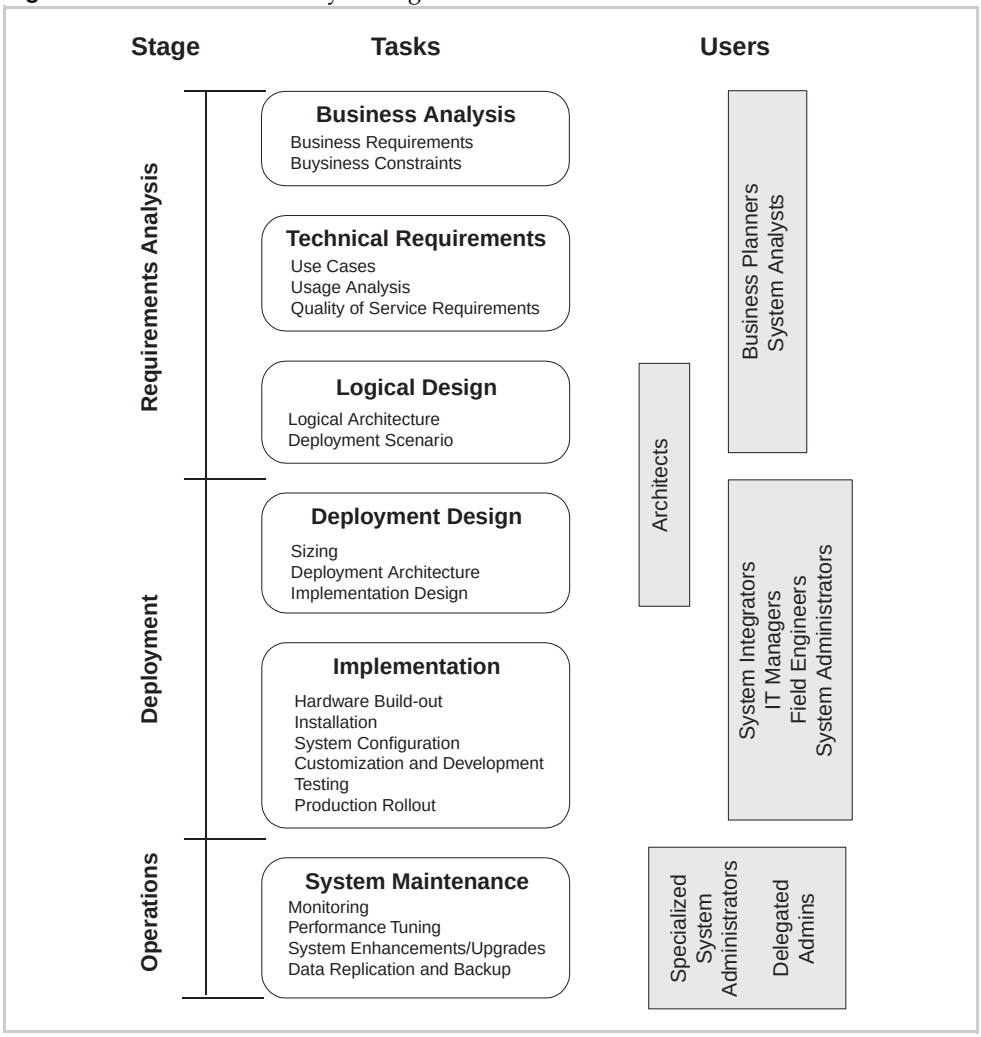
Table 1-1 Java Enterprise System Components *(Continued)*

System Component	Services Provided
Sun Java System Identity Server	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 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.
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. It 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 Portal Server Mobile Access	Provides wireless access to the Portal Server from mobile devices and voice access to the Portal Server from telephones.
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 ONE Web Server	Provides Java 2 Platform, Enterprise Edition (J2EE™ platform) web container services for Java web components, such as Java Servlet and JavaServer Pages™ (JSP™) components. The Web Server also supports other web application technologies for delivering static and dynamic web content, such as CGI scripts and Active Server Pages.

Working With Java Enterprise System

Creating business solutions based on Java Enterprise System software involves a complex set of tasks that can be divided into three stages: requirements analysis, deployment, and operations, as shown in the following figure.

Figure 1-2 Solution Life-Cycle Stages



The Java Enterprise System life-cycle stages can be described briefly as follows:

- **Requirements analysis.** Translate an analysis of business needs into a deployment scenario: a logical architecture and quality of service requirements. The deployment scenario serves as a specification for a software deployment.
- **Deployment.** Translate a deployment scenario into a deployment architecture that meets business needs and can be used as a basis for project approval and budgeting. This architecture is also the basis for an implementation design that provides the details needed to build, test, and roll out into a production environment.
- **Operations.** Run a deployed software solution, monitoring and optimizing its performance and upgrading it to include new functionality as necessary.

The tasks involved in each of these stages are shown in Figure 1-2 and discussed more fully in Chapter 4, "Life-Cycle Concepts."

Figure 1-2 shows the kind of Java Enterprise System user needed to perform the various Java Enterprise System tasks. If you are working with Java Enterprise System, your job should fit one or more of the user categories shown in Figure 1-2. The following table describes the skills and background needed to perform the corresponding Java Enterprise System tasks.

Table 1-2 User Categories for Java Enterprise System Tasks

User Profile	Skills and Background
Business planner System analyst	General, rather than in-depth technical knowledge Understands strategic direction of the business Knows business processes, objectives, and requirements
Architect	Highly technical Has broad knowledge of deployment architectures Familiar with latest technologies Understands business requirements and constraints
System integrator IT manager Field engineer System administrator	Highly technical Intimately familiar with IT environments Experienced in implementing distributed software solutions Knows network architecture, protocols, devices, security Knows scripting and programming languages
Specialized system administrator Delegated administrator	Specialized technical or product knowledge Familiar with hardware, platforms, directories, databases Skilled at monitoring, troubleshooting, upgrading software Knows UNIX system administration

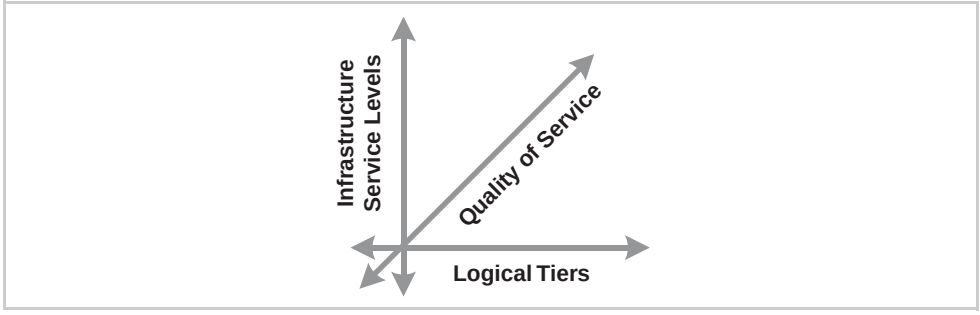
Chapter 2

Java Enterprise System Architecture

This chapter provides an overview of the architectural concepts upon which Java Enterprise System deployments are based.

The chapter describes a framework in which Java Enterprise System *deployment architectures* is analyzed along three dimensions: logical tiers, infrastructure service levels, and quality of service. These three dimensions, shown schematically as orthogonal axes in the following figure, help to clarify the architectural functions of Java Enterprise System components. The three-dimensional framework is key to designing successful deployment architectures for business software solutions.

Figure 2-1 Three Dimensions of Java Enterprise System Architectural Framework



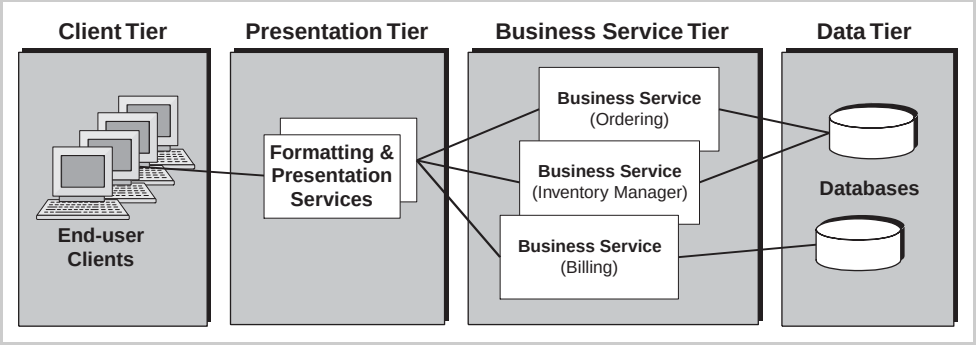
This chapter explores each of the three dimensions shown in Figure 2-1 independently, followed by a synthesis of the three dimensions into a single framework. The chapter contains the following sections:

Dimension 1: Logical Tiers

The standard *architecture* for distributed applications separates application logic into a number of tiers. These tiers signify a logical and physical organization of components into an ordered chain of service providers and consumers. Components within a tier typically consume the services provided by components in an adjacent provider tier and provide services to one or more components in an adjacent consumer tier.

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

Figure 2-2 Dimension 1: Logical Tiers for Distributed Enterprise Applications



Description of Logical Tiers

This section provides brief descriptions of the four logical tiers shown in Figure 2-2. The descriptions, for the sake of example, refer to 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 HTML or XML delivery 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.

Presentation tier components are often reusable components that can be customized and plugged into an application. You can also replicate presentation services for failover and scalability, and you can map these services to computing nodes in a way that optimizes network bandwidth and computing resources.

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 EJB components or message-driven beans (MDBs). 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* 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.

The various implementations of business services encapsulate specific application functions that can reside and run on a particular computing node. This approach allows for reusable components that you can customize and plug into an application. As with presentation tier logic, you can replicate these business service providers for failover and scalability, and you can map these service providers to computing nodes in a way that optimizes network bandwidth and computing resources.

Data Tier

The data tier consists of data used by business logic. The data can be persistent 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 can also include data feeds from external sources or data accessible from legacy computing systems.

Logical and Physical Independence

The services shown in the presentation and business service tiers of Figure 2-2 are the centerpiece of this model. These services are multi-threaded software processes capable of supporting large numbers of *clients*. These clients can be either end-user clients or other services.

The architectural dimension illustrated in Figure 2-2, emphasizes the logical and physical independence of the four tiers, facilitating the partitioning of application logic across the various computing nodes in a networked environment:

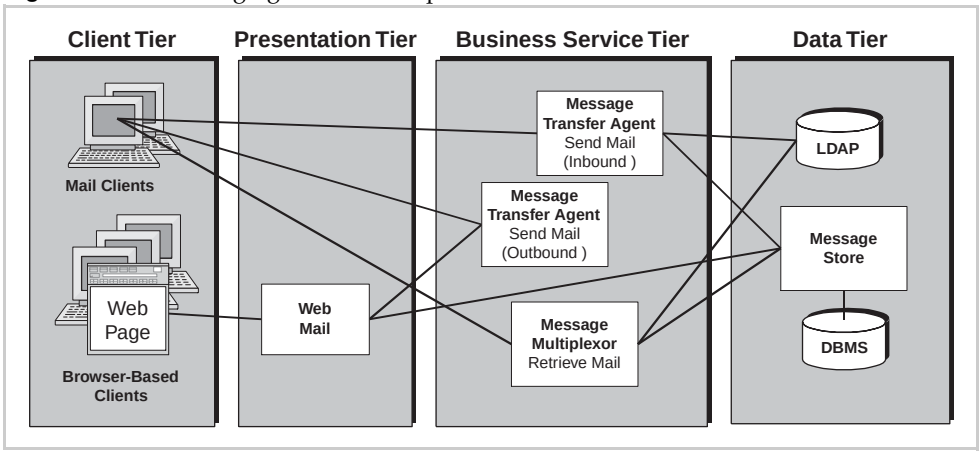
- **Logical independence.** The four tiers in the architectural model represent logical independence: you can modify application logic in one tier (for example, in the business service tier) independently of the logic in other tiers. You can change your implementation of business logic without having to change or upgrade logic in the presentation tier or client tier. This independence means, for example, that you can introduce new types of clients without having to modify the business logic.
- **Physical independence.** The four tiers also represent physical independence: you generally deploy the logic in different tiers on different hardware platforms (that is, different cpu configurations, chip sets, and operating systems). The independence allows you to run distributed application components on the computing nodes best suited to their individual computing requirements and best suited to maximizing network bandwidth.

  How you map application components to a hardware environment (that is, your deployment architecture) depends on many factors: the speed and power of the different computers, the speed and bandwidth of network links, security and firewall considerations, and the need to replicate components for failover (high availability) and load balancing (scalability). The mapping you choose also depends upon the sizing, performance, and overall cost requirements of a particular solution.

Example of Tiered Architecture

The e-mail communication services provided by Messaging Server are an example of the use of logical tiers in architectural design. E-mail services are implemented using a number of Messaging Server components, as shown in the following figure.

Figure 2-3 Messaging Server: Example of Tiered Architecture



The logical separation of Messaging Server functions into independent components allows these components to be distributed across different computing nodes in a physical environment. The physical separation allows for easy replication of these components, allows for different availability solutions for the different components, and enables different approaches to the security of different components.

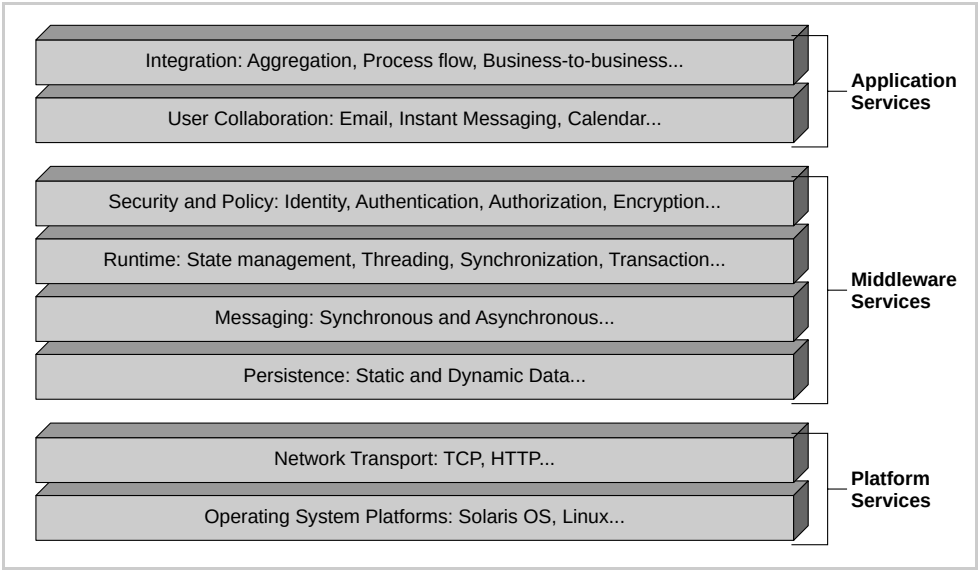
Dimension 2: Infrastructure Service Levels

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 set of distributed services constitutes an infrastructure upon which distributed components can be built.

Distributed Infrastructure Services

Distributed infrastructure services can be conceptualized as a set of distributed services at many different levels. These services constitute the infrastructure service levels dimension of deployment architecture as illustrated in the following figure.

Figure 2-4 Dimension 2: Distributed Infrastructure Service Levels



The levels in Figure 2-4 reflect a general dependence of the various distributed 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.

However, 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. Also, the levels represented in Figure 2-4 are not strictly prescribed. Other service levels, such as a monitoring or management service, might also be included in this conceptual illustration.

In general, the services shown in Figure 2-4 fall into three broad groupings: low-level platform services, high-level application services, and a group of middleware services, so named for their location between the other two groupings.

The following paragraphs briefly describe the different services, with reference, where relevant, to Java programming language artifacts. The services are described from lowest to highest, as shown in Figure 2-4:

- **Operating system platform.** Provides the basic support for any process running on a computing node. The operating system (such as Solaris™ Operating System, Linux, or Windows) manages physical devices as well as memory, threads, and other resources necessary to support the Java Virtual Machine (JVM™ machine).

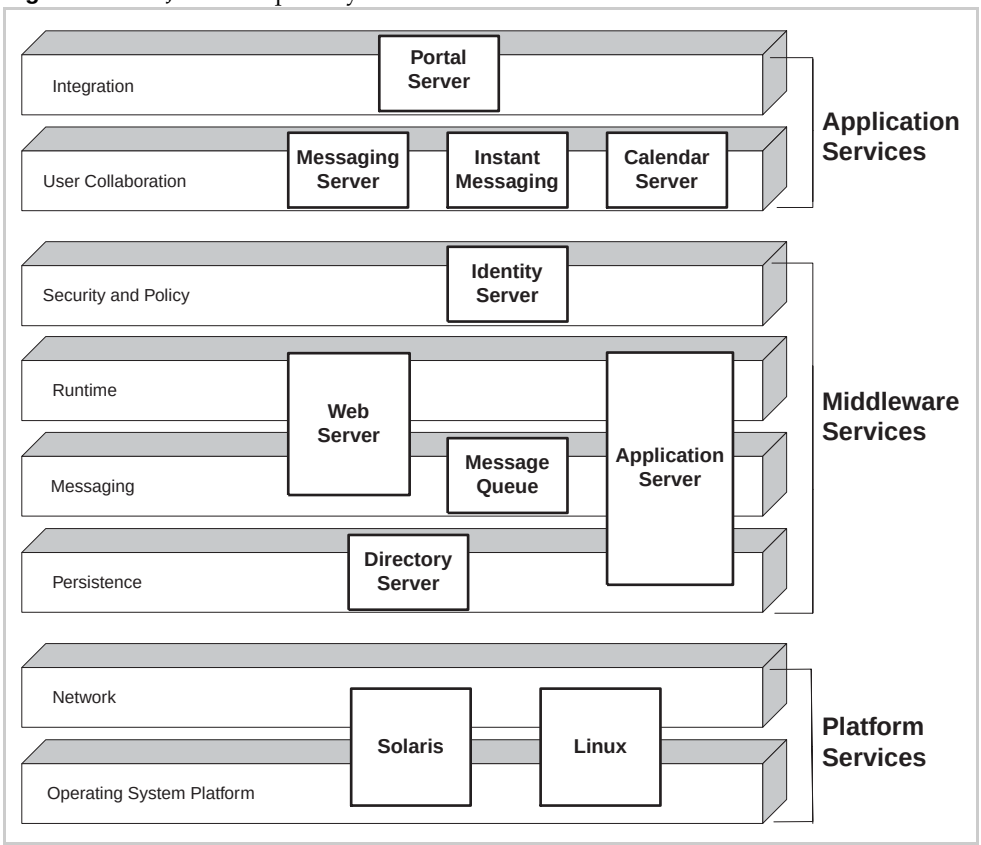
- **Network transport.** Provides basic networking support for communication between distributed application components running on different computing nodes. These services include support for protocols such as TCP and HTTP. Other higher-level communication protocols (see the Message layer) depend on 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 being updated).
- **Messaging.** Provides support for both synchronous and asynchronous communication between application components. Synchronous messaging is real-time sending and receipt of messages; it 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 (MDB) 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 standalone servers (such as an e-mail server or calendar server).

- **Integration.** Provides the services that aggregate existing business services, either by providing a common interface for accessing them, as in a portal, or by integrating them through a process engine that coordinates them within a production workflow. Integration can also take place as business-to-business interactions between different enterprises.

Java Enterprise System Implementation

Java Enterprise System implements the distributed infrastructure services dimension shown in Figure 2-4. The positioning of Java Enterprise System components within the different levels is shown in the following illustration:

Figure 2-5 Java Enterprise System: Distributed Infrastructure Services



The Java Enterprise System implementation of distributed infrastructure services shown in Figure 2-5 consists of discrete software servers (*system servers*) that provide services at various levels within the distributed infrastructure service stack. These service providers are multi-threaded server process capable of supporting a large number of clients.

NOTE A number of Java Enterprise System components are not shown in Figure 2-5 because they do not directly provide distributed infrastructure services. Rather, these components provide the following support functions:

- Portal Server Mobile Access provides access to Portal Server from wireless clients.
- Portal Server Secure Remote Access provides access to Portal Server from browser-based clients outside an enterprise firewall.
- Directory Proxy Server provides access to Directory Server from browser-based clients outside an enterprise firewall.
- Sun Cluster provides high availability to infrastructure services and is discussed under the quality of service dimension of architecture (see "Example: Sun Cluster" on page 38).

Java Enterprise System Servers

Java Enterprise System servers collectively implement all of the levels shown in Figure 2-5. Each system server provides a particular service or set of services in support of distributed enterprise applications. These *system services* are what uniquely characterize each server (see Table 1-1 on page 20 for a brief description of the services provided by each system server).

Dependencies Between System Servers

In general, each system server depends on servers below it in the infrastructure and supports servers above it. Table 2-1 shows the specific dependencies between the different Java Enterprise System servers, listed from top to bottom, as shown in Figure 2-5.

Table 2-1 Java Enterprise System Server Interdependencies

Java Enterprise System Component	Provides Support To	Depends On
Portal Server		Identity Server Application Server or Web Server Directory Server If configured to use Portal Server Channels: Calendar Server Messaging Server Instant Messaging
Messaging Server	Calendar Server (for e-mail notifications) Portal Server (for messaging channel)	Identity Server (for single sign-on) Web Server (Web interface) Directory Server
Instant Messaging	Portal Server (for instant messaging channel)	Identity Server (for single sign-on) Directory Server
Calendar Server	Portal Server (for calendar channel)	Messaging Server (for e-mail notification service) Identity Server (for single sign-on) Web Server (Web interface) Directory Server
Identity Server	Portal Server If configured for single sign-on: Calendar Server Instant Messaging Messaging Server	Application Server or Web Server Directory Server
Application Server	Portal Server Identity Server	Message Queue Directory Server (optional)
Message Queue	Application Server	Directory Server (optional)
Web Server	Portal Server Identity Server	Identity Server (optional: access control)
Directory Server	Portal Server Calendar Server Messaging Server Instant Messaging Identity Server	None

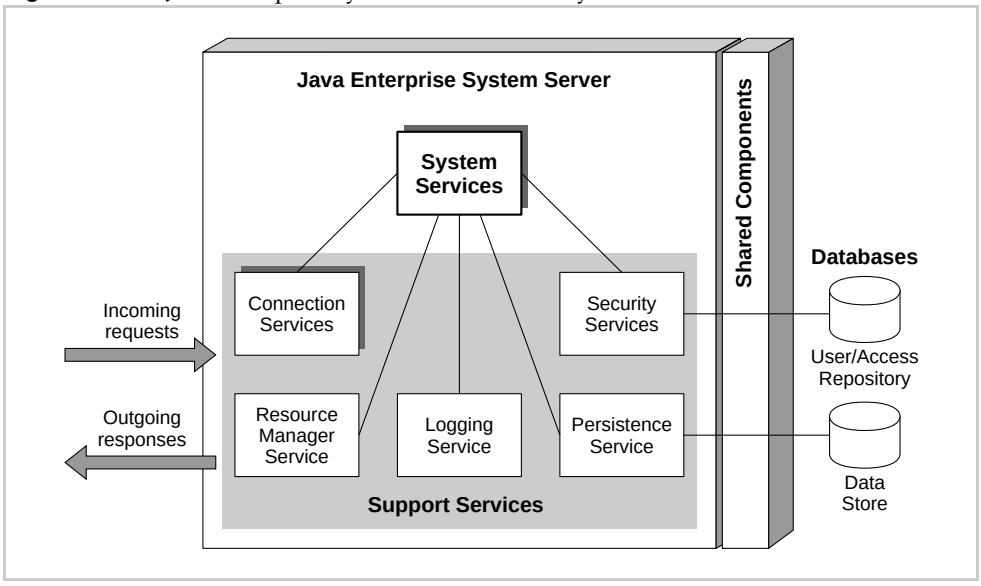
Anatomy of a System Server

Despite the different services each Java Enterprise System server provides, all system servers share some common characteristics. In general, each system server employs the following kinds of software components or subcomponents:

- System services subcomponents
- Support services subcomponents
- Shared components

These subcomponents are shown schematically in the following figure and are described briefly in the sections that follow.

Figure 2-6 Java Enterprise System Server Anatomy



System Services Subcomponents

Table 1-1 on page 20 summarizes the main system services provided by each of the Java Enterprise System servers.

Each server has its own way of implementing the services it provides. Some servers are written in the Java language, others in C or C++. Some use a single subcomponent to implement their unique system services, others use a number of subcomponents. For example, Portal Server uses Rewriter, Desktop, and NetMail subcomponents to provide the main system services of the Portal Server.

Support Services Subcomponents

Each system server contains a number of subcomponents that provide various *support services* upon which the system services depend. The support services shown in Figure 2-6 typically correspond to distributed infrastructure services provided by Java Enterprise System. For example:

- Connection services depend upon network transport services and might also require middleware messaging services.
- Security services typically depend upon identity and policy services, as well as persistence services.
- Resource manager services depend upon platform services.

In some cases, support services are provided externally by other Java Enterprise System servers. In many cases, however, the implementation of support services is internal to the server. The goal of the Java Enterprise System is to extract common internal services and implement them as system-level services, such as a logger service or communication services, and so forth.

Shared Components

In addition to support services, most Java Enterprise System servers also depend upon a number of local services, often used to provide portability across different operating systems. These services are libraries installed locally as *shared components* that all system servers running on a particular computing node can use. 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 76.

Dimension 3: Quality of Service

The previous two architectural dimensions (logical tiers and infrastructure service levels) largely define the 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.

As internet and E-commerce services have become more critical to business operations, the performance, availability, security, scalability, and serviceability of these services has become a key requirement of large-scale, high-performance deployment architectures.

In other words, meeting business requirements regarding a number of important service qualities has become an important dimension of deployment architecture. The qualities most often used to specify quality of service requirements are summarized in the table below.

Table 2-2 Dimension 3: Service Qualities Impacting Deployment Architecture

System 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, often expressed as the *uptime* of a system.
Security	A complex combination of factors that describe the integrity of a system and its users. Security includes authentication and authorization of users as well as the secure transport of information.
Scalability	The ability to add capacity (and users) 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 tasks such as monitoring the system, repairing problems that arise, and upgrading hardware and software components.

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

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

Applying Quality of Service Requirements

Quality of service requirements for the system qualities shown in Table 2-2 are normally stated on a system-wide level, that is, they apply to the system as a whole. However, the overall functioning of a software system is the result of complicated interactions among the various application and infrastructure components in the system.

For this reason, quality of service requirements generally apply to all tiers at all infrastructure service levels within an architecture. These requirements often apply on a component by component basis.

For example, if a system is supposed to be highly available, you need to consider the most likely points of failure in the system and focus first on those failures that would have the most impact. A high availability solution for such high-risk components might be more demanding than a high availability solution for components that are less frequently used or which do not cause overall system failure.

Similar issues come into play when considering performance, security, and scalability. A good deal of analysis is required to understand potential weak points or bottlenecks in a system and to incorporate architectural solutions that make sense for each component in a system.

Example: Sun Cluster

One Java Enterprise System component specifically addresses the quality of service architectural dimension: Sun Cluster.

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

A cluster is a set of loosely coupled computing nodes that collectively provides a single client view of services, system resources, and data. Internally, the cluster uses redundant computing nodes, interconnects, data storage, and network interfaces to provide high availability to cluster-based services and data. 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.

In addition, Cluster agents continuously monitor software services hosted by the cluster. In case of failure, these software agents act to fail over or restart the services they monitor. Cluster agents are available for all of the Java Enterprise System servers, and you can write custom Cluster agents for any distributed components or service implementations that run on top of the Java Enterprise System infrastructure. In this way, Cluster software provides for highly available services. (This availability is at the service level, and does not provide for session-level failover.)

Because of the control afforded by Cluster software, a cluster 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, Cluster software can therefore provide for both high availability and scalability in a distributed enterprise application.

Because of the redundancy necessary to support Cluster services, inclusion of these services in a solution has a large impact on your computing environment. Inclusion of Cluster services substantially increases the number of computing nodes and network links required in your physical topology.

Unlike the services provided by Java Enterprise System servers, Cluster services are distributed peer-to-peer services. Cluster software therefore needs to be installed on every computing node in a cluster.

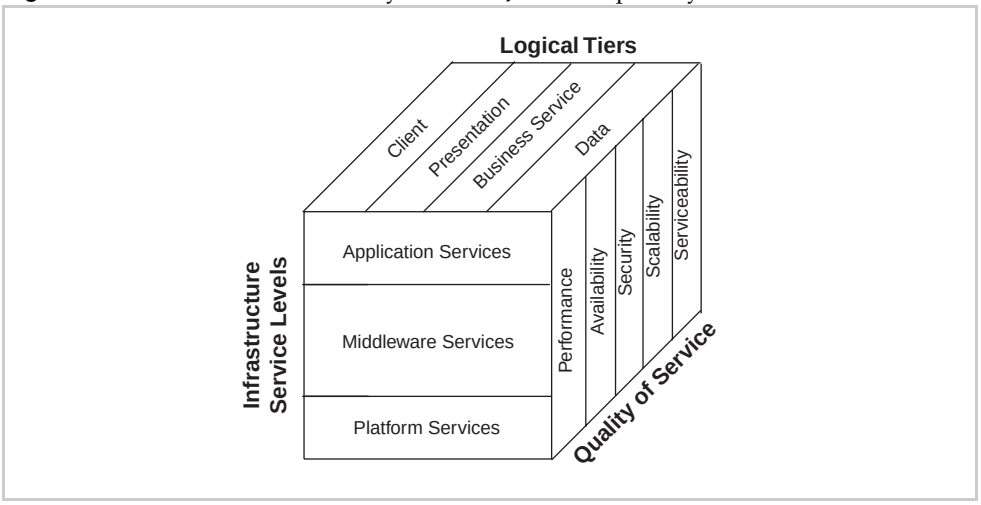
Three Dimensional Synthesis

The three architectural dimensions discussed separately in the preceding sections, when pulled together, provide a framework within which to understand the role of any application or infrastructure component within an architectural design.

Basically, distributed components in each logical tier of a deployment architecture (the first dimension) need to be supported by appropriate infrastructure services (the second dimension). Each component within that two dimensional matrix must be deployed so as to meet quality of service requirements (the third dimension).

The synthesis of these three dimensions is conceptually represented in the following figure.

Figure 2-7 Three Dimensional Synthesis of Java Enterprise System Architecture



Within this framework, for example, Directory Server would be categorized as a back-end, low-level Java Enterprise System component. Consequently, many other components depend on Directory Server, and therefore, its failure would have a tremendous impact on a business system. This signifies that Directory Server must be highly available.

Because Directory Server is used to store sensitive user or configuration information, a breach of security would also have a tremendous impact. This signifies that Directory Server and all communication channels that interact with it should be highly secure.

It is beyond the scope of this book to outline design methodologies for using the architectural framework in Figure 2-7. However, the three-dimensional architecture highlights aspects of Java Enterprise System that are important to understand in using Java Enterprise System to deliver distributed enterprise deployments.

Chapter 3

System-Level Features

This chapter provides conceptual and technical background for understanding the following two system-level features provided by Java Enterprise System:

These two features are key to integrating Java Enterprise System components into a single software system.

The Java Enterprise System Integrated Installer

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

The Java Enterprise System installer is an integrated framework that transfers Java Enterprise System software to a host system. It lets you select the Java Enterprise System components you need for any given computing node in your environment and installs those components on that computer. To set up your distributed environment, you use the Java Enterprise System installer to install the appropriate components on each node, one node at a time, in your environment.

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

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

Pre-Existing Software Checking

The installer performs checks at several levels to make sure that all previously installed components are at the appropriate release level to interoperate successfully.

The installer examines the computer on which you are installing and identifies the Java Enterprise System component products that are already installed. It informs you about those that are incompatible and must be upgraded or removed.

Similarly, the installer checks for Java Enterprise System shared components (see "Shared Components" on page 36), such as J2SE or NSS, that are already installed. If the installer finds shared components whose versions 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 cross checking of components to verify that the installation components you select will function properly.

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 will receive a warning but be able to proceed (under the assumption that the dependency will be satisfied by a component on a different host computer).

Initial Configuration

Many Java Enterprise System component require some degree of initial configuration before they can be started up. Depending on the component product, the Java Enterprise System installer can perform this initial configuration.

You can choose to have the installer perform this initial configuration (Configure Now) or to skip the initial configuration (Configure Later), in which case you will have to perform initial configuration manually after installation is complete.

If you choose Configure Now, configuration information will be required 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 component products that were installed on the local computer by the Java Enterprise System installer. The uninstaller checks dependencies, and issues warnings when it discovers a dependency. The uninstaller, like the installer, can be run in graphical, text-based, or silent mode.

Integrated Identity and Security Services

An important feature of the 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 User Identity

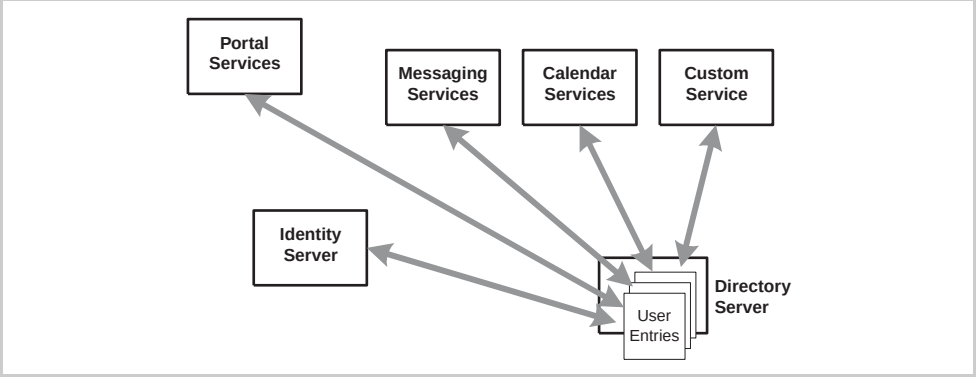
Within a Java Enterprise System environment, an end user has a *single identity*. Based on that identity, a user can be allowed access to various resources, such as a portal, web pages, and services such as messaging, calendar, and instant messaging.

This integrated identity and security capability is based on close collaboration between Directory Server, Identity Server, and other Java Enterprise System components.

User access to a Java Enterprise System 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, as well as an e-mail 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 Enterprise System component, that service uses Identity Server to authenticate the user and authorize access to specific resources. The requested service can then look up information in the user's directory entry needed to perform the work requested by the user, as shown in the following figure.

Figure 3-1 Single User Entry in Directory 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 Enterprise System service, and thereby be automatically authenticated to other system services. This capability, known as *single sign-on*, is a powerful feature provided by Java Enterprise System.

Directory Basics

A directory is 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. User accounts are entries in a directory.

LDAP directories store data in a hierarchical directory structure. The data is stored in entries that consist of a set of attributes and their respective values. A user entry, for example might have attributes such as `sn` (surname), `telephoneNumber`, and `userPassword`. Directory entries can describe people, organizations, hardware devices, software configurations, or other kinds of objects.

The set of attributes that describe an object, or some aspect of an object, is called an LDAP *object class*. For example, the attributes mentioned above that might be found in a user entry are defined by a `person` object class. The set of object classes and corresponding attributes that can be stored in a directory are collectively referred to as the directory *schema*.

In general, the designer of a business service decides what types of directory entries are needed by that service and what attributes values are needed for that entry. The designer also decides what kind of hierarchy characterizes the directory. The service must be aware of this hierarchical structure to look up entries in the directory.

In the case of a directory that stores end-user attributes, the directory schema might include object classes and attributes for enterprise, organization, and organizational unit. There might also be object classes to define groups within the organization to which individual users might belong.

A directory schema therefore includes all the object classes and attributes needed to represent and characterize directory data. The schema also specifies the data type and format of attribute values.

Fortunately, business service designers rarely need to create a directory schema from scratch because standard LDAP schemas already exist for many business needs. However, an application or service might need user attributes not covered by such standards, and in this case, a designer defines service-specific object classes that extend the standard schema. Nearly all the Java Enterprise System services, for example, extend the standard schema.

Directory Server Schema

Java Enterprise System directory services are provided by Directory Server, which implements an LDAP directory. By default, Directory Server provides a schema based on LDAP standards (LDAPv3), with additional Directory Server-specific extensions. The LDAPv3 schema defines a set of core object classes and attributes that reflect a human-oriented perspective: person, organization, group, and other such objects.

The default Directory Server schema also includes object classes and attributes for defining the schema itself and for Directory Server configuration.

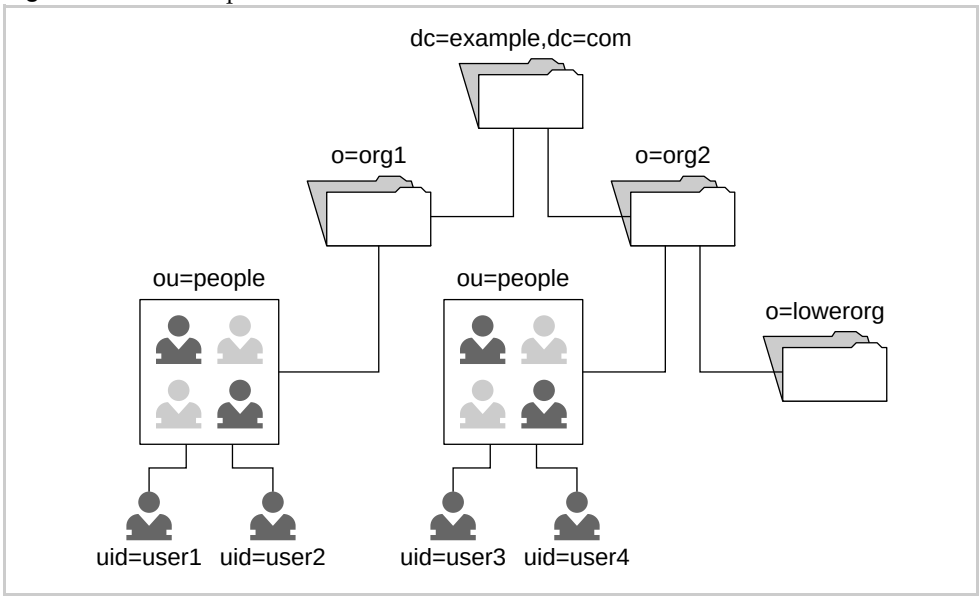
All Java Enterprise System services can use the object classes and hierarchical structure rules specified by the default schema. However, for various services to store user-specific information in the same entry for a given user, the default schema must be extended to include all attributes needed by those services.

Directory Information Tree

To use the identity, security, and other services provided by Java Enterprise System, you need to plan how to structure user information in your directory. You do this by designing a *directory information tree* (DIT). A DIT is a hierarchical structure that reflects the organization of your enterprise. The DIT affects how services find information in a directory and also has an impact on how you deploy and administer the directory.

The directory tree organizes user data, for example, by group, by people, by geographical location, and so forth. An example directory information tree is shown in the following figure.

Figure 3-2 Example DIT Structure



The root of the (inverted) directory tree, known as the root suffix, is shown at the top of Figure 3-2. A suffix is a branch or subtree whose entire contents are treated as a unit for the purpose of directory administration tasks. For example, an entire suffix can be initialized in a single operation. Indexing and directory replication (for load balancing or failover) is performed at the level of a suffix.

In most cases, the root suffix represents your main organizational or network domain, in this case `example.com`. Below the domain level in Figure 3-2 are two organization branches, each of which has an organizational unit (`people`), which contains user entries. Each branch point represents a directory entry and each is named according to one or more of its attributes. For example, `o=org1`, means that the `o` (organization) attribute has the value `org1` for that entry. The attribute names `dc` (domain component), `ou` (organizational unit), `uid` (user ID), and `cn` (common name) are standard attribute names defined in the directory schema.

An entry can have more than one value for a given attribute: `cn=Mattew Doe` and `cn=Matt Doe`, or `dc=example` and `dc=com`, for example. In addition to the branch points shown in Figure 3-2, it is common to have a branch `ou=group`*x* that lists members in a group *x*, which might be the basis for access to particular resources. For example, you might group users by management level.

To specify any entry, you provide the full path to the entry through the directory tree. For example, the full path for `user3` is `uid=user3, ou=people, o=org2, dc=example, dc=com`. This full path is called the entry's distinguished name (DN).

When designing a directory tree, you want to use a structure that closely represents the structure of your organization. This enables you to administer or replicate branches that are meaningful in your enterprise. You also want to choose names that are not likely to change over time.

Authentication and Authorization

Java Enterprise System authentication and authorization services are provided by Identity Server. Identity Server uses information in Directory Server to broker the interaction of users with Java Enterprise System web services or other web-based services in an enterprise.

Identity Server functions require extensions to the default schema provided by Directory Server. These extensions provide object classes needed to define policies, roles, and other aspects of Identity Server's authentication and authorization services.

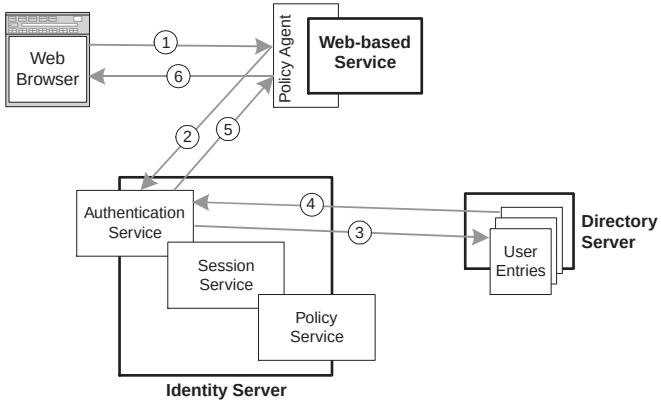
Identity Server also makes use of an external component known as a policy agent. The policy agent plugs into the web server hosting a service or resource being secured by Identity Server. The policy agent intercedes on behalf of the Identity Server in requests made by users to the secured resources. For some Java Enterprise System components, for example, Messaging Server and Calendar Server, the functionality of the policy agent is built into the component.

Authentication

Identity Server 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 3-3. A policy agent intercedes in the phone book log-on request, sending the request to the Authentication Service. The Authentication Service checks the user ID and password against information stored in Directory Server. If the log-in request is valid the user is authenticated and the company phone book is displayed to the employee. If the log-in request is not valid, an error is generated and authentication fails. (The Authentication Service also supports credentials-based authentication over HTTPS)

Figure 3-3 Authentication Scenario



Single Sign-On

The authentication scenario discussed above omits an important step. When a user's authentication request is verified, the Identity Server's Session Service is brought into play. The Session Service generates a session token, which holds the user's identity information and a token ID. The session token is sent back to the policy agent which forwards it (as a cookie) to the browser from which the authentication request was made.

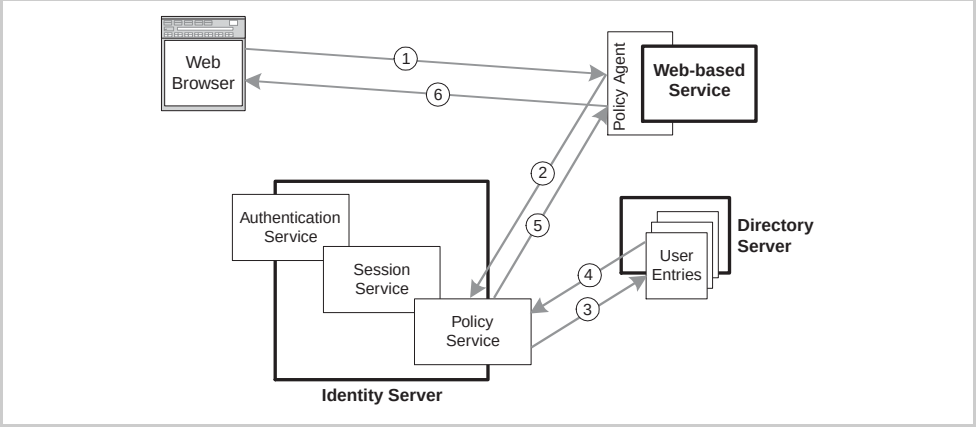
When the authenticated user attempts to access another secured service, the browser passes the session token to the corresponding policy agent. The policy agent verifies with the Session Service that the user's previous authentication is still valid, and the user is granted access the second service without being asked to re-enter a user ID and password. In the case of Java Enterprise System components that have policy agent functionality built-in, the single sign-on process is essentially the same.

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

Authorization

Identity Server also includes a Policy Service that provides access control to web-based resources in a Java Enterprise System. The authorization sequence is shown in the following figure.

Figure 3-4 Authorization Scenario



When an authenticated user makes a request for any Identity Server-secured resource, the policy agent notifies the Policy Service, which uses information in Directory Server to evaluate the access *policy* covering the resource to see if the user has permission to access the resource. A policy is a rule that describes who is authorized to access a specific resource under specific conditions.

Identity Server 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.

Identity Server 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.

Setup Issues

Setting up a Java Enterprise System environment to store user data needed to authenticate users, authorize access to specific resources, and provide various services to users involves two important procedures: extending directory schema and provisioning users.

Extending Directory Schema

To support Java Enterprise System services and custom services in your environment, object classes and attributes are required that are not present in Directory Server's default schema.

For example, Identity Server requires specialized object classes and attributes for performing policy services, and Messaging Server and Calendar Server both require schema extensions beyond those required for Identity Server.

The Java Enterprise System installer generally accommodates these extensions. For example, when you install Identity Server, the installer can run initial configuration scripts that import the required schema into the default Directory Server schema. When you install Portal Server and Instant Messaging, the installer automatically extends the Identity Server schema. When you install Messaging Server or Calendar Server, however, you have to manually run a script (`comm_dssetup`) to import the required additional schema into the Identity Server schema.

NOTE The versions of Messaging Server and Calendar Server that predate Java Enterprise System use a different schema and directory tree structure (called Sun ONE LDAP Schema 1) than the schema supported by Java Enterprise System (Schema 2). Therefore, if you want to preserve the data in your earlier directory, but want to use the authentication, authorization, and portal services provided by Java Enterprise System, you have to migrate the directory data to Schema 2. Java Enterprise System provides a utility for performing this migration. For more information, see the *Sun Java Systems Communications Services Schema Migration Guide*.

If you have custom-developed services that need to be supported by the Java Enterprise System infrastructure, and these services require additional schema extensions, you can extend the directory schema accordingly using Directory Server tools. For example, you can use the Directory Server Console, a graphical user interface, to add new schema, or you can define the new schema elements in an LDAP Interchange Format (LDIF) file, which can be imported into the directory schema using the Directory Server `ldapmodify` command.

User Provisioning

The procedure for enabling end users to access and use system services is called *user provisioning*. It involves creating an entry for each user in the directory. The user entry must contain all the user-specific attribute values required by the services that the user will want to access.

Historically, each component of Java Enterprise System was an independent product that used a directory more or less independently of other products. Therefore each product had its own provisioning tool for creating user entries in its own directory and for populating that entry with data needed by that product.

However, Java Enterprise System supports a single user entry in a single directory (a single user identity) for *all* Java Enterprise System services (as well as custom-developed services in the environment). This capability requires a provisioning tool that can be extended to support all object classes and attributes needed by the system.

Identity Server's provisioning tools, the graphical Identity Server Administration Console and the command line `amadmin` utility, are designed to support this requirement. The Identity Server includes a capability by which additional object classes and their attributes can be registered as a new "service." Identity Server uses the registration information when creating or modifying a directory entry.

This registration capability has been used to extend Identity Server's provisioning tools to support user provisioning for Portal Server and Instant Messaging services. However, Identity Server provisioning tools have not been extended to support user provisioning for Messaging Server and Calendar Server.

Instead, Java Enterprise System provides a separate, command line User Management Utility. This provisioning tool registers all attributes required for Messaging Server and Calendar Server, and thereby lets you provision users for all Java Enterprise System components, including Messaging Server and Calendar Server.

When adding new users on an occasional basis graphical provisioning tools are appropriate. However, in situations where large numbers of new users must be provisioned, such tools are cumbersome. To batch provision large numbers of users, you can provide the User Management Utility with an input file that contains all the required user entry data.

NOTE Because of special semantic relationships required between Identity Server attributes, if Identity Server is included in a Java Enterprise System deployment, you should not provision users by creating batch LDIF files and importing them directly into Directory Server using the ldapmodify command. Use the User Management Utility instead.

The following table summarizes Java Enterprise System provisioning tools.

Table 3-1 Java Enterprise System User Provisioning Tools

Tool	Description	Used For:
Identity Server Console	Graphical administration interface provided with Identity Server	Provisioning for Portal Server, Instant Messaging, and Identity Server, but not for Messaging Server or Calendar Server. Refer to: *Sun Java System Identity Server Administration Guide* (http://docs.sun.com/doc/817-5709)
Identity Server `amadmin` utility	Command line administration interface provided with Identity Server	Also supports provisioning for custom services. Refer to: *Sun Java System Identity Server Administration Guide* (http://docs/sun.com/doc/817-5709) and *Identity Server Developer's Guide* (http://docs.sun.com/doc/817-5710), Chapter 6 "Service Management," Service Definition
User Management Utility (`commadmin`)	Command line provisioning tool installed with Identity Server	Provisioning specifically for or Messaging Server and Calendar Server. Also supports provisioning for all other Java Enterprise System components, but you have to know the attributes and data types. Refer to: *Sun Java System Communications Services User Management Utility Administration Guide* (http://docs.sun.com/doc/817-5703)
Directory Server Console	Graphical administration interface provided with Directory Server	Generic Directory Server provisioning tool. Don't use for provisioning users for Java Enterprise System components. Refer to: *Sun Java System Directory Server Technical Overview*, Chapter 3 "A Quick Look at Directory Server Console," Managing Entries (http://docs.sun.com/doc/817-5217)
Directory Server `ldapmodify` command	Command line directory management tool provided with Directory Server	Generic Directory Server directory management tool. Don't use for provisioning users for Java Enterprise System components. Refer to: *Sun Java System Directory Server Technical Overview*, Chapter 4 "A Quick Look at Directory Server Command-Line Utilities," Adding, Changing, and Deleting Entries (http://docs.sun.com/doc/817-5217)

Table 3-1 Java Enterprise System User Provisioning Tools *(Continued)*

Tool	Description	Used For:
Delegated Administrator	Graphical provisioning tool provided with Messaging Server	Schema 1 provisioning for Messaging Server. Not compatible with Schema 2. Don't use for provisioning users for Java Enterprise System components. Refer to: *iPlanet Delegated Administrator for Messaging and Collaboration 1.2 Installation and Administration Guide* (http://docs.sun.com/doc/816-6011-10)

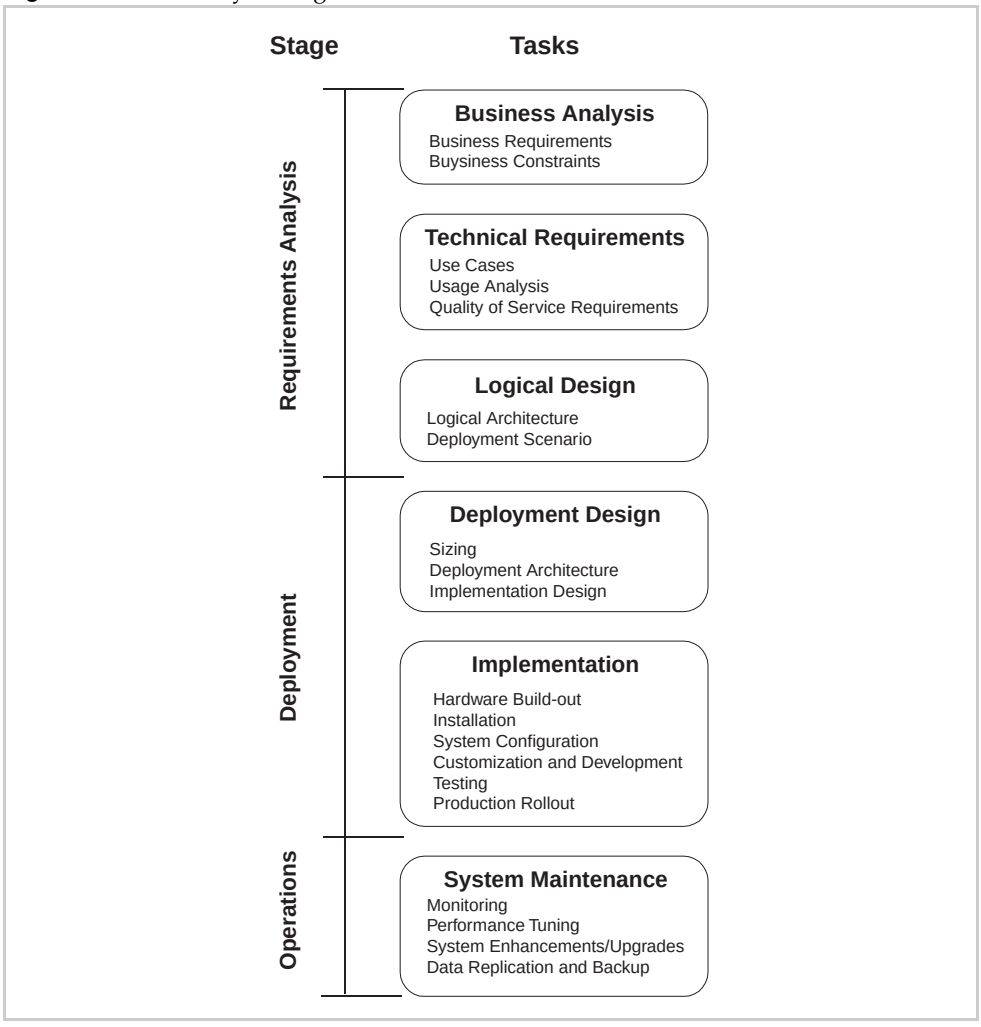
Chapter 4

Life-Cycle Concepts

Business solutions based on Java Enterprise System software involve a complex set of tasks that can be divided into the three life-cycle stages, shown in Figure 4-1.

This chapter describes the tasks involved in each of these stages, discussing concepts and terminology relevant to each:

Figure 4-1 Life-Cycle Stages



Requirements Analysis

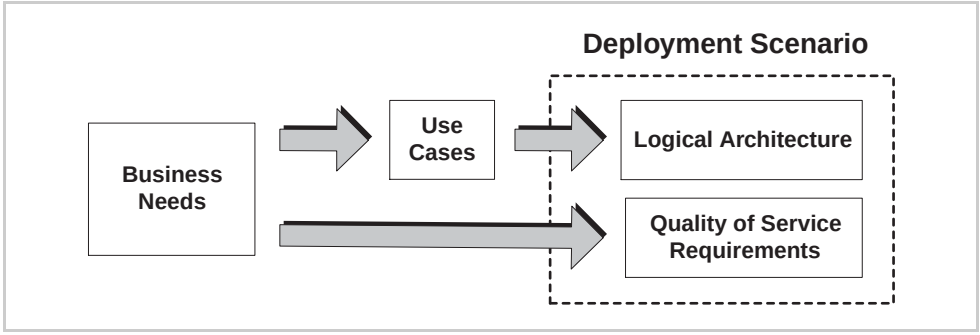
In the *requirements analysis* stage of the life cycle, you translate an analysis of business needs into a *deployment scenario*. The deployment scenario serves as a specification for a deployment design.

The requirements analysis stage can be divided into three phases, as shown in Figure 4-1:

- Business analyses. In this phase, you define the business goals of a proposed deployment 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 a proposed deployment. You also anticipate usage patterns for those use cases. In addition, you determine quality of service requirements (see Table 2-2 on page 37) for the proposed deployment, based on the business analysis.
- Logical Design. In this phase, you use the use cases to determine which Java Enterprise System infrastructure components and which custom-developed components are needed to provide end-user services. The use cases and other requirements are the basis for designing a *logical architecture*. The *logical architecture* depicts all the distributed components and infrastructure components required of a particular software solution and shows the interactions between those components.

The logical architecture, combined with performance, availability, security, and other quality of service requirements, is encapsulated in your *deployment scenario*, as shown in the following figure. For more information on the requirements analysis stage of the life cycle, see the *Java Enterprise System Deployment Planning White Paper*.

Figure 4-2 Requirements Analysis Results in a Deployment Scenario



Deployment

In the deployment stage of the life cycle, you translate a deployment scenario into a deployment design, which you then implement, prototype, and roll out in a production environment.

The deployment process depends not only on a solution's logical architecture, but also on performance, availability, security, scalability, serviceability, and other quality of service requirements. In other words, the quality of service dimension of deployment architecture plays a big role in the deployment stage.

The deployment process generally encompasses software components in all the tiers and in all the infrastructure service levels required to support an application. Thus, in addition to any specific distributed application components (J2EE components, web services, or other servers) you might deploy into a physical environment, you must also deploy the Java Enterprise System components (*system components*) needed to support the application.

In general, the deployment stage is a complex iterative process that involves a number of tasks. This section looks at the following two phases of the process:

- "Deployment Design"
- "Deployment Implementation" on page 60

Deployment Design

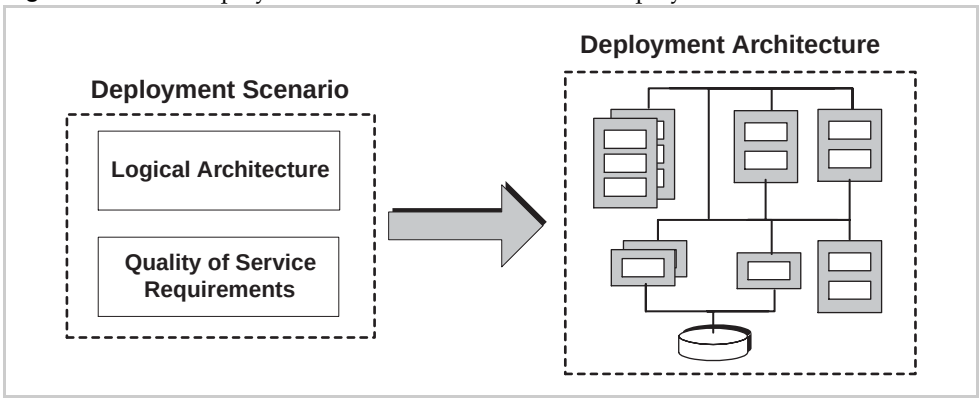
In the deployment design phase, you create a high-level *deployment architecture* followed by a low-level implementation design:

Deployment Architectures

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.

In other words, the deployment scenario is translated into a deployment architecture, as shown in the following figure.

Figure 4-3 A Deployment Scenario Translates into a Deployment Architecture



One aspect of this architectural design is sizing the physical environment to meet performance, availability, security, and other quality of service requirements. Once the sizing has been completed, you assign system servers and application components to the computing nodes in the physical environment. The resulting deployment architecture, must take into account the capabilities of different computing nodes, characteristics of system infrastructure services, and restrictions on the total cost of ownership or total cost of availability.

The larger the number of Java Enterprise System components in the deployment scenario, and the more demanding your quality of service requirements, the more demanding your design is on high-power computing nodes and high network bandwidth. Where hardware is limited, or prohibitively expensive, you might have to make 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.

Because designing a deployment architecture is an inexact science, architectures often evolve in an iterative fashion. You incrementally expand an existing system, noting bottlenecks, and adjusting the hardware or modifying the architecture to remove bottlenecks.

As a starting point for deployment design, the 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 a specific quality of service requirements. In the reference architecture, the application is deployed across a specific physical environment in a way that meets all the quality of service requirements specified in the deployment scenario. Performance testing is based on the same set of use cases from which the deployment scenario was developed.

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 Design.

The implementation design provides details needed to implement the deployment architecture. This includes specifying the actual hardware, operating systems, network structure, and other aspects of a physical environment. The detailed design specification might also include specifying directory information needed for provisioning end users for access to system services.

Deployment Implementation

Deployment implementation starts with an implementation design and consists of the following general tasks:

- "Hardware Build-Out"
- "Software Installation" on page 61
- "System Configuration" on page 61
- "Customization and Development" on page 61
- "Testing" on page 62
- "Production Rollout" on page 62

The ordering of these tasks is not rigid: the deployment process is iterative in nature. Nevertheless, the following subsections discuss each of the major deployment tasks independently, in the order they are normally performed.

Hardware Build-Out

The implementation design specifies the configuration of your physical environment: the computers, network design, network hardware (including cabling, switches, routers, and load balancers), storage devices, and so forth. All this needs to be assembled as the hardware platform that will support your Java Enterprise System-based solution.

Software Installation

The implementation design tells you which application components and which Java Enterprise System components are to reside on each computer node in your physical environment. You can use the Java Enterprise System integrated installer to install the different system components and shared components on each computer.

The installer and its features are described in "The Java Enterprise System Integrated Installer" on page 42.

System Configuration

There are a number of system configuration tasks that you have to complete for the various system components to work together as an integrated system. There are initial configuration steps needed for each individual system component to start up, and these might depend upon first starting up another system component upon which the first depends. For example, the Directory Server must be started before you can configure and start up the Identity Server, which depends upon LDAP schema extensions being written into the directory.

In any case each Java Enterprise System component must be configured to communicate with those components upon which it depends, and then internally configured for the feature set desired. High availability must also be configured, depending on your availability implementation for each component. Users need to be provisioned so they can access various services, and authentication and authorization controls need to be set up.

For information relating to user provisioning, authentication, single sign-on, and authorization, see "Integrated Identity and Security Services" on page 44.

Customization and Development

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

For some solutions, it might be sufficient to customize existing system servers, such as the Portal Server, to achieve the functionality required. In such cases, you normally do some minimal testing to make sure that your system was configured successfully, before going ahead with customization.

for other solutions, development might be quite extensive, requiring you to develop new business and presentation services from scratch using J2EE components that run in an Application Server or Web Server environment. In such cases it is wise to prototype your solution and perform proof-of-concept testing before embarking on a full development effort.

In the case of solutions requiring extensive development, Java Enterprise System does not provide tools for programming distributed components or business services. These tools are available in Sun Java Studio, which simplifies the programming and testing of applications supported by the Java Enterprise System infrastructure.

Testing

At some point, depending on the degree of customization or development work, you need to verify your deployment architecture. In other words, you need to test the solution against the use cases and verify that you can meet quality of service requirements.

In cases where you have relatively few custom-developed services (a mostly out of the box deployment), you might perform 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.

If this testing reveals shortcomings in your deployment architecture, you need to 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 up distributed applications and infrastructure services in a production environment, provisioning production system end users, setting up *single sign-on*, 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.

Operations

In the *operations* stage of the life cycle, you run a deployed application, monitoring and optimizing its performance and upgrading the application to include new functionality.

Java Enterprise System 2004Q2 does not provide a common monitoring and management infrastructure or administration tools for managing the system as a whole. Each system component has its own administration tools for configuring, tuning, or managing its operations. The goal is to provide system-wide administration for the Java Enterprise System in the future.

Appendix A

Reference Listing: Java Enterprise System Components

This appendix provides a reference listing of all Java Enterprise System components, broken down into the following categories:

- **Java Enterprise System Server Components.** The component products that provide distributed infrastructure software services needed to support distributed, enterprise applications.
- **Java Enterprise System Client Components.** Front ends to Java Enterprise System servers. Some are administration tools and some are used to support end-user access to system services.
- **Shared Components.** Local libraries that can be shared by all Java Enterprise System servers running on a particular host computer.

In this appendix, Java Enterprise System components are listed alphabetically within the three categories above.

Java Enterprise System Server Components

Java Enterprise System server components (Java Enterprise System component products) provide distributed infrastructure services needed to support distributed, enterprise applications.

For a roadmap to component product documentation, refer to the *Java Enterprise System Documentation Roadmap* (`http://docs.sun.com/doc/817-4715`).

Java Enterprise System includes the server components listed below:

- Sun Cluster 3.1 4/04 and Sun Cluster Sun ONE Agents
- Sun ONE Application Server 7 Update 3, Standard and Platform Editions
- Sun Java System Calendar Server 6 2004Q2
- Sun Java System Directory Server 5 2004Q2
- Sun Java System Directory Proxy Server 5 2004Q2
- Sun Java System Identity Server 2004Q2
- Sun Java System Instant Messaging 6 2004Q2
- Sun Java System Message Queue 3.5 Service Pack 1, Enterprise and Platform Editions
- Sun Java System Messaging Server 6 2004Q2
- Sun Java System Portal Server 6 2004Q2
- Sun Java System Portal Server Mobile Access 6 2004Q2
- Sun Java System Portal Server Secure Remote Access 6 2004Q2
- Sun ONE Web Server 6.1 Service Pack 2

Sun Cluster 3.1 4/04 and Sun Cluster Sun ONE Agents

Sun Cluster software is a component of the SunPlex™ system. The SunPlex system is an integrated hardware and Sun Cluster software solution that extends the Solaris operating system into a cluster operating system. A cluster, or plex, is a collection of loosely coupled computing nodes that provides a single client view of network services or applications, including databases, web services, and file services.

After setting up a cluster, you create highly available data services by installing and configuring the data service's Sun Cluster agent and application on the cluster. For example, to create a highly available Messaging Server data service, you install and configure the Sun Cluster agent for Messaging Server as well as the Messaging Server component product.

The Java Enterprise System installer provides the Sun Cluster Core and the Sun Cluster Agents as separately installable components. Additional Sun Cluster agents are available on separate CDs.

Sun ONE Application Server 7 Update 3

Sun ONE 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 Administration Client provides graphical clients and command-line administration clients that allow you to manage and configure Application Server installations and hosted applications. The Administration Client also assists with deploying applications.

Application Server is available in two editions:

- **Standard Edition (default).** Allows management of multiple application server instances from a central administration console. Includes the ability to partition web application traffic through a web server tier proxy. Supports configuration of multiple application server instances per administration domain. SNMP can be used to monitor the Standard Edition application server.
- **Platform Edition.** Limited to single application server instances, that is, single virtual machines for the Java platform Java virtual machine (JVM™). Multi-tier deployment topologies are supported, but the web server tier proxy does not perform load balancing. Administrative utilities are limited to local clients only.

The Java Enterprise System installer provides Application Server as a single installable component. The following Application Server subcomponents can be installed separately:

- Application Server Core (Standard Edition or Platform Edition)
- Application Server Administration Client
- PointBase Server 4.2

Sun Java System Calendar Server 6 2004Q2

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 Enterprise System installer provides Calendar Server as a single installable component.

Sun Java System Directory Server 5 2004Q2

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 Enterprise System installer provides Directory Server as a single installable component.

Sun Java System Directory Proxy Server 5 2004Q2

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 Enterprise System installer provides Directory Proxy Server as a single installable component.

Sun Java System Identity Server 2004Q2

Sun Java System Identity Server (Identity Server) 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 Enterprise System installer provides Identity Server as a single installable component. The following Identity Server 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 Enterprise System resources based on users' identities. This component includes the Identity Server SDK.
- **Identity Server SDK.** This software development kit (SDK) provides the tools and templates developers need to customize Identity Server to meet their company's needs.
- **Identity Server 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 Instant Messaging 6 2004Q2

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.

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

- Instant Messaging Server Core

- Instant Messaging Resources
- Identity Server Instant Messaging Service

Sun Java System Message Queue 3.5 Service Pack 1

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

In addition to being a JMS provider, Message Queue has features that exceed the minimum requirements of the JMS specification. With 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 Enterprise System installer provides Message Queue Enterprise Edition and Message Queue Platform Edition as separately installable components.

Sun Java System Messaging Server 6 2004Q2

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 Enterprise System installer provides Messaging Server as a single installable component.

Sun Java System Portal Server 6 2004Q2

Sun Java System Portal Server (Portal Server) is an identity-enabled portal server solution. Portal Server provides all the user, policy, and identity management to enforce security, web application single sign-on, and access capabilities to user communities. In addition, Portal Server combines key portal services, such as personalization, aggregation, security, integration, and search. Unique capabilities that enable secure remote access to internal resources and applications round out a complete portal platform for deploying robust business-to-employee, business-to-business, and business-to-consumer portals.

The Java Enterprise System installer provides Portal Server as a single installable component. Sun Java System Portal Server Mobile Access is a subcomponent of Portal Server.

Sun Java System Portal Server Mobile Access 6 2004Q2

Sun Java System Portal Server Mobile Access (Portal Server Mobile Access) software extends the services and capabilities of the Portal Server platform to mobile devices, such as mobile phones and personal digital assistants. Portal Server Mobile Access software enables portal site users to obtain the same content that they access using web browsers. Portal Server Mobile Access uses the Identity Server Administration Console.

The Java Enterprise System installer provides Portal Server Mobile Access as a subcomponent of Portal Server.

Sun Java System Portal Server Secure Remote Access 6 2004Q2

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. Portal Server Secure Remote Access is a cost-effective, secure access solution that is accessible to users from any web browser, eliminating the need for client software. Integration with Portal Server ensures that users receive secure encrypted access to the content and services that they have permission to access.

Portal Server Secure Remote Access provides the following services:

- **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.
- **NetFile.** Is a file manager application that allows remote access and operation of file systems and directories.
- **Netlet.** 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.
- **Proxylet.** Enables users to access intranet web pages through Gateway.
- **Rewriter.** Provides secure access to corporate intranet web pages from outside of the intranet by transforming web links and creating rule sets for handling intranet web pages.

The Java Enterprise System installer provides Portal Server Secure Remote Access as a single installable component. The following Portal Server Portal Server Secure Remote Access components can be installed separately:

- Portal Server Secure Remote Access Core
- Gateway
- NetFile
- Netlet
- Proxylet
- Rewriter

Sun ONE Web Server 6.1 Service Pack 2

Sun ONE 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 Enterprise System installer provides Web Server as a single installable component.

Java Enterprise System Client Components

Java Enterprise System client components are front ends to Java Enterprise System servers. Some are administration tools and some are used to support end-user access to system services.

Java Enterprise System includes the client components listed below:

- Sun Java System Administration Server (and Console) 5 2004Q2
- Sun Java System Communications Express 6 2004Q2
- Sun Java System Communications Services User Management Utility 6 2004Q2
- Sun Java System Connector for Microsoft Outlook 6
- Sun Remote Services Net Connect 3.1

Sun Java System Administration Server (and Console) 5 2004Q2

Sun Java System Administration Server and Server Console together provide a graphical tool that lets you manage Directory Server and other server software in your enterprise. 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 Enterprise System software in your enterprise.

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

Sun Java System Communications Express 6 2004Q2

Sun Java System Communications Express 6 2004Q2 (Communications Express) provides an integrated web-based communication and collaboration client that consists of three client modules: Calendar, Address Book, and Mail. Configurable to provide either mail or calendar service, or both, Communications Express works with either Sun Java System LDAP Schema, Version 1 (Schema 1) or Schema 2.

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

Sun Java System Communications Services User Management Utility 6 2004Q2

The Sun Java System Communications Services User Management Utility is a command-line utility (`commadmin`) for provisioning users, groups, domains, and resources for Calendar Server, Messaging Server and other Java Enterprise System service providers.

User Management Utility is automatically installed when you choose to install Identity Server.

Sun Java System Connector for Microsoft Outlook 6

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 Remote Services Net Connect 3.1

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 Enterprise System 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 Enterprise System component products depend. The Java Enterprise System installer automatically installs any shared components required to support Java Enterprise System server components installed on a host computer.

Java Enterprise System includes the shared components listed below:

- Ant (Jakarta ANT Java/XML-based build tool)
- Apache Common Logging
- Apache SOAP (Simple Object Access Protocol)
- ICU (International Components for Unicode)
- IMAPI (Sun Java System Instant Messaging and Presence APIs)
- J2SE™ platform 1.4.2_04 (Java 2 Platform, Standard Edition)
- JAF (JavaBeans™ Activation Framework)
- JATO (Java Application Framework)
- JavaHelp™ Runtime
- JASB (Java Architecture for XML Binding)
- JAXM Client Runtime (Java API for XML Messaging)
- JAXP (Java API for XML Processing)
- JAXR (Java API for XML Registries)
- JAX-RPC (Java APIs for XML-based Remote Procedure Call)
- JCAPI (Java Card API)
- JDMK (Java Dynamic Management Kit)
- JSS (Java Security Services)
- KT search engine
- LDAP C Language 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
- XML C Library (libxml)

Java Enterprise System Key Terms

This glossary defines and explains key terms introduced in this *Java Enterprise System Technical Overview*. Refer to the *Java Enterprise System Glossary* (`http://docs.sun.com/doc/816-6873`) for a complete list of terms that are used in the Java Enterprise System documentation set.

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 A *distributed 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.

component product A previously-independent Sun ONE product that has been integrated into the *Java Enterprise System*, but which can be individually licensed.

computing node One of a number of computers in a network or Internet environment. Distributed applications are deployed across this environment, with different distributed components, *business service*, and *servers* running on the various computing nodes.

deployment A stage of the application 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.

deployment architecture A high-level design that depicts the mapping of a *logical architecture* to a physical computing environment. The physical environment includes the *computing nodes* in an intranet or Internet environment, the network links between them, and any other physical devices needed to support the software.

deployment scenario A *computing node* and the quality of service requirements that a 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 phase in the deployment process, in which a logical architecture is implemented (either through programming or customization) and tested.

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.

directory information tree a hierarchy that reflects the organizational structure of your enterprise or other information base.

distributed component A unit of software logic from which distributed applications are built. A distributed component usually conforms to a distributed component model (CORBA, J2EE) and performs some specific computing function. Distributed components—singly or combined—provide *business services*, and can be encapsulated as *web services*.

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.

instance (server instance) A distinct execution of a *server* process on a *computing node*. In general, multiple instances of a server can run on a single node or on multiple nodes, and each instance can be configured independently.

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 needed to support them.

object class The set of attributes that describe an object, or some aspect of an object, stored in an LDAP directory.

operations A stage of the application life-cycle process in which distributed applications are started up, monitored, tuned to optimize performance, and dynamically upgraded to include new functionality.

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.

reference deployment architecture A specific *deployment scenario* mapped to and deployed across a specific hardware topology and tested for performance. Reference deployment architectures are used as starting points for designing custom solution deployment architectures.

requirements analysis A stage of the application life-cycle process in which business needs are translated into a *deployment scenario*: a *logical architecture* and a set of quality of service requirements the solution must meet.

schema The set of *object classes* and corresponding attributes that can be stored in a *directory*. Also specifies attribute data types and formats.

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.

service A software function performed for one or more *clients*. This function might be a very low-level *support service*, such as a memory management, or a high-level *business service*, such as a credit check. Services can be local (available to local clients) or distributed (available to remote clients). A system-level service can consist of a family of individual services.

shared component A Java Enterprise System component (*system component*), usually a library, that provides local services to other system components. By contrast, a *system server* provides distributed services to other system components (or to application-level components).

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 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.

support service One or more *services* needed to support a *system service*. These include communication services, persistence services, security services, memory management services, logging services, and so forth. These might be provided by internal server components or, externally, by *system servers*.

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 servers* which provide distributed *services*, Cluster software, which provides availability and scalability services, and *shared components* that provide local services to other system components.

system server A *component product*—a *server*—included in the Java Enterprise System, and which provides one or more *system services* within the distributed service infrastructure.

system service One or more distributed *services* that define the unique functionality provided by a *system server*. System services normally require the support of a number of internal *support services* and/or a number of *shared components*.

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.

user provisioning A procedure for enabling end users to access and use system services. Provisioning involves creating for each end user an account in a directory service and populating the account with the user-specific information needed by each service.

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.

Index

D

E

G

I

J

L

M

N

O

P

Q

R

S

T

U

W